

## CAPÍTULO

# 27

## GESTIÓN DEL CAMBIO

### CONCEPTOS CLAVE

auditorías ..... 813

cambio ..... 797

control de la versión .. 822

control del cambio ... 810

depósito ..... 803

ECS ..... 801

estándares .... 824

GC WebApp ... 814

gestión del contenido .. 817

identificación .. 807

informe de estado .... 814

líneas base .... 800

objetos de configuración .. 807

proceso de GCS .806

SVC ..... 809

### UN VISTAZO RÁPIDO

**¿Qué es?** Cuando se construye software de computadora los cambios ocurren. Y puesto que ocurren, es necesario gestionarlos con eficacia. La gestión del cambio, también llamada gestión de la configuración del software (GCS), es un conjunto de actividades diseñadas para gestionar el cambio al identificar los productos de trabajo que probablemente cambien, establecer relaciones entre ellos, definir mecanismos para gestionar diferentes versiones

de estos productos de trabajo, controlar los cambios impuestos y auditar e informar los cambios realizados.

**¿Quién lo hace?** Todos los involucrados en el proceso de software están involucrados con la gestión del cambio en alguna medida, pero en ocasiones se crean posiciones de soporte especializado para gestionar el proceso de GCS.

**¿Por qué es importante?** Si no se controla el cambio, él toma el control. Y eso nunca es bueno. Es muy fácil que una corriente de cam-

**E**l cambio es inevitable cuando se construye software de computadora. Y el cambio aumenta el grado de confusión entre los ingenieros de software que trabajan en un proyecto. La confusión surge cuando los cambios no se analizan antes de realizarlos, no se registran antes de implementarlos, no se reportan a quienes deben saberlo o no se controlan en una forma que mejorará la calidad y reducirá el error. Babich [BAB86] aborda esto cuando afirma:

El arte de coordinar el desarrollo de software para minimizar... la confusión se llama gestión de la configuración. La gestión de la configuración es el arte de identificar, organizar y controlar modificaciones al software que se construye por medio de un equipo de programación. La meta es maximizar la productividad al minimizar las equivocaciones.

La gestión del cambio, más usualmente llamada *gestión de la configuración del software* (GCS o GC), es una actividad protectora (sombrilla) que se aplica a lo largo del proceso de software. Puesto que el cambio puede ocurrir en cualquier momento, las actividades de GCS se desarrollan para 1) identificar el cambio, 2) controlar el cambio, 3) garantizar que el cambio se implementará de manera adecuada, y 4) reportar los cambios a otros que pudieran estar interesados.

Es importante distinguir con claridad entre soporte de software y gestión de la configuración del software. El soporte es un conjunto de actividades de ingeniería del software que ocurren después de que éste se ha entregado al cliente y fue puesto en operación. La gestión de la configuración del software es un conjunto de actividades de seguimiento y control que se inician cuando comienza un proyecto de ingeniería del software y terminan sólo cuando éste se retira de operación.

bios incontrolados convierta en caótico un proyecto de software bien implementado. Por esta razón, la gestión del cambio es una parte esencial de la buena gestión del proyecto y de una sólida práctica de ingeniería de software.

**¿Cuáles son los pasos?** Puesto que muchos productos de trabajo se producen cuando se construye el software, cada uno debe identificarse en forma individual. Una vez hecho esto se establecen los mecanismos de control de versión y cambio. El proceso se audita para garantizar que la calidad se conserva conforme el cambio se realiza y que quienes tienen necesidad de conocerlo reciben información acerca de los cambios mediante los informes respectivos.

**¿Cuál es el producto obtenido?** Un plan de gestión de la configuración del software define la estrategia del proyecto para la gestión del cambio. Además, cuando se pide una GCS formal, el proceso de control del cambio produce solicitudes de cambio de software, informes y peticiones de cambio de ingeniería.

**¿Cómo puedo estar seguro de que lo he hecho correctamente?** Cuando cualquier producto de trabajo puede explicarse, seguirse y controlarse; cuando los cambios pueden seguirse y analizarse; cuando todos los que necesitan saber acerca de un cambio han sido informados, el trabajo se ha hecho bien.

Una meta primordial de la ingeniería del software es mejorar la facilidad con la que los cambios se pueden acomodar y reducir el esfuerzo cuando los cambios se deben realizar. En este capítulo se estudian las acciones específicas que permiten gestionar el cambio.

## 27.1 GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE

La salida del proceso de software es información que se puede dividir en tres amplias categorías: 1) programas de computadora (tanto al nivel de fuente como de formas ejecutables); 2) productos de trabajo que describen los programas de computadora (dirigidos tanto a profesionales técnicos como a usuarios), y 3) datos (internos o externos al programa). Los elementos que comprenden la información producida como parte del proceso de software se denominan colectivamente *configuración del software*.

Si cada elemento de configuración simplemente condujera a otros elementos habría poca confusión. Por desgracia, otra variable entra en el proceso: el *cambio*. Éste puede ocurrir en cualquier momento, por cualquier razón. De hecho, la primera ley de la ingeniería de sistemas [BER80] afirma: "No importa dónde se encuentre en el ciclo de vida del sistema, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida".

"No hay nada permanente, excepto el cambio."

Heráclito, 500 a.C.

¿Cuál es el origen de estos cambios? La respuesta es tan variada como los cambios mismos. Sin embargo, existen cuatro fuentes fundamentales de cambio:

¿Cuál es el origen de los cambios que se requieren para el software?

- Nuevas condiciones en el negocio o mercado dictan los cambios en los requisitos del producto o las reglas del negocio.
- Nuevas necesidades del cliente demandan la modificación de los datos que producen los sistemas de información, de la funcionalidad que entregan los productos o los servicios que entrega un sistema basado en computadora.
- La reorganización o el crecimiento o reducción del negocio provocan cambios en las prioridades del proyecto o en la estructura del equipo de ingeniería del software.
- Restricciones presupuestales o de calendarización inducen una redefinición del sistema o producto.

La gestión de la configuración del software es un conjunto de actividades que se han desarrollado para gestionar el cambio a lo largo del ciclo de vida del software de computadora. La GCS se considera como una actividad de aseguramiento de la calidad del software que se aplica a lo largo del proceso respectivo. En las secciones siguientes se examinan las principales tareas de la GCS y conceptos importantes que ayudan a gestionar el cambio.

### 27.1.1 Un escenario de GCS<sup>1</sup>

Un típico escenario operativo de GCS involucra un gestor de proyecto a cargo de un grupo de software; un gestor de configuración a cargo de los procedimientos y políticas de GC; los ingenieros de software responsables del desarrollo y mantenimiento del producto de software, y el cliente que emplea el producto. En el escenario, supóngase que el producto es pequeño e involucra cerca de 15 000 líneas de código que desarrollará un equipo de seis personas. ( Nótese que son posibles otros escenarios de equipos menores o mayores, pero, en esencia, existen conflictos genéricos que cada uno de estos proyectos enfrenta en relación con la GC.)

En el ámbito operativo el escenario involucra diversos papeles y tareas. La meta del gestor del proyecto es garantizar que el producto se entregue dentro de cierto periodo. En consecuencia, el gestor supervisa el progreso del desarrollo y reconoce y reacciona ante los problemas. Esto se hace al generar y analizar los informes acerca del estado del sistema de software y al realizar revisiones en el sistema.

Las metas del gestor de configuración son garantizar que se siguen los procedimientos y políticas para crear, cambiar y poner a prueba el código, así como posibilitar el acceso a la información acerca del proyecto. La implementación de técnicas para mantener el control sobre los cambios de código requiere que este gestor introduzca mecanismos para solicitar oficialmente cambios, evaluarlos (mediante una junta de control de cambios, que es la responsable de aprobar los cambios al siste-

¿Cuáles son las metas y las actividades realizadas por cada uno de los participantes involucrados en la gestión del cambio?

