Grado en Ingeniería Informática. 2º Curso
Curso Académico 2021-2022

Área de Lenguajes y Sistemas Informáticos
Departamento de Ingeniería Civil
Universidad de Burgos

Ingeniería del Software
Unidad Docente 1. Tema 2.1.1

Ciclo de Vida del Software

1. Definiciones

- **Ciclo de vida**:
  Conjunto de etapas que se han de llevar a cabo para crear, explotar y mantener un Sistema Informático.
- **Métodos**:
  Son las normativas que marcan las directrices que se han de seguir para llevar a cabo una tarea. Responde a la pregunta **QUÉ**.
- **Técnicas**:
  Es un modo de representación para la solución de un problema concreto. Responde a la pregunta **CÓMO**.
- **Metodología**:
  Es un conjunto coherente de métodos y técnicas que cubren más de una etapa del ciclo de vida.
- **Herramientas**:
  Proporcionan un soporte automático o semi-automático para el proceso y para los métodos.

2. Necesidades de las organizaciones

1. **Definir las actividades** necesarias en el desarrollo de un Sistema de Información.
2. Mantener una **coherencia entre todos los proyectos** de una misma organización.
3. **Introducir puntos de control** para realizar revisiones y controles de calidad, toma de decisiones.

- **Investigación de paradigmas o Modelos de Desarrollo**.
- **Ciclo de vida del software**: "*Marco de referencia que contiene los procesos, 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.*" ***Norma ISO 12207-1***

3. Procesos del Software

- **Un proceso del software**: Es un conjunto de actividades que conducen a la creación de un producto software.
- **¿En qué consisten estas actividades?**
  - Desarrollo del Software desde cero.
  - Desarrollo del Software ampliando y modificando los sistemas existentes y configurando e integrando software comercial o componentes del sistema.
- **Las herramientas CASE** pueden ayudar a algunas actividades del proceso.
- ¿Existe un proceso ideal?:

  NO.

  - Para **sistemas críticos** → Proceso de Desarrollo muy estructurado.
  - Para **sistemas de negocio**, con requerimientos rápidamente cambiantes → proceso flexible y ágil.

3. Procesos del Software

- **Actividades fundamentales comunes a cualquier proceso de software**:
  - **Especificación del Software**: Definir la funcionalidad del software y las restricciones en su operación.
  - **Diseño e implementación del Software**: Producir software que cumpla su especificación.
  - **Validación del Software**: Asegurarse de que el software hace lo que el cliente desea.
  - **Evolución del software**: Evoluciona para cubrir las necesidades cambiantes del cliente.
- **¿Cómo mejorar los procesos del Software?**
  - **La estandarización**: Mejora la comunicación, reduce el tiempo de formación, ayuda a introducir nuevos métodos, técnicas y buenas prácticas de Ingeniería del Software.

4. Modelos del Proceso del Software

- **Un Modelo general**:
  - Son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo del software.
  - Son marcos de trabajo del proceso que pueden ser extendidos y adaptados para crear procesos más específicos de ingeniería del software.
- **Un Modelo del Proceso del Software**:
  - Es una representación abstracta de un proceso del software.
  - Cada modelo de proceso representa un proceso desde una perspectiva particular → proporciona solo información parcial sobre ese proceso.

5. Paradigmas o Modelos de Desarrollo

- **Los paradigmas o modelos de desarrollo de Software**:
  - Son estrategias de desarrollo para organizar las diversas etapas y actividades del ciclo de vida del software.
  - Describen las transiciones entre las etapas, especificando qué actividades desarrollar en cada momento.
- **Selección de un modelo o paradigma específico**:
  - Dependiendo de la **naturaleza del proyecto** y/o aplicación, los métodos, las herramientas a utilizar, los controles y entregas que se requieren.
- **Modelado del proceso software**:
  - **Objetivo de las herramientas de tecnología de procesos**: Construir un modelo automatizado de la estructura común del proceso.
  - **Beneficios**: determinar flujo de trabajo típico, estructuras alternativas de menor tiempo o coste, organizar tareas, controlar el proceso y los productos que se generan, gestionar la calidad técnica, coordinar el uso de otras herramientas CASE, etc.

