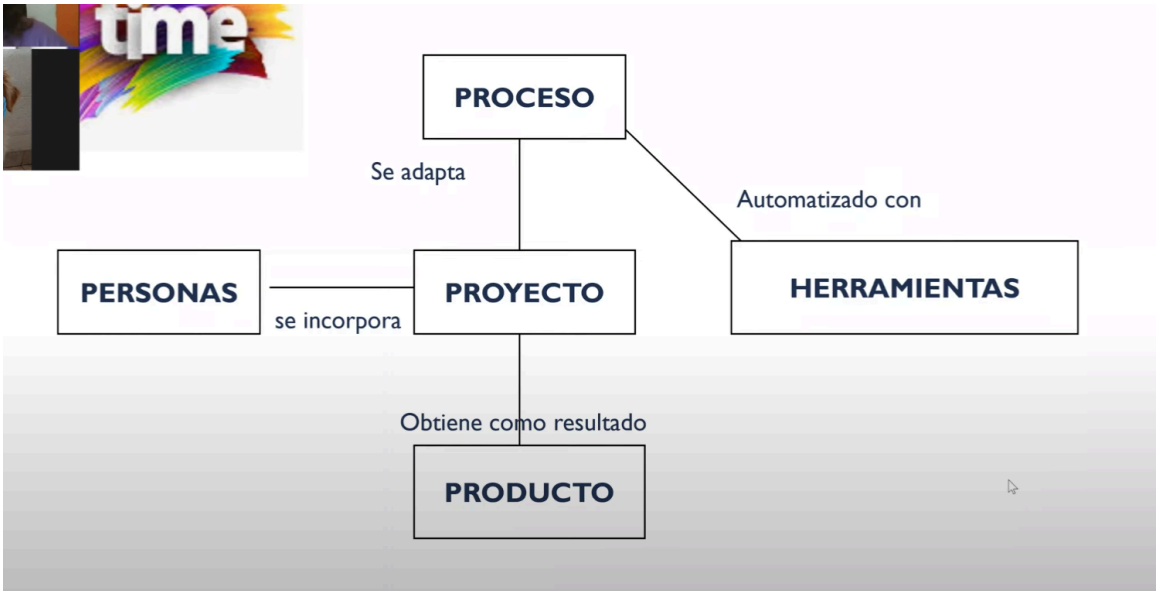
Primera clase

Unidad	Unidad 1
tipo	teórico
Archivos & enlaces	01 SCM.pdf

Componentes de Proyectos de software

Ejes sobre los que la ingeniera y calidad de software trabaja



Entregar un producto de calidad a nuestros clientes esa seria la motivación fundamental.

Se ocupa del producto como objetivo final.

El concepto de proyecto es el medio por el cual yo obtengo un producto de software.

El proyecto para poder existir necesita de una serie de elementos.

Necesita de un método de una forma.

El proceso da una definición teórica de que es lo que debería hacer para hacer software. El proyecto debe tomar de esa definición de proceso lo que le parece que hace falta, lo que es mejor con un criterio de necesidad y de adecuación.

Toma ese conjunto de tareas de lo que es necesario hacer y define el alcance de proyecto que son tareas o actividades.

Es importante que nosotros tengamos esta distinción en nuestra cabeza → entre proyecto, producto, proceso.

Hacemos foco en el concepto proyecto.

Tiene que ver como construyo yo el producto

Análisis y diseño hablan del proceso pero no del producto.

Hay dos tipos

En los procesos empíricos no hay ninguno que sea un proceso completo. Si bien el proceso unificado intenta cumplir todo lo necesario para desarrollar software. A un scrum no se les dice proceso , porque es una parte del proceso. La idea es que como son empíricos el equipo defina todo lo necesario para el proceso

Proceso

Definición de un conjunto de actividades que toman como entrada requerimientos y obtienen como salida un producto o servicios.

El proceso unificado define todo lo que hay que hacer para obtener el producto final. Es la teoría

El proyecto

Es una unidad de gestion, es un medio por el cual yo administro lo que necesito y las personas que van a estar integradas y trabajando para obtener como resultado un producto o servicio (software en nuestro caso). El proyecto empieza decidiendo y eligiendo la gente que va a sumarse a trabajar con distintos roles en el proyecto. Incorpora personas que van a cumplir roles → esos roles van a estar definidos en el proceso.

> 💡 El proceso se instancia en el proyecto . El proyecto es el que la vida al proceso. El proyecto arma el conjunto de tareas reales que se van a llevar a cabo. El proceso es teoría.

