Ingeniería de software I


Herramientas
Métodos
Proceso
Compromiso con la calidad

El proceso de software


Cómo ven los USUARIOS a los INFORMÁTICOS
Cómo ven los INFORMÁTICOS a los USUARIOS
UGH!
OOGH!
UG UG!!
TOC TOC

Un **proceso**

es un conjunto de actividades, acciones y tareas que se ejecutan para crear un producto del trabajo.

Un ***proceso de software*** es una secuencia de actividades que conducen a la elaboración de un producto de software

La **estructura del proceso** establece el **fundamento** para el proceso completo de la ingeniería de software que sean aplicables a **todos los proyectos, sin importar su tamaño o complejidad.**

Secuencia simple de resolución de problemas:

Decidir qué hacer (cuál es el problema)

Decidir cómo hacerlo

Hacerlo

Probar el resultado

Usar el resultado

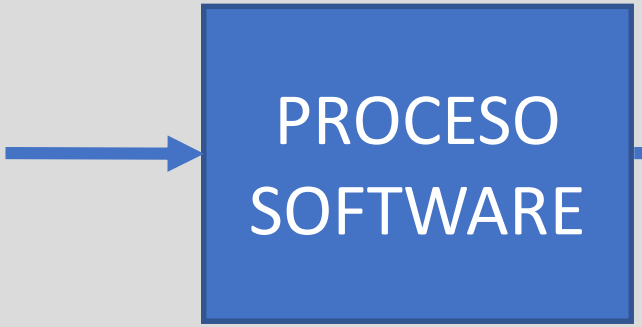
Nivel muy abstracto

El proceso de construcción de software también es una actividad de resolución de problemas

Resolución general de problemas:

- Identificar el problema
- Definir y representar el problema
- Explorar las posibles estrategias
- Aplicar y mejorar las estrategias
- Mirar atrás y evaluar los efectos de la actividad realizada.

Problema o necesidad



Solución o software

VALIDEZ
Problema o necesidad
Dominio de aplicación
Modelo conceptual
Modelo formal
Dominio de implementación
Solución o software
CORRECCIÓN O VERIFICACIÓN

SOLUTION

Proceso según Pressman

La *estructura del proceso* establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de *actividades estructurales*

Las *actividades estructurales son* aplicables a todos los proyectos de software, sin importar su tamaño o complejidad

La estructura del proceso incluye un conjunto de **actividades sombrilla** que son aplicables a través de todo el proceso del software

Una estructura de proceso general para la ingeniería de software consta de cinco actividades:

Comunicación

Planeación

Modelado

Construcción

Despliegue

Los detalles del proceso de software serán distintos en cada caso, pero las actividades estructurales son las mismas.

Las actividades estructurales se aplican en forma iterativa a medida que avanza el proyecto. La **comunicación,** la **planeación,** el **modelado,** la **construcción** y el **despliegue** se ejecutan un cierto número de repeticiones del proyecto. Cada iteración produce un *incremento del software.* Conforme se produce cada incremento, el software se hace más y más completo.

Las actividades estructurales del proceso de ingeniería de software son complementadas por cierto número de *actividades sombrilla*

Seguimiento y control del proyecto de software: evalúa el progreso comparándolo con el plan del proyecto y toma acciones necesarias para apegarse a la programación de actividades.

Administración del riesgo: evalúa los riesgos que puedan afectar el resultado del proyecto o la calidad del producto.

Aseguramiento de la calidad del software: define y ejecuta actividades requeridas para garantizar la calidad del software.

Revisiones técnicas: evalúa los productos del trabajo de la ingeniería de software para descubrir y eliminar errores antes de que se propaguen a la siguiente actividad.

Medición: define y reúne mediciones del proceso, proyecto y producto para ayudar al
equipo a entregar el software que satisfaga las necesidades de los participantes.

Administración de la configuración del software: administra los efectos del cambio a lo largo del proceso del software.

Administración de la reutilización: define criterios para volver a usar el producto del trabajo y establece mecanismos para obtener componentes reutilizables.

Preparación y producción del producto del trabajo: agrupa las actividades requeridas para crear productos del trabajo, tales como modelos, documentos, registros, formatos y listas

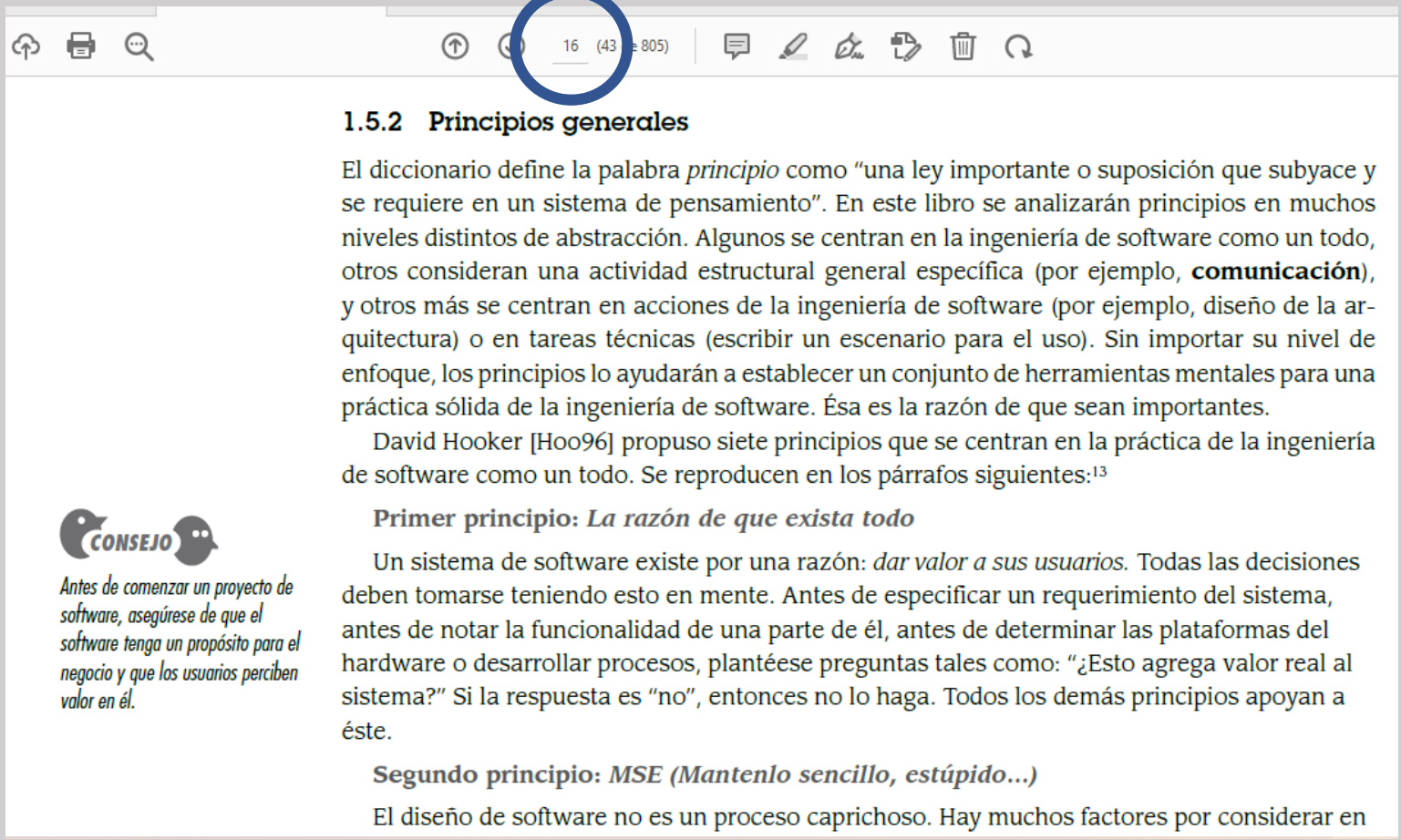
El proceso de ingeniería de software no es una ***prescripción rígida*** que deba seguir en forma dogmática el equipo que lo crea. Debe ser ágil y adaptable (al problema, al proyecto, al equipo y a la cultura organizacional).