5. Paradigmas o Modelos de desarrollo

Paradigma o Modelo de desarrollo genérico:

El trabajo asociado a la ingeniería del Software puede dividirse en tres fases fundamentales, independientemente del área de aplicación:

- **FASE DE DEFINICIÓN**
- **FASE DE DESARROLLO**
- **FASE DE MANTENIMIENTO**

5. Paradigma o Modelos de desarrollo

Paradigma o Modelo de desarrollo genérico

Fase de Definición → ¿QUÉ?

- Qué información ha de ser procesada.
- Qué función y rendimiento se desea.
- Qué comportamiento del sistema.
- Qué interfaces van a ser establecidas.
- Qué restricciones de diseño existen.
- Qué criterios de validación se necesitan para definir.

Dependiendo del paradigma o modelo se definen un conjunto específico de actividades, pero las tareas principales son: ingeniería del sistema o de información, planificación del proyecto del software, y análisis de los requisitos.

Fase de Desarrollo → ¿CÓMO?

- Cómo han de diseñarse las estructuras de datos.
- Cómo ha de implementarse la función como una arquitectura del software.
- Cómo han de caracterizarse las interfaces.
- Cómo ha de traducirse el diseño en un lenguaje de programación
- Cómo ha de realizarse la prueba.

Las tareas principales son: diseño del software, generación de código y prueba de software.

Fase de Mantenimiento

Fase centrada en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software, y a cambios producidos por los requisitos cambiantes del software.

Cuatro tipos de cambio:

Corrección, Adaptación (Cambio de sistema operativo, reglas de la empresa, etc.), Mejora y Prevención (reingeniería).

Actividades a realizar:

Gestión de riesgos, revisiones técnicas formales, mediciones, garantía de calidad del software, seguimiento y gestión del proyecto de software, gestión de reutilización, …

5. Paradigmas o Modelos de desarrollo

Concepto de Modelo de Ciclo de Vida:

- Es una descripción de un proceso de software que se presenta desde una perspectiva particular.
- Es una abstracción de un proceso real.
- Existe una gran variedad de modelos diferentes "genéricos" o paradigmas de desarrollo de software.

5. Paradigma o Modelos de desarrollo

Paradigma o Modelo de desarrollo genérico

- Desglosando las fases anteriores, obtendríamos **las principales fases o etapas del ciclo de vida del software**:
  - Identificación del sistema y definición de requerimientos.
  - Análisis.
  - Diseño.
  - Desarrollo e implementación.
  - Integración y prueba del software.
  - Documentación.
  - Entrenamiento y uso.
  - Mantenimiento del software.

5. Paradigma o Modelos de desarrollo

Paradigma o Modelo de desarrollo genérico

- **Tres Modelos de proceso genéricos** que se utilizan ampliamente en la práctica actual de Ingeniería del Software:
  - El modelo en cascada.
  - Desarrollo evolutivo.
  - Ingeniería del Software basada en componentes.
- ¿Se pueden desarrollar los subsistemas dentro de un sistema más grande utilizando enfoques diferentes?

  Sí. Se pueden combinar.
- Los procesos basados en transformaciones formales se usan básicamente en ámbitos especializados.

