CICLO DE VIDA DEL SOFTWARE

1. Concepto de Ciclo de Vida
2. Procesos del Ciclo de Vida del Software
3. Modelo en cascada
4. Modelo incremental
5. Modelo en espiral
6. Prototipado
7. La reutilización en el Ciclo de Vida
8. Síntesis automática de Software
9. Comparación de Ciclos de Vida
10. Modelos para desarrollo de sistemas Orientados a Objetos.

CONCEPTO DE CICLO DE VIDA

"Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software"

IEEE 1074

"Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso"

ISO 12207-1

CICLO DE VIDA DEL SOFTWARE

3.020

PROCESOS DEL CICLO DE VIDA SOFTWARE

PROCESOS PRINCIPALES I

- **Proceso de Adquisición**
- **Proceso de Suministro**

PROCESOS PRINCIPALES II

Proceso de Desarrollo I

- Análisis de Requisitos del Sistema
- Diseño de la Arquitectura del Sistema
- Análisis de los Requisitos del Software
- Diseño de la Arquitectura del Software
- Diseño Detallado del Software
- Codificación y Prueba del Software

PROCESOS PRINCIPALES III

Proceso de Desarrollo II

- Integración del Software
- Prueba del Software
- Integración del Sistema
- Prueba del Sistema
- Instalación del Software
- Soporte del proceso de Aceptación del Software

PROCESOS PRINCIPALES IV

- **Proceso de Explotación**
- **Proceso de Mantenimiento**

PROCESOS DE SOPORTE I

- **Proceso de Documentación**
- **Proceso de Gestión de la Configuración**

PROCESOS DE SOPORTE II

- **Proceso de Aseguramiento de la Calidad**
- **Proceso de Verificación**
- **Proceso de Validación**

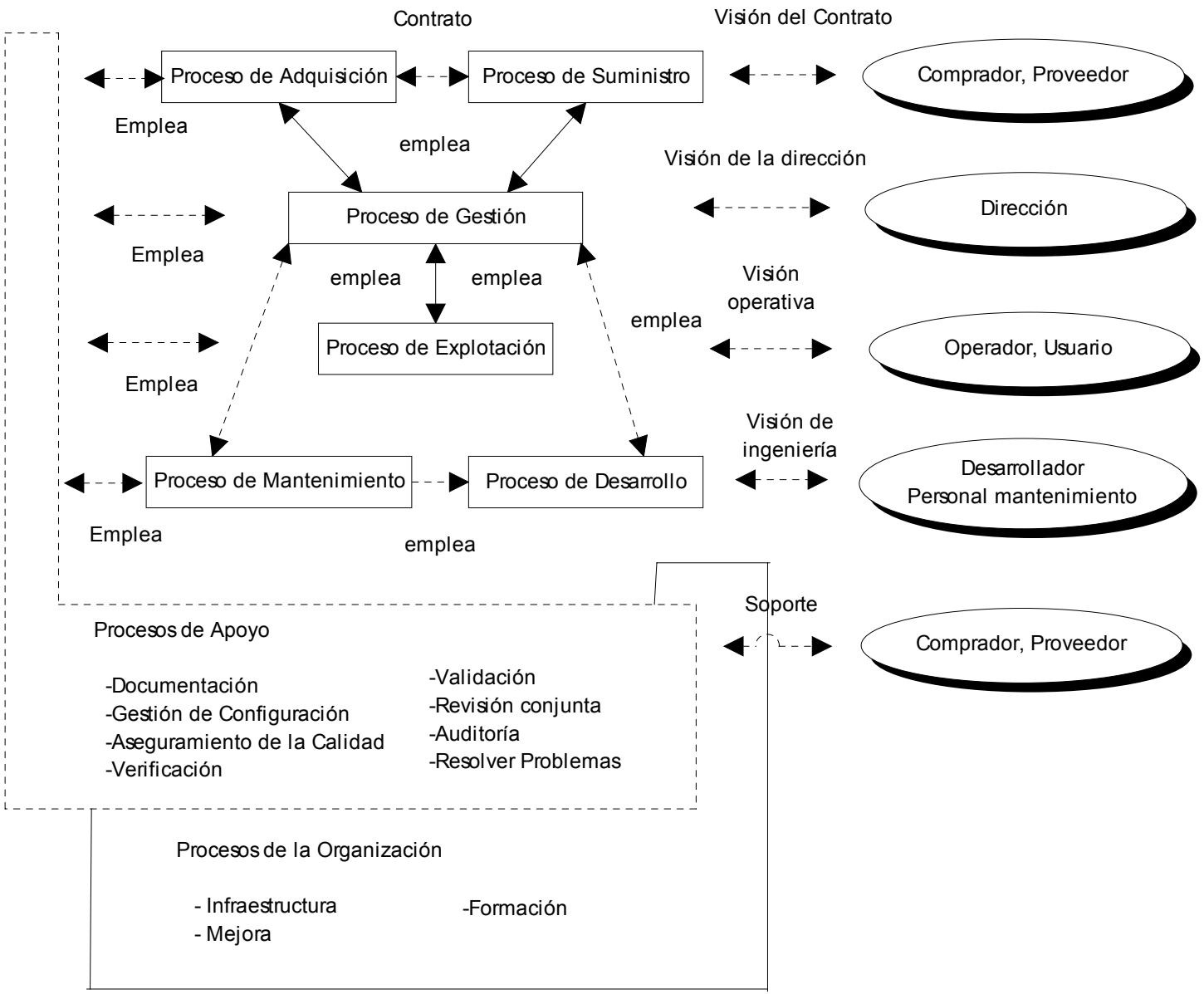
PROCESOS DE SOPORTE III

- **Proceso de Revisión Conjunta**
- **Proceso de Auditoría**
- **Proceso de Resolución de Problemas**

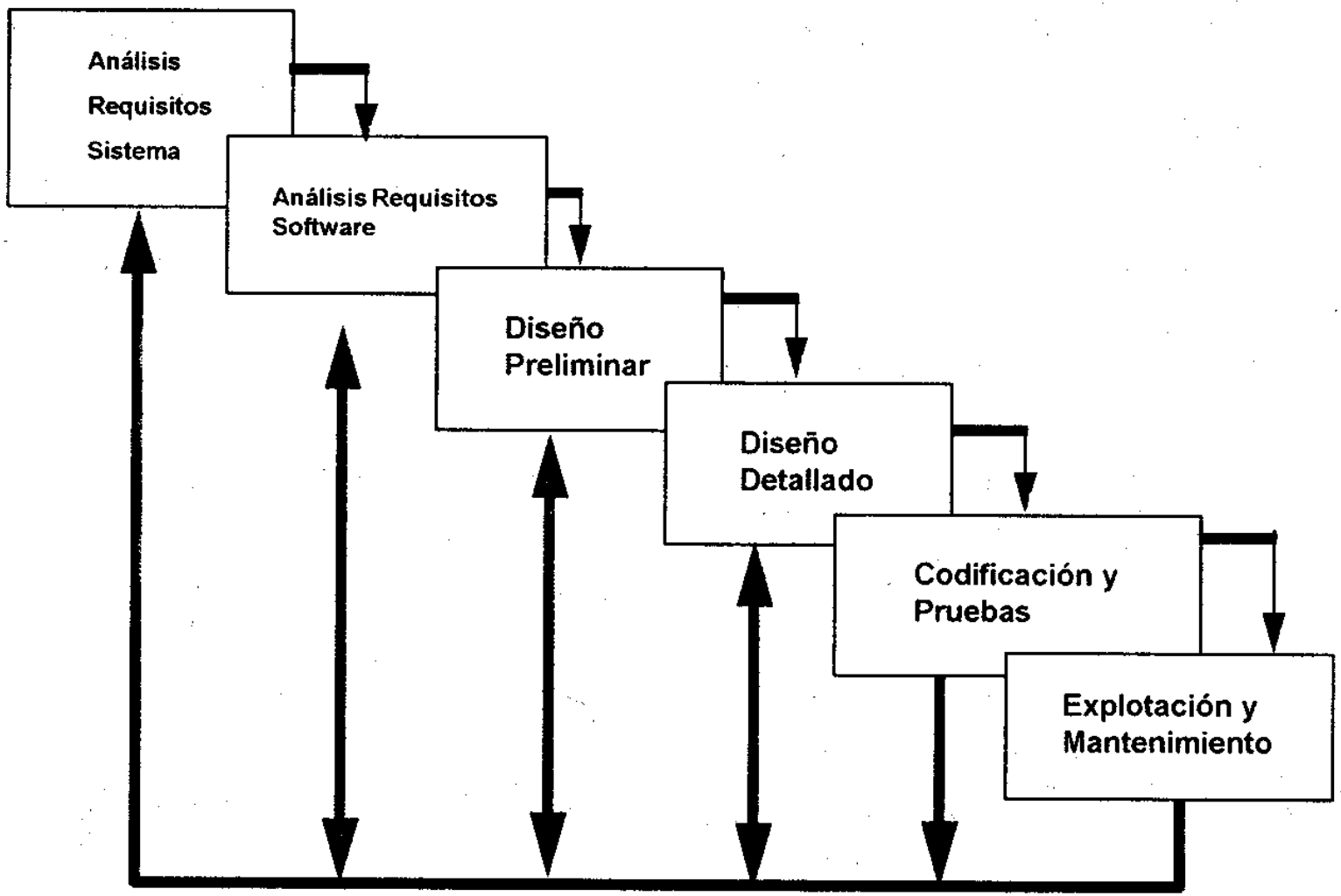
PROCESOS GENERALES

- **Proceso de Gestión**
- **Proceso de Infraestructura**
- **Proceso de Mejora**
- **Proceso de Formación**

