

29

Gestión de configuraciones

Objetivos

El objetivo de este capítulo es introducir el proceso de gestión del código y la documentación de un sistema de software evolutivo. Cuando termine de leer este capítulo:

- comprenderá por qué es importante la gestión de la configuración del software;
- habrá sido introducido en cuatro actividades principales de la gestión de configuraciones: planificación de la gestión de configuraciones, gestión de los cambios, gestión de versiones y entregas, y construcción de sistemas;
- aprenderá cómo se pueden utilizar las herramientas CASE para apoyar los procesos de gestión de configuraciones.

Contenidos

- 29.1 Planificación de la gestión de configuraciones**
- 29.2 Gestión del cambio**
- 29.3 Gestión de versiones y entregas**
- 29.4 Construcción del sistema**
- 29.5 Herramientas CASE para la gestión de configuraciones**

La gestión de configuraciones (CM) es el desarrollo y aplicación de estándares y procedimientos para gestionar un sistema software evolutivo. Como se expuso en el Capítulo 7, los requerimientos del sistema siempre cambian durante su desarrollo y su uso, y se tienen que incorporar estos requerimientos en nuevas versiones del sistema. Es necesario gestionar estos sistemas que evolucionan, porque es fácil perder la pista de los cambios que se han incorporado dentro de cada versión. Las versiones incorporan propuestas de cambios, correcciones de fallos y, adaptaciones para hardware y sistemas operativos diferentes. Pueden existir varias versiones en desarrollo y en uso al mismo tiempo. Si no se tienen unos procedimientos de gestión de configuraciones adecuados, se puede hacer un esfuerzo inútil modificando la versión errónea de un sistema, entregar una versión incorrecta a los clientes o perder la pista de dónde ha sido guardado el código fuente.

Los procedimientos de gestión de configuraciones definen cómo registrar y procesar los cambios propuestos al sistema, cómo relacionar éstos con los componentes del sistema y los métodos utilizados para identificar las diversas versiones del sistema. Las herramientas de gestión de configuraciones se utilizan para almacenar las versiones de los componentes del sistema, construir sistemas a partir de estos componentes y llevar el registro de entregas de las versiones del sistema a los clientes.

Algunas veces, la gestión de configuraciones es parte de un proceso general de gestión de la calidad del software (expuesto en el Capítulo 27), donde el mismo gestor comparte las responsabilidades de la gestión de la calidad y de la configuración. Los desarrolladores del software entregan éste al equipo de garantía de calidad, quienes son responsables de la comprobación de que el sistema es de calidad aceptable. Aquí comienza a ser un sistema controlado, lo que significa que los cambios en el sistema tienen que ser acordados y registrados antes de ser implementados. Algunas veces, los sistemas controlados se denominan «líneas base» puesto que son el punto de inicio para la evolución controlada.

Hay muchas razones por las que los sistemas existen en diversas configuraciones. Éstas se producen para diferentes computadoras, para diferentes sistemas operativos, para incorporar funciones específicas del cliente, etcétera (véase la Figura 29.1). Los gestores de la configuración son responsables de llevar los registros de las diferencias entre las versiones del software, para asegurar que las nuevas versiones se deriven de forma controlada y para entregar las nuevas versiones a los clientes correctos en el momento justo.

La definición y uso de gestión de configuraciones es esencial para la certificación de calidad ISO 9000 y para los estándares CMM y CMMI (Pault *et al.*, 1995; Ahern *et al.*, 2001; Peñach, 1996). Un ejemplo de tal estándar es el IEEE 828-1998, que define un estándar para los planes de la gestión de configuraciones. Dentro de una organización, estos estándares se publican en un manual de gestión de configuraciones o como parte del manual de calidad. Los estándares externos se utilizan como base para estándares organizacionales detallados y se ajustan a un entorno específico.

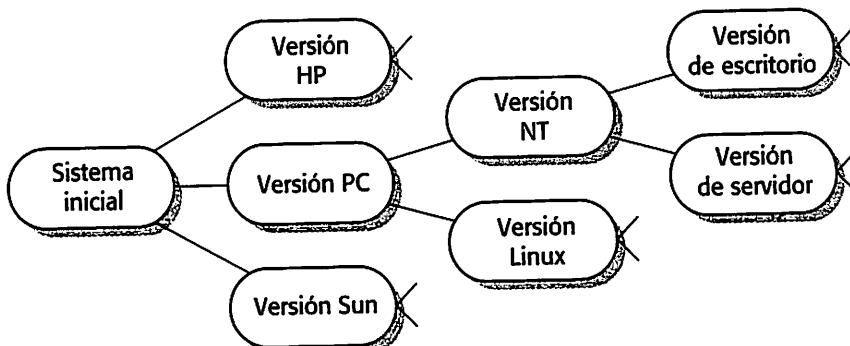


Figura 29.1
Familias de sistemas.

En un proceso tradicional de desarrollo de software basado en el modelo en «cascada» (véase el Capítulo 4), el software se entrega al equipo de gestión de configuraciones después de que el desarrollo haya sido completado y se hayan probado los componentes de software. Luego este equipo tiene la responsabilidad de construir el sistema completo y gestionar las pruebas del sistema. Los fallos encontrados durante las pruebas del sistema se devuelven al equipo de desarrollo para su reparación. A continuación reparan el fallo y entregan una nueva versión del componente reparado al equipo de la garantía de calidad. Si la calidad es aceptable, éste pasa a ser la nueva línea base para el desarrollo del sistema.

Este modelo, donde el equipo de garantía de calidad controla la integración del sistema y el proceso de pruebas, ha influenciado el desarrollo de estándares para la gestión de configuraciones. Muchos estándares de garantía de calidad suponen que se utiliza el modelo en cascada para el proceso del software en el desarrollo de sistemas (Bersoff y Davis, 1991). Esto significa que los estándares tienen que adaptarse a las aproximaciones modernas de desarrollo de software basadas en especificación y desarrollo incremental. Hass (Hass, 2003) discute alguna de estas adaptaciones para los procesos de desarrollo software tales como el desarrollo ágil.

Para ajustarse a este desarrollo incremental, algunas organizaciones han desarrollado un nuevo enfoque para gestionar las configuraciones que permite el desarrollo concurrente y las pruebas del sistema. Este enfoque se basa en la forma regular (a menudo diaria) de construcción del sistema completo a partir de sus componentes:

1. La organización que lleva a cabo el desarrollo fija una hora de entrega (por ejemplo, a las 14.00 horas) de los componentes del sistema. Si los desarrolladores tienen nuevas versiones de los componentes que están escribiendo, entonces deben entregarlos todos a esa hora. Los componentes podrían estar incompletos, pero deben ser capaces de proveer alguna funcionalidad básica que se pueda probar.
2. Una nueva versión del sistema se construye a partir de estos componentes compilándolos y vinculándolos para formar un sistema completo.
3. Entonces el sistema se entrega al equipo de pruebas, que lleva a cabo un conjunto predefinido de pruebas sobre el sistema. Al mismo tiempo, los desarrolladores siguen trabajando en sus componentes, agregando funcionalidades y reparando los fallos descubiertos en las pruebas previas.
4. Los fallos encontrados durante las pruebas del sistema se documentan y se devuelven a los desarrolladores del sistema. Éstos reparan dichos fallos en una versión ulterior del componente.

La principal ventaja de utilizar construcciones diarias del software es que se incrementan las oportunidades de encontrar problemas que surgen a partir de las interacciones al inicio del proceso. Más aún, las construcciones diarias provocan las pruebas de las unidades de los componentes. Psicológicamente, los desarrolladores se ven presionados «para no derribar el edificio», es decir, para no entregar versiones de los componentes que provoquen un fallo en todo el sistema. Por lo tanto, son reticentes a entregar nuevas versiones de los componentes que no se hayan probado adecuadamente. Si se encuentran fallos en el software y éstas se abordan durante las pruebas de las unidades, se invierte menos tiempo en las pruebas del sistema.