5. Paradigmas o Modelos de desarrollo

Principales Modelos del Ciclo de Vida:

- Ciclo de vida en cascada o modelo tradicional (WaterFall).
- Prototipado.
- Modelo en cascada con prototipo desechable.
- Modelos o ciclos de vida evolutivos:
  - Modelo Incremental.
  - Modelo en Espiral.
- Programación Exploratoria.
- Transformaciones formales.
- Modelos de desarrollo orientados a objetos.
- …

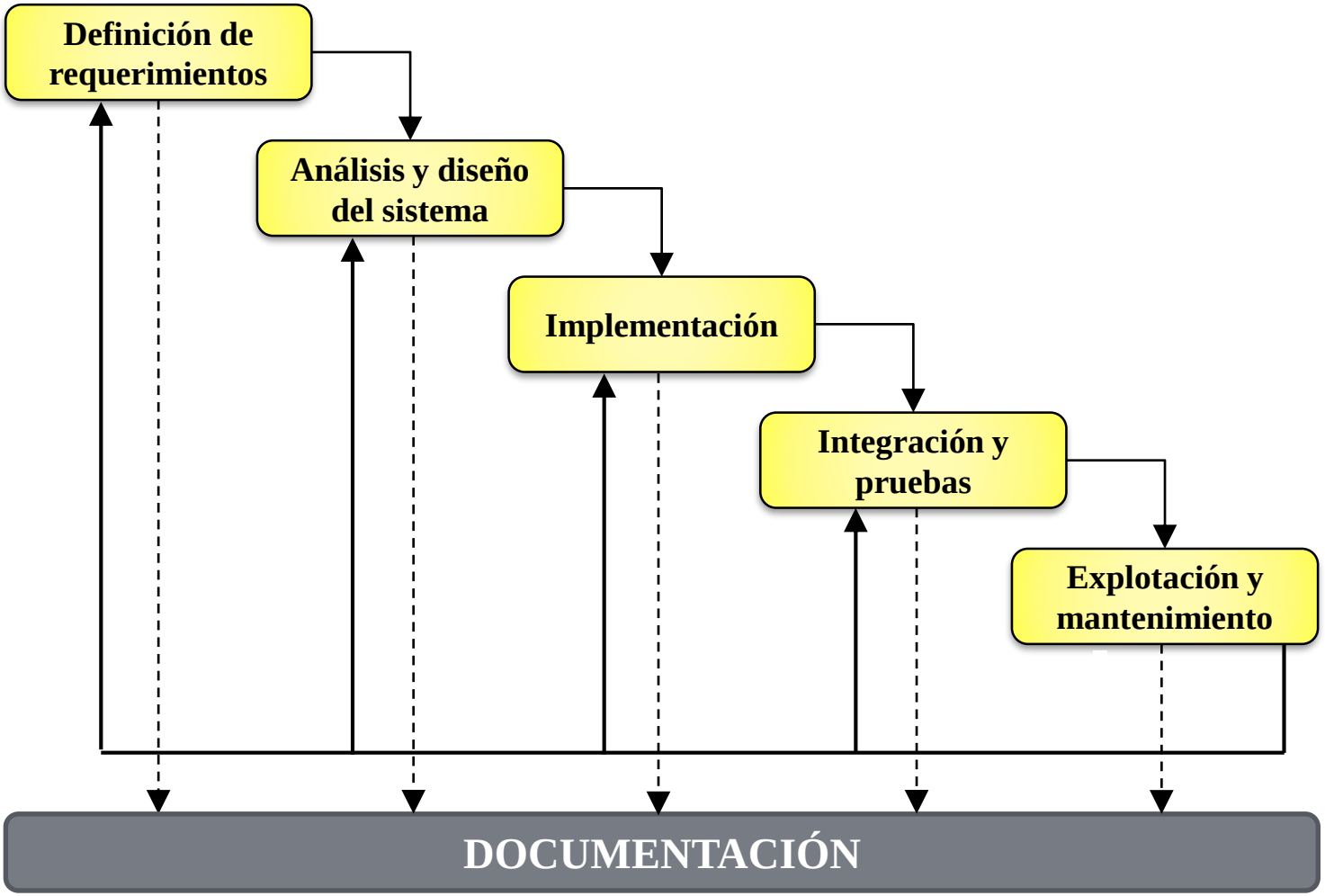
Modelo en cascada (Waterfall model) o modelo tradicional

