

Tema 5.

Gestión de la Configuración del Software

- 4 Definición
- 4 Identificación
- 4 Control de versiones
- 4 Control de cambios
- 4 Auditorías e informes
- 4 Construcción del sistema

Bibliografía

Captítulo 29. Gestión de configuraciones. Software Engineering Sommerville 7ª edición.

Capítulo 9. Gestión de configuraciones. Ingeniería del software. 5ª edición. Roger S. Pressman.

Definición

Actividad de autoprotección para:

- ❑ Identificar el cambio
- ❑ Controlar el cambio
- ❑ Garantizar que el cambio se implementa adecuadamente
- ❑ Informar del cambio a todos aquellos que les interese

Gestión de configuraciones \neq Mantenimiento

Planificación de la GC

- ❑ Definición de entidades a gestionar y esquema formal de identificación
- ❑ Identificación de responsables de los procedimientos de la GC
- ❑ Descripción de cómo los registros de documentos del proceso de GC deberían ser mantenidas
- ❑ Descripción de las herramientas usadas
- ❑ Definición de la base de datos de configuración (BDC) usada para almacenar la información

Identificación de EC

Los elementos de configuración son aquellos documentos que se van a requerir para un futuro mantenimiento

Nombrado jerárquico:

- Asociado a proyectos particulares
- No reusable

No debería cambiar de forma arbitraria

Base de datos de configuraciones (I)

Usada para registrar cualquier información relevante relacionada con las configuraciones

Sirve de apoyo a la evaluación del impacto del cambio

Debe proporcionar respuestas a preguntas cómo:

- (1) ¿A qué clientes se les ha entregado una versión particular del sistema?
- (2) ¿Qué hw y SO son necesarios para ejecutar una determinada versión?
- (3) ¿Cuántas versiones del sistema se han creado y cuáles son sus fechas de creación?

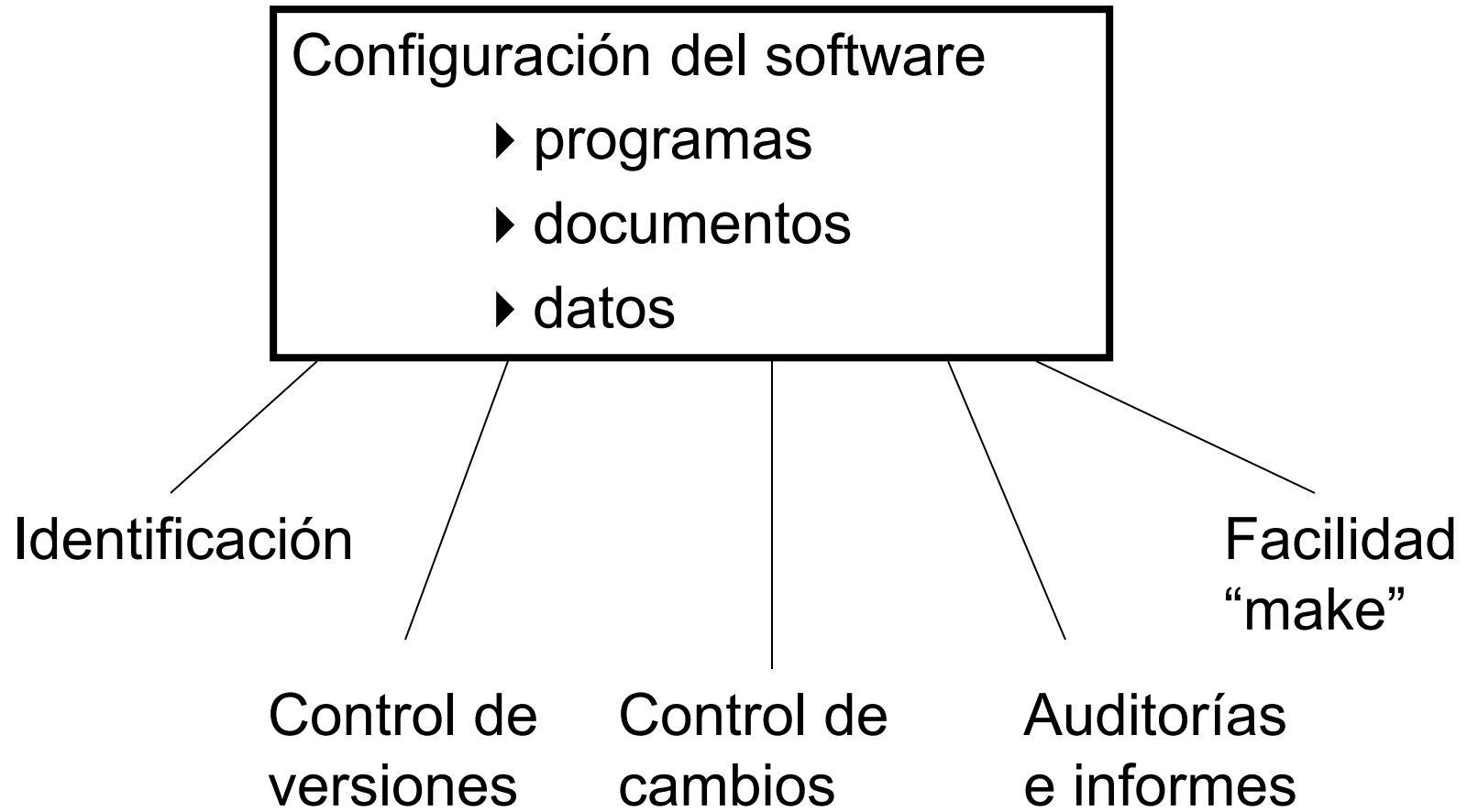
Base de datos de configuraciones (II)