Un proceso adoptado para un proyecto puede ser ***significativamente distinto*** de otro adoptado para otro proyecto.

Principios

Principio “una ley importante o suposición que subyace y se requiere en un sistema de pensamiento

Siete principios que se centran en la práctica de la ingeniería de software como un todo



1.5.2 Principios generales

El diccionario define la palabra *principio* como "una ley importante o suposición que subyace y se requiere en un sistema de pensamiento". En este libro se analizarán principios en muchos niveles distintos de abstracción. Algunos se centran en la ingeniería de software como un todo, otros consideran una actividad estructural general específica (por ejemplo, **comunicación**), y otros más se centran en acciones de la ingeniería de software (por ejemplo, diseño de la arquitectura) o en tareas técnicas (escribir un escenario para el uso). Sin importar su nivel de enfoque, los principios lo ayudarán a establecer un conjunto de herramientas mentales para una práctica sólida de la ingeniería de software. Ésa es la razón de que sean importantes.

David Hooker [Hoo96] propuso siete principios que se centran en la práctica de la ingeniería de software como un todo. Se reproducen en los párrafos siguientes:[13]

CONSEJO

Antes de comenzar un proyecto de software, asegúrese de que el software tenga un propósito para el negocio y que los usuarios perciben valor en él.

Primer principio: *La razón de que exista todo*

Un sistema de software existe por una razón: *dar valor a sus usuarios.* Todas las decisiones deben tomarse teniendo esto en mente. Antes de especificar un requerimiento del sistema, antes de notar la funcionalidad de una parte de él, antes de determinar las plataformas del hardware o desarrollar procesos, plantéese preguntas tales como: "¿Esto agrega valor real al sistema?" Si la respuesta es "no", entonces no lo haga. Todos los demás principios apoyan a éste.

Segundo principio: *MSE (Mantenlo sencillo, estúpido...)*

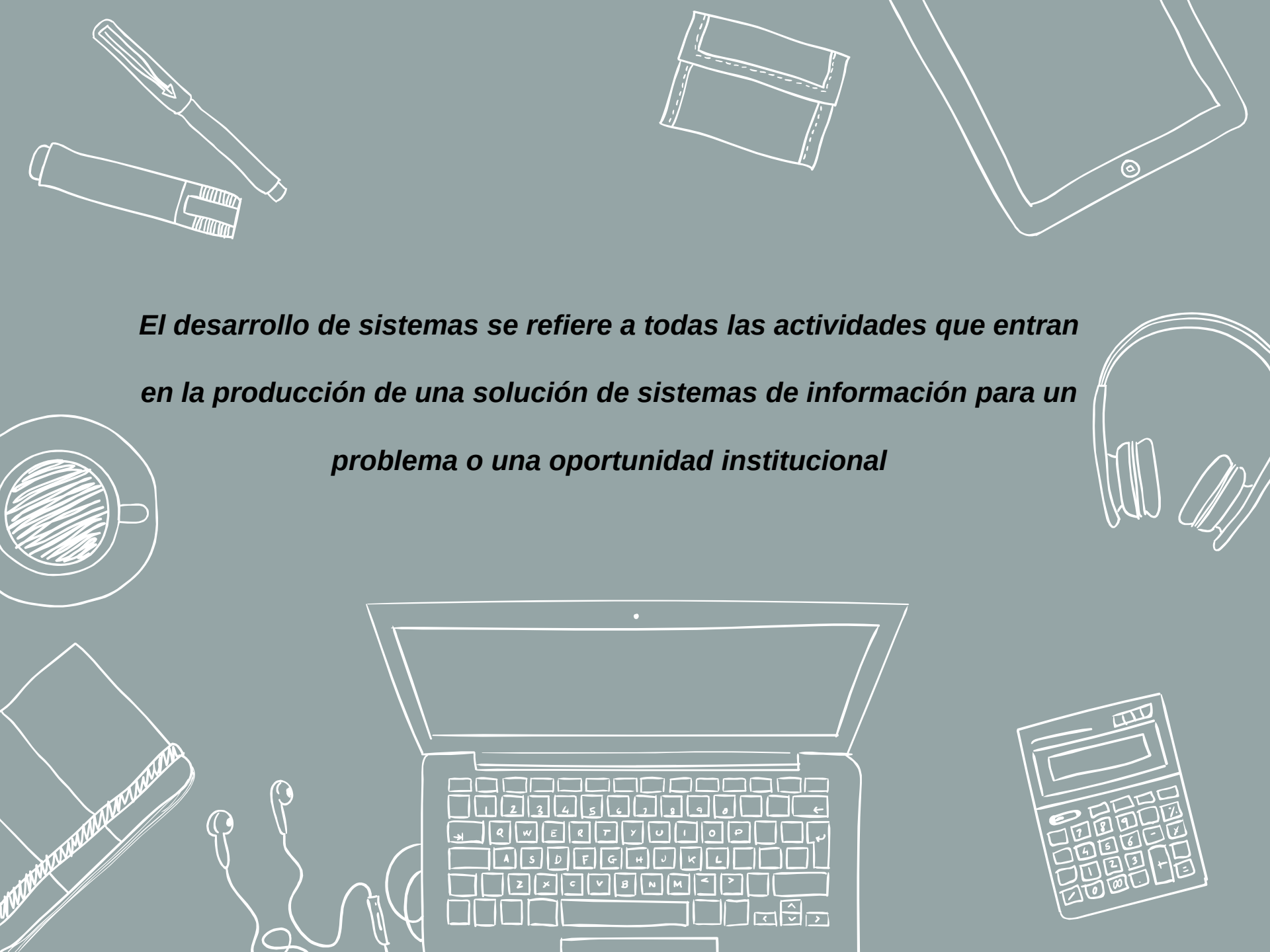
El diseño de software no es un proceso caprichoso. Hay muchos factores por considerar en