<sup>1</sup> Esta sección procede de [DAR01]. El permiso especial para reproducir "Spectrum of Functionality in CM Systems" de Susan Dart [DAR01], © 2001 por Carnegie Mellon University, lo otorgó el Software Engineering Institute.


**CLAVE**

Debe haber un mecanismo para garantizar que los cambios simultáneos al mismo componente son seguidos, gestionados y ejecutados adecuadamente.

ma de software) y autorizarlos. El gestor crea y distribuye las listas de tareas para los ingenieros y básicamente crea el contexto del proyecto. Además, el gestor recopila estadísticas acerca de componentes en el sistema de software, por ejemplo: la información que determina cuáles componentes son problemáticos en el sistema.

La meta de los ingenieros de software es trabajar con eficiencia. Esto significa que no interfieren de manera innecesaria unos con otros en la creación y prueba del código ni en la producción de los documentos de soporte. No obstante, al mismo tiempo, intentan comunicarse y coordinarse de manera eficiente. Específicamente, los ingenieros utilizan herramientas que ayudan a construir un producto de software consistente. Ellos se comunican y coordinan al notificarse mutuamente las tareas que se requieren y las tareas cumplidas. Los cambios se propagan por medio del trabajo de cada uno mediante archivos fusionados. Existen mecanismos para asegurar que, respecto de los componentes que experimentan cambios simultáneos, existe alguna forma de resolver los conflictos y fusionar los cambios. La historia de la evolución de todos los componentes del sistema se mantiene junto con un registro de las razones de los cambios y otro de lo que cambió en realidad. Los ingenieros tienen su propio espacio de trabajo para crear, cambiar, probar e integrar código. En cierto punto, el código se convierte en una línea base a partir de la que continúa el desarrollo posterior y desde la que se realizan las variantes para otras máquinas que también sean el objetivo.

El cliente emplea el producto. Dado que el producto lo controla la GC, el cliente sigue procedimientos formales para solicitar cambios e indicar los *bugs* en el producto.

Idealmente, un sistema de GC utilizado en este escenario apoyaría todas estas funciones y tareas; esto es, las funciones determinan la funcionalidad requerida de un sistema de GC. El gestor del proyecto ve una GC como un mecanismo de auditoría; el gestor de configuración, como un mecanismo de creación de control, seguimiento y políticas; el ingeniero de software, como un mecanismo de control del cambio, la construcción y el acceso; y el usuario, como un mecanismo de garantía de la calidad.

### 27.1.2 Elementos de un sistema de gestión de la configuración

En su detallado artículo acerca de la gestión de la configuración del software, Susan Dart [DAR01] identifica cuatro importantes elementos que deben estar presentes cuando se desarrolla un sistema de gestión de la configuración:

- *Elementos de componentes*: conjunto de herramientas acopladas dentro de un sistema de gestión de archivos (por ejemplo, una base de datos) que permiten el acceso y la gestión de cada elemento de configuración del software.
- *Elementos de proceso*: serie de procedimientos y tareas que definen un enfoque eficaz con el cual gestionar el cambio (y actividades relacionadas) para todos los participantes en la gestión, ingeniería y utilización del software de computadora.

- *Elementos de construcción:* conjunto de herramientas que automatizan la construcción del software al asegurar que se ha ensamblado un conjunto adecuado de componentes validados (es decir: la versión correcta).
- *Elementos humanos:* la implementación de una GCS eficaz requiere que el equipo de software utilice un conjunto de herramientas y características de procesos (que abarcan otros elementos de GC).

Estos elementos (que se estudiarán con más detalle en secciones venideras) no son mutuamente excluyentes. Por ejemplo, los elementos de componentes trabajan en conjunto con los de construcción conforme avanza el proceso de software. Los elementos de proceso guían muchas actividades humanas que se relacionan con GCS y, por tanto, también pueden considerarse elementos humanos.



*La mayoría de los cambios de software están justificados, así que no hay razón para quejarse acerca de ellos. Más bien, es necesario asegurarse de que se tienen los mecanismos apropiados para manejarlos.*

### 27.1.3 Líneas base

El cambio es un hecho de vida en el desarrollo del software. Los clientes quieren modificar los requisitos. Los desarrolladores quieren modificar el enfoque técnico. Los gestores quieren modificar la estrategia del proyecto. ¿Por qué todas estas modificaciones? La respuesta, en realidad, es bastante simple. Conforme pasa el tiempo, todos los participantes saben más (acerca de lo que necesitan, qué enfoque sería el mejor, cómo hacerlo y aun así obtener dinero). Este conocimiento adicional es la fuerza impulsora detrás de la mayoría de los cambios y conduce a una expresión difícil de aceptar para muchos profesionales de la ingeniería del software: *¡la mayoría de los cambios están justificados!*

Una *línea base* es un concepto de gestión de la configuración del software que ayuda a controlar el cambio sin impedir seriamente el cambio justificable. El IEEE (IEEE Std. No. 610.12-1990) define una línea base como:

Una especificación o producto que se ha revisado formalmente y se está de acuerdo con los resultados, y que a partir de ahí sirve como la base para el desarrollo ulterior y que puede cambiarse sólo por medio de procedimientos formales de control del cambio.

Antes de que un elemento de configuración del software se convierta en línea base, es posible realizar el cambio rápida e informalmente. Sin embargo, una vez establecida una línea base, metafóricamente se pasa a través de una puerta giratoria de una sola dirección. Los cambios se pueden realizar, pero se debe aplicar un procedimiento específico formal para evaluar y verificar cada uno.

En el contexto de la ingeniería del software, una línea base es un hito en el desarrollo del software. Se marca una línea base para la entrega de uno o más elementos de configuración del software (ECS) que se han aprobado como consecuencia de una revisión técnica formal (capítulo 26). Por ejemplo, los elementos de un modelo de diseño se han documentado y revisado. Se han encontrado errores y se han corregido. Una vez que todas las partes del modelo se han revisado, corregido y luego aprobado, el modelo de diseño se convierte en línea base. Los cambios posteriores a la arquitectura del programa (documentados en el modelo de diseño) sólo se pue-



*Un producto de trabajo de ingeniería del software se convierte en línea base sólo después de que se ha revisado y aprobado.*



*Es preciso asegurarse de que la base de datos del proyecto se mantiene en una ubicación central controlada.*

den efectuar después de que cada uno se ha evaluado y aprobado. Aunque las líneas base se pueden definir en cualquier grado de detalle, en la figura 27.1 se muestran las líneas base de software más comunes.

En la figura 27.1 también se muestra la progresión de eventos que conducen a una línea base. Las tareas de ingeniería del software producen uno o más ECS. Después de que éstos se revisan y aprueban se colocan en una *base de datos del proyecto* (también llamada *librería del proyecto* o *depósito de software*, que se examinan en la sección 27.2). Cuando un miembro de un equipo de software quiere modificar un ECS que se ha convertido en línea base, se copia de la base de datos del proyecto en el espacio de trabajo privado del ingeniero. Sin embargo, este ECS extraído sólo se puede modificar si se siguen los controles de la GCS (tratados más adelante en este capítulo). Las flechas en la figura 27.1 ilustran la trayectoria de modificación para un ECS convertido en línea base.

