Gestión de Configuración

* Es la disciplina que nos asegura mantener la integridad de nuestro producto

¿Por qué pasa? Porque el sw cambia, si no cambia es porque no se usa.

¿Por qué cambia? Cambios de rq, incompatibilidades de sw, errores de sw, el negocio cambia la prioridad

* Es transversal a todo el proyecto
* Es propio de incluirla en el proceso de fabricación de un sistema
* Surge en 1950s
* IDENTIFICAR Y DOCUMENTAR
* ITEMS DE CONFIGURACION
* CONTROL DE CAMBIOS
* AUDITAR QUE SE CUMPLA TODO

¿Por qué gestionar la configuración? Establecer y mantener la integridad de los productos de sw a lo largo de su ciclo de vida

Involucra: Identificarla en un momento dado – Controlar sistemáticamente los cambios – Mantener su integridad y orden

INTEGRIDAD DEL PRODUCTO 🡪 satisface las necesidades del usuario – Puede ser fácil y rastreable durante su ciclo de vida – satisface criterios de performance – Cumple con sus expectativas de costo

PROBLEMAS EN EL MANEJO DE COMPONENTES 🡪 Perdida de un componente, perdida de cambios, sincronía fuerte-objeto-ejecutable, regresión de fallas, doble mantenimiento, superposición de cambios, cambios no validos

CONCEPTOS CLAVE:

ITEM DE CONFIGURACION: Cada uno de los artefactos que forman parte del producto o del proyecto, que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución. EJ: Propuestas de cambios, código fuente, casos de prueba, documento de despliegue, plan de calidad, requerimientos, etc.

VERSION: la forma particular de un artefacto en un instante o contexto dado. El control de versiones se refiere a la evolución de un único ítem de configuración o de cada IC por separado. Se puede representar con un grafo.

LA CONFIGURACIÓN DEL SOFTWARE: un conjunto de ítems con su correspondiente versión en un momento determinado.

Identificación de la Línea Base: se utilizan etiquetas para marcar la baseline, no confundir con la versión del producto. Es una configuración que ha sido revisada formalmente y sobre la que se ha llegado a un acuerdo. Sirve como base para desarrollos posteriores y puede cambiarse a través de un procedimiento formal de control de cambios. Si algo malo pasa, podemos regresar a esta línea base. Pongo esta etiqueta para delimitar la línea base.

No necesariamente tiene que existir código para que se haga una línea base. Tampoco es necesario que existan todos los componentes, puedo tener solo una parte del sw.

Puede ser:

* De especificación (Requerimientos, Diseño)
* De productos que han pasado por un control de calidad definido previamente.

VARIANTE: es una versión de un ítem de configuración (o de la configuración), que evoluciona por separado. Representan configuraciones alternativas. Un producto de sw puede adoptar distintas formas de configuración dependiendo del lugar donde se instale, pro ejemplo en Windows, MAC o Linux.

RAMAS

Existe una rama principal, sirven para bifurcar el desarrollo. Pueden surgir para experimentar algo alternativo. Permiten experimentar. Las puedo descartar o integrar.

Integración de ramas 🡪 La operación se llama merge, lleva esos cambios a la rama principal, pueden surgir conflictos al hacerlo (se resuelven con diff). Y las ramas o son integradas o son descartadas.

REPOSITORIO: mantiene la historia de cada IC con sus atributos y relaciones. Usado para hacer evaluaciones de impacto de los cambios propuestos. Pueden ser una o varias BD.

Tengo dos operaciones:

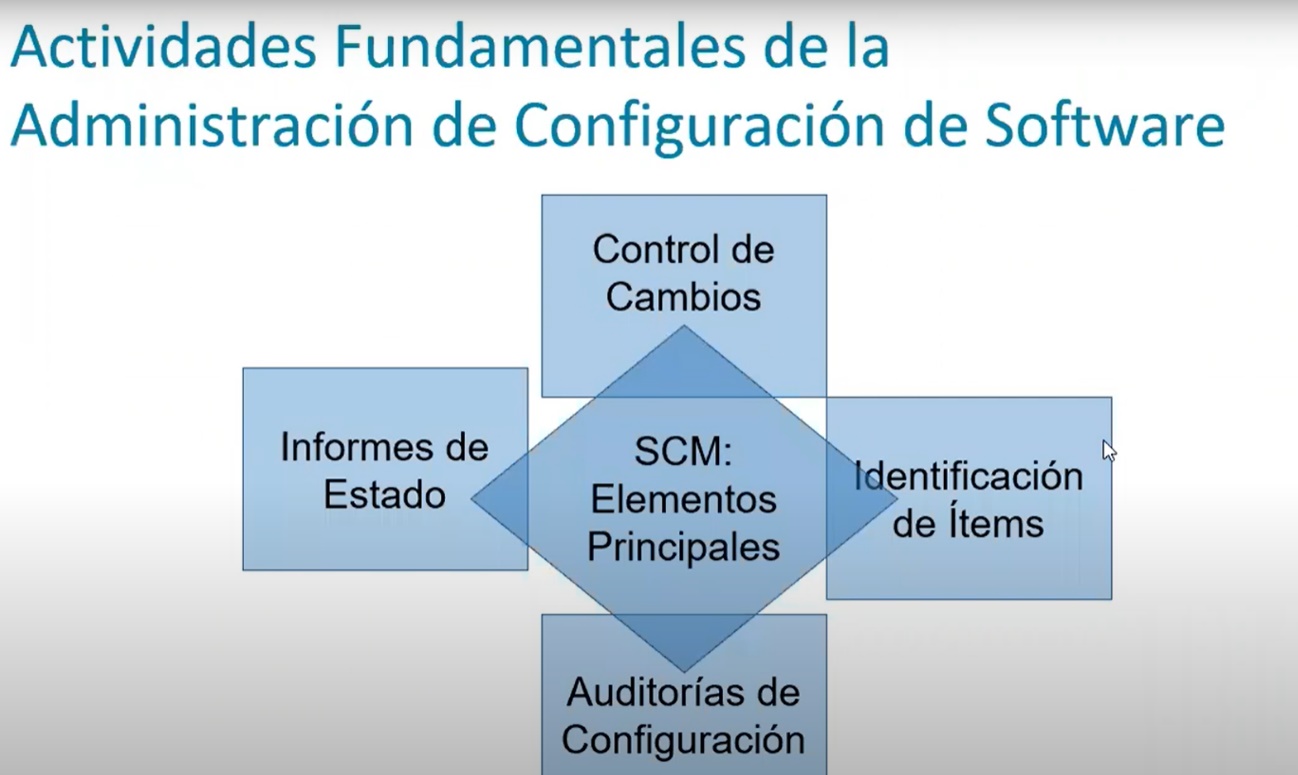
CHECK IN 🡪 DEVOLUCION

CHECK OUT 🡪 EXTRACCION

Puedo guardar documento, códigos, lo que sea que tenga ese IC.

DOS TIPOS DE TRABAJAR CON REPOSITORIOS:

* Repositorio Centralizado: Un servidor tiene todos los archivos con sus versiones, los administradores tienen control sobre el repositorio, falla el servidor y cagamos todos. De a uno puede hacer el check in y check out.
* Repositorio Descentralizado: Cada cliente tiene una copia exactamente igual del repositorio completo. Si alguno falla, solo hay que hacer ctrl c ctrl v. Posibilita otros workflows no disponibles en el centralizado.