- (4) ¿Qué versiones de un sistema podrían verse afectadas si una componente particular es cambiada?
- (5) ¿Cuántas peticiones de cambio se han hecho sobre una determinada versión?
- (6) ¿Cuántos fallos se han registrado para una versión particular?

Puede ser creada como:

- Un sistema separado
- Integrada con la gestión de versiones y sistema de control que almacena los documentos formales del proyecto

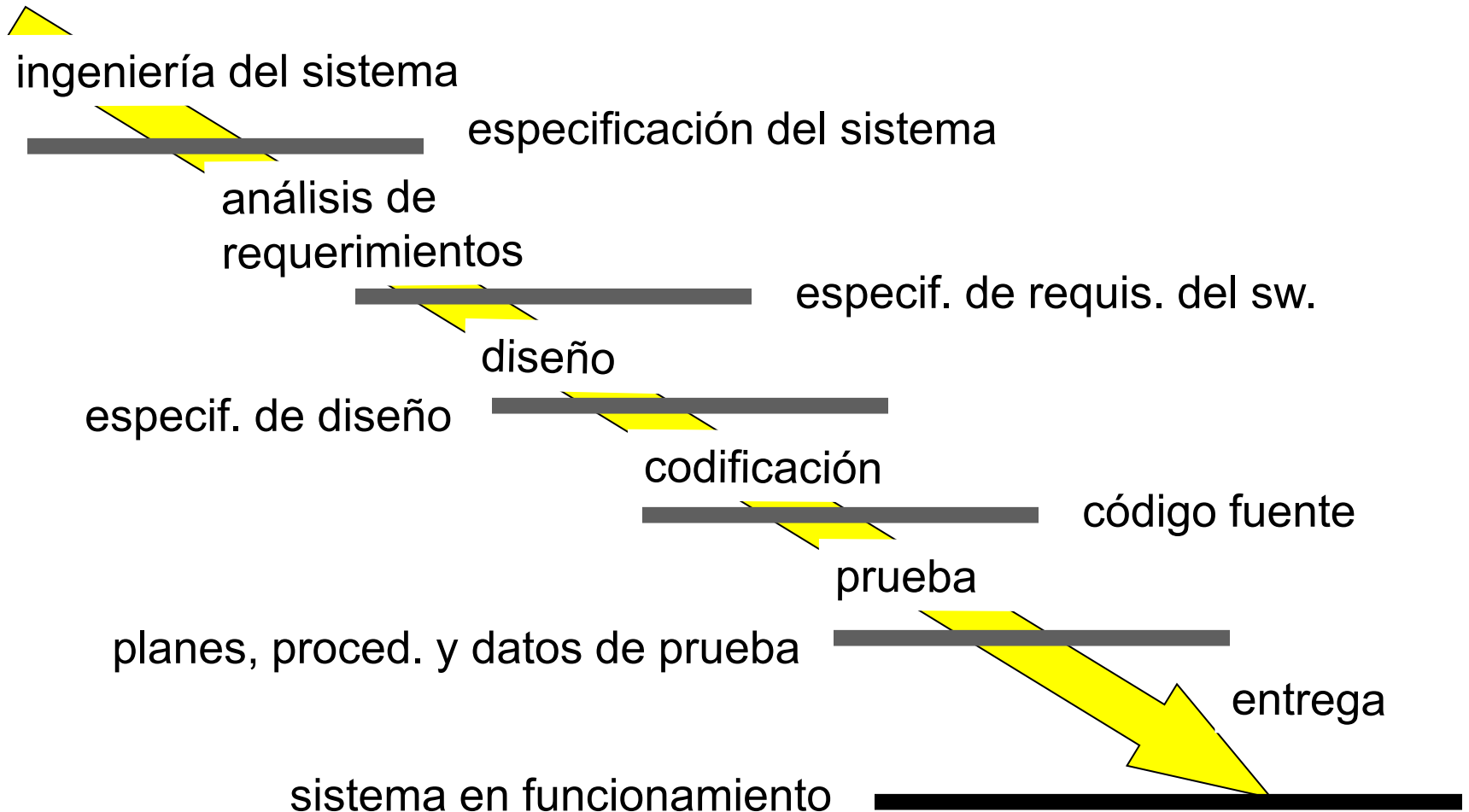