- Primer modelo empleado (Royce 1970).
- También denominado: “ciclo de vida clásico” o “paradigma clásico”, “orientado a fases”, “lineal secuencial”.
- **En principio**:
  - **Ejecución secuencial de una serie de fases**. Diferentes etapas, las cuales son procesadas de un modo lineal. Una fase no debe comenzar hasta que la fase previa no haya finalizado.
- **En la práctica**:
  - Las fases se pueden superponer, proporcionándose información entre ellas.
  - Se hace un número reducido de iteraciones y la fase de Especificación de requisitos se congela.
- Cada fase genera documentación para la siguiente. El resultado de cada fase es uno o más documentos firmados (aprobados).
- **Finalidad**: **Establecer orden en el desarrollo de grandes productos de software**.

Modelo en cascada (Waterfall model) o modelo tradicional

- Base de muchos otros modelos, levemente mejorada y retocada a lo largo del tiempo.
- Aún en nuestros días sigue siendo muy utilizado.
- Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución.
- Estas actividades las representa como fases separadas del proceso: especificación de requerimientos, diseño del software, implementación, pruebas, etc.

Modelo en cascada (Waterfall model) o modelo tradicional



Modelo en cascada (Waterfall model) o modelo tradicional

- Anima a especificar lo que el sistema ha de hacer (definición de requerimientos) antes de la construcción del sistema.
- Planea los componentes que van a interaccionar.
- Gestiona el encuentro de errores.
- Genera un conjunto de documentos para más tarde ser utilizados y permitir un buen testeo y mantenimiento del sistema.
- Reducir los costes de desarrollo y mantenimiento.
- Referente a las tareas a realizar: Organización estructurada.
- **Principal característica del modelo**: la implantación del sistema descendente. (Ello requiere la abstracción del sistema, de los subsistemas y finalmente, de los módulos).

Ciclo de vida en cascada: Etapas

1. **Definición de requerimientos**: Estudio detallado de la situación actual del problema a tratar, definición de los requerimientos que debe cumplir el nuevo sistema.
2. **Análisis y diseño del sistema**: Descomposición modular de toda la aplicación, descripción detallada de cada uno de los módulos y sus interrelaciones, todo ello para poder facilitar al máximo la fase de codificación.
3. **Implementación (codificación)**: Cada módulo como resultado de la fase anterior es traducido a la herramienta o lenguaje apropiado.
4. **Integración y pruebas**: Verificación del correcto funcionamiento de cada módulo y todo el sistema una vez ha sido integrado, detectar errores en la codificación, definiciones de requerimiento y de diseño.
5. **Explotación y mantenimiento**: Garantizar el mantenimiento del sistema, corrección de errores detectados en esta fase, adaptación del sistema a nuevos entornos.

¿Cuál es la etapa que absorbe la mayoría del tiempo?

La fase de explotación y mantenimiento, y es un coste adicional para el cliente.

Ciclo de vida en cascada: Ventajas

- La documentación se produce en cada fase.
- Este modelo encaja perfectamente con otros modelos del proceso de ingeniería.
- Es sencillo de implementar y de comprender.
- En la práctica, se sigue utilizando para el desarrollo del software, particularmente cuando éste es parte de proyectos grandes de ingeniería de sistemas.

Ciclo de vida en cascada: Críticas

- Su inflexibilidad a dividir el proyecto en distintas etapas.
- Se deben hacer compromisos en las etapas iniciales → hace difícil responder a los cambios de requerimientos del cliente.
- Sólo se debe utilizar cuando los requerimientos se comprendan bien, y con poca probabilidad de cambiar radicalmente durante el desarrollo del sistema.
- El establecimiento explícito de todos los requisitos del sistema al principio del desarrollo.
- Poca flexibilidad para cambios en el sistema. No muestra interactividad entre fases.
- Nada hecho hasta el final. La validación de los requisitos iniciales no realizada hasta el final.