Producto

El producto es software.

> ¿Que es software? → Es conocimiento empaquetado a distintos niveles de abstracción

Cada tarea del proyecto tiene que tener un entregable.

Todo lo que sale como resultado de la ejecución de tareas de un proyecto es software.

Cuando en visual studio ponemos nuevo proyecto en realidad estamos creando un nuevo producto.

El ciclo del vida del producto es mas grande ya que va a vivir mucho mas que el proyecto que lo crea.

▼ SCM (Gestion de configuración del software)

Se encarga de rastrear y controlar los cambios realizados a un sistema de software a lo largo de todo su ciclo de vida.
Esta práctica asegura que el software se desarrolle, mantenga y despliegue de forma **consistente y controlada**, mejorando su calidad, trazabilidad y estabilidad

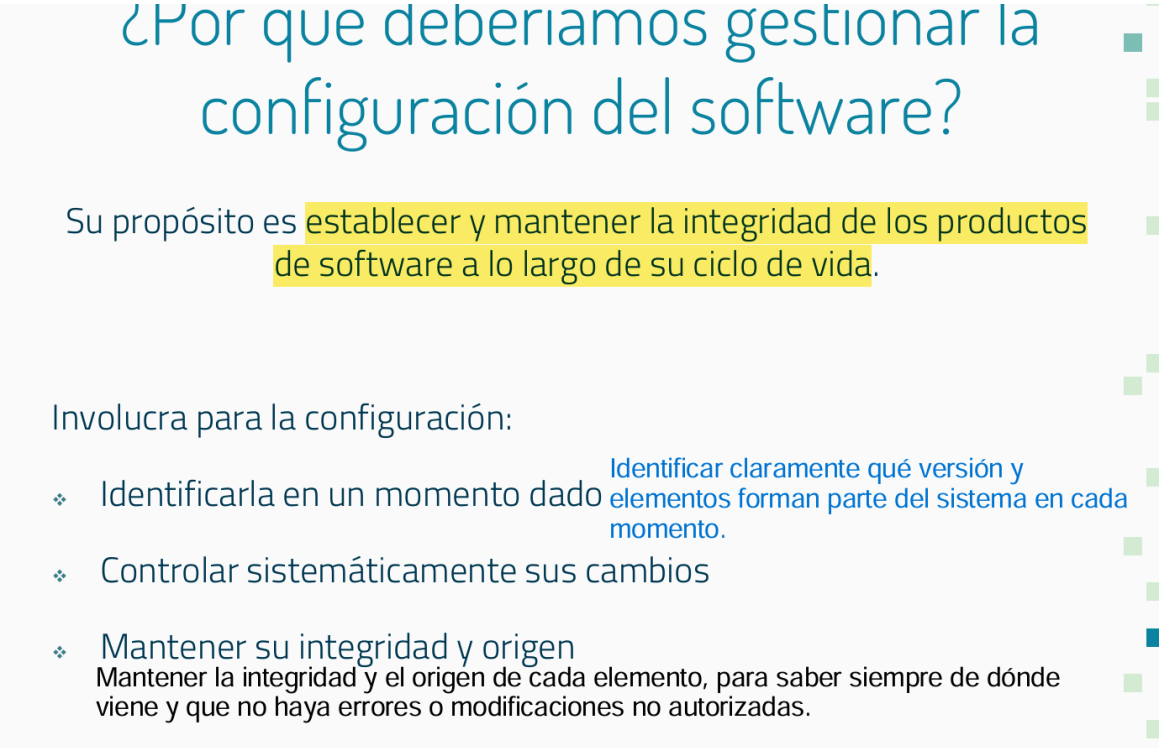
La **gestión de configuración de software** es básicamente una forma **ordenada de guardar y controlar todo lo que forma parte de un proyecto de software**, para que los cambios no generen caos.

Esta definición dice, en palabras más simples, que **SCM** es una forma organizada de:

1. **Saber qué cosas forman parte del proyecto** (código, documentos, imágenes, etc.) y dejarlo todo bien identificado y anotado.
2. **Controlar los cambios** que se hagan sobre esas cosas, para que nada se modifique sin seguimiento.
3. **Registrar y avisar** de qué cambios se hicieron, en qué estado están y quién los hizo.

4. **Comprobar** que lo que tenemos coincide con lo que se pidió en los requerimientos.

Es como ser el “administrador y guardián” de todos los elementos de un proyecto de software para que siempre estén ordenados, actualizados y correctos.



