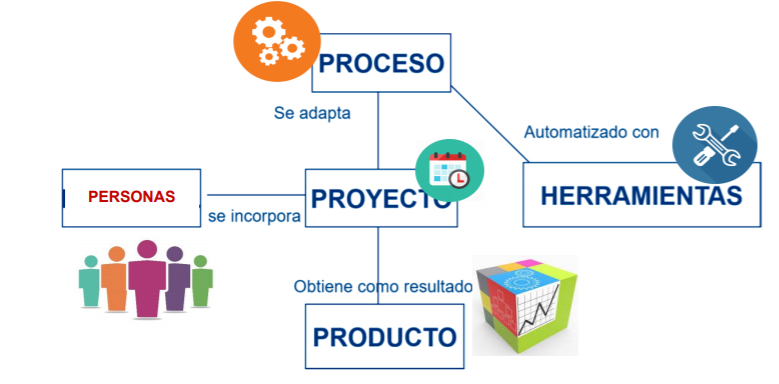
**Software Configuration Management (SCM) – Videos**

Se usa mucho en las clases prácticas.

Tenemos un proceso definido o empírico que se adapta o se instancia particularmente en proyectos, que toman como base lo que está definido en ese proceso, pero se instancia en cada proyecto. Donde una primicia es que cada proyecto es único, es decir, tiene un principio, un fin y un objetivo único. Que finalmente, cada proyecto tiene como resultado un producto de SW, y que para ejecutar ese proyecto necesitamos un conjunto de personas o colaboradores que funcionen entre sí formando un equipo de trabajo para trabajar en esa instancia del proyecto y obtener como resultado el producto de SW. De manera transversal tenemos distintas herramientas que ayudan a que el proceso se automatice lo más posible. A través de la automatización obtenemos innumerables beneficios a la hora de la ejecución del proyecto.



El punto es que hablábamos de lo que tiene que ver con construir un producto de SW, pero a partir de acá surge un concepto que tiene que ver que si pensamos en la palabra SW, ¿Qué se nos viene a la mente? Código, programa… Pero es verdad que lo que tiene que ver con SW en sí mismo es mucho más amplio que solo pensar en el código del producto. El SW incluye la documentación, datos, artefactos que tiene que ver por ej. con una especificación, procedimientos a tener en cuenta a la hora de implementar el SW. El SW no es solamente un conjunto de programas, es un concepto mucho más amplio.

Y en este contexto la gestión de configuración nos sirve porque va más allá que simplemente pensar en el SW como un conjunto de programas.

Sabemos que para poder construir un producto de SW tenemos que seguir un proceso que puede ser definido, empírico, pero de alguna manera tenemos que seguir un proceso para finalmente poder obtener desde el planteo inicial, la construcción del producto de SW cuya representación puede tener múltiples características (funcionales y no funcionales), muchas formas, etc.

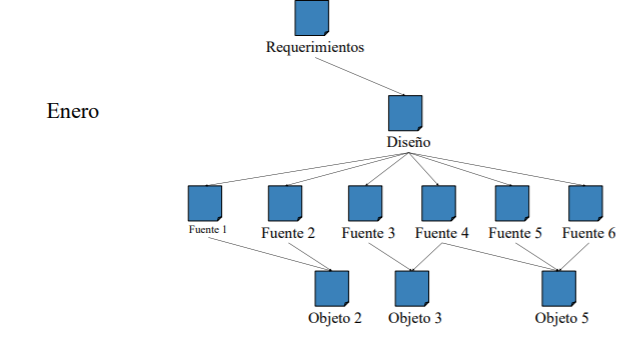
Todo esto tiene que ver con darle algo de contexto a lo que tiene que ver con de que se trata un producto de SW para poder entender por qué es importante la SCM.

Información:

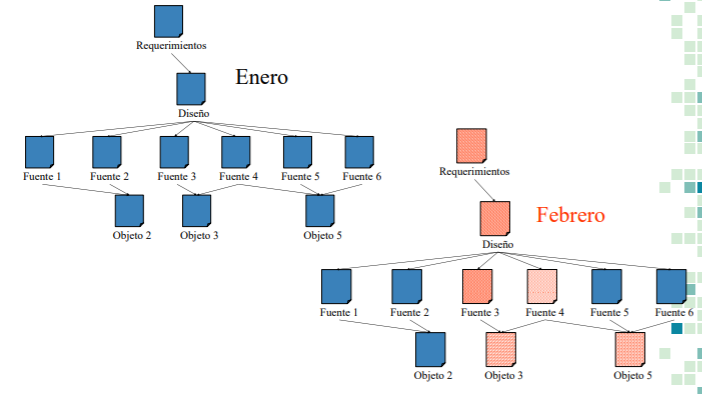
* Estructurada con propiedades lógicas y funcionales
* Creada y mantenida en varias formas y representaciones
* Confeccionada para ser procesada por computadora en su estado más desarrollado

Supongamos que nosotros tenemos un SW donde tenemos rqs, diseño, el código fuente y los datos persistentes.

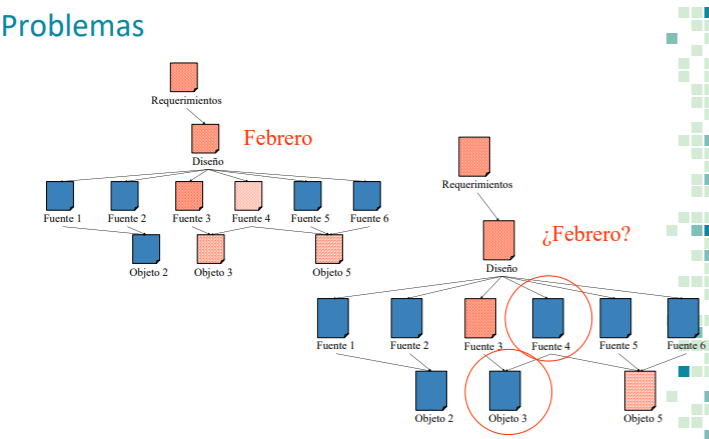
Esto es el SW en enero



Esto es el SW en febrero



Y en algún punto de febrero no sabemos bien cual es el SW.



A partir de acá aparece una premisa fundamental que nos viene a resolver el SCM, cuya palabra clave a la hora de definirla es que la SCM es la disciplina que nos permite asegurar, mantener la **integridad** de nuestro producto.

¿Qué es la integridad de nuestro producto? Tiene que ver con que lo que está planteado en la imagen no nos pase, es decir, que en un determinado tiempo tenga la posibilidad de identificar si el Fuente que tiene que estar vigente es el de arriba o el de abajo.