Ciclo de vida en cascada: Críticas

- La implementación del sistema de un modo ascendente implica las pruebas modulares, después la de los subsistemas, y finalmente, la del sistema completo. Los problemas graves suelen encontrarse en las fronteras entre subsistemas.
- Los errores de análisis y diseño son difíciles de eliminar, y se propagan a las etapas siguientes con un efecto conocido como “bola de nieve”.
- En la práctica, el modelo tiende a deformarse, y todo el peso de validación y mantenimiento recae, en su mayor parte, sobre el código fuente.
- El software va deteriorándose y resulta cada vez más difícil de mantener.
- Es poco realista, los proyectos reales raramente pueden seguir el flujo secuencial que se propone.
- En general, establecer todos los requisitos al principio del proceso es un “mito inalcanzable”.

Ciclo de vida en cascada: Críticas

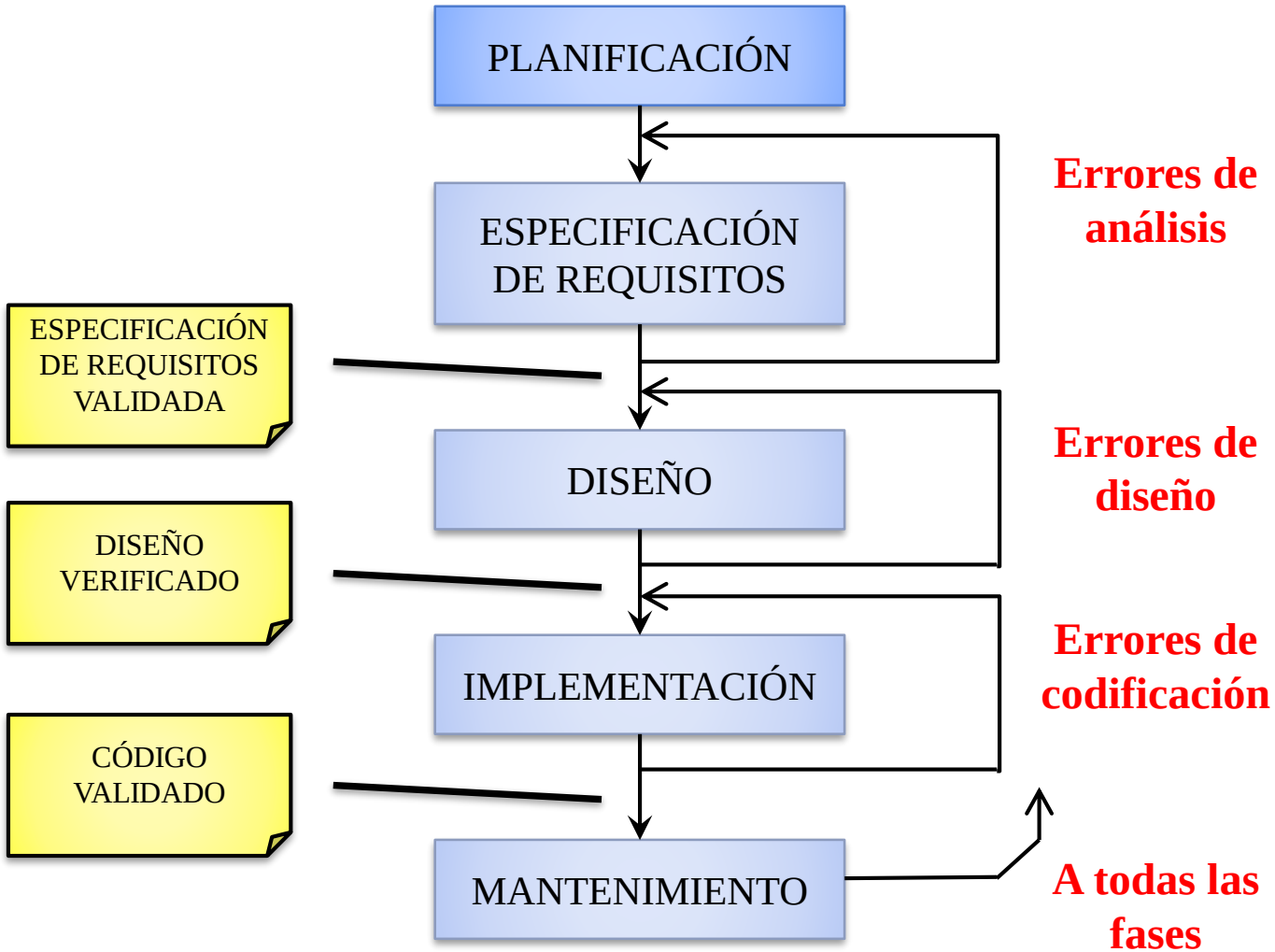
- Cuando el proyecto se alarga … La complejidad pasa a ser alta.
  - Las decisiones especulativas se incrementan y complican, ya que los requisitos se congelan y no existe retroalimentación a partir de implementación y pruebas reales.
  - En general, las cuestiones de alto riesgo no se abordan lo suficientemente pronto, no existe un intento activo de identificar y mitigar en primer lugar las cuestiones de riesgo.