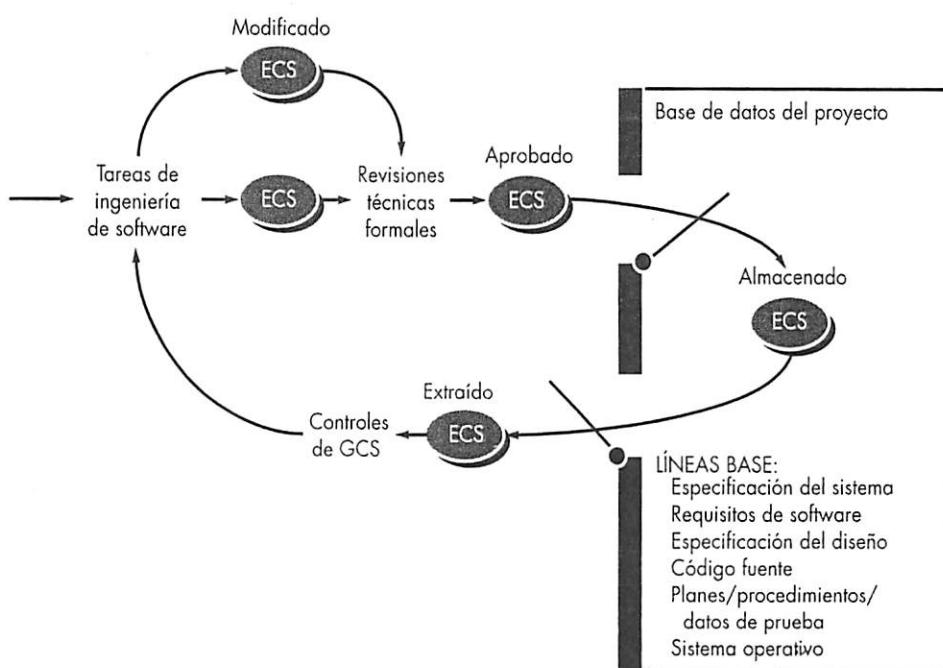
#### 27.1.4 Elementos de configuración del software

Un elemento de configuración del software (ECS) es información que se crea como parte del proceso de ingeniería del software. En el extremo, se puede considerar que un ECS es una sola sección de una gran especificación o un caso de prueba de un gran conjunto de pruebas. De manera más realista, un ECS es un documento, un conjunto completo de casos de prueba o un componente de un programa dado (por ejemplo, una función C++ o un *applet* de Java).

Además de los ECS provenientes de los productos de trabajo de software, muchas organizaciones de ingeniería del software también colocan las herramientas respectivas bajo control de configuración. Esto es: versiones específicas de editores, com-

**FIGURA 27.1**

ECS convertidos en línea base y base de datos del proyecto.

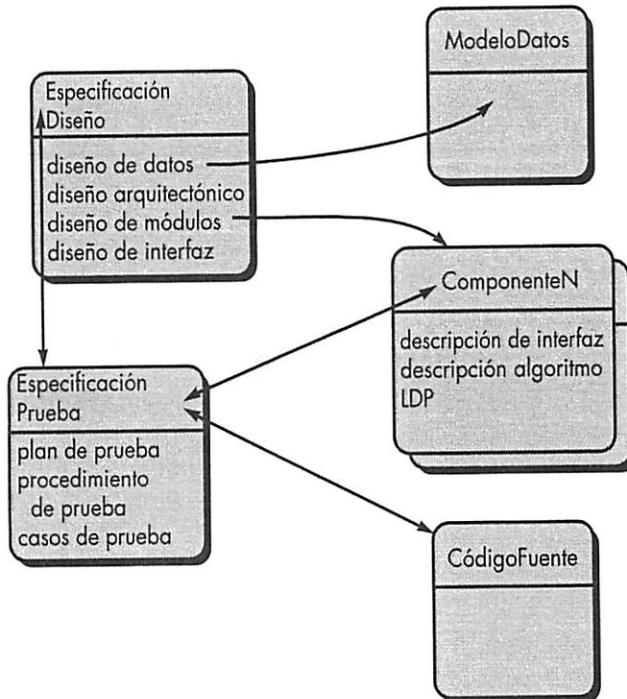


piladores, navegadores y otras herramientas automatizadas se “congelan” como parte de la configuración del software. Puesto que dichas herramientas se utilizaron para producir documentación, código fuente y datos, deben estar disponibles al realizar cambios en la configuración del software. Aunque los problemas son raros, es posible que una nueva versión de una herramienta (por ejemplo, un compilador) produzca resultados diferentes a los de la versión original. Por esta razón, las herramientas, al igual que el software que ayudan a producir, pueden convertirse en línea base como parte de un proceso global de gestión de configuración.

En realidad, los ECS están organizados para formar objetos de configuración susceptibles de catalogar en la base de datos del proyecto con un solo nombre. Un *objeto de configuración* tiene un nombre, atributos y está “conectado” con otros objetos por medio de relaciones. Si se observa la figura 27.2, los objetos de configuración **EspecificacionDiseño**, **ModeloDatos**, **ComponenteN**, **CodigoFuente** y **EspecificacionPrueba** están definidos por separado. Sin embargo, cada uno de los objetos se relaciona con los otros como lo muestran las flechas. Una flecha curva indica una relación de composición. Esto es: **ModeloDatos** y **ComponenteN** son parte del objeto **EspecificacionDiseño**. Una flecha recta con doble punta indica una interrelación. Si se realizase un cambio al objeto **CodigoFuente**, las interrelaciones permiten que un ingeniero de software determine qué otros objetos (y ECS) pueden afectarse.<sup>2</sup>

**FIGURA 27.2**

Objetos de configuración.



<sup>2</sup> Estas relaciones se definen dentro de la base de datos. La estructura de la base de datos (almacén) se estudia con mayor detalle en la sección 27.2.

## 27.2 EL DEPÓSITO DE ECS

En los primeros días de la ingeniería del software los elementos de configuración se conservaban como documentos de papel (¡o tarjetas perforadas!), que se colocaban en cartapacios o carpetas de anillos y se almacenaban en archiveros metálicos. Este enfoque era problemático por muchas razones: 1) con frecuencia era difícil encontrar un elemento de configuración cuando se le necesitaba; 2) usualmente era un reto determinar cuál elemento había sido cambiado, cuándo y por quién; 3) la construcción de una nueva versión de un programa existente consumía mucho tiempo y era proclive al error; 4) la descripción de relaciones detalladas o complejas entre elementos de configuración era virtualmente imposible.

En la actualidad, los ECS se conservan en una base de datos o depósito del proyecto. El diccionario Webster define la palabra *depósito* como “cualquier cosa o persona que se considera como centro de acumulación o almacenamiento”. En los inicios de la ingeniería del software, el depósito de hecho era una persona: el programador, quien tenía que recordar la ubicación de toda la información relevante para un proyecto de software; además, tenía que recuperar la información que nunca se había respaldado por escrito y reconstruir la información perdida. Tristemente, emplear a una persona como “centro de acumulación y almacenamiento” (aunque concuerde con la definición del diccionario) no funciona muy bien. Hoy el depósito es una “cosa”: una base de datos que actúa como el centro tanto de la acumulación como del almacenamiento de la información de ingeniería del software. El papel de la persona (el ingeniero de software) es interactuar con el depósito mediante las herramientas que tiene integradas.

### 27.2.1 El papel del depósito

El depósito de ECS es el conjunto de mecanismos y estructuras de datos que permite que un equipo de software maneje el cambio en una forma eficaz. El depósito proporciona las funciones obvias de un sistema de gestión de base de datos pero, además, el depósito realiza o impulsa las siguientes funciones [FOR89]:



- La *integridad de los datos* incluye funciones para validar las entradas al depósito, garantizar la consistencia entre objetos relacionados y automáticamente realizar modificaciones “en cascada” cuando un cambio en un objeto demanda algún cambio a los objetos relacionados con él.
- El *compartir información* ofrece un mecanismo para distribuir la información entre múltiples desarrolladores y herramientas, manejar y controlar los accesos a los datos por parte de múltiples usuarios y cerrar y abrir los objetos de modo que los cambios no sean trasladados inadvertidamente hacia otros.
- La *integración de herramientas* establece un modelo de datos al que se puede tener acceso mediante muchas herramientas de ingeniería del software, controlar el acceso a los datos y realizar funciones adecuadas de gestión de la configuración.

