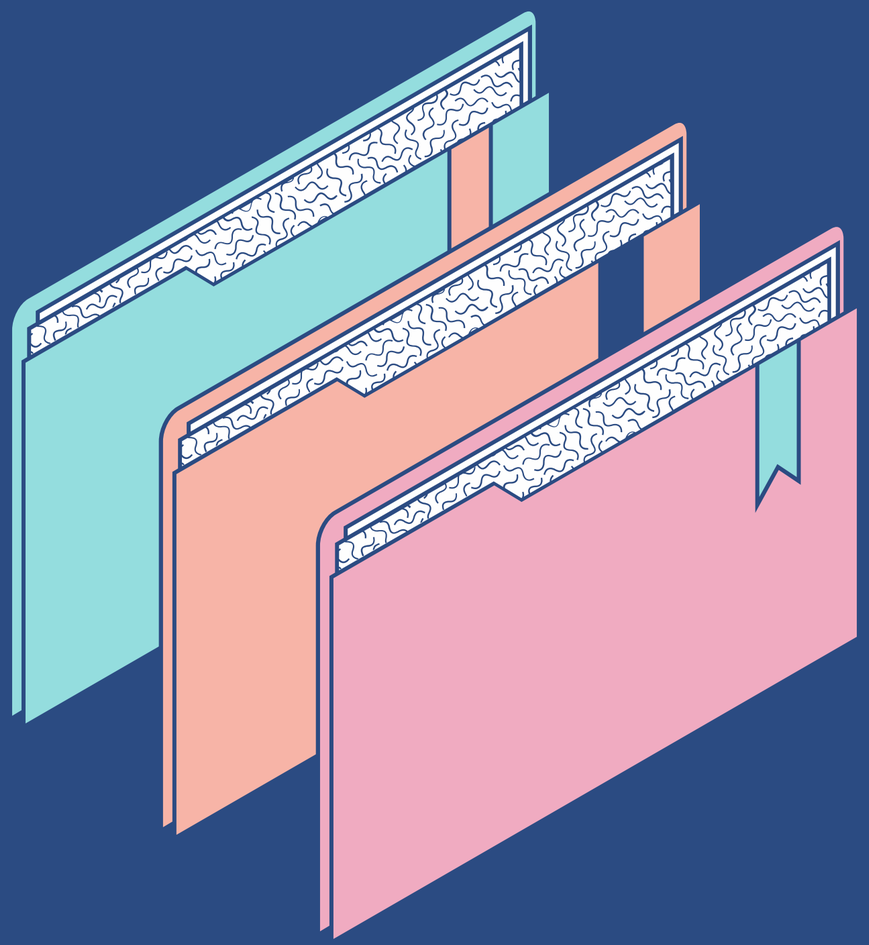
PROCESO DE APOYO

SCM (Software Configuration Management)

¿Cómo gestionamos los cambios?

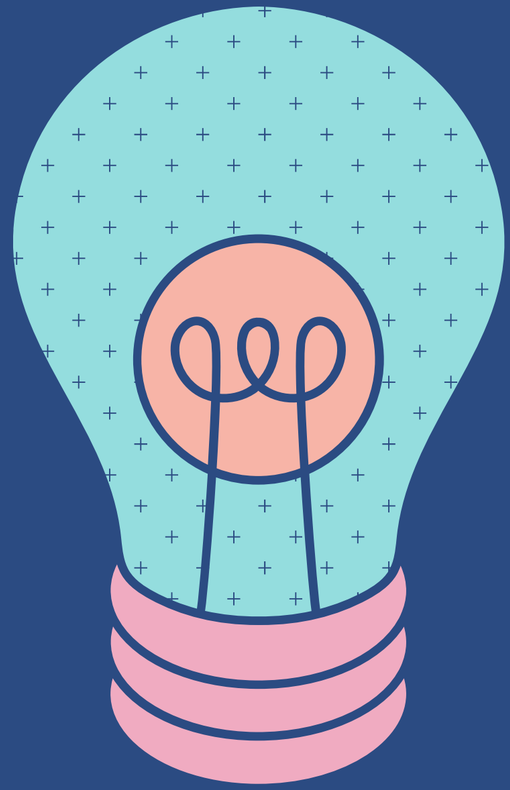


Agenda

TEMAS CLAVE EN ESTA PRESENTACIÓN

- Definiciones
- Problemas del cambio
- ¿Por qué es importante manejar el cambio?
- Elementos de la configuración del Software (ECS)
- Actividades de SCM
- Repositorio

"Es el arte de identificar, organizar y controlar modificaciones al software que está siendo construido por un equipo de programadores con el objetivo de maximizar la productividad minimizando los errores."

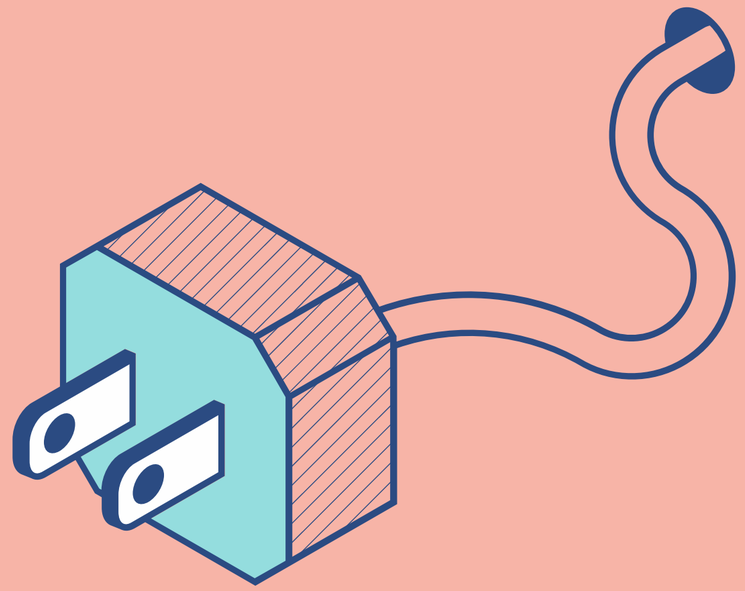


WAYNE A. BABICH

Software Configuration Management

Es el proceso de aplicar procedimientos técnicos y administrativos a lo largo del ciclo de vida, para **identificar**, definir las **piezas de software**; controlar modificaciones y **versiones** de estas piezas; registrar y reportar el **estado** de cada pieza y las solicitudes de modificaciones; asegurar la **completitud, consistencia y correctitud** de las piezas de software; y controlar el almacenamiento, manipulación y entrega de los productos de software

[ISO/IEC 12207]