Ciclo de vida en cascada: Mejoras Ciclo de Vida Estructurado

Objetivos principales de cada una de las fases:

1. **Estudio del sistema actual y viabilidad del nuevo sistema**: Identificación de usuarios envueltos, estudio de su puesto de trabajo, deficiencias actuales, sugerencias de cara al futuro. Establecer los objetivos del nuevo sistema. Determinar la viabilidad proponiendo diversas soluciones. Planificación de desarrollo. 5% o 10% del tiempo total del desarrollo.
2. **Análisis**: Especificación estructurada utilizando diferentes técnicas de diagramas para modelar el sistema nuevo.
3. **Diseño**: Establecer un conjunto de módulos e interfaces entre ellos, desglosando la especificación obtenida en la fase de análisis, facilitando la tarea de codificación, transformación de los modelos lógicos.
4. **Implementación**: Programación estructurada descendente e integración de los módulos.
5. **Generación de pruebas de aceptación**: Especificación de un conjunto de pruebas.
6. **Garantía de calidad**.
7. **Descripción de los procedimientos**: Toda la documentación necesaria para describir tanto los procesos como el producto resultante.
8. **Instalación e implementación del nuevo sistema al entorno**.

Modelo en cascada ideal



Modelo de Desarrollo Evolutivo

- Entrelaza las necesidades de especificación, desarrollo y validación.

1. **Desarrollo rápido**: Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas.
   - **Versión inicial**: Basada en la especificación.
2. **Refinamiento**: Después se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.
   - **Versiones intermedias**: Mediante un refinamiento a través de diferentes versiones basadas en el desarrollo.
3. **Versión Final**: Tras la validación de la última versión refinada.

Modelo de Desarrollo Evolutivo

- Tipos de Desarrollo Evolutivo:

1. **Desarrollo Exploratorio**

   - **Objetivo**: Se comienza por lo más entendible y luego se trabaja con el cliente para explorar sus requerimientos y entregar un sistema final.

2. **Prototipos desechables**

   - **Objetivo**: Comprender los requerimientos del cliente y desarrollar una definición mejorada de los requerimientos finales del sistema.

Prototipado

Utilizados principalmente en el desarrollo de sistemas donde existe un pobre conocimiento de los requerimientos de un sistema o la rápida evolución de los mismos a través del tiempo.

