**Unidad 3 - Software Configuration Manager**

**Gestión de software como producto**

Escala de tiempo

Descripción generada automáticamenteEl proceso se adapta al proyecto. El proyecto incorpora personas y obtiene como resultado un producto. El proceso es automatizado con herramientas

El software es intangible, lo que implica que es difícil de estimar, pero fácil de modificar (no necesariamente es una ventaja). Es un conjunto de: programas, procedimientos, reglas, documentación y datos.

El software es más que simplemente código, es documentación, es herramientas, es procesos, también es lo que obtenemos como resultado final cuando terminamos un proyecto (un producto), un servicio. Las herramientas que usamos para desarrollar, almacenar y testear también son software.

El software es información que es:

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

**Problema del software**

El software evoluciona, se van modificando las partes del mismo a través del tiempo. Esto genera un problema, que es poder mantener la integridad del producto a lo largo de toda su vida

**Cambios del software**

Dichos cambios 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

Debemos aceptar y abrazar los cambios.

**SCM como disciplina de soporte**

Es una actividad paraguas, es transversal a todo el proyecto con aplicación en las diferentes disciplinas. Nos garantiza un nivel mínimo de integridad del producto

Administración de configuración de software:

* Control de calidad de proceso
* Control de calidad de producto
* Prueba de software

Los tres puntos forman parte del embudo de “Aseguramiento de calidad de software”. Dicho embudo es como una red de contención de la integridad y calidad del producto.

El aseguramiento de calidad puede incluir testing, pero el testing viene cuando está hecho el producto, mientras que el asegurar la calidad se hace mientras se construye el producto. El testing por sí solo no es aseguramiento de la calidad.

**Historia**

Esta disciplina surge cuando se declara la crisis del software en los años 50. La configuración de software se hacia de manera manual, como en los libros (no había tantas herramientas como hoy en día)

Esta disciplina nace debido al disparador de mantener el producto INTEGRO en todo su ciclo de vida. Esto mismo intenta cubrir la disciplina SCM. Los cambios en el software hacen que aparezca esta necesidad…

**Gestión de Configuración (CM)**

La gestión de configuraciones es una disciplina de soporte, forma parte de las discplinas protectoras y tiene el propósito de establecer y mantener la integridad de los productos de software a lo largo de su ciclo de vida. Involucra para la configuración:

* Identificarla en un momento dado.
* Controlar sistemáticamente sus cambios
* Mantener su integridad y origen

**Gestión de Configuración de Software (SCM)**

Una disciplina de SOPORTE que aplica dirección y monitoreo administrativo y técnico a:

* identificar y documentar las características funcionales y técnicas de los ítems de configuración (identificar ítems de configuración)
* controlar los cambios de esas características (controlar sus cambios sistemáticamente)
* registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos. (mantener su integridad y origen)

Es transversal a todo el proyecto, durante todo su ciclo de vida

**Integridad del producto**

El objetivo de un desarrollador de software debería ser la construcción de un producto que satisfaga las necesidades del cliente, y que le brinde valor. Dicho objetivo también se conoce como conseguir la integridad del producto. Se mantiene y tiene integridad el producto cuando:

* Satisface las necesidades del usuario: considera los requerimientos funcionales
* Es fácil y completamente rastreado durante su ciclo de vida.
* Satisface criterios de performance: referido a los requerimientos no funcionales
* Cumple con sus expectativas de costo

La integridad del producto es una medida de la satisfacción de las necesidades del cliente.

Rastreabilidad: facilita determinar el impacto de cambios

Trazabilidad: Vínculos entre componentes en las distintas etapas del ciclo de vida. Me permite que, si encuentro un error, poder dar un análisis de impacto más real.

Existe la trazabilidad hacia adelante y hacia atrás. Si modifico un requerimiento puedo saber como es el impacto en el resto. No se puede construir al final, sino mientras se construye el producto. Tiene un costo, pero es menor si lo hago cuando lo tengo que hacer.