Problemas del Cambio

1 — 2 — 3

ACTUALIZACIONES SIMULTEANEAS

Dos o más personas trabajaron separadamente sobre el mismo objeto

El último elimina el trabajo del resto

DEPENDENCIA DE PRODUCTOS

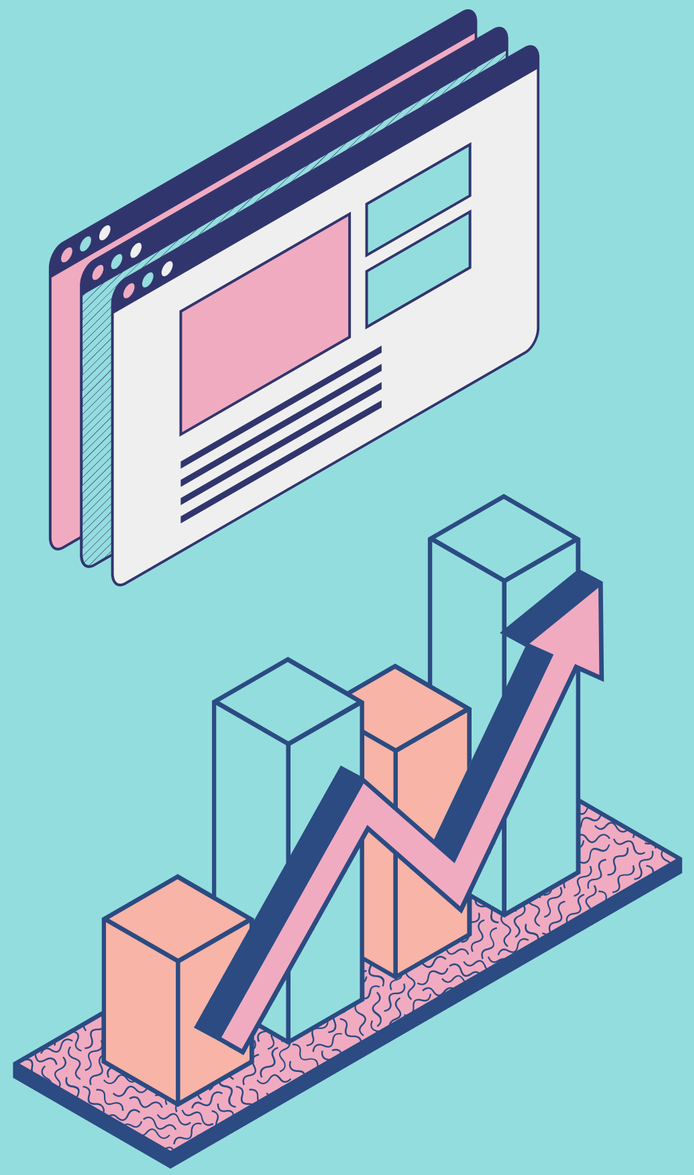
Los productos dependen de otros recursos para su desarrollo, no se comunica la evolución de los recursos de los cuales dependen

No se utiliza la versión adecuada del recurso para la elaboración del producto

VERSIONES

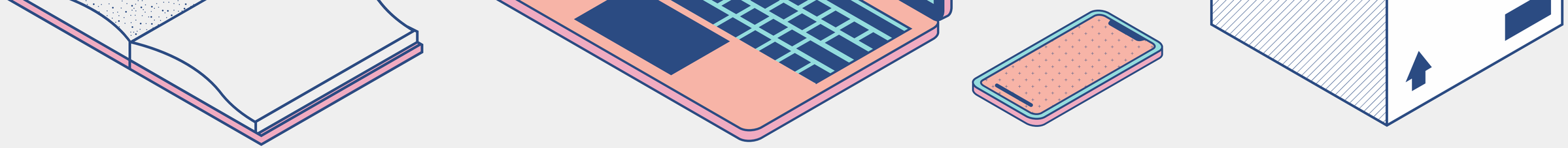
El software evoluciona en versiones y contiene variantes. No se lleva una trazabilidad de su evolución

Se produce re-trabajo y pérdida de información



¿Por qué es importante manejar el cambio?

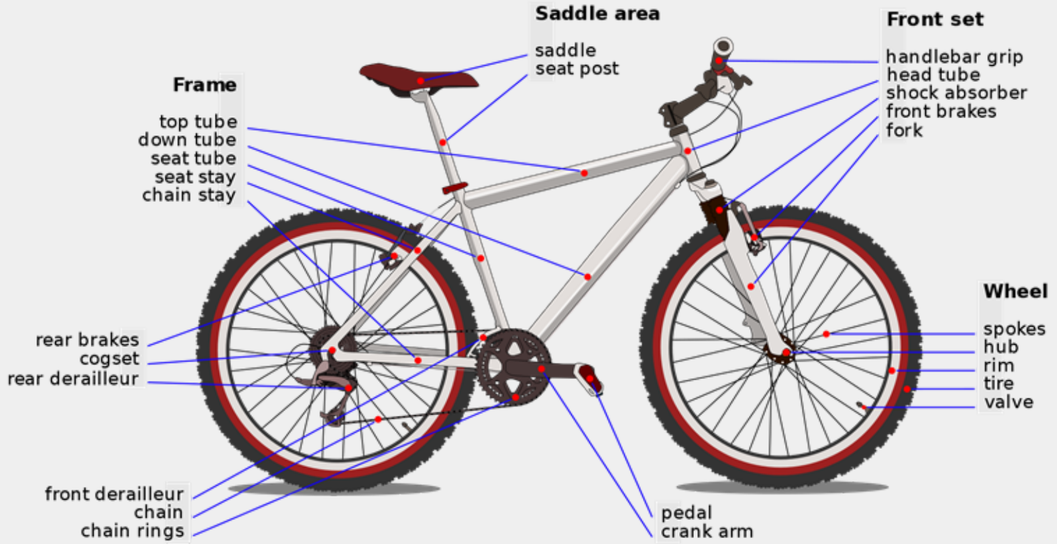
- Es **inherente** al software (es parte de su esencia)
- Los **requerimientos** probablemente cambien.
  - Es muy difícil visualizar algo abstracto
- El **negocio** probablemente cambie.
  - El software automatiza elementos del negocio
- "Tirar" un sistema es algo que **no queremos hacer**.
  - No existe y probablemente no exista nunca una máquina que haga software en masa, va a ser siempre algo humano
- Siempre sabemos más en el futuro que en el presente