¿Por qué nos pasa esto? Porque el SW cambia, y esto es propio de la naturaleza del SW, si no cambia, significa que no está en uso.

¿Cuáles son los motivos por los que el SW cambia? Un montón, los más deseados son los que tienen que ver con nuevos requerimientos o cambios en el negocio. Pero también puede ocurrir que resulta que el SW corre sobre un SO que deja de estar vigente, y puede que mi SW no sea compatible con el nuevo SO. Me doy cuenta de que tengo que plantear una modificación sustancial o no. Puede que el negocio al que el SW le da soporte, cambia sus prioridades. Todas estas cosas son motivos en el cambio del SW, que en algún punto, decíamos que básicamente el SW podía clasificarse en el mantenimiento adaptativo, evolutivo, correctivo.Esos son los motivos de cambio en el SW.

Tienen su origen en

* Cambios del negocio y nuevos requerimientos
* Soporte de cambios de productos asociados
* Reorganización de las prioridades de la empresa por crecimiento
* Cambios en el presupuesto
* Defectos encontrados a corregir
* Oportunidades de mejora

**SCM como disciplina de soporte**

Es una actividad “paraguas”, transversal a todo el proyecto con aplicación en las diferentes disciplinas.

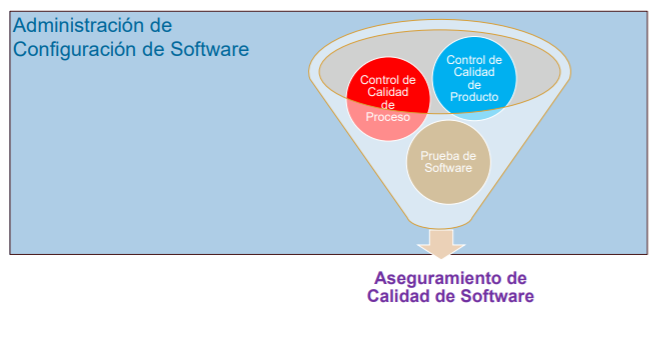
La SCM viene a ayudarnos a mantener la integridad del producto de SW a lo largo del tiempo, sabiendo que los cambios en el mismo son algo habitual.

¿Cómo describimos esta disciplina? Es una actividad paraguas (de soporte) y que es transversal, justamente a todo el proyecto o muchas otras disciplinas que trabajan al momento de construir el producto de SW.

¿Por qué? Porque esta actividad de administración de configuración de SW tenemos que tenerla en cuenta y aplicarla a lo largo de muchas actividades de manera transversal, para que la integridad del producto del SW pueda mantenerse.

Es propio incluir estas actividades, dentro de las actividades de construcción del producto de SW que voy ejecutando a lo largo del proyecto.

Cuando hablamos de administración de configuración de SW hay una relación necesaria con otras actividades de aseguramiento de calidad que tienen que ver con el testing y con el control de calidad de proceso y de producto.



**Historia**

Tiene su origen a mediados de 1950s, cuando CM (por Configuration Management) originalmente utilizado para desarrollo de hardware y control de producción, fue utilizado en el desarrollo de software.

¿Cuándo surge? A medida que se empieza a usar el desarrollo de SW y HW empieza a surgir la disciplina de CM (Configuration Management), que tiene que ver con como hago para controlar la integridad de mi producto de SW, o de mi producto de SW relación con el HW que estoy construyendo. Tiene su origen a mediados del 1950. Al principio se usaba para cuestiones espaciales.

Esta disciplina se va refinando a lo largo del tiempo.

**Definición de la IEEE**

“Una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y documentar las características funcionales y técnicas de los ítems de configuración, controlar los cambios de esas características, registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos”

La definición de la IEEE, indica que la disciplina implica monitoreo, dirección administrativa y técnica. Y está apuntada, primero a la palabra clave: “Identificar y documentar las características de los **ítems de configuración**”. Después hablamos del control de cambio sobre estas características de los ítems de configuración, lo que queremos hacer es preservar la integridad de nuestro producto, y para esto necesitamos el control de cambio, y con el hecho de poder registrar y reportar los cambios y sus estados. Y finalmente, poder registrar y auditar, o poder verificar que todo esto que mencionamos anteriormente se esté cumpliendo.

Gestión de proyecto es una actividad del aseguramiento de la calidad.

**¿Por qué tenemos que gestionar la configuración?**

Su propósito es establecer y mantener la integridad de los productos de software a lo largo de su ciclo de vida.

Para la integridad del producto de SW a lo largo de todo el ciclo de vida – Misión fundamental. Hay que desmenuzar lo de la definición de la IEEE.

* Poder identificar en un momento dado cuál es la configuración del SW
* Poder controlar de manera sistemática los cambios que se hacen sobre esa configuración, justamente para poder mantener la integridad
* Mantener la integridad y su origen

**Lo que la SCM resuelve no está relacionado con una dificultad en una toma de requerimientos.**

**Pregunta:** Veo al SCM como un control de versiones, versiones que se van actualizando, ¿Es algo más que eso?Si, es algo más, pero eso de control de versiones está incluido.

**Ejemplo**

Somos un equipo de trabajo, los 5 trabajamos sobre el mismo producto de SW, y resulta que alguien en un determinado momento me pisa un cambio que estoy haciendo, y mis cambios se pierden. Esto es algo que la SCM me debería ayudar a resolver.

¿Qué es lo importante entonces? Que la SCM **no** resuelve solamente el control de versiones.

Vamos a necesitar tener en cuenta algunos conceptos específicos que hacen a la gestión de la configuración para entender a lo que nos referimos.

**Integridad del Producto**

* Satisface las necesidades del usuario. Por ej: nos equivocamos en las capturas de los requerimientos o el proceso no estuvo bien hecho, entonces no se logra la integridad del producto. El producto que estoy construyendo NO satisface las necesidades del cliente.
* Puede ser fácil y completamente rastreado durante su ciclo de vida. En cualquier momento puede hacer la trazabilidad hacia atrás para entender en qué punto del ciclo de vida estoy, que no solamente tiene que ver con un control de versiones, va un poquito más allá.

¿Qué es rastrear?

El término rastrear es una herramienta que se usa en la gestión de configuración, que se llama matriz de rastreabilidad o matriz de trazabilidad. Y que nos permite que nosotros nos podemos parar por ej. en un caso de prueba, y poder rastrear hacia atrás para identificar en ese caso de prueba cual es el caso de uso que está testeando, cual es el escenario de caso de uso que está testeando, en término de diseño, en donde impacta ese caso de prueba en el diseño del producto, y en el código, donde impacta ese caso de prueba en lo que tiene que ver con el código.

