



# INGENIERIA DE SOFTWARE II 2025

Clase 2

# Contenidos

---

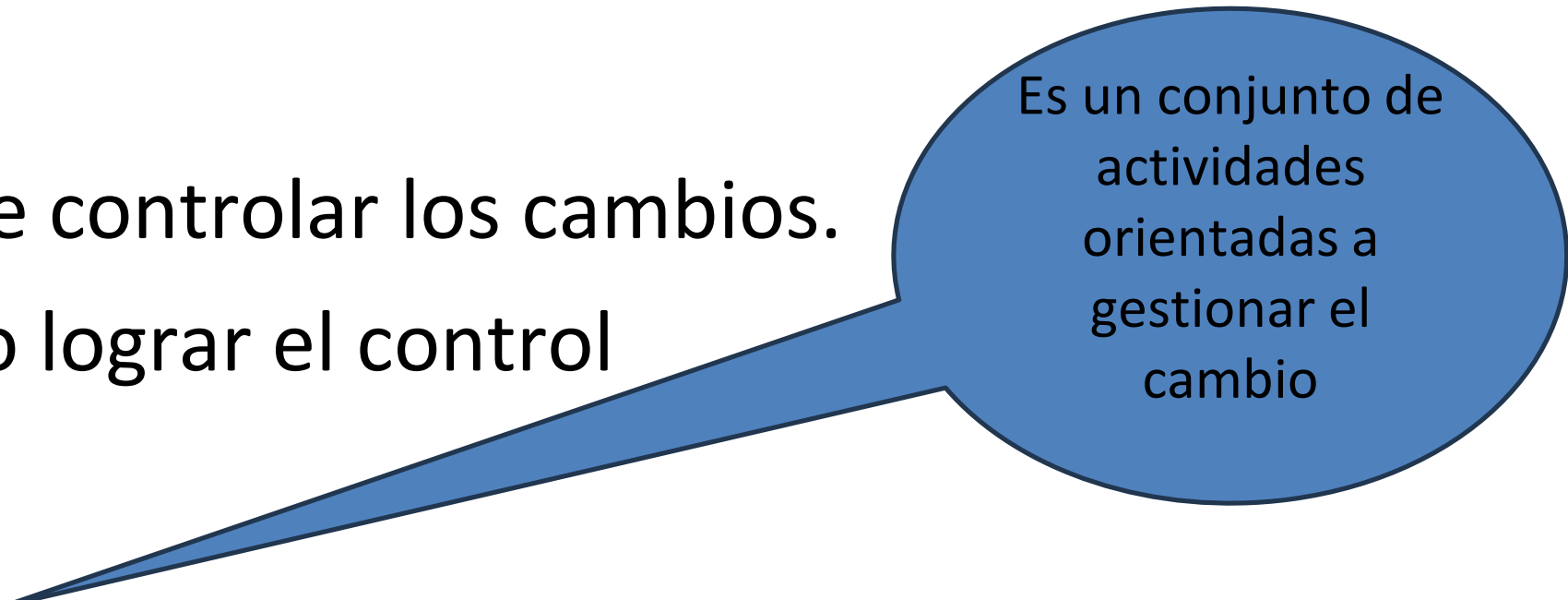
- ❖ Gestión de la Configuración del Software (GCS)
- ❖ Gestión de Proyectos
  - Planificación Organizativa

# Gestión de la Configuración del Software (GCS)

---

Los cambios suceden ??

- Importancia de controlar los cambios.
- Perjuicio de no lograr el control



Es un conjunto de actividades orientadas a gestionar el cambio

GCS: ¿Qué es ?

# Gestión de la Configuración del Software (GCS)

---

Gestión de Configuración es el proceso de:

- Identificar y definir los elementos en el sistema, controlando el cambio de estos elementos a lo largo de su ciclo de vida, registrando y reportando el estado de los elementos y las solicitudes de cambio, y verificando que los elementos estén completos y que sean los correctos.

Es una actividad de autoprotección que se aplica durante el proceso del software.

## ¿Quién hace la GCS?

---

Todos los involucrados en el proceso de software se relacionan en cierta medida con la gestión del cambio, pero en ocasiones se crean posiciones de apoyo especializadas para administrar el proceso de aseguramiento de la calidad del software.

# Elementos de la GCS (ECS)

---

Se dividen en 3 categorías:



## Programas

Aplicaciones de software utilizadas para tareas.



## Productos de Trabajo

Resultados creados durante el proceso de software.



## Datos

Información utilizada y generada por el software.

# Elementos de la GCS (ECS)

***¿Cuáles podrían ser elementos de la configuración?***

<ul style="list-style-type: none"><li>✓ Especificación del sistema</li><li>✓ Plan del proyecto software</li><li>✓ Especificación de diseño:<ul style="list-style-type: none"><li>✓ a) Diseño preliminar</li><li>✓ b) Diseño detallado</li></ul></li><li>✓ Listados del código fuente</li><li>✓ Planificación y procedimiento de prueba<ul style="list-style-type: none"><li>✓ Casos de prueba y resultados registrados</li></ul></li></ul>	<ul style="list-style-type: none"><li>✓ Manuales de operación y de instalación</li><li>✓ Programas ejecutables</li><li>✓ Descripción de la base de datos<ul style="list-style-type: none"><li>a) Esquema, modelos</li><li>b) Datos iniciales</li></ul></li><li>✓ Manual de usuario</li><li>✓ Documentos de mantenimiento</li><li>✓ Estándares y procedimientos de ingeniería del software</li></ul>
--	---



# Organización de un repositorio de ECS

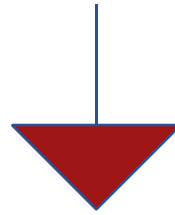




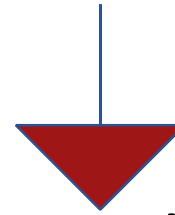
# Gestión de la Configuración del Software (GCS)

---

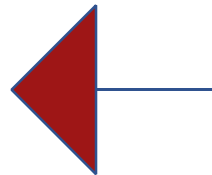
Elementos de la configuración (ECS)