La utilización exitosa de las construcciones diarias requiere procesos de gestión del cambio muy rigurosos para llevar un registro de los problemas encontrados y resueltos. También conduce a la gestión de un número muy grande de versiones del sistema y de los componentes. Por lo tanto, una buena gestión de configuraciones es esencial para que este enfoque tenga éxito.

La gestión de configuraciones en el desarrollo ágil y desarrollo rápido no pueden basarse en rígidos procedimientos y papeleo burocrático. Aunque éstos pueden ser necesarios para proyectos grandes o complejos, pueden ralentizar el proceso de desarrollo. Mantener registros cuidadosos es esencial para sistemas grandes y complejos desarrollados desde diferentes ubicaciones, pero innecesario en proyectos pequeños. En estos proyectos, todos los miembros del equipo trabajan juntos en la misma habitación, y la sobrecarga asociada a mantener los registros ralentiza el proceso de desarrollo. Sin embargo, esto no significa que, cuando se requiera un desarrollo rápido, la gestión de configuraciones deba ser totalmente abandonada. Los procesos ágiles utilizan herramientas simples de gestión de configuraciones, como un gestor de versiones y herramientas para la construcción del sistema, que incorporarán algo de control. Todos los miembros del equipo tienen que aprender a utilizar estas herramientas y asumir las disciplinas que ellas imponen.

29.1 Planificación de la gestión de configuraciones

Un plan de gestión de configuraciones describe los estándares y procedimientos utilizados para la gestión de la configuración. El punto de inicio para desarrollar el plan es un conjunto de estándares generales de gestión de la configuración de toda la compañía adaptables a cada proyecto específico. El plan de la CM se organiza en varios capítulos que incluyen:

1. La definición de lo que se debe gestionar (los elementos de configuración) y el esquema formal para identificar estas entidades.
2. Un enunciado de quién toma la responsabilidad de los procedimientos de gestión de configuraciones y quién envía las entidades controladas al equipo de gestión de configuraciones.
3. Las políticas de gestión de configuraciones utilizadas para gestionar el control de los cambios y las versiones.
4. Una descripción de las herramientas a utilizar para la gestión de configuraciones y el proceso a aplicar cuando se utilizan estas herramientas.
5. Una definición de la base de datos de la configuración que se utilizará para registrar la información de la configuración.

En el plan de la CM se incluye información adicional de la gestión del software por parte de los proveedores externos y los procesos de auditoría para el proceso de la CM.

Una parte importante del plan de la CM es la definición de responsabilidades. Define quién es el responsable de la entrega de cada documento o de cada componente de software para la garantía de la calidad y la gestión de la configuraciones. También define los revisores de cada documento. La persona responsable de la entrega de los documentos no es preciso que sea la misma responsable de producir el documento. Para simplificar las interfaces, a menudo es conveniente hacer que los gestores de proyectos o los líderes del equipo sean responsables de todos los documentos producidos por su equipo.

29.1.1 Identificación de los elementos de configuración

En un sistema grande de software, puede haber cientos de módulos de código fuente, scripts de pruebas, documentos de diseño, etc. Éstos son producidos por diferentes personas y, cuando fueron creados, pudieron tener nombres similares. Para seguir el registro de toda esta in-

formación y encontrar el fichero adecuado cuando éste se precise, se necesitará un esquema consistente de identificación para todos los elementos del sistema de gestión de configuraciones.

Durante el proceso de planificación de la gestión de configuraciones, se decide exactamente qué elementos (o clases de elementos) se van a controlar. Los documentos o grupos de documentos relacionados del control de la configuración son documentos formales o elementos de la configuración. Normalmente, los planes del proyecto, las especificaciones, los diseños, los programas y los conjuntos de datos de prueba son elementos de la configuración. Todos los documentos que son necesarios para el mantenimiento futuro del sistema deben ser controlados por el sistema de control de configuraciones.

Sin embargo, esto no significa que todos los documentos o archivos deban estar bajo el control de configuraciones. Documentos como, por ejemplo, documentos técnicos de trabajo que presentan la captura de ideas para el posterior desarrollo, minutos de las reuniones de grupo, bosquejo del plan y propuestas, no tendrán relevancia a largo plazo y no serán necesarios para el futuro mantenimiento del sistema.

El esquema de asignación de nombres a los documentos debe asignar un nombre único a todos los documentos de control de la configuración. Este nombre debe reflejar el tipo de elemento, la parte del sistema en la que se utiliza y el creador del elemento, entre otros. En su esquema de nombres, también deberá reflejar las relaciones entre elementos para asegurar que los documentos relacionados tengan una raíz común de su nombre. Esto conduce a un esquema de asignación de nombres jerárquico. Algunos ejemplos de los nombres son:

PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE/CODE

PCL-TOOLS/EDIT/HELP/QUERY/HELPFRAMES/FR-1

La parte inicial del nombre es el nombre del proyecto, PCL-TOOLS. En este proyecto, existen cuatro herramientas diferentes; el nombre de la herramienta (EDIT) se utiliza como la siguiente parte del nombre. Cada herramienta está compuesta de módulos nombrados de forma distinta, cuyos nombres pasan a ser parte del identificador (FORMS, HELP). Este proceso de descomposición continúa hasta que se haga referencia a los documentos formales en el nivel base (véase la Figura 29.2). Las hojas de la jerarquía de la documentación son elementos de configuración formales. La Figura 29.2 muestra que se requieren tres documentos

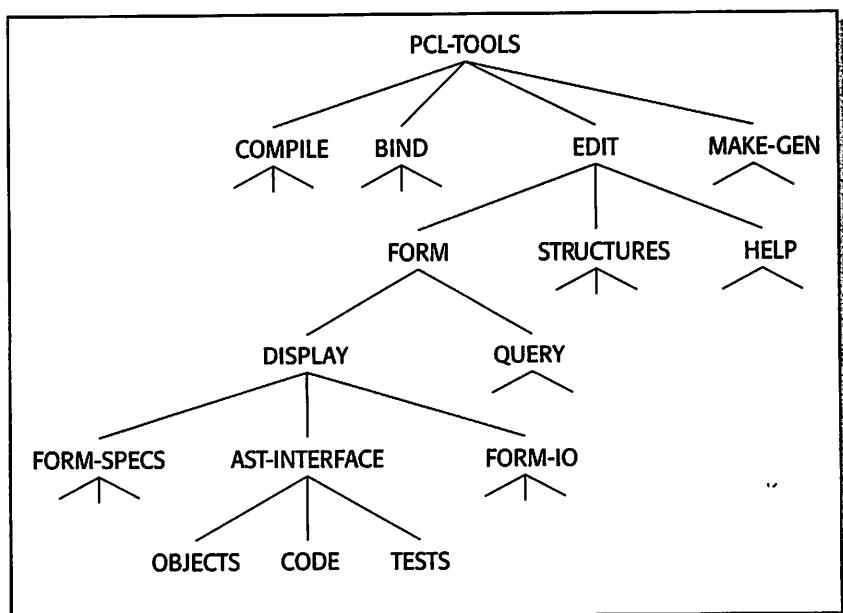


Figura 29.2
Jerarquía de la configuración usada para la asignación de identificadores.

formales para cada componente: una descripción de los objetos (OBJECTS), el código del componente (CODE) y un conjunto de pruebas para el código (TEST). Elementos como la ayuda son también gestionados y tienen diferentes nombres (FR-1, en el ejemplo anterior).

La asignación de nombres jerárquica es simple y fácil de entender, y a veces copia la estructura de directorios utilizada para almacenar los archivos del proyecto. Sin embargo, reflejan la estructura del proyecto cuando se desarrolló el software. Los nombres de los elementos de configuración asociados a un proyecto particular pueden reducir las oportunidades de reutilización. Puede ser muy difícil encontrar componentes relacionados (por ejemplo, todos los componentes desarrollados por el mismo programador) donde el esquema de nombres no refleja esta relación.

