



Resumen primer parcial

[Enlace al notion](#)

▼ TEMAS

Unidad 1 completa (incluye paper No silver bullet)

- Introducción a la Ingeniería del Software. ¿Qué es?
- Estado Actual y Antecedentes. La Crisis del Software.
- Disciplinas que conforman la Ingeniería de Software.
- Ejemplos de grandes proyectos de software fallidos y exitosos.
- Ciclos de vida (Modelos de Proceso) y su influencia en la Administración de Proyectos de Software.
- Procesos de Desarrollo Empíricos vs. Definidos.
- Ciclos de vida (Modelos de Proceso) y Procesos de Desarrollo de Software
- Ventajas y desventajas de c/u de los ciclos de vida.
- Criterios para elección de ciclos de vida en función de las necesidades del proyecto y las características del producto.
- Componentes de un Proyecto de Sistemas de Información.

- Vinculo proceso-proyecto-producto en la gestión de un proyecto de desarrollo de software.
- Paper No silver bullet

Unidad 2

- Gestión de Producto
- Requerimientos Ágiles
- User Stories
- Estimaciones Ágiles
- SCRUM

Unidad 3

- SCM

▼ Unidad 1



Fuentes de información:

-

Ingeniería de software de Ian Sommerville

- PPTS

- Resúmenes de años anteriores

▼ Qué es el **software**?

*"set de **programas** junto con la **documentación** que lo acompaña → necesaria para desarrollar y mantener los programas ejecutables que se entregan al cliente"*

▼ Es sólo código?

No, es un concepto más **integral** → a Judith le gusta definirlo como "conocimiento o información" que está a diferentes niveles de abstracción (menos o más detalle)

▼ Qué contempla esta definición?

- Código
- Archivos de configuración
- Documentación para el usuario
- Documentación del sistema
- Herramientas usadas para la construcción
- Datos
- Programas

▼ Por qué el SW es distinto a la **manufactura**?

- El SW es menos predecible
- El SW es intangible
- No hay producción de SW en masa, casi ningún producto de software es igual a otro, incluso cuando la idea resulte similar
- No todas las fallas son errores
- El software no se gasta, puede dejar de tener vigencia porque los requerimientos del negocio van cambiando y se tiene que adaptar pero no se desgasta en términos de materiales
- El software no está gobernado por las leyes de la física
- No se construye software en una línea de producción.

▼ El SW como información

El SW es 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

Y como lo definimos como información cada artefacto del PUD es SW.

▼ Por qué **cambia** el software?

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

▼ Qué **problemas** puede haber en el desarrollo de software?

- La versión final del producto no satisface las necesidades del cliente
- No es fácil extenderlo y/o adaptarlo. Agregar más funcional en otra versión es casi una misión imposible.
- Mala documentación
- Mala calidad
- más tiempos y costos que los presupuestados.

▼ Qué es la **Ingeniería de software**?

Es la disciplina de la ingeniería que se preocupa de todos los aspectos de la producción de un software; **desde** las primeras etapas de la especificación **hasta** el mantenimiento del sistema una vez operando - (*Ian Somerville*)

▼ Cómo lo definió Parmas [1987] ?

Parmas [1987] definió a la ingeniera en software como "*Multi-person construction of multi-version software*". En esta definición existen dos conceptos clave:

▼ **Función del ingeniero**

Los ingenieros aplican teorías, métodos y herramientas de la manera más conveniente siempre tratando de descubrir soluciones a los problemas teniendo en cuenta que deben trabajar con restricciones financieras y organizacionales por lo que buscan soluciones contemplando estas restricciones

▼ **Disciplinas** que conforman la ingeniería de software



- Técnicas: ayudan a construir el **producto** (recordá lo de las 4P)
 - Ej: Análisis, diseño, implementación, prueba, despliegue, etc.
- De gestión: planificación, monitoreo, control del **proyecto**.
- De soporte: gestión de configuración, métricas, aseguramiento de la calidad
 - Son transversales, es decir que están presentes todo el tiempo

▼ De dónde nace la ingeniería de SW?