Gestión de configuraciones



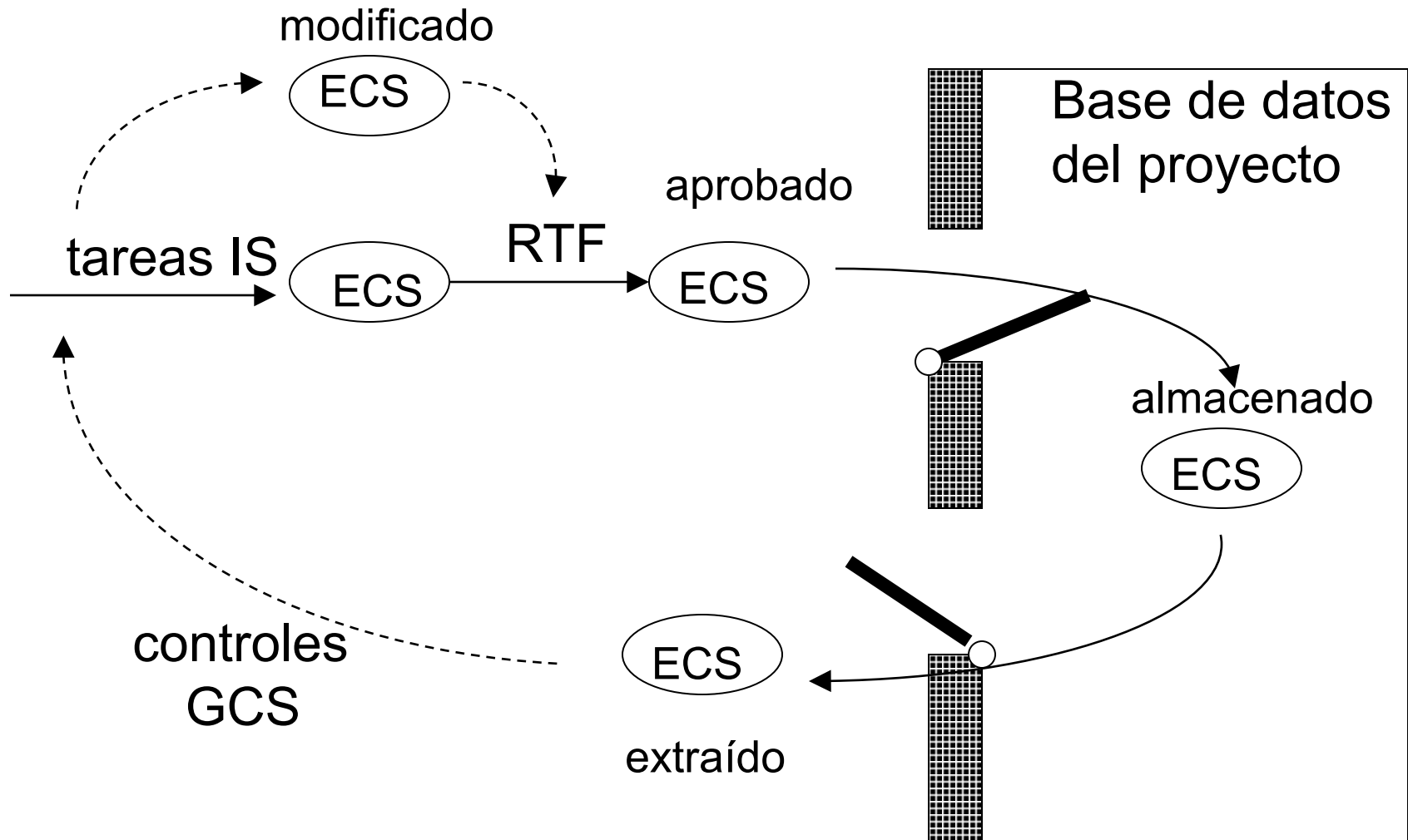
ECS'S típicos

- ❑ Especificación del sistema
- ❑ Plan del proyecto software
- ❑ Especificación de requerimientos del software
- ❑ Manual de usuario preliminar
- ❑ Especificación de diseño
- ❑ Diseño preliminar y detallado
- ❑ Listados de código fuente
- ❑ Planes y procedimientos de prueba
- ❑ Manual de instalación, operación y usuario
- ❑ Ejecutables software