29.1.2 La base de datos de configuraciones

La base de datos de configuraciones se utiliza para registrar toda la información relacionada con las configuraciones y sus elementos. Sus funciones principales son ayudar a la evaluación del impacto de los cambios en el sistema y proveer información de la gestión acerca del proceso de la CM. Además de definir el esquema de la base de datos de la configuración, como parte del proceso de planificación de la CM también se deben definir los formularios y los procedimientos para registrar y recuperar la información del proyecto.

Una base de datos de configuraciones no incluye información acerca de los elementos de configuración. Registra información acerca de los usuarios de los componentes, los clientes del sistema, plataformas de ejecución, cambios propuestos, etc. Le debe ser posible suministrar respuestas a una variedad de consultas acerca de las configuraciones del sistema. Algunas consultas podrían ser:

1. ¿A qué clientes se les ha entregado una versión particular del sistema?
2. ¿Qué configuración de hardware y del sistema operativo se requiere para ejecutar una versión dada del sistema?
3. ¿Cuántas versiones del sistema se han creado y cuáles son sus fechas de creación?
4. ¿Qué versiones del sistema se ven afectadas si se cambia un componente particular?
5. ¿Cuántas peticiones de cambios están pendientes para una versión particular?
6. ¿Cuántos fallos declarados existen en una versión particular?

De forma ideal, la base de datos de configuraciones se integra en el sistema de gestión de las versiones utilizado para almacenar y gestionar los documentos formales del proyecto. Este enfoque, apoyado por algunas herramientas CASE integradas, hace posible vincular los cambios de forma directa con los documentos y componentes afectados por el cambio. Se da mantenimiento a los vínculos entre los documentos, como los documentos de diseño y el código del programa, con el fin de que sea relativamente fácil encontrar todo lo que debe modificarse cuando se propone un cambio.

Sin embargo, las herramientas CASE integradas para la gestión de configuraciones son caras. Muchas compañías no las utilizan, sino que mantienen su base de datos de configuraciones como un sistema independiente de su sistema de control de versiones. Los elementos de la configuración se almacenan en archivos o en el sistema de gestión de versiones como el CVS (Berliner, 1990), que se verá más adelante en este capítulo.

Esta base de datos de configuraciones almacena información de los elementos de la configuración y hace referencia a sus nombres de archivos en el sistema de gestión de versiones. Aunque éste es un enfoque relativamente económico y flexible, su desventaja es que los ele-

mentos de la configuración se pueden cambiar sin que la base de datos de configuraciones tenga conocimiento. Por lo tanto, no se puede estar seguro de que la base de datos de la configuración sea una descripción actualizada del estado del sistema.

29.2 Gestión del cambio

El cambio es un hecho en la vida de los sistemas de software grandes. Como se comentó en los primeros capítulos, las necesidades organizacionales y los requerimientos cambian durante el tiempo de vida de un sistema. Esto significa que habrá que hacer los correspondientes cambios al software del sistema. Para asegurarse de que estos cambios se hagan de forma correcta, se necesitará un conjunto de herramientas de soporte para los procedimientos de gestión de cambios.

Los procedimientos de gestión de cambios se ocupan del análisis de costes y beneficios de los cambios propuestos, aprobando aquellos cambios que merecen la pena y registrando los componentes del sistema que se tienen que cambiar. El proceso de gestión de cambios (véase la Figura 29.3) se lleva a cabo cuando el software o la documentación asociada se pone bajo el control del equipo de gestión de configuraciones.

La primera etapa del proceso de gestión del cambio es completar un formulario de solicitud de cambios (CRF) en el cual el solicitante señala los cambios requeridos en el sistema. Además de registrar los cambios requeridos, el CRF registra las recomendaciones pertinentes al cambio, los costes estimados del cambio y las fechas en las que se solicita, prueba, implementa y valida el cambio. También incluye una sección donde el analista señala cómo implementar el cambio.

En la Figura 29.4 se muestra un ejemplo de un formulario de solicitud de cambios, el cual se encuentra parcialmente llenado. Por lo general, los formularios de solicitud de cambios se definen durante el proceso de planificación de la CM. Éste es un ejemplo de CRF que puede utilizarse en sistemas grandes y complejos. Para pequeños proyectos, recomiendo que las peticiones de cambios sean formalmente registradas, pero el CRF debe centrarse más en la descripción del cambio requerido y menos en la implementación. El ingeniero que hace el cambio decide cómo implementarlo en estas situaciones.

Solicitar cambios completando un formulario de solicitud de cambios

Analizar la solicitud de cambios

if cambio es válido **then**

 Evaluar cómo implementar el cambio

 Evaluar los costos del cambio

 Registrar la petición del cambio en una base de datos

 Remitir la petición a la oficina de control de cambios

if cambio es aceptado **then**

repeat

 Hacer cambios al software

 Registrar cambios y vincularlos a la petición de cambios asociada

 Remitir el software cambiado para aprobar la calidad

until calidad del software sea adecuada

 Crear nueva versión del sistema

else

 Rechazar petición de cambios

else

 Rechazar petición de cambios

Figura 29.3
El proceso de
gestión de cambios.

Una vez emitido el formulario, se registra en la base de datos de configuraciones. Se analiza la petición de cambios para comprobar que el cambio solicitado es necesario. Algunas peticiones de cambio se deben a los malentendidos más que a los fallos del sistema y, por lo tanto, no es necesario modificar el sistema. Otras se refieren a fallos ya conocidos. Si el análisis descubre que el cambio solicitado es inválido, está duplicado o ya ha sido considerado, el cambio se rechaza. La razón del rechazo se devuelve a la persona que emitió la solicitud del cambio.

Para cambios válidos, la siguiente etapa del proceso es evaluar y asignar costes al cambio. Se comprueba el impacto del cambio en el resto del sistema. Se lleva a cabo un análisis técnico de cómo implementar el cambio. Esto implica identificar todos los componentes afectados por el cambio utilizando la base de datos de configuraciones y el código fuente. Si hacer los cambios implica hacer más cambios en otros lugares del sistema, se incrementarán los costes de implementación del cambio. A continuación, se valoraran los cambios requeridos en los módulos del sistema. Finalmente, se estima el coste del cambio, teniendo en cuenta los costes de modificar estos componentes.

El consejo de control de cambios (CCB) revisa y aprueba todas las peticiones de cambios a menos que el cambio simplemente comprenda la corrección de errores menores en los despliegues de la pantalla, las páginas web o en los documentos. El CCB considera el impacto del cambio desde el punto de vista estratégico y organizacional más que desde el punto de vista técnico. Decide si el cambio se justifica económica y si existen buenas razones organizacionales para aceptarlo.

La denominación «consejo de control de cambios» implica que un grupo toma las decisiones del cambio. Los proyectos militares requieren consejos de control de cambios estructurados formalmente, que incluyen a los clientes y los proveedores. Sin embargo, para proyectos pequeños o de medio tamaño, el consejo de control de cambios simplemente se compone de un gestor de proyectos más uno o dos ingenieros que no están directamente involucrados en el desarrollo del software. En algunos casos, existe sólo un revisor del cambio que aconseja si los cambios son justificables.

Formulario de petición de cambios	
Proyecto: Proteus/PCL-Tools	Número: 23/03
Solicitante del cambio: I. Sommerville	Fecha: 1/12/02
Cambio solicitado: Cuando un componente se seleccione de una estructura, desplegar el nombre del archivo donde se almacena.	
Analizador del cambio: G. Dean	Fecha de análisis: 10/12/02
Componentes afectados: Display-Icon.Select, Display-Icon.Display	
Componentes asociados: FileTable	
Evaluación del cambio: Relativamente fácil de implementar puesto que se dispone de una tabla de los nombres de los archivos. Requiere el diseño e implementación de un campo de despliegue. No se requieren cambios a los componentes asociados.	
Prioridad del cambio: Baja	
Implementación del cambio:	
Esfuerzo estimado: 0,5 días	
Fecha para CCB: 15/12/02	Fecha de decisión del CCB: 1/2/03
Decisión del CCB: Aceptar cambio. Cambio a implementar en versión 2.1.	
Implementador del cambio:	
Fecha de remisión para QA:	Fecha del cambio:
Fecha de remisión a CM:	Decisión de QA:
Comentarios:	

Figura 29.4
Un formulario de petición de cambios llenado parcialmente.

