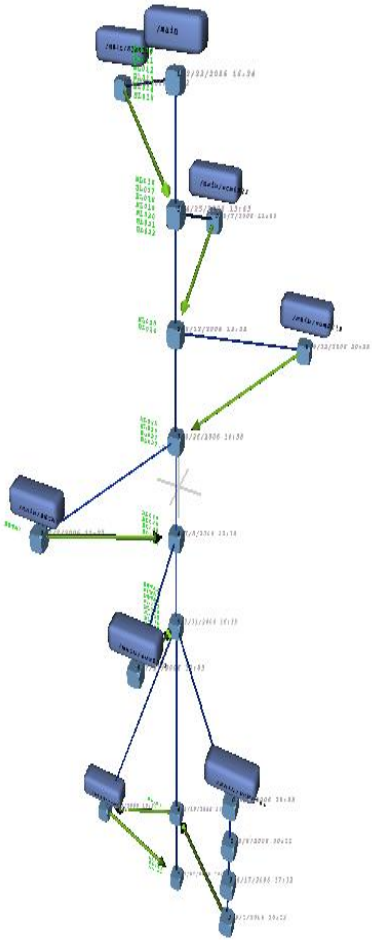
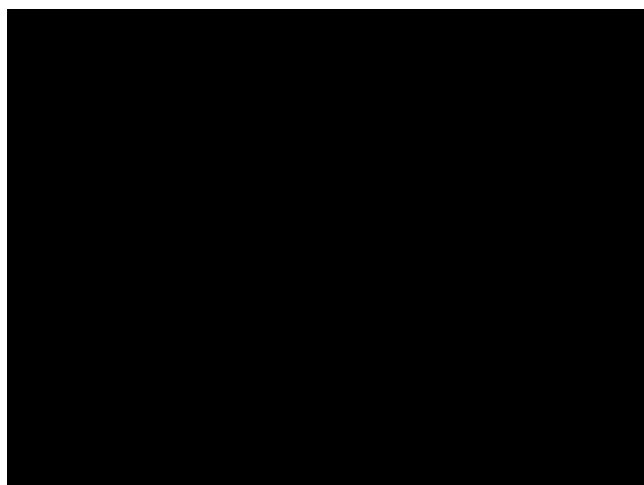


Gestión de Configuración de Software

“No hay nada permanente, excepto el cambio”
Heráclito, 500 a.C.

Juan José Vanzetti





“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”

Origen de los cambios

Por **cambio** se entiende toda *alteración* de algún *componente del negocio* que surge como una *necesidad interna*, una exigencia del mercado, una *obligación legal* o una *innovación tecnológica*

Origen de los cambios

- Nuevos negocios o condiciones comerciales que dictan los cambios en los requerimientos del producto o en las normas comerciales.
- Nuevas necesidades del cliente que demanda la modificación de los datos, funcionalidades o servicios.
- Reorganización y/o reducción del volumen comercial que provoca cambios en las prioridades del proyecto o en la estructura del equipo.
- Restricciones presupuestarias o de planificación que provocan redefinición del sistema o producto.

El cambio como problema

- ▶ Problema de *coordinación* entre grupos de trabajo
- ▶ Mala *comunicación* entre áreas comerciales y operativas
- ▶ Mala *actualización* de procesos y documentos
- ▶ Gran *esfuerzo de retrabajo* y gran cantidad de errores
- ▶ Baja *productividad* personal
- ▶ Mala *calidad* frente al cliente

Para convivir con el cambio es necesario

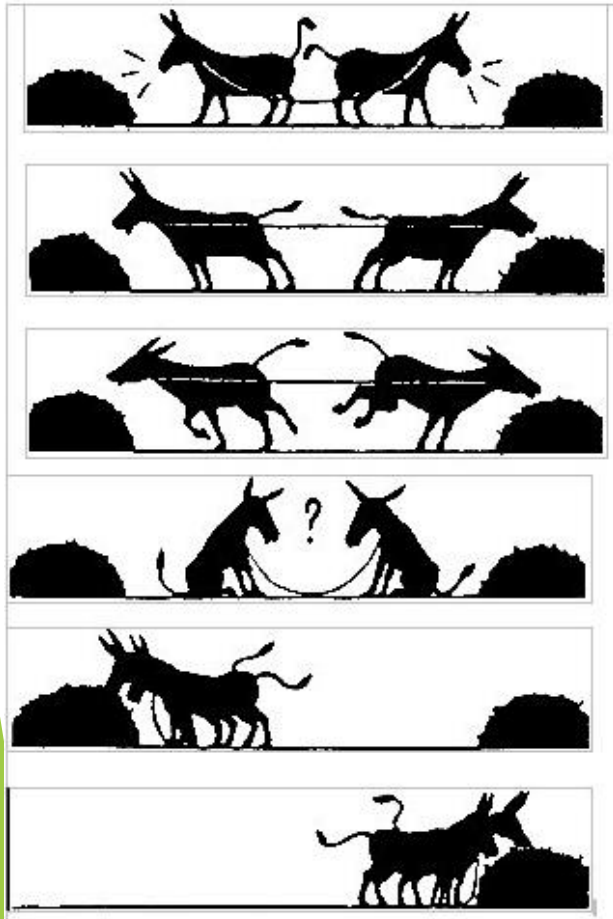
- ▶ Conocerlo
- ▶ Controlarlo
- ▶ Administrarlo
- ▶ Medirlo
- ▶ Identificar los componentes del negocio
- ▶ Facilitar el trabajo por versiones

Origen de los cambios

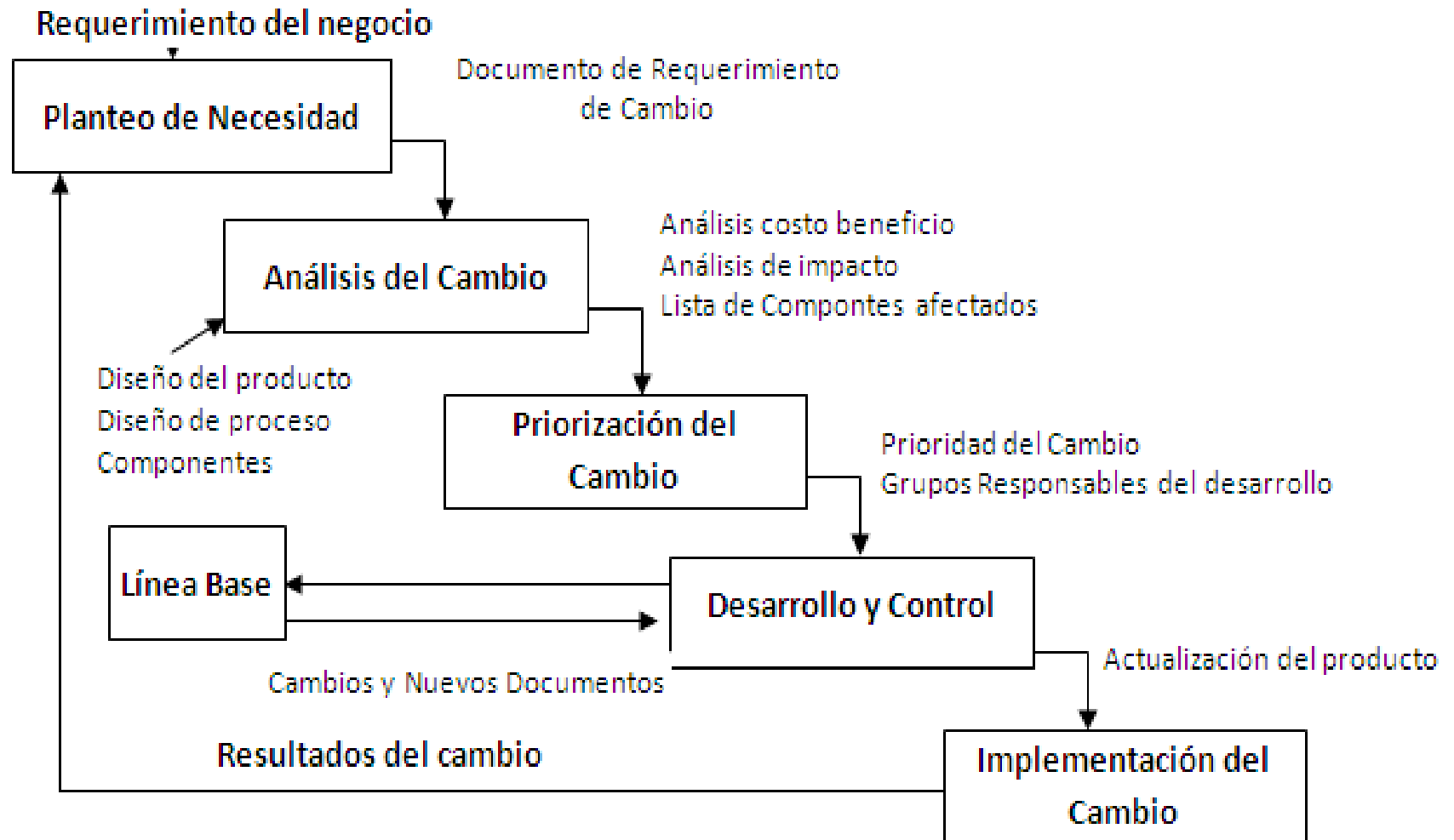
Para entender el cambio es necesario conocer sus particularidades

- ▶ Un cambio *impacta* en más de un *componente del negocio*
- ▶ Un cambio *trivial* se puede expandir en forma *descontrolada*
- ▶ Un cambio puede tener *relación* con *otros cambios*
- ▶ Un cambio *mal aplicado* puede afectar *productos no involucrados* en su implementación