- La *integración de los datos* brinda funciones de base de datos que permiten que varias tareas de GCS se realicen en uno o más ECS.
- El *fortalecimiento de la metodología* define un modelo de entidad-relación guardado en el depósito que implica un modelo de proceso específico para la ingeniería del software; como mínimo, las relaciones y objetos definen un conjunto de pasos que se deben llevar a cabo para construir los contenidos del depósito.
- *Estandarización de los documentos* es la definición de los objetos en la base de datos que conduce directamente a un enfoque estándar para la creación de documentos de ingeniería del software.

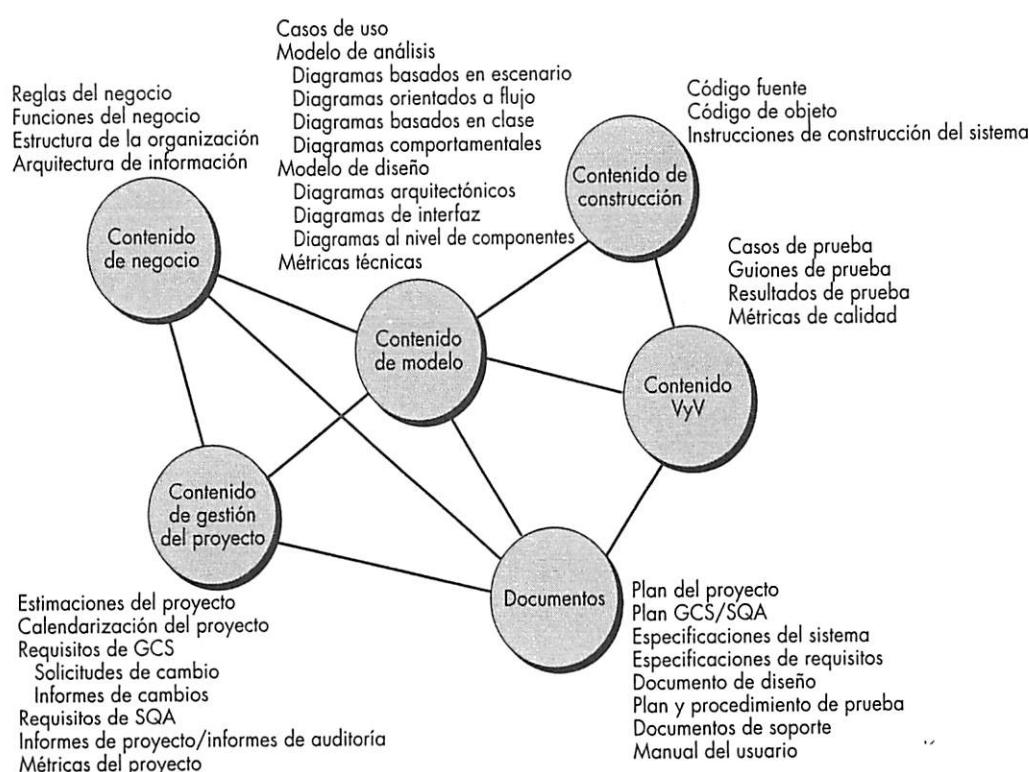
El depósito se define en función de un metamodelo. Para lograr estas funciones el *metamodelo* determina cómo se guarda la información en el depósito, cómo se tiene acceso a los datos mediante las herramientas y cómo los visualizan los ingenieros de software, cuán bien se puede mantener la seguridad e integridad de los datos, y cuán fácilmente se puede ampliar el modelo existente para adecuar las nuevas necesidades. Para mayor información, el lector interesado debe consultar [SHA95] y [GRI95].

### 27.2.2 Características y contenido generales

Las características y el contenido del depósito se comprenden mejor si se les observa desde dos perspectivas: qué se guardará en el depósito y qué servicios específicos ofrece éste. En la figura 27.3 se presenta un análisis detallado de los tipos de representaciones, documentos y productos de trabajo que se guardan en el depósito.

**FIGURA 27.3**

Contenido del depósito.



**Referencia Web**

Ejemplos de depósitos disponibles comercialmente se pueden obtener en [www.software.hp.com/products/SCMGR](http://www.software.hp.com/products/SCMGR) o en [otn.oracle.com/documentation/repository.html](http://oracle.com/documentation/repository.html).

**CLAVE**

El depósito debe ser capaz de mantener los ECS relacionados con muchas versiones diferentes del software. Más importante, debe ofrecer los mecanismos para ensamblar dichos ECS en una configuración específica de versión.

Un depósito robusto proporciona dos clases diferentes de servicios: 1) los mismos tipos de servicios que se pueden esperar de cualquier sistema sofisticado de gestión de base de datos, y 2) servicios específicos del entorno de la ingeniería del software.

Un depósito que atienda a un equipo de ingeniería de software debe 1) integrarse con o directamente apoyar las funciones de gestión de proceso; 2) apoyar reglas específicas que rigen la función de GCS y los datos conservados dentro del depósito; 3) ofrecer una interfaz a otras herramientas de ingeniería del software; y 4) acomodar el almacenamiento de datos sofisticados (por ejemplo, texto, gráficos, video, audio).

### 27.2.3 Características de la GCS

El apoyo a la GCS requiere que el almacén o depósito tenga un conjunto de herramientas que ofrezca soporte para las siguientes características:

**Versiones.** Conforme un proyecto progrese se crearán muchas versiones (sección 27.3.2) de productos de trabajo individuales. El depósito debe ser capaz de guardar todas estas versiones para permitir la gestión eficaz de las liberaciones de producto y permitir que los desarrolladores regresen a versiones anteriores durante las pruebas y la depuración.

El depósito debe ser capaz de controlar una amplia variedad de tipos de objetos, incluso texto, gráficos, mapas de bits, documentos complejos y objeto únicos como definiciones de pantallas e informes, archivos de objeto, datos de prueba y resultados. Un depósito maduro sigue las versiones de los objetos con grados arbitrarios de granularidad; por ejemplo, se puede seguir una sola definición de datos o un conjunto de módulos.

**Gestión del seguimiento de la dependencia y del cambio.** El depósito gestiona una amplia variedad de relaciones entre los objetos de configuración que guarda. Se incluyen relaciones entre entidades y procesos empresariales, entre las partes de un diseño de aplicación, entre componentes de diseño y la arquitectura de información del proyecto, entre elementos de diseño y otros productos de trabajo, etcétera. Algunas de estas relaciones son meramente asociaciones, y algunas son dependencias o relaciones obligatorias.

La habilidad con que se da seguimiento a todas estas relaciones es crucial para la integridad de la información guardada en el depósito y la generación de productos de trabajo basados en ella, y es una de las aportaciones más importantes del concepto de depósito a la mejora del proceso de desarrollo de software. Por ejemplo, si se modifica un diagrama de clase UML, el depósito puede detectar si las clases relacionadas, las definiciones de interfaz y los componentes de código también requieren modificación y pueden colocar en la atención del desarrollador los ECS afectados.

**Seguimiento de requisitos.** Esta función especial ofrece la habilidad de seguir todos los componentes y entregables de diseño y construcción que resulten de una determinación específica de requisitos (seguimiento hacia adelante o seguimiento

propriamente dicho). Además, proporciona la habilidad de identificar qué requisitos generaron algún producto de trabajo dado (seguimiento hacia atrás o rastreo).

**Gestión de la configuración.** Una gestión de la configuración facilita la conservación del rastro de una serie de configuraciones que representan hitos específicos del proyecto o liberaciones de producción.