La gestión de cambios para sistemas estándar de software debe ser manejada de una forma ligeramente diferente a los sistemas hechos a medida. En estos sistemas estándar, los clientes no están directamente implicados, por lo que un cambio en el negocio del cliente no nos afecta. Las peticiones de cambio están generalmente asociadas con errores en el sistema que han sido descubiertos durante las pruebas del sistema o por nuestros clientes una vez que ha sido entregado el mismo. Los clientes pueden enviar los informes de error a través de una página web o por e-mail. Un equipo de gestión de errores comprobará que los errores remitidos son válidos y los traducirá a peticiones de cambio formales del sistema. Los cambios tienen que ser priorizados para su implementación y los errores no pueden ser reparados si los costes de reparación son muy elevados.

Cuando se crean las nuevas versiones del sistema por medio de una construcción diaria, se utiliza un proceso de gestión del cambio sencillo. Los problemas en los cambios aún deben registrarse, pero los cambios que sólo afectan a los componentes y módulos individuales no necesitan ser evaluados de forma independiente. Se pasan directamente al desarrollador del sistema. Éste los acepta o indica por qué ya no se requieren. Los cambios que afectan a los módulos del sistema producidos por diferentes equipos de desarrollo son evaluados por alguna autoridad de control del cambio quien decide si se implementan.

En algunos métodos ágiles, como programación extrema, los clientes son directamente implicados en decidir cuándo un cambio debe ser implementado. Cuando ellos proponen un cambio en los requerimientos del sistema, trabajan con el equipo en evaluar el impacto del cambio y deciden cuándo deben tener prioridad respecto a los planes establecidos para el próximo incremento del sistema. Sin embargo, los cambios que atañen a mejoras del software se dejan a discreción de los programadores que trabajan en el sistema. La reconstrucción, donde el software es mejorado continuamente, no se ve como una sobrecarga, sino como una parte necesaria del proceso de desarrollo.

Conforme se cambian los componentes del software, se da mantenimiento al registro de los cambios realizados a cada componente. A veces éstos se denominan *historial* del componente. La mejor forma de dar mantenimiento a tales registros es por medio de una cabecera estandarizada, ubicada al inicio del componente (véase la Figura 29.5). Éste se refiere a la solicitud de cambio asociada como el cambio del software. Pueden escribirse scripts sencillos que analicen todos los componentes y procesen los historiales para generar informes de cambios para los diversos componentes. Para las páginas web se puede utilizar una aproximación similar. Para documentos publicados, los registros de cambios de cada versión son generalmente mantenidos en una portada del documento.

// Proyecto BANKSEC (IST 6087)				
//				
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE				
//				
// Objeto: Regla Actual				
// Autor: N. Perwaiz				
// Fecha de creación: 10 de noviembre de 2002				
//				
// (c)Lancaster University 2002				
//				
// Historial de modificaciones				
// Versión	Modificador	Fecha	Cambio	Razón
// 1.0	J. Jones	1/12/2002	Agregar encabezado	Remitido CM
// 1.1	N. Perwaiz	9/04/2003	Nuevo campo	Petición R07/02

Figura 29.5
Información
de la cabecera
del componente.

29.3 Gestión de versiones y entregas

La gestión de las versiones y entregas es el proceso de identificar y mantener los registros de las diversas versiones y entregas de un sistema. Los gestores de las versiones diseñan procedimientos para asegurar que las diversas versiones de un sistema se puedan recuperar cuando se requieran y que no se cambien de forma accidental por parte del equipo de desarrollo. También trabajan con los responsables de marketing y con los clientes de sistemas personalizados, en planificar las entregas y distribuciones.

Una versión de un sistema es una instancia de un sistema que difiere, de alguna manera, de otras instancias. Las nuevas versiones de un sistema tienen diferente funcionalidad, mejor rendimiento o incorporan reparaciones de los fallos del sistema. Algunas versiones son funcionalmente equivalentes pero diseñadas para diferentes configuraciones de hardware y software. Si sólo existen pequeñas diferencias entre las versiones, éstas se denominan *variantes*.

Una entrega de un sistema es una versión que se distribuye a los clientes. Cada entrega incluye nueva funcionalidad o está concebida para diferentes plataformas de hardware. Siempre existen más versiones de un sistema que las entregas, puesto que las versiones se crean dentro de una organización para el desarrollo interno o pruebas y nunca se entregan a los clientes.

En la actualidad, la gestión de versiones se apoya siempre en herramientas CASE, como se explica en la Sección 29.5. Estas herramientas administran el almacenamiento de cada versión del sistema y controlan el acceso a los componentes del sistema. Se apoyan en el sistema para llevar a cabo las ediciones. Cuando los componentes se reintroducen en el sistema, se crea una nueva versión y el sistema de gestión de versiones le asigna un identificador. A pesar de que las herramientas difieren obviamente de funcionalidades y de interfaz, la base de todas estas herramientas de soporte es la gestión de versiones.

29.3.1 Identificación de versiones

Para crear una versión particular del sistema, se tienen que especificar las versiones de cada uno de los componentes que deben incluirse en él. Dentro de un sistema de software grande, existen cientos de componentes de software, cada uno de los cuales tiene varias versiones diferentes. Debe definirse una forma no ambigua de identificar cada versión de los componentes para asegurar que se incluyen los componentes adecuados en el sistema. Sin embargo, no se puede utilizar el nombre del elemento de configuración para identificar las versiones debido a que hay diferentes versiones para cada elemento de configuración.

Existen tres técnicas básicas utilizadas para la identificación de componentes:

1. *Numeración de las versiones*. Al componente se le asigna un número de versión explícito y único. Éste es el esquema de identificación más utilizado.
2. *Identificación basada en atributos*. Cada componente tiene un nombre (como el nombre utilizado para nombrar los elementos de configuración, que no es único a lo largo de las versiones) y un conjunto asociado de atributos para cada versión del componente (Estublier y Casallas, 1994). Por lo tanto, los componentes se identifican por su nombre y por los valores de los atributos.
3. *Identificación orientada al cambio*. Cada sistema se nombra a partir de los atributos, pero también se asocia con una o más solicitudes de cambios (Munch *et al.*, 1993).

Cada versión del documento es creada en respuesta a una o más peticiones de cambios. La versión del sistema se identifica por el conjunto de los cambios implementados en el componente.

Numeración de versiones

En un esquema sencillo de numeración de versiones, al nombre del componente o del sistema se le añade un número de versión. Por lo tanto, se puede hablar de Solaris 4.3 (versión 4.3 del sistema Solaris) y la versión 1.4 del componente getToken. La primera versión se denomina 1.0, las subsiguientes versiones son 1.1, 1.2 y así sucesivamente. En alguna etapa, se entrega una nueva versión (versión 2.0) y el proceso comienza otra vez en la versión 2.1. El esquema es lineal y se basa en la suposición de que las versiones del sistema se crean en secuencia. Muchas herramientas de gestión de versiones (véase la Sección 29.5) como RCS (Tichy, 1985) y CVS (Berliner, 1990) permiten esta forma de identificación de versiones.

En la Figura 29.6 se muestra este enfoque y la derivación de algunas versiones diferentes del sistema a partir de versiones previas. Las flechas en este diagrama apuntan desde la versión fuente hasta una nueva versión del sistema creada a partir de la fuente. Observe que la derivación de versiones no es necesariamente lineal y las versiones con números de versión consecutivos se producen a partir de diferentes líneas base. Esto se muestra en la Figura 29.6, donde la versión 2.2 se crea a partir de la versión 1.2 en lugar de la versión 2.1. En principio, cualquier versión existente se puede utilizar como un punto de inicio para una nueva versión del sistema.