CICLO DE VIDA DEL SOFTWARE



MODELO EN CASCADA

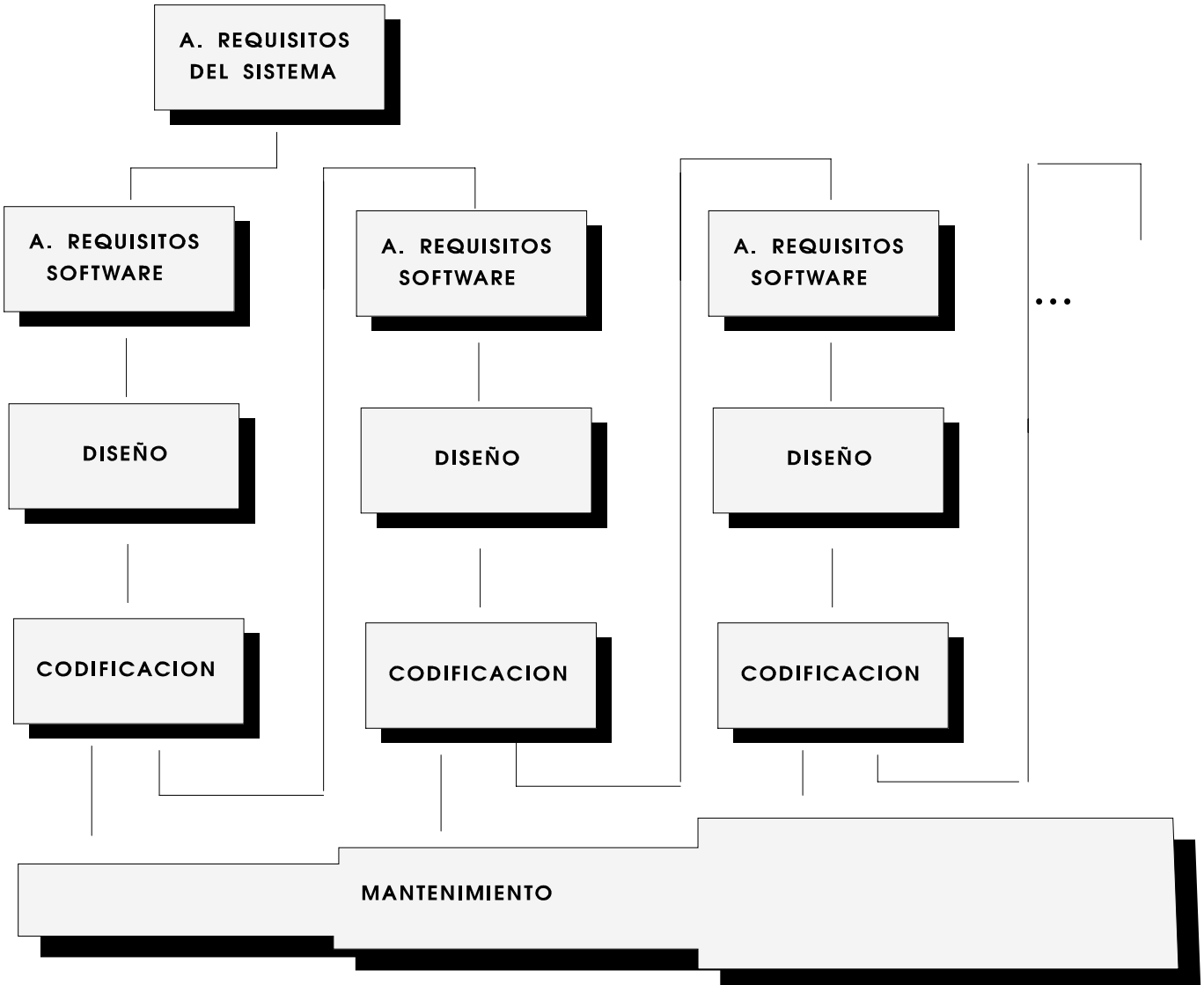


MODELO EN CASCADA

CRITICAS:

☹ No refleja realmente el proceso de desarrollo del software
☹ Se tarda mucho tiempo en pasar por todo el ciclo
☹ Perpetua el fracaso de la industria del software en su comunicación con el usuario final
☹ El mantenimiento se realiza en el código fuente
☹ Las revisiones de proyectos de gran complejidad son muy difíciles
☹ Impone una estructura de gestión de proyectos