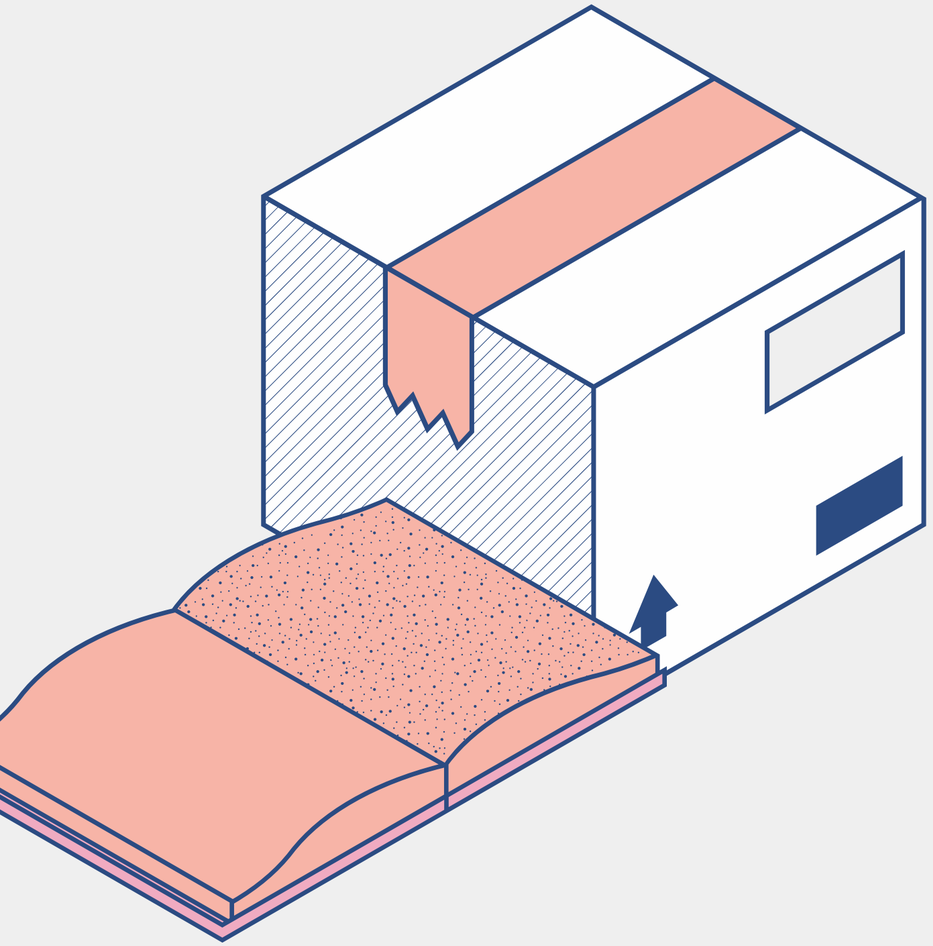
Elementos de la Configuración de Software (ECS)

Elementos que integran un proyecto y que es necesario identificar, organizar y resguardar.

- Código Fuente
- Ejecutables y librerías
- Documentos
- Bases de datos
- Recursos



Ejemplos de ECS Procesos de Ingeniería



Ingeniería de Requerimientos:

- Especificación de Requerimientos
- Documentos de casos de uso
- Bocetos de interfaz de usuario

Diseño:

- Descripción del diseño arquitectonico
- Descripción del diseño de interfaces, módulos y objetos.
- Descripción del diseño de bases de datos

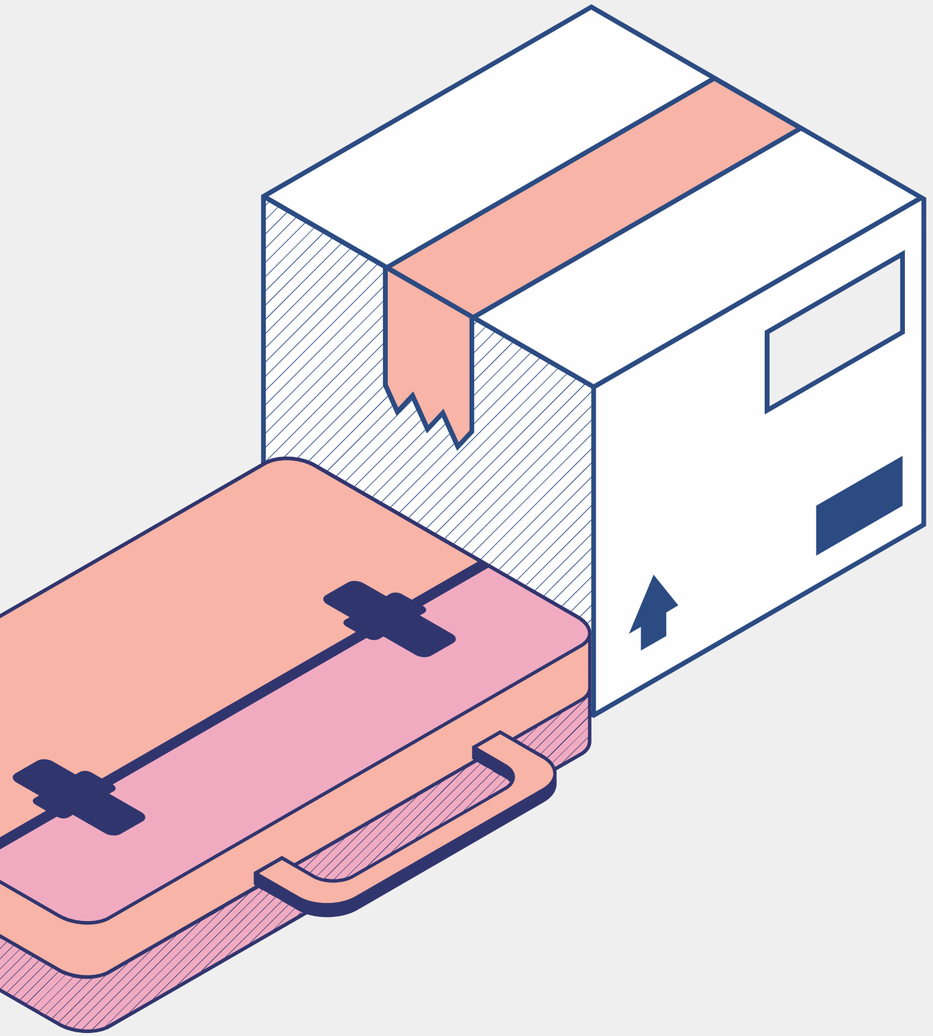
Construcción:

- Código Fuente
- Ejecutables y recursos

Prueba:

- Plan y procedimiento de pruebas.
- Casos de prueba y resultados.
- Datos de prueba

Ejemplos de ECS Procesos de Apoyo



Gestión de Proyectos:

- Plan del Proyecto.
- Contratos.