Gestión de Cambio



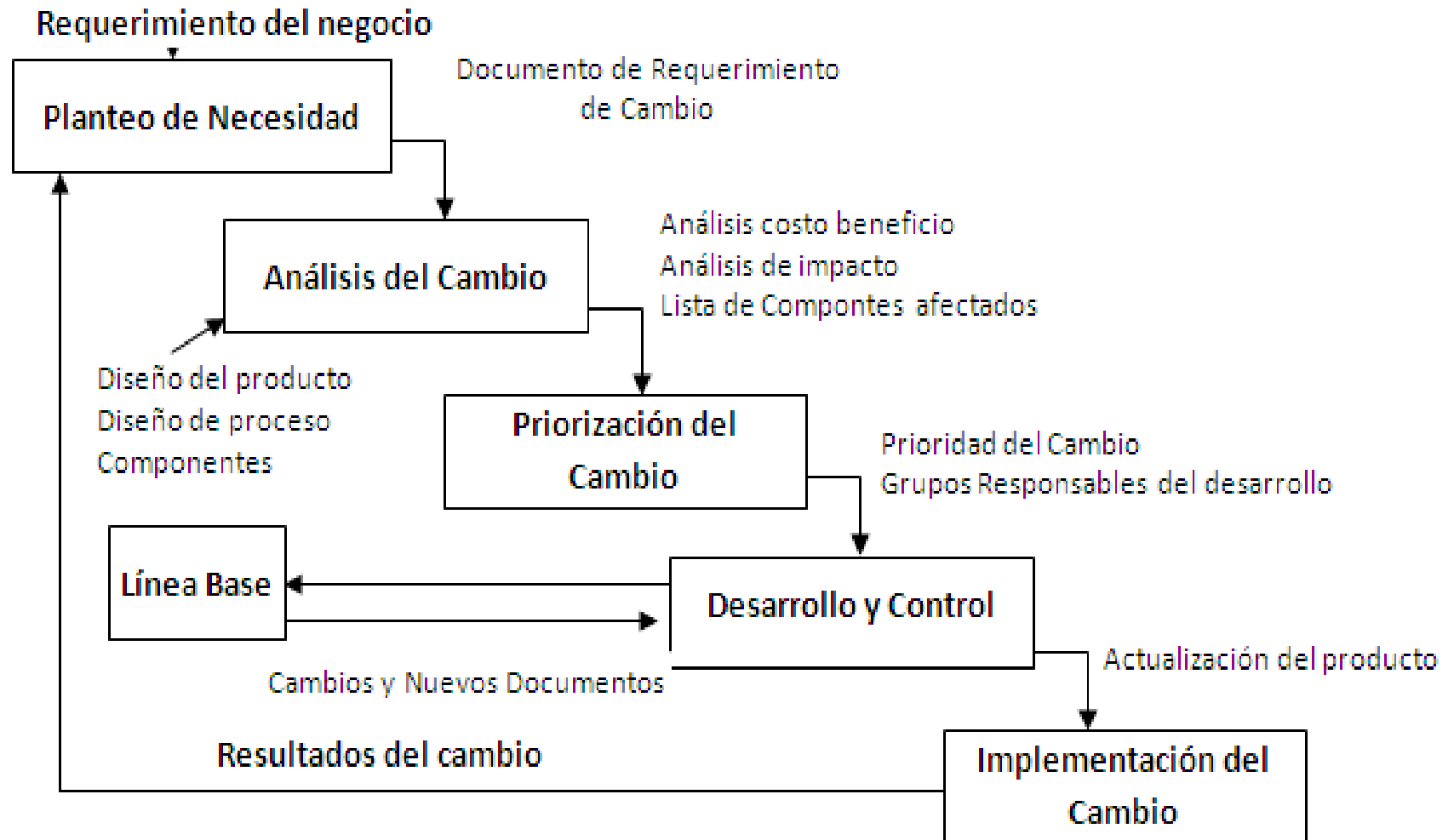
Los procedimientos de gestión de cambios se ocupan del análisis de costos y beneficios de los cambios propuestos, aprobando aquellos cambios que merecen la pena y registrando los componentes del sistema que se tienen que cambiar.



Transforma un requerimiento del negocio en un planteo formal de **Cambio.**

El documento de necesidad debe poseer:

- ▶ Objetivo del requerimiento
- ▶ Área productora
- ▶ Producto/s afectado/s
- ▶ Detalle del cambio a realizar
- ▶ Beneficio esperados por el cambio
- ▶ Tiempo de liberación al mercado.



En base al documento de necesidad, se elabora;

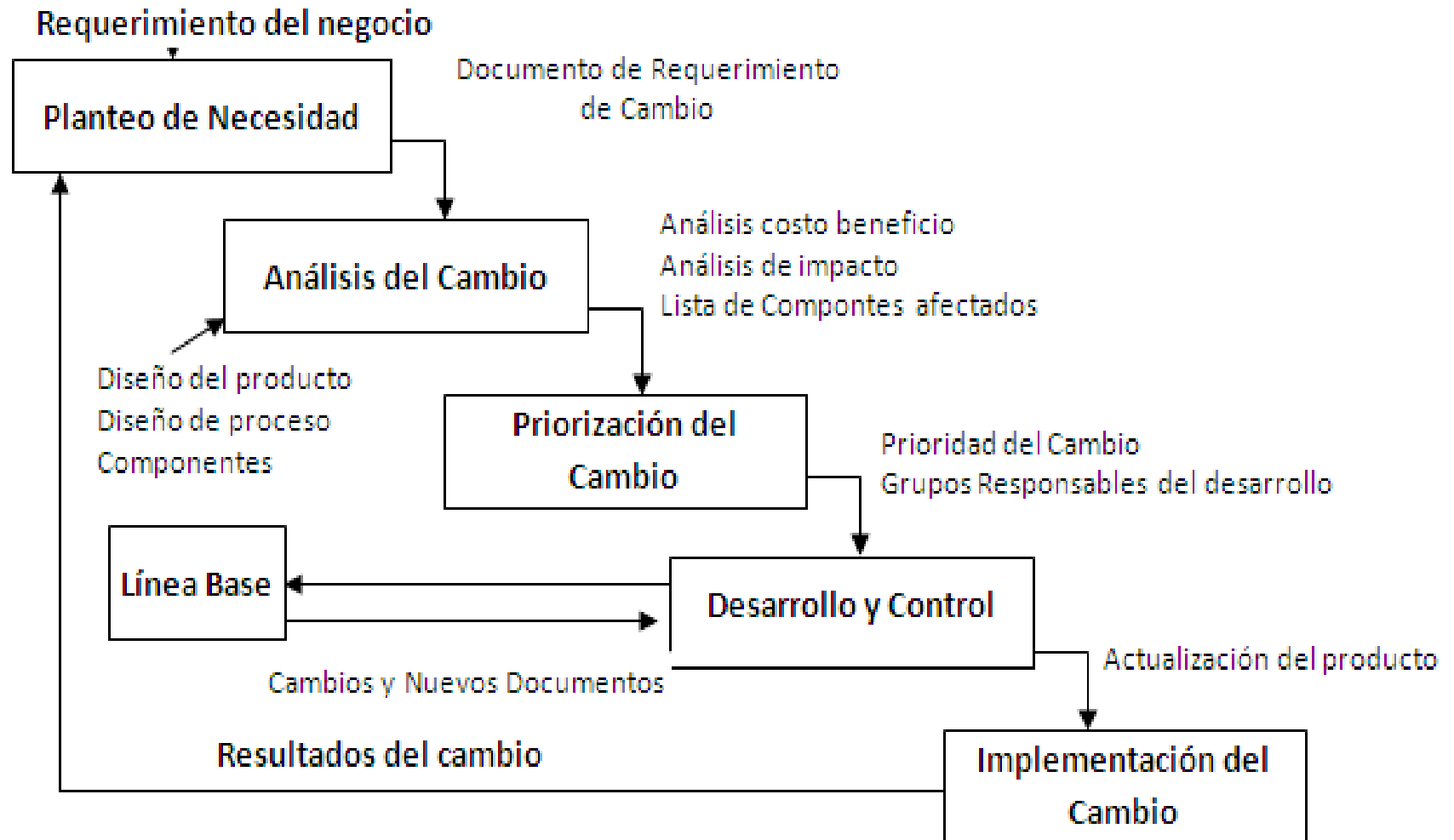
- 1 Análisis de impacto del cambio.(Componentes del negocio afectados y esfuerzo estimado de desarrollo).
- 2 Análisis costo beneficio -
- 3 Análisis de la relación del cambio con otros en ejecución o espera de desarrollo
- 4 Un plan preliminar de trabajo

Análisis de Impacto

- ▶ Análisis de impacto es el conjunto de actividades que permiten determinar los *efectos de un cambio* en los componentes de un producto, proceso u organización
- ▶ Análisis de impacto es la evaluación de los posibles *riesgos* asociados con un cambio a un determinado conjunto de componentes del negocio, incluyendo las estimaciones de sus efectos en recursos, tiempos y actividades

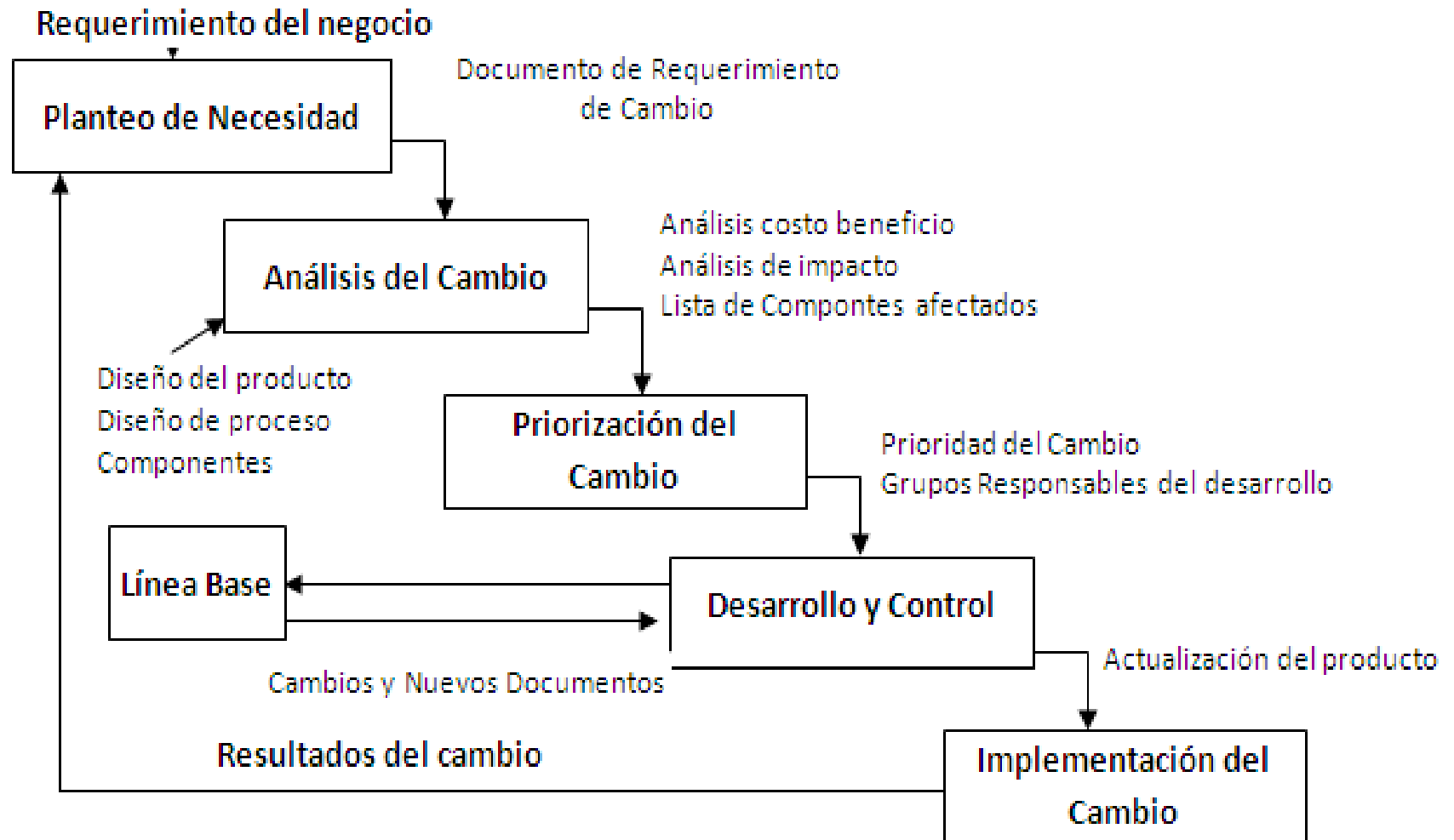
- ▶ Para entender y hacer un efectivo análisis de impacto es necesario:
 - Disponer de un proceso documentado para el desarrollo de productos
 - Disponer de los procesos y productos elaborados bien documentados y estandarizados
 - Establecer interrelaciones entre cada uno de los componentes del negocio

- ▶ Es fundamental definir un documento que contenga:
 - Identificación del cambio
 - Descripción del cambio
 - Lista de componentes afectadas
 - Lista de las partes de cada componente afectado
 - Ponderación del impacto en cada componente,
 - Etc