Líneas base (I)



Líneas base (II)



Control de versiones (I)

Implica la identificación y seguimiento de diferentes versiones y “*releases*” del sistema

- Diseño de procedimientos para asegurar que diferentes versiones del sistema puedan recuperarse cuando se requieran, y evitar que no sean cambiadas accidentalmente
- Hay un contacto con el cliente para planificar cuándo deberían distribuirse nuevas *releases*

Control de versiones (II)

VERSION: instancia de un sistema que difiere de algún modo de otras instancias (funcionalidad, rendimiento, fallos...)

RELEASE: es una versión del sistema distribuida a los clientes (funcionalidades nuevas o nueva plataforma)

Incluye:


- Ficheros de configuración
- Ficheros de datos
- Instalación del programa
- Documentación electrónica y sobre papel describiendo el sistema

Control de versiones (III)

Nueva versión  nuevo fuente + construcc. stma. (1)

Nueva release  (1) + fich. datos y conf + nueva docum.

El grupo de GC debe decidir cuándo los componentes afectados por un cambio deben ser reconstruidos en una nueva versión o una nueva release

Decisión forzada a veces por fallos descubiertos por el cliente
 parcheo del código objeto (mejor solución: versión nueva sin documentación)

Herramientas de gest. de versiones

Unix : SCCS, RCS

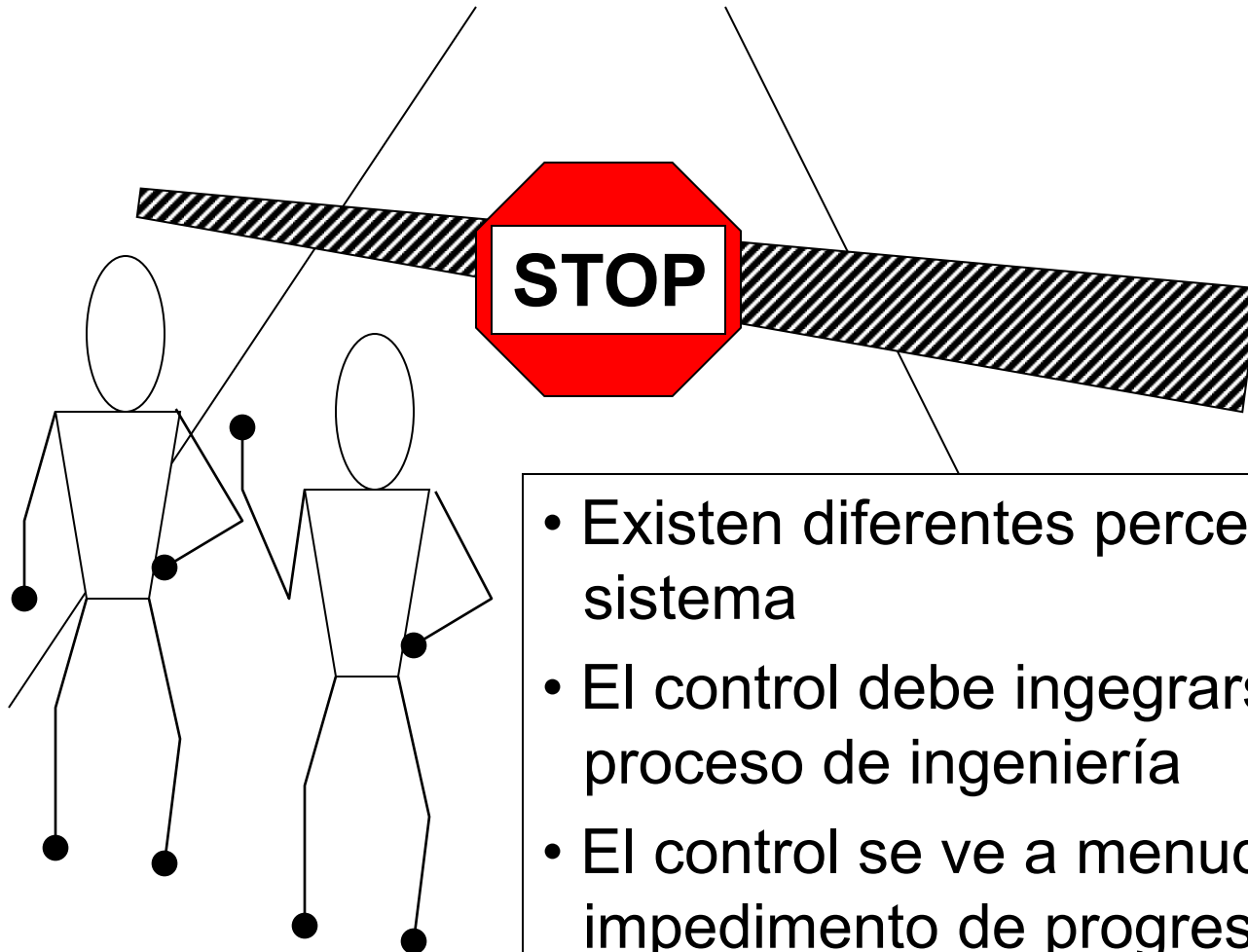
- Identificación de versiones y releases
- Cambios controlados
- Gestión de almacenamiento
- Registro de la historia de cambios