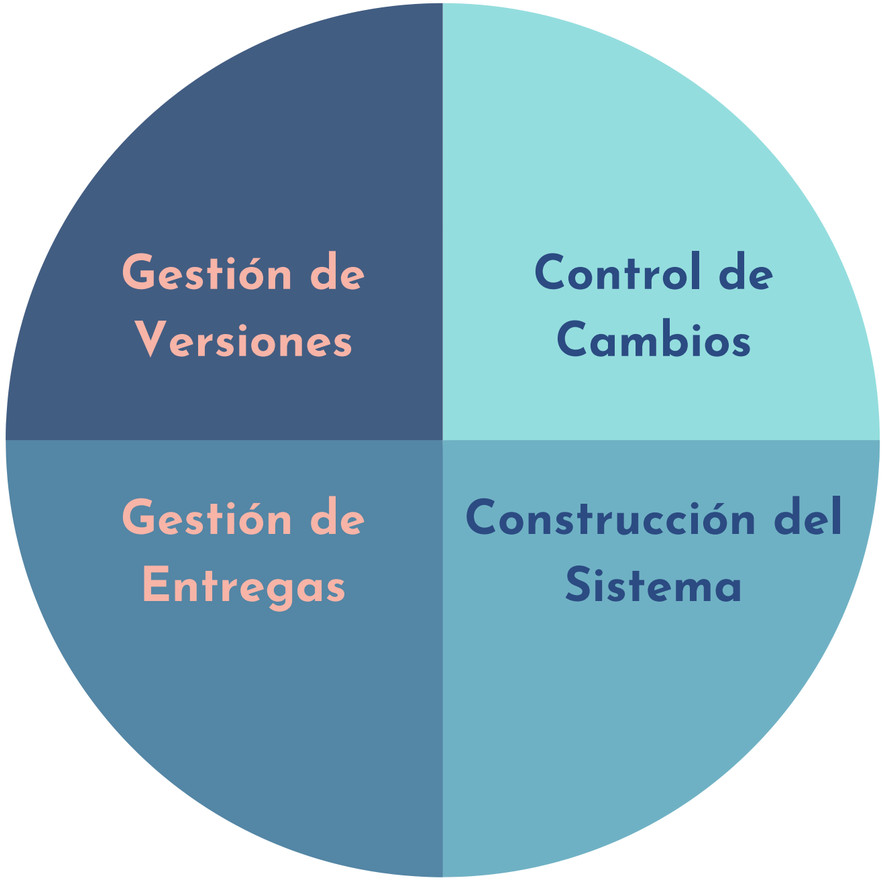
Aseguramiento de la calidad:

- Plan de SQA.
- Estándares y procedimientos.
- Métricas y resultados.
- Herramientas.

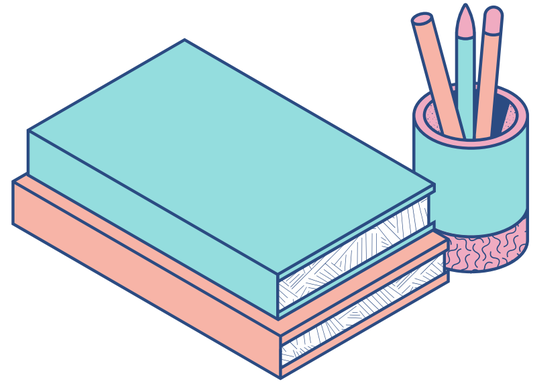
Actividades de SCM

Dentro de la gestión de versiones:

- Identificación de ECS.
- Control de versiones
- Organización del repositorio



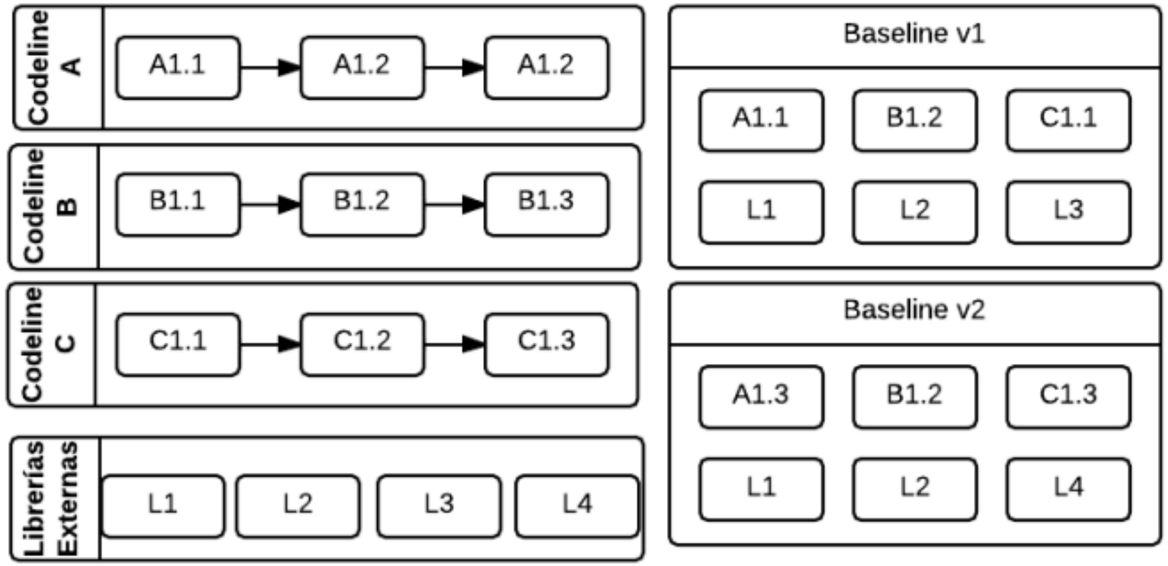
Identificación de ECS



Se requiere identificarlos en forma única:

- Identificador / Nombre / Proyecto
- Descripción:
  - Tipo (programa, documentación, datos, etc).
  - Información de la versión.
  - Información del cambio
- Control de versiones
- Organización del repositorio

Los ECS evolucionan en versiones contenidas en líneas de código (**codeline**) que pueden ser aprobadas y organizados en líneas base (**baseline**) para ser distribuidos.