Esta actividad, en base al resultado del **Análisis del Cambio**, asigna una **prioridad**, y asigna los **recursos necesarios** para su desarrollo

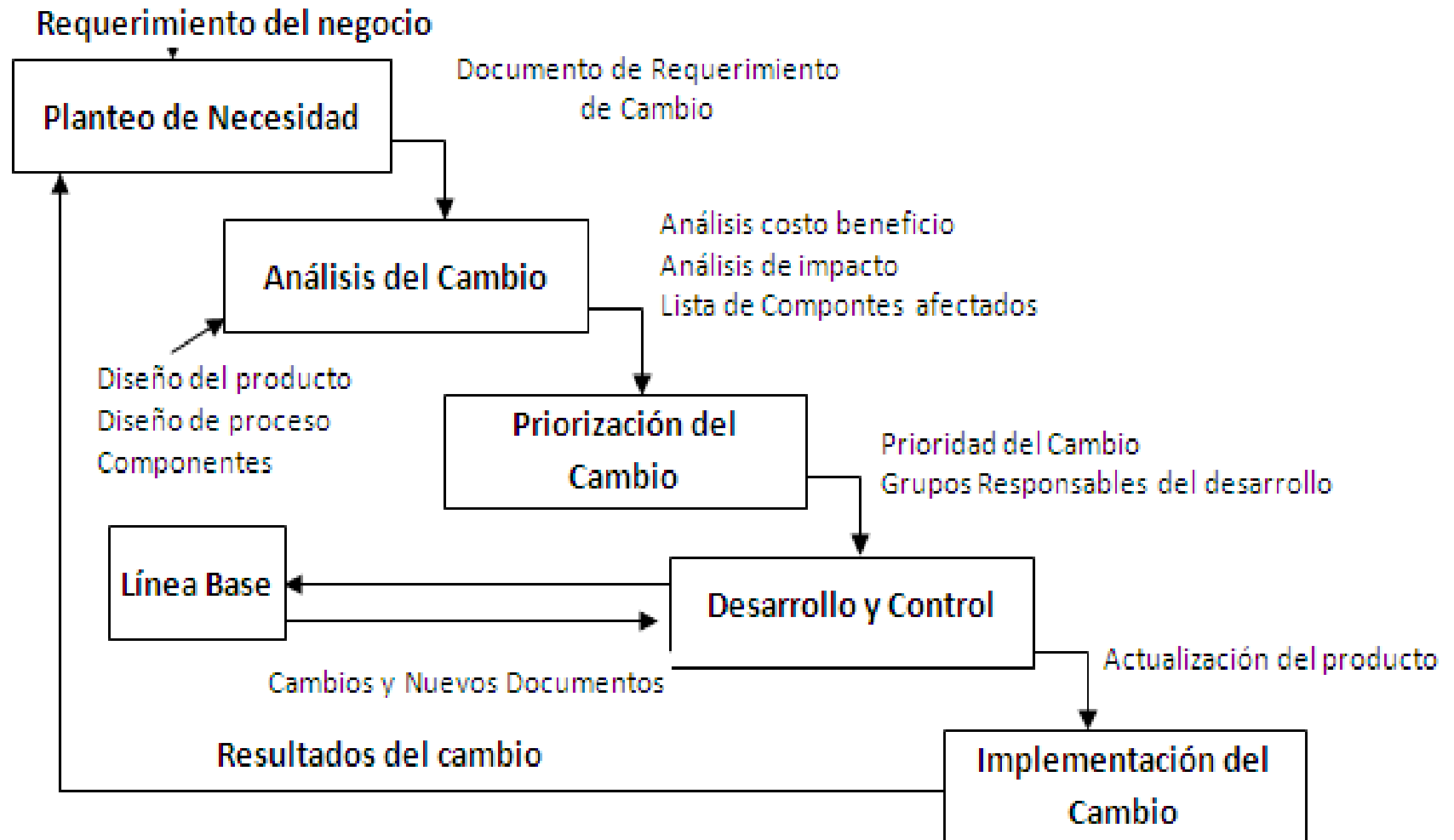
- ▶ Se deben ajustar planes y lista de cambios
- ▶ Esta actividad se puede integrar con el Análisis del Cambio.
- ▶ Facilita las actividades de coordinación y uso de los recursos



- ▶ Plasma el cambio dentro de cada uno de los componentes afectados, a partir de diseñar una solución adecuada.
- ▶ Produce todo los materiales necesarios para efectuar revisiones del diseño del cambio.
- ▶ Genera la nueva versión de cada documento, producto y proceso.
- ▶ Arma el plan de implantación de los cambios.

- ▶ Los procedimientos de control de los componentes deben asegurar:
 - ▶ Que un componente particular afectado por el cambio este en un solo lugar a la vez
 - ▶ Que se este trabajando con la versión correcta
 - ▶ La identificación de la versión necesaria para el cambio
 - ▶ La facilidad de almacenamiento de cada componente

- ▶ El control de calidad debe asegurar :
 - ▶ Que el cambio satisface los requerimientos
 - ▶ Que el plan de desarrollo se cumple dentro de los presupuestos
 - ▶ Que se respeten todos los estándares del proceso
 - ▶ Que los componentes afectados están bajo control



Implementación del Cambio

- ▶ Instala el cambio en la organización
- ▶ Capacita a los recursos afectados por el cambio
- ▶ Renueve documentos obsoletos
- ▶ Apoya la instalación de tecnología
- ▶ Monitoree el nuevo proceso
- ▶ Recopila las sugerencias y la satisfacción del cliente

Concepto de Configuración

- ▶ *"Característica físicas y funcionales del software y/o hardware de un producto como se describe en la documentación técnica" IEE STD 610*
- ▶ Una **"configuración"** es una combinación de versiones particulares de los componentes que forman un sistema consistente
- ▶ Desde el punto de vista de evolución, es el conjunto de las versiones de los objetos componentes en un instante dado

Gestión de la Configuración de Software - Definición

SCM es la gestión de la identificación única, almacenamiento controlado, control de cambios e informe de estado de elementos intermedios de trabajo seleccionados, componentes del producto y productos durante el ciclo de vida de un sistema

Disciplina de la Ingeniería de Software que comprende las herramientas y técnicas. Tiene como objetivo mantener la integridad de los componentes del producto de software, evaluar y controlar los cambios

Babich: El arte de coordinar el desarrollo de software para minimizar errores... se denomina GCS. La GCS es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación

Gestión de la Configuración de Software - Definición

*Disciplina que se ocupa de **identificar** la **configuración** de un sistema en **puntos discretos en el tiempo** con el propósito de controlar sistemáticamente los cambios a esta configuración y mantener la **integridad y traceability** de esta configuración a través del ciclo de vida del sistema*

Bersoff

La disciplina dedicada a **identificar y documentar** las características físicas y funcionales de un **ítem de configuración**, controlar **cambios** a esas características, registrar y reportar cambios y **estados** de la configuración y verificar el cumplimiento de los requerimientos

IEEE STD 610

Propósito del SCM

- ▶ El propósito de la función de SCM es mantener el seguimiento y la integridad de los activos de proyecto, durante su evolución en el proceso de desarrollo de software.
- ▶ El SCM está relacionado con la estructura del producto.
- ▶ El SCM se ocupa de identificar los artefactos, versiones y dependencias entre artefactos, así también como la identificación de las configuraciones que forman conjuntos consistentes de artefactos interrelacionados.
- ▶ También se ocupa de organizar el trabajo de forma tal de los desarrolladores individuales del equipo no estén constantemente pisándose entre ellos.
- ▶ Los miembros del equipo de proyecto deben ser capaces de identificar y ubicar los artefactos, seleccionar la versión adecuada de cada uno de ellos, revisar su historia de cambios para entender su estado actual y las razones de los cambios, como así también quien es el responsable actual de los mismos.

Los horrores que SCM resuelve ...

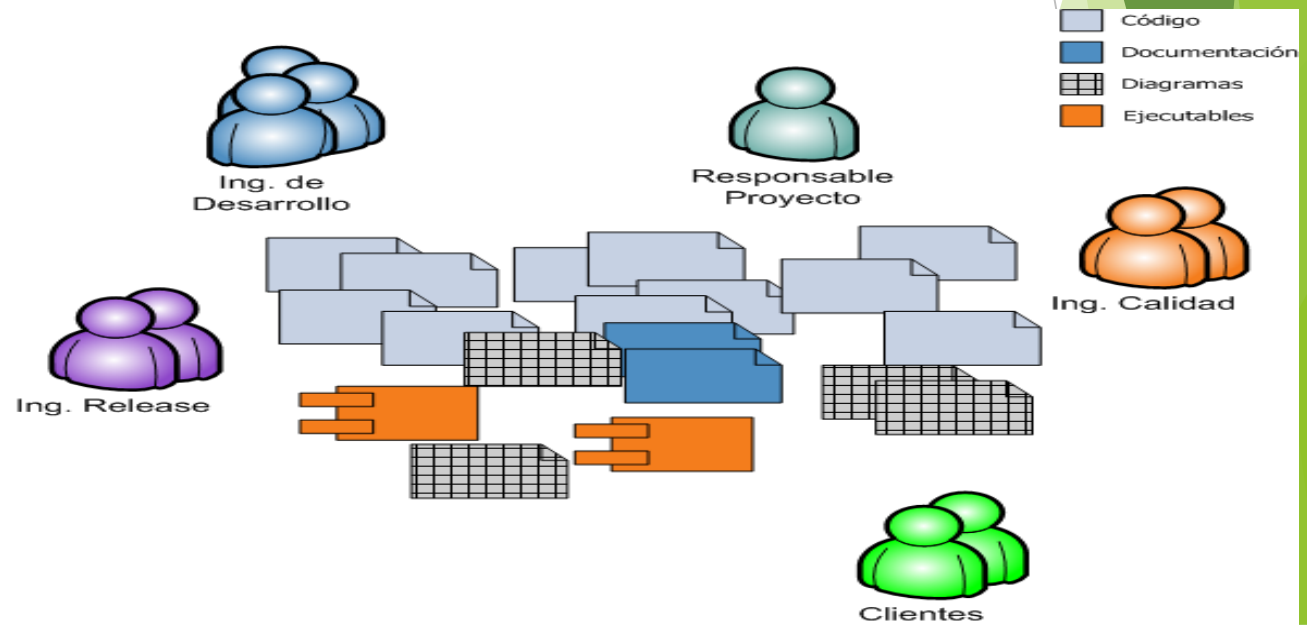
- El *software perdido*: "se que lo escribí, pero no dónde lo puse ..."
- Enlaces desaparecidos: "solía funcionar, pero usa componentes que ya no están ..."
- Pisar el código de otro: desarrolladores que hacen distintos cambios en el mismo código *sobrescribiendo* su trabajo mutuamente
- No hay *botón deshacer*: los nuevos cambios son peores, pero no se puede volver atrás ...
- ¿Qué versión tiene el cliente? ¿A cuál corresponde el *bug*?