RCS:

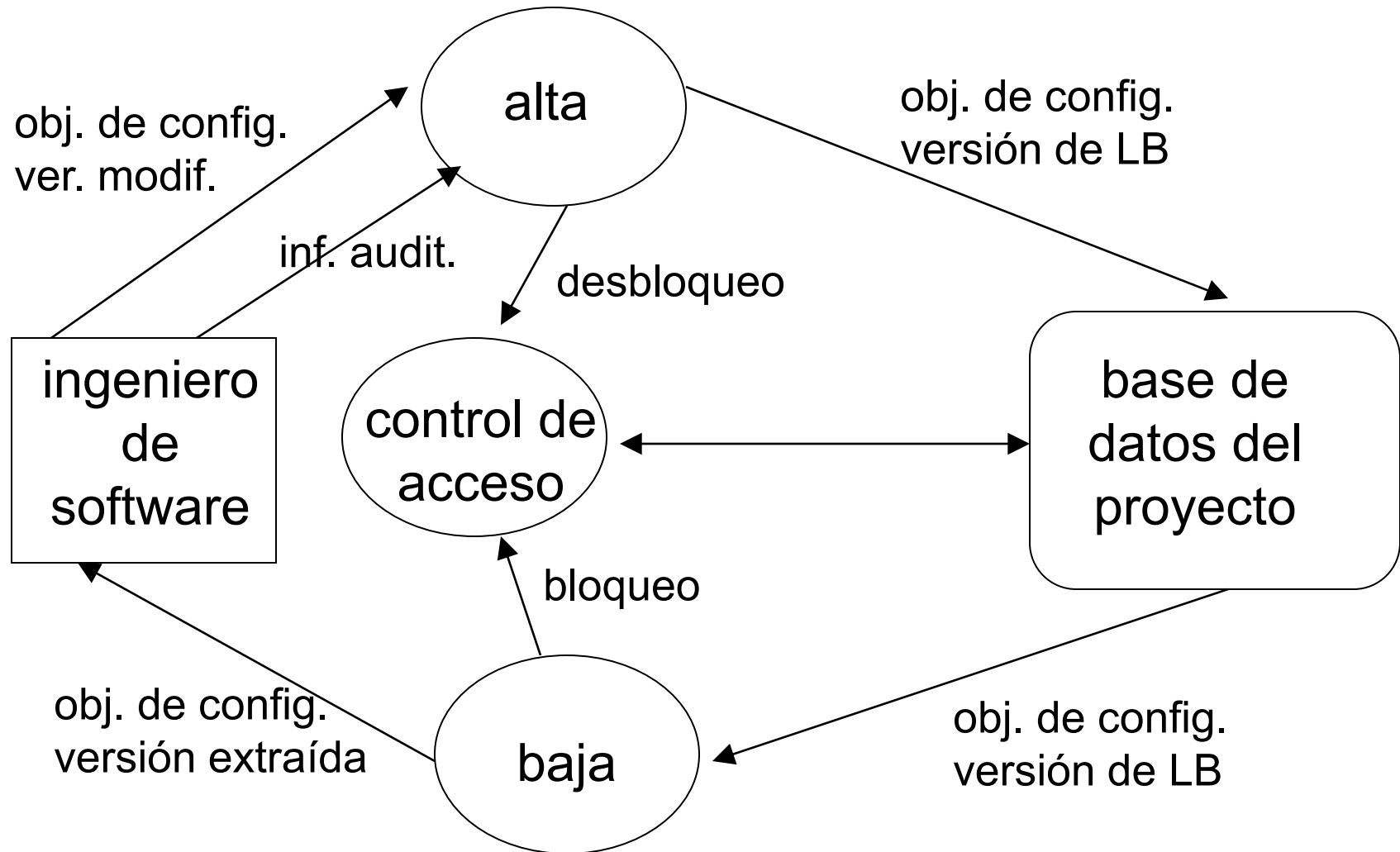
- Graba el código fuente de la versión más reciente
- Soporta el desarrollo paralelo de diferentes releases
- Capacidad de “mezcla” de versiones