El concepto “ingeniería de software” se propuso originalmente en 1968, en una conferencia realizada para discutir lo que entonces se llamaba la **“crisis del software”** (Naur y Randell, 1969). Se volvió claro que los enfoques individuales al desarrollo de programas no escalaban hacia los grandes y complejos sistemas de software. Éstos no eran confiables, costaban más de lo esperado y se distribuían con demora - (*Ian Somerville*)

▼ Cuáles son los costos de la ingeniería de SW (en %)?

El 60% de los costos son de desarrollo y el 40% de testing - (libro de *Ian Somerville*)

▼ Cuáles son los principales retos que enfrenta la ingeniería de software?

Se enfrentan con una diversidad creciente, demandas por tiempos de distribución limitados y desarrollo de software confiable. - (libro de *Ian Somerville*)

▼ Qué es la **crisis** del software?

Hace referencia a un conjunto de hechos relativos al software, planteados en la conferencia de OTAN en 1968 por Friedrich Bauer, quien recalcó la dificultad para generar software libre de defectos, fácilmente comprensibles y que sean verificables.

▼ Cuáles son las causas?

- Evolución del hardware que permite crear sistemas más complejos no acompañada con una evolución de los procesos de desarrollo de software.
- Demanda creciente de sistemas complejos, difíciles de estimar, con muchas solicitudes de cambios.
- Falta de una disciplina que intervenga en los aspectos de la producción del software

▼ Cuáles son los motivos de los productos de SW **exitosos**

- Involucramiento del usuario 15.9 %
- Apoyo de la Gerencia 13.0 %
- Enunciado claro de los requerimientos 9.6 %
- Planeamiento adecuado 8.2 %
- Expectativas realistas 7.7 %
- Hitos intermedios 7.7 %
- RRHH competentes 7.2 %

▼ Cuáles son los motivos de los productos de SW **fallidos**

- Requerimientos incompletos 13.1 %
- Falta de involucramiento del usuario 12.4 %
- Falta de recursos 10.6 %

- Expectativas poco realistas 9.3 %
- Falta de apoyo de la Gerencia 8.7 %
- Requerimientos cambiantes 8.1 %

- ▼ Cómo nos encontramos hoy en día?
 - ▼ Ejemplos de grandes proyectos de software **exitosos**
 - ▼ Ejemplos de grandes proyectos de software **fallidos**
-

- ▼ Qué es un **proceso** de software?

El Proceso de Software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software.

Estas actividades varían dependiendo de la organización y el tipo de sistema que debe desarrollarse y el proceso debe ser explícitamente modelado si va a ser administrado.

Según la IEEE, un proceso es una secuencia de pasos ejecutados para un propósito dado. Mientras que un proceso de software es un conjunto de actividades, métodos, prácticas, y transformaciones que la gente usa para desarrollar o mantener software y sus productos asociados.

Dentro de un proceso vamos a encontrar la parte de procedimientos y métodos, herramientas y equipos y personas con habilidades, entrenamiento y motivación. Estos tres elementos son importantes y conforman esta visión de proceso de software.

- ▼ Cuál es el objetivo de un proceso de SW?

Obtener un producto de software o un servicio asociado

- ▼ Cuáles son las entradas de un proceso de SW?

Son dinámicas. Incluyen requerimientos pero también personas, materiales, energía, equipamiento y procedimientos

- ▼ Importancia de las personas

Como ingenieros en sistemas de información, trabajamos en una industria que es humana-intensiva, significa que el software lo hacen personas y que el aporte más importante para que el producto llegue a las manos de los usuarios interesados, es el aporte que hacen las personas, y de hecho en términos de costos, lo

más caro de hacer software es pagarles a las personas que participan del proyecto.

▼ Unidad 2

▼ Unidad 3

▼ SCM

- ▼ Qué significan las siglas SCM?

Software configuration management

- ▼ Qué es gestionar?

Mantener controlado, administrar, integrar.

- ▼ Qué es el SCM? → **definición**

SCM es una disciplina protectora 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 - (*ANSI/IEEE 828, 1990*)

- ▼ Desglosamos esta definición:

- ▼ Qué es?

Una disciplina protectora

- ▼ Qué aplica?

Dirección y monitoreo (administrativo y técnico)

- ▼ A qué aplica esto?