Puedo trazar desde un requerimiento puntual a lo largo de todo el ciclo de vida del producto cuál es el impacto de ese requerimiento. Me sirve fundamentalmente para cuando hago un cambio. Si yo hago un cambio en un requerimiento, como hago (con producto construido, diseno, casos de prueba y código), a través de ese cambio que planteo en los requerimientos para replicarlo a lo largo de todo lo que tiene que ver con el producto de SW, dijimos que el producto es TODO (requerimientos, diseño, casos de prueba, codigo), como hago yo para hacer esa modificación e impactar en todo el ciclo de vida.

Para poder hacer eso necesito tener esta trazabilidad o rastreabilidad.

El punto acá es que los cambios los vamos a tener que hacer, si ponemos como ejemplo el tema de los requerimientos, si hay cambios en los requerimientos, no necesariamente significa que hicimos mal la captura de los requerimientos sino que pueden haber cambiados los requerimientos en el negocio, o le cambiaron al usuario. No es que yo me equivoque, sino que eso cambió, y si los requerimientos cambiaron, tengo que tener una manera de trazar ese cambio a lo largo de todos los artefactos de mi producto para poder impactarlos.

La gestión de configuración ofrece herramientas para resolver estas problemáticas.

* Satisface criterios de performance
* Cumple con sus expectativas de costo

**¿Cuáles son los problemas que podemos tener?**

* Pérdida de un componente
* Perdida de cambios (el componente que tengo no es el último)
* Sincronía fuente - objeto - ejecutable
* Regresión de fallas
* Doble mantenimiento
* Superposición de cambios

Problemas en los que tienen que ver con el manejo de los componentes, algunos relacionados con los cambios, y otros que tienen que ver con alguna cuestión de versionamiento, de suposición de cambio o de tener en distintas fuente-objeto-ejecutable versiones diferentes.

Ya teniendo en claro cuales son las dificultades con las que nos vamos a encontrar para mantener la integridad del producto de SW cuando lo estamos construyendo, vamos a hablar de algunos conceptos claves que tiene que ver con la gestión de configuración, para después entender cómo es todo el proceso.

**Conceptos claves para la gestión de configuración de Software**



* **Ítem de configuración de SW (SCI)**

“Se llama ítem de configuración (IC) a todos y 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”

Es cada una de las unidades, o cada uno de los artefactos que forman parte del producto de SW que nosotros estamos construyendo (puede ser una especificación de requerimiento, una porción de código, un caso de prueba).

¿Cómo hago para identificar dentro de mi proyecto, de todos los documentos, código fuente, código ejecutable, cuales son los ítems de configuración?

Hay conceptos claves.

Es un ítem de configuración si:

* Puede sufrir cambios.
* Si tiene que ser compartido entre los distintos miembros de un equipo de trabajo, porque naturalmente, si necesita ser compartido, una de las alternativas es que pueda tener cambios por alguno de los miembros del equipo.
* A lo largo de la vida del producto, necesitamos conocer cómo evoluciona.

Si un artefacto reúne todas estas características, es un ítem de configuración. Y es importante, cuando nosotros empezamos a construir un producto de SW, poder identificar cuales son mis items de configuración.

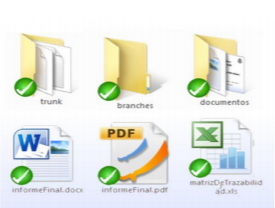
Acá surge un punto importante, yo voy a identificar los ítems de configuración antes de crearlos a los mismos, no cuando lo estoy creando. La idea es que cuando haga mi plan de gestión de configuración, yo pueda identificar cuáles son los ítems de configuración de mi proyecto.

Entonces digo: En mi proyecto para construir este producto de SW, la especificación de requerimientos va a ser un ítem de configuración, los documentos de diseño van a ser un ítem de configuración, el código fuente va a ser un ítem de configuración, el código ejecutable va a ser un ítem de configuración. Depende de cada proyecto, yo en cada proyecto defino cuales son, pero la idea es que yo pueda definirlos con anticipación a crearlos.

Es importante identificar los ítems de configuración. ¿Y qué significa identificarlos?

Identificar no es solo decir: “Los requerimientos son un ítem de configuración”. Identificar implica otras cosas mas, como por ejemplo, decir en qué lugar lo voy a guardar, decir cual va a ser la regla de nombrado, todas estas cosas la idea es que las tenga definidas antes de crearlos.

**Pregunta**: ¿Una base de datos puede ser un ítem de configuración?Si, una BD puede ser un ítem de configuración.



**Algunos ejemplos de ítems de configuración**

* Plan de CM
* Propuesto de cambio
* Visión
* Riesgos
* Plan de desarrollo
* Prototipo de interfaz
* Guia de estudio de IHM
* Manual de usuario
* Requerimientos
* Plan de calidad
* Arquitectura de SW
* Plan de integración
* Planes de iteración
* Estandares de codificacion
* Casos de prueba
* Código fuente
* Gráficos, iconos…
* Instructivo de ensamble
* Programa de instalación
* Documento de despliegue
* Lista de control de entrega
* Formulario de aceptación
* Registro del proyecto

**Pregunta**: Esta herramienta, ¿Aplica tanto a metodologías ágiles como a tradicionales?Si, aplica a las dos.

Uno de los tps (video) tiene que ver con las técnicas de gestión de configuración que se usan en las metodologías ágiles.

Claramente en un contexto ágil un plan de proyecto NO va a ser un ítem de configuración, porque en un contexto ágil no tengo un plan de proyecto. Pero la disciplina existe igual y se ejecuta igual. Existen algunas particularidades que se caen solas naturalmente.

Yo tengo una disciplina que es gestión de configuración, pero tengo identificados los valores y principios del manifiesto ágil, entonces como hago la unión entre esta disciplina de mantener la integridad de mi producto vs los principios y los valores que están en el manifiesto ágil. Que cosas hago y qué cosas pongo para poder ejecutarla. Pero la disciplina sigue existiendo en el contexto de los proyectos ágiles

Probablemente en un proyecto ágil no tengamos ni plan de proyecto, ni de integración ni de calidad. Pero si vamos a tener la arquitectura del SW, el código fuente, prototipo y los casos de prueba.

¿Entonces, que no vamos a tener en agil? Los artefactos que genero en un proyecto tradicional gestionado con un proceso definido, esos no los voy a tener porque no existen en el otro contexto.