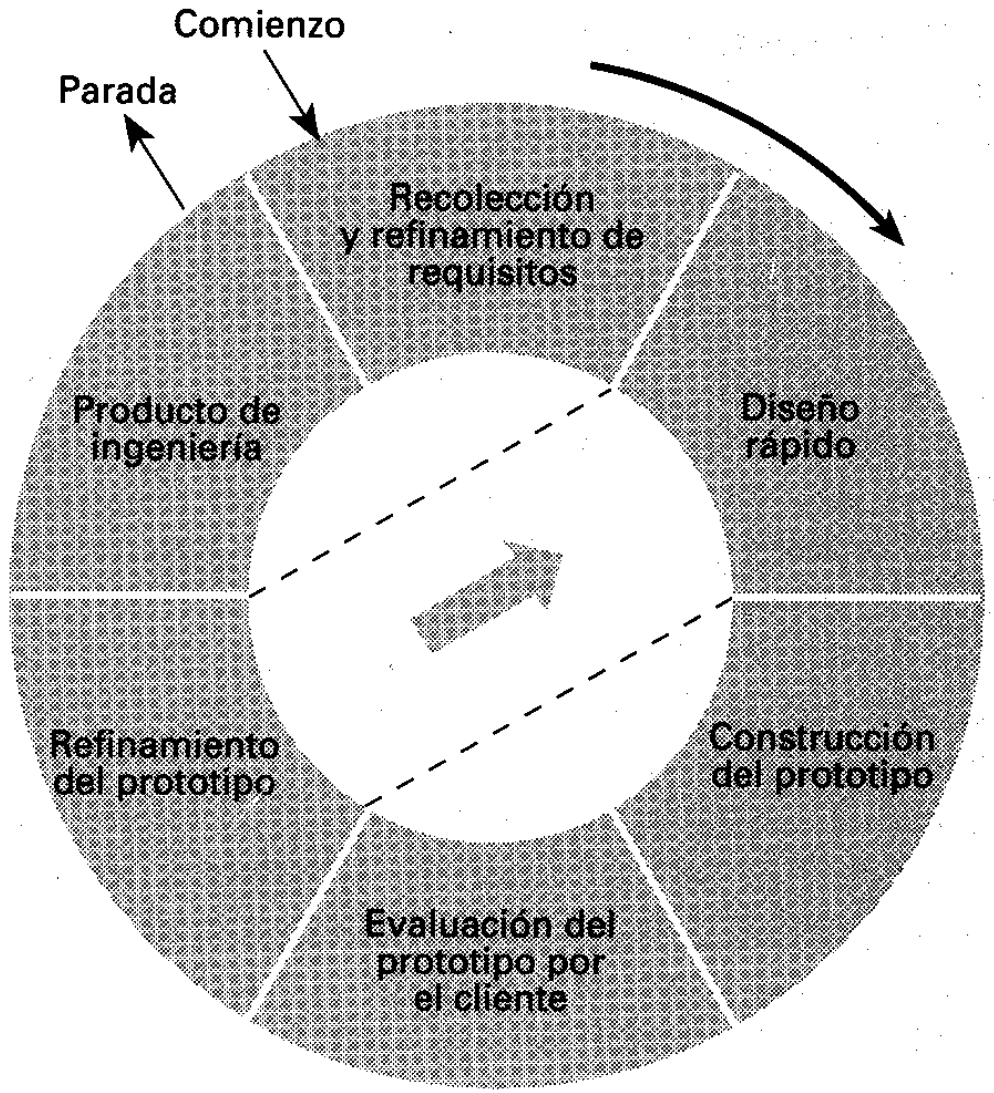
MODELO INCREMENTAL



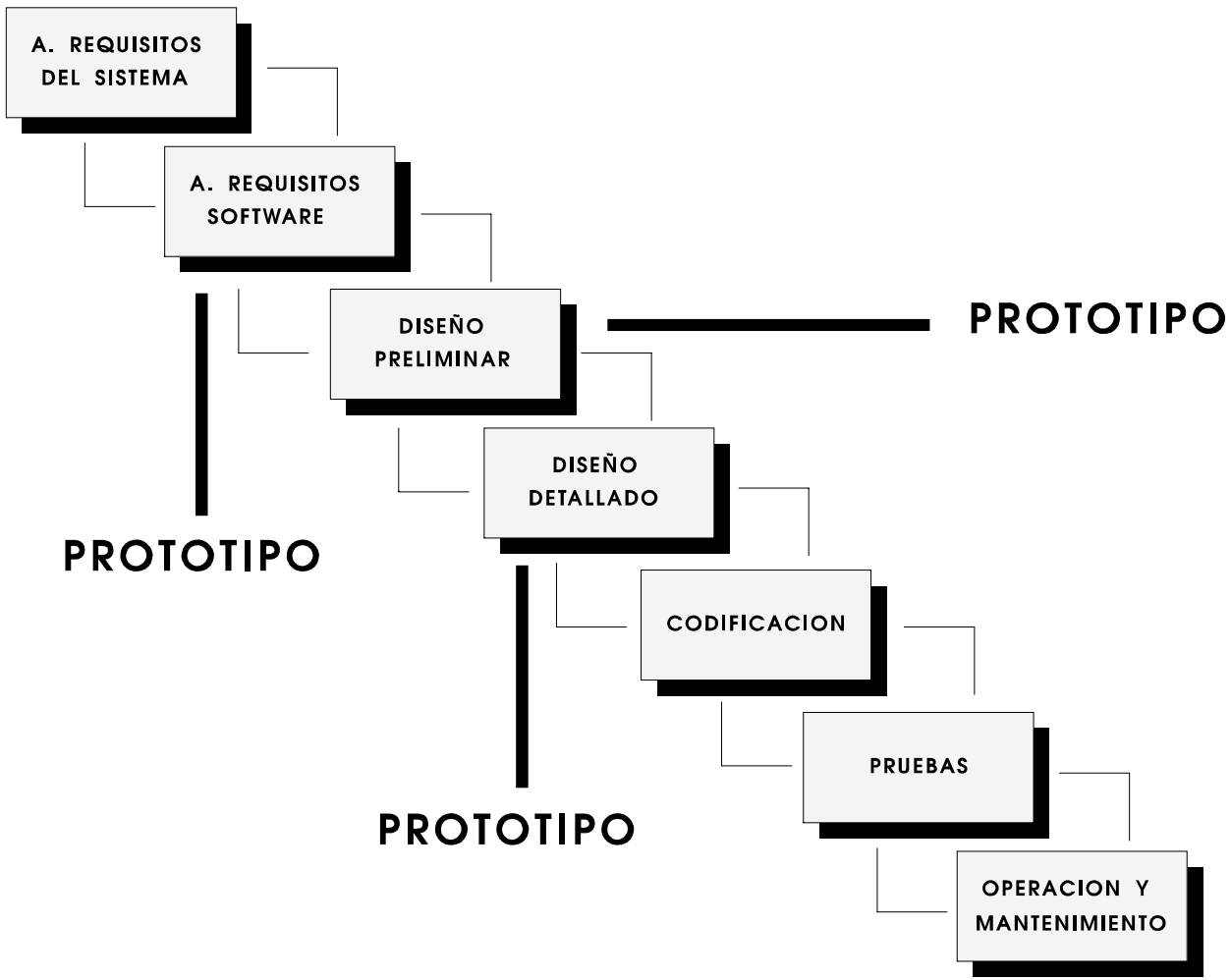
MODELO INCREMENTAL

- Se evitan proyectos largos y se entrega "*Algo de valor*" a los usuarios con cierta frecuencia
- El usuario se involucra más
- Difícil de evaluar el coste total
- Difícil de aplicar a sistemas transaccionales que tienden a ser integrados y a operar como un todo
- Requiere gestores experimentados
- Los errores en los requisitos se detectan tarde.
- El resultado puede ser muy positivo

MODELO DE PROTOTIPO



EL PROTOTIPADO "RAPIDO"



MODELO DE PROTOTIPO

- No modifica el flujo del ciclo de vida
- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios
- Reduce costos y aumenta la probabilidad de éxito
- Exige disponer de las herramientas adecuadas
- No presenta calidad ni robustez
- Una vez identificados todos los requisitos mediante el prototipo, se construye el producto de ingeniería.

EL PROTOTIPADO

PARA QUE SEA EFECTIVO:

- Debe ser un sistema con el que se pueda experimentar
- Debe ser comparativamente barato (< 10%)
- Debe desarrollarse rápidamente
- Enfasis en la interfaz de usuario
- Equipo de desarrollo reducido
- Herramientas y lenguajes adecuados

"El prototipado es un medio excelente para recoger el 'feedback' (realimentación) del usuario final"

PELIGROS DEL PROTOTIPO

- El cliente ve funcionando lo que para el es la primera versión del prototipo que ha sido construido con “plastilina y alambres”, y puede desilusionarse al decirle que el sistema aun no ha sido construido.
- El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y de mantenimiento que tiene con el cliente.

EL PROTOTIPADO EVOLUTIVO

- Construcción de una implementación parcial que cubre los requisitos conocidos, para ir aprendiendo el resto y, paulatinamente, incorporarlos al sistema

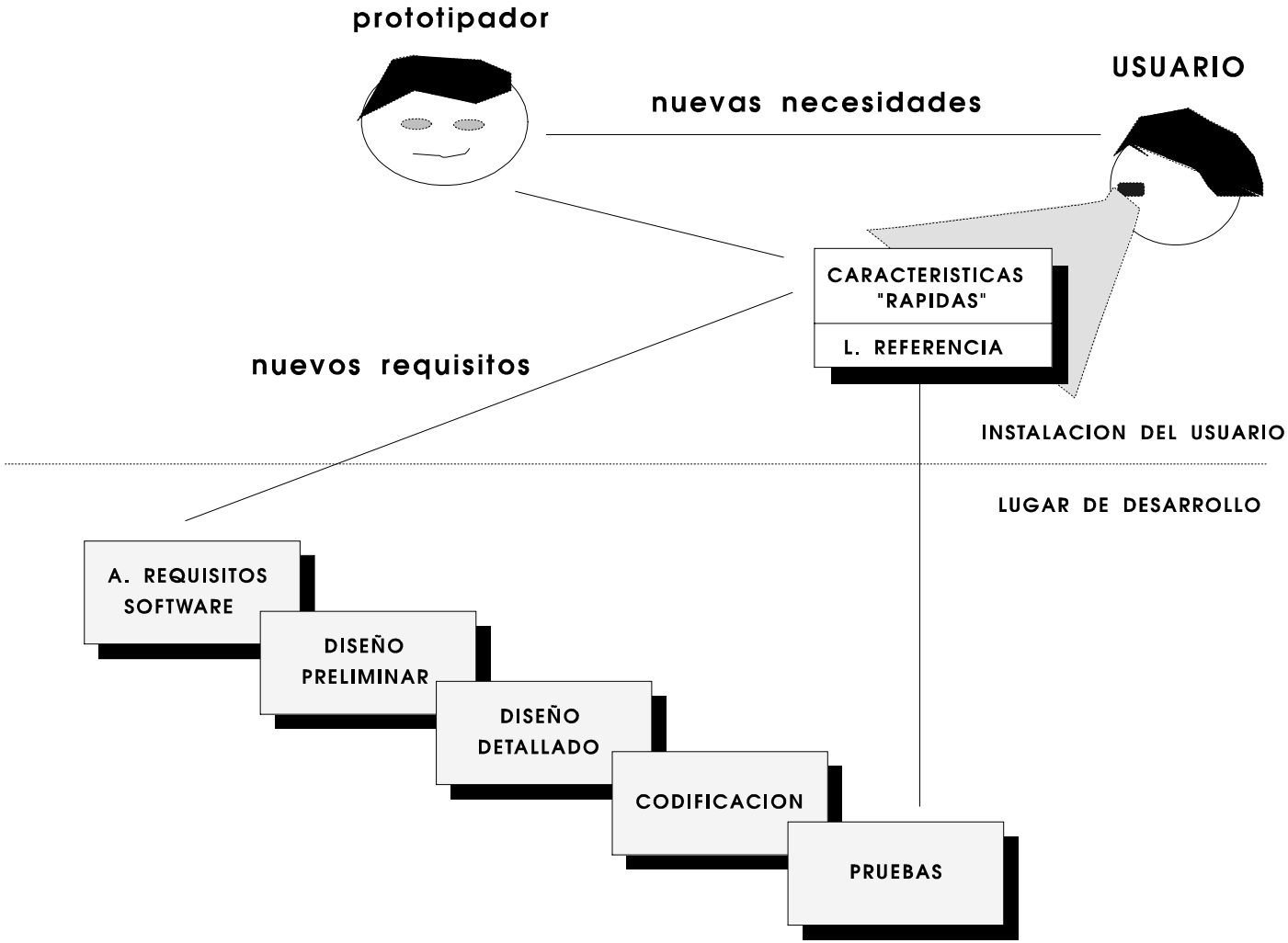
- Reduce el riesgo y aumenta la probabilidad de éxito

- No se conocen niveles apropiados de calidad y documentación

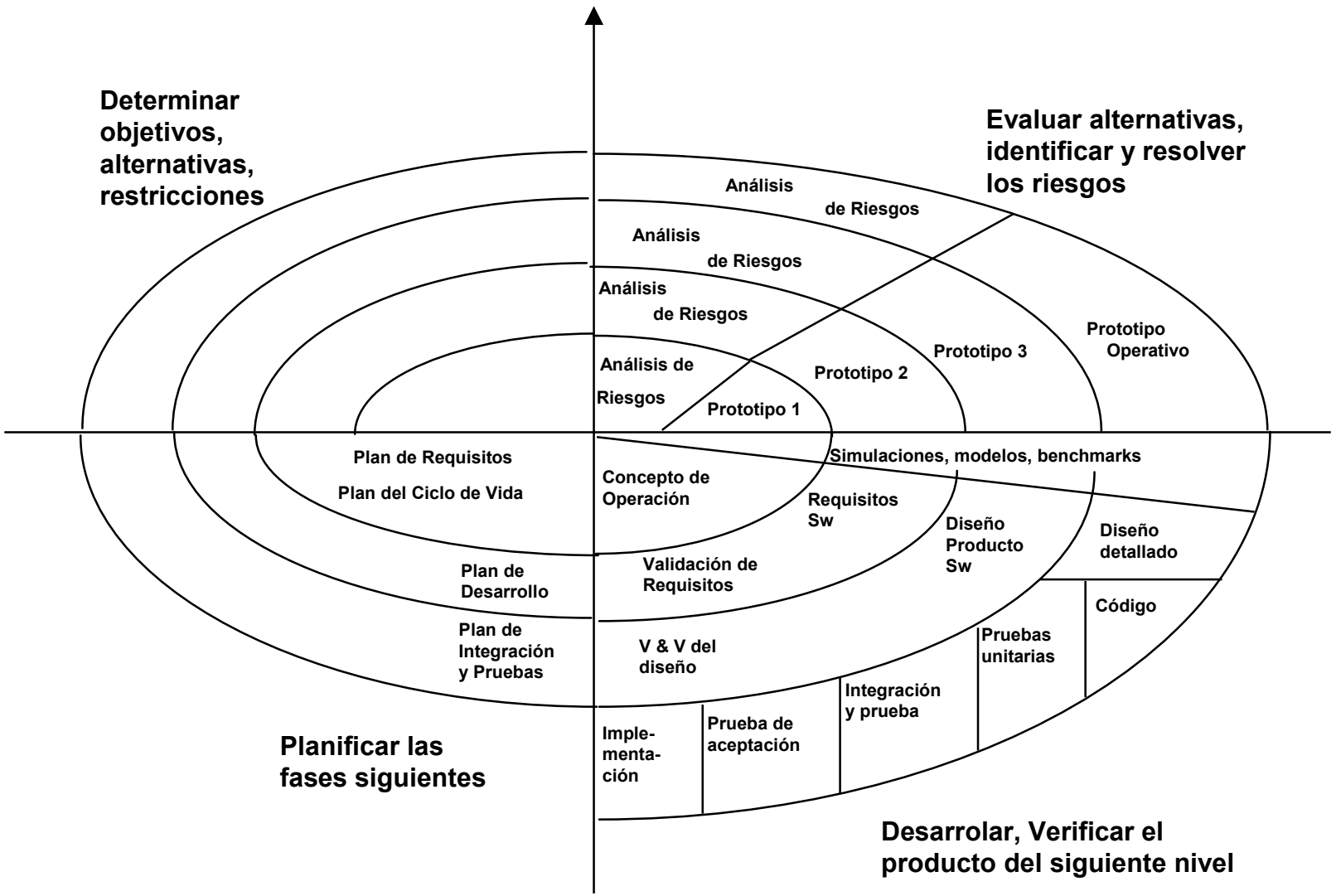
- Problemas de gestión de configuración

Construir software para que pueda ser modificado fácilmente es un "arte desconocido"