- A identificar y documentar características (funcionales y técnicas) de **ítems de configuración**
 - Controlar cambios de las características de los **ítems de configuración**
 - Registrar y reportar cambios y estados de implementación
 - Verificar correspondencia con requerimientos

- ▼ Por qué se dan los cambios en el SW?

- 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

▼ Qué tipo de disciplina es SCM?

Es una disciplina de soporte

▼ Qué implica que sea de soporte?

Es una actividad “paraguas”  → Implica que sea transversal a todo el proyecto y relevante al producto a lo largo de todo su ciclo de vida - *Charles*

▼ Cuál es el objetivo del SCM?

El propósito es mantener la **integridad** del producto de SW a lo largo de todo el ciclo de vida

▼ A qué se refiere con INTEGRIDAD?

- Satisfacer las necesidades del cliente 
- Poseer cambios fácil y completamente rastreables durante su ciclo de vida 
- Buena performance 
- Cumplir con expectativas de costo \$\$

▼ Qué problemas surgen en el manejo de componentes?

- Pérdida de un componente
- Pérdida 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

▼ Qué es un ÍTEM DE CONFIGURACIÓN?

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**

▼ Desglosamos esta definición

▼ Qué son?

Artefactos

▼ De qué forman parte?

Del producto o del proyecto

▼ Características

- Pueden sufrir cambios
- Necesitan ser compartidos
- Necesitamos saber su estado y evolución

▼ Ejemplos

Algunos ejemplos de Ítems de Configuración

- | | |
|-----------------------------|-------------------------------|
| ❖ Plan de CM | ❖ Planes de Iteración |
| ❖ Propuestas de Cambio | ❖ Estándares de codificación |
| ❖ Visión | ❖ Casos de prueba |
| ❖ Riesgos | ❖ <u>Código fuente</u> |
| ❖ Plan de desarrollo | ❖ Gráficos, iconos, ... |
| ❖ Prototipo de Interfaz | ❖ Instructivo de ensamble |
| ❖ Guía de Estilo de IHM | ❖ Programa de instalación |
| ❖ Manual de Usuario | ❖ Documento de despliegue |
| ❖ Requerimientos | ❖ Lista de Control de entrega |
| ❖ Plan de Calidad | ❖ Formulario de aceptación |
| ❖ Arquitectura del Software | ❖ Registro del proyecto |
| ❖ Plan de Integración | |

Es todo lo que está en el repositorio

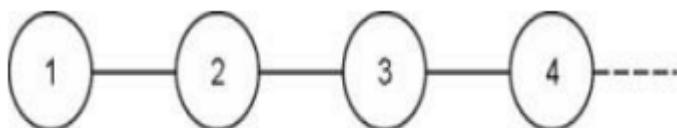
▼ Qué es la VERSIÓN de un IC?

Es la forma particular de un artefacto en un instante o contexto dado

▼ A qué se refiere con *control de versiones*?

El control de versiones se refiere a la evolución de un único ítem de configuración (IC), o de cada IC por separado.

Puede representarse gráficamente en forma de grafo.

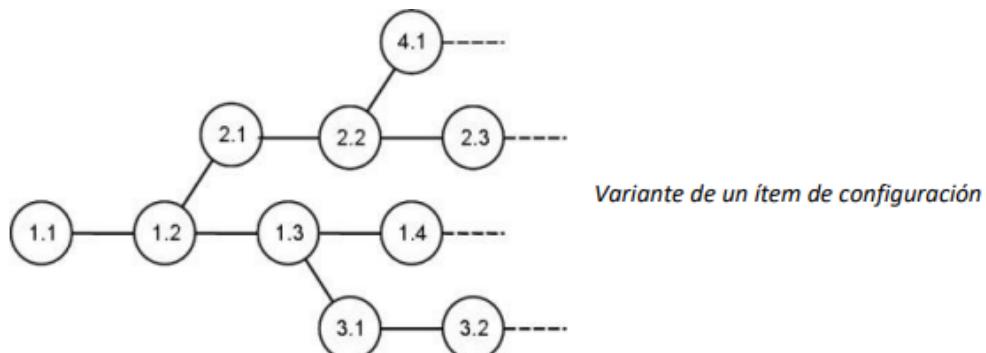


Evolución lineal de un ítem de configuración

▼ Qué es la VARIANTE de un IC?