Es por esto que es importante pensar en el contexto de nuestro proyecto cuáles son los ítems de configuración, en **mi** proyecto, independientemente que pueda tener una checklist para elegir cual si o no. Pero tienen que aplicar a **mi** proyecto, teniendo en cuenta que son artefactos que yo voy a generar, que van a ser compartidos por el equipo, que probablemente tengan cambios a lo largo de la construcción del producto y que eso hace que necesite estar sometido a la gestión de configuración.

La intención igualmente cuando hablamos de metodologías ágiles, es pensar de antemano, cuales son los ítems de configuración que yo voy a tener en mi proyecto. Los identifico (implica no solo decir “el código fuente va a ser un ítem de configuración”, sino que implica decir como lo voy a nombrar para poder identificarlo de manera única y en donde lo voy a guardar. Porque todo esto es parte del concepto de identificación de los ítems de configuración).

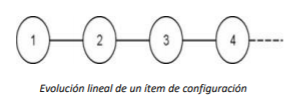
* **Versión**

Una versión se define, desde el punto de vista de la evolución, como 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 (IC), o de cada IC por separado.

La evolución puede representarse gráficamente en forma de grafo.

¿A qué nos referimos? La versión la trabajamos sobre un ítem de configuración, y decimos que la evolución de un ítem de configuración es lo que nos da lo que tiene que ver con el control de versión. El control de versión tiene que ver con la evolución de cada uno de los ítems de configuración a lo largo del tiempo. La evolución en principio para simplificar, la imaginamos como una evolución lineal.



El concepto de versión es decir, en un momento determinado la versión del ítem de configuración es la 1, en otro momento del tiempo la versión es la 2, en otro momento la 3, y en otro la 4. Yo puedo sacar una foto por decir de una manera de un ítem de configuración y esa foto es la versión justamente de ese artefacto o de ese ítem en un instante.

Por ahora, yo tengo identificados mis ítems de configuración, esos ítems van a evolucionar a lo largo del tiempo, y esa evolución a lo largo del tiempo la voy a manejar por versiones. Cada ítem va a tener una serie de versiones.

Tengo los items de configuración, y la evolución lineal de un ítem de configuración o la evolución de un ítem en el tiempo, en donde en distintos instantes puedo saber cual es el estado del ítem, entonces lo llamo versión.

**Pregunta:** Usted dijo que la gestión de configuración tenía como un conjunto de herramientas, ¿es algo estandarizado o se puede usar cualquier tipo de herramientas? Por ejemplo, con el controlador de versiones, ¿se puede usar cualquier herramienta o tiene que ser algo estandarizado?

En particular para lo que tiene que ver con el control de versiones hay muchas herramientas, algunas suelen ser más útiles para versionar código y otras para versionar documentación. En principio, hay distintas herramientas y cualquiera de ellas puede ser de utilidad (Ejemplo: Git, SharePoint).

* **Configuración del Software**

Un conjunto de ítems de configuración con su correspondiente versión en un momento determinado.

Cuando tengo un conjunto de ítems, cada uno versionado en una foto, en un momento del tiempo determinado, decimos que eso es una configuración del software.

* **Identificación de la Línea Base**
  + Se utilizan etiquetas para “marcar” las baseline
  + No confundir con la versión del Producto

Es lo que nos va a permitir poder identificar o marcar en un determinado momento del tiempo con etiquetas para saber que en ese momento tengo una configuración que es estable y a la que puedo volver en un determinado momento del tiempo. Sabemos que esa configuración conserva las características que identificamos como integridad del producto.

Entonces, no nos alcanza simplemente con identificar la configuración del software, si no que nos alcanza con identificar además la línea base.

**¿Qué es esta línea base?**

* Es una configuración (conjunto de ítems de configuración con su versión) que ha sido revisada formalmente y sobre la que se ha llegado a un acuerdo

Se llega al acuerdo de que esto es una configuración estable del producto, nos hemos puesto de acuerdo que nos puede servir como base. Si seguimos avanzando en el tiempo y en algún momento detectamos que nuestro producto no es íntegro, podemos volver a esa configuración que ya está revisada y en la que hemos llegado a un acuerdo que nos permite asegurarnos que sabemos que es lo que hay ahí.

* Sirve como base para desarrollos posteriores y puede cambiarse sólo a través de un procedimiento formal de control de cambios

Si yo quisiera hacer un cambio sobre esa línea base, necesitaría tener definido un procedimiento formal para hacer un control de cambio. ¿Por qué? Porque si yo no tengo un procedimiento formal para hacer un cambio sobre una línea base no voy a poder hacer todo lo que quiero hacer definiendo la línea base.

* Permiten ir atrás en el tiempo y reproducir el entorno de desarrollo en un momento dado del proyecto

Para repasar un poco, tenemos la versión de cada uno de los items, la configuracion que tiene un conjunto de ítems con su correspondiente versión y una línea base que es una configuración que ha sido revisada formalmente y que me sirve de base para poder seguir avanzando en la construcción de mi producto de Software.

**¿Cómo hago para identificar una línea base?** Es una etiqueta. Decimos: “Esta configuración está revisada” y le ponemos una etiqueta. Trazo una línea base y a partir de ahí sigo evolucionando mi producto. Si tengo algún problema en el futuro con mi producto, puedo volver a esa línea base y puedo reproducir en el entorno de desarrollo la foto que tengo en ese momento de la línea base porque justamente es la configuración revisada formalmente y que nosotros acordamos y sabemos que es estable.

**¿La línea base se escribe en la documentación?** En la documentación defino en donde voy a trazar la línea base. Uno puede definir una línea base en un momento donde todavía no tiene código. No necesariamente la línea base tiene que ser cuando ya tengo código. Puedo trazar la línea base en cualquier momento del tiempo. En el plan de gestion de configuracion (donde decimos cuales son los items, como los vamos a nombrar, donde los vamos a guardar) también decimos cuáles van a ser las líneas base, en qué momento del tiempo yo voy a trazar esas líneas base. En la documentación, en el plan de gestión de configuración, también tengo que identificar cuales son las líneas base.

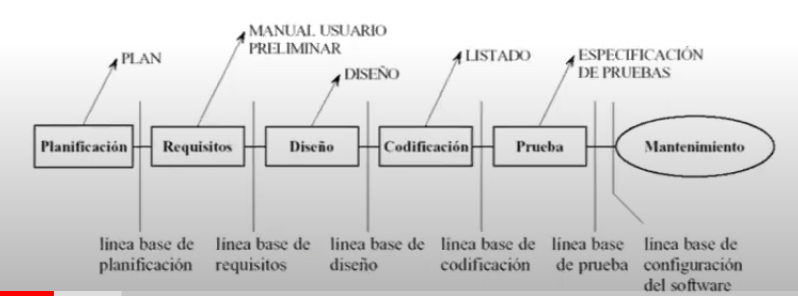
La línea base es una etiqueta de una foto de la configuración. NO necesariamente (dependiendo de qué momento del tiempo este) incluye todos los ítems de configuración. Porque si hay ítems de configuración que en el momento en el que trace la línea base todavía no existían, esa línea base todavía no lo incluye.