ECS - Cambian constantemente

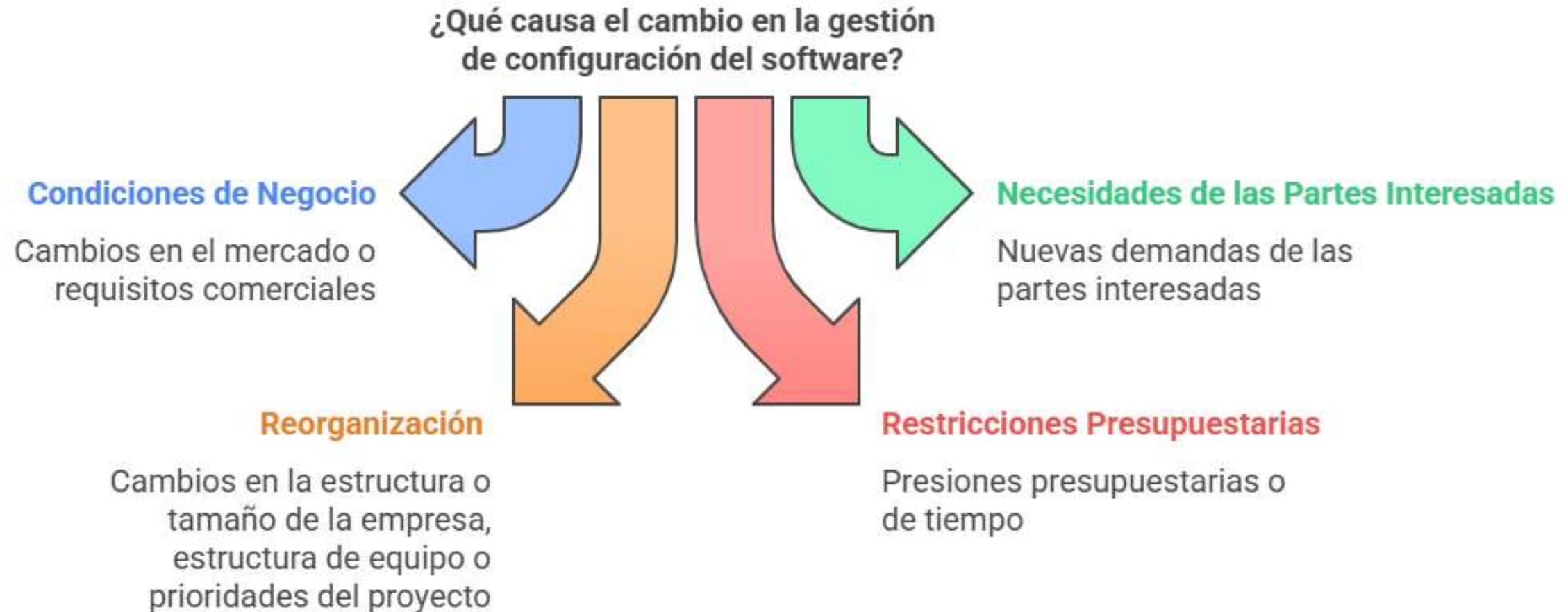


GCS



Control muy exhaustivo de esos cambios

# Gestión de la Configuración del Software (GCS)



# GCS – Línea base

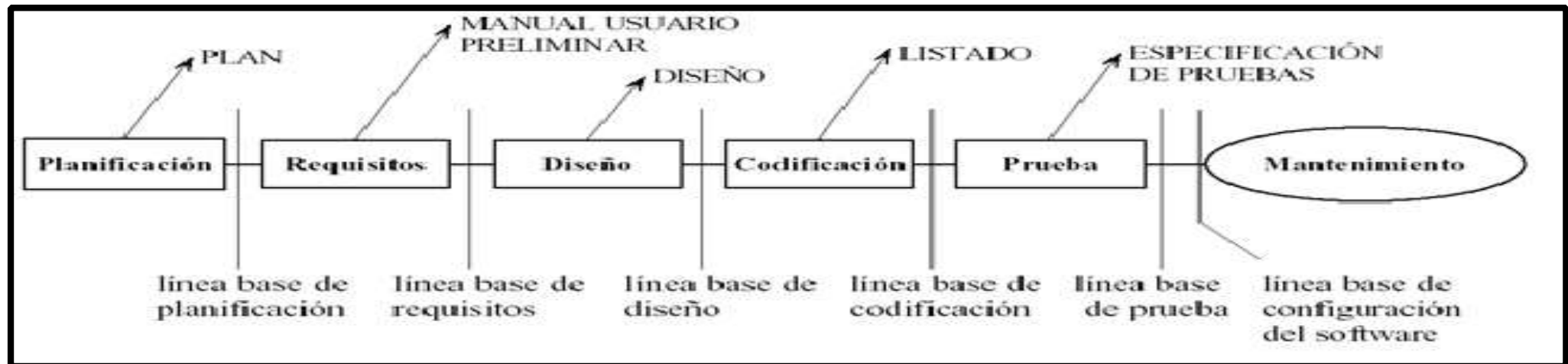
Una línea base es un concepto de GCS que nos ayuda a controlar los cambios

Definición de la IEEE

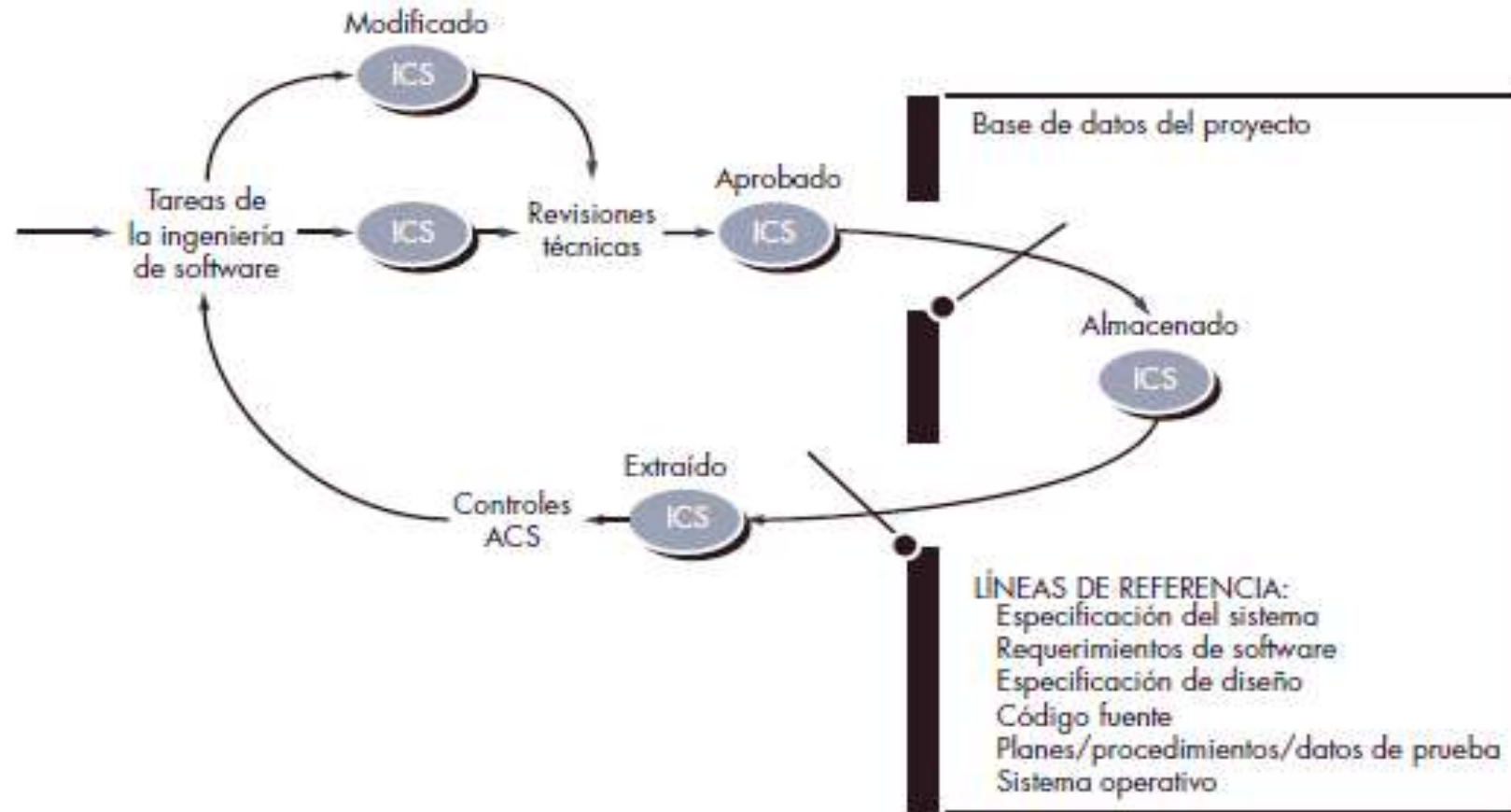
*Una especificación o producto que se ha revisado formalmente y sobre el que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambio*