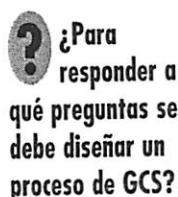
**Rutas de auditoría.** Una ruta de auditoría establece información adicional acerca de cuándo, por qué y por quién se hicieron los cambios. La información acerca de la fuente de los cambios se puede ingresar como atributos de objetos específicos en el depósito.

### 27.3 EL PROCESO DE GCS

El proceso de gestión de la configuración del software define una serie de tareas que tienen cuatro objetivos principales: 1) identificar todos los elementos que colectivamente definen la configuración del software; 2) gestionar los cambios a uno o más de dichos elementos; 3) facilitar la construcción de diferentes versiones de una aplicación; y 4) garantizar que la calidad del software se conserva conforme la configuración evoluciona a lo largo del tiempo.

Un proceso que logra estos objetivos no necesita ser burocrático y molesto, pero sí debe caracterizarse en una forma que permita que un equipo de software desarrolle respuestas a un conjunto de preguntas complejas:

- ¿Cómo identifica un equipo de software los elementos discretos de una configuración de software?
- ¿Cómo gestiona una organización las numerosas versiones existentes de un programa (y su documentación) en una forma que permita que el cambio se acomode eficientemente?
- ¿Cómo controla una organización los cambios antes y después de que el software se libere al cliente?
- ¿Quién tiene la responsabilidad de aprobar y clasificar los cambios?
- ¿Cómo se garantiza que los cambios se hayan realizado adecuadamente?
- ¿Con qué mecanismo se valoran otros cambios que se realizan?

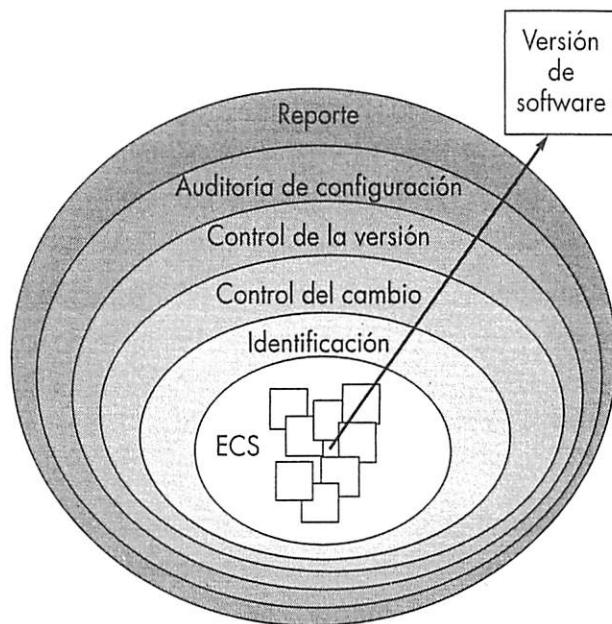


Estas preguntas conducen a la definición de las cinco tareas de la GCS ilustradas en la figura 27.4: identificación, control de la versión, control del cambio, auditoría de la configuración e informe.

En la misma figura las tareas de la GCS se aprecian como capas concéntricas. Los ECS fluyen hacia afuera a través de dichas capas a lo largo de su vida útil, y a final de cuentas se convierten en parte de la configuración del software de una o más versiones de una aplicación o sistema. Conforme un ECS se mueve a través de una capa, las acciones que implica cada capa de proceso de la GCS pueden o no aplicarse. Por ejemplo, cuando se crea un nuevo ECS debe ser identificado. Sin embargo, si

**FIGURA 27.4**

Capas del proceso de GCS.



no se solicitan cambios para el ECS, la capa de control de cambio no se aplica. El ECS se asigna a una versión específica del software (entran en juego mecanismos de control de la versión). Se conserva un registro del ECS (su nombre, fecha de creación, designación de versión, etc.) para propósitos de auditoría de la configuración e informes a quienes necesiten saberlo. En las secciones siguientes se examinan con más detalle cada una de estas capas del proceso de GCS.

### 27.3.1 Identificación de objetos en la configuración del software

El control y la gestión de elementos de configuración del software requiere nombrar cada uno por separado y luego organizarlo mediante un enfoque orientado a objetos. Es posible identificar dos tipos de objetos [CHO89]: básicos y agregados.<sup>3</sup> Un *objeto básico* es una unidad de información creada por un ingeniero de software durante el análisis, el diseño, el código o las pruebas. Por ejemplo, un objeto básico puede ser una sección de una especificación de requisitos, parte de un modelo de diseño, código fuente para un componente, o un conjunto de casos de prueba que se utilizan para ejercitarse el código. Un *objeto agregado* es una colección de objetos básicos y otros objetos agregados. En la figura 27.2 **EspecificacionDiseño** es un objeto agregado. Conceptualmente, es posible verlo como una lista nombrada (identificada) de punteros que especifican objetos básicos como son **ModeloDatos** y **ComponenteN**.

<sup>3</sup> El concepto de objeto agregado [GUS89] se ha propuesto como un mecanismo para representar una versión completa de una configuración de software.

## PUNTO CLAVE

Las interrelaciones establecidas para los objetos de configuración permiten que un ingeniero de software evalúe el impacto del cambio.

## CONSEJO

*Incluso si la base de datos del proyecto ofrece la habilidad para establecer dichas relaciones, éstas consumen tiempo en su establecimiento y dificultan mantener la actualización. Aunque son muy útiles para el análisis de impacto, no son esenciales para la gestión global del cambio.*

## PUNTO CLAVE

Una facilidad “de hechura” permite a un ingeniero de software obtener todos los objetos de configuración relevantes y construir una versión específica del software.

Cada objeto tiene un conjunto de características distintivas que lo identifican de manera exclusiva: un nombre, una descripción, una lista de recursos, y una “realización”. El nombre del objeto es una cadena de caracteres que identifican al objeto sin ambigüedades. La descripción del objeto es una lista de elementos de datos que identifican el tipo de ECS (por ejemplo, elemento modelo, programa, datos) que representa el objeto, un identificador de proyecto e información de cambio y/o versión.

La identificación del objeto de configuración también puede considerar las relaciones entre los objetos nombrados. Por ejemplo, con la utilización de notación simple

**DiagramadeClase <parte de> ModeloAnálisis;**

**ModeloAnálisis <parte de> EspecificacionRequisitos;**

se crea una jerarquía de ECS.

En muchos casos, los objetos están interrelacionados a través de ramas de jerarquía de objetos. Dichas relaciones estructurales cruzadas se representan en la forma siguiente:

**ModeloDatos <interrelacionado> ModeloFlujoDatos**

**ModeloDatos <interrelacionado> CasoPruebaClaseM**

En el primer caso la interrelación es entre un objeto compuesto, mientras que la segunda relación es entre un objeto agregado (**ModeloDatos**) y un objeto básico (**CasoPruebaClaseM**).

El esquema de identificación para los objetos de configuración debe reconocer que los objetos evolucionan a lo largo del proceso de software. Antes de que un objeto se convierta en línea base puede cambiar muchas veces, e incluso después de establecida una línea base los cambios quizás sean muy frecuentes.

### 27.3.2 Control de la versión

El control de la versión combina procedimientos y herramientas para gestionar diferentes versiones de objetos de configuración que se crean durante el proceso del software. Un sistema de control de la versión implementa o está directamente integrado con cuatro grandes capacidades: 1) una base de datos del proyecto (depósito) que guarda todos los objetos de configuración relevantes; 2) una capacidad de gestión de la versión que almacena todas las versiones de un objeto de configuración (o permite que se construya cualquier versión empleando diferencias de versiones anteriores); 3) una *facilidad de hechura* que permite al ingeniero de software recopilar todos los objetos de configuración relevantes y construir una versión específica del software. Además, los sistemas de control de la versión y de control del cambio con frecuencia implementan una capacidad de seguimiento de conflictos (también llamada *seguimiento de bugs*) que permiten al equipo registrar y hacer el seguimiento del estado de todos los conflictos destacados que se asocian con cada objeto de configuración.