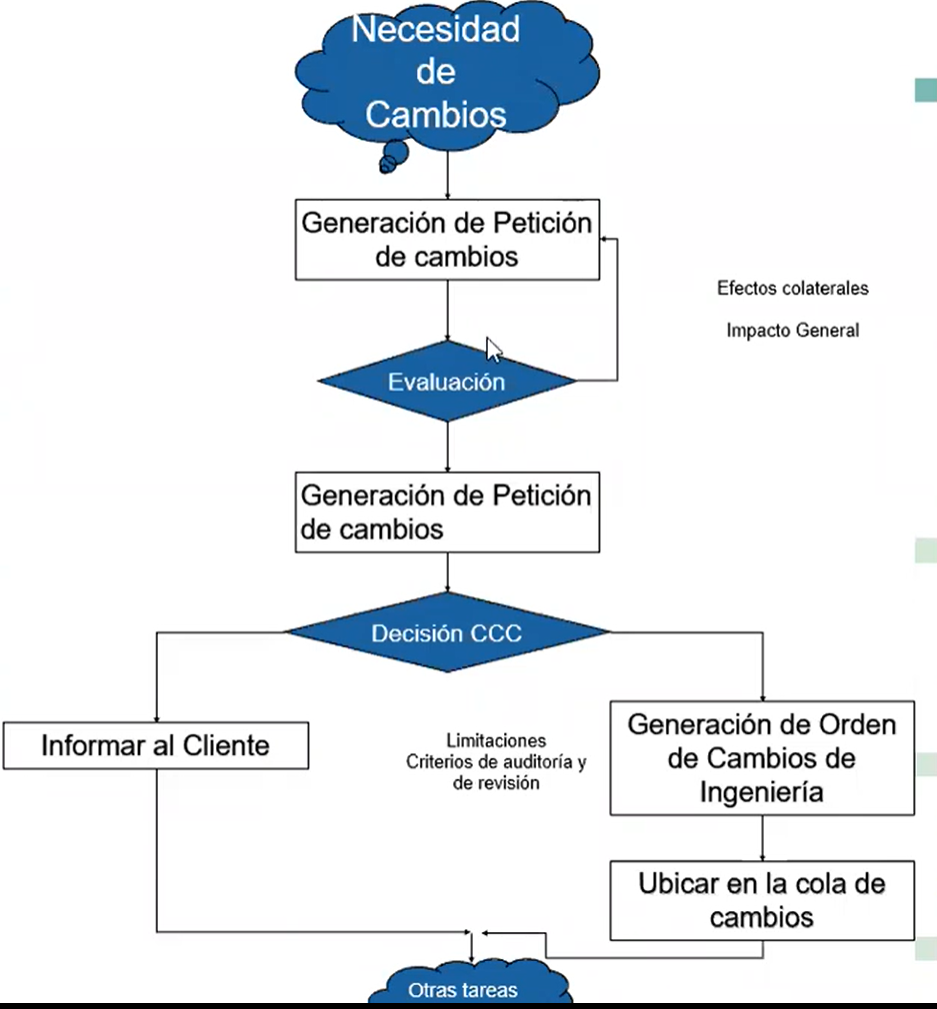


IDENTIFICACION DE IC:

* Identificación univoca de cada IC
* Convenciones y reglas de nombrado
* Definición de la Estructura del Repositorio
* Ubicación dentro de la estructura del repositorio.

CONTROL DE CAMBIOS:

* Cuando queremos cambiar una línea base.
* Tiene su origen en un Requerimiento de cambio a uno o varios ítems de Configuración que se encuentran en una línea base. Es un procedimiento formal que involucra a diferentes actores y una evaluación del impacto del cambio.
* Se define un comité de control de cambios formado por representantes de todas las áreas involucradas en el desarrollo: ANALISIS, DISEÑO, IMPLEMENTACION, TESTING, OTROS INTERESADOS. Se define desde un comienzo. Y lo convoco cuando me surgen estos cambios.
* Se recibe la petición de cambios
* Se trabaja analizando la propuesta de cambio, el análisis de su impacto, la revisión de las partes, y se decide si se acepta o se rechaza. Y esto se notifica, sea cual sea la resolución.
* Lo que se busca evaluar es el impacto de ese cambio sobre la línea base.



AUDITORIAS DE CONFIGURACION DE SOFTWARE:

* Están centradas en dos aspectos: Auditoría Física y Funcional
* Física: Verifica que ese producto sea consistente con la documentación. Que haya respetado los nombres, donde lo guardo, como es la estructura del repositorio, etc.
* Funcional: Verifica que el producto sea consistente con los requerimientos
* Se usa para verificarlos una matriz de rastreabilidad de requerimientos.

Dos procesos básicos:

Validación: el problema es resuelto de manera apropiada que el usuario obtenga el producto correcto. Se usa en la verificación funcional

Verificación: asegura que un producto cumple con los objetivos preestablecidos, definidos en la documentación de líneas base. Todas las funciones son llevadas a cabo con éxito y los test cases tengan su status “ok” o bien consten como “problemas reportados” en la nota de reléase. Se usa en la verificación física.

INFORMES DE ESTADO:

* Se ocupa de mantener los registros de la evolución del sistema
* Maneja mucha información y salidas por lo que se suele implementar dentro de procesos automáticos.
* Incluye reportes de rastreabilidad de todos los cambios realizados a las líneas base durante el ciclo de vida

Responde preguntas como: cual es el estado del ítem?, un rq de cambio ha sido aprobado o no?, que versión del item implementa un rq de cambio aprobado?, cial es la diferencia entre dos versiones?.