En el contexto de la Ingeniería de Software:

*Una línea base es un punto de referencia en el desarrollo del software que queda marcado por el envío de uno o más ECS y su aprobación*



# GCS – Línea base



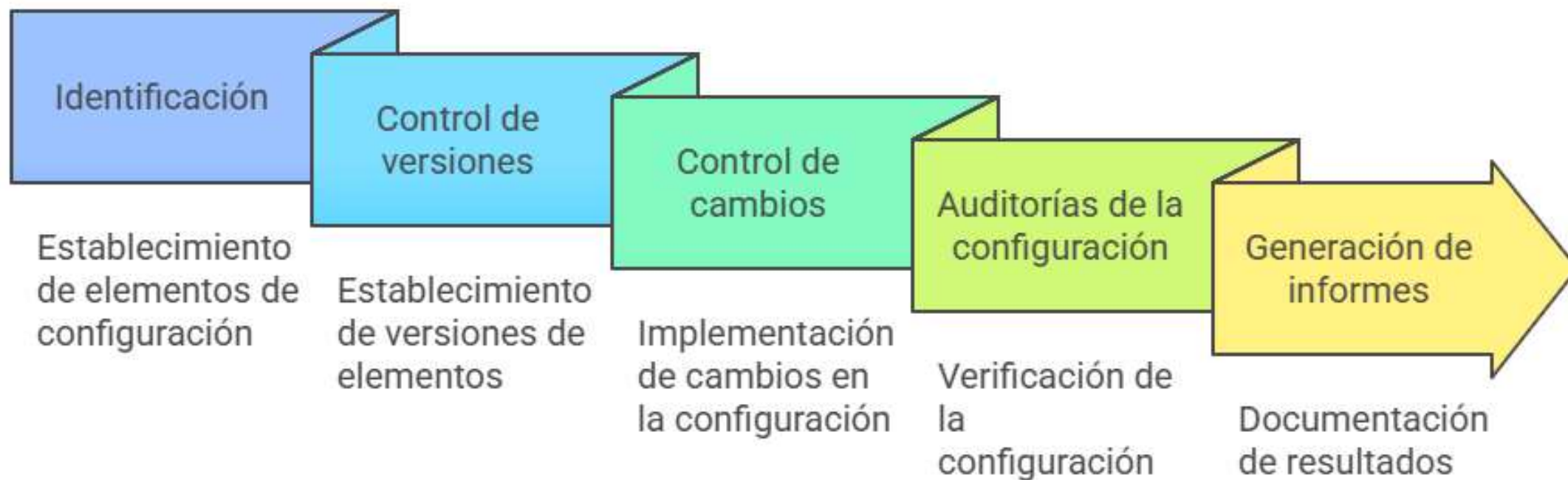
# GCS - Importancia

---

- ❖ ¿Cómo identifica y gestiona una organización las diferentes versiones existentes de un programa (y su documentación) de forma que se puedan introducir cambios eficientemente?
- ❖ ¿Cómo controla la organización los cambios antes y después de que el software sea distribuido al cliente?
- ❖ ¿Quién tiene la responsabilidad de aprobar y de asignar prioridades a los cambios?
- ❖ ¿Cómo podemos garantizar que los cambios se han llevado a cabo adecuadamente?
- ❖ ¿Qué mecanismo se usa para avisar a otros de los cambios realizados?

# GCS - Proceso

## Proceso de Gestión de Configuración





# GCS - Proceso

## 1- Identificación de los elementos en la GCS

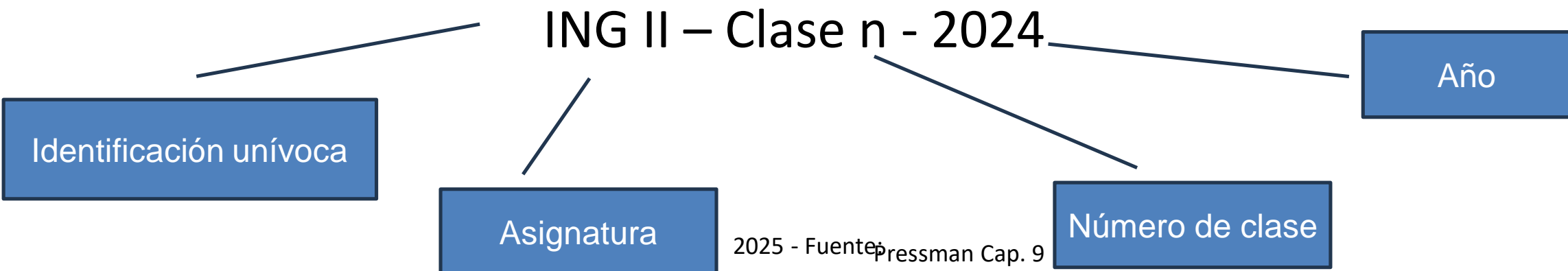
*Nombre: cadena de caracteres sin ambigüedad*

*Descripción: lista de elementos de datos que identifican:*

Tipo de ECS (documento, código fuente, datos)

Identificador del proyecto

Información de la versión y/o cambio



# GCS - Proceso

---

## 2 - Control de versiones

*Combinación de procedimientos y herramientas para gestionar las versiones de los ECS que se crean a lo largo del proceso de software.*