**Pregunta:** En un repositorio Git, ¿sería un tag la línea base? Claro, exactamente.

**Pregunta:** ¿A qué tipo de software se aplican estos conceptos? A todos. Cuanta más gente hay vinculada a la construcción del producto de software, más necesario es. Aunque haya una sola persona, lo mismo aplica.

**Representacion de Lineas Base**

Puedo tener una línea base de la planificación, una de requerimientos, una de codificación, una del producto, hay muchos tipos de líneas base.



Pueden ser:

* De especificación (Requerimientos, Diseño)
* De productos que han pasado por un control de calidad definido previamente
* **Variante**
  + Una variante es una versión de un ítem de configuración (o de la configuración) que evoluciona por separado
  + Las variantes representan configuraciones alternativas
  + Un producto de software puede adoptar distintas formas (configuraciones) dependiendo del lugar donde se instale
  + Por ejemplo, dependiendo de la plataforma (máquina + SO) que la soporta, o de las funciones opcionales que haya de realizar o no (una funciona con Windows y la otra funciona con Linux)
  + Cada una evoluciona por separado, no es que solo una evoluciona

**Pregunta:** ¿Un ejemplo puede ser que yo desarrolle un sistema y que dependiendo el mercado tiene diferentes características? Claro, exactamente.

* **Ramas**
  + Existe una rama principal (trunk, master)
  + Sirven para bifurcar el desarrollo

Es decir, yo tengo una rama que es la rama principal, que es en donde yo tengo el producto estable que estoy construyendo y empiezo a tener algunas bifurcaciones en el desarrollo, por ejemplo, si quiero probar algo, porque quiero experimentar con alguna funcionalidad que a lo mejor la agrego o a lo mejor no, o porque quiero evolucionar en el desarrollo por un determinado lugar alternativo.

Da la posibilidad de poder avanzar en estas bifurcaciones del desarrollo en una manera paralela al desarrollo del master del producto.

* + Pueden tener razones de creación con semántica
  + Permiten la experimentación
  + Pueden ser descartadas o integradas

Tiene que ver con la parte práctica, con cómo empezamos a trabajar con las herramientas de versionado y cómo empezamos a trabajar en esto de poder evolucionar el software que nosotros vamos construyendo.

**Integración de ramas**

* La operación se llama merge
* Lleva los cambios a la rama principal
* Pueden surgir conflictos (resolvemos con diff)

Cuando se sincroniza esa rama y se hace el merge, puede ocurrir que haya algunas cuestiones de consistencia que haya que resolver. Definir, cuando hay diferencias, con cuáles nos quedamos.

Supongamos que queremos integrar y hay algún conflicto, hay algo que está en la rama que quiero integrar que no es compatible con lo que está en el master, el diff nos ayuda a ver dónde está la diferencia para poder decidir cual es la que continúa y cuál es la que se descarta,el diff tiene que ver con las inconsistencias.

* Todas las ramas deberían eventualmente integrarse a la principal o ser descartadas

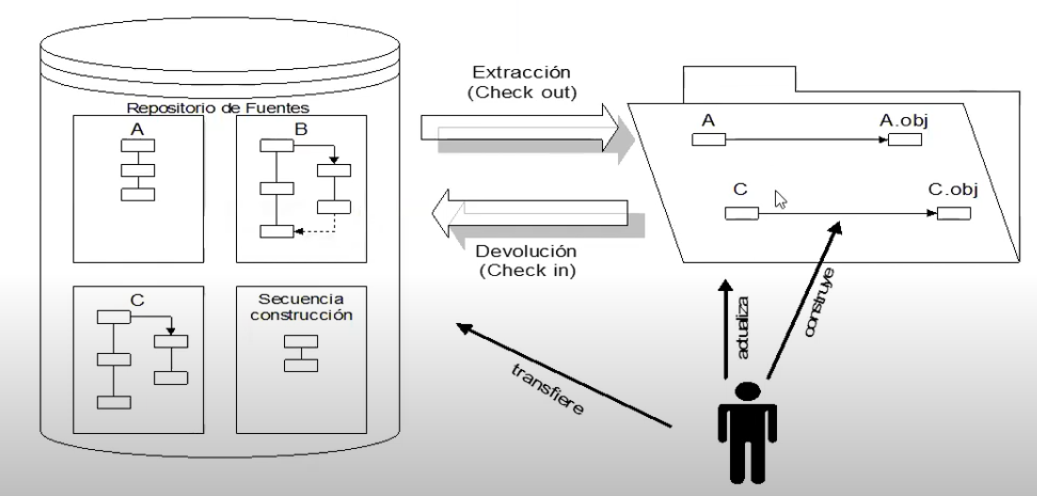
Si es que quiero integrar la rama que desarrollé al producto o a la rama principal, si no lo descarto (puede ser que quería experimentar algo que no resultó). En algún momento esas ramas tienen que unificarse con el master.

**¿Qué es un repositorio?**

* Un repositorio de información conteniendo los ítems de configuración (ICs)
* 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 bases de datos

Para cada uno de los ítems tengo que definir una identificación única porque si no después no los voy a poder encontrar y los voy a almacenar en un repositorio.

El repositorio guarda cada ítem, con toda su historia y con todo lo que tengo que ir guardando relacionado a ellos.



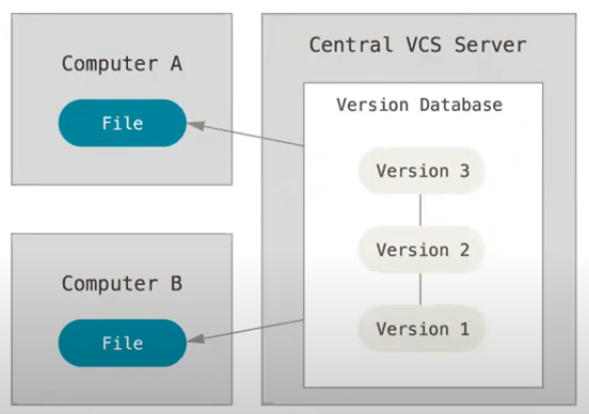
Armamos una estructura y dentro de ella vamos a tener, por un lado lo que tiene que ver con la **construcción** y por el otro lado con el **almacenamiento** finalmente en el repositorio. Hay un proceso en el medio que se llama **Check In** y **Check Out**. El Check Out es el proceso que le permite a las personas que están trabajando o desarrollando el producto de software acceder a lo que está disponible de cada uno de los ítems de configuración en el repositorio. El check In tiene que ver con aplicar los cambios que la persona hizo en el ítem de configuración o los ítems de configuración nuevos que la persona creó en el repositorio. Hay dos actividades fundamentales, Check Out y Check In.