Este esquema es sencillo pero requiere una buena gestión de la información para llevar a cabo los registros de las diferentes versiones y las relaciones entre los cambios propuestos y versiones del sistema. Por ejemplo, las versiones 1.1 y 1.2 de un sistema pueden diferir en que la versión 1.2 ha utilizado una librería gráfica diferente. En consecuencia, se necesitará mantener registros en la base de datos de configuraciones que describan cada versión y por qué ha sido producida. Puede necesitarse vincular explícitamente la petición de cambios con las diferentes versiones de cada componente.

Identificación basada en atributos

Un problema fundamental con los esquemas explícitos de asignación de nombres de las versiones es que no reflejan los muchos atributos diferentes utilizados para identificar las versiones. Ejemplos de estos atributos que permiten la identificación son:

- El cliente
- El lenguaje de desarrollo

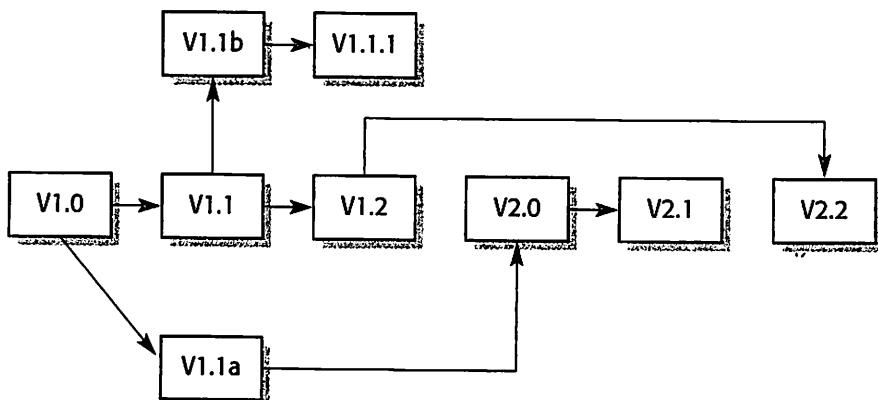


Figura 29.6
Estructura
de derivación
de versiones.

- El estado del desarrollo
- La plataforma de hardware
- La fecha de creación

Si cada versión es identificada por un conjunto único de atributos, es fácil agregar nuevas versiones derivadas a partir de cualquiera de las versiones existentes. Éstas se identifican utilizando un conjunto único de valores de los atributos. Comparten muchos de estos valores con sus versiones padre, por lo que se conserva la relación entre las versiones. Las versiones se recuperan especificando los valores de los atributos requeridos. Las funciones de los atributos permiten consultas como «la versión creada más recientemente», «la versión creada en fechas dadas», etcétera. Por ejemplo, la versión del sistema de software AC3D desarrollada en Java para Windows XP en enero de 2003 se identifica como:

AC3D (lenguaje = Java, plataforma = XP, fecha = Ene2003)

Usando la especificación general de los componentes en AC3D, la herramienta de gestión de versiones selecciona los componentes que tienen los atributos «Java», «XP» y «Ene2003».

La identificación basada en atributos se implementa directamente por el sistema de gestión de versiones, utilizando los atributos de los componentes mantenidos en la base de datos del sistema. Alternativamente, se puede implementar un sistema de identificación de atributos como una capa superior de un esquema de numeración de versiones oculto. La base de datos de configuraciones mantiene los vínculos entre los atributos de identificación y el sistema subyacente y las versiones del componente.

Identificación orientada al cambio

La identificación basada en atributos de las versiones del sistema elimina algunos de los problemas de la recuperación de versiones encontrados en los esquemas sencillos de numeración. Sin embargo, para recuperar una versión, se tienen que conocer los atributos asociados. Más aún, se necesita utilizar un sistema de gestión de cambios independiente para descubrir las relaciones entre las versiones y los cambios.

La identificación orientada al cambio se utiliza más para identificar los sistemas que los componentes. Las versiones de los componentes individuales están ocultas a los usuarios del sistema de la CM. Cada cambio del sistema propuesto tiene un conjunto de cambios asociado que describe los cambios realizados a los diferentes componentes del sistema para implementar este cambio. Los conjuntos de cambios se aplican en secuencia, por lo que, al menos en principio, se puede crear una versión del sistema que incorpore cualquier conjunto arbitrario de cambios. Por ejemplo, los cambios incorporados para adaptar el sistema a Linux en lugar de Solaris, seguido de los cambios necesarios para incorporar un nuevo sistema de base de datos. De igual forma, a los cambios Linux/Solaris pueden seguirlos las modificaciones para traducir la interfaz de usuario de inglés a italiano.

En la práctica, por supuesto, no es posible aplicar conjuntos arbitrarios de cambios a un sistema. Los diversos conjuntos de cambios pueden ser incompatibles, por lo que aplicar un conjunto de cambios A seguido de un conjunto de cambios D puede crear un sistema no válido. Más aún, los conjuntos de cambios pueden entrar en pugna debido a que los diversos cambios afectan al mismo código del sistema. Para afrontar estas dificultades, las herramientas de gestión de versiones que apoyan la identificación orientada a cambios permiten especificar las reglas de consistencia que delimitan las formas de combinar los conjuntos de cambios.

29.3.2 Gestión de entregas

Una entrega del sistema es una versión del sistema que se distribuye a los clientes. Los gestores de entregas del sistema son los responsables de decidir cuándo se entrega el sistema, de gestionar el proceso de creación de las entregas y de los medios de distribución y documentación de la entrega para asegurar que se puedan recuperar de la misma forma en que se distribuyeron, en caso necesario.

Una entrega del sistema no es sólo el código ejecutable del sistema. Las entregas también incluyen:

1. *Archivos de configuración*, que definen cómo configurar el sistema para instalaciones específicas.
2. *Los archivos de datos* necesarios para el funcionamiento del sistema.
3. *El programa de instalación* utilizado para ayudar a instalar el sistema en el hardware destino.
4. *La documentación electrónica y en papel* que describe al sistema.
5. *El embalaje y la publicidad asociados* diseñados para esta entrega.

Los administradores de las entregas no pueden suponer que los clientes siempre instalarán las nuevas versiones del sistema. Algunos usuarios del sistema están a gusto con una versión existente del sistema. Consideran que no vale la pena gastar en cambiar a una nueva entrega. Por lo tanto, las nuevas entregas del sistema no pueden depender de la existencia de entregas previas. Consideremos el siguiente escenario:

1. La entrega 1 de un sistema se distribuye y se pone en funcionamiento.
2. La entrega 2 requiere la instalación de nuevos archivos de datos, pero algunos clientes no necesitan los recursos de la entrega 2, por lo que conservan la entrega 1.
3. La entrega 3 requiere los archivos de datos instalados en la entrega 2 y no agrega nuevos archivos de datos.

El distribuidor de software no puede suponer que los archivos requeridos para la entrega 3 se han instalado en todos los lugares. Algunos sitios van directamente de la entrega 1 a la entrega 3, saltándose la entrega 2. Algunos sitios pueden haber modificado los archivos de datos asociados con la versión 2 para adaptarlos a circunstancias locales. Por lo tanto, los archivos de datos deben ser distribuidos e instalados con la versión 3 del sistema.

Toma de decisiones de la entrega

Preparar y distribuir una entrega del sistema es un proceso costoso, particularmente para los productos de software de mercados en masa. Si las entregas son muy frecuentes, los clientes pueden no actualizarse a las nuevas, especialmente si no son gratuitas. Si las entregas son infrecuentes, se puede perder cuota de mercado puesto que los clientes consideran sistemas alternativos. Esto, por supuesto, no es aplicable al software desarrollado específicamente para una organización. Sin embargo, para este tipo de software, las entregas infrecuentes significan un crecimiento divergente entre el software y los procesos de negocios que pretenden apoyar.