Versión

Una instancia de **ECS que difiere, en alguna forma, de otras instancias del mismo ítem. Las versiones tienen un identificador único**

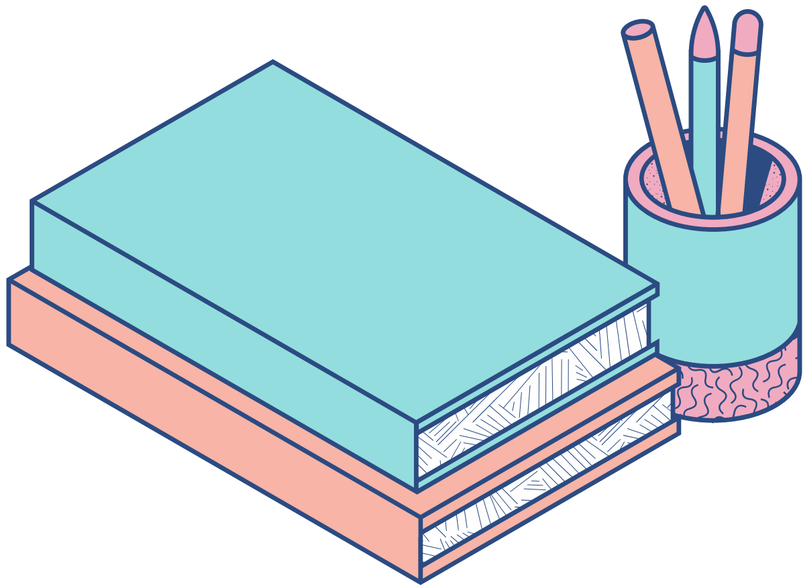
- **A1.1 -> Nombre del ECS + número de versión**

Líneas de Código(*codeline*)

Conjunto de versiones de un ECS y otros ítems de configuración de lso cuales depende dicho componente

Línea Base(base*line*)

- **Es una colección aprobada** de versiones de ECS que construyen un sistema.
- Determina un **punto de referencia** en el desarrollo del software

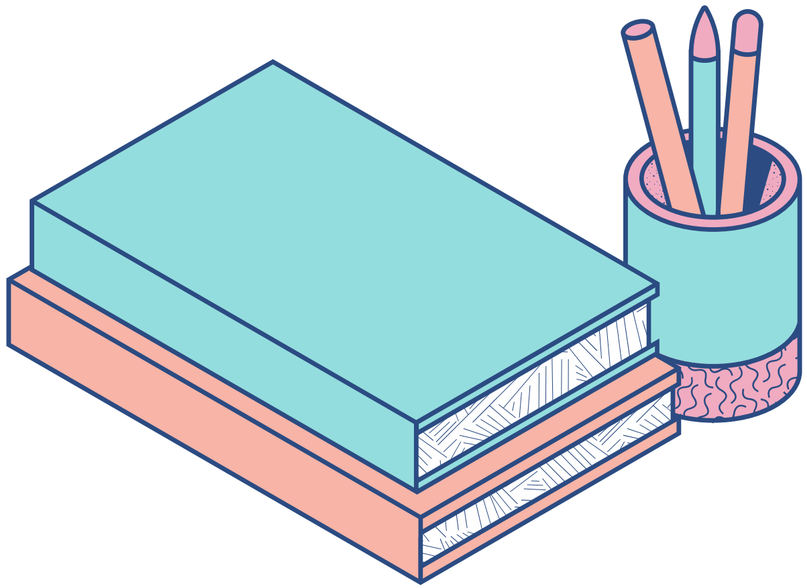


Control de Versiones

Permite especificar configuraciones alternativas del software mediante la selección de las versiones adecuadas de los componentes.

Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar y construir una configuración describiendo el conjunto de atributos deseados.

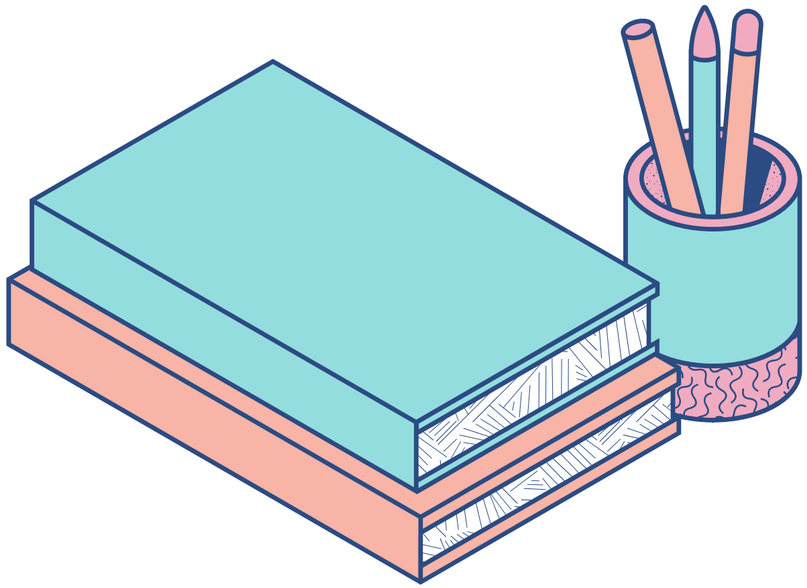
[Clemm]



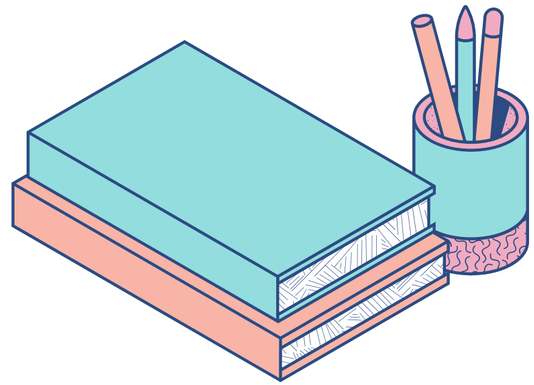
- Identificación de las versiones
- Gestión del almacenamiento.
- Operaciones de acceso concurrentes.
- Registro de historial de cambios.