A trabajar

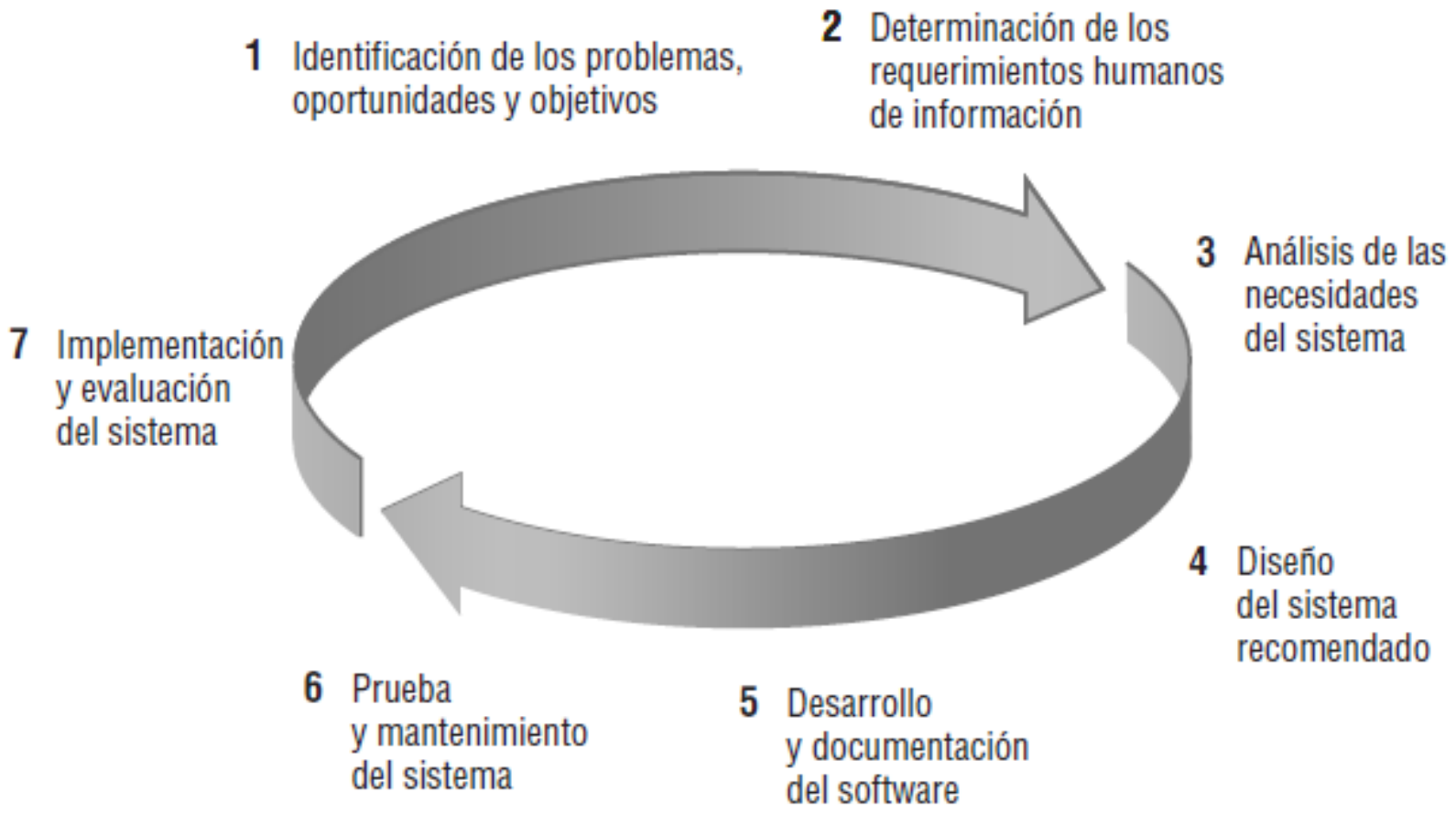
Desarrollo de software

Ideas para los sistemas de información

- Usuarios finales
- Departamento de sistemas de información
- Alta dirección



El desarrollo de sistemas se refiere a todas las actividades que entran en la producción de una solución de sistemas de información para un problema o una oportunidad institucional


1 Identificación de los problemas, oportunidades y objetivos
2 Determinación de los requerimientos humanos de información
3 Análisis de las necesidades del sistema
4 Diseño del sistema recomendado
5 Desarrollo y documentación del software
6 Prueba y mantenimiento del sistema
7 Implementación y evaluación del sistema

- Identificación de problemas, oportunidades y objetivos

Problemas

- Identificación de problemas, oportunidades y objetivos

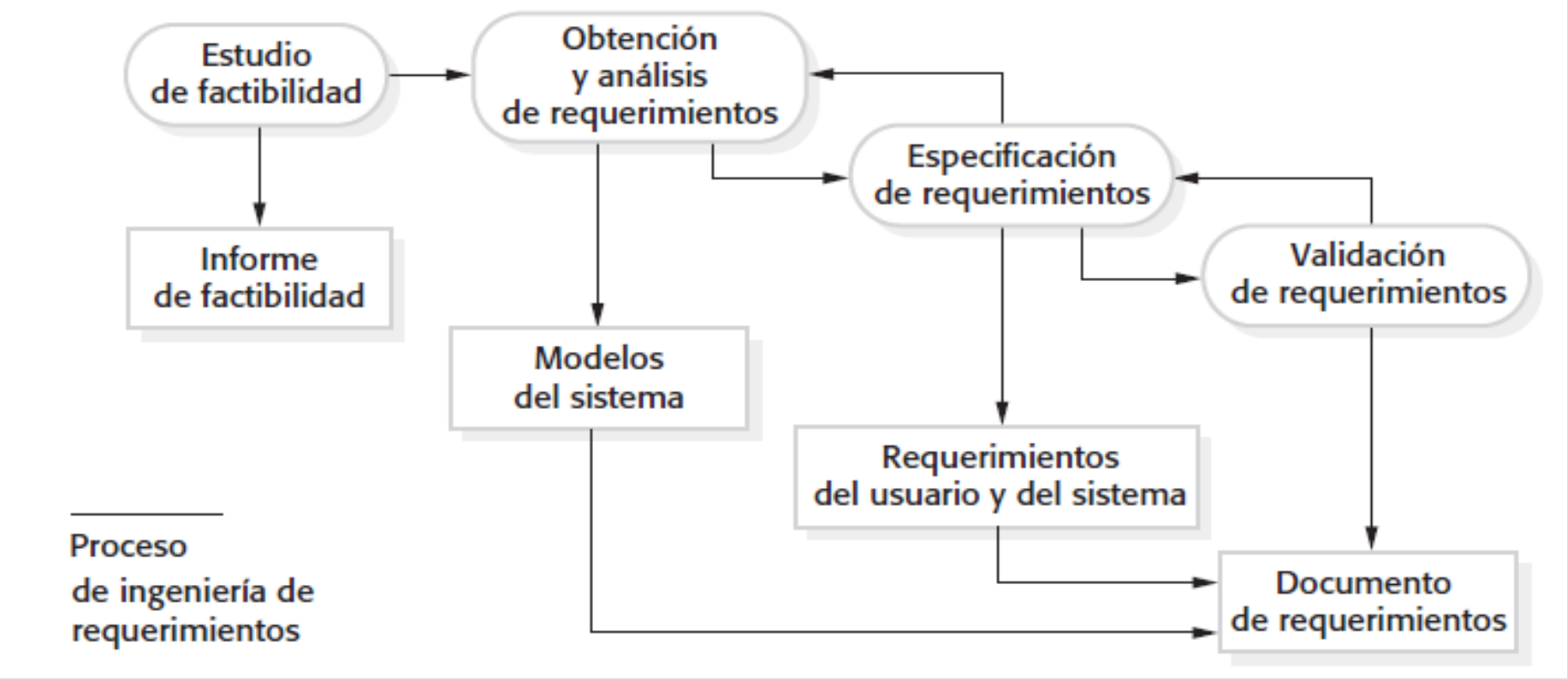
Oportunidades y Objetivos

- Identificación de problemas, oportunidades y objetivos

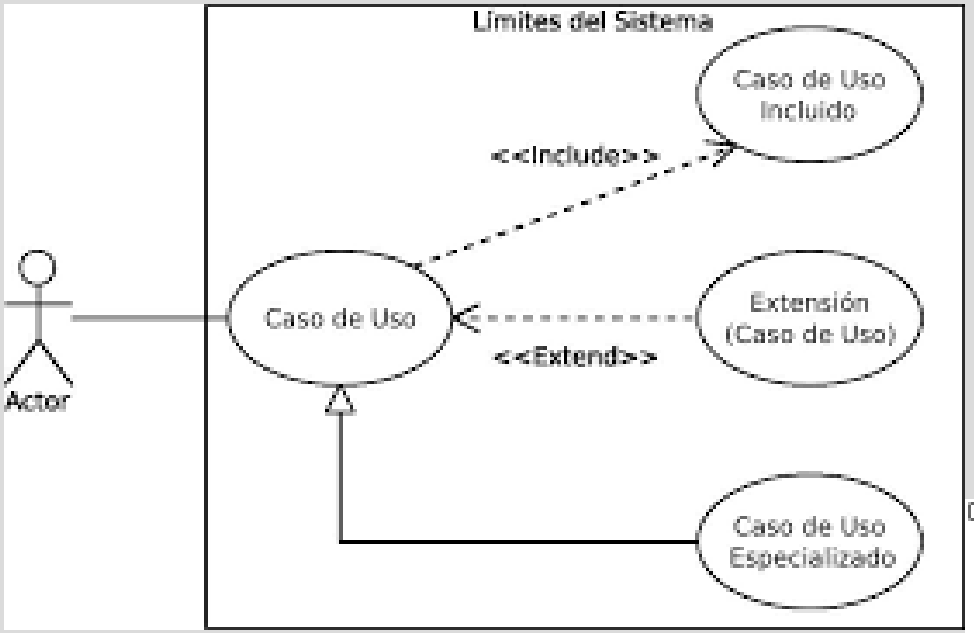
Factibilidad:

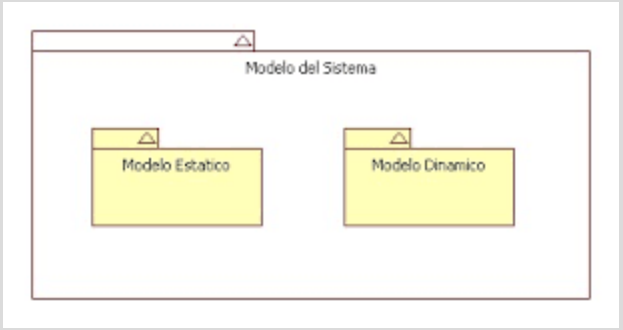
- Técnica
- Económica
- Operativa
- Política

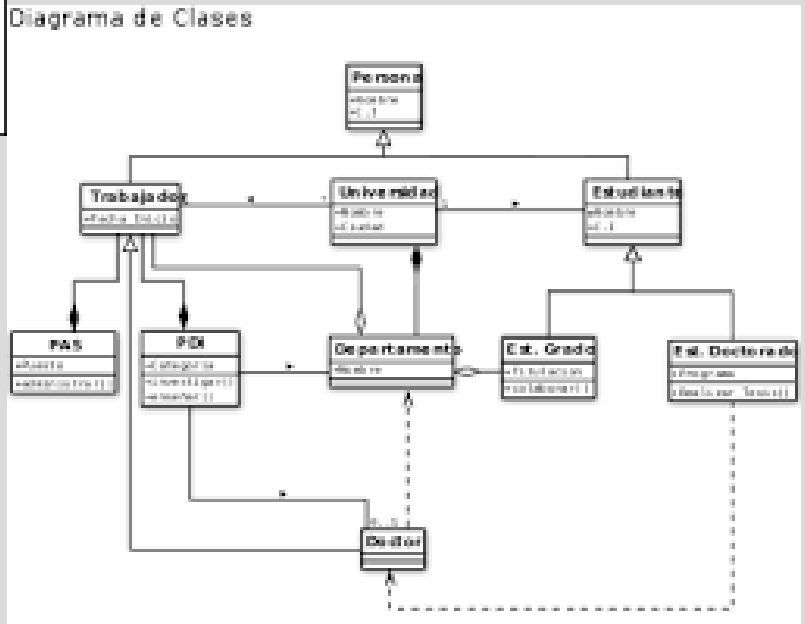
- Determinación de los requerimientos de información



- Análisis de las necesidades del sistema



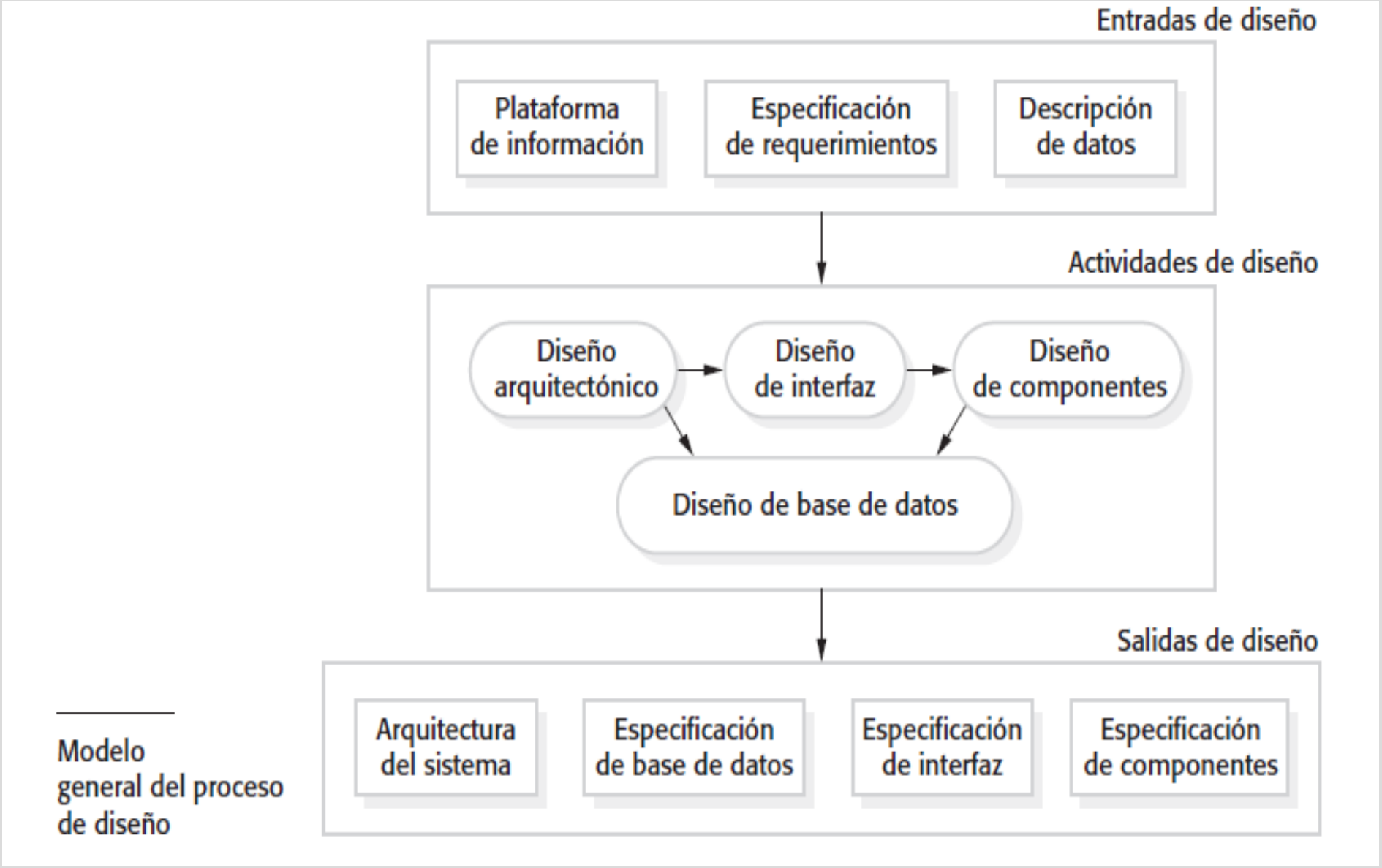




- Determinación de los requerimientos de información
- Análisis de las necesidades del sistema

1. Estudio de factibilidad
2. Obtención y análisis de requerimientos
3. Especificación de requerimientos
4. **Validación de requerimientos**