Para que un software sea “íntegro” debe:

1. **Cumplir lo que el usuario necesita**.
2. **Poder rastrear todo su historial**: saber qué versión está en uso, qué cambios se hicieron y cuándo.
3. **Rendir bien** (cumplir criterios de performance).
4. **Mantenerse dentro del presupuesto**.

Conceptos Clave para la gestion de configuración de software

▼ Item de configuración

En palabras simples, un **ítem de configuración** es **cualquier cosa importante del proyecto** (un archivo, documento, código, imagen, manual, etc.) que:

- **Forma parte del producto o del proyecto**.
- **Puede cambiar** con el tiempo.
- **Necesitamos controlar y saber en qué versión está**.
- **Debe ser fácil de identificar y compartir** con el equipo.

💡 **Ejemplo**:

En un proyecto de software, un ítem de configuración puede ser:

- El **código fuente** del programa.
- El **manual de usuario**.
- Los **casos de prueba**.
- Un **prototipo de interfaz**.
- El **plan de desarrollo**.

Básicamente, es **todo lo que es lo suficientemente importante como para llevarle un control de cambios** y asegurarnos de que siempre trabajamos con la versión correcta.

▼ Versión

- **Versión**: es una "foto" de un ítem de configuración (por ejemplo, tu código) en un momento específico.

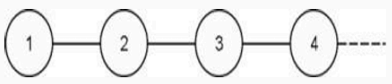
  Ejemplo: *versión del lunes*, *versión del miércoles*, *versión final*.

- **Control de versiones**: es el seguimiento de cómo ese ítem va cambiando a lo largo del tiempo.

  Cada cambio importante genera una nueva versión.

Vamos a tener items de configuración (por ejemplo los archivos de nuestro proyecto) y vamos a tener versiones de esos items de configuración , que van a ser "fotos" de esos items en el tiempo.

La evolución puede representarse gráficamente en forma de grafo.



Evolución lineal de un ítem de configuración

▼ Variante

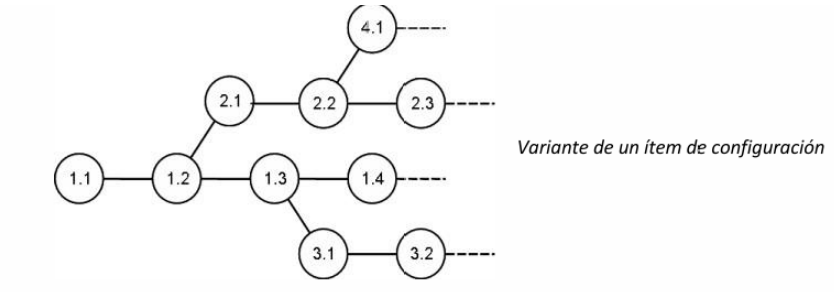
Es como un camino paralelo que se realiza de una versión de un item de configuración que se mantiene o desarrolla de forma independiente.

No es simplemente la siguiente versión, sino una **rama diferente** que tiene sus propias características.

📌 Ejemplos:

- Tu bot para la clínica podría tener:
  - **Variante A:** versión para Windows.
  - **Variante B:** versión para Linux.
  - **Variante C:** versión con módulo de turnos online.

Cada una evoluciona por separado porque tienen diferencias técnicas o funcionales, aunque todas provengan de un mismo origen.