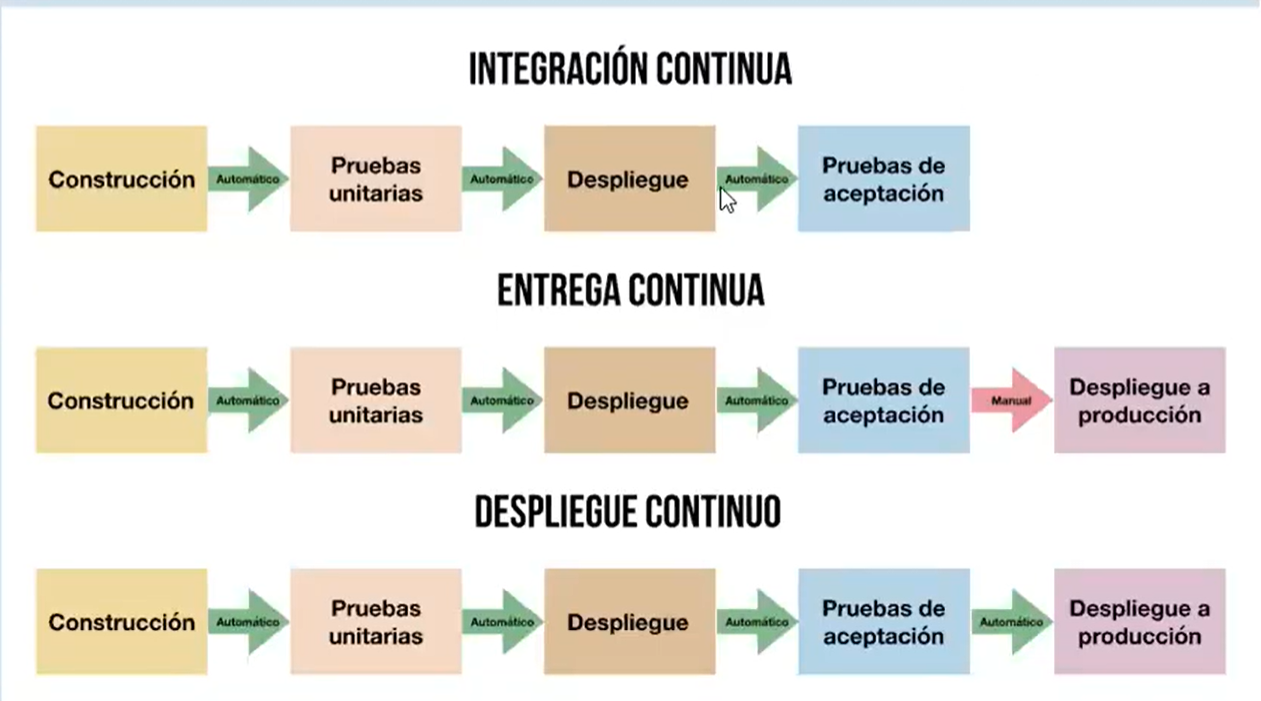
PLAN DE GESTION DE CONFIGURACIÓN

Se incluye en el plan:

* Reglas de nombrado de los IC
* Herramientas a utilizar para SCM
* Roles e integrantes del comité
* Procedimiento formal de cambios
* Plantillas de formularios
* Procesos de auditoría

EVOLUCION DE LA GESTION DE CONFIGURACION DE SOFTWARE:

* Integración Continua
* Entrega Continua
* Despliegue Continuo



GESTION DE CONFIGURACIONES DE SOFTWARE EN AMBIENTES AGILES

SCM sirve al equipo de desarrollo

Hace seguimiento y coordina el desarrollo en lugar de controlar a los desarrolladores

Responde a los cambios en lugar de tratar de evitarlos

Esforzarse por ser transparente y sin fricción automatizando tanto como sea posible

Coordinación y automatización frecuente y rápida

Eliminar el desperdicio no agregar nada más que valor

Documentación lean y trazabilidad

Feedback continuo y visible sobre calidad estabilidad e integridad

Es responsabilidad de todo el equipo

Automatizar lo más posible

Educar al equipo

Tareas de scm embebidas en las demás tareas requeridas para alcanzar el objetivo del sprint