- Diseño del sistema recomendado

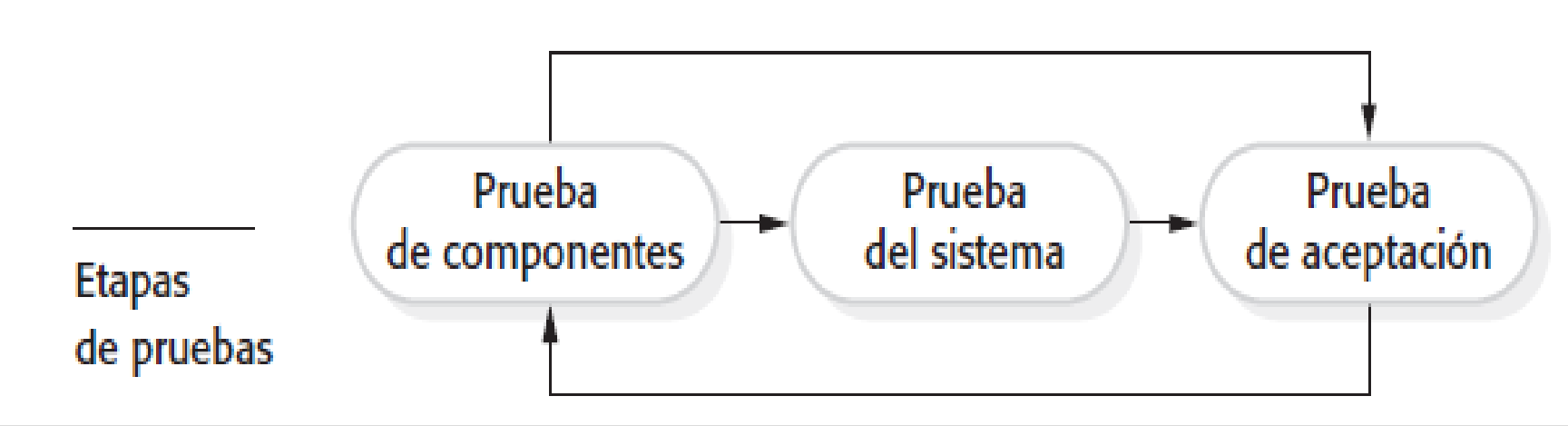


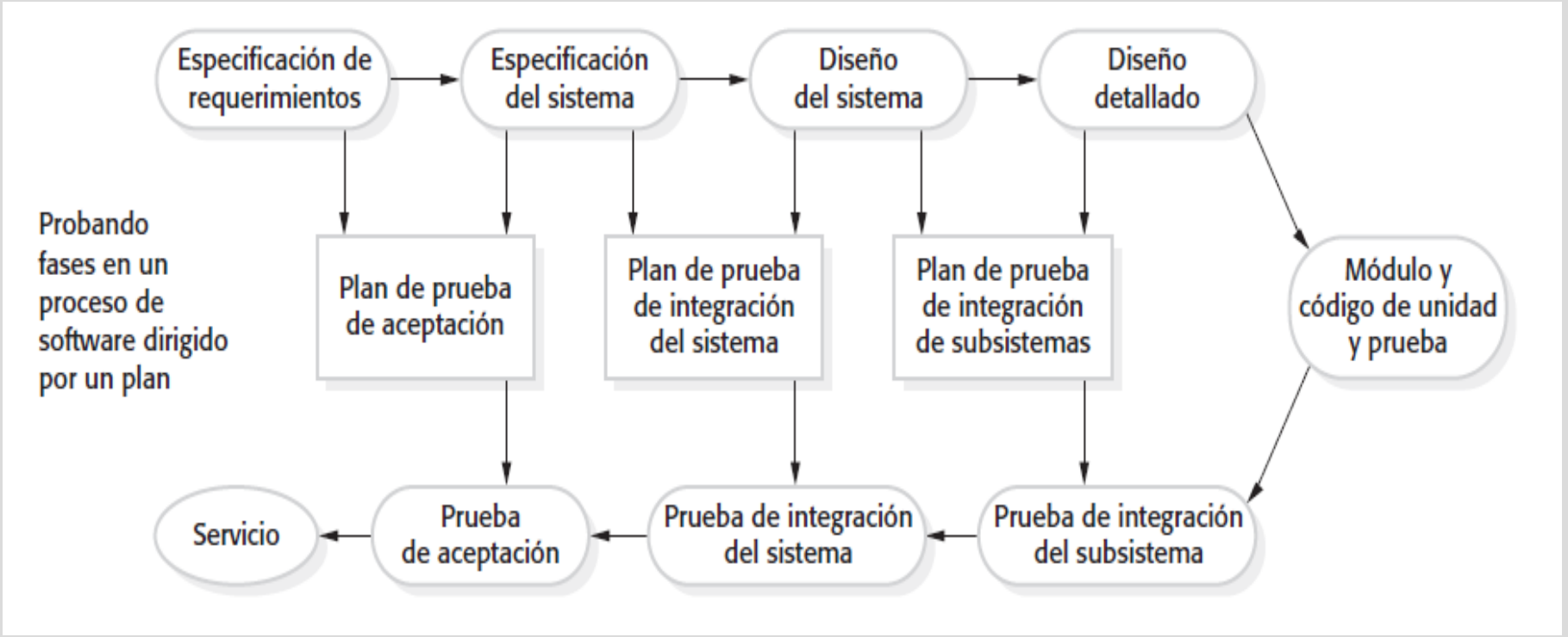
Modelo general del proceso de diseño

- Programación

  La programación es una actividad personal y no hay un proceso que se siga de manera general

- Pruebas


Etapas
de pruebas
Prueba
de componentes
Prueba
del sistema
Prueba
de aceptación


Probando fases en un proceso de software dirigido por un plan
Especificación de requerimientos
Especificación del sistema
Diseño del sistema
Diseño detallado
Plan de prueba de aceptación
Plan de prueba de integración del sistema
Plan de prueba de integración de subsistemas
Módulo y código de unidad y prueba
Servicio
Prueba de aceptación
Prueba de integración del sistema
Prueba de integración del subsistema

Prueba Alfa

Prueba de aceptación interna

Prueba Beta

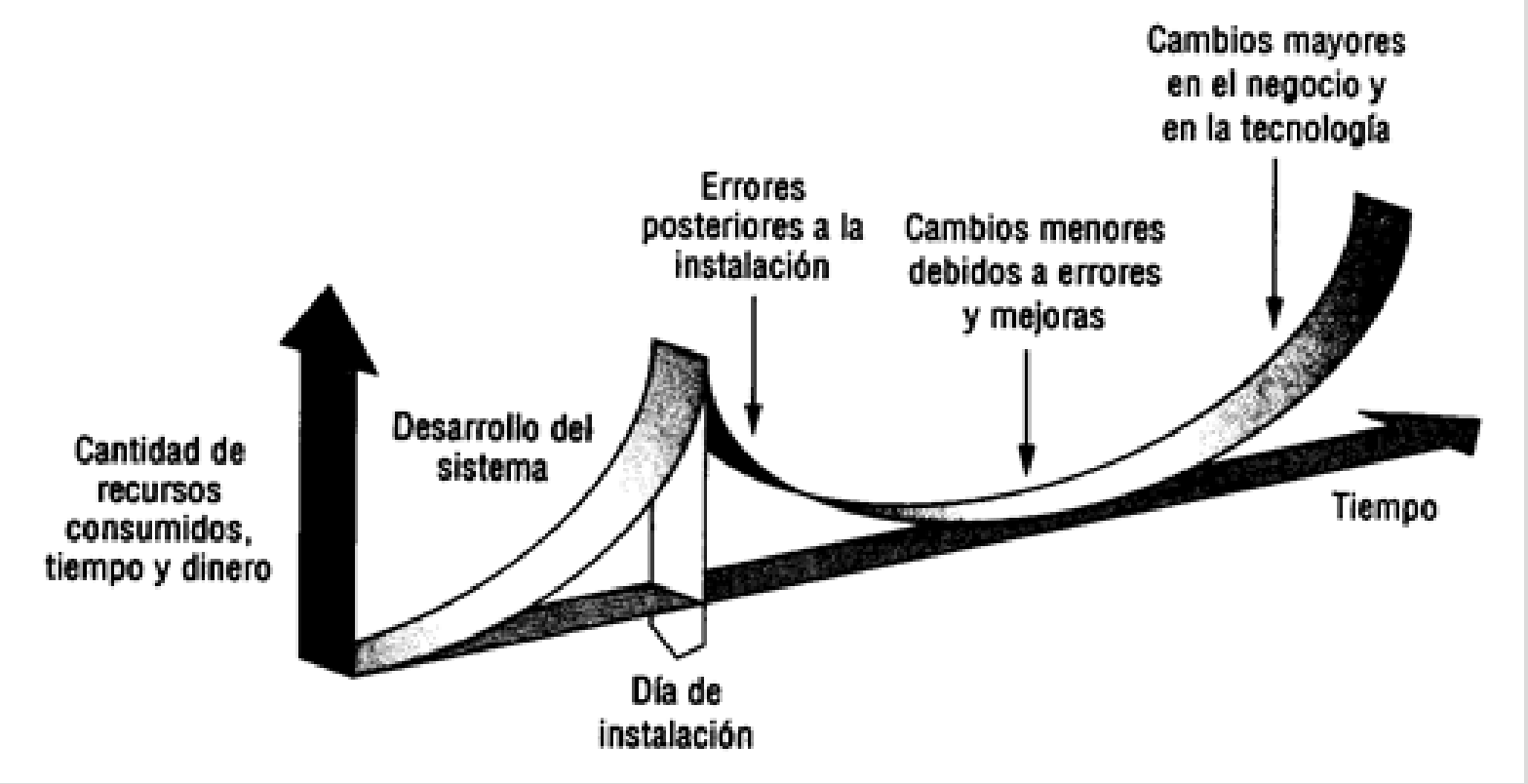
Pruebas realizadas por usuarios reales en un entorno real.