Las decisiones sobre cuándo entregar una nueva versión del sistema están dirigidas por varios factores técnicos y organizacionales, como se muestra en la Figura 29.7.

Factor	Descripción
Calidad técnica	Si se reportan fallos que afecten en la forma en la que los clientes utilizan el sistema, es necesario emitir una versión para reparar el fallo. Sin embargo, los fallos menores del sistema se reparan mediante parches (a menudo distribuidos por Internet) que se aplican a las entregas actuales del sistema.
Cambios en la plataforma	Usted puede tener que crear nuevas entregas de una aplicación software cuando aparece una nueva versión del sistema operativo.
Quinta ley de Lehman (véase el Capítulo 21)	Ésta sugiere que el incremento de la funcionalidad incluida en cada versión sea aproximadamente constante. Por lo tanto, si existe una versión del sistema con funcionalidades completamente nuevas, ésta debe seguirse de una entrega de reparación.
Competencia	Se necesita una nueva versión del sistema cuando está disponible el producto competidor.
Requerimientos de marketing	El departamento de marketing de una organización puede estar interesado en tener lista la entrega en una fecha particular.
Propuestas de cambios el cliente	Para sistemas personalizados, los clientes hacen un pago por un conjunto específico de cambios en el sistema y esperan a que el sistema se entregue tan pronto como estos cambios sean implementados.

Figura 29.7
que influyen en la estrategia de entregas.

Creación de la entrega

La creación de las entregas es el proceso de crear una colección de archivos y documentación que incluyen todos los componentes de la entrega del sistema. El código ejecutable de los programas y todos los archivos de datos asociados se recogen e identifican. Se describen las configuraciones diferentes para hardware y sistemas operativos y las instrucciones para los clientes que necesiten configurar sus propios sistemas. Si se entregarán manuales, las copias electrónicas deben almacenarse con el software. Se deben escribir las guías para la instalación. Finalmente, cuando toda la información está disponible, se prepara un disco de entrega para la distribución.

Actualmente, los discos ópticos (CD-ROM y DVD), que almacenan desde 600 MBytes hasta 4 GBytes, son el medio de distribución normal para la entrega del sistema. Adicionalmente, muchos productos de software también se entregan a través de Internet. Sin embargo, muchas personas, encuentran muy largo el tiempo de descarga de archivos y prefieren la distribución en CD-ROM.

La distribución de las nuevas entregas tiene asociados unos altos costes de marketing y de empaquetado, por lo que los vendedores crean usualmente nuevas entregas para nuevas plataformas o cuando se añaden funcionalidades nuevas significativamente. Ellos cobran a los usuarios por este software nuevo. Cuando se descubren problemas en una entrega, los vendedores usualmente hacen parches, que se pueden descargar vía web, para reparar el software existente.

Además de los costes de encontrar y descargar la nueva entrega, el problema es que muchos clientes nunca descubren la existencia de estos parches o no tienen los conocimientos técnicos para instalarlos. En lugar de hacer esto, pueden continuar utilizando el existente, faltando el sistema con los consiguientes riesgos para sus negocios. En algunas situaciones, en que el parche es diseñado para reparar fallos de seguridad, el riesgo de no hacer la instalación del parche puede significar que el negocio sea susceptible de ataques externos.

Documentación de las entregas

Cuando se produce la entrega de un sistema, debe estar documentada para asegurar que se puede reconstruir exactamente en el futuro. Esto es particularmente importante para los sistemas personalizados de larga vida. Los clientes utilizan una sola entrega de estos sistemas por muchos años y requieren cambios específicos para una entrega particular del software mucho después de la fecha de entrega original.

Para documentar una entrega, se tienen que registrar las versiones específicas de los componentes del código fuente utilizados para crear el código ejecutable. También se deben mantener copias del código fuente y ejecutable y todos los archivos de datos y de configuración. También se deben registrar las versiones del sistema operativo, las bibliotecas, los compiladores y otras herramientas utilizadas para construir el sistema. Éstos pueden requerirse para construir exactamente el mismo sistema en alguna fecha posterior. En estos casos, las copias de las plataformas de software y las herramientas también se almacenan en un sistema de gestión de versiones.

29.4 Construcción del sistema

La construcción del sistema es el proceso de compilar y vincular los componentes del software en un programa que se ejecuta en una configuración particular. Cuando se construye un sistema a partir de sus componentes, se tienen que hacer las siguientes preguntas:

1. ¿Todos componentes de un sistema se incluyen en las instrucciones de la construcción?
2. ¿La versión apropiada de cada componente se incluye en las instrucciones de la construcción?
3. ¿Están disponibles todos los archivos de datos requeridos?
4. Si los archivos de datos están asociados a componentes, ¿el nombre que se utiliza es el mismo que el nombre de los archivos de datos en la máquina donde se ejecutará el software?
5. ¿Están disponibles las versiones adecuadas del compilador y otras herramientas requeridas? Las versiones actuales de las herramientas de software pueden ser incompatibles con las versiones anteriores utilizadas para desarrollar el sistema.

Hoy en día, las herramientas de la CM se utilizan para automatizar el proceso de construcción del sistema. El equipo de la CM escribe una secuencia de comandos (script) que define las dependencias entre los diferentes componentes del sistema. También especifica las herramientas utilizadas para compilar y vincular los componentes del sistema. La herramienta de construcción del sistema interpreta dicha secuencia de comandos y llama a otros programas requeridos para construir el sistema ejecutable. Esto se ilustra en la Figura 29.8. En algunos entornos de programación (como los entornos de desarrollo de Java), el script para construir el sistema se crea automáticamente analizando el código fuente y estableciendo los componentes utilizados. Por supuesto, en esta situación, el nombre de los componentes almacenados tiene que ser el mismo nombre que el componente del programa.

Las dependencias entre los componentes se especifican en la secuencia de comandos de construcción, por lo que el sistema de construcción decide cuándo los componentes deben recompilarse y cuándo se puede reutilizar el código objeto existente. En muchas herramientas,

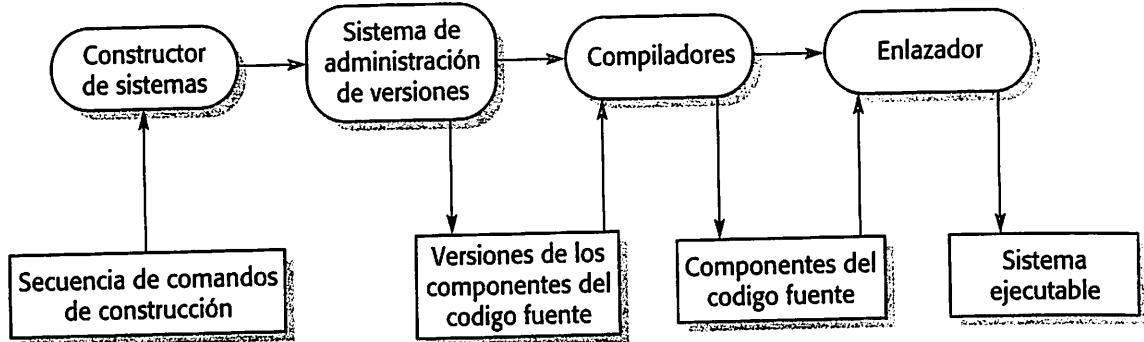


Figura 29.8 Construcción del sistema.

las dependencias de la secuencia de comandos de construcción se especifican como dependencias entre los archivos en los cuales los componentes de código fuente se almacenan. Sin embargo, cuando existen múltiples archivos de código fuente representando a las diferentes versiones de los componentes, algunas veces no se ve claramente qué archivos fuente se utilizaron para derivar los componentes del código objeto. Esta confusión aparece cuando la correspondencia entre los archivos de código fuente y objeto comparten el mismo nombre pero diferente sufijo (por ejemplo, .c y .o). La única forma de evitar este problema es que las herramientas de gestión de versiones y de construcción del sistema estén integradas.