EL PROTOTIPADO OPERACIONAL



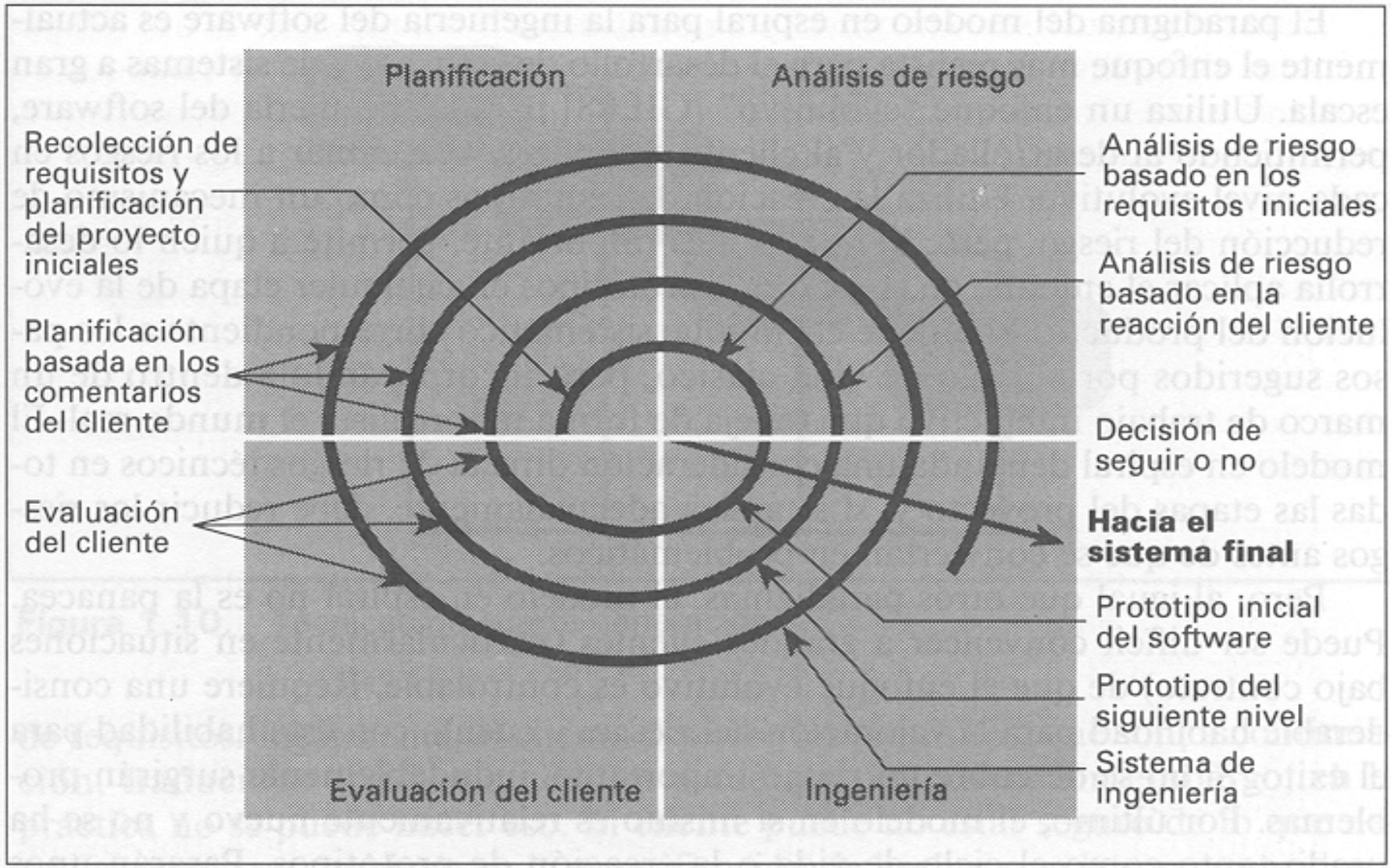
MODELO EN ESPIRAL



CICLO DE VIDA DEL SOFTWARE

3.240

MODELO EN ESPIRAL



MODELO EN ESPIRAL

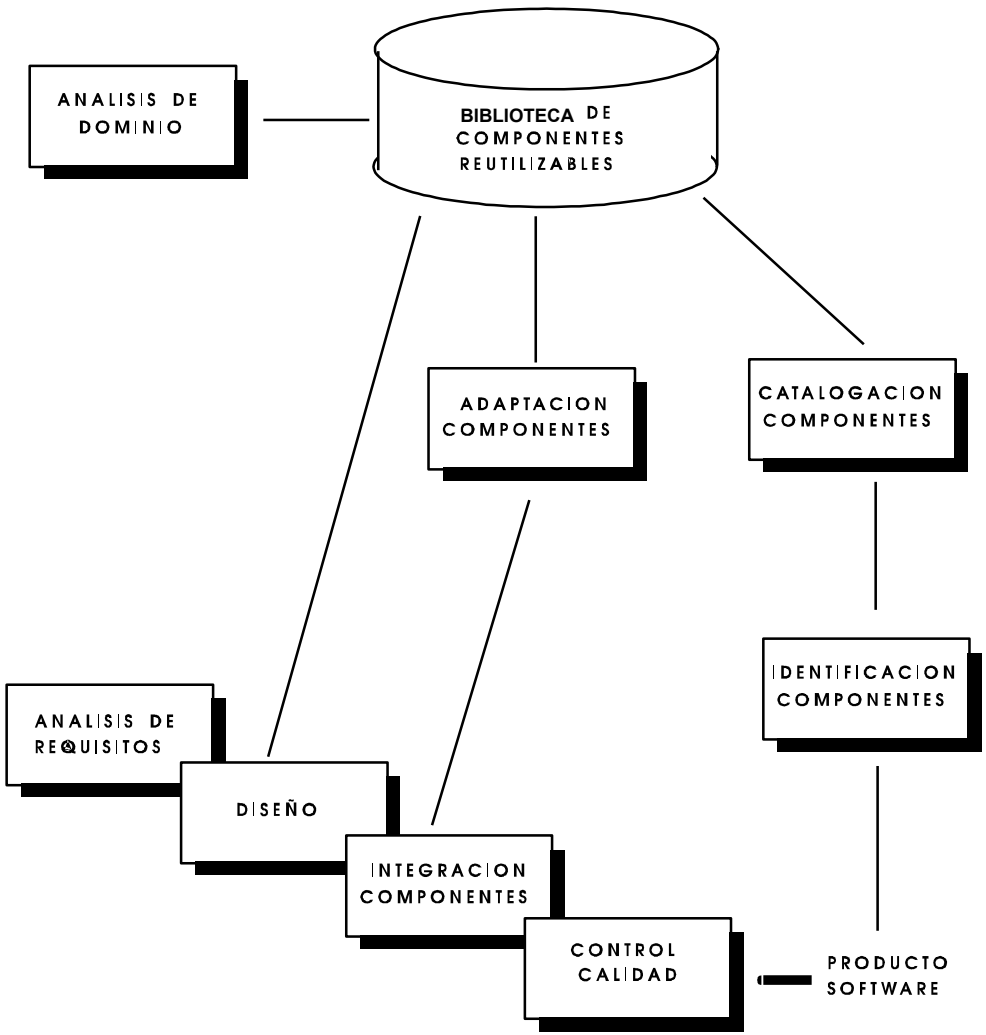
- Trata de mejorar los ciclos de vida clásicos y prototipos.
- Permite acomodar otros modelos
- Incorpora objetivos de calidad y gestión de riesgos
- Elimina errores y alternativas no atractivas al comienzo
- Permite iteraciones, vuelta atrás y finalizaciones rápidas
- Cada ciclo empieza identificando:
  - Los objetivos de la porción correspondiente
  - Las alternativas
  - Restricciones
- Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente

MODELO EN ESPIRAL

Diferencias entre modelo en espiral y modelos tradicionales

- Reconocimiento explícito de las diferentes alternativas.
- Identificación de riesgos para cada alternativa desde el comienzo.
- Al dividir el proyecto en ciclos, al final de cada uno existe un acuerdo para los cambios que hay que realizar en el sistema.
- El modelo se adapta a cualquier tipo de actividad adicional