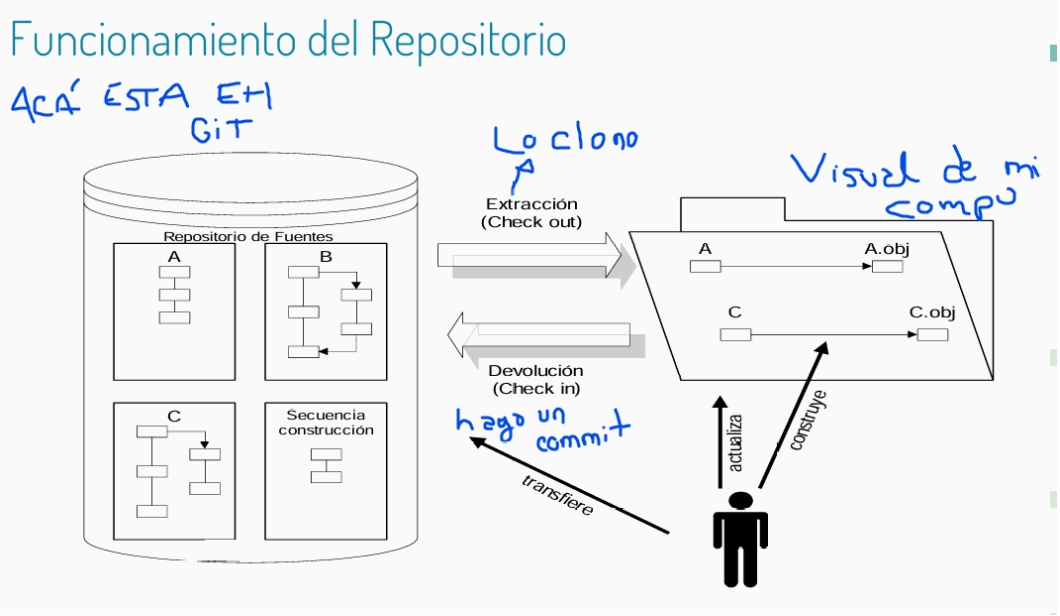
Variante de un ítem de configuración

▼ Configuración del software

La version de todos los items de config de un proyecto en un momento determinado

▼ Repositorio

Un **repositorio** es como una “gran caja organizada” donde guardamos todos los ítems de configuración de un proyecto y sus versiones.



📌 Repositorios Centralizados

- **Cómo funcionan:**

  Todo el código y sus versiones están en **un único servidor central**.

  Cada persona que trabaja en el proyecto se conecta a ese servidor para obtener o subir cambios.

- **Ventajas:**
  - Los administradores tienen **más control**.
  - Sencillo de configurar.
- **Desventajas:**
  - Si el servidor central se cae o se pierde, **nadie puede trabajar** ni acceder al historial.

- Dependencia total de la conexión al servidor.

💡 Ejemplo: **Subversion (SVN)**, **Perforce**.

📌 Repositorios Descentralizados

- **Cómo funcionan:**

  Cada persona tiene en su computadora **una copia completa** del repositorio con **todo el historial**.

  Esto significa que no dependes de un único servidor para trabajar o tener el historial.

- **Ventajas:**
  - Si el servidor principal se cae, cualquier copia local puede restaurar todo.
  - Permite más formas de trabajo (por ejemplo, trabajar sin conexión y luego sincronizar).
- **Desventajas:**
  - Cada copia ocupa más espacio, porque guarda todo el historial.

💡 Ejemplo: **Git**, **Mercurial**.

En tu caso, como usás **Git**, estás trabajando con un **repositorio descentralizado**:

- Vos y tu compañero tienen **el mismo historial completo** en sus máquinas.
- Si GitHub se cayera, todavía podrían seguir trabajando y luego sincronizar cambios cuando vuelva a estar disponible.

▼ Linea Base (Baseline)

- Imagina que en un proyecto guardas una “foto” de cómo está todo en un momento exacto.
- Esa foto es **la línea base**: una versión aprobada de los documentos, el diseño o el código.
- Se etiqueta para identificarla y **no se puede cambiar sin un proceso formal** de control de cambios.
  PARA QUE SIRVE?

- Es la referencia oficial para seguir trabajando.
- Permite volver atrás y reproducir cómo estaba el proyecto en ese momento.
- Asegura que todos trabajen sobre la misma versión aprobada.