"Cualquier cambio, incluso uno para mejorar, está acompañado con inconvenientes e incomodidades."

**Arnold Bennett**

Varios sistemas de control de la versión establecen un *conjunto de cambio* —una colección de todos los cambios (con cierta configuración de línea base)— que requiere la creación de una versión específica del software. Dart [DAR91] advierte que un conjunto de cambios “captura todos los cambios de todos los archivos en la configuración junto con la razón para los cambios y detalles de quién los hizo y cuándo”.

Es posible identificar varios conjuntos de cambio nombrados para una aplicación o sistema. Esto permite que un ingeniero de software construya una versión del software al especificar los conjuntos de cambio (por nombre) que se deben aplicar a la configuración de línea base. Esto se logra aplicando un enfoque de *modelado de sistema*. El modelo de sistema contiene 1) una *plantilla* que incluye una jerarquía de componentes y un “orden de construcción” para los componentes que describe cómo se debe construir el sistema, 2) reglas de construcción y 3) reglas de verificación.<sup>4</sup>

Durante las pasadas dos décadas se han propuesto varios enfoques automatizados para el control de la versión. La principal diferencia en los enfoques es la sofisticación de los atributos que se utilizan en la construcción de versiones específicas y variantes de un sistema y los mecanismos del proceso de construcción.

## HERRAMIENTAS DE SOFTWARE



### **El Sistema de Versiones Concurrentes (SVC)**

El empleo de herramientas con que lograr el control de la versión es esencial para una gestión del cambio eficaz. El *sistema de versiones concurrentes* (SVC; CVS, Concurrent Versions System) es una herramienta ampliamente empleada en el control de versiones. Originalmente diseñada para código fuente, pero útil para cualquier archivo basado en texto, el sistema SVC 1) establece un depósito simple, 2) conserva todas las versiones de un archivo en un archivo con un solo nombre al almacenar sólo las diferencias entre versiones progresivas del archivo original, y 3) protege un archivo contra cambios simultáneos al establecer diferentes directorios para cada desarrollador, con lo que se aislan uno de otro. El SVC mezcla los cambios cuando cada desarrollador completa su trabajo.

Es importante notar que el SVC no es un sistema “de construcción”; esto es, no construye una versión específica

del software. Esto se logra integrando al SVC otras herramientas (por ejemplo, *Makefile*). El SVC no implementa un proceso de control de cambio (por ejemplo, solicitudes de cambio, informes de cambio, seguimiento de bugs).

Pese a sus limitaciones, el SVC “es un sistema dominante en el control de versiones, transparente respecto a la red y de fuente abierta [que] es útil para todos, desde desarrolladores individuales hasta grandes equipos segmentados” [CVS02]. Su arquitectura cliente/servidor permite que los usuarios accedan a los archivos mediante conexiones de Internet y su filosofía de fuente abierta facilita su disponibilidad en la mayoría de las plataformas populares.

El SVC está disponible sin costo para entornos Windows, Macintosh y Unix. Visítese [www.cvshome.org](http://www.cvshome.org) para mayores detalles.

<sup>4</sup> También es posible consultar el modelo de sistema para valorar cómo un cambio en un componente impactará a otros componentes.

### 27.3.3 Control del cambio

La realidad del control del cambio en un contexto moderno de ingeniería del software la resumió bellamente James Bach [BAC98]:

El control del cambio es vital. Pero las fuerzas que lo hacen necesario también lo tornan irritante. Nos preocupamos por los cambios porque una pequeña perturbación en el código puede crear una gran falla en el producto. Pero también puede resolver una gran falla o permitir maravillosas nuevas capacidades. Nos preocupamos por los cambios porque un solo desarrollador solitario podría hundir el proyecto; aunque en las mentes de dichos solitarios se originan ideas brillantes, y un proceso de control del cambio gravoso podría desalentarlos efectivamente de realizar trabajo creativo.

Bach reconoce que se enfrenta un acto de equilibrio. Demasiado control del cambio, y se crean problemas; poco, y se crean otros problemas.

*"El arte del progreso es preservar el orden entre el cambio, y preservar el cambio entre el orden."*

Alfred North Whitehead

## PUNTO CLAVE

Se debe destacar que varias solicitudes de cambio pueden combinarse para resultar en una sola OCI y que las OCI usualmente resultan en cambios a múltiples objetos de configuración.

En un gran proyecto de ingeniería de software el cambio incontrolado conduce rápidamente al caos. Respecto a tales proyectos el control del cambio combina procedimientos humanos y herramientas automatizadas. En la figura 27.5 se ilustra esquemáticamente el proceso de control del cambio. Se emite una *solicitud de cambio* y se estima para evaluar los méritos técnicos, los potenciales efectos colaterales, el impacto global sobre otros objetos de configuración y funciones del sistema, y el costo proyectado del cambio. Los resultados de la evaluación se presentan como un *informe de cambio*, que lo utiliza una *autoridad del control del cambio* (ACC): una persona o grupo que toman la decisión final acerca del estado y la prioridad del cambio. Se genera una *orden de cambio en la ingeniería* (OCI) para cada cambio aprobado. La OCI describe el cambio que se debe realizar, las restricciones insoslayables y los criterios de revisión y auditoría.

El objeto que se cambiará se coloca en un directorio que controle exclusivamente el ingeniero de software que realiza el cambio. Un sistema de control de la versión (véase el recuadro acerca de SVC) actualiza el archivo original una vez realizado el cambio. Como alternativa, el objeto que se cambiará puede "salir" de la base de datos del proyecto (depósito), realizar el cambio y aplicar las actividades apropiadas de SQA. Luego el objeto "entra" a la base de datos y se aplican mecanismos adecuados de control de versión (sección 27.3.2) para crear la siguiente versión del software.

Estos mecanismos de control de la versión, integrados en el proceso de control de cambios, implementan dos importantes elementos de gestión del cambio: control del acceso y de la sincronización. El *control del acceso* rige qué ingenieros de software están autorizados para ingresar y modificar un objeto de configuración particular. El *control de la sincronización* ayuda a garantizar que los cambios paralelos, efectuados por dos personas diferentes, no se sobrescriben uno sobre otro [HAR89].