LA REUTILIZACION EN EL CICLO DE VIDA



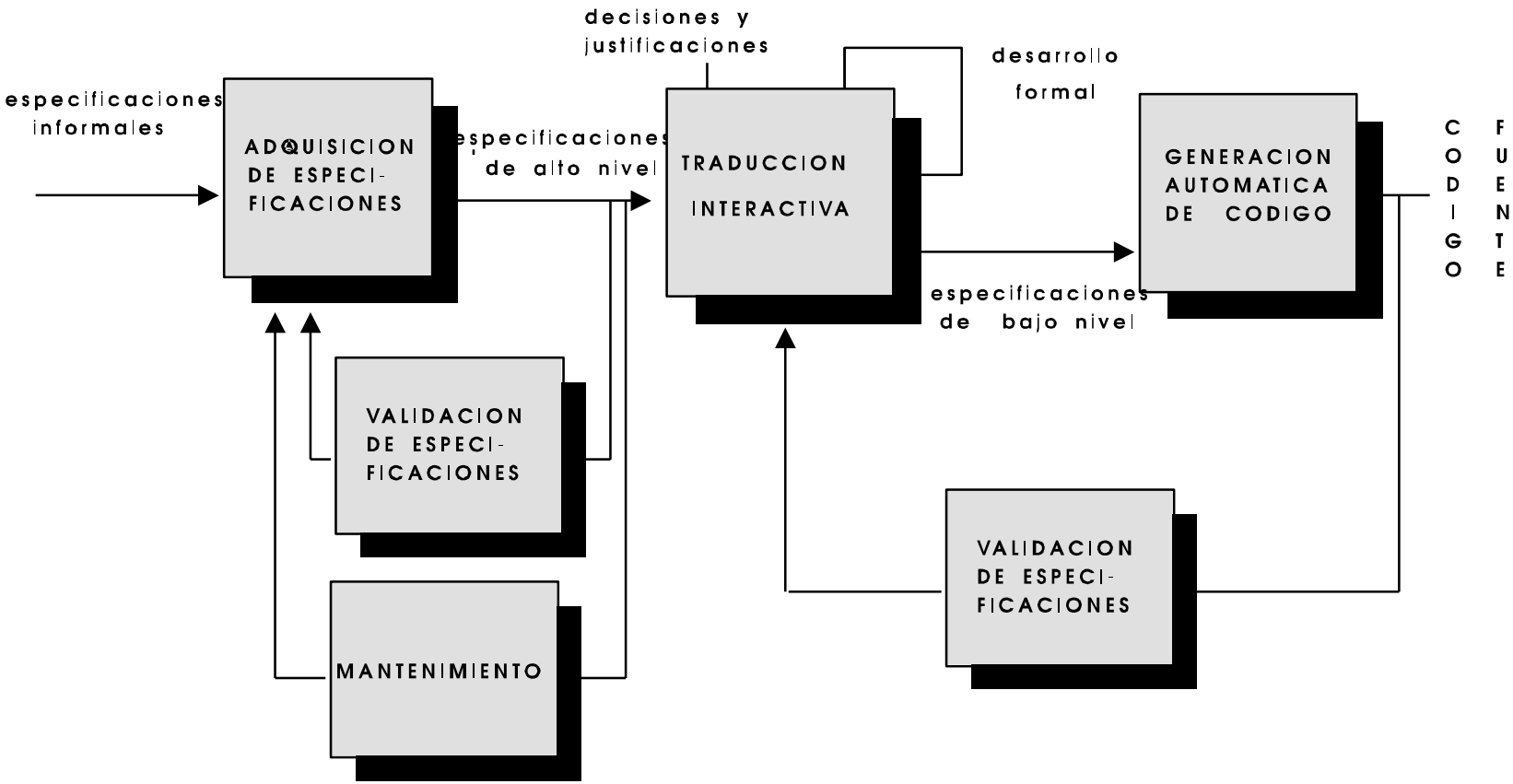
LA REUTILIZACION EN EL CICLO DE VIDA

😐 Principios de la reutilización:

→ Existen similitudes entre distintos sistemas de un mismo dominio de aplicación
→ El software puede representarse como una combinación de módulos
→ Diseñar aplicaciones = especificar módulos + interrelaciones
→ Los sistemas nuevos se pueden caracterizar por diferencias respecto a los antiguos

☺ Reduce tiempos y costes de desarrollo
☺ Aumenta la fiabilidad
☹ Dificultad para reconocer los componentes potencialmente reutilizables
☹ Dificultad de catalogación y recuperación
☹ Problemas de motivación
☹ Problemas de gestión de configuración