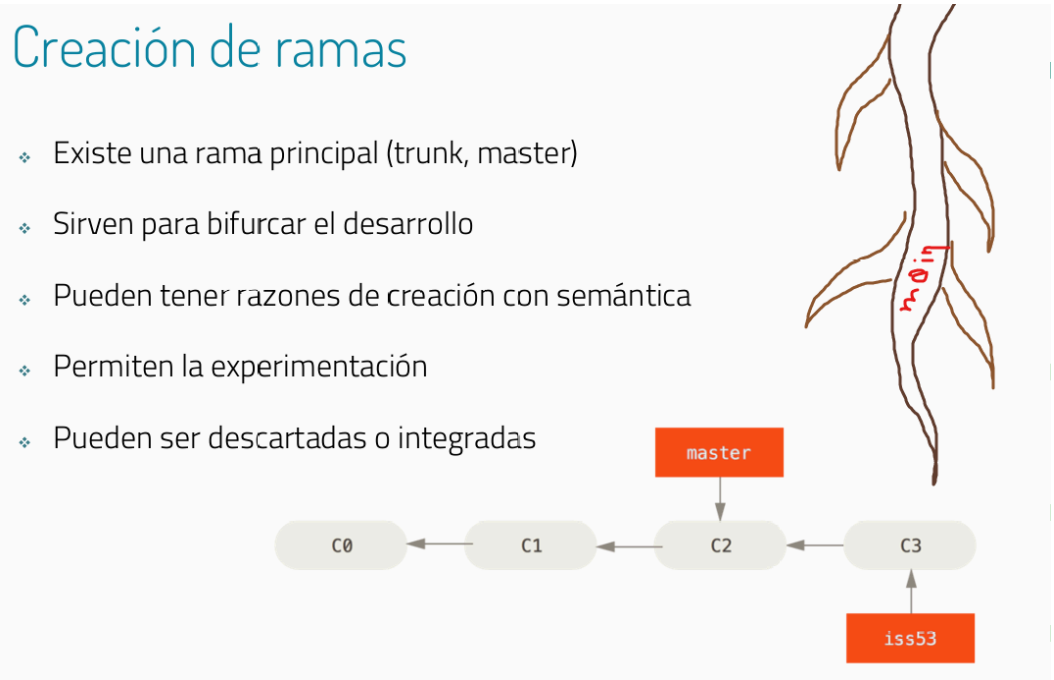
Ejemplos

- Línea base de **requisitos**: lista aprobada de lo que el sistema debe hacer.
- Línea base de **diseño**: diagramas y planos aprobados.
- Línea base de **código**: versión estable del software.

▼ Ramas

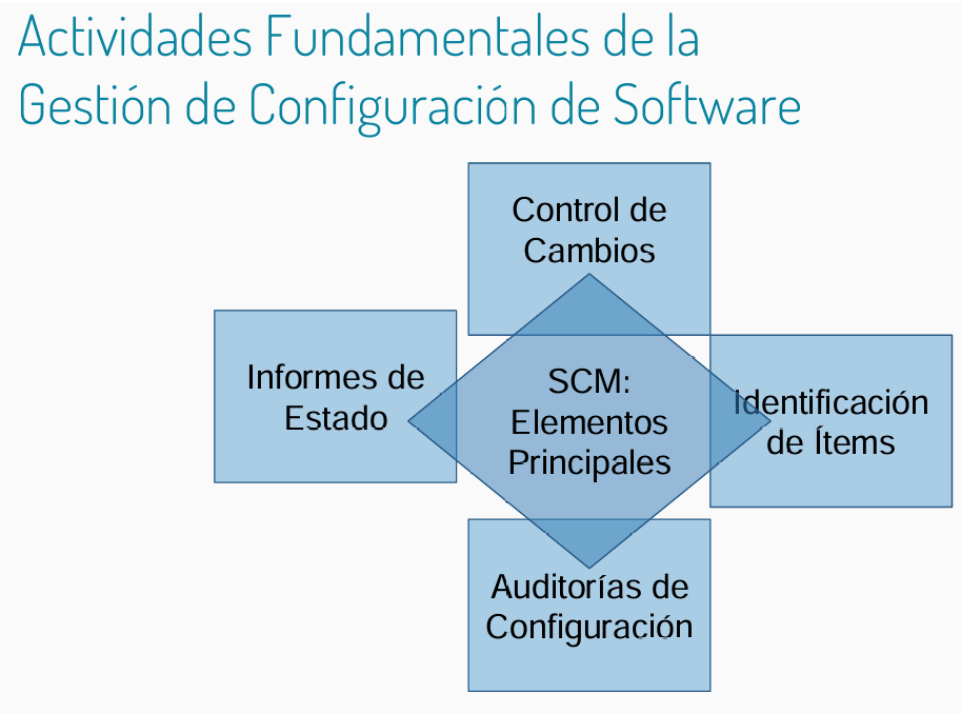
En tu proyecto de la clínica:

- `main` → es la rama principal donde está la versión estable del código.
- Si querés hacer cambios grandes (por ejemplo, reescribir todo el `services`), podrías crear una **nueva rama** llamada, por ejemplo, `refactor-services`.
- Ahí trabajarías tranquilo, sin afectar `main`.
- Cuando termines, harías un **merge** para que esos cambios pasen a `main`.



Gestion de la configuración de software

La **Gestión de Configuración de Software (SCM)** es como tener un sistema de control y organización para todo lo que compone tu proyecto. Sirve para saber **qué versiones de qué cosas tienes, quién las cambió, cuándo y por qué**.



▼ Identificación de items de configuración

En tu ejemplo del proyecto de la clínica, **crear carpetas para separar y estructurar el código por funcionalidad** es justamente una parte de la **identificación de ítems de configuración**.

Por ejemplo:

- Una carpeta `services` para la lógica de negocio.
- Otra carpeta `routes` para manejar las rutas del backend.
- Otra `models` para las entidades y conexión con la base de datos.