Una variante es una versión de un IC que evoluciona por separado (paralelo)

Representan configuraciones alternativas



Variante de un ítem de configuración

▼ Qué es la CONFIGURACIÓN DE SW?

Un conjunto de **IC** con su correspondiente **versión** en un momento determinado

▼ Qué es un REPOSITORIO?

Contenedor de ítems de configuración

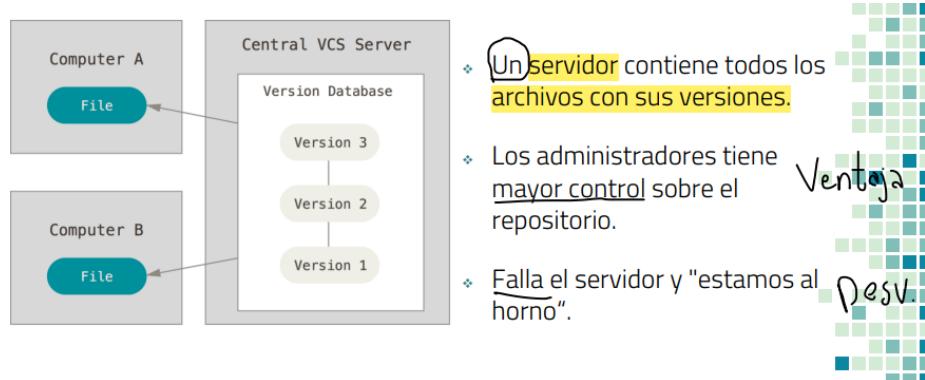
▼ Para qué sirve?

- Mantiene la historia de cada IC con sus atributos y relaciones.

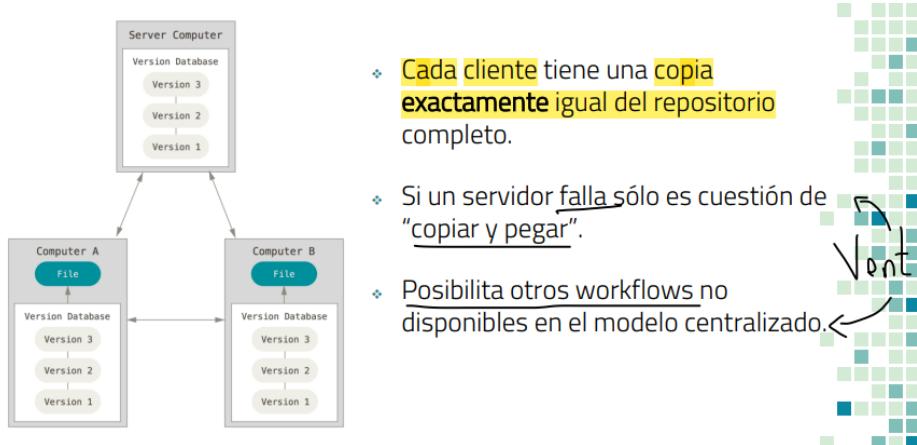
- Usado para hacer evaluaciones de impacto de los cambios propuestos.

▼ Cuáles son los 2 tipos de repositorios?

▼ Centralizados (características, ventajas y desventajas)



▼ Descentralizados (características, ventajas y desventajas)



▼ Cómo funciona?

Básicamente, tenemos el repositorio fuente, a partir del cual se puede realizar 2 acciones:

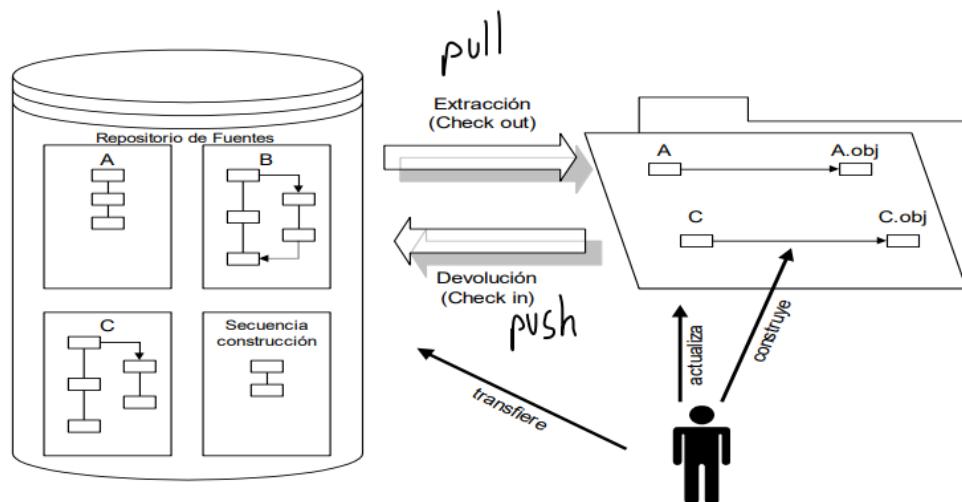
- **Extracción (Check out)**: Es el proceso mediante el cual un desarrollador toma una copia del código fuente desde el repositorio hacia su entorno local de trabajo.

Esto sería análogo al **git pull**

- **Devolución (Check in)**: Una vez que el desarrollador ha realizado cambios en el código, devuelve esos cambios al repositorio.

Esto sería análogo al **git push**

Entonces habría una actividad intermedia entre que uno construye los datos en su entorno de trabajo y los envía nuevamente al repositorio, lo cual sería básicamente hacer un **git commit**



▼ Qué es una LÍNEA BASE?

Es una configuración que ha sido revisada formalmente y sobre la que se ha llegado a un acuerdo → versión ESTABLE del repositorio

Se "marca" con una etiqueta

▼ Es lo mismo Línea Base que versión del producto?

NO

▼ Para qué sirve?

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

▼ Qué permite?

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

▼ Qué tipos de línea base hay?

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

▼ Para qué sirve una RAMA 

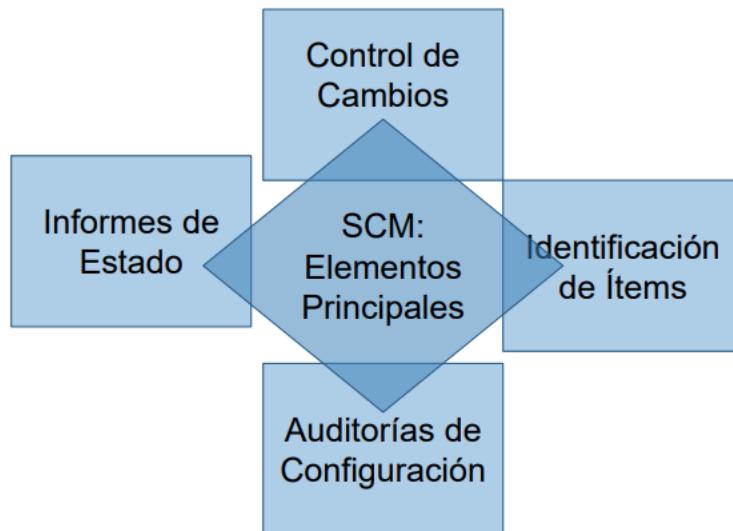
- Sirven para bifurcar el desarrollo (trabajar en paralelo)
- Permiten la experimentación

▼ En qué consiste *integrar* una rama?

En llevar los cambios a la rama principal (hacer un merge)

Todas las ramas deberían eventualmente integrarse a la principal o ser descartadas → no pueden quedar inconclusas

▼ Cuáles son las **4 actividades** de la gestión de configuración de software?



▼ En qué consiste la **identificación** de IC?

- Identificación única de los ítems de configuración
- Definición de convenciones y reglas de nombrado

- Definición de la estructura del repositorio
- Ubicación dentro de la estructura
- ▼ De qué tipo pueden ser los IC?