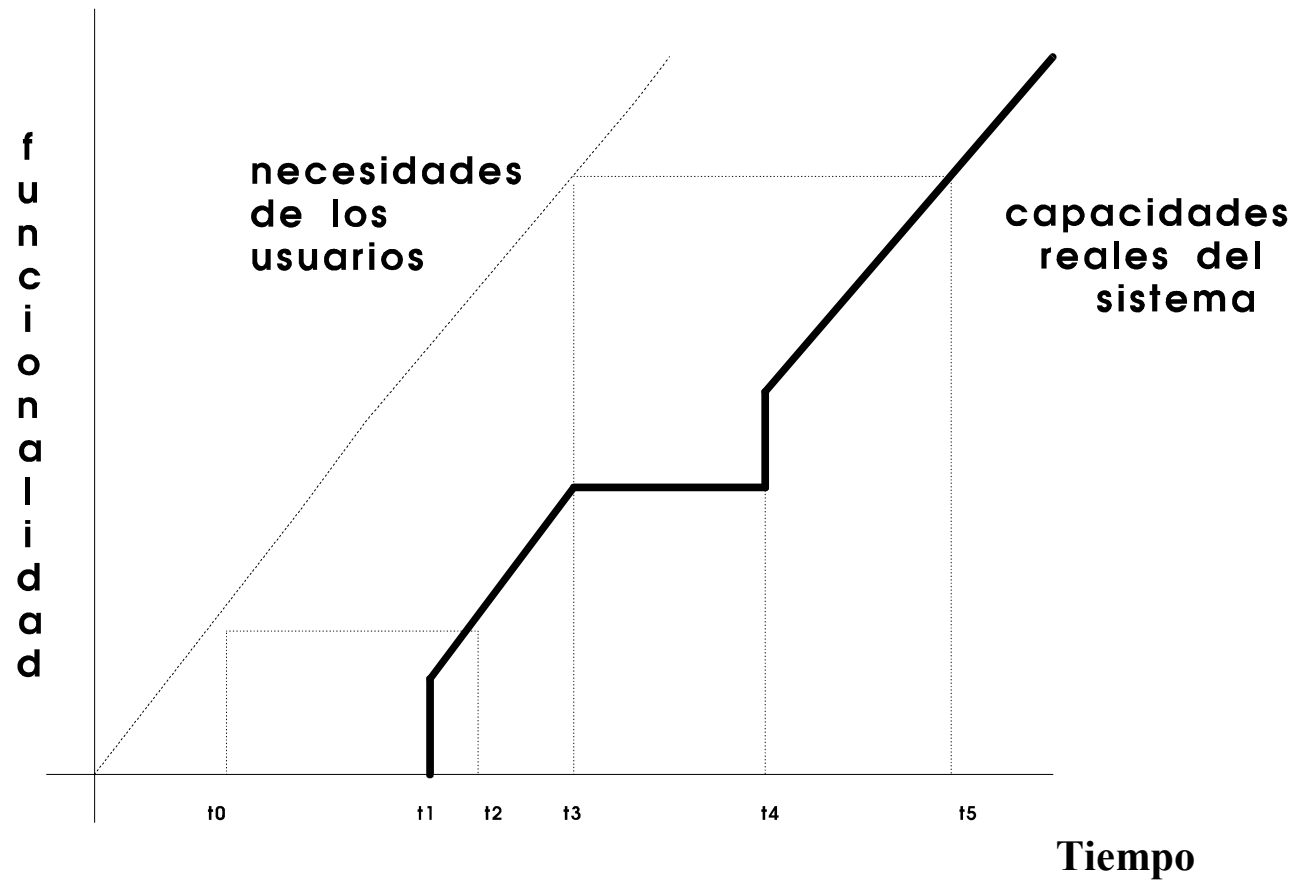
SÍNTESIS AUTOMÁTICA DE SOFTWARE



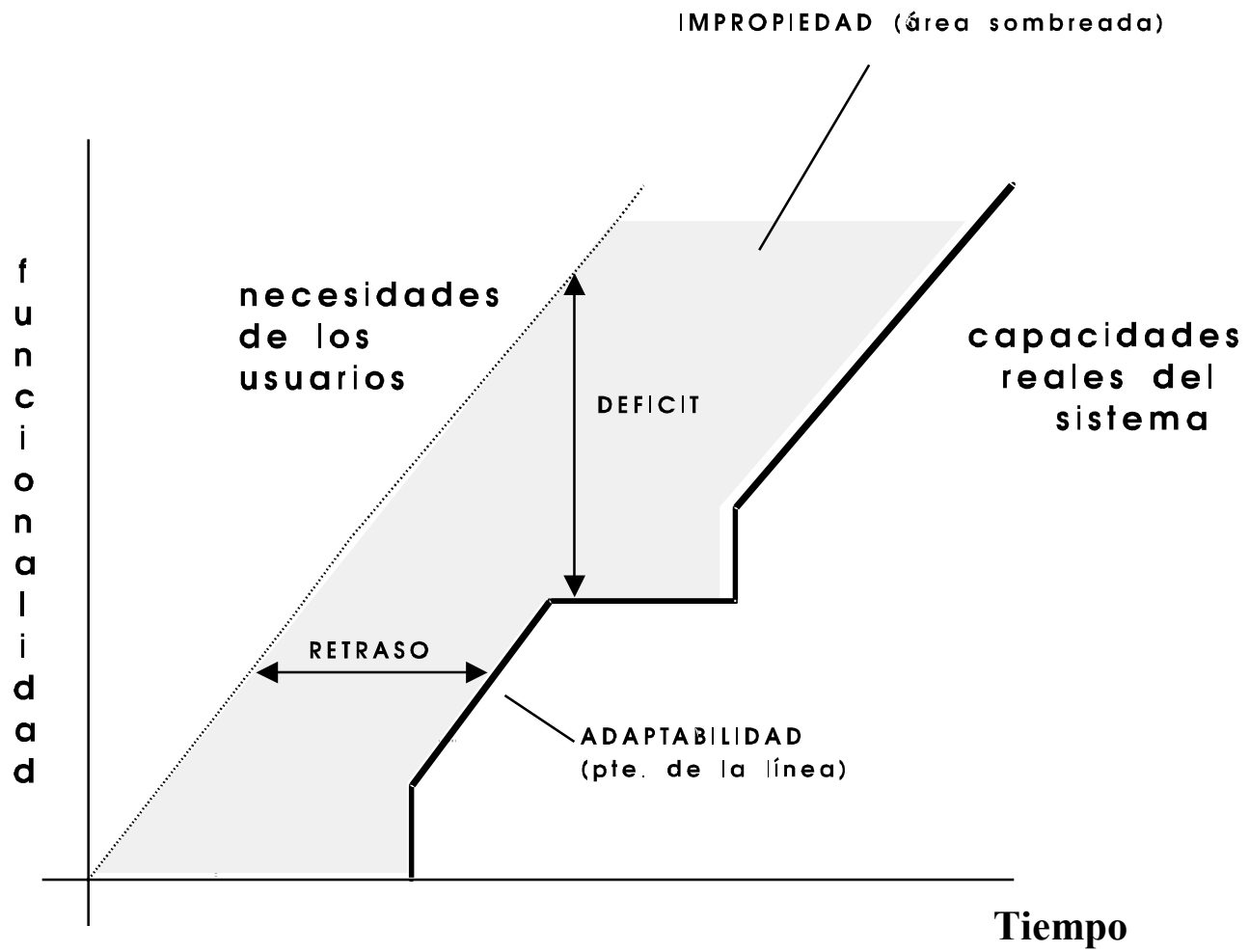
SINTESIS AUTOMATICA DEL SOFTWARE

- Se define el sistema utilizando un lenguaje formal
- La implementación es automática, asistida por el ordenador
- La documentación se genera de forma automática
- El mantenimiento se realiza “por sustitución” no mediante “parches”
- Dificultad en la participación del usuario
- Diseños poco optimizados

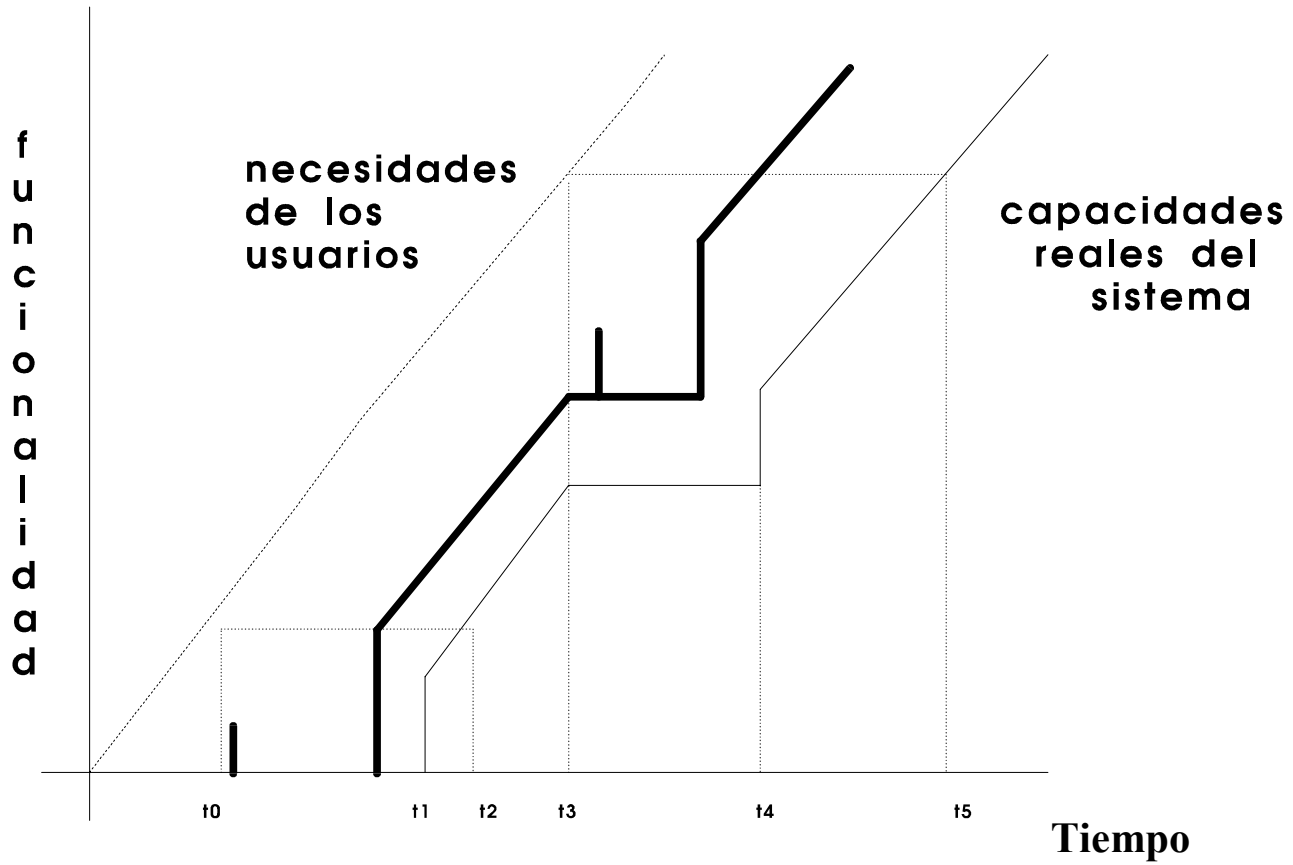
COMPARACION DE CICLOS DE VIDA (Clásico)



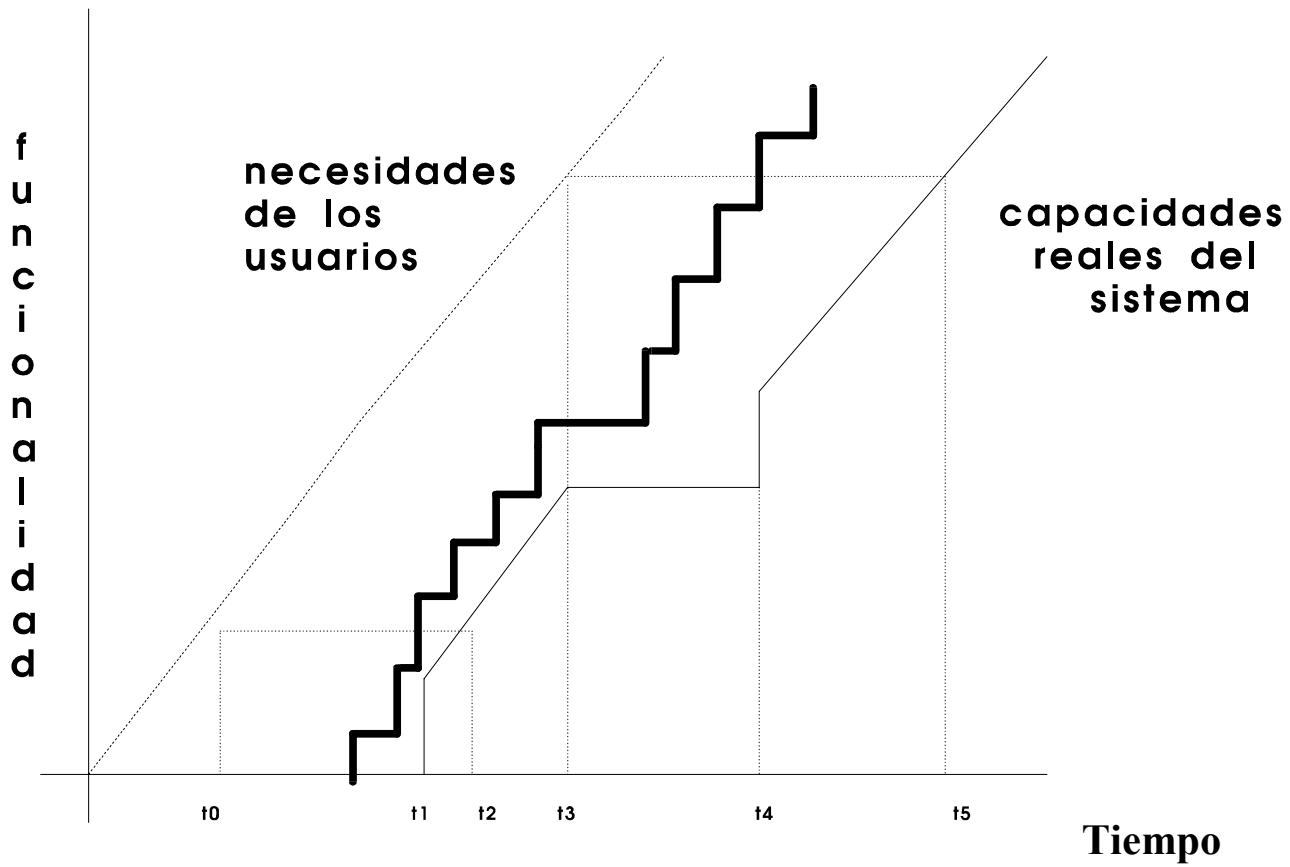
COMPARACION DE CICLOS DE VIDA (Clásico)