Diseñados para trabajar con texto ASCII

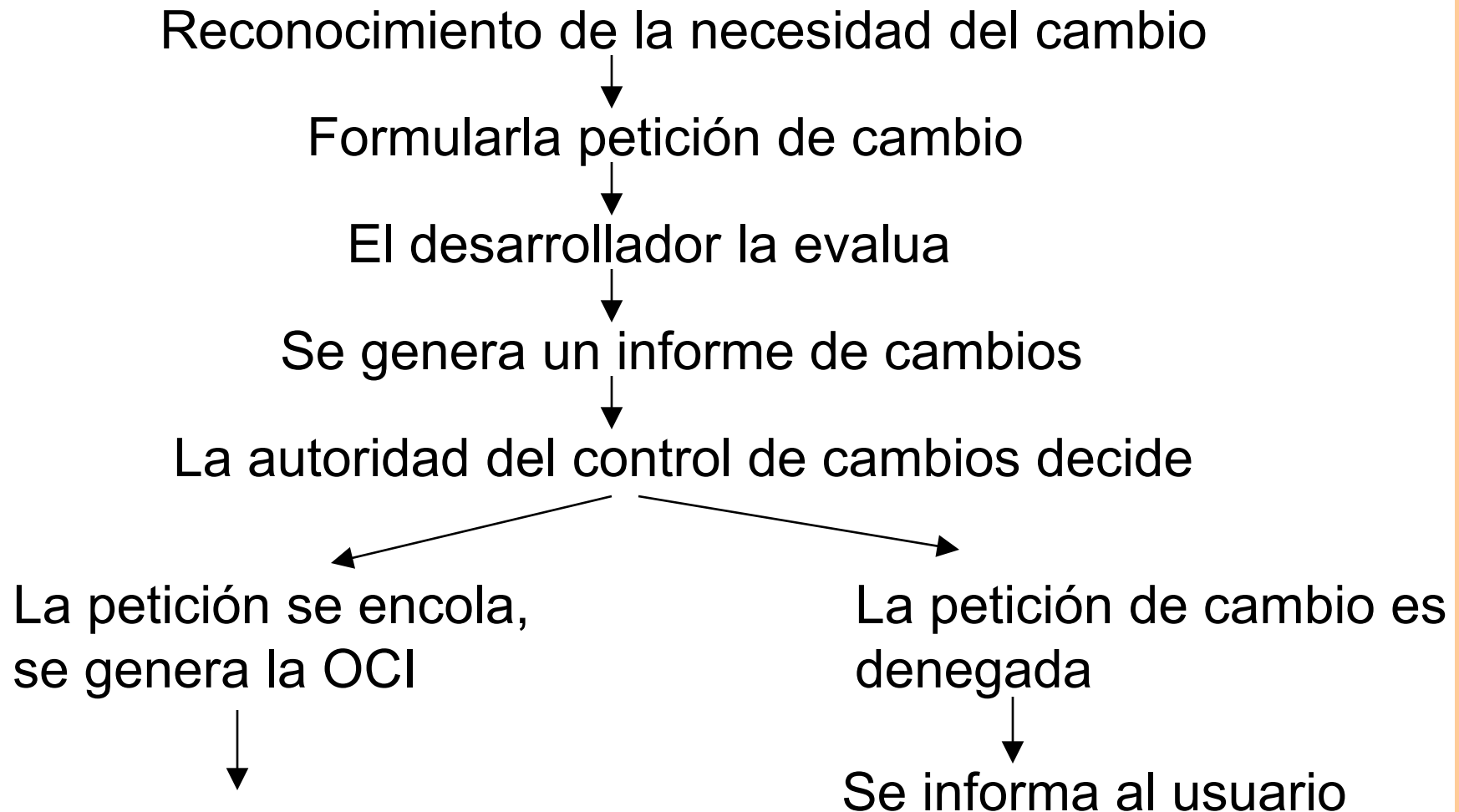
Control de Cambios



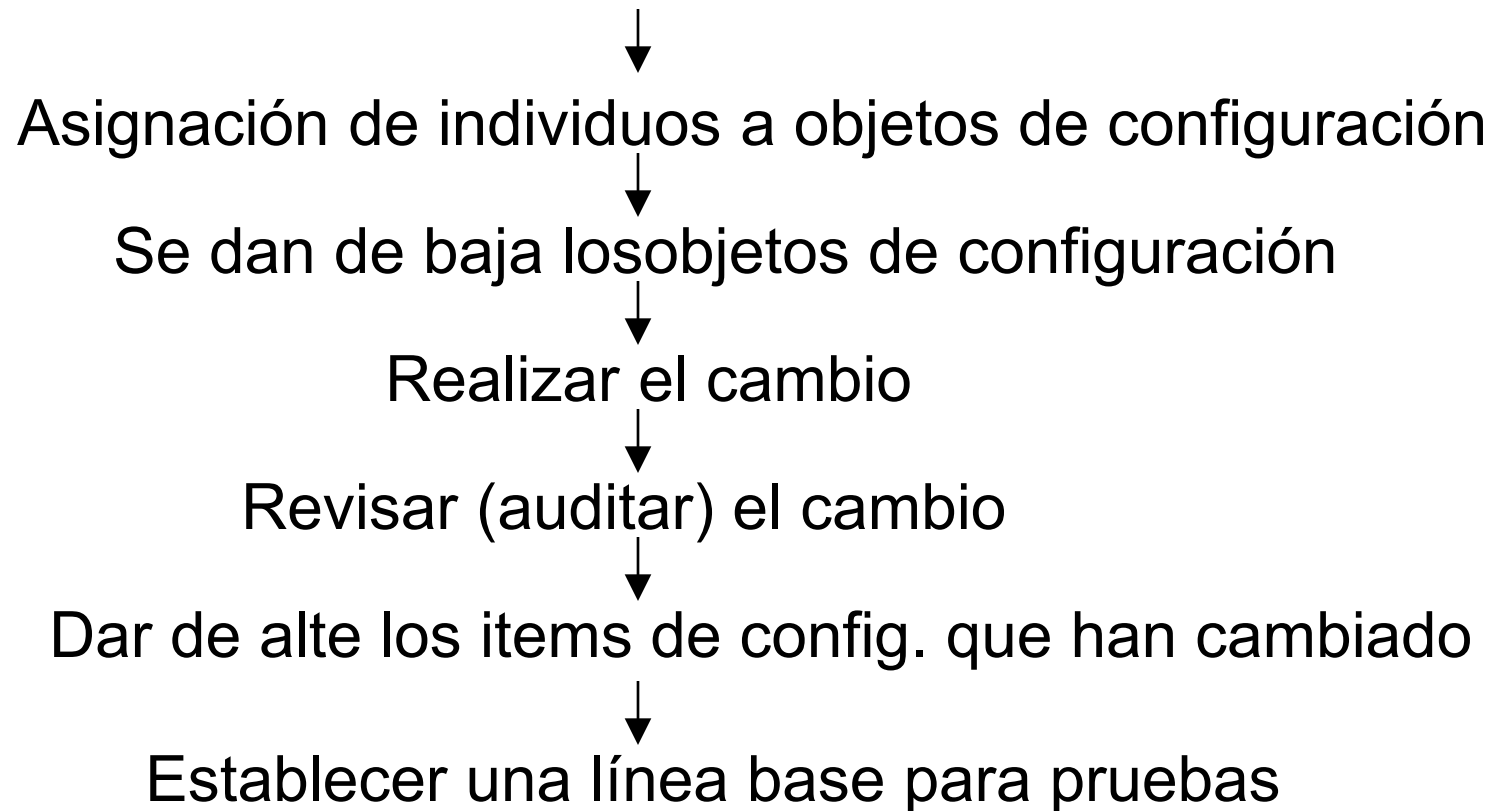
Control de Acceso



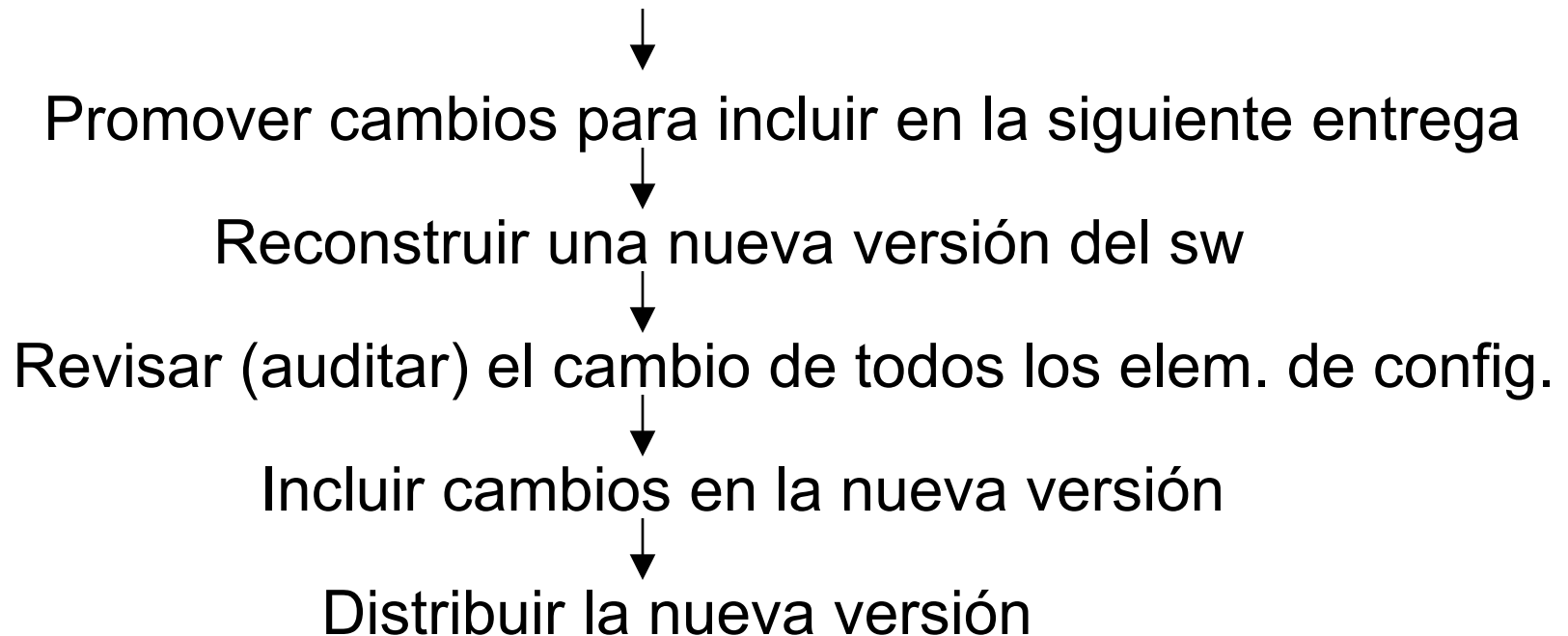
Proceso de control de cambios (I)



Proceso de control de cambios (II)



Proceso de control de cambios (III)



Auditoría de la Configuración

¿Cómo podemos asegurar que el cambio se ha implementado correctamente?

- ❑ Revisiones técnicas formales

«corrección técnica»

- ❑ Auditorías de configuración

«responde a las siguientes preguntas:

- ❑ ¿se ha hecho el cambio especificado?

- ❑ ¿se han seguido los estándares de IS?

- ❑ ¿se han seguido procedimientos para señalar el cambio, registrarlo y divulgarlo?

- ❑ ¿se han actualizado adecuadamente todos los ECS relacionados? »

Informes de estado

Responde a las siguientes preguntas:

- ❑ ¿Qué pasó?
- ❑ ¿Quién lo hizo?
- ❑ ¿Cuándo pasó?
- ❑ ¿Qué más se vió afectado?

Construcción del sistema (I)