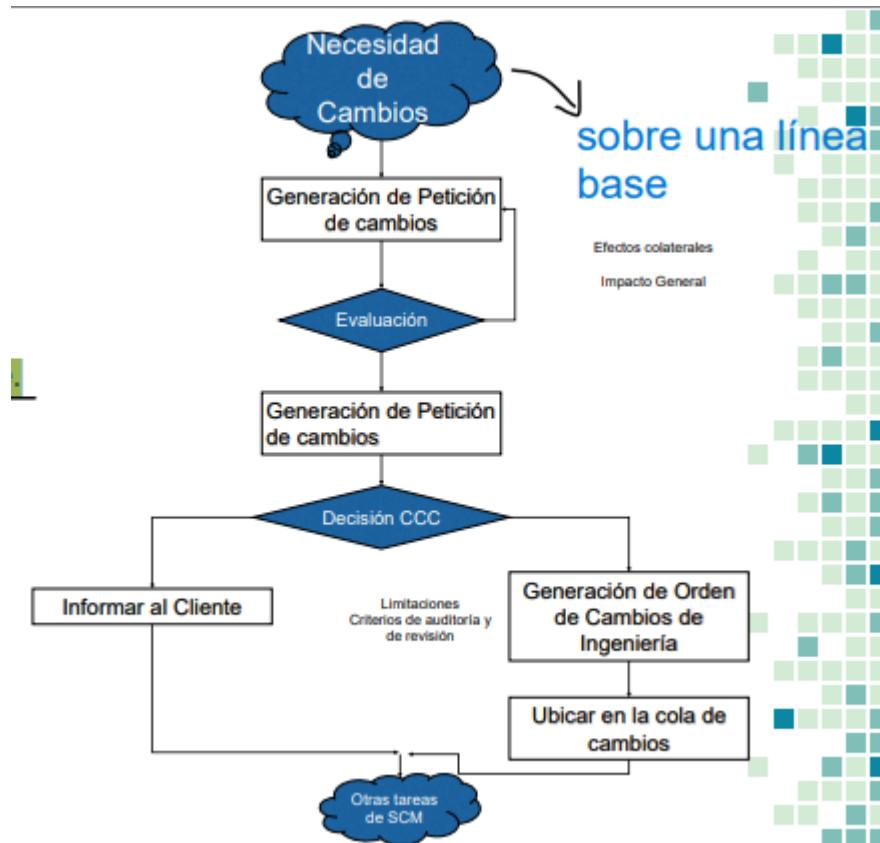


▼ En qué consiste el **control de cambios**?

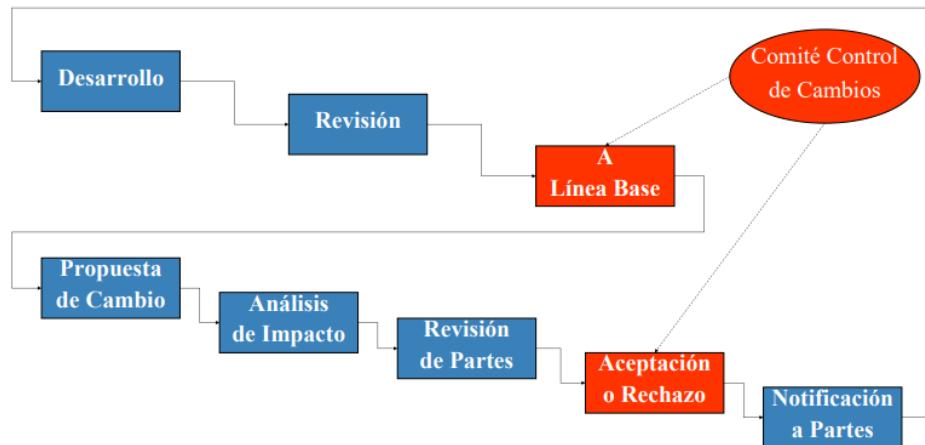
Es un procedimiento formal que involucra diferentes personas y una evaluación del impacto del cambio

▼ Cuál es el origen del control de cambios?

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



▼ Gráfico Proceso de control de cambios



▼ Qué conforma el comité de control de cambios?