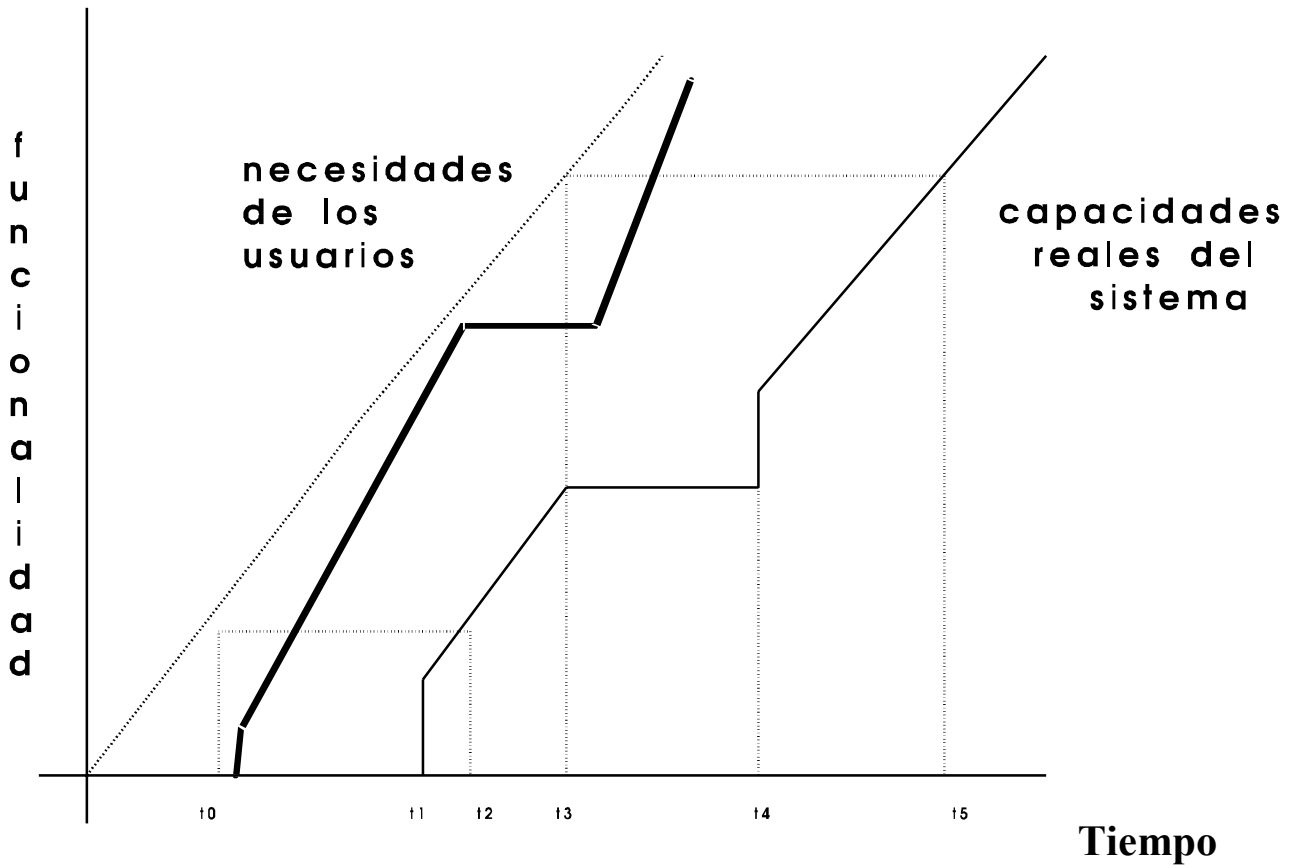
COMPARACION DE CICLOS DE VIDA (Prototipo rápido)



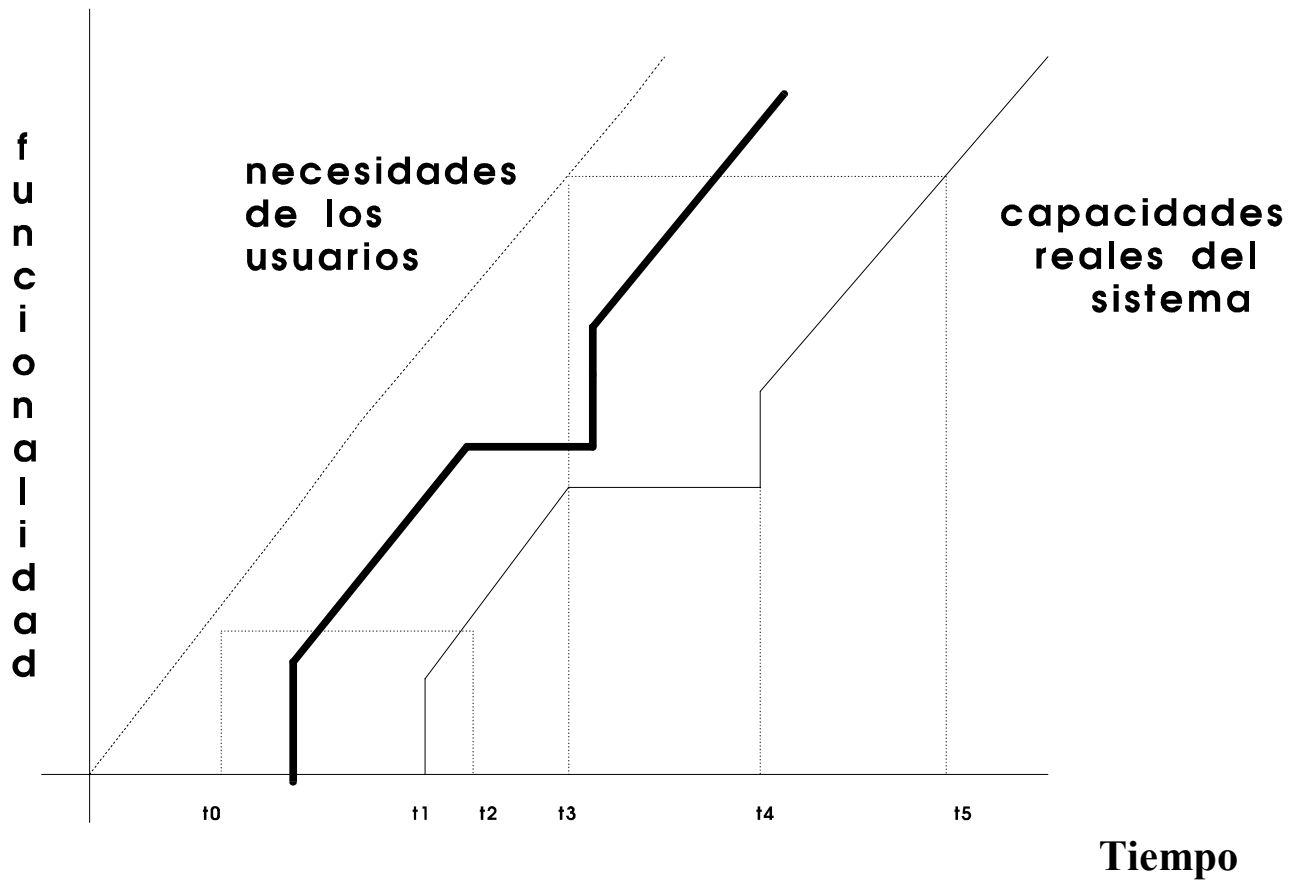
COMPARACION DE CICLOS DE VIDA (Incremental)



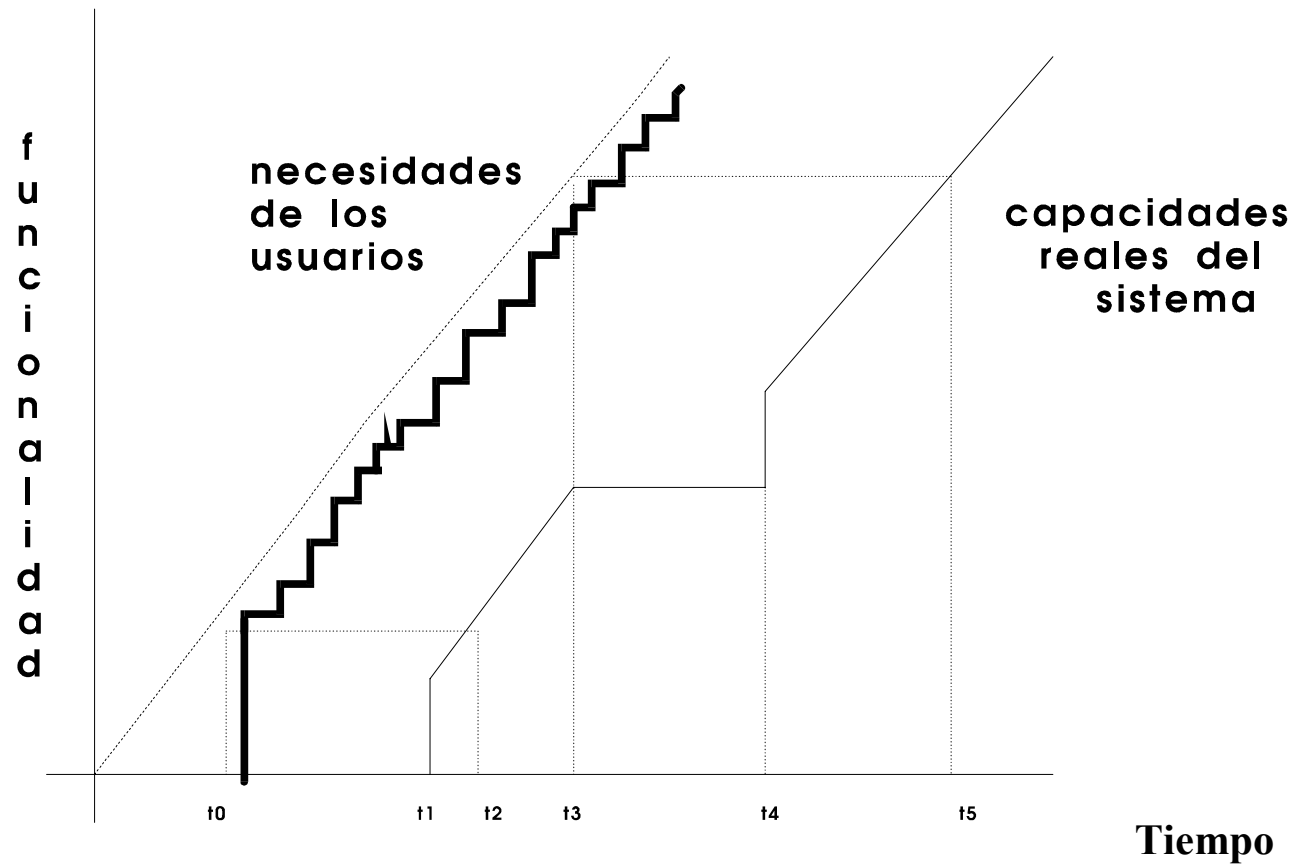
COMPARACION DE CICLOS DE VIDA
(Prototipado evolutivo)