Dentro de cada una de las carpetas o directorios vamos a ir identificando y guardando cada uno de los ítems de configuración, estos pueden ser código, especificación de requerimientos, documento, etc.

Es lo mismo que “Push and Pull” o “Commit and Update”.

Hay dos tipos de repositorios o dos tipos de maneras de trabajar con un repositorio:

* **Repositorio Centralizado**
  + Un servidor contiene todos los archivos con sus versiones
  + Los administradores tienen mayor control sobre el repositorio
  + Falla el servidor y “estamos al horno”

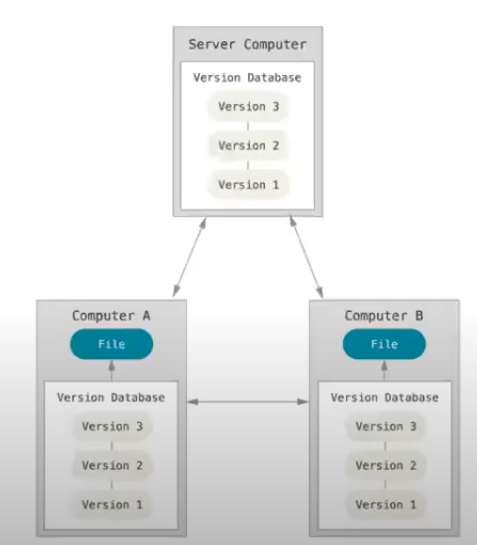


Es donde yo tengo en el repositorio todos los archivos y todas las versiones y los administradores tienen un control sobre ese repositorio, sobre lo que ahí ocurre. En lo que tiene que ver con el Check Out, se trabaja un archivo individual y cuando se termina ese archivo (o un conjunto de archivos), se suben de nuevo a ese repositorio centralizado.

¿Que tiene este tipo de estructuras? Tienen un fuerte control de los administradores sobre lo que hay en el repositorio.

¿Qué desventaja tiene? Si algo pasa en el servidor tenemos un problema. Porque en el trabajo individual no tenemos versionado este servidor.

* **Repositorios Descentralizados** 
  + Cada cliente tiene una copia **exactamente** igual del repositorio completo
  + Si un servidor falla solo es cuestion de “copiar y pegar”
  + Posibilita otros workflows no disponibles en el modelo centralizado



Cada uno trabaja con los ítems de configuración que necesita modificar y después hace el check in de las modificaciones que hizo sobre esta versión para que los demás en algún momento actualicen.

Este tipo de repositorio es el que más se utiliza. Si falta algo en un repositorio tengo forma de reconstruirlo de alguna manera. Permite el uso de workflow de interacción entre los clientes que no está en el modelo centralizado.

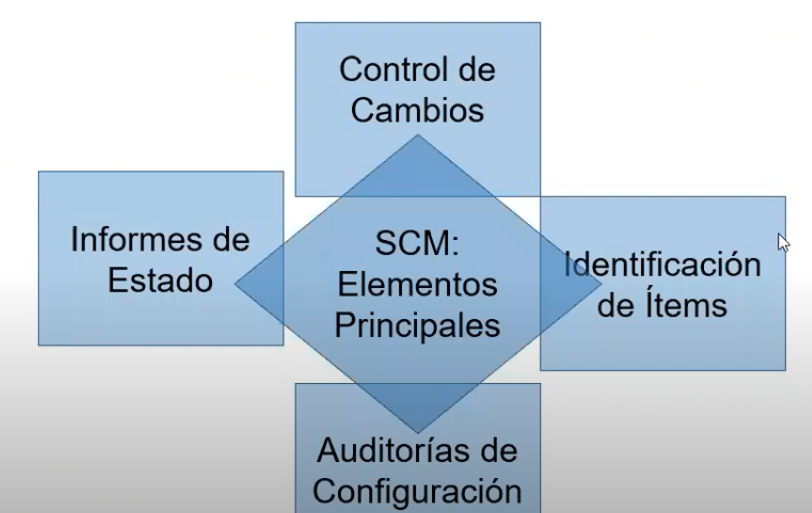
El esquema de funcionamiento que vimos antes funciona en los dos tipos.

**Actividades Fundamentales de la Administración de Configuración de Software**

“Una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y documentar las características funcionales y técnicas de los ítems de configuración, controlar los cambios de esas características, registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos”

Podemos identificar cuatro actividades que son las que conforman la gestión de configuración:

1. Identificar ítem de configuración
2. Controlar cambios
3. Tiene que ver con el informe y control del estado de los cambios (es una actividad que está prácticamente totalmente automatizada)
4. Tiene que ver con la auditoría (es una actividad que, en contexto ágil, es candidata a prescindirse por las características que tienen los proyectos que se trabajan en contextos ágiles)



1. **Identificación de ítems de configuración**

* Identificación unívoca de cada ítem de configuración
* Convenciones y reglas de nombrado
* Definición de la Estructura del Repositorio (esta estructura puede estar formada por paquetes, que adentro tengan productos. Es el lugar en donde voy a guardar el ítem)
* Ubicación dentro de la estructura del repositorio

Hay que identificarlo de manera única. Si yo defino, por ejemplo, que un ítem de configuración va a ser un caso de prueba, tengo que poder identificar ese caso de prueba de manera única y saber que ese caso de prueba es ese y no es otro.

¿Cómo logro esa identificación unívoca? Lo logro con las convenciones y las reglas de nombrado. Tengo que buscar una regla de nombrado genérica para mis items de configuración que cuando yo la quiera especializar, me permite nombrar de manera única cada uno de esos ítems.

Por ejemplo, nosotros queremos nombrar de manera única la especificación de requerimientos (es fácil porque tenemos una sola), la vamos a nombrar como primera parte del nombre el nombre del producto de software que estoy construyendo, y después poniéndole “especificación de requerimientos de software” como va a haber una sola en todo mi proyecto, es suficiente. Ahora bien, si defino que cada uno de mis casos de prueba es un ítem de configuración, identificarlo como “Caso de Prueba 1” es suficiente? No, no tengo forma de saber cual es el 1. Entonces, normalmente cuando tengamos que identificar un ítem de configuración deberíamos indicar que se trata de un caso de prueba, ponerle alguna etiqueta en donde vaya el número de caso de prueba, después a lo mejor puede ir el número del caso de uso y el escenario del caso de prueba. Pero como hasta acá voy a tener solo números, le quiero poner también un nombre, para que cuando lo lea lo pueda identificar. Entonces, le pone el número del caso de prueba, el nombre, voy a decir cual es el caso de uso al que se refiere y cual es el escenario de ese caso de prueba. Se pone el nombre del producto de software, digo que se trata de un caso de prueba y pone el nombre y el número. Es un nombre genérico, los ítems de los casos de pruebas todavía no existen.