### Ejemplo de versiones

Un programa puede contener los módulos 1-2-3-4-5

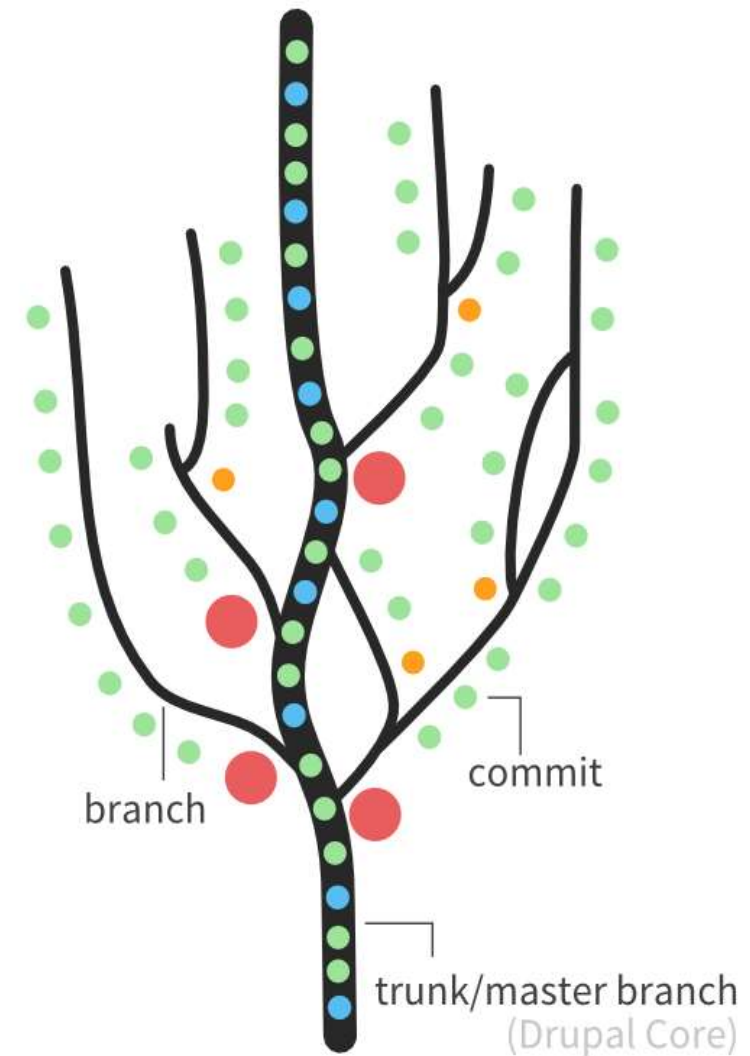
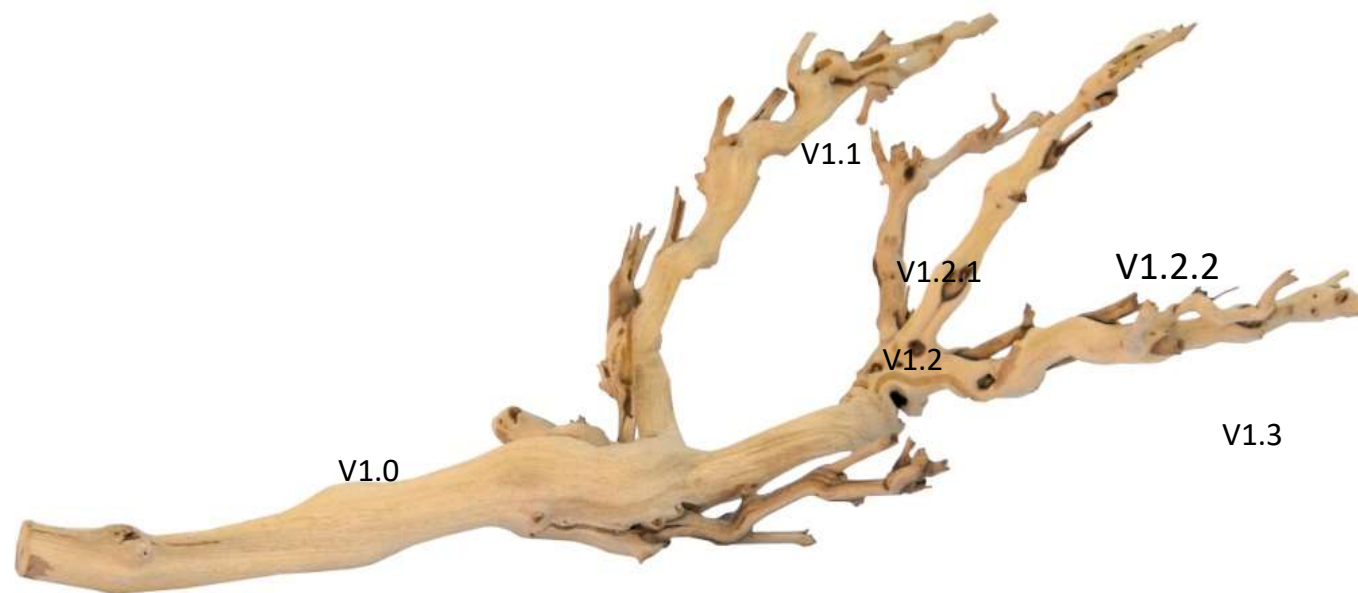
Una versión puede utilizar los módulos 1-2-3-5

Otra versión puede utilizar los módulos 1-2-4-5

Dos variantes de un mismo programa

# GCS - Proceso

## 2 - Control de versiones



# GCS - Proceso

## 2 - Control de versiones

<b>Repositorio</b>	Se almacenan los archivos actualizados e históricos de cambio del proyecto.
<b>Versión</b>	Determina un conjunto de archivos
<b>Master</b>	Conjunto de archivos principales del proyecto
<b>Abrir rama – branch</b>	Bifurcación del máster para trabajar sobre dos ramas de forma independiente
<b>Desplegar – check-out</b>	Copia de trabajo local desde el repositorio.
<b>Publicar - Commit</b>	Una copia de los cambios hechos a una copia local es escrita o integrada sobre repositorio
<b>Conflicto</b>	Problema entre las versiones de un mismo documento
<b>Cambio – diff</b>	Representa una modificación específica
<b>Integración – Merge</b>	Fusión entre dos ramas del proyecto
<b>Actualización – sync o update</b>	Integra los cambios que han sido hechos en el repositorio y las copias locales

# Gestión de la Configuración del Software (GCS)

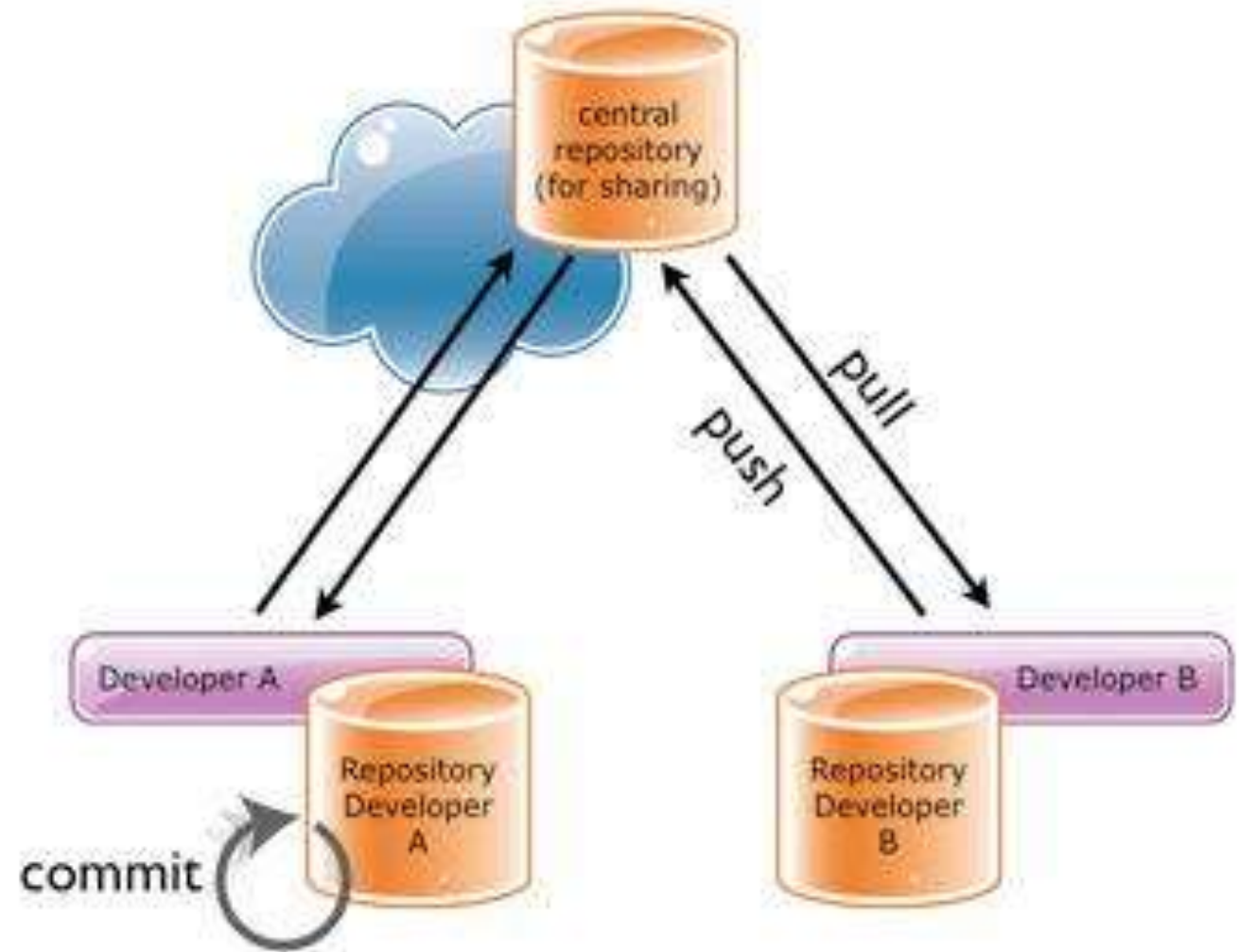
## »Proceso de la GCS

### 2 - Control de versiones

Puede utilizarse el software

Concurrent Versions System (CVS)