- Implementación

  Capacitación

  Conversión

Mantenimiento



Correctivo. Adaptativo. Perfectivo (ampliación o eficiencia). Preventivo (Reingeniería de software)

Proceso software vs ciclo de vida

Proceso de software:

actividades que comienza con la identificación de una necesidad y finaliza con el retiro del software

Ciclo de vida:

determina el orden de las fases del proceso software. Establece los criterios de transición para pasar de una fase a la siguiente

No existe único ***CV*** que define los estados por los que pasa un producto software

CV: sirve para tener un esquema que sirva como base para planificar, organizar, asignar personal, coordinar, presupuestar y dirigir las actividades

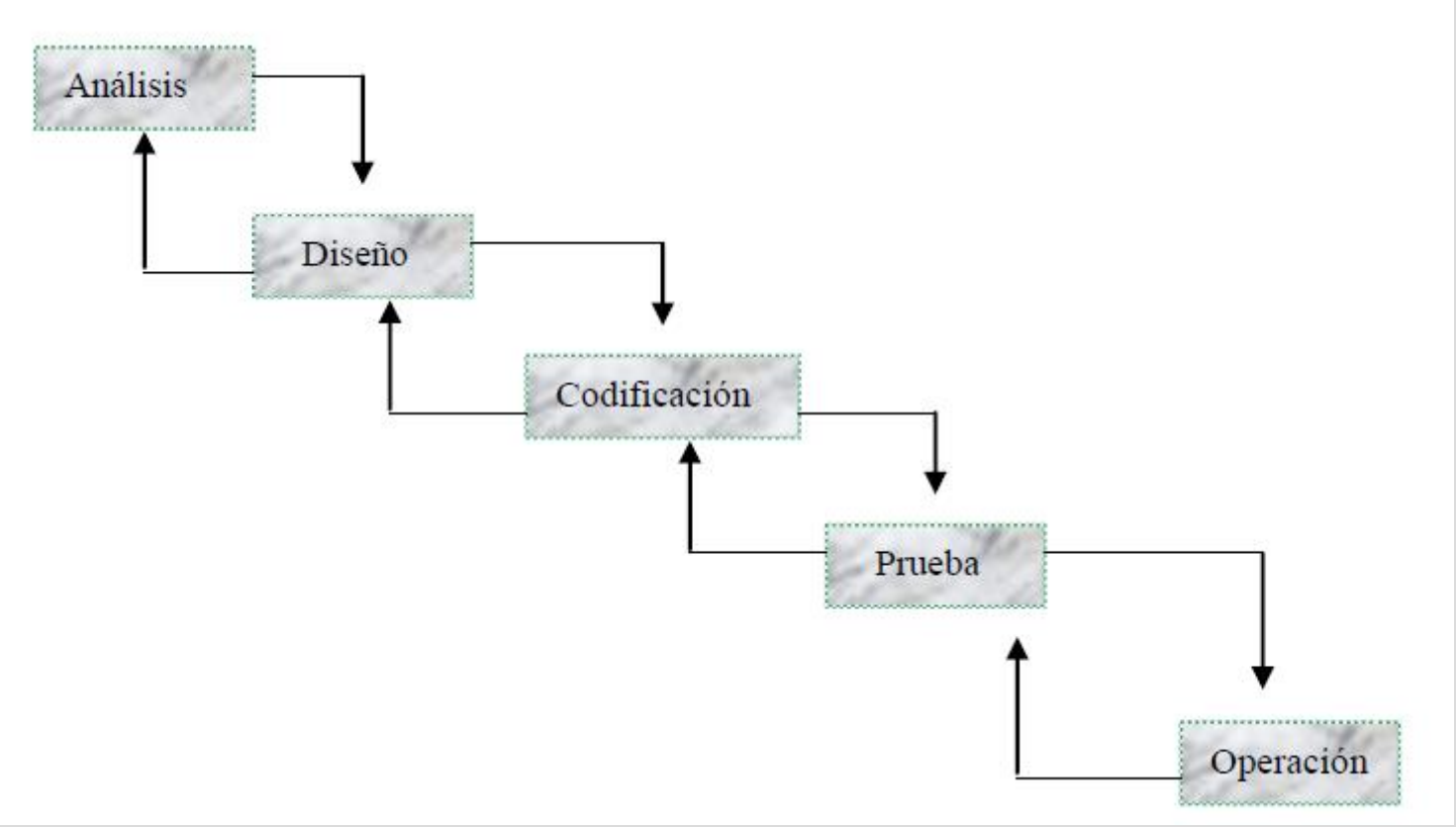
El CV apropiado se elige en base a :

- Cultura de la organización
- Deseo de asumir riesgos
- Área de aplicación
- Volatilidad de los requisitos
- Entendimiento de los requisitos

1) Modelos prescriptivos

1er ciclo de vida
Royce 1970

El ***modelo en cascada*** es un ejemplo de un proceso dirigido por un plan; en principio, se debe planear y programar todas las actividades del proceso, antes de comenzar a trabajar con ellas.



El ***modelo en cascada :***

Las principales etapas del modelo en cascada reflejan directamente las actividades fundamentales del desarrollo.

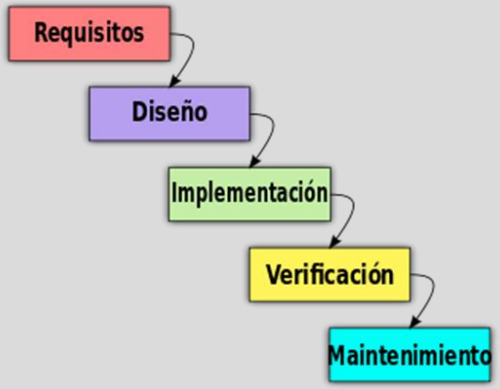
El resultado de cada fase consiste en uno o más documentos que se autorizaron (“firmaron”).

La siguiente fase no debe comenzar sino hasta que termine la fase previa.

Debido al paso de una **fase en cascada a otra**, este modelo se conoce como “modelo en cascada” o ciclo de vida clásico

Es un proceso dirigido por un plan; se debe **planear y programar** todas las actividades del proceso, **antes** de comenzar a trabajar con ellas.

El modelo en cascada sólo debe usarse cuando los requerimientos se entiendan bien y sea improbable el cambio radical durante el desarrollo del sistema



Problemas ciclo de vida en cascada

Es difícil para el cliente **enunciar** en forma **explícita** todos los requerimientos.

El modelo de la cascada tiene dificultades para aceptar la **incertidumbre** natural que existe al principio de muchos proyectos.

El cliente debe tener paciencia. No se dispondrá de una **versión funcional** del programa hasta que el proyecto esté muy avanzado.

Sirve como un modelo de proceso útil en situaciones en las que los **requerimientos son fijos** y el trabajo avanza en forma lineal hacia el final.