## CONSEJO

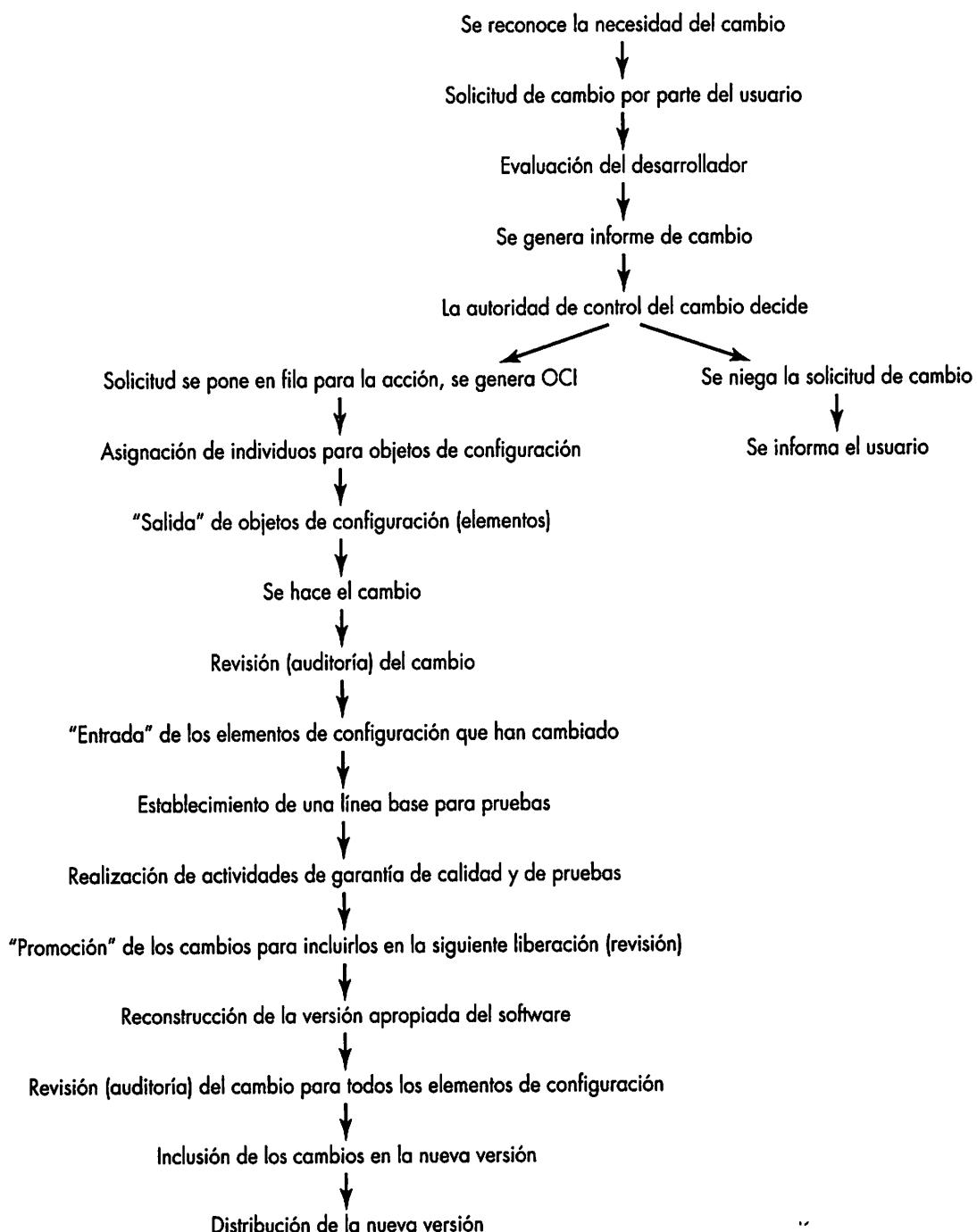
La confusión conduce a errores, algunos de ellos bastante serios. El control del acceso y de la sincronización evitan confusión. Empléense herramientas de control de la versión y el cambio que implementen ambos.

Algunos lectores quizá comiencen a sentirse incómodos con el grado de burocracia que implica la descripción del proceso de control del cambio mostrada en la figura 27.5. Este sentimiento es común. Sin las salvaguardas adecuadas el control del cambio puede retrasar el progreso y crear burocracia y papeleo innecesarios. La mayoría de los desarrolladores de software con mecanismos de control del cambio (desafortunadamente muchos no los tienen) ha creado varias capas de control para ayudarse a evitar los problemas mencionados aquí.

Antes de que un ECS se convierta en línea base sólo se necesita aplicar *control de cambio informal*. El desarrollador del objeto de configuración (ECS) en cuestión tiene

**FIGURA 27.5**

El proceso de control del cambio.





Óptese por un poco más de control de cambio del que se crea que necesitará. Es probable que demasiado será la cantidad correcta.

la posibilidad de realizar cualesquiera cambios justificados por el proyecto y los requisitos técnicos (en tanto los cambios no afecten requisitos de sistema más amplios que se encuentran fuera del ámbito de trabajo del desarrollador). Una vez que el objeto haya experimentado una revisión técnica formal y haya sido aprobado, se puede crear una línea base.<sup>5</sup> Una vez que un ECS se convierte en línea base se implementa un *control de cambio a nivel del proyecto*. Ahora, para realizar un cambio, el desarrollador debe obtener la aprobación del gestor del proyecto (si el cambio es "local") o de la ACC si el cambio afecta otros ECS. En algunos casos, la generación formal de las solicitudes de cambio, los informes de cambio y las OCI se distribuyen. Sin embargo, se lleva a cabo la evaluación de cada cambio, y todos los cambios se siguen y revisan.

Cuando el producto de software se libera entre los clientes se instituye el *control de cambio formal*. En la figura 27.5 se ha esbozado el procedimiento de control de cambio formal.

"El cambio es inevitable, excepto para las máquinas expendedoras."

**Calcomanía en un parachoques**

La autoridad de control del cambio juega un papel activo en la segunda y tercera capas del control. Dependiendo del tamaño y carácter de un proyecto de software, la ACC puede estar compuesta de una persona (el gestor del proyecto) o varias personas (por ejemplo, representantes de software, hardware, ingeniería de bases de datos, soporte, mercadotecnia). El papel de la ACC es tener una visión global, esto es: evaluar el impacto del cambio más allá del ECS en cuestión. ¿Cómo afectará el cambio al hardware? ¿Cómo afectará al desempeño? ¿Cómo modificará la percepción del cliente acerca del producto? ¿Cómo afectará la calidad y fiabilidad del producto? Estas y muchas otras preguntas las aborda la ACC.

## HOGARSEGURO

### Problemas en la GCS



**El escenario:** Oficina de Doug Miller cuando comienza el proyecto del software HogarSeguro.

**Los actores:** Doug Miller (gerente del equipo de ingeniería del software de HogarSeguro) y Vinod Raman, Jamie Lazar y otros miembros del equipo de ingeniería del software del producto.

#### La conversación:

**Doug:** Yo sé que es muy pronto, pero tenemos que hablar acerca de la gestión del cambio.

**Vinod (ríe):** Apenas. Mercadotecnia llamó esta mañana con unos cuantos "segundos pensamientos". Nada importante, pero es sólo el comienzo.

<sup>5</sup> Se puede crear una línea base también por otras razones. Por ejemplo, cuando se crean "construcciones diarias" todos los componentes que entran en un tiempo dado se convierten en la línea base para el trabajo del día siguiente.

**Jamie:** Hemos sido bastante informales acerca de la gestión del cambio en proyectos anteriores.

**Doug:** Lo sé, pero éste es mayor y más visible, y según recuerdo...

**Vinod (asiente con la cabeza):** Fuimos asesinados por cambios incontrolados en el proyecto de control de la iluminación en el hogar... recuerda las demoras que...

**Doug (frunce el ceño):** Una pesadilla que prefiero no revivir.

**Jamie:** Así que qué hacemos.

**Doug:** Como lo veo, tres cosas. Primero, tenemos que desarrollar, o pedir prestado, un proceso de control de cambio.

**Jamie:** ¿Te refieres a cómo la gente solicita los cambios?

**Vinod:** Sí, pero también cómo evaluamos el cambio, decidimos cuándo hacerlo (si eso es lo que decidimos) y

cómo conservamos los registros de lo que afecta el cambio.

**Doug:** Segundo, tenemos que obtener una herramienta de GCS realmente buena para control del cambio y de la versión.

**Jamie:** Podemos construir una base de datos para todos nuestros productos de trabajo.

**Vinod:** Se llaman ECS en este contexto, y la mayoría de las buenas herramientas ofrecen cierto soporte para eso.

**Doug:** Ese es un buen comienzo, ahora tenemos que...

**Jamie:** Oye, Doug: dijiste que eran tres cosas...

**Doug (sonríe):** Tercera: todos tenemos que comprometernos a seguir el proceso de gestión del cambio y usar las herramientas, sin importar cuáles sean, ¿de acuerdo?