**Problemas en el manejo de componentes:**

* Perdida de un componente (es muy normal en las empresas, cuando se está desarrollando un producto, que el equipo pueda perder un determinado componente y atrasar los tiempos por consecuencia).
* Perdida de cambios (el componente que tengo no es el último).
* Sincronía fuente – objeto – ejecutable.
* Regresión de fallas
* Doble mantenimiento
* Superposición de cambios
* Cambios no validados

**Conceptos claves para poder entender los 4 elementos de la disciplina**

Esos elementos son: ítems de configuración, repositorio, baseline y ramas (branch)

**1) Ítem de Configuración de Software (SCI):**

* **Ítem de configuración:** Se llama ítem de configuración a todos y cada uno de los artefactos, que forman parte del producto o del proyecto. Que puede sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución. (Deben estar versionados)

Este ítem puede guardarse en un repositorio, no importa el tipo de extensión, puede ser, códigos, documentos, diseño.

* **Versión:**
  + Se puede definir, desde el punto de vista de la evolución como la forma particular de un artefacto en un instante de tiempo 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. Le doy una identificación unívoca (nombre y número). También hay versiones de cada build (conjunto de ítems que conforman un incremento de producto)
  + Cuando componemos los ítems para entregar a un cliente (un paquete), eso también debe tener la versión, ejemplo “Windows”.
* **Variante:**
  + Una variante es una versión de un ítem de configuración (o de la configuración) que evoluciona por separado.
  + Por ejemplo, dependiendo de la plataforma (maquina + S.O) que la soporta, o de las funciones opcionales que haya de realizar o no.

**La configuración del software:**

* Un conjunto de ítems de configuración con su correspondiente versión en un momento determinado. (versión o estado de cada item)
* Cambia todo el tiempo la configuración, eso hace que requiera la necesidad de ser controlada.

**2) Repositorio**

* Es un contenedor de los ítems de configuración (IC). Mantiene la historia de cada IC con sus atributos y relaciones, y puede ser usado para hacer evaluaciones de impacto de los cambios propuestos.
* Pueden ser base de datos, o varias, o git por ejemplo.
* **Funcionamiento del repositorio:**
  + Está almacenado en un servidor.
  + Nosotros bajamos la parte (de la que tenemos acceso) del repositorio a la máquina local.
  + Modifico y hago commit para actualizar los cambios en el repositorio. No acciono directamente sobre el repositorio, sino que trabajo sobre la copia local
  + Checkout: Extracción de la copia a la máquina local.
  + Check In: Subir cambios al repositorio (devolución)
* Repositorios centralizados: Un servidor contiene todos los archivos con sus versiones, los administradores tienen mayor control sobre el repo, y en caso de tener una falla en el servidor, estamos en problemas, ya que todos necesitan acceder al mismo server.
* Repositorios descentralizados o distribuidos: Cada cliente tiene una copia exactamente igual del repositorio completo. Si un servidor falla solo es cuestión de copiar y pegar. En este tipo de repositorios tenemos muchas ventajas, pero tenemos problemas de sincronización al ser descentralizado. Posibilita otros workflows distintos al centralizado

**3) Línea base:**

* Es distinto a la versión del producto. Es un IC o más de uno (configuración) que han sido revisados y construidos y están en un estado de madurez tal que puedo usar de referencia para mi proyecto (y para desarrollos posteriores) o presentarlo al cliente. Sirve para mostrar que llegué a una etapa de madurez del producto. Las líneas base se las presenta en forma de etiquetas (hay etiquetas para la línea base y otras para sus updates).
* Hay ítems que no forman parte de una línea base. Es una especie de versión del producto destacada, estable y revisada. Permite ir atrás en el tiempo y reproducir el entorno de desarrollo en un momento dado del proyecto.
* Los ítems deben estar en listas y revisados para formar parte de la línea base. La línea base no se puede alterar sin un procedimiento pactado de control de cambios, y se debe llegar a un acuerdo.
* Cada línea base, y cada update de la misma representan la evolución del software durante su ciclo de vida.