Problemas Típicos

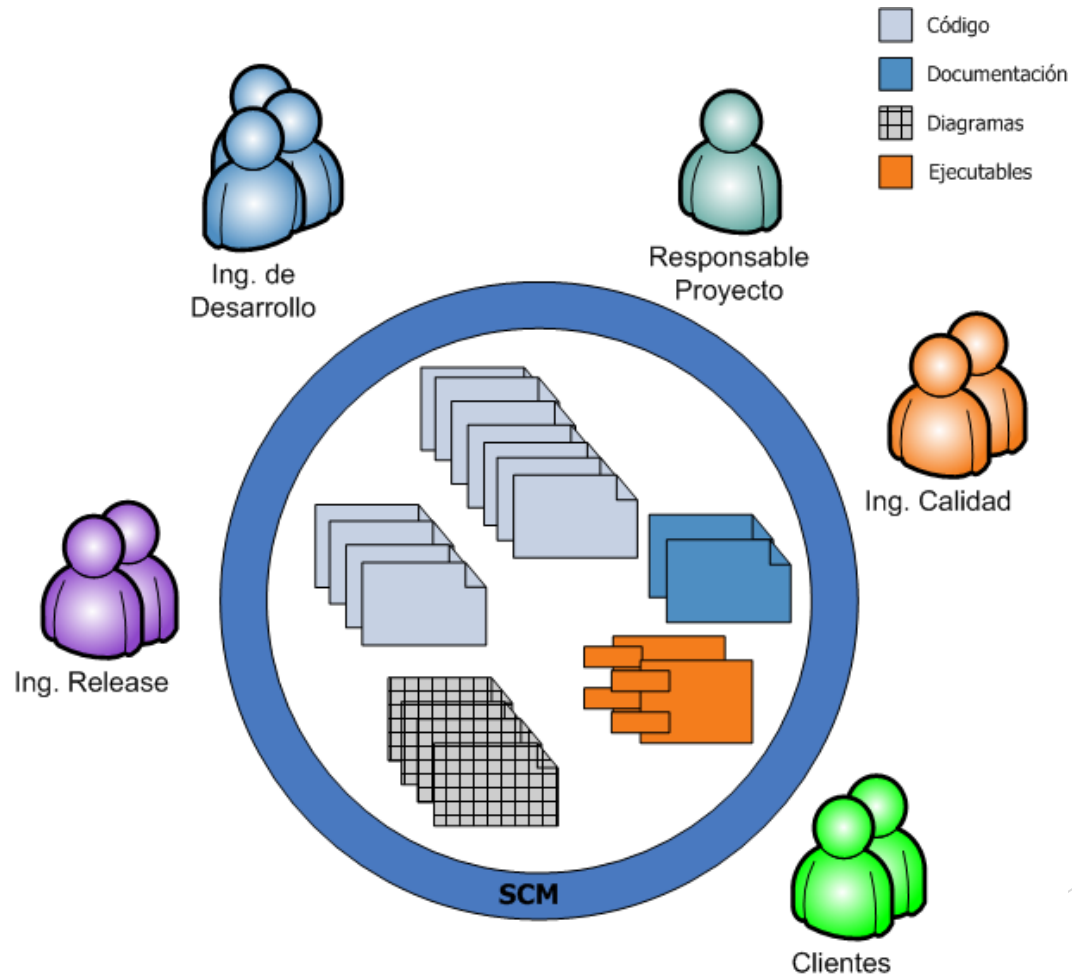
- ▶ **Doble mantenimiento:** El problema de tener múltiples copias idénticas o semejantes de piezas de sw
- ▶ **Mantenimiento simultaneo** cuando dos programadores trabajan simultáneamente en la misma pieza de sw
- ▶ **Código compartido** cuando se corrige un bug o se modifica una pieza de código compartida por otros componentes, cuyos responsables no son notificados del cambio

Problemas del Desarrollo de Software sin SCM

- La versión actual del código se sobrescribe por una anterior
- Una actualización crítica se descarta de la versión final
- Se hacen cambios a una versión incorrecta del código
- Reaparecen errores ya corregidos
- No se logra determinar qué versiones van en una entrega
- Las construcciones no son reproducibles
- Muchas posibilidades de error cuando se mantienen múltiples versiones
- No hay historia de los cambios
- Los jefes de proyecto no pueden medir el avance
- Pobre comunicación del equipo



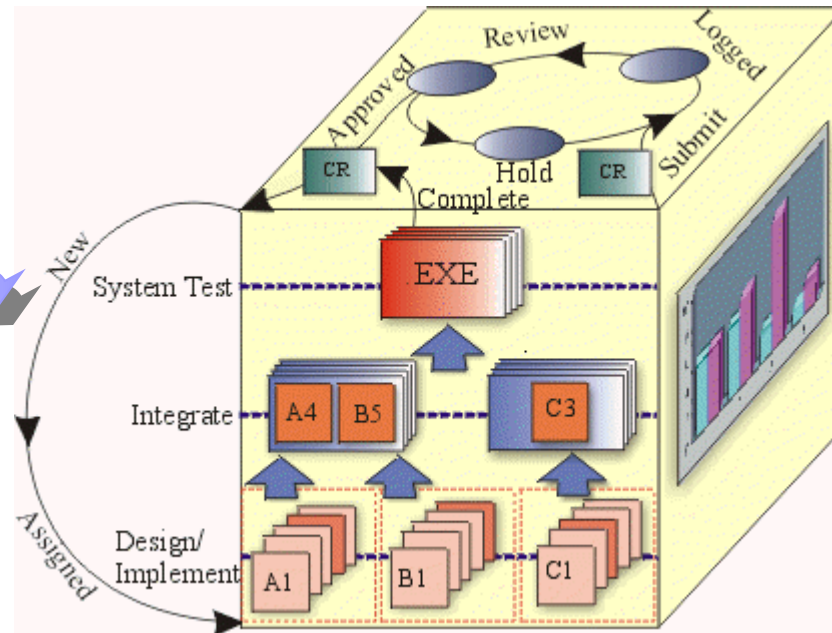
Desarrollo de software con SCM



SCM

¿Qué contempla?

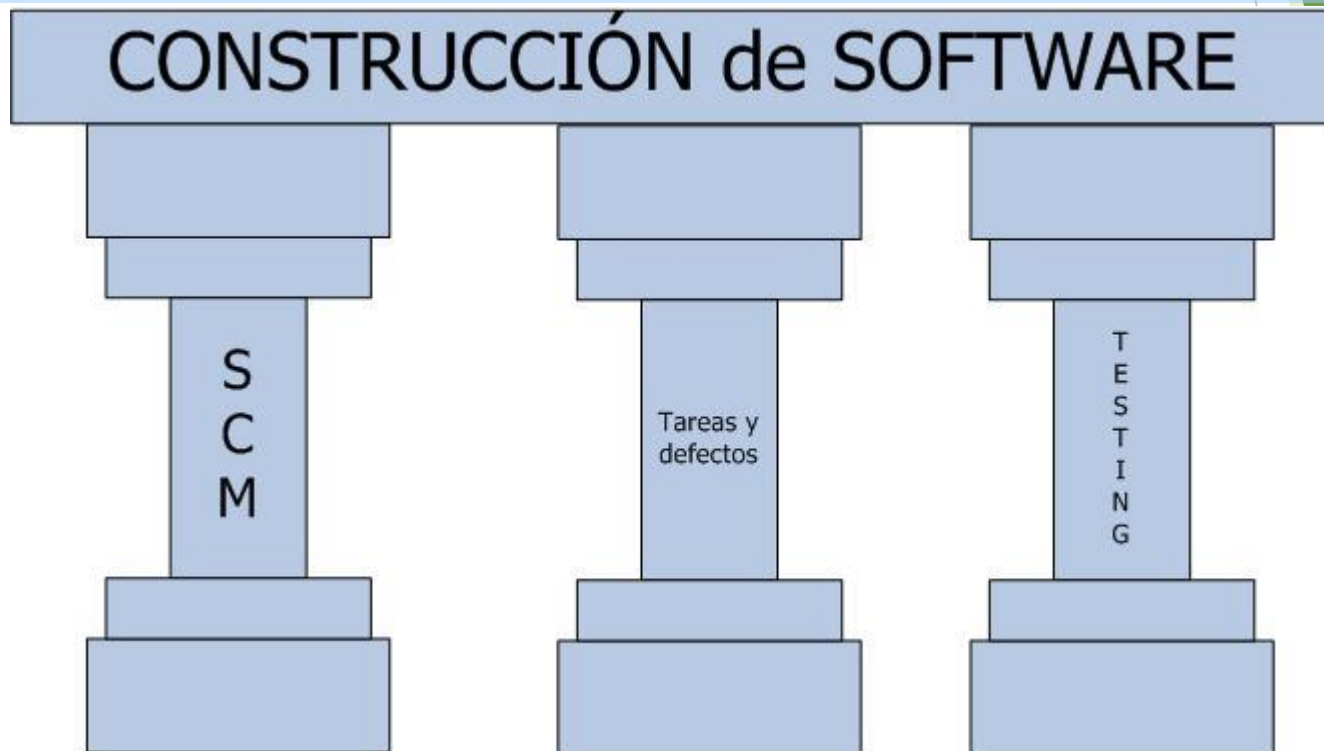
Gestión de Versiones



Gestión de Cambios

Métricas

- Mejora la productividad
- Costos de mantenimiento más bajos
- Reducción de defectos
- Independencia de personas



Que me permite conocer la GCS

- ▶ ¿Quién hizo los cambios?
- ▶ ¿Qué cambios se hicieron al software?
- ▶ ¿Cuándo se hicieron los cambios?
- ▶ ¿Por qué se hicieron los cambios?

La GCS es una actividad de garantía de calidad del software que se aplica en todas las fases del proceso de ingeniería del software.

Ningún Software existe en una única versión. La necesidad de múltiples versiones para:

- Soportar configuraciones de hardware y software alternativas
- Soportar la evolución de la funcionalidad
- Mejorar la calidad
- Encarar desarrollos en forma incremental
- Correr sobre hardware y software totalmente diferente

IEEE

Identificación de la Configuración
Control de Cambios en la Configuración
Generación de Informes de Estado
Auditoria de la Configuración

CMM

Planificación de las actividades de Gestión de Configuración
Identificación de los ECS
Control de cambios a los ECS
Informar a los grupos e individuos involucrados de los cambios a los ECS
Auditoria de la Configuración

ISO

Identificación de la configuración
Control de cambios a la configuración
Informe del estado de la Configuración
Auditoria de la configuración



CM (Configuration Management).