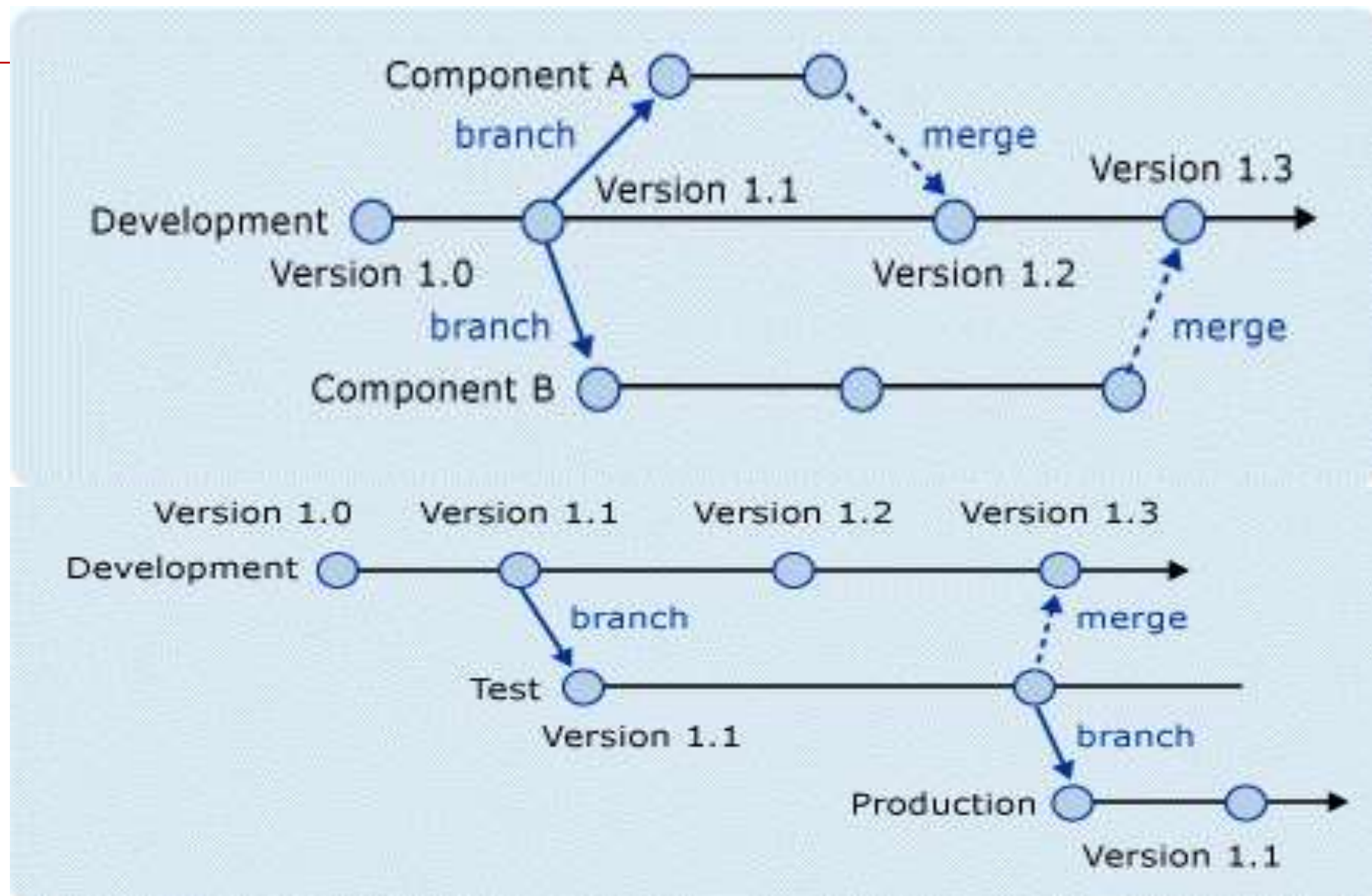
<https://www.nongnu.org/cvs/>





# GCS - Proceso

## 2 - Control de versiones





# GCS - Proceso

---

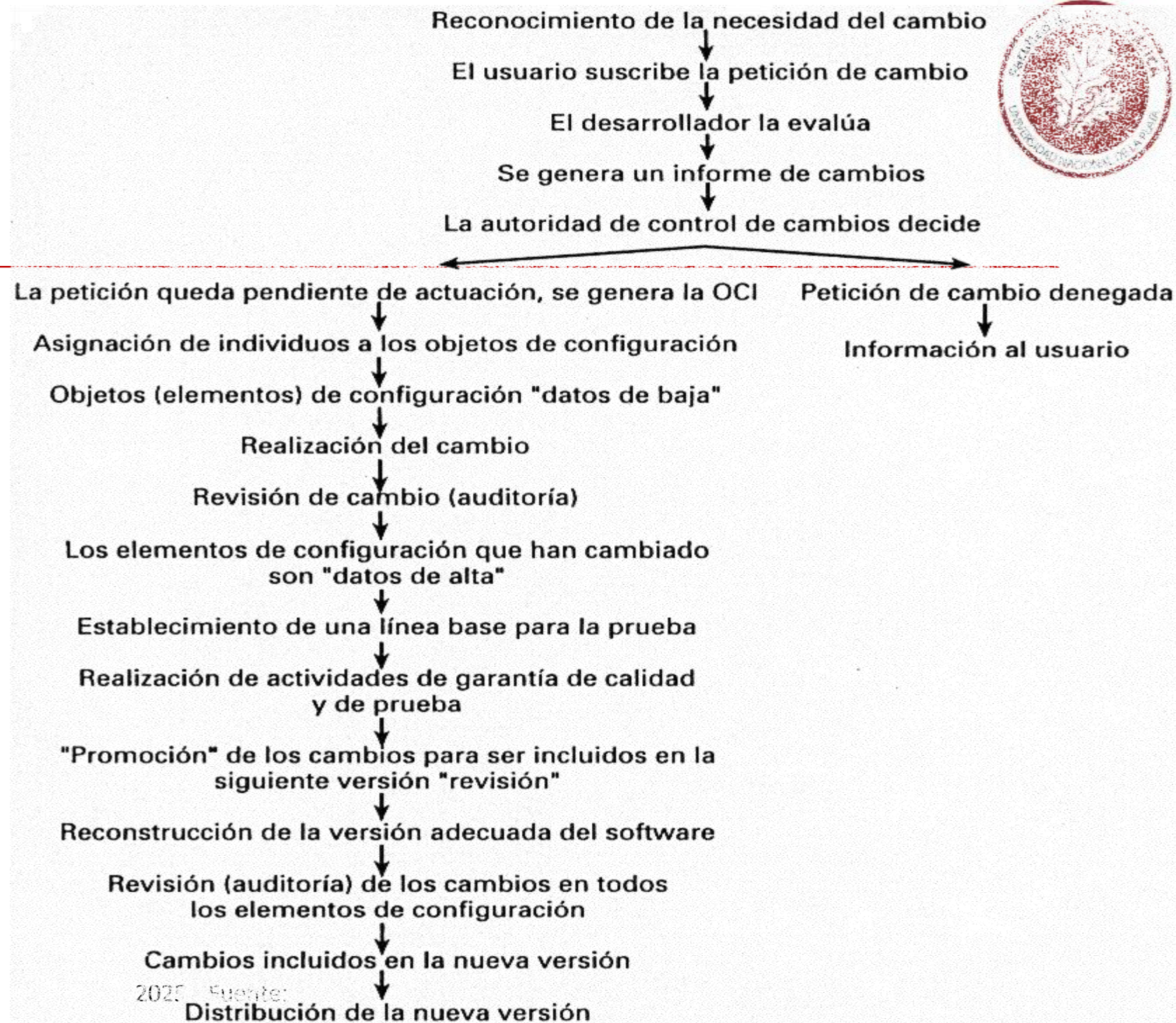
## 3 - Control de cambios

A lo largo del proyecto los cambios son inevitables y el control es vital para el desarrollo del mismo

Combina los procedimientos humanos y las herramientas adecuadas para proporcionar un mecanismo para el control del cambio

# GCS - Proceso

## 3 -Control de cambios



# GCS - Proceso

---

## 3 -Control de cambios

*La autoridad de control de cambios (ACC) evalúa:*

¿Cómo impactará el cambio en el hardware?

¿Cómo impactará el cambio en el rendimiento?

¿Cómo alterará el cambio la percepción del cliente sobre el producto?

¿Cómo afectará el cambio a la calidad y a la fiabilidad?

...

# GCS - Proceso

---

## 4 - Auditoría de la configuración

*La identificación y el control de versiones y el control de cambio, ayudan al equipo de desarrollo de software a mantener un orden, pero sólo se garantiza hasta que se ha generado la orden de cambio.*

*Cómo aseguramos que el cambio se ha realizado correctamente*

Revisiones técnicas formales

Auditorías de configuración



# GCS - Proceso

---

## 4 - Auditoría de la configuración responde:

*¿Se ha hecho el cambio especificado en la Orden de Cambio? ¿Se han incorporado modificaciones adicionales?*

*¿Se ha llevado a cabo una RTF para evaluar la corrección técnica?*

*¿Se han seguido adecuadamente los estándares de IS?*

*¿Se han reflejado los cambios en el ECS: fecha, autor, atributos?*