- Captura de requerimientos → "**diseño rápido**"
- El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles al usuario. El prototipo es evaluado por el cliente y el usuario, y utilizado para refinar los requerimientos del software a ser desarrollado.

Prototipado: Críticas

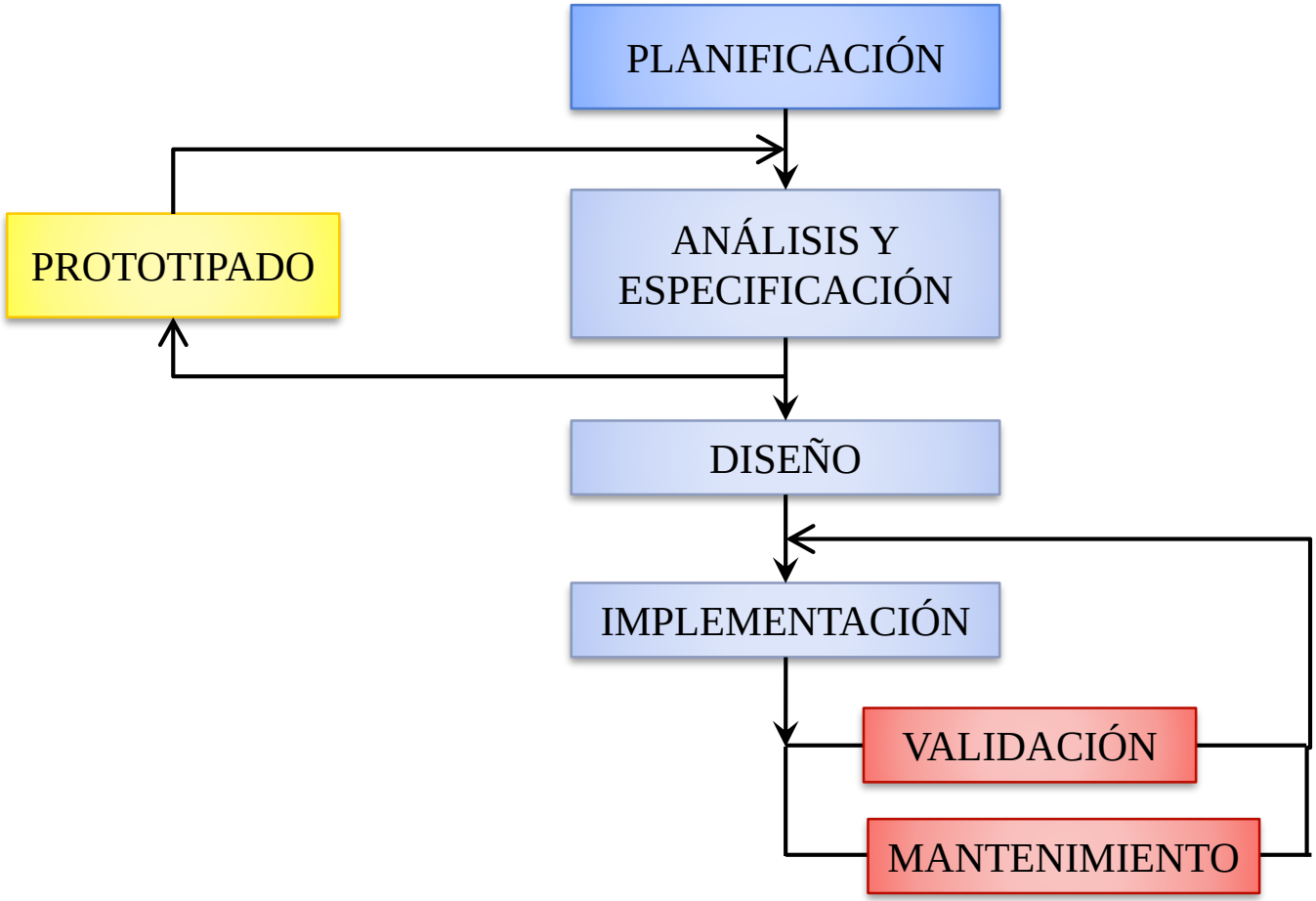
- El diseño rápido indica muchas de las veces el utilizar fragmentos de programas ya existentes y herramientas que faciliten la rápida generación de programas.