Identificación
Control
Auditoria
Contabilidad de Estado

SCM y Modelos de Madurez CMM

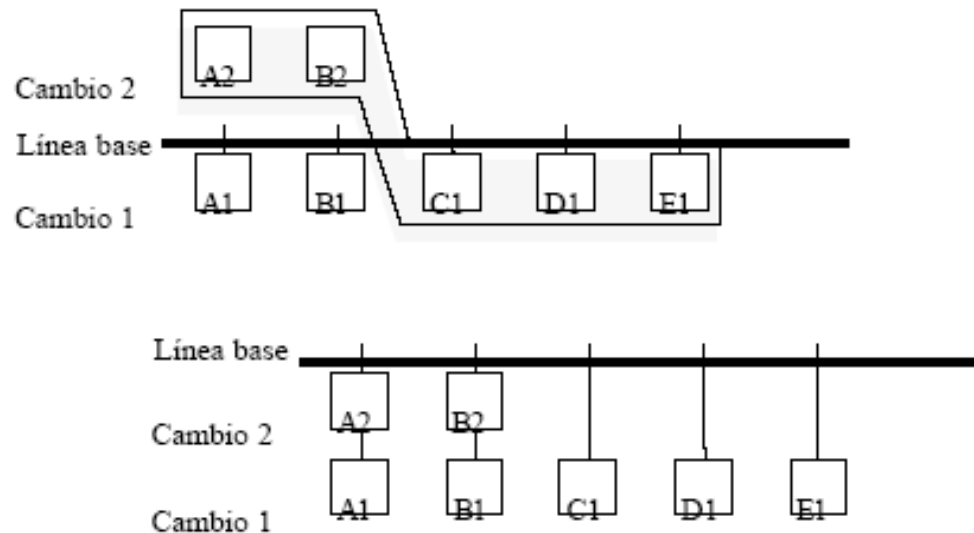
5			Innovación y despliegue organizativo	Análisis causal Innovación y despliegue organizativo
	4	Gestión cuantitativa de proyecto	Rendimiento de proceso organizativo	
	3	Validación Verificación Integración de producto Solución técnica Desarrollo de requisitos	Formación organizativa Definición de proceso organizativo Enfoque en el proceso organizativo	Análisis de decisiones y soluciones
		Gestión de requisitos	Gestión de acuerdos con proveedores. Seguimiento y control de proyecto. Planificación de proyecto.	Gestión de configuración. Aseguramiento de calidad del proceso y del producto. Medición y Análisis.
		Ingeniería	Gestión de proyecto	Gestión de proceso

Soporte

Línea base

La IEEE 610.12-1990 define una línea base como: Una especificación o producto que se ha revisado formalmente y sobre el que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios

Evolución de la Línea Base



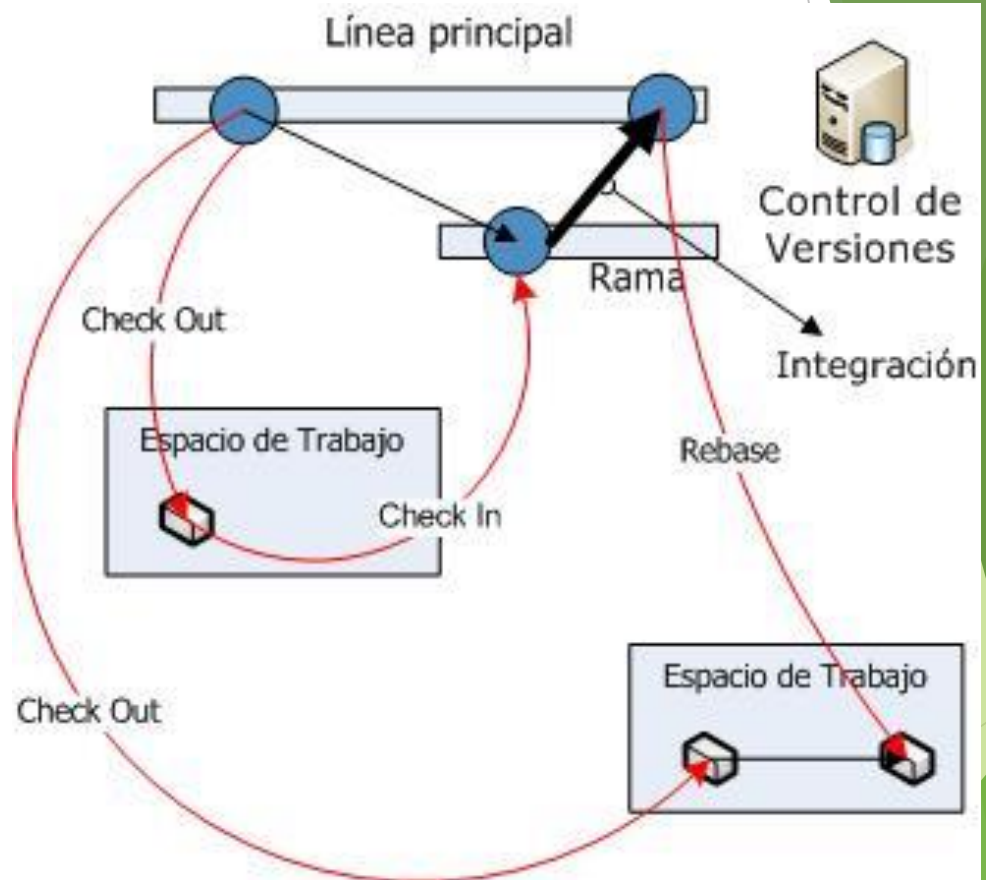
Objetivos de la línea base

- **Sin confusión:** En la base de componentes del software
- **Sin Ambigüedad:** Provee el estándar oficial sobre el que se basa todo trabajo posterior
- **Esta controlada:** Solo se pueden realizar sobre ella cambios autorizados

- La Línea base contiene la copia definitiva de un componente
- Un programador modifica la copia de la línea base
- El trabajo se realiza sobre una copia de la línea base en el espacio de trabajo privado
- El espacio de trabajo privado contiene información **copiada** de la línea base, pero no elimina la componente de la línea base

Operaciones elementales

- Check In
- Check Out
- Integración (*merge*)
- Rebase (update)

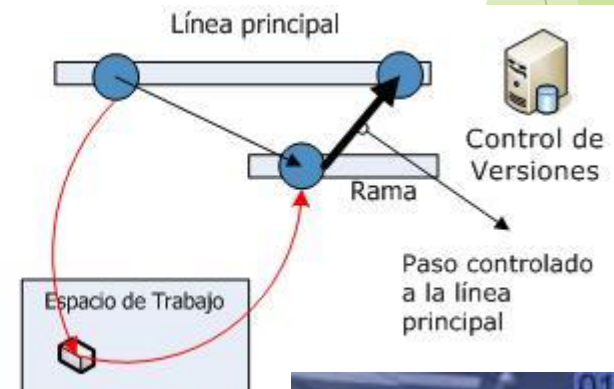
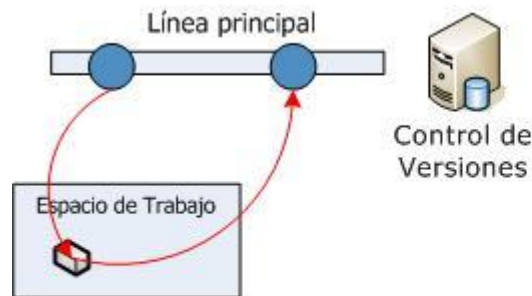
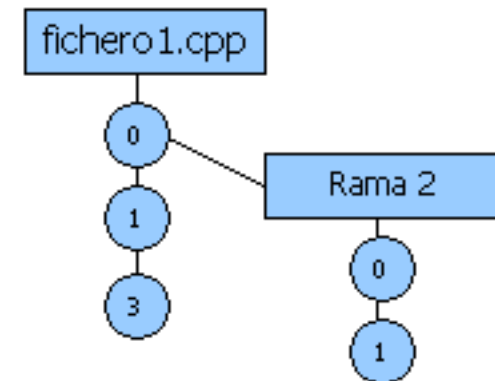


Elementos avanzados de un sistema de control de cambios

- Ramas (*branches*)

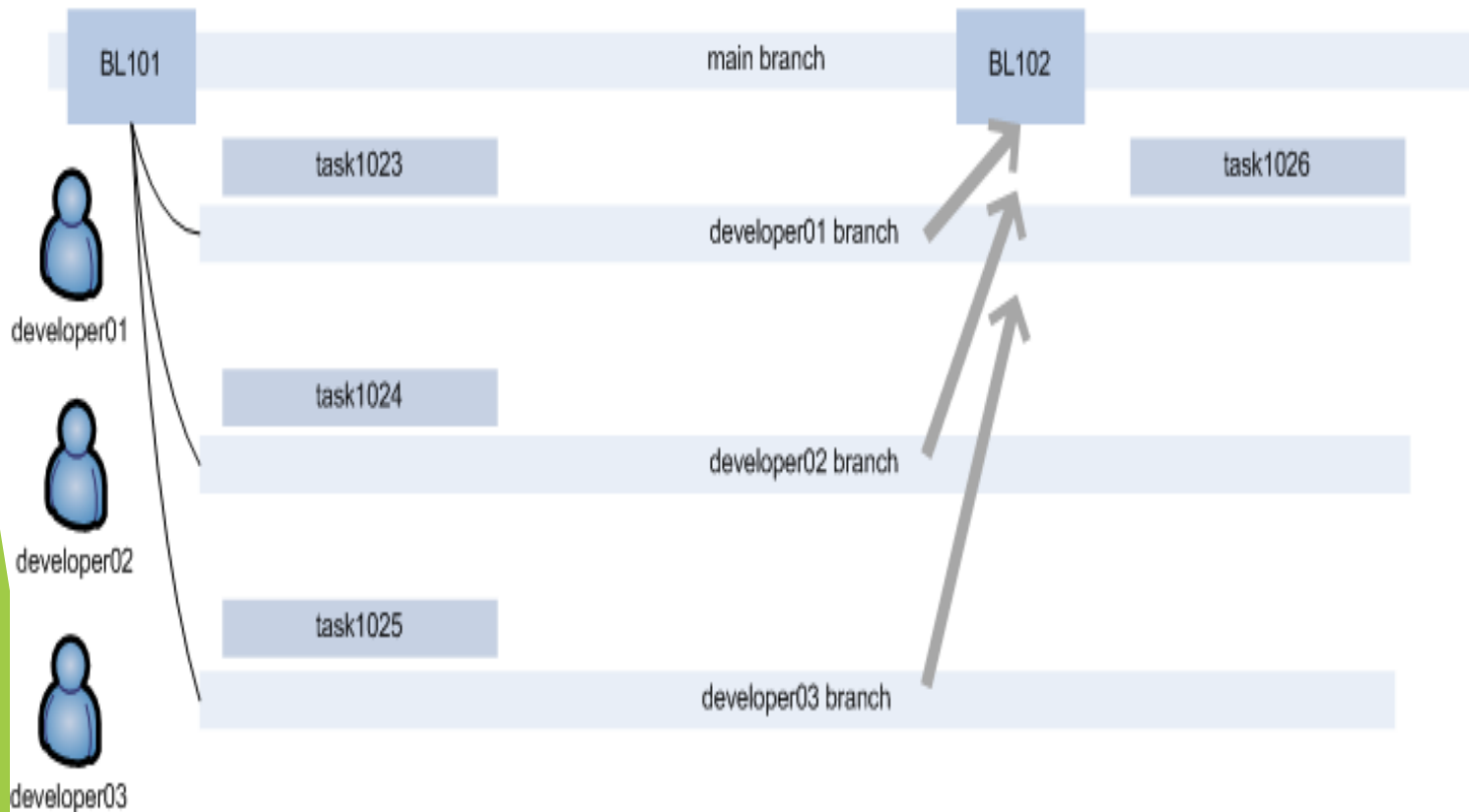
Característica que permite la implementación de muchas estrategias de gestión de la configuración

Las ramas permiten una gestión más estricta del proceso, y dan servicios adicionales. **Desarrollo paralelo**