Estamos diciendo que vamos a encontrar una manera o regla de nombrado para que cada vez que creamos un nuevo caso de prueba sepa qué nombre le voy a poner.

La definición de la estructura del repositorio es distinta a la regla de nombrado.

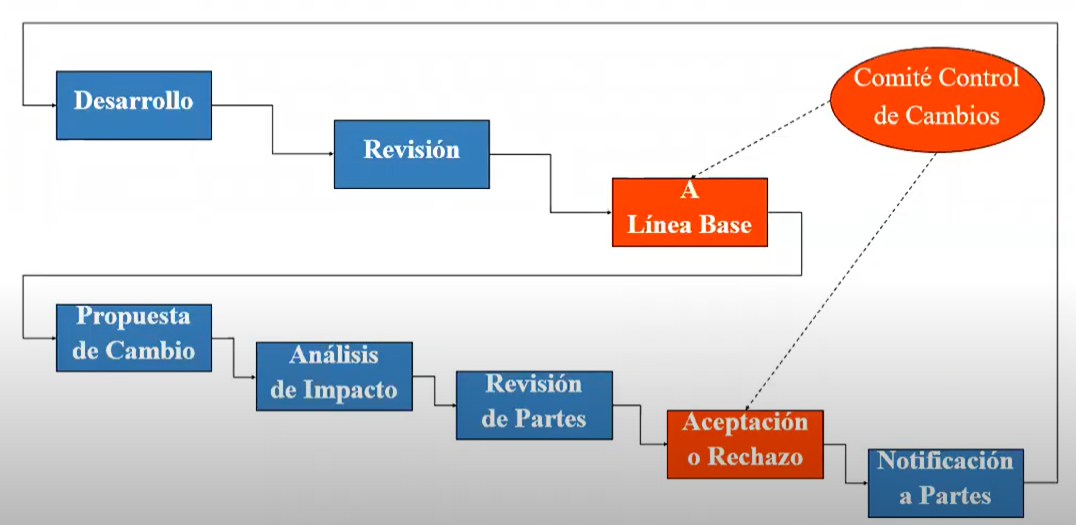
Con la regla de nombrado estamos definiendo cómo los vamos a nombrar a los ítems que vamos a crear de acá en adelante, porque todavía no los creamos.

Es una actividad que vamos a trabajar en el práctico.

1. **Control de Cambios**

¿Sobre qué necesitamos un procedimiento formal de control de cambios? En la línea base. Cuando queremos modificar una línea base, justamente porque era una configuración que habíamos aprobado y acordado, y cuando queremos modificarla tenemos que someterla a un procedimiento formal de control de cambios.

Esta actividad tiene que ver con eso, con poder definir cuál es ese procedimiento formal de control de cambios que nosotros vamos a ejecutar en nuestro proyecto cuando haya una petición de cambio para modificar algo de una línea base.



El comité de control de cambios está formado por representantes de todas las áreas involucradas en el desarrollo:

* Analisis, Diseno
* Implementación
* Testing
* Otros Interesados

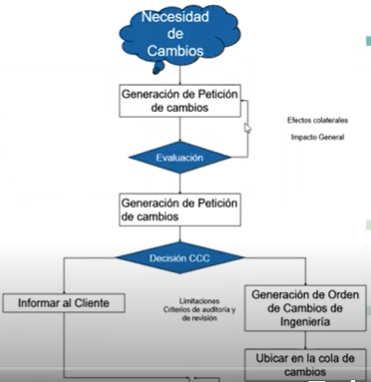
Normalmente se define un comité de control de cambios que dependiendo como esté conformado el proyecto puede estar conformado por distintas personas, normalmente está formado por las distintas áreas involucradas en el negocio (porque como nos vamos a poner de acuerdo en el cambio que vamos a hacer sobre la línea base y vamos a decir si el cambio que nos están pidiendo lo vamos a hacer o no lo vamos a hacer tiene que haber representantes de todas las áreas involucradas en el desarrollo). En ese comité de control de cambios es donde se recibe la petición de cambios o la propuesta de cambio sobre la línea base. Y sobre esa línea base, se trabaja la propuesta de cambio, se hace un análisis de impacto y una revisión de qué significa esa propuesta de cambio y por qué se está solicitando esa propuesta de cambio sobre la línea base. Y a partir de ahí, este comité de control de cambios decide si acepta el cambio sobre la línea base o lo rechaza. Si lo acepta, se notifica que se va a modificar la línea base y si se rechaza queda todo igual.

No es lo mismo cambiar una línea base que ya existe que crear una nueva.

¿Quién decide quién puede estar en un comité? Normalmente se define cuando el líder de proyecto arma el plan de gestión de configuración. No se crea un comité cada vez que hay que modificar una línea base sino que el comité ya existe desde que empieza el proyecto con esa función y se convoca cuando hay que hacer algún cambio sobre la línea base.

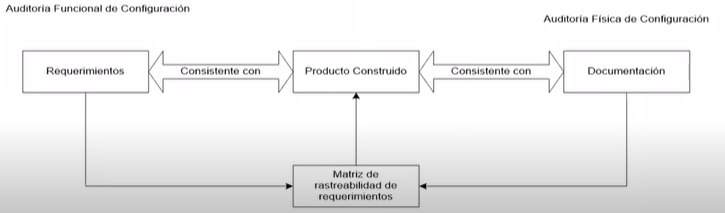
Tiene su origen en un Requerimiento de Cambio a uno o varios ítems de configuración que se encuentra en una línea base.

Es un procedimiento formal que involucra diferentes actores y una evaluación de impacto del cambio

El procedimiento es genérico, cuando llega esa petición de cambio, lo que se busca evaluar es cuál va a ser el impacto de ese cambio sobre la línea base, para que se pueda aprobar este cambio es necesario que pase por este comité.

1. **Auditorías de Configuración de Software**

Se encarga de verificar la correspondencia con los requerimientos (como se menciona en la definición de la IEEE). Aunque la auditoría va más allá de la verificación, porque hay dos tipos de auditorías.