Ventajas:

Etapas **organizadas** en orden lógico. El diseño espera a los requisitos, el código espera al diseño…

Cada etapa necesita una **aceptación** del producto para pasar a siguiente etapa

Pasa el **menor** nro de **errores** de una etapa a la siguiente

Limitaciones:

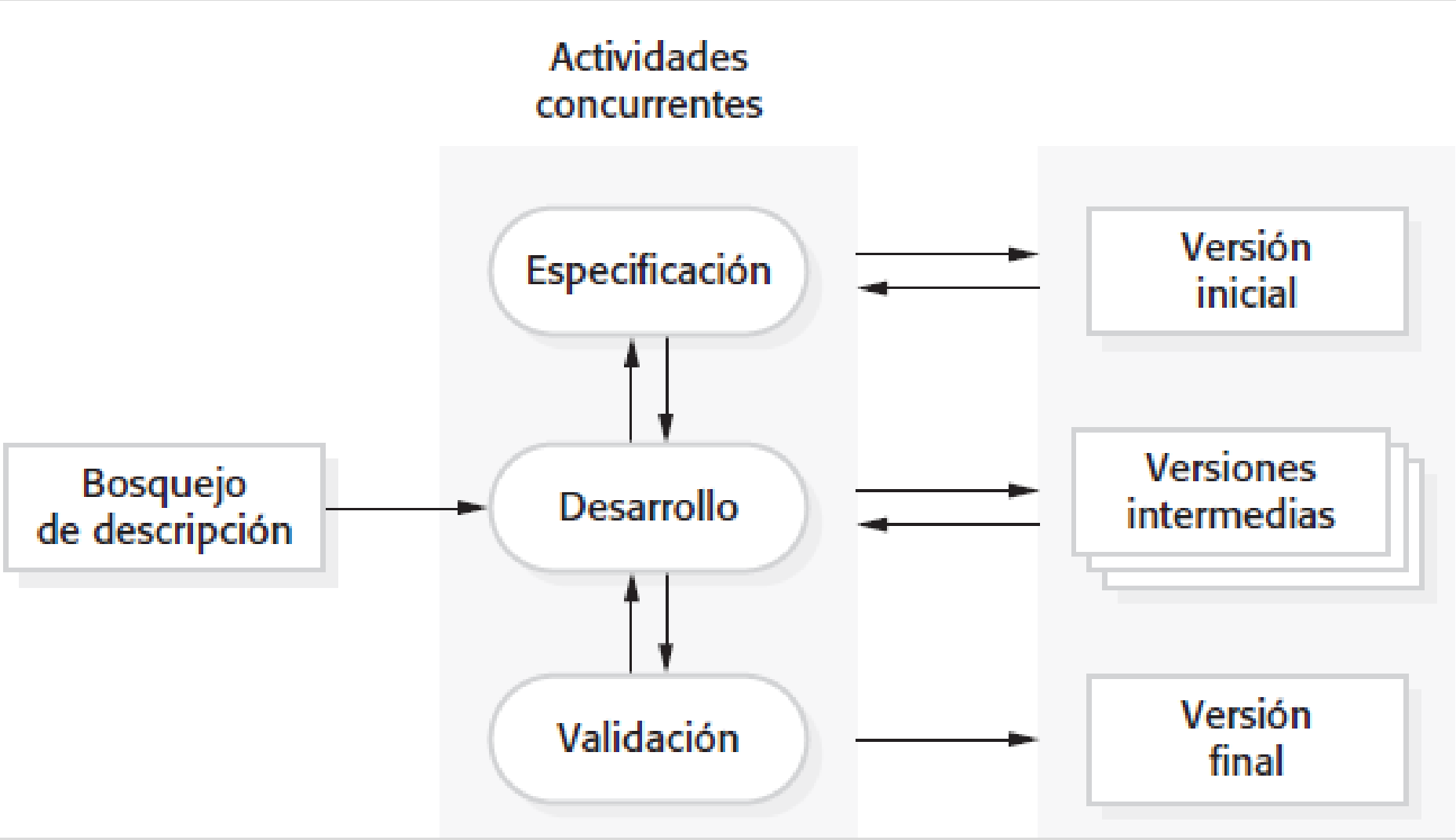
Los requisitos deben ser congelados antes de comenzar el diseño. (Pensar en requisitos tecnológicos)

Envía al cliente el primer producto solo después de consumir el 99% de los recursos.

Modelo desarrollo incremental

El sistema se desarrolla como una serie de versiones (incrementos), y **cada versión añade funcionalidad a la versión anterior**.

El desarrollo incremental se basa en la idea de diseñar una implementación inicial, exponer ésta al comentario del usuario, y luego desarrollarla en sus diversas versiones hasta producir un sistema adecuado


Actividades concurrentes
Especificación
Desarrollo
Validación
Bosquejo de descripción
Versión inicial
Versiones intermedias
Versión final

Beneficios importantes

Se **reduce el costo** de adaptar los requerimientos cambiantes del cliente

Es más sencillo obtener **retroalimentación** del cliente sobre el trabajo de desarrollo que se realizó

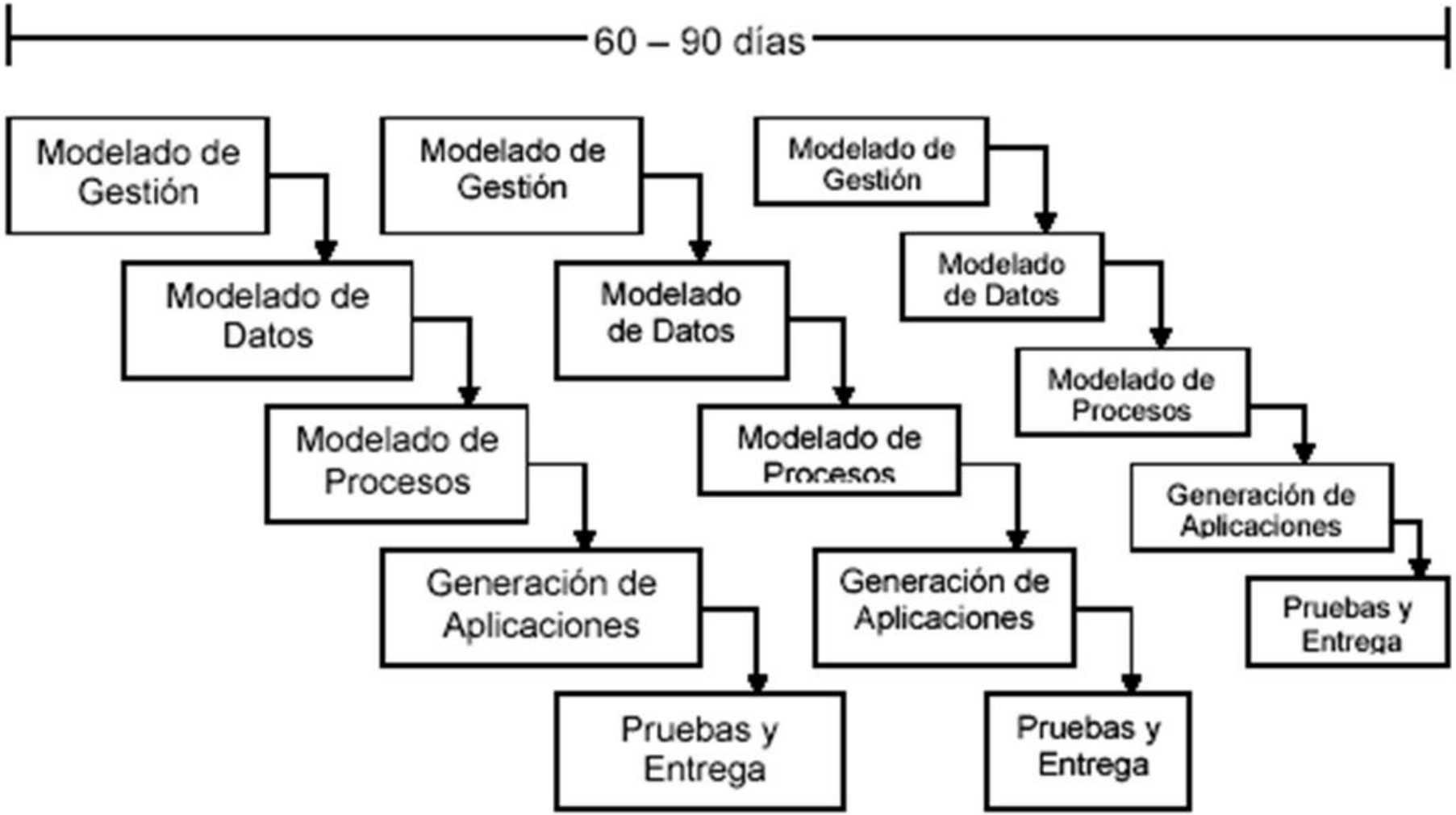
Es posible que sea **más rápida** la entrega e implementación de software útil al cliente, aun si no se ha incluido toda la funcionalidad

Problemas

Los problemas del desarrollo incremental se tornan particularmente agudos para sistemas **grandes**, **complejos** y de **larga duración**, donde diversos equipos desarrollan diferentes partes del sistema.