Control de Cambios

Proporcionar mecanismos para evitar el caos ante cambios no controlados. Teniendo un proceso de control de cambios, siendo este proceso por el cual se implementan los mecanismos de control de cambios

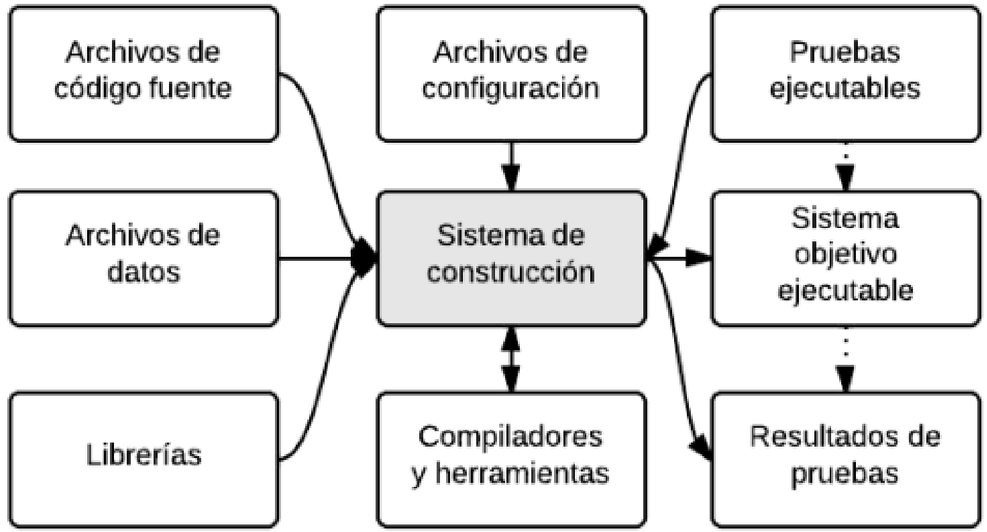


Construcción del sistema



Proceso de ensamblar los componentes del programa, datos y librerías. Compilarlos y vincularlos para crear un sistema ejecutable

- Generación de rutinas (scripts) de construcción
- Integración del sistema de control de versiones.
- Creación de sistema ejecutable.
- Automatización de pruebas.
- Informes y generación de documentación.

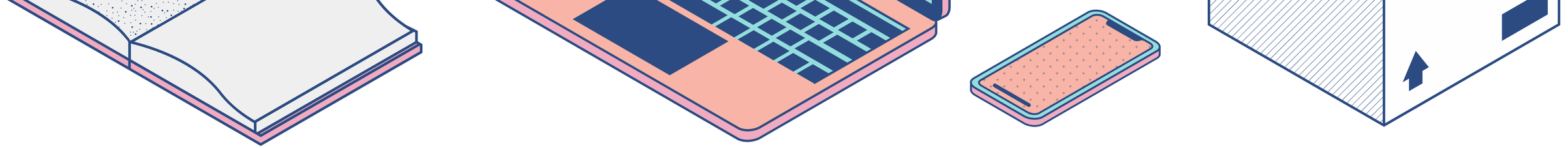


Repositorio

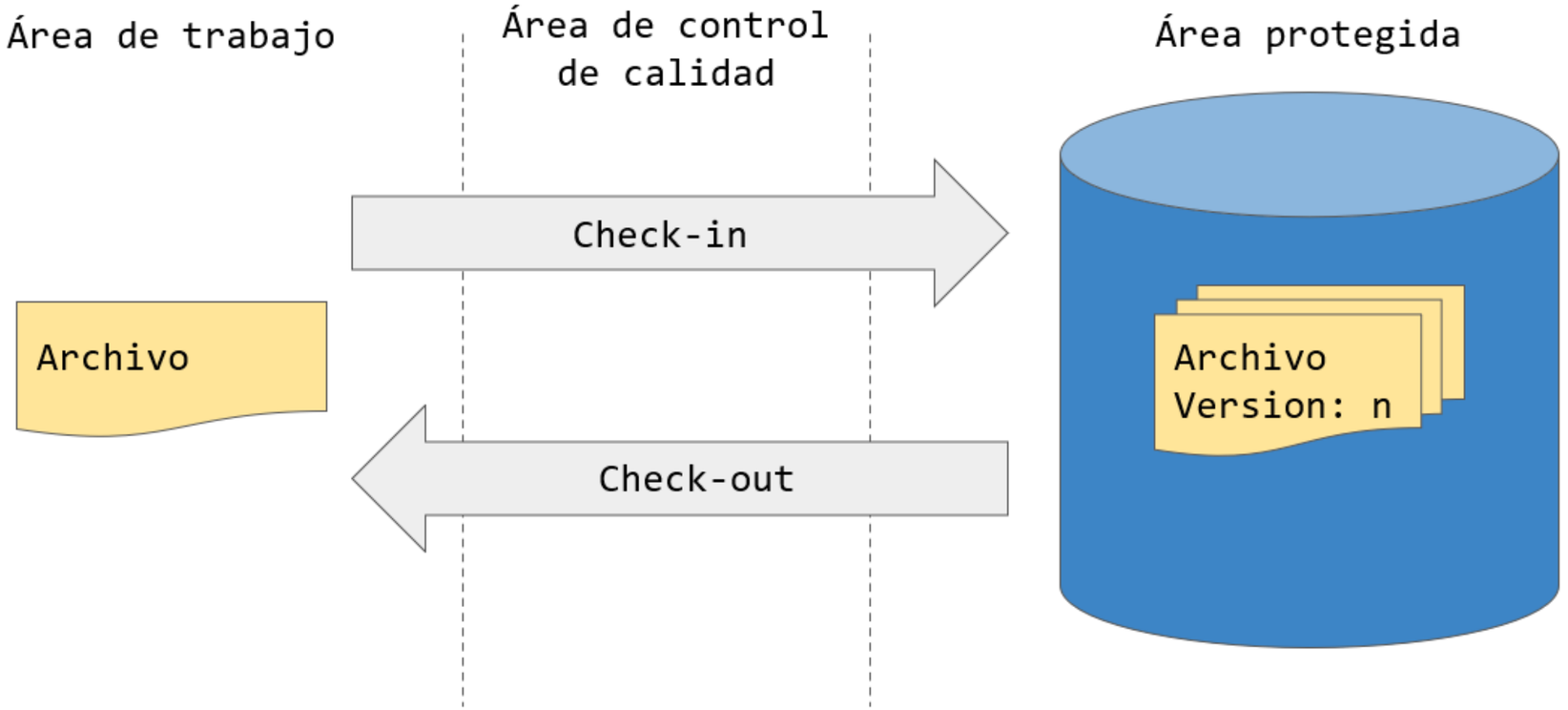
¿DÓNDE LLEVAMOS LOS ECS?