  - No se tiene en cuenta la calidad del software, ni su mantenimiento.
  - Ineficiencia de los programas, utilización de recursos, utilización de lenguajes inadecuados.

Prototipado: Fases

1. Preliminar análisis y especificación de los requerimientos de usuario.
2. Diseño e implementación de un prototipo.
   Énfasis en la interfaz de usuario, equipo pequeño para minimizar los costes de comunicación. Utilización de herramientas de ayuda al desarrollo.
3. Ejercicio del prototipo.
4. Refinamiento iterativo del prototipo.
5. Refinamiento de los requerimientos.
6. Diseño e implementación de un sistema.
   A partir de la fase 6 se sigue con el estándar del ciclo de vida.

Modelo en cascada con Prototipado desechable



Ingeniería del Software basada en componentes (CBSE)

- Se basa en la reutilización.
- Está compuesto de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para éstos.
- Los componentes también pueden ser sistemas por sí mismos.
- Ventajas:
  - Reduce la cantidad de software a desarrollar → Reducir costos y riesgos.
  - Permite una entrega más rápida del software.

- **Etapas**: Todas ellas en un orden secuencial.
  1. Especificación de requerimientos.
  2. Análisis de Componentes.
  3. Modificación de requerimientos.
  4. Diseño del sistema con reutilización.
  5. Desarrollo e integración.
  6. Validación del sistema.

Modelos basados en el Desarrollo Iterativo

- Modelo basado en la entrega incremental.
- Modelo con desarrollo en espiral.

Modelo basado en la entrega incremental

- Es un enfoque intermedio entre el modelo de desarrollo en cascada y el evolutivo.
- Se basa en:
  - La especificación, el diseño y la implementación del software se dividen en una serie de incrementos.
  - Los incrementos se desarrollan por turnos.
  - Los incrementos deben ser relativamente pequeños.
  - Cada incremento ha de entregar alguna funcionalidad del sistema.

Modelo basado en la entrega incremental

- **Etapas**:
  1. Definir esbozos de requerimientos.
  2. Asignar requerimientos a los incrementos.
  3. Diseñar la arquitectura del sistema.
  4. Desarrollar incrementos del sistema.
  5. Validar incrementos.
  6. Integrar incrementos.
  7. Validar sistema: ¿Decidir?
     a) Volver a la etapa 4.
     b) o Considerarlo SISTEMA FINAL.

Modelo basado en la entrega incremental

- **Inconvenientes**:
  - Dificultad de adaptar los requerimientos del cliente a incrementos de tamaño apropiado.
  - Muchos sistemas requieren un conjunto de recursos que se utilizan en diferentes partes del sistema.
  - Puede ser difícil identificar los recursos comunes que requieren todos los incrementos, ya que los requerimientos no se definen en detalle hasta que un incremento se implementa.

Modelo basado en la entrega incremental

- Tiene una variante: **Programación extrema**.
  - Desarrollo y entrega de incrementos de funcionalidad muy pequeños.
  - El cliente participa en el proceso, en la mejora constante del código y en la programación por parejas.

Modelo con desarrollo en Espiral

- Descrito por **Boehm**, mejores características de los dos modelos anteriormente expuestos.
- Incorpora el factor "riesgo del proyecto" al modelo de ciclo de vida.
- Se produce una cadena continua de productos, los cuales están disponibles para la examinación y evaluación por parte del cliente.
- Provee mecanismos para la aseguración de la calidad del software.
- La reevaluación después de cada fase permite cambios en las percepciones de los usuarios, avances tecnológicos o perspectivas financieras.