Patrones de trabajo de mayor aislamiento

developer branch

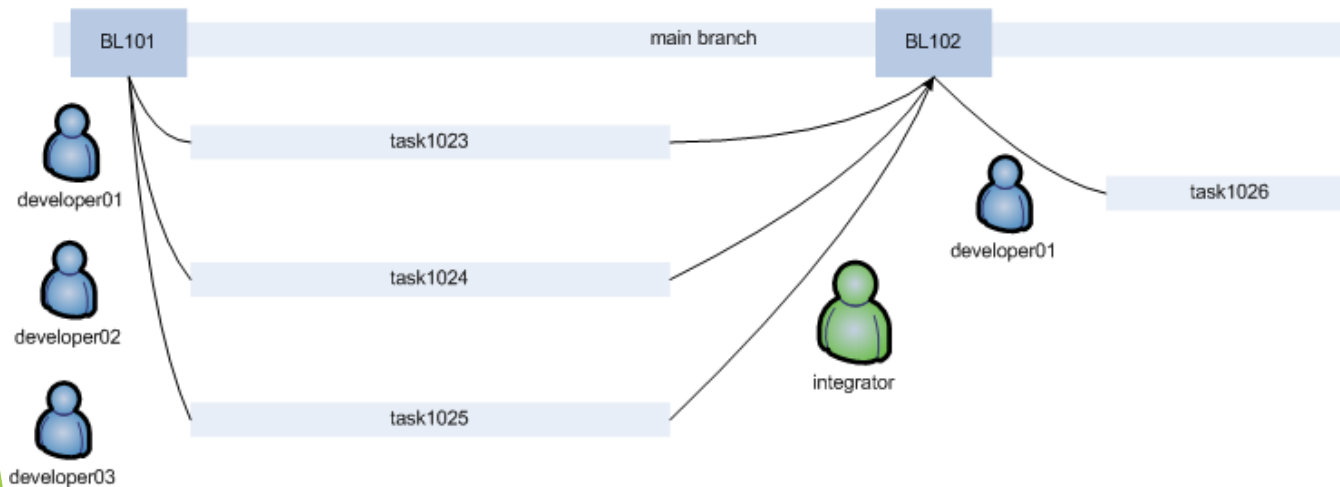


Ventajas

- Mejora el aislamiento
- Programadores con un *servicio* adicional
- Mayor control

Rama por tarea

branch per task pattern



Ventajas

- Máximo paralelismo
- Productividad
- Blindaje de la estabilidad del producto
- Trazabilidad total

Patrones de creación de ramas

Appleton, B., Berczuk, S., Cabrera, R., Orestein, R.: Streamed Lines: Branching Patterns for Parallel Software Development. In Proceedings of the 1998 Pattern Languages of Programs Conference, PLoP'98.

Dimensiones de *branching*

- Física: se *ramifican* ficheros, componentes y subsistemas
- Funcional: para unidades entregables (correcciones, construcciones, ...)
- De entorno: para distintas plataformas, sistemas operativos
- Organizativa: estructurar el trabajo, coordinar
- Procedimental: comportamiento del equipo

Algunos patrones de creación de ramas

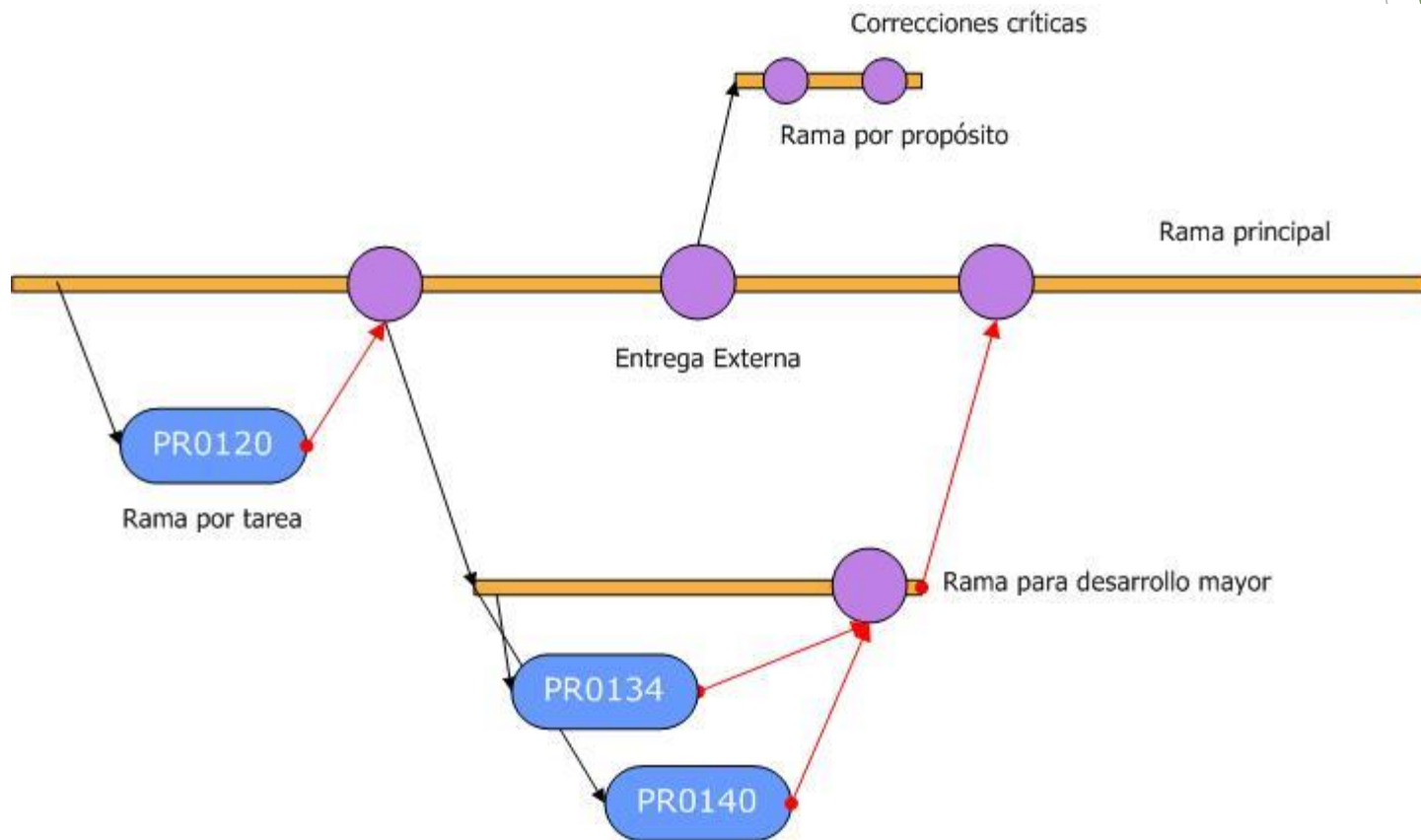
Tipos de patrones

- Patrones de política de trabajo (cuando hacer check in, propagar ...)
- Patrones de creación de ramas (subproyecto)
- Patrones de estructuración (rama para desarrollo externo)

Ejemplos de patrones

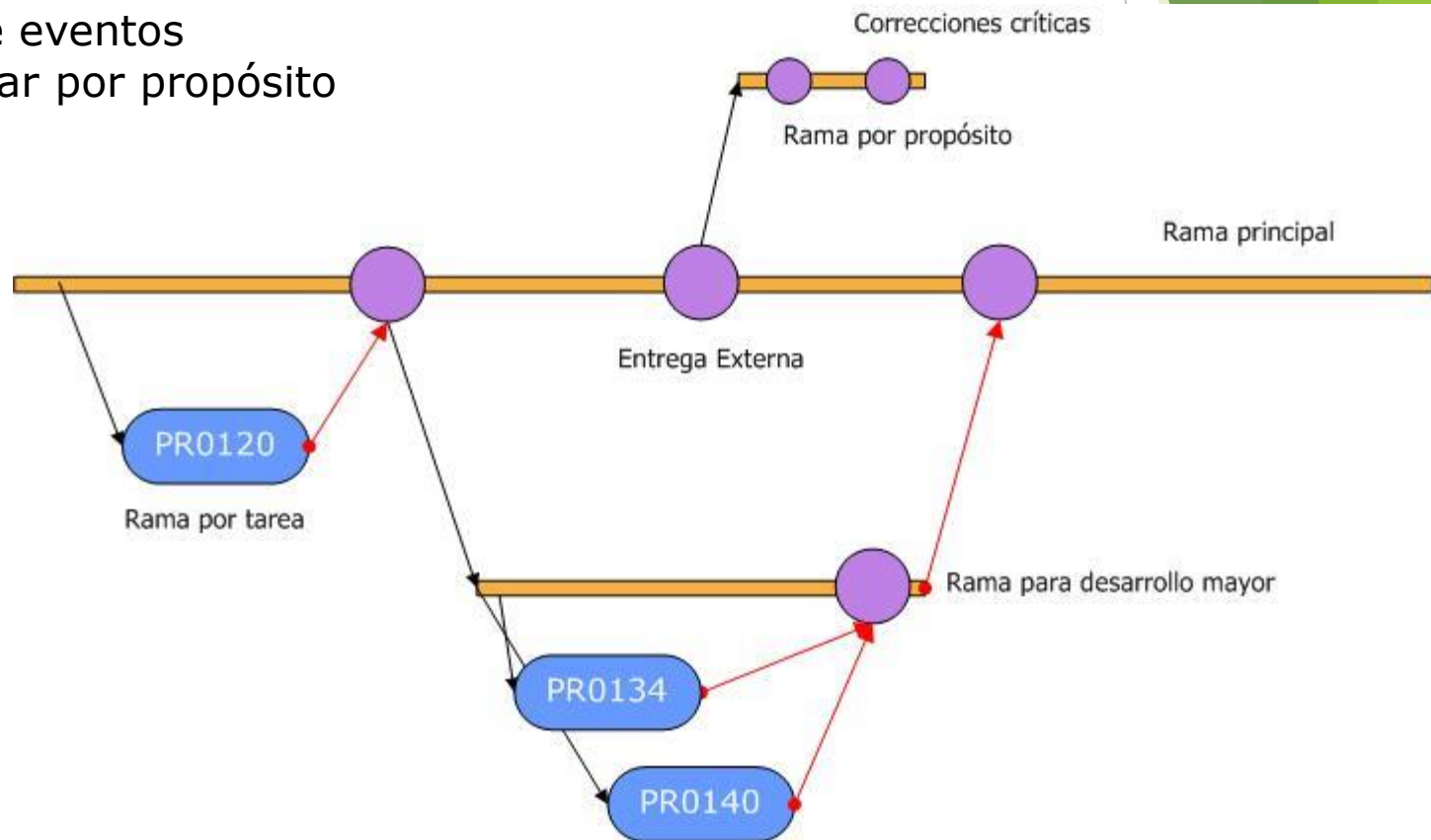
- Rama de proyecto
- Rama de mantenimiento
- Ramificar por tarea
- Soporte de diferentes plataformas

Patrones de creación de ramas



Patrones recomendados

- En tareas diarias
 - Ramificar por tarea
 - Ramificar por desarrollo mayor
- Política ante eventos
 - Ramificar por propósito



Las funciones y actividades básicas de **la administración de la configuración** son:

- Control de la configuración
- Administración de cambios
- Identificación de la configuración
- Registro del estado de la configuración
- Versiones, revisiones deltas branches, código condicional