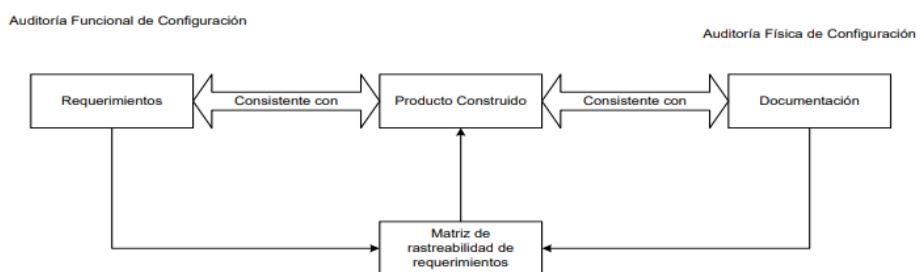
Formado por representantes de todas las áreas involucradas en el desarrollo:

- Análisis, Diseño
- Implementación
- Testing
- Otros interesados

▼ En qué consisten las **auditorías** de configuración?

Consiste en realizar un control para corroborar la **consistencia** del producto deseado con los requerimientos (auditoría funcional) y la documentación (auditoría física) → fijate en el gráfico

▼ Gráfico del proceso de una auditoría



▼ Cuáles son los tipos de auditorías?

- **Auditoría física de configuración (PCA)** → Asegura que lo que está indicado para cada ICS en la línea base o en la actualización se ha alcanzado realmente.
- **Auditoría funcional de configuración (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.

▼ Para qué procesos sirve la auditoría?

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 (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

▼ En qué consisten los **informes de estado**?

- Consiste en mantener los registros de la evolución del sistema (el paso por los diferentes estados)
- Manejar información y salidas → como maneja mucha, se suele implementar automatización 
- Realizar reportes de rastreabilidad de los cambios realizados a las líneas base durante el ciclo de vida

▼ Qué deberíamos incluir en un plan de gestión de la configuración?

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

▼ Cómo funciona el SCM en ambientes ágiles?

- ❖ Sirve a los practicantes (equipo de desarrollo) y no viceversa.
- ❖ 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

▼ Prácticas continuas (opcional)

Integración continua: se refiere a la práctica de combinar frecuentemente y de manera automatizada los cambios de código realizados por diferentes miembros del equipo en un repositorio compartido. Esto permite detectar y solucionar errores de integración tempranamente, lo que ayuda a garantizar la calidad del código

Entrega continua: es una extensión de la integración continua, que implica la automatización del proceso de construcción, prueba y empaquetado del software en cada cambio de código, y la entrega de estos paquetes a un entorno de pruebas o de producción para su evaluación. La entrega continua permite a los equipos de desarrollo detectar y corregir errores en etapas tempranas del ciclo de vida del software.

Despliegue continuo: es una extensión de la entrega continua, que implica la automatización de todo el proceso de despliegue del software en un entorno de producción. Esto permite que los cambios de código sean entregados a los usuarios finales con mayor rapidez y frecuencia, lo que reduce el tiempo de lanzamiento de nuevas funcionalidades y mejora la experiencia de los usuarios

▼ No Silver Bullet

▼ Cuál es la analogía que plantea el autor?

Que un proyecto de software es como un hombre lobo, puede pasar de lo familiar o de lo sencillo hacia un monstruo de forma repentina.

Para un hombre lobo buscamos balas de plata, pero para un proyecto de software no es tan sencillo

▼ Qué representa el monstruo del proyecto de software?

Los plazos incumplidos, objetivos fallados y productos defectuosos

▼ Qué pasa en el software a diferencia del hardware?

Ningún invento de los que mejoraron la productividad, fiabilidad y simplicidad en el hardware, como la electrónica, los transistores etc, harán lo mismo por el software.

No podemos esperar a que los costos del sw caigan tan rápidamente como los del hw

▼ Según el autor la anomalía es que el sw progrese lento?

NO, que el hw progrese rápido → de forma exponencial

▼ Qué 2 cosas debemos hacer según el autor?

1. Darnos cuenta de que la anomalía está en que el HW progresó demasiado rápido
2. Examinar las dificultades de la tecnología del SW

▼ Cuál es la esencia de una entidad software?

Es una construcción de conceptos entrelazados:

- conjuntos de datos,
- relaciones entre los datos,
- algoritmos y
- llamadas a funciones

Esta esencia nos indica que uno de estos conceptos abstractos construidos tiene muchas representaciones. Sin embargo es muy preciso y muy detallado.