*¿Se han seguido procedimientos de GCS para señalar el cambio, registrarlo y divulgarlo?*

*¿Se han actualizado adecuadamente todos los ECS relacionados?*

# GCS - Proceso

---

## 5 - Generación de informes de estado de la configuración (auditoría)

*Responde*

¿Qué pasó?

¿Quién lo hizo?

¿Cuándo pasó?

¿Qué más se vio afectado?

*La generación de informes de estado de la configuración desempeña un papel vital en el éxito del proyecto*





# Ingeniería de software II

Gestión De Proyectos

Fuente:

# ¿Qué es un proyecto ?

---

» Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único.

» Características

Temporal

*Tiene un comienzo y fin definido.*

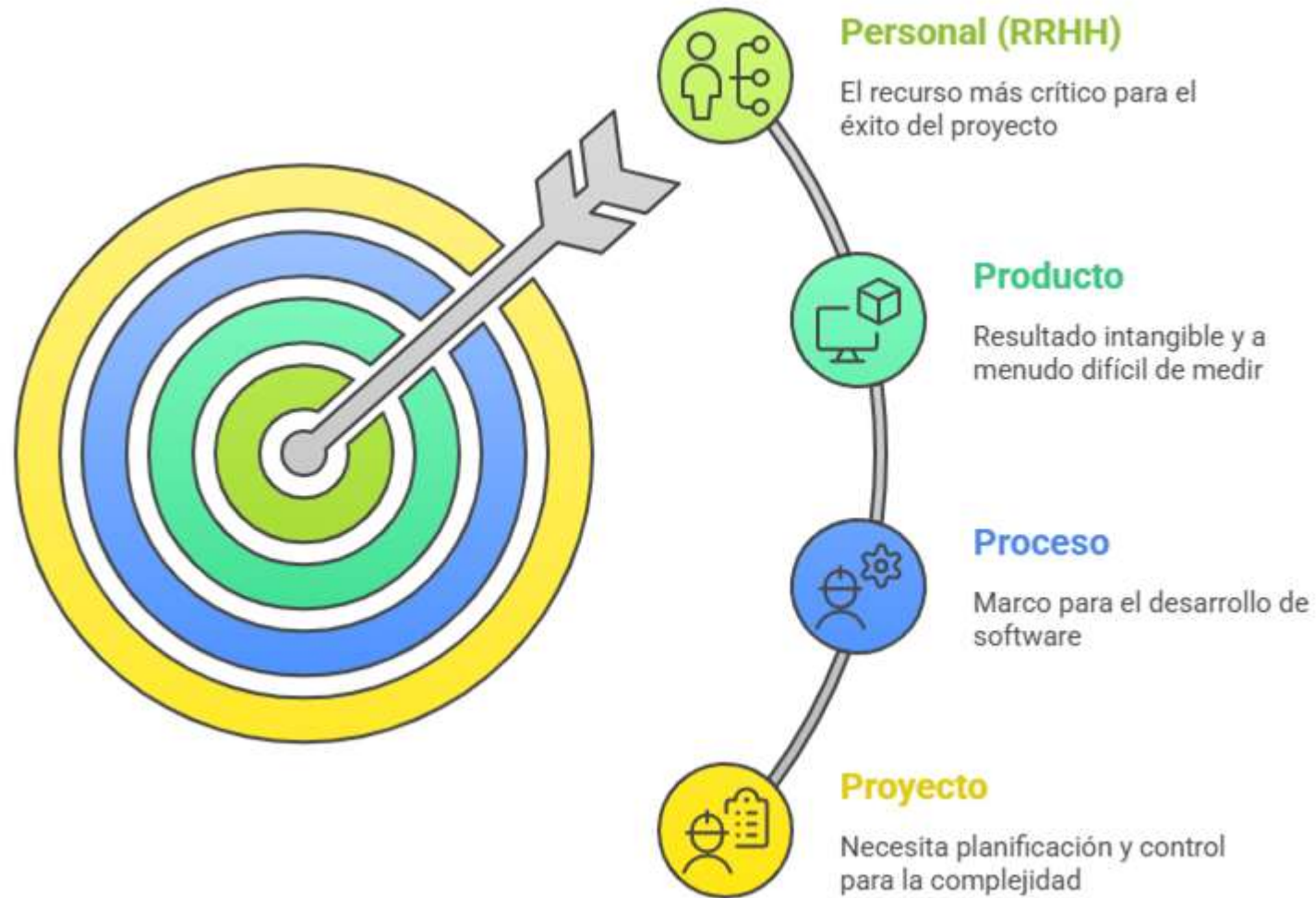
Resultado

*Productos, servicios o resultados únicos*

Elaboración gradual

*Desarrollar en pasos e ir aumentando mediante incrementos*

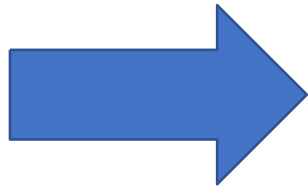
# Las 4 P de la Gestión de Proyectos de Software



# Manifestación de una mala gestión de proyectos

---

- » Incumplimiento de plazos
- » Incremento de los costos
- » Entrega de productos de mala calidad



Perjuicio  
económico



# Elementos clave de la gestión de proyectos

---

- » Métricas
  - » Estimaciones
  - » Calendario temporal
  - » Organización del personal
  - » Análisis de riesgos
  - » Seguimiento y control
- } PLANIFICACIÓN

*La gestión de proyectos cubre todo el proceso  
de desarrollo*



# Gestión de Proyectos

---

## Planificación





# Planificación

---

» La planificación indica:

1. qué debe hacerse,
2. con qué recursos
3. y en qué orden.