29.5 Herramientas CASE para gestión de configuraciones

Los procesos de gestión de configuraciones por lo general están estandarizados e involucran la aplicación de procedimientos predefinidos. Requieren la gestión cuidadosa de grandes cantidades de datos, y la atención a los detalles es esencial. Cuando se construye un sistema a partir de las versiones de los componentes, un simple error en la gestión de configuraciones puede implicar que el software no trabaje de forma adecuada. En consecuencia, las herramientas CASE de apoyo son esenciales para la gestión de configuraciones, y desde los 70 se ha producido un gran número de diferentes herramientas que abordan diferentes áreas de la gestión de configuraciones.

Estas herramientas pueden ser combinadas para crear entornos de trabajo de gestión de configuraciones que soporten todas las actividades CM. Hay dos tipos de entornos de trabajo CM:

1. *Entornos de trabajo abiertos.* Las herramientas para cada etapa del proceso CM son integradas de acuerdo con procedimientos organizacionales estándar. Hay muchas herramientas CM comerciales y open-source disponibles para propósitos específicos. La gestión de cambios puede llevarse a cabo con herramientas de seguimiento (bug-tracking) como Bugzilla, la gestión de versiones a través de herramientas como RCS (Tichy, 1985) o CVS (Berliner, 1990), y la construcción del sistema con herramientas como Make (Feldman, 1979; Oram y Talbott, 1991) o Imake (Dubois, 1996). Todas estas herramientas son open-source y están disponibles de forma gratuita.
2. *Entornos integrados.* Estos entornos ofrecen facilidades integradas para gestión de versiones, construcción del sistema o seguimiento de los cambios. Por ejemplo, el proceso de gestión de Cambios Unificado de Rational se basa en un entorno CM integrado que incorpora ClearCase (White, 2000) para la construcción y gestión de versiones.

del sistema y ClearQuest para el seguimiento de los cambios. Las ventajas de los entornos CM integrados es que el intercambio de datos es sencillo, y que el entorno incluye una base de datos CM integrada. Los entornos integrados SCM provienen de sistemas antiguos como Lifespan (Whitgift, 1991) para gestión de cambios y DSEE (Leblang y Chase, 1987) para gestión de versiones y construcción del sistema. Sin embargo, los entornos integrados CM son complejos y costosos, y muchas organizaciones prefieren utilizar herramientas individuales más simples y económicas.

Muchos sistemas grandes son desarrollados desde diferentes lugares, y necesitan herramientas SCM que soporten el trabajo desde múltiples lugares con múltiples almacenes de datos para los elementos de configuración. Mientras muchas herramientas SCM están diseñadas para trabajar desde un único lugar, otras como CVS facilitan el trabajo desde múltiples sitios.

29.5.1 Apoyo a la gestión de cambios

Cada persona involucrada en el proceso de gestión de cambios es responsable de alguna actividad. Una vez que completa esta actividad pasa los formularios y elementos de configuración asociados a otra persona. La naturaleza de procedimiento de este proceso significa que un cambio en el modelo del proceso se diseña e integra con un sistema de gestión de versiones. Entonces este modelo se interpreta para que los documentos correctos se pasen a la persona indicada en el momento justo.

Hay varias herramientas de gestión de cambios disponibles, desde herramientas relativamente simples, open-source como Bugzilla hasta sistemas absolutamente integrados como Racional ClearQuest. Estas herramientas proveen alguna o todas de las siguientes características de apoyo al proceso:

1. Un editor de formularios que permite cambiar los formularios propuestos a crear y llenar por las personas que hacen las peticiones de cambios.
2. Un sistema de flujo de trabajo que permiten al equipo de la CM especificar las personas que deben procesar las solicitudes de peticiones de cambio y el orden del procesamiento. Este sistema también pasa de forma automática los formularios a las personas indicadas en el momento justo e informa a los miembros relevantes del equipo del progreso del cambio. El correo electrónico se utiliza para suministrar actualizaciones del progreso a aquellos involucrados en el proceso.
3. Una base de datos de cambios que se utiliza para gestionar todas las propuestas de cambios y que puede vincularse al sistema de gestión de versiones. Por lo general, se proveen las características de consulta que permiten al equipo de la CM encontrar propuestas específicas de cambios.
4. Un sistema de gestión de informes de cambios que genera informes sobre el estado de la peticiones de cambio recibidas.

29.5.2 Soporte para gestión de versiones

La gestión de versiones implica gestionar grandes cantidades de información para asegurar que los cambios en el sistema se registren y controlen. Las herramientas de gestión de versiones controlan un repositorio de elementos de configuración donde el contenido de ese repositorio es inmutable (no cambia). Para trabajar sobre un elemento de la configuración, debe extraerse del re-

positorio y colocarlo en un directorio de trabajo. Después de hacer los cambios en el software, se introducirá de nuevo en el repositorio, creándose automáticamente una nueva versión. Todos los sistemas de gestión de versiones proveen un conjunto básico de capacidades comparables aunque algunos son más sofisticados que otros. Ejemplos de estas capacidades son:

1. *Identificación de versiones y entregas.* A las versiones gestionadas se les asignan identificadores cuando se introducen en el sistema. Varios sistemas permiten los diferentes tipos de identificación de versiones tratados en la Sección 29.3.1.
2. *Gestión del almacenamiento.* Para reducir el espacio de almacenamiento requerido por las diferentes versiones, los sistemas de gestión de versiones proveen las características de gestión del almacenamiento donde las versiones se describen por sus diferencias a partir de alguna versión maestra. Las diferencias entre las versiones se representan con una *delta* que encapsula las instrucciones requeridas para reconstruir la versión asociada del sistema. Esto se ilustra en la Figura 29.9, que muestra cómo se aplican las deltas inversamente a la última versión de un sistema para reconstruir las versiones anteriores.
3. *Registro del historial del cambio.* Todos los cambios realizados al código de un sistema o componente se registran y son listados. En algunos sistemas, estos cambios se utilizan para seleccionar una versión particular del sistema.
4. *Desarrollo independiente.* Las diferentes versiones de un sistema se pueden desarrollar en paralelo y cada versión cambia de forma independiente. Por ejemplo, la entrega 1 se modifica después del desarrollo de la entrega 2 agregando nuevas deltas de nivel 1. El sistema de gestión de versiones mantiene un registro de los componentes que han sido extraídos para su edición y asegura que no interfieran los cambios que diferentes desarrolladores hayan hecho a los mismos componentes. Algunos sistemas sólo permiten que se extraiga una instancia de un componente para su edición; otros resuelven los conflictos potenciales cuando los componentes editados se reintroducen en el sistema.
5. *Apoyo al proyecto.* El sistema puede soportar múltiples proyectos así como múltiples archivos. En sistemas de apoyo a proyectos, como CVS, es posible introducir y extraer todos los ficheros asociados a un proyecto en lugar de tener que trabajar con un solo fichero a la vez.

29.5.3 Apoyo a la construcción del sistema

La construcción de sistemas es un proceso intensivo de computación. Compilar y vincular todos los componentes de un sistema grande puede llevar varias horas. Puede haber cientos de archivos involucrados con la consecuente posibilidad de errores humanos si éstos se compilan y vinculan de forma manual. Las herramientas de construcción de sistemas automatizan

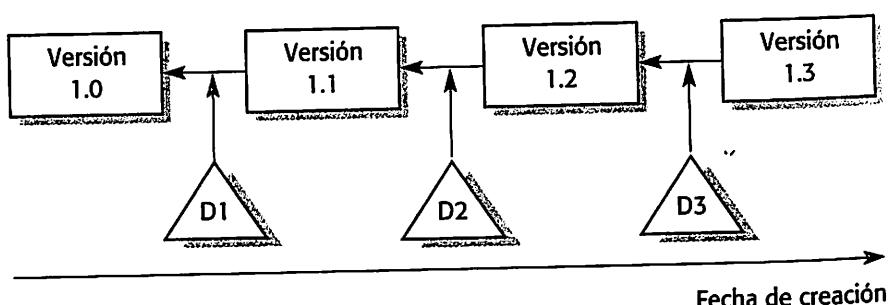


Figura 29.9
Versiones basadas en delta.

el proceso de construcción para reducir los errores humanos potenciales y, donde sea posible, disminuir el tiempo requerido para la construcción del sistema.