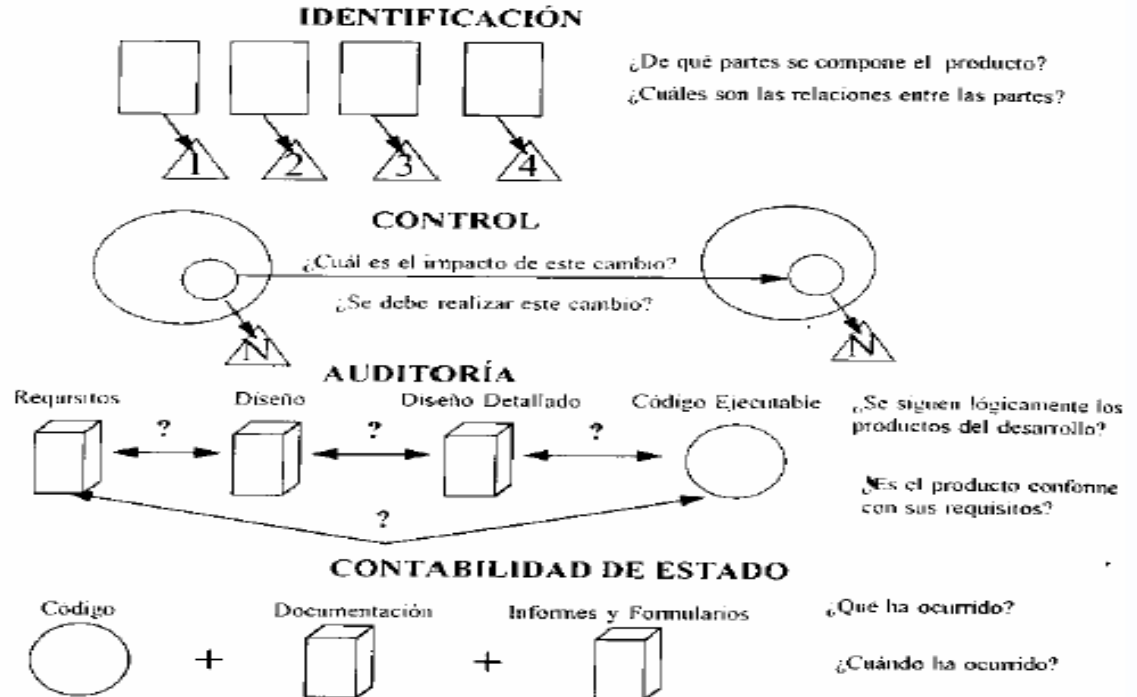
La **administración de la configuración** debe proveer:

- Control de la línea base, con servicio flexible a programadores
- Protección contra cambios no autorizados
- Capacidad para que programadores puedan modificar y testear el código
- Reserva (lockeo) de componentes

El proceso de GCS

Las principales actividades del proceso de GCS son:

- Identificación de elementos en la configuración del software
- Control de versiones
- Control de cambios
- Auditorías de configuración
- Generación de Informes



Identificación de elementos

Para poder controlar y gestionar los elementos de configuración, se debe identificar cada uno de ellos en forma única.

Tipos de objetos

- **Elementos básicos** (un listado fuente de un módulo, un conjunto de casos de prueba que se usan para probar el código)
- **Elementos compuestos** (colección de objetos básicos y otros objetos compuestos)

OBJETIVO: Identificar la estructura del SW

PREGUNTAS:

¿Cual es la configuración del SW?

¿Que versión de archivo es esta?

¿Cuales son los componentes del SW?

Elementos de configuración

- Programas de computadora: tanto en forma de código fuente como ejecutable.
- Documentos que describen los programas de computadora
- Datos: contenidos en el programa o externos a él.

Ejemplos

1. Especificación del sistema
2. Plan de Proyecto de Software
3. Especificación de Requerimientos (ERS)
 1. Modelos de análisis gráficos
 2. Especificaciones de proceso
 3. Prototipos
 4. Especificación matemática
4. Especificación de diseño
 1. Descripción del diseño de datos
 2. Descripción del diseño arquitectónico
 3. Descripciones del diseño de los módulos
 4. Descripciones del diseño de las interfaces
 5. Descripciones de los objetos

Elementos de configuración

5. Listados del código fuente
6. Especificación de las pruebas
 1. Plan y procedimiento de prueba
 2. Casos de prueba y resultados registrados
7. Manuales de operación y de instalación
8. Programa ejecutable
 1. Código ejecutable de módulo
 2. Módulos enlazados
9. Descripción de la base de datos
 1. Esquema y estructura de archivos
 2. Contenido inicial
10. Manual del usuario
11. Documentos de mantenimiento
 1. Informes de problemas del software
 2. Peticiones de mantenimiento
 3. Órdenes de cambio de ingeniería
12. Estándares y procedimientos
13. Versiones de editores, compiladores y herramientas CASE

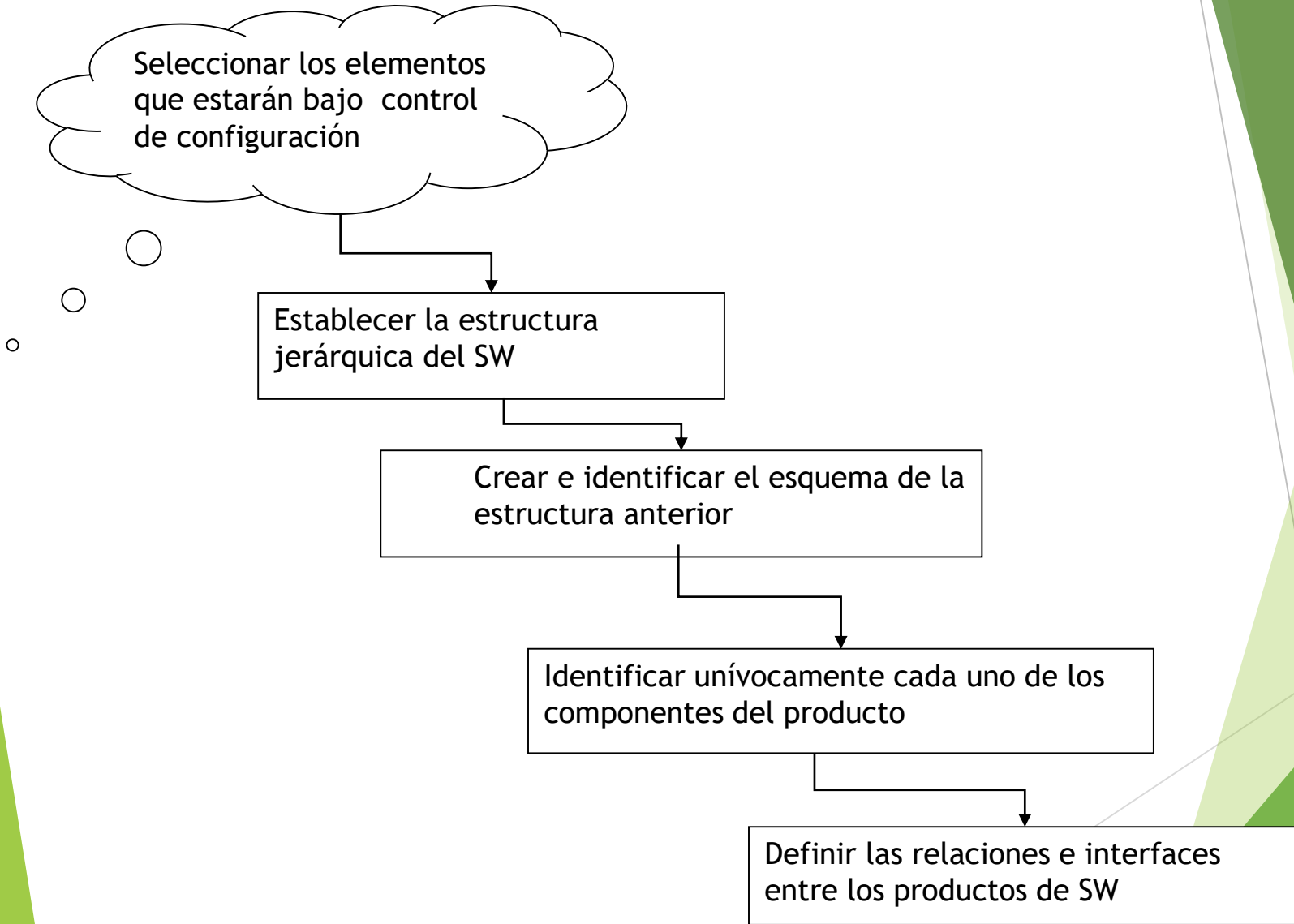
Seleccionar los elementos
que estarán bajo control
de configuración

Establecer la estructura
jerárquica del SW

Crear e identificar el esquema de la
estructura anterior

Identificar unívocamente cada uno de los
componentes del producto

Definir las relaciones e interfaces
entre los productos de SW



Versión es la Forma particular que adopta un objeto en un contexto dado

1. Cómo se pueden establecer las referencias de todos los módulos, documentos, casos de prueba de la versión 1.4?
2. Cómo se puede saber qué usuarios tienen en la actualidad la versión 2.1?
3. Cómo se puede estar seguro de que los cambios en el código fuente de la versión 2.1 han sido reflejados adecuadamente en la correspondiente documentación de diseño?

Un elemento clave para responder estas preguntas es la IDENTIFICACIÓN

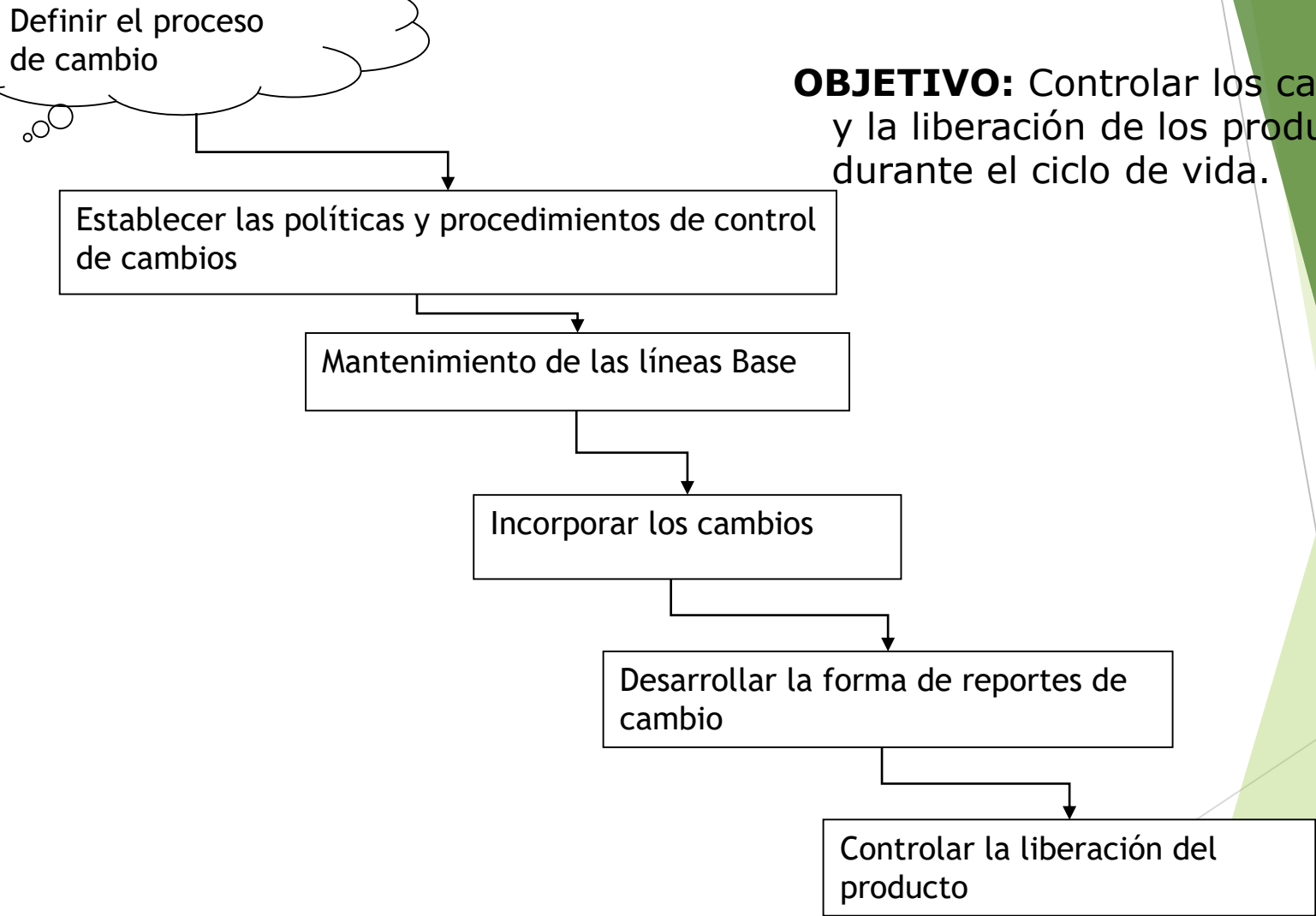
Un **cambio** es el paso de una versión de la línea base a la siguiente. Puede incluir modificaciones del contenido de algún componente, y/o modificaciones de la estructura del sistema, añadiendo o eliminando componentes.