Establece una secuencia operativa.

# Planificación organizativa

---

- » El personal que trabaja en una organización de software es el activo más grande, representa el capital intelectual.
- » Una mala administración del personal es uno de los factores principales para el fracaso de los proyectos
- » Debido a su importancia, se creó el Modelo de madurez de capacidades de personal (P-CMM). Se reconoce que toda organización debe mejorar de manera continua su habilidad de atraer, desarrollar, motivar, organizar y conservar el personal.

# Planificación organizativa

---

## » Participantes

Gerentes ejecutivos (dueños del producto)

*Definen los temas empresariales*

Gerentes de proyecto (líderes de equipo)

*Planifican, motivan, organizan y controla a los profesionales*

Profesionales especializados

*Aportan habilidades técnicas*

Clientes

*Especifican los requerimientos*

Usuarios finales

*Interactúan con el software*

# Líderes de equipo

---

El Equipo de software debe organizarse de manera que maximice las habilidades y capacidades de cada persona.

Las prácticas que realizan líderes ejemplares son:

- ❖ Modelar el camino: Practicar lo que predicán. Demostrar compromiso con el equipo y el proyecto.
- ❖ Inspirar y visión compartida: Es importante motivar a los miembros del equipo, esto implica involucrar a las partes al principio del proceso de establecimiento de metas
- ❖ Desafiar el proceso: Tomar la iniciativa de buscar formas innovadoras para mejorar el trabajo. Animar a los miembros a experimentar y asumir riesgos.
- ❖ Permitir que otros actúen: Fomentar habilidades colaborativas del equipo, generando confianza y facilitando relaciones.
- ❖ Alentar: Celebrar los logros individuales. Generar espíritu comunitario.

# El equipo de software

---

La mejor estructura de equipo depende del estilo gerencial de la organización, el número de personas que conformarán el equipo y sus niveles de habilidad, y la dificultad del problema en general.

Como regla general, el equipo no debe tener más de 10 miembros. De todos modos eso depende del tamaño del proyecto.

# El equipo de software

---

## Factores a considerar cuando se planea una estructura de equipo

1. Dificultad del problema a resolver
2. Tamaño del programa resultante
3. Tiempo que el equipo permanecerá unido
4. Grado en que puede dividirse en módulos el problema a resolver
5. Calidad y confiabilidad requerida por el sistema a construir
6. Rigidez de la fecha de entrega
7. Grado de sociabilidad requerido para el proyecto



# El equipo de software

---

Para evitar :

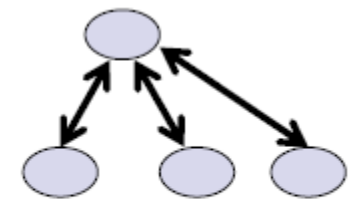
1. **Atmósfera de trabajo frenética**-> El equipo tiene acceso a toda la información y las principales metas y objetivos no se modifican a menos que sea absolutamente necesario.
2. **Fricción entre los miembros del equipo**-> Se define responsabilidad a cada uno
3. **Proceso de software fragmentado o mal coordinado**-> Entender lo que se va a crear y permitir que el equipo seleccione el modelo de proceso
4. **Definición imprecisa de roles**-> Definir enfoques correctivos ante un miembro que no cumple
5. **Exposición continua y repetida al fracaso**-> Utilización de técnicas basadas en la retroalimentación y solución de problemas

# Paradigmas Organizacionales del equipo

---

1- **Paradigma cerrado** estructura un equipo conforme a una jerarquía de autoridad tradicional. (Comunicación vertical entre jefe y miembros del equipo)

Tales equipos pueden trabajar bien cuando producen software muy similar al de esfuerzos anteriores, pero será menos probable que sean innovadores cuando trabajen dentro de este paradigma.



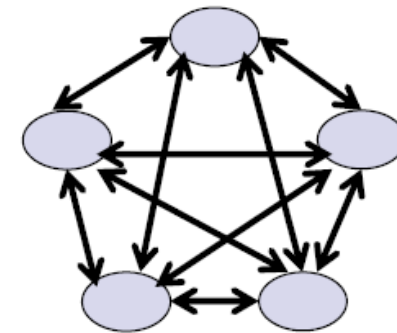
**Trayectorias de  
Comunicación**

# Paradigmas Organizacionales del equipo

---

2- **Paradigma abierto** intenta estructurar un equipo de manera que logre algunos de los controles asociados con el paradigma cerrado, pero también mucha de la innovación que ocurre cuando se usa el paradigma aleatorio. El trabajo se realiza de manera colaborativa; la gran comunicación y la toma de decisiones consensuadas constituyen las características de los equipos de paradigma abierto. Es un equipo democrático y con decisiones consensuadas.

Las estructuras de equipo de este paradigma son muy adecuadas para la solución de problemas complejos, pero pueden no desempeñarse tan eficazmente como otros equipos.



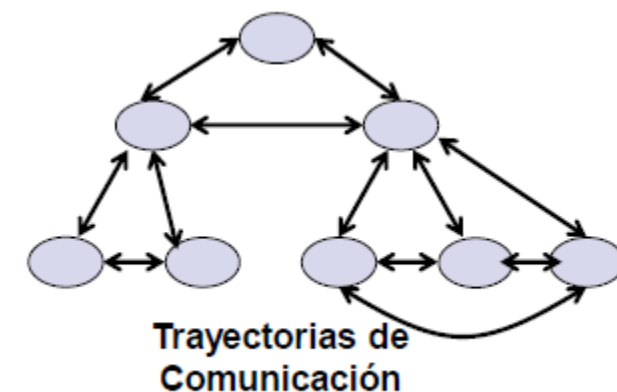
**Trayectorias de  
Comunicación**

# Paradigmas Organizacionales del equipo

3- **Paradigma síncrono** se apoya en la compartimentalización natural de un problema y organiza a los miembros del equipo para trabajar en trozos del problema con poca comunicación activa entre ellos. Es un equipo descentralizado y controlado a la vez, utiliza la política de divide y vencerás.

Este tipo de organizaciones bueno cuando la complejidad y tamaño del problema es elevada.

El fraccionamiento del problema puede llevar a crear un producto no requerido debido a que los subgrupos no tienen adecuada comunicación.



# Paradigmas Organizacionales del equipo

---

4- **Paradigma aleatorio** estructura un equipo de manera holgada y depende de la iniciativa individual de los miembros del equipo. Los miembros se organizan entre ellos y no se requiere de un líder definido. Las trayectorias de comunicación pueden ser cualquiera de los otros tres paradigmas.

Cuando se requiere innovación o avance tecnológico, destacarán los equipos que siguen este paradigma, pero pueden batallar cuando se requiera “desempeño ordenado”.