Las herramientas de construcción del sistema pueden ser independientes como las derivadas de la utilidad MAKE de Unix (Oran y Talbott, 1991) o integradas con las herramientas de gestión de versiones. Las características suministradas por las herramientas CASE de construcción de sistemas son:

1. *Una dependencia del lenguaje de especificación o del intérprete asociado.* Las dependencias de los componentes se describen y se disminuye la recompilación. Esto se explica más adelante en esta sección.
2. *Selección de herramientas y apoyo a la instancia.* Se especifican e instancian, como se requiera, los compiladores y otras herramientas de procesamiento utilizadas para procesar los archivos de código fuente.
3. *Compilación distribuida.* Algunos constructores de sistemas, especialmente aquellos que son parte de los sistemas integrados de la CM, permiten la compilación distribuida en redes. En lugar de que todas las compilaciones se lleven a cabo en una sola máquina, los constructores de sistemas buscan los procesadores desocupados sobre la red y seleccionan varios compiladores en paralelo. Esto reduce notablemente el tiempo requerido para construir un sistema.
4. *Gestión de los objetos derivados.* Los objetos derivados son objetos que se crean a partir de otros objetos fuente. La gestión de los objetos derivados vincula el código fuente y los objetos derivados y sólo vuelve a derivar un objeto cuando se requiere por los cambios del código fuente.

Gestionar los objetos derivados y disminuir la recompilación se explica mejor utilizando un ejemplo sencillo. Consideremos una situación donde un programa denominado **comp** se crea a partir de los módulos objeto **scan.o**, **syn.o**, **sem.o** y **cgen.o**. Para cada módulo objeto, existe un módulo del código fuente (**scan.c**, **syn.c**, **sem.c** y **cgen.c**). Un archivo de declaraciones denominado **defs.h** es compartido por **scan.c**, **syn.c** y **sem.c** (véase la Figura 29.10). En la Figura 29.10 las flechas significan «depende de». Por lo tanto, **comp** depende de **scan.o**, **syn.o**, **sem.o** y **cgen.o**, **scan.o** depende de **scan.c**, etcétera.

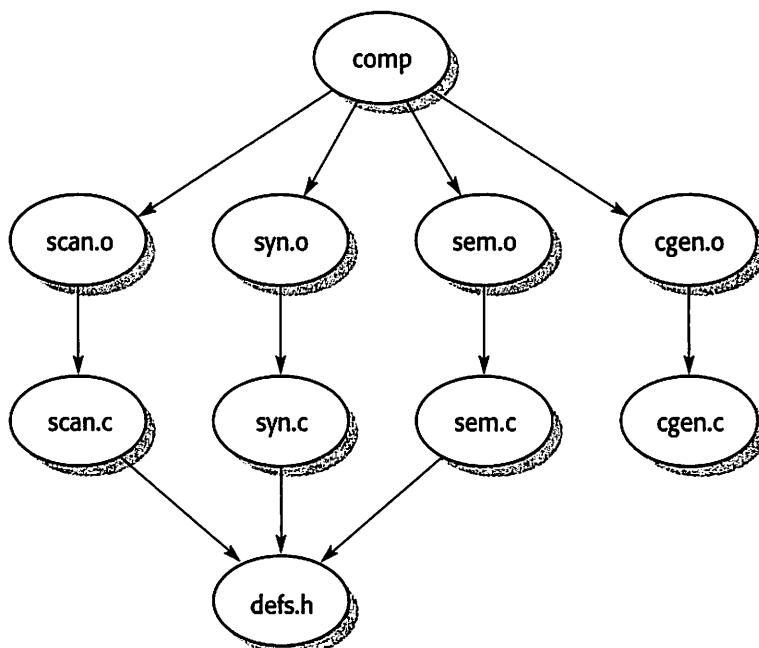


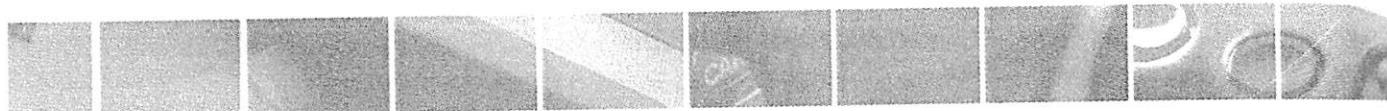
Figura 29.10
Dependencias entre componentes.

Si `scan.c` cambia, la herramienta de construcción del sistema puede detectar que el objeto derivado `scan.o` debe reconstruirse e llamar al compilador de C para compilar `scan.c` con el fin de crear un nuevo objeto derivado, `scan.o`.

Después la herramienta de construcción utiliza el vínculo de dependencia entre `comp` y `scan.o` para detectar que `comp` debe reconstruirse vinculando `scan.o`, `syn.o`, `sem.o` y `cgen.o`. El sistema puede detectar que los otros componentes del código objeto no cambiaron, por lo que no es necesaria la recompilación del código fuente asociado.

Algunos constructores de sistemas utilizan el archivo de la fecha de modificación como atributo clave para decidir si se requiere la recompilación. Si un archivo de código fuente se modifica después de su correspondiente archivo de código objeto, entonces este último se reconstruye. Esto normalmente significa que existe un objeto derivado sólo para la versión del código fuente más reciente. Cuando se crea una nueva versión del código objeto la versión anterior de éste se pierde.

Otros sistemas utilizan un enfoque más sofisticado para derivar la gestión de los objetos. Agregan objetos derivados al identificador de la versión del código fuente utilizado para generar estos objetos. Dentro de los límites de la capacidad de almacenamiento, conservan todos los objetos derivados. Por lo tanto, generalmente es posible recuperar el código objeto de todas las versiones de los componentes del código fuente sin recompilación.



PUNTOS CLAVE

- La gestión de configuraciones es la gestión de los cambios en el sistema. Cuando se mantiene un sistema, el papel del equipo de CM es asegurar que los cambios se incorporen de forma controlada.
- En un proyecto grande, se establece y utiliza un esquema formal de nombres de documentos como una base para mantener el registro de las diferentes versiones de todos los documentos del proyecto.
- El equipo de CM se apoya en una base de datos de configuraciones que registra la información de los cambios en el sistema y de las peticiones de cambio pendientes. Los proyectos deben tener algún medio formal de petición de cambios al sistema.
- Cuando se fija un esquema de gestión de configuraciones, se establece un esquema consistente de identificación de versiones. Éstas se identifican por el número de versión, por un conjunto de atributos o por los cambios propuestos e implementados en el sistema.
- Las entregas de los sistemas incluyen el código ejecutable, los archivos de datos, los archivos de configuración y la documentación. La gestión de las entregas comprende tomar decisiones sobre cuándo entregar un sistema, preparar toda la información para la distribución y la documentación de cada entrega del sistema.
- La construcción de sistemas es el proceso de ensamblar los componentes del sistema en un programa ejecutable para que opere en un sistema informático específico.
- Existen herramientas CASE para apoyar las actividades de gestión de la configuración. Éstas comprenden herramientas como CVS para gestionar las versiones del sistema, herramientas para la gestión del cambio y herramientas para la construcción de sistemas.
- Las herramientas CASE para la CM pueden ser herramientas autónomas que permiten gestionar el cambio, gestionar las versiones y la construcción de sistemas o pueden ser sistemas integrados que proveen una sola interfaz para las tareas de CM.