Estructura de directorios en disco, donde se **almacenan** los elementos de software producidos a lo largo de **todo el proyecto**

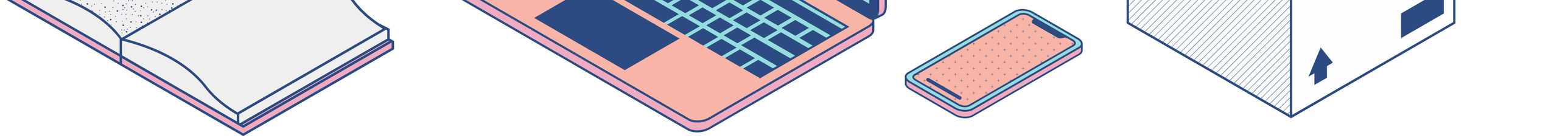
- **Área de desarrollo o trabajo.**
- **Área de control de calidad.**
- **Área protegida.**



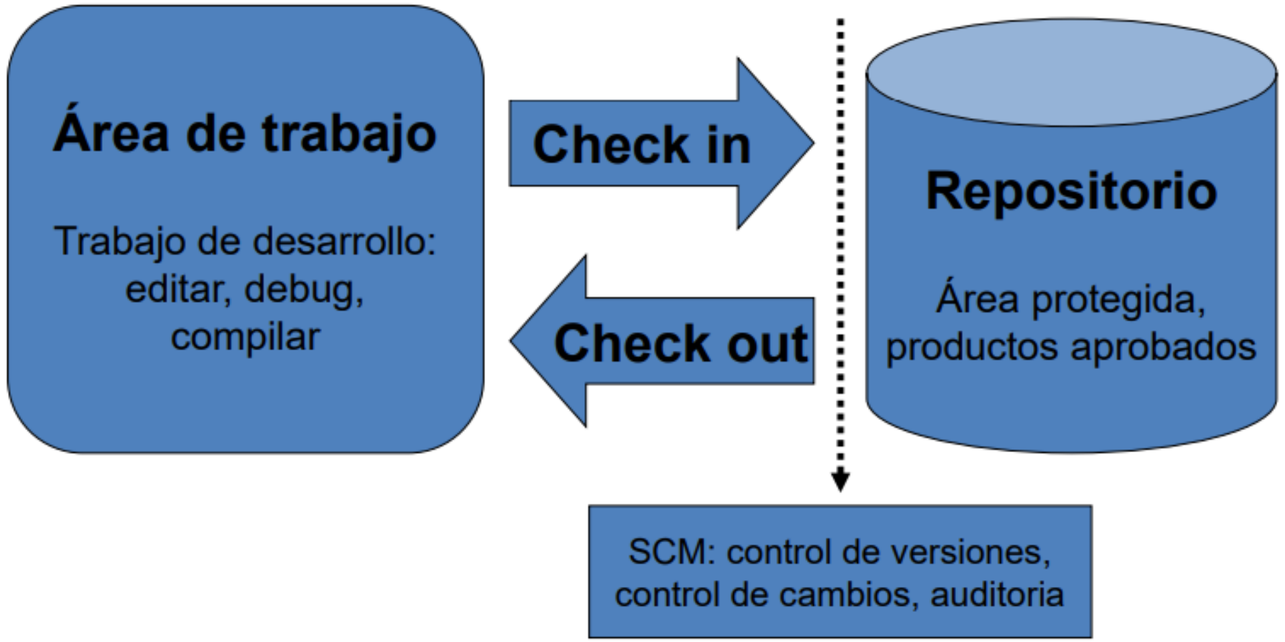
Dos Operaciones



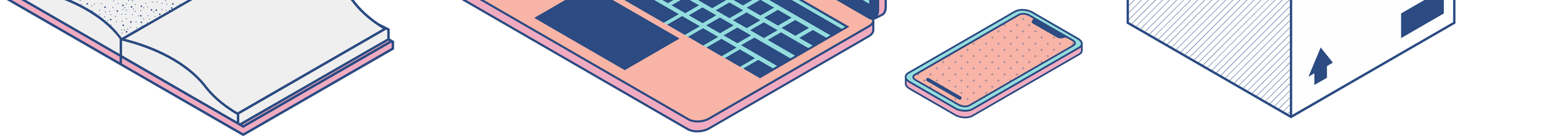
El objetivo de definir dos operaciones de acceso al repositorio (**check in / check out**) es controlar la **CONCURRENCIA** de los cambios.



Dos Operaciones



- Con **Check In** podemos ingresar un nuevo objeto al repositorio, o ingresar una nueva copia de un objeto ya existente.
- Con **Check Out** podemos obtener una copia de algún objeto del repositorio, sobre el que trabajará la persona. El objeto queda en el área de trabajo del usuario para ser editado.



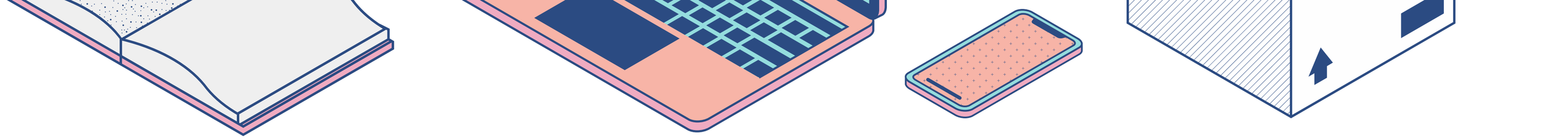
Dos modelos de Serialización de cambios

Lock-Modify-Unlock

- **Ventajas:**
  - No hay conflictos por cambios en paralelo.
- **Desventajas:**
  - Problemas administrativos: dependencia entre usuarios por desbloqueo de archivos.
  - Serialización innecesaria: Granularidad del bloqueo. Cambios en un mismo archivo pero en diferentes partes del mismo. Bloqueo innecesario

Copy-Modify-Merge

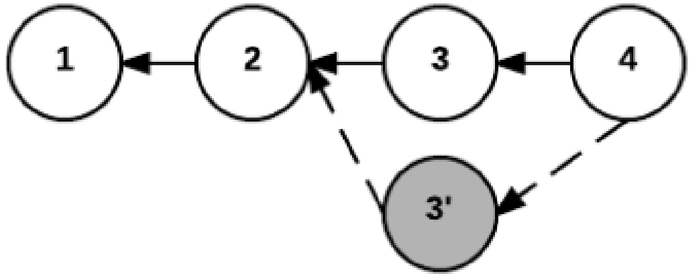
- **Ventajas:**
  - Minimiza la serialización de los cambios
- **Desventajas:**
  - **Dificulta la resolución de conflictos por cambios concurrentes sobre mismas partes del elemento de configuración.**
  - **Supone que los conflictos pueden ser resuelto por el usuario (formato de archivos entendibles por el usuario)**



Dos tipos de Repositorio

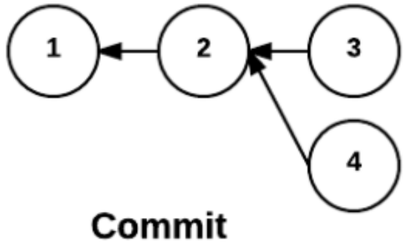
Centralizado

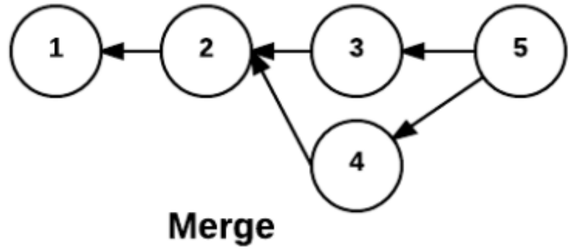
Los repositorios centrlizados buscan simplificar la evolución de las versiones en forma lineal. Cuando hay cambios concurrentes obliga a combinar los cambios antes del commit (*merge before commit*)