▼ Según el autor cuál es la parte más dura de construir SW?

La especificación, el diseño y prueba del concepto construido → no el trabajo de representarlo y comprobar la fidelidad de la representación

"Todavía tendremos errores de sintaxis, evidentemente; pero eso es trivial si lo comparamos con los errores conceptuales en la mayoría de los sistemas"

▼ Cuáles son las propiedades inherentes de la esencia de los sistemas modernos de SW?

▼ **Complejidad**

▼ Por qué decimos que son complejas las entidades de SW?

- Por su tamaño
- Porque no hay 2 partes iguales. No hay elementos repetidos, a diferencia de otros sistemas.
- Tiene un gran número de estados → lo que hace que su concepción, descripción, y pruebas sean muy duras
- Ante un incremento → el número de interacciones entre los elementos cambia de una forma no lineal con el número de elementos y la complejidad se incrementa a un ritmo mucho más que lineal.
- Dificultad de comunicación entre los miembros del equipo

▼ La complejidad es una propiedad esencial o accidental?

La complejidad del software es una propiedad esencial, no accidental

▼ Qué pasa si intentamos eliminar la complejidad?

Podemos eliminar su esencia. No hay que buscar eliminar la complejidad porque es parte de la esencia del SW

▼ Qué cosas derivan de la complejidad?

No sólo se derivan problemas técnicos, sino también problemas de gestión

Hace difícil tener una visión de conjunto, impidiendo la integridad conceptual.

▼ **Conformidad**

▼ Cuál es la diferencia entre la gente del SW y los físicos o matemáticos que en teoría también trabajan con complejidades

Las labores del físico descansan en un profunda fe de que hay principios unificadores que deben encontrarse.

Nada de esa fe conforta al ingeniero de software. La mayoría de la complejida que el debe controlar es arbitraria, forzada sin ritmo ni razón por las muchas instituciones humanas y sistemas a los que debe ajustarse

▼ De dónde viene la complejidad?

La mayoría de la complejidad viene del tener que ajustarse a otros interfaces; esta complejidad no puede simplificarse solamente con un rediseño del software

▼ Variabilidad

▼ A qué está sometido constantemente el SW?

A cambios o presiones para que cambie

▼ Es difícil cambiar SW?

No, puede cambiar fácilmente → es puro pensamiento, infinitamente maleable

▼ Por qué cambia el SW?

El software vive dentro de un entorno cultural de aplicaciones, usuarios, leyes y hardware. Todo este entorno cambia continuamente, y esos cambios inexorablemente fuerzan a que se produzcan cambios en el software.

- Cuando un SW triunfa → la gente intenta usarlos para nuevas aplicaciones en el límite o más allá del dominio original para el que se diseño
- El software exitoso sobrevive más allá de la vida del hardware para el que fue diseñado → nuevos computadores, nuevos discos, pantallas, impresoras... y el software debe adaptarse para estas nuevas oportunidades

▼ Invisibilidad

▼ Por qué decimos que el SW es invisible e in-visualizable?

Es difícil representarlo espacialmente

En el momento que intentamos crear diagramas de las estructuras de software, descubrimos que no está constituido por uno, sino por varios gráficos en distintas direcciones, superpuestos unos sobre otros.

▼ Y en qué influye que el SW sea invisible?

Nuestra mente no puede visualizar de forma completa el SW y por lo tanto se dificulta pensar en el SW y en su diseño

▼ Cuáles son los tres pasos el desarrollo de la tecnología del software?

▼ **Lenguajes de Alto Nivel**

▼ Qué beneficios hubo en la aparición de lenguajes de alto nivel?

Productividad, fiabilidad, simplicidad y comprensibilidad

Libera a un programa de mucha de su dificultad accidental

Reduce el trabajo intelectual del usuario ya que elimina parte de la abstracción del programa

▼ **Tiempo compartido**

▼ Qué beneficios hubo en la aparición del tiempo compartido?

Permite la inmediatez, y esto nos permite tener una visión de conjunto de la complejidad.

▼ **Entornos de desarrollo unificados**

▼ **Guía SCREAM (scrum)**