Proceso de combinar componentes en un programa que se ejecuta sobre una configuración destino particular

Implica:

- Compilación
- Linkado

Hay que llevar especial cuidado con sistemas desarrollados con un sistema distinto del de destino.

Construcción del sistema (II)

Factores a considerar:

- ❑ ¿Han sido incluidas todas las componentes?
- ❑ ¿Tienen la versión adecuada?
- ❑ ¿Están disponibles todos los ficheros de datos?
- ❑ ¿Los datos tienen el mismo nombre en la componente y en la máquina destino?
- ❑ ¿La versión del compilador es la adecuada?

El proceso se refiere normalmente a componentes físicos.

Herram. de construcc. del sistema

- ❑ La herramienta más ampliamente usada para sistemas UNIX es MAKE.
- ❑ Mantiene una correspondencia entre el código fuente y las versiones de código objeto de un sistema
- ❑ El usuario especifica las dependencias de los componentes y MAKE fuerza automáticamente la recompilación de los ficheros cuyo código fuente haya cambiado después de que el código objeto fuese creado

Limitaciones de “make”

- ❑ Basado en un modelo físico de dependencias (no lógico)
- ❑ Las especificaciones de dependencias (Makefiles) crecen rápidamente, llegando a ser complejas, difíciles de comprender y “caras” de mantener
- ❑ Usa simplemente un modelo de cambio basado en fechas de actualización de ficheros. Puede que cambios en el código fuente no necesiten recompilación
- ❑ No permite (fácilmente) especificar la versión de utilización de herramientas como el compilador
- ❑ No “finamente” enlazado con herramientas de gestión de configuraciones como RCS

Beneficios GCS

- ❑ Reduce el esfuerzo necesario para gestionar y realizar el cambio - mejora la productividad
- ❑ Conduce a una mejora de la integridad y seguridad del software - incremento de la calidad
- ❑ Genera información sobre el proceso - mejora de la gestión del control
- ❑ Mantiene una base de datos de desarrollo de software - mejor registro y seguimiento de informes