Representación de líneas base: Pueden ser:

* *De especificación*: En ella se definen modelos (requerimientos, análisis, diseño). Son previas al código, de diseño, planes, etc.
* *Operacional*: Asociado al reléase de un producto que han pasado por un control de calidad definido previamente.

**4) Ramas:**

* Tenemos la rama principal (master). Generalmente armo ramas adicionales para cambiar algo, y al final de la iteración actualizar la main, hago un merge.
* Sirve para bifurcar el desarrollo.
* Pueden tener razones de creación con semántica. Ej: Una rama para Windows, otra para iOS, etc.
* Permiten la experimentación.
* Deberían tener un propósito.
* Pueden ser descartadas o integradas.
* **Integración de ramas:**
  + Es lo que se llama merge.
  + Lleva los cambios a la parte principal.
  + Puede haber conflictos (resolvemos con diff).
  + Todas las ramas deberían integrarse a la principal o ser descartadas.

**Actividades fundamentales de la administración de configuración de software.**

El gestor de configuración es un rol que se integra a otras personas del equipo, nunca es a tiempo completo, no hay personas que solo sean gestores de configuración

* Control de cambios
* Informes de estado
* Identificación de ítems
* Auditorias de configuración

1. **Identificación de ítems de configuración**

Implica una definición cuidadosa de sus componentes de la línea base. Cambios a esos componentes necesitan ser definidos, ya que dichos cambios, junto con la línea base, determinan la evolución del software.

* Identificación univoca de cada ítem de configuración
* Convenciones y reglas de nombrado.
* Definición de la estructura del repositorio.
* Ubicación paralela dentro del repositorio.

Un IC inicia en un estado borrador donde se van haciendo los cambios, luego pasa a uan etapa de revisión y finalmente queda aceptado. Una vez aceptado, queda en un estado de línea base, donde no es posible modificarlo si no es a través de un proceso formal de cambio.

Al declarar los ítems de configuración, tengo que tener en cuenta su ciclo de vida.

Diagrama

Descripción generada automáticamente

Los ítems con ciclo de vida distintos deberían tener distintas formas de nombrarlos.

**b) Control de cambios**

Establecer procesos de control de cambios sobre los IC que se encuentran en la línea base, porque el resto de IC que no pertenecen tienen libertad para cambiar.

Define procesos, mecanismos, para manejar la petición de cambios, de ítems. Es un procedimiento formal que involucra diferentes actores y una evaluación del impacto del cambio. Provee de un mecanismo para preparar, evaluar y aprobar/desaprobar las propuestas de cambios.

Control de cambios: Ante una necesidad de cambios se genera una propuesta de cambio (no debería haber cambios en la línea base en medio de un sprint). Luego se lleva a cabo un análisis de impacto del cambio, donde se evalúa el esfuerzo técnico, efectos secundarios, impacto global sobre otras funciones, se le asigna una prioridad para que posteriormente se realice la revisión de partes, en base a todo este análisis el comité acepta o rechaza el cambio y notifica a las partes involucradas.

El comité de cambios

* Está formado por representantes de todas las áreas involucradas en el desarrollo ( Análisis, Diseño, Implementación, Testing y otros interesados). Si el cambio se origina en el cliente, se lo incluye.

**c) Auditorias de gestión de configuración**

Las auditorias son aplicadas a cada línea base. Sirven para incrementar la visibilidad del software y establecer la trazabilidad a lo largo del ciclo de vida del producto. Pero dicha visibilidad y trazabilidad vienen asociadas con un costo monetario y de dinero. Con dicha visibilidad es posible evaluar la integridad del producto de software mientras se desarrolla, y resolver problemas encontrados mediante las auditorías. La auditoría provee trazabilidad entre el ciclo de vida del desarrollo del producto y los requerimientos de dicho producto. De esta manera quedan trazados los requerimientos entre cada línea base.