COMPARACION DE CICLOS DE VIDA (Reutilización)

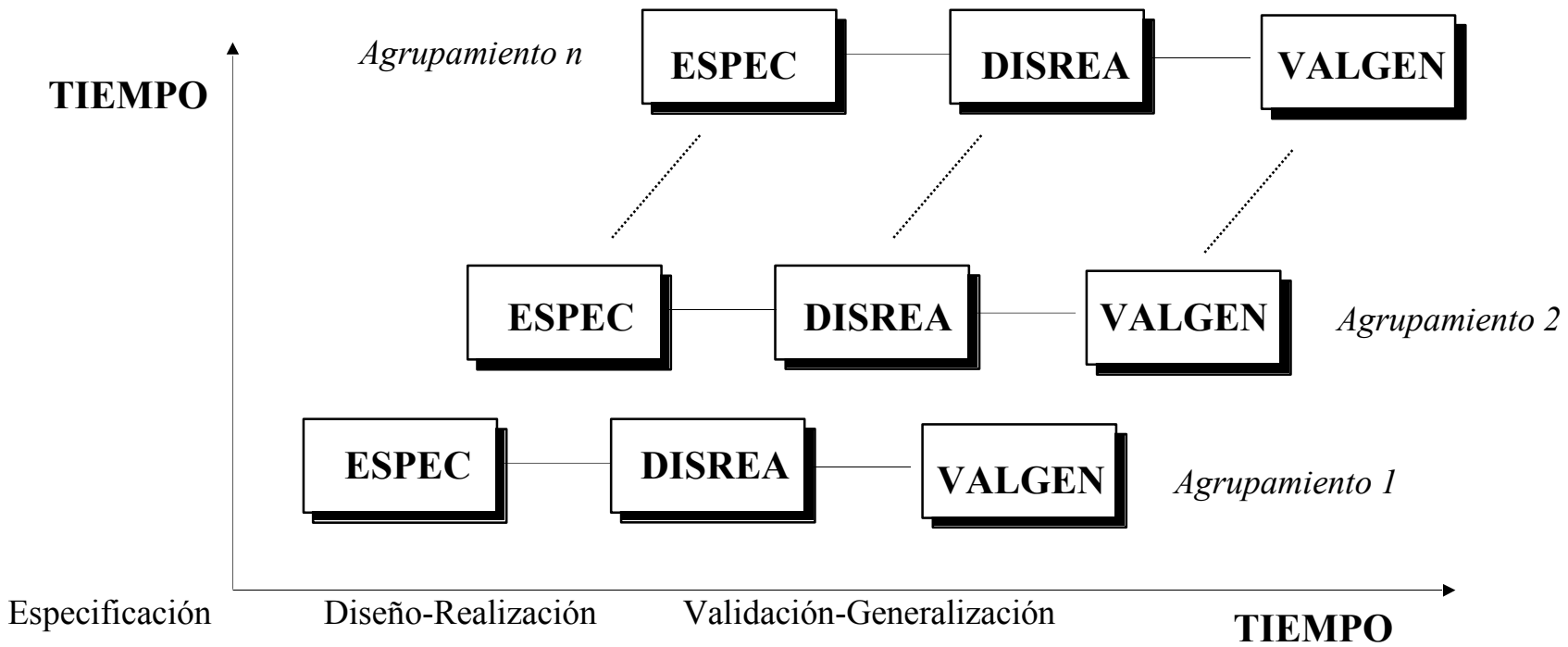


COMPARACION DE CICLOS DE VIDA (Síntesis automática)



MODELOS PARA DESARROLLO DE SISTEMAS ORIENTADOS A OBJETOS

MODELO DE AGRUPAMIENTO



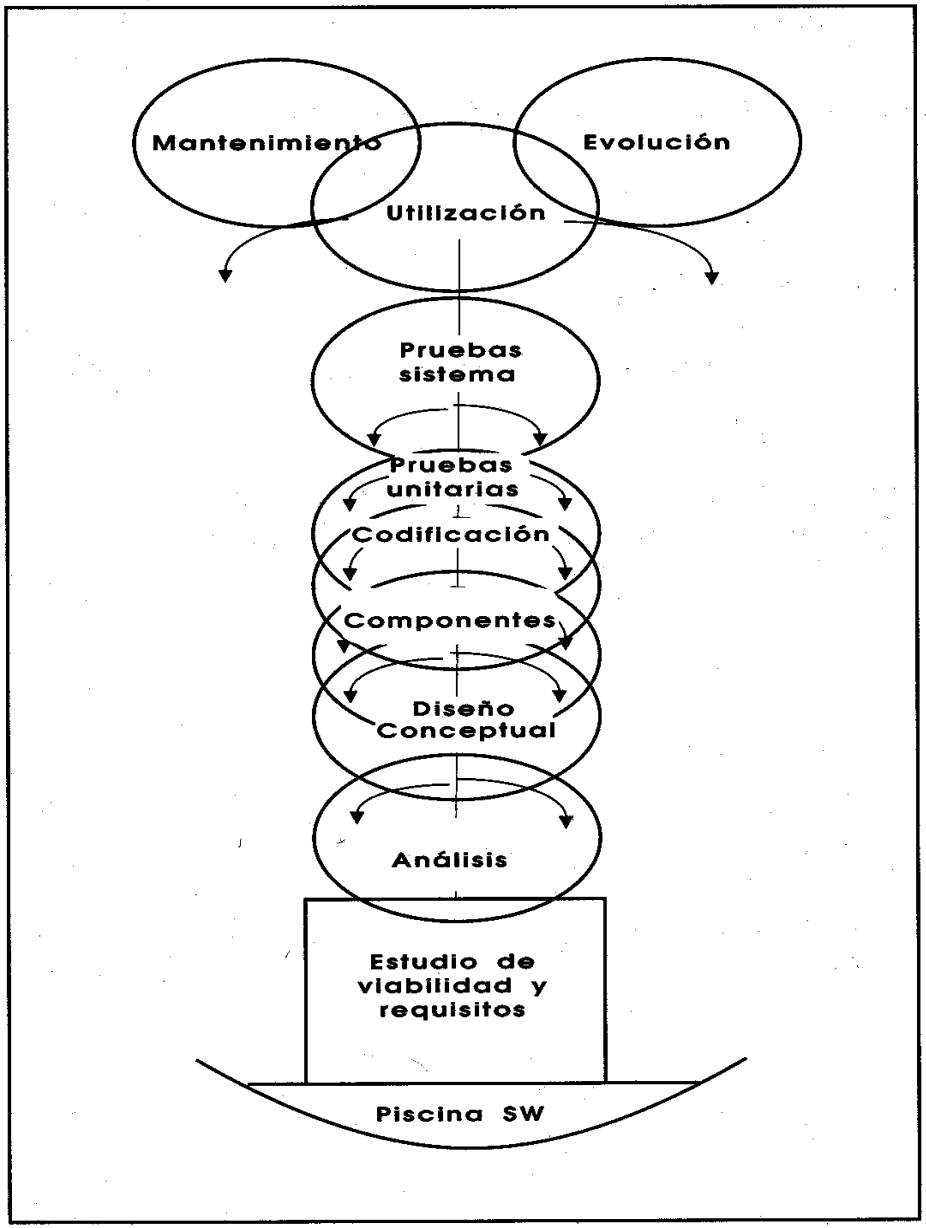
MODELO REMOLINO

- Amplitud
- Profundidad
- Madurez
- Alternativas
- Alcance

MODELO PINBALL

- La *pelota* representa un proyecto completo o un subproyecto.
- El *jugador* es el equipo de desarrollo.
- Se procede de forma iterativa a encontrar clases, atributos métodos e interrelaciones y definir colaboraciones, herencia, agregación y subsistemas.
- Por último se pasa a la programación, prueba e implementación.
- Hay dos estilos a la hora de "jugar":
  - Seguro → tecnologías y métodos probados.
  - Al límite → Mayor riesgo, más ventajas.

MODELOS OO: FUENTE



CONSIDERACIONES SOBRE MODELOS OO

- Se eliminan fronteras entre fases debido a la naturaleza iterativa del desarrollo orientado al objeto.
- Aparece una nueva forma de concebir los lenguajes de programación y su uso al incorporarse bibliotecas de clases y otros componentes reutilizables.
- Hay un alto grado de iteración y solapamiento, lo que lleva a una forma de trabajo muy dinámica.

EJERCICIOS

Ejercicio 1

¿Qué factores influyen a la hora de elegir un ciclo de vida para resolver un problema dado?

¿Qué ciclo de vida elegiría para resolver un problema que se comprende bien desde el principio y está muy estructurado? Una vez elegido el ciclo de vida, ¿qué procesos escogería para dicho ciclo de vida, teniendo en cuenta que el desarrollo informático para resolver el problema anterior lo realiza una única persona?

EJERCICIOS

Ejercicio 2

Se supone que se va desarrollar una aplicación relativa a la gestión de pedidos de una empresa. En este caso el cliente no tiene todavía muy claro qué es lo que quiere. Además, el personal informático va a utilizar un tecnología que le resulta completamente nueva. Discútase qué tipo de ciclo de vida es más apropiado y qué procesos se deberían utilizar para desarrollar esta aplicación.

EJERCICIOS

Ejercicio 3

Indicar la(s) respuesta(s) correcta(s) y razonar la respuesta:

El ciclo de vida:

a)Comienza con una idea o necesidad que satisfacer y acaba con las pruebas satisfactorias del producto.

b)No existe ningún estándar que describa sus procesos y actividades.

c)No se trata sólo de realizar el análisis, diseño, codificación y pruebas; también incluye, entre otros, procesos de soporte.

d)El mantenimiento lo constituyen las actividades para mantener sin cambios el sistema.

e)En la actividad de análisis de los requisitos software los desarrolladores obtienen de los futuros usuarios los requisitos que piden al sistema.