La auditoría está centrada en dos aspectos

* Verificación: el problema es resuelto de manera apropiada que el usuario obtenga el producto correcto. Fisica
* Validación: asegura que un producto cumple con los objetivos preestablecidos, definidos en la documentación de líneas base(línea base).Todas las funciones son llevadas a cabo con éxito y los test cases tengan status “ok” o bien consten como “problemas reportados” en la nota de release. Funcional

**¿Qué significa construir el producto correcto?**

Es el que espera el cliente, entonces en la validación trabajo con esta comparación entre el producto construido y los requerimientos, esto se valida en la auditoría funcional de configuración. En cambio, en la verificación nos aseguramos de que estamos construyendo el producto correctamente, es decir, lo que definimos en la documentación lo estoy haciendo de esa manera. Para poder verificar esto utilizamos la auditoría física de configuración.

Para poder verificar esto utilizamos

* Auditoría física de configuración(PAC): Asegurar que lo que está indicado para cada ICS en la línea base o en la actualización se ha alcanzado correctamente.
* Auditoria funcional de configuración(FAC): Evaluación independiente de los productos de software, controlando que la funcionalidad y performance reales de cada ítem de configuración sean consistentes con las especificaciones de requerimientos

Luego de tener el producto construido podemos realizar alguna de estas dos auditorías. Una se basa enverificar que ese producto construido se pueda verificar con la documentación, es decir, si en el plan de control de configuración un ítem se va a llamar de una determinada manera, lo voy a guardar de una determinada manera porque la estructura del repositorio está armada de una determinada manera, la **auditoría física de configuración** se va a encargar de verificar que esto sea así, se cumpla lo que se documentó.

Por otro lado, **la auditoria funcional de configuración** valida que el producto construido sea consistente con los requerimientos, esto responde a dos aspectos básicos.

Estos conceptos se vuelven a ver en revisiones técnicas y con testing de software,pero en otro contexto.

1. **Informes o reportes 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 bases durante el ciclo de vida.

Está actividad es fuertamente automatizable, porque si yo utilizo herramientas para versionar y ver la evolución del sistema, lo que yo espero es que estas herramientas me permitan tener este informe o registro de estado, que me permite responder a las siguientes preguntas

* ¿Cual es el estado del ítem?
* Tuvimos un requerimiento de cambio sobre la línea de base ¿Este ha sido aprobado o rechazado por el CCB?
* ¿Qué versión de item implementa un requerimiento de cambio aprobado (saber cual es el componente que contiene la mejora)?
* ¿Cuál es la diferencia entre una versión y otra dada?

La idea es que podamos obtener las respuestas a estas preguntas sin hacer ningún trabajo manual, sino que se logre por la funcionalidad de las herramienta.

**Hora de un plan**

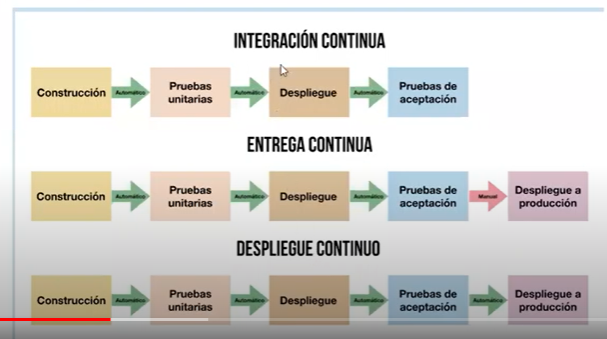
En el plan de gestión de configuración se debe incluir

* Reglas de nombrado de los CI
* Herramientas a utilizar para SCM
* Roles e integrantes de Comité
* Procedimiento formal de cambios
* Plantillas de formularios
* Proceso de auditoría

En este plan se definen los ítems de configuración, quienes forman parte del comité de control de cambio, quienes van a hacer los auditores y cuando se van a hacer.

En este caso estamos hablando de un proyecto con un proceso definido, si estamos hablando de metodologías ágiles difícilmente vamos a tener un plan de configuración, sin embargo si vamos a tener actividades de gestión de configuración, capaz que no con esta formalidad del plan.

**Evolución de la Gestión de Configuración de Software**



Este es el contexto dentro de la gestión de configuración en donde aparecen estas tres actividades (Integración continua, entrega continua, despliegue continuo) y ambas se usan cuando utilizamos metodologías ágiles, también en las tradicionales pero mucho más en ágiles.

Diferencia entre entrega continua y planes continuos - tema de trabajo conceptual, es lo que hay que investigar.

**Gestión de la Configuración de Software en ambientes ágiles**

Dentro del manifiesto ágil

Valores

* Interacción entre las personas por sobre procesos definidos y herramientas
* Software funcionando por sobre la documentación
* Personas interrelacionadas por sobre contratos
* Responder al cambio por sobre seguir al plan

La gestión de configuración es fundamental para mantener la integridad dentro del producto, no importa qué proceso use es, por eso en el manifiesto ágil hay que utilizarlo, pero hay que tener en cuenta cómo aplicar la gestión de configuración teniendo en cuenta los valores, por eso puede ocurrir que algunas actividades pesan más que otras, probablemente algunas las hacemos con menos intensidad.

SCM en Agile

* Sirve a los practicantes( equipo de desarrollo) y no viceversa.
* Hace seguimiento y coordinar el desarrollo en lugar de controlar a los desarrolladores
* Responde a los cambios en lugar de tratar de evitarlos
* Esforzarse por ser transparentes 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

No va a haber actividades de control, no hay control pero sí una gestión porque ágil busca responder a los cambios, entonces tiene que tener incorporado lo que tiene que ver con las actividades de la gestión de configuración. Vamos a tratar de automatizar lo que más se pueda, en scrum el equipo busca introducir esta configuración dentro del sprint, no son actividades que ocurran aparte y se coordinan como promueven las ágiles que es la auto organización y la autogestión, y privilegiando lo que está planteado en los principios.

**SCM en ágil, algunos tips**

* Es responsabilidad de todo el equipo

Sabemos que los equipos son autogestionados y que prácticamente todos hacen todo y las tareas están embebidas dentro de las otras, a diferencia de un tradicional que las tareas quedan más separadas que las otras.

* Automatizar lo más posible
* Educar al equipo
* Tareas de SCM embebidas en las demás tareas requeridas para alcanzar el objetivo del Sprint

**Referencias**

Bersodd, E.H, “elements of Software Configuration Managmenet”,

IEE Transactions on Software Enginieering, vol 10, nro. 1, enero 1984, pp 78-87

Little Book of Configuration Managment - <http://www.spmn.com>

SCM & the Agile Manifiesto - <http://www.scmpatterns.com/agilecm/> (no me lleva a ninguna pagina)