#### 27.3.4 Auditoría de la configuración

La identificación, el control de la versión y el control del cambio ayudan al desarrollador del software a mantener el orden en lo que de otro modo sería una situación caótica e inestable. Sin embargo, incluso el mecanismo de control más exitoso sólo sigue un cambio hasta que no se genera una OCI. ¿Cómo se puede garantizar que el cambio se ha implementado con propiedad? La respuesta es doble: 1) revisiones técnicas formales y 2) auditoría de la configuración del software.

La revisión técnica formal (presentada con detalle en el capítulo 26) se enfoca en la corrección técnica del objeto de configuración que se ha modificado. Los revisores evalúan el ECS para determinar su consistencia con otros ECS, omisiones o potenciales efectos colaterales. Se debe realizar una revisión técnica formal en casi la mayoría de los cambios triviales.

Una *auditoría de configuración del software* complementa la revisión técnica formal al abordar las siguientes preguntas:

1. ¿Se ha realizado el cambio especificado en la OCI? ¿Se han incorporado modificaciones adicionales?
2. ¿Se ha realizado una revisión técnica formal para evaluar la corrección técnica?
3. ¿Se ha seguido el proceso de software? ¿Se han aplicado adecuadamente los estándares de ingeniería del software?
4. ¿El cambio se ha resaltado en el ECS? ¿Se han especificado la fecha y el autor del cambio? ¿Los atributos del objeto de configuración reflejan el cambio?

 ¿Cuáles son las principales preguntas que se plantean durante una auditoría de configuración?

5. ¿Se han seguido los procedimientos de GCS para destacar el cambio, registrarlo e informar de él?
6. ¿Todos los ECS relacionados se han actualizado de manera adecuada?

En algunos casos, las preguntas de la auditoría se plantean como parte de una revisión técnica formal. Sin embargo, cuando la GCS es una actividad formal, la auditoría de GCS la lleva a cabo por separado del grupo de aseguramiento de la calidad. Tales auditorías formales de configuración también aseguran que los ECS correctos (por versión) se han incorporado en una construcción específica y que toda la documentación está actualizada y es consistente con la versión que se ha construido.



*Elabórese una lista de "necesita conocer" para cada objeto de configuración y consérvese actualizada. Cuando se realice un cambio, es necesario asegurarse de que todos los de la lista sean notificados.*

### 27.3.5 Informe de estado

El *informe de estado de la configuración* (a veces llamado *contabilidad de estado*) es una tarea de GCS que responde las siguientes preguntas: 1) ¿qué ocurrió? 2) ¿quién lo hizo? 3) ¿cuándo ocurrió? 4) ¿qué otra cosa será afectada?

En la figura 27.5 se muestra el flujo de información para el informe de estado de la configuración (IEC). Cada vez que se le asigna una identificación nueva o actualizada a un ECS se efectúa una entrada de IEC. Cada vez que la ACC aprueba un cambio (es decir, se expide una OCI) se genera una entrada en el IEC. Cada vez que se realiza una auditoría de la configuración los resultados se reportan como parte de la tarea de IEC. El resultado del IEC es posible colocarlo en una base de datos en línea o en un sitio Web, de modo que los desarrolladores y los encargados del mantenimiento del software pueden tener acceso a la información del cambio mediante categorías clave. Además, se genera un IEC con regularidad y su finalidad es mantener a los gestores y profesionales alertas ante de los cambios importantes.

## HERRAMIENTAS DE SOFTWARE



### Soporte de la GCS

**Objetivo:** Las herramientas de GCS proporcionan soporte a una o más de las actividades del proceso estudiadas en la sección 27.3.

**Mecánica:** La mayoría de las modernas herramientas de GCS funciona en conjunto con un depósito (un sistema de base de datos) y ofrecen mecanismos para identificar, control de la versión y el cambio, auditoría e informe.

### Herramientas representativas<sup>6</sup>

CCC/Harvest, distribuida por Computer Associates ([www.cai.com](http://www.cai.com)), es un sistema de GCS multiplataforma.

ClearCase, desarrollada por Rational ([www.rational.com](http://www.rational.com)), ofrece una familia de funciones de GCS.

Concurrent Versions System (SVC), una herramienta de fuente abierta ([www.cvshome.org](http://www.cvshome.org)), es uno de los sistemas de control de versión más ampliamente empleados en la industria (véase un recuadro anterior).

PVCS, distribuida por Merant ([www.merant.com](http://www.merant.com)), ofrece un conjunto completo de herramientas de GCS que son aplicables tanto en software convencional como en WebApps.

<sup>6</sup> Las herramientas mencionadas sólo representan una muestra de esta categoría. En la mayoría de los casos los nombres de las mismas son marcas registradas por sus respectivos desarrolladores.

SourceForge, distribuida por VA Software ([sourceforge.net](http://sourceforge.net)), ofrece gestión de versión, capacidades de construcción, seguimiento de problemas/bugs y muchas otras características de gestión.

SurroundSCM, desarrollada por Seapine Software ([www.seapine.com](http://www.seapine.com)), proporciona capacidades completas de gestión del cambio.

Vesta, distribuida por Compac ([www.vestasys.org](http://www.vestasys.org)), es un sistema de GCS de dominio público que puede dar soporte tanto a proyectos pequeños (< 10 KLDC) como a grandes (10,000 KLDC).

Una extensa lista de herramientas comerciales y entornos para GCS, se puede encontrar en [www.cmtoday.com/yp/commercial.html](http://www.cmtoday.com/yp/commercial.html).

## 27.4 GESTIÓN DE LA CONFIGURACIÓN PARA INGENIERÍA WEB

En la parte 3 de este libro se estudió la naturaleza especial de las aplicaciones Web y el proceso de ingeniería Web necesario para construirlo. Entre las muchas características que diferencian a las WebApps del software convencional se encuentra la naturaleza ubicua del cambio.

La ingeniería Web utiliza un modelo de proceso incremental iterativo (capítulo 16) que aplica muchos principios derivados del desarrollo de software ágil (capítulo 4). Al utilizar este enfoque, un equipo de ingeniería con frecuencia desarrolla un incremento de WebApp en un periodo muy corto mediante un enfoque basado en el cliente. Los incrementos subsecuentes agregan contenido y funcionalidad adicionales, y tal vez cada uno implemente cambios que conduzcan a contenido aumentado, mejor facilidad de uso, estética mejorada, mejor navegación, desempeño aumentando y mayor seguridad. En consecuencia, en el mundo ágil de la ingeniería Web el cambio se ve de manera un poco diferente.

Los ingenieros Web deben adoptar el cambio, e incluso un típico equipo ágil evita todas las cosas que parecen pesados procesos, burocráticos y formales. Por lo general, se considera (aunque incorrectamente) que la gestión de la configuración del software posee estas características. Esta aparente contradicción se resuelve al no rechazar los principios, prácticas y herramientas de la GCS, sino más bien moldeándolas para satisfacer las necesidades especiales de los proyectos de ingeniería Web.

### 27.4.1 Problemas en la gestión de la configuración para WebApps

Conforme las WebApps se vuelven cada vez más importantes para la sobrevivencia y el crecimiento de los negocios, crece la necesidad de la gestión de la configuración. ¿Por qué? Porque sin controles eficaces los cambios inadecuados a una WebApp (recuérdese que la inmediatez y la evolución continua son los atributos dominantes de muchas WebApps) conducirían a la difusión no autorizada de información de un nuevo producto; funcionalidad errónea o pobremente probada que frustra a los visitantes a un sitio Web; hoyos en la seguridad que ponen en peligro los sistemas internos de la compañía; y otras consecuencias económicamente desagradables o incluso desastrosas.

¿Qué impacto tiene un cambio descontrolado sobre una WebApp?