Modelo de desarrollo rápido de aplicaciones

El modelo DRA: desarrollo rápido de aplicaciones



2) Modelos de proceso evolutivo

El software ***evoluciona en el tiempo***. Los ***requerimientos*** del negocio y del producto ***cambien*** conforme avanza el desarrollo, lo que hace que no sea realista trazar una trayectoria rectilínea hacia el producto final

Se necesita un modelo de proceso diseñado explícitamente para adaptarse a un producto que evoluciona con el tiempo.

Los modelos evolutivos son iterativos se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software.

Prototipo y modelo en espiral

Modelo prototipado

El cliente no sabe que necesita

El ing de software no está seguro de la viabilidad

Esta técnica ofrece una versión del software de **funcionalidad reducida**

El sistema completo puede desarrollarse a través de un **proceso continuo de revisión** y **refinamiento** de las especificaciones de entrada

El prototipado ayuda a comprender los requisitos del usuario

Verificar la **viabilidad** del sistema

Es una herramienta iterativa. El **prototipado evoluciona hasta llegar al sistema final.**

Modelos derivados del uso del prototipo

Prototipo desechable

Prototipo maqueta

Prototipo evolutivo

Modelo en espiral

Modelo en espiral

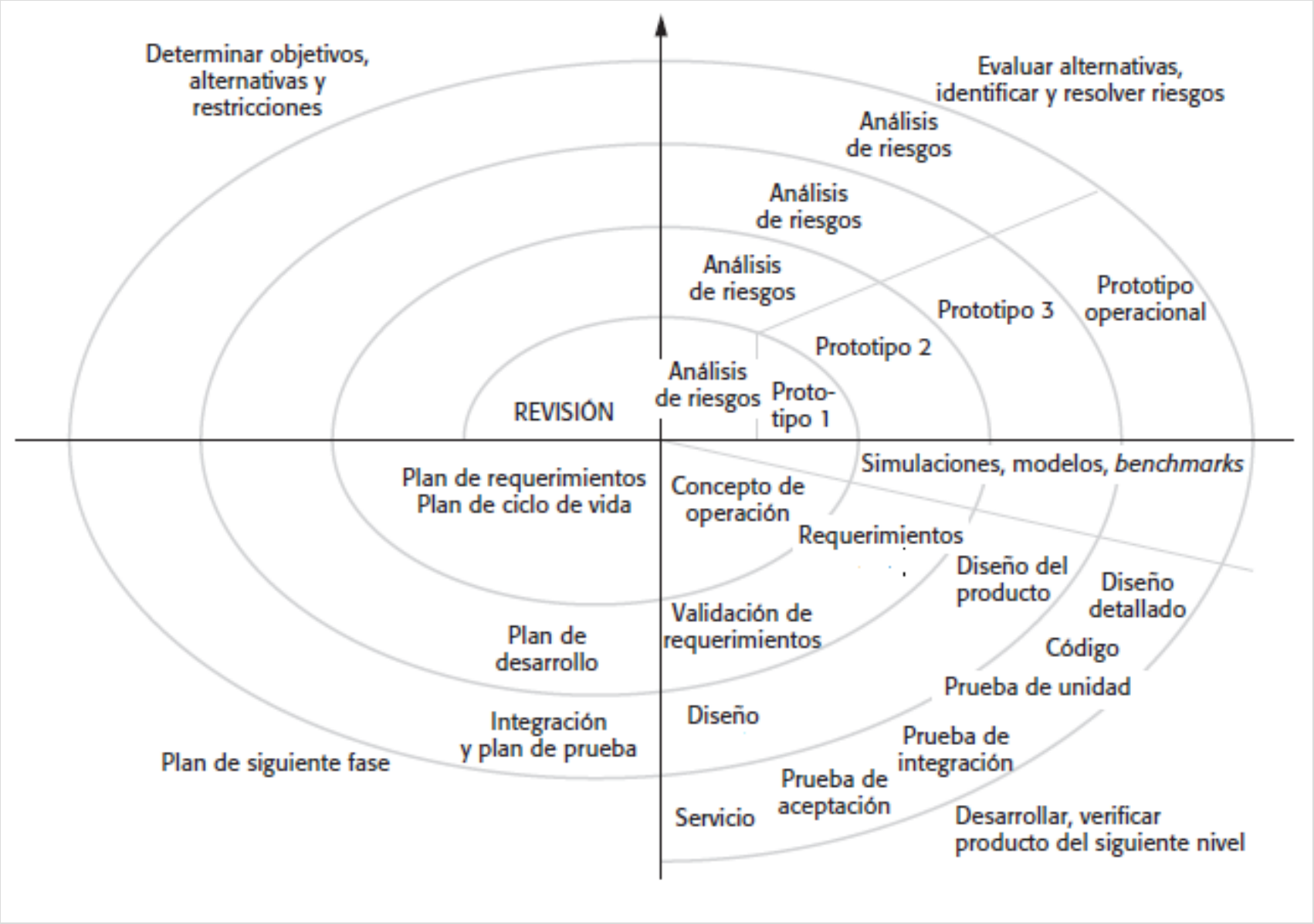
Boehm (1988) propuso un marco del proceso de software dirigido por el riesgo, el proceso de software se representa como una espiral, y no como una secuencia de actividades con cierto retroceso de una actividad a otra.

Modelo en espiral

Representa:

- Ciclos de desarrollo iterativo en forma de espiral, denotado por ciclos internos del ciclo de vida, análisis y prototipado precoz
- El modelo clásico en los ciclos externos

Espiral


Determinar objetivos, alternativas y restricciones
Evaluar alternativas, identificar y resolver riesgos
Análisis de riesgos
Análisis de riesgos
Análisis de riesgos
Prototipo 3
Prototipo operacional
Prototipo 2
Análisis de riesgos
Proto-tipo 1
REVISIÓN
Simulaciones, modelos, benchmarks
Plan de requerimientos
Plan de ciclo de vida
Concepto de operación
Requerimientos
Diseño del producto
Diseño detallado
Validación de requerimientos
Plan de desarrollo
Código
Prueba de unidad
Diseño
Integración y plan de prueba
Prueba de integración
Plan de siguiente fase
Prueba de aceptación
Servicio
Desarrollar, verificar producto del siguiente nivel

Durante las primeras iteraciones, lo que se entrega puede ser un **prototipo**. En las iteraciones posteriores se producen **versiones** cada vez más **completas** del sistema.

El primer circuito alrededor de la espiral da como resultado el desarrollo de una especificación del producto; las vueltas sucesivas se usan para desarrollar un prototipo y, luego, versiones cada vez más sofisticadas del software

El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala.

Ventajas de definir un proceso software

Un proyecto sin estructura es un proyecto inmanejable

Los **procesos software** se usan como:

- **Guía prescriptiva** sobre la documentación a producir
- **Bases** para determinar herramientas, técnicas y metodologías
- **Marco** para estimar consumo de recursos

Procesos principales

1. Proceso de **selección** de un **ciclo de vida**
2. Proceso de **gestión** del proyecto
3. Procesos orientados al desarrollo del software: producen, instalan, operan, mantienen y retiran
    1. Pre desarrollo
    2. Desarrollo
    3. Post desarrollo
4. Procesos integrales del proyecto

Un **mejor proceso** conlleva un **producto mejor**.

La definición del proceso permite al equipo comunicarse y trabajar conjuntamente hacia el objetivo

Ventajas:

- Facilita la comunicación
- Mejora la comprensión de la gestión
- Facilita la reutilización del proceso
- Ayuda a la gestión del proceso

Un proceso bien definido es más fácil de mejorar

https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.XKEvhphKjIU