Distribuido/Descentralizado

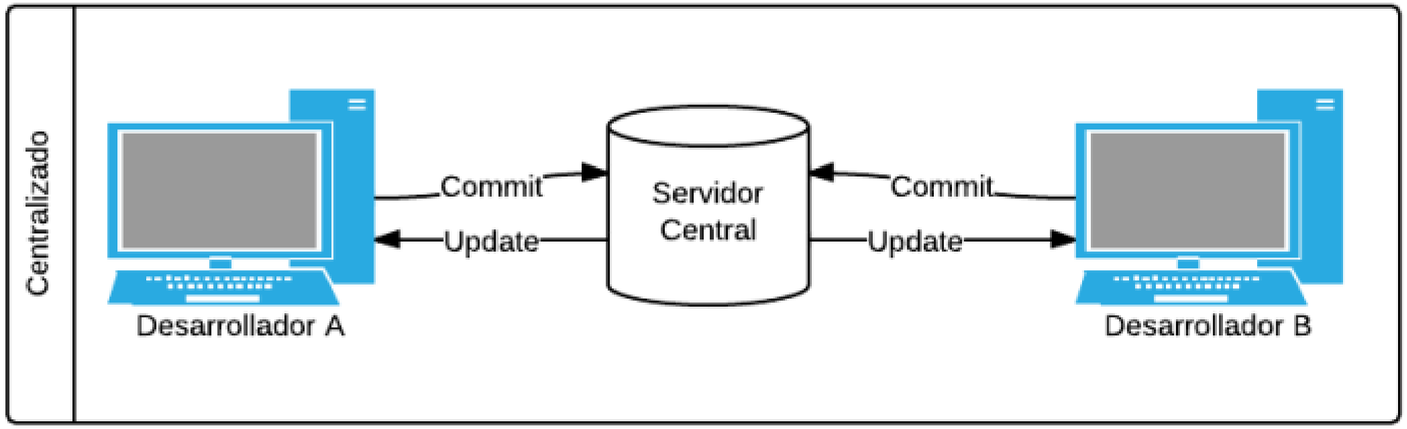
Promueve el uso de ramificaciones (*branches*) en el desarrollo. Si hay cambios concurrentes se guardan en el repositorio las versiones concurrentes que eventualmente se pueden combinar (*commit before merge*)

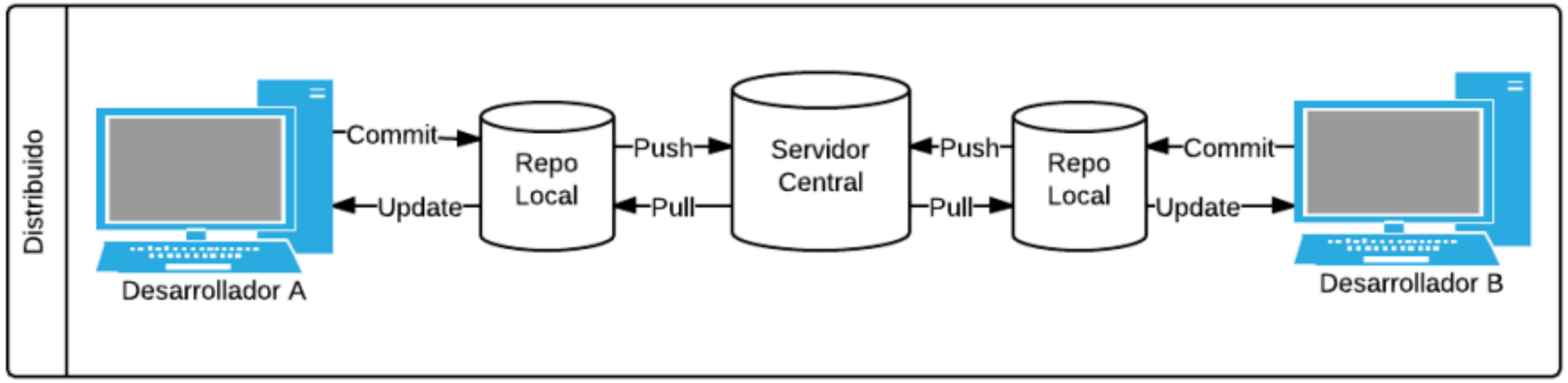


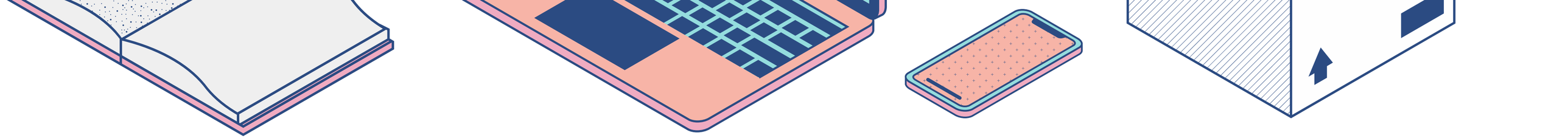


Dos tipos de Repositorio

En el CENTRALIZADO los cambios van directo al servidor central, mientras que en el **DISTRIBUIDO** hay un repositorio local de por medio



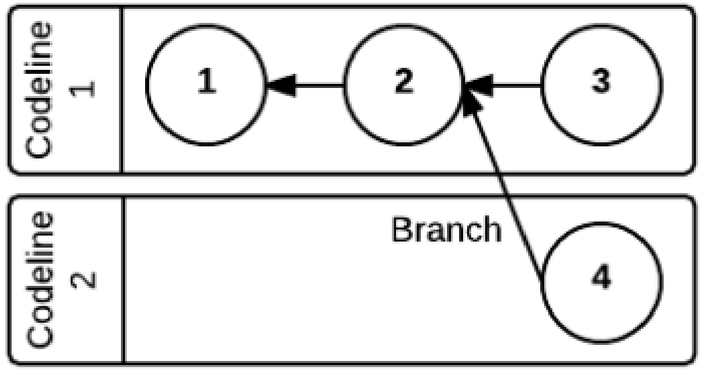




Ramificación y Combinación

Ramificación (*branching*)

La creación de una nueva línea de código a partir de una versión en una línea de código existente



Combinación (*merge*)

La creación de una nueva versión al combinar versiones separadas de diferentes líneas de código