# Resumen: Paradigmas Organizacionales del equipo

¿Qué paradigma organizacional debería adoptar el equipo?

## Paradigma Cerrado

Adecuado para proyectos de software similares a los anteriores, pero puede limitar la innovación.

## Paradigma Síncrono

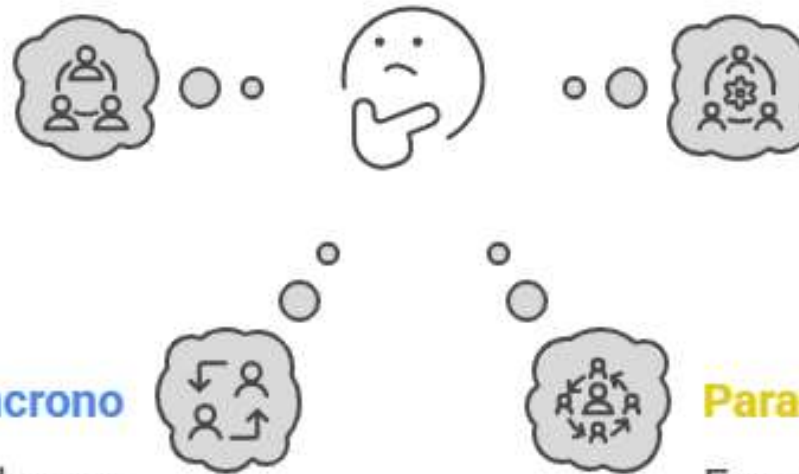
Efectivo para problemas grandes y complejos con menos comunicación, pero riesgo de falta de coordinación.

## Paradigma Abierto

Fomenta la innovación y la colaboración, ideal para la resolución de problemas complejos.

## Paradigma Aleatorio

Excelente para la innovación, pero puede carecer de orden en el rendimiento.





# El equipo de software

---

Sin importar la organización del equipo, el objetivo a alcanzar es lograr un equipo que muestre cohesión. Que se consolide.

Los miembros de un equipo consolidado son mucho más productivos y motivados.

Un grupo **cohesivo** tiene beneficios:

1. Puede establecer sus propios estándares de calidad
2. Los individuos aprenden de los demás y se apoyan mutuamente
3. El conocimiento se comparte
4. Se alienta la refactorización y el mejoramiento continuo

# El equipo de software

---

Sin importar la organización del equipo, el objetivo a alcanzar es lograr un equipo que muestre cohesión. Que se consolide.

Los miembros de un equipo consolidado son mucho más productivos y motivados.

Un equipo **no consolidado** suele sufrir toxicidad de equipo :

1. Atmósfera de trabajo frenética
2. Fricción entre los miembros del equipo
3. Proceso de software fragmentado o mal coordinado
4. Definición imprecisa de roles
5. Exposición continua y repetida al fracaso

# El equipo de software

---

## » Comunicación Grupal

Las comunicaciones en un grupo se ven influenciadas por factores como: status de los miembros del grupo, tamaño del grupo, composición de hombres y mujeres, personalidades y canales de comunicación disponible.

Se deben establecer mecanismos para la comunicación formal e informal entre los miembros del equipo

La comunicación formal: a través de reuniones, escritos y otros canales no interactivos

La comunicación informal: Los miembros comparten ideas sobre la marcha.

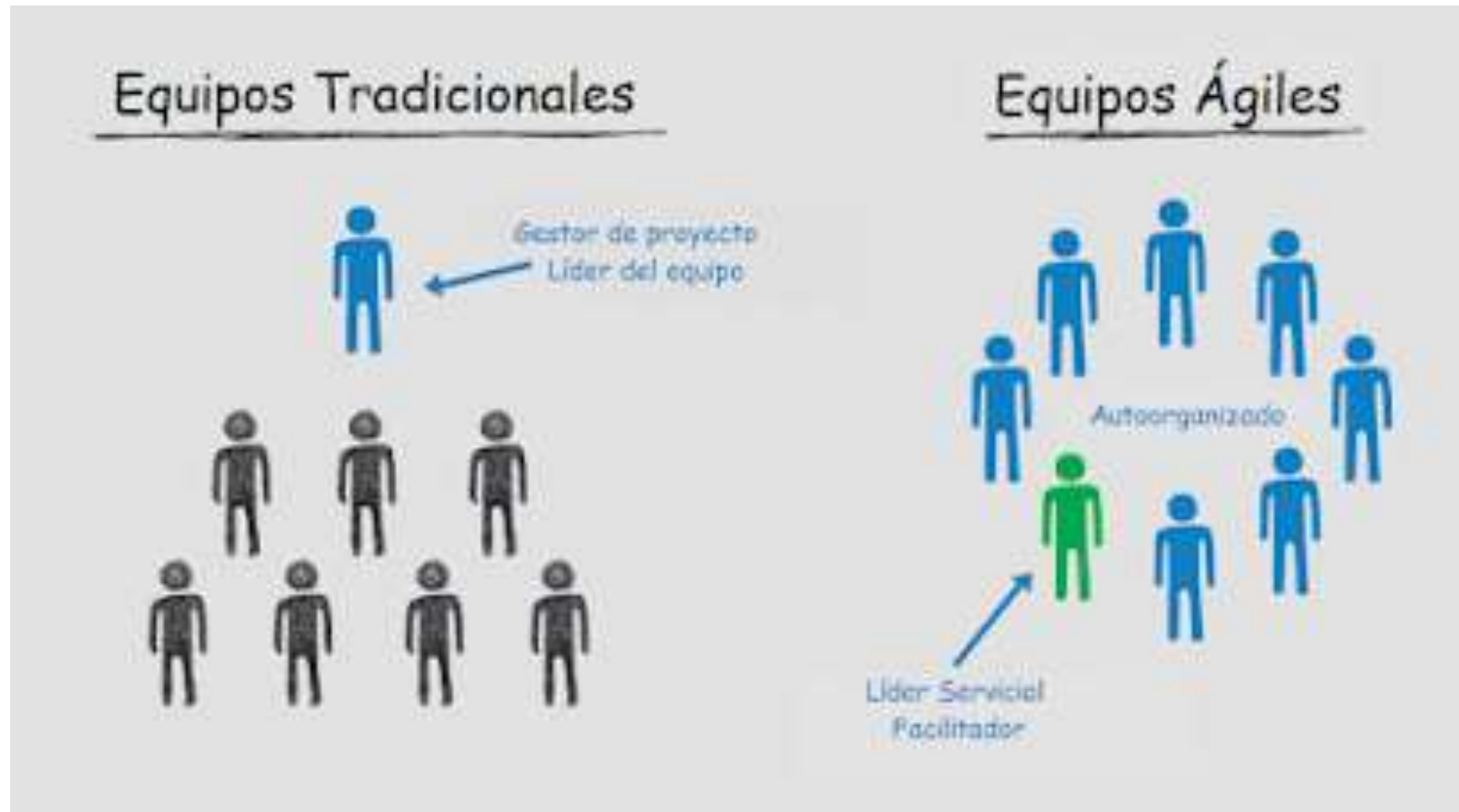
# El equipo de software ágil

---

Muchas organizaciones de software defienden el desarrollo ágil de software, lo que conlleva a crear equipos pequeños y muy motivados, lo que se denomina equipo ágil.

- Se hace hincapié en la competencia individual y colaboración grupal.
- Son autoorganizados.

# El equipo de software ágil



Se parece a un  
paradigma  
aleatorio con  
pocos miembros