En un proyecto de desarrollo de software, el cambio no controlado lleva rápidamente al caos. El control de cambios combina procedimientos humanos y herramientas automáticas para proporcionar una respuesta a este problema.

El control de cambios incluye:

- **CONTROL DE ACCESO:** gobierna los derechos de los ingenieros del software a acceder y modificar objetos de configuración concretos.
- **CONTROL DE SINCRONIZACIÓN:** asegura que los cambios en paralelo, realizados por personas diferentes, no se sobreescriben mutuamente.

Definir el proceso
de cambio



```
graph TD; A([Definir el proceso de cambio]) --> B[Establecer las políticas y procedimientos de control de cambios]; B --> C[Mantenimiento de las líneas Base]; C --> D[Incorporar los cambios]; D --> E[Desarrollar la forma de reportes de cambio]; E --> F[Controlar la liberación del producto];
```

The diagram illustrates a five-step process for change control. It begins with a cloud-shaped box labeled 'Definir el proceso de cambio'. An arrow points down to a rectangular box 'Establecer las políticas y procedimientos de control de cambios'. From there, the process continues through 'Mantenimiento de las líneas Base', 'Incorporar los cambios', 'Desarrollar la forma de reportes de cambio', and finally 'Controlar la liberación del producto'. The steps are arranged in a descending staircase pattern. The background features abstract green and yellow geometric shapes on the right side.

Establecer las políticas y procedimientos de control
de cambios

Mantenimiento de las líneas Base

Incorporar los cambios

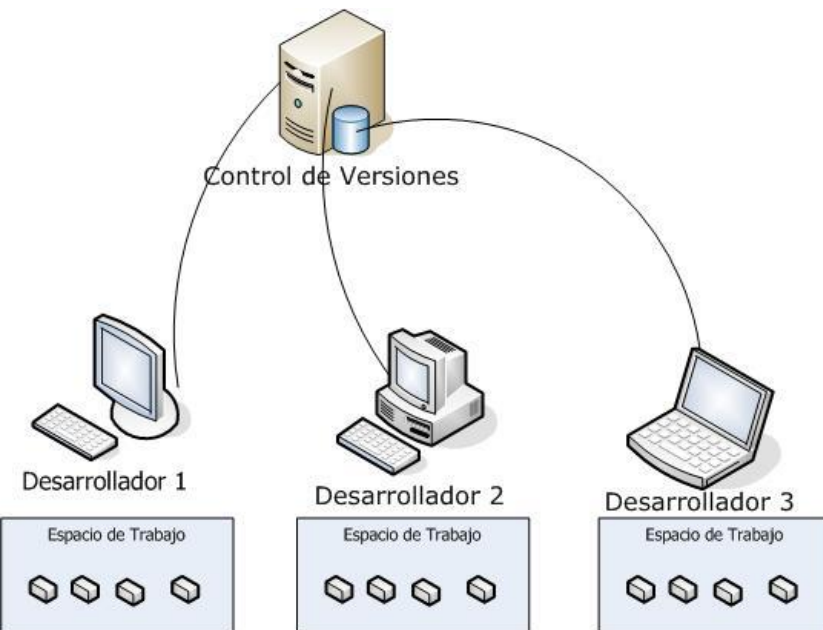
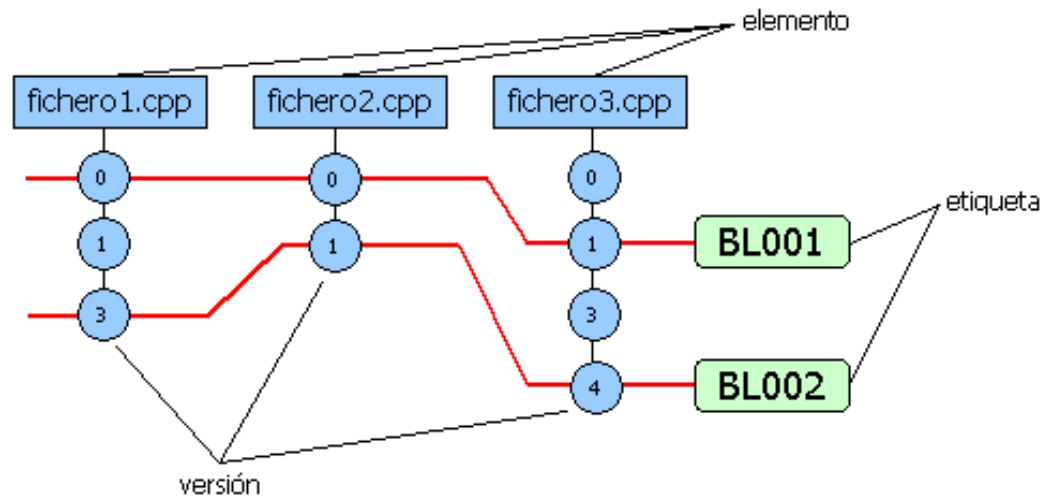
Desarrollar la forma de reportes de
cambio

Controlar la liberación del
producto

OBJETIVO: Controlar los cambios
y la liberación de los productos
durante el ciclo de vida.

Elementos básicos de un control de cambios

- Elementos
- Versiones / Revisiones
- Etiquetas



- Espacios de trabajo

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

- 1) **Revisiones técnicas formales** : se centran en la corrección técnica del elemento de configuración que ha sido modificado.
- 2) **Auditorias de configuración del software:** complementa la revisión técnica formal

1. ¿Se ha hecho el cambio especificado en la “Orden de Cambio Ingeniería”?
2. ¿Se han incorporado modificaciones adicionales?
3. ¿Se ha llevado a cabo una revisión técnica formal para evaluar la corrección técnica?
4. ¿Se han seguido adecuadamente los estándares de ingeniería del software?
5. ¿Se han especificado “la fecha del cambio y el autor”?
6. ¿Se han seguido procedimientos de Gestión de Configuración de Software para señalar el cambio, registrarlo y divulgarlo?
7. ¿Se han actualizado adecuadamente todos los Elementos de Configuración relacionados?

Auditoría de configuración

- **OBJETIVO** Verificar que el producto de SW integrado satisface los requerimientos estándares o acuerdos contractuales y que los componentes que se integran corresponden con las versiones vigentes.
- **META** Verificar que todos los productos de SW han sido producidos descritos e identificados correctamente y que todas las solicitudes de cambio han sido procesadas.

Se generan Informes de Estado cada vez que:

- Se asigna una nueva identificación a un elemento
- La Autoridad de Control de Cambios aprueba un cambio
- Se lleva a cabo una auditoría
- Regularmente, a fin de mantener al equipo al tanto de los cambios importantes

- ¿Que pasó?
- ¿Quién lo hizo?
- ¿Cuándo pasó?
- ¿Qué mas se vio afectado

Tipos de releases

- ▶ Todo release cae dentro de una de estas dos categorías, dependiendo del tipo de modificaciones realizadas:
- ▶ **Release mayor:** Implica un cambio de arquitectura y/o una modificación mayor en un conjunto de procesos o funcionalidades.
- ▶ **Release menor:** Modificación en un componente o funcionalidad particular.
- ▶ Cualquiera de las categorías anteriores puede incluir además **bug fixing** (corrección de errores). Una corrección de errores siempre se considera una modificación menor.
- ▶ Se considera un tipo de release denominado **parche**, cuyo objetivo es introducir en la release actual la corrección de errores si la necesidad de generar una nueva release. En general, los parches se incluyen de forma automática en la release inmediata posterior.
- ▶ Dado que esta clasificación **no es formal**, la categorización de una release debe ser consensuada y analizada (tiempo de desarrollo, cambio de funcionalidad, costo).

- La nomenclatura de los releases será la siguiente:

Rmm-nn donde:

R: es una letra fija. Indica que se trata de un release.

mm: Número de release mayor. Se utiliza numeración sucesiva a partir de 01, siempre con dos dígitos.

nn: Número de modificación menor. Se utiliza numeración sucesiva a partir de 00, siempre con dos dígitos.

- Ejemplo: R01-00 ► R01.01 ► R01.02 ► R02.00

Ambiente de desarrollo

- ▶ El ambiente de desarrollo es el ámbito de codificación, compilación y test unitario. Es propiedad exclusiva del desarrollador.
- ▶ La creación del ambiente de desarrollo es responsabilidad del desarrollador.
- ▶ Un ambiente de desarrollo debe ser inicializado con el correspondiente versión de release que será modificada.
- ▶ Todo componente a modificar debe ser previamente reservado mediante la operación de check-out y descargado sobre el ambiente de desarrollo.

Test Unitario

- ▶ Durante las sucesivas modificaciones de un componente, no es necesario realizar el check-in de sus diferentes versiones. No existe una separación formal entre codificación y test Unitario.
- ▶ Para casos particulares, se podrían tener en configuración las diferentes versiones probadas en forma unitaria.

Ambiente de integración

- ▶ Es donde el responsable de realizar la integración (en gral. es mismo desarrollador) recompila todos los componentes modificados que formarán parte de una entrega de software (generalmente una nueva release).
- ▶ El resultado de esta compilación será probada en integración y posteriormente entregada al cliente.
- ▶ Es condición necesaria para que un componente sea incluido en el test de integración, que esté incluido en la herramienta de control de versiones (CVS).
- ▶ El conjunto de archivos/revisiones que forman parte cada modificación, debe quedar registrado

- ▶ **SCM efectivo** no es solo tener una **herramienta** que registre **quién** hace que cambios al código o documentación y **cuándo**. Es también la creación de **procedimientos, políticas y convenciones de nombres** para asegurar que todas las partes relevantes están involucradas en los cambios de software.
- ▶ **SCM** no es solo un conjunto de practica estándares que se aplican uniformemente a los proyectos, esta practicas deben ser ajustadas a la media de cada proyecto, de acuerdo a sus características (tamaño, volatilidad, proceso desarrrrollo, etc).

- ▶ El mejor lugar para registrar como realizar SCM para un proyecto, es el Plan de SCM.
- ▶ El **Plan de SCM** documenta **que actividades** de SCM van a ser llevadas a cabo, **quiénes** son responsables de realizar las distintas actividades, **cuándo** van a suceder, y qué **recursos** son requeridos

- ▶ El plan de SCM describe las ***políticas y prácticas*** a ser utilizadas en el proyecto para implementar el SCM: versiones, variantes, workspaces, procedimientos para la administración del cambio, builds y administración de releases.
- ▶ El plan de SCM define las ***reglas y responsabilidades***.
- ▶ El plan de SCM es parte de ***plan de desarrollo de software***.
- ▶ ***No existen reglas estrictas.***
- ▶ ***Hay que discutir que es lo que mejor se ajusta en general, y para cada producto en particular.***