* Auditoria física(PCA): Asegura que lo que está indicado para cada ICS en la línea base o en la actualización se ha alcanzado realmente, y que el SW y la documentación son internamente consistentes. Es decir, vela por la integridad del repositorio. Sin línea base no puedo auditar, la línea base nos indica que auditar, no se audita todo el repositorio (hace foco en verificar).
* Auditoria funcional(FCA): Evaluación independiente de los productos de software, controlando que la funcionalidad y performance reales de cada ítem de configuración sean consistentes con la especificación de requerimientos. Hace foco en la validación, que los requerimientos tengan trazabilidad
* El auditor no hace censo, hace muestreo. Sirve a dos procesos básicos: la validación y la verificación:
* Validación: El problema es resuelto de manera apropiada que el usuario obtenga el producto correcto.
* 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 status “ok” o bien consten como “problemas reportados” en la nota de reléase.

**d) Informes de estado**

Es un mecanismo para mantener registro de cómo el sistema ha evolucionado y donde se encuentra el sistema en un instante determinado de tiempo. Mantiene registros de las otras tres actividades asociadas al SCM

* 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.
* Genera reportes, inventarios.
* Incluye reportes de rastreabilidad de todos los cambios realizados a las líneas base durante el ciclo de vida.

**Plan de Gestión de Configuración**

Incluye momentos en los que voy a auditar. No hay fecha y hora, sino cuando esté lista alguna funcionalidad, por ejemplo.

* Reglas de nombrado de los CI
* Herramientas para utilizar para SCM
* Roles e integrantes del comité
* Procedimiento formal de cambios
* Plantilla de formularios
* Procesos de auditoria

**Buenas Prácticas**

* Hacer la CM el trabajo de todos.
* Crear un ambiente y un proceso de ingeniería que permita la CM
* Definir y documentar el proceso de SCM, luego de seleccionar la/las herramientas que le den soporte al proceso.
* El personal de SCM debe contar con individuos con expertise técnica para dar soporte al desarrollo y mantenimiento del producto.
* Los procedimientos y el plan de SCM debe realizarse en las etapas iniciales del proyecto.

**SCM en Agile**

**SCM y el manifiesto Agile**

SCM es crucial para el éxito de los proyectos ágiles. Un proceso ligero y bien diseñado de SCM puede acelerar el proyecto, y permite sincronizar al equipo.

Hay ciertos valores del manifiesto agile que se relacionan con el SCM

**Individuos e interacciones sobre procesos y herramientas**: Las herramientas y procesos de SCM deben apoyar la forma en la que trabajas, no al revés. Muchas veces los motivos de fracaso del SCM es tratar de imponer procesos rígidos en un equipo, o imponer el uso de herramientas que imponen un proceso que es diferente al que la organización necesita.

**Software funcionando sobre documentación comprensiva:** SCM puede automatizar políticas de desarrollo y procesos, con conocimiento ejecutable antes que confiar en el conocimiento de la documentación. Se pueden hacer las cosas sencillas para los desarrolladores, y asegurar que los procesos necesarios sean hechos mediante scripts en lugar de procedimientos documentados.

**Colaboración con el cliente sobre negociación contractual**: SCM puede facilitar la comunicación e interacción entre stakeholders y ayudar a manejar expectativas. Las herramientas y procesos adecuados pueden proveer visibilidad del estado de un proyecto a los clientes.

**Responder al cambio sobre seguir un plan**: SCM me permite facilitar el cambio, no evitarlo. Se pueden usar las políticas y estructuras de SCM para permitir que el equipo de desarrolladores progrese a una velocidad adecuada.

**Características de SCM en agile**

* Sirve a los practicantes (equipo de desarrollo) y no viceversa.
* Hace seguimiento y coordina el desarrollo en lugar de controlar a los desarrolladores.
* Se esfuerza por ser transparente y sin fricción, automatizando tanto como sea posible.
* Eliminar el desperdicio, no agregar nada más que lo que agregue valor.
* Documentación lean y trazabilidad
* Feedback continuo y visible sobre calidad, estabilidad e integridad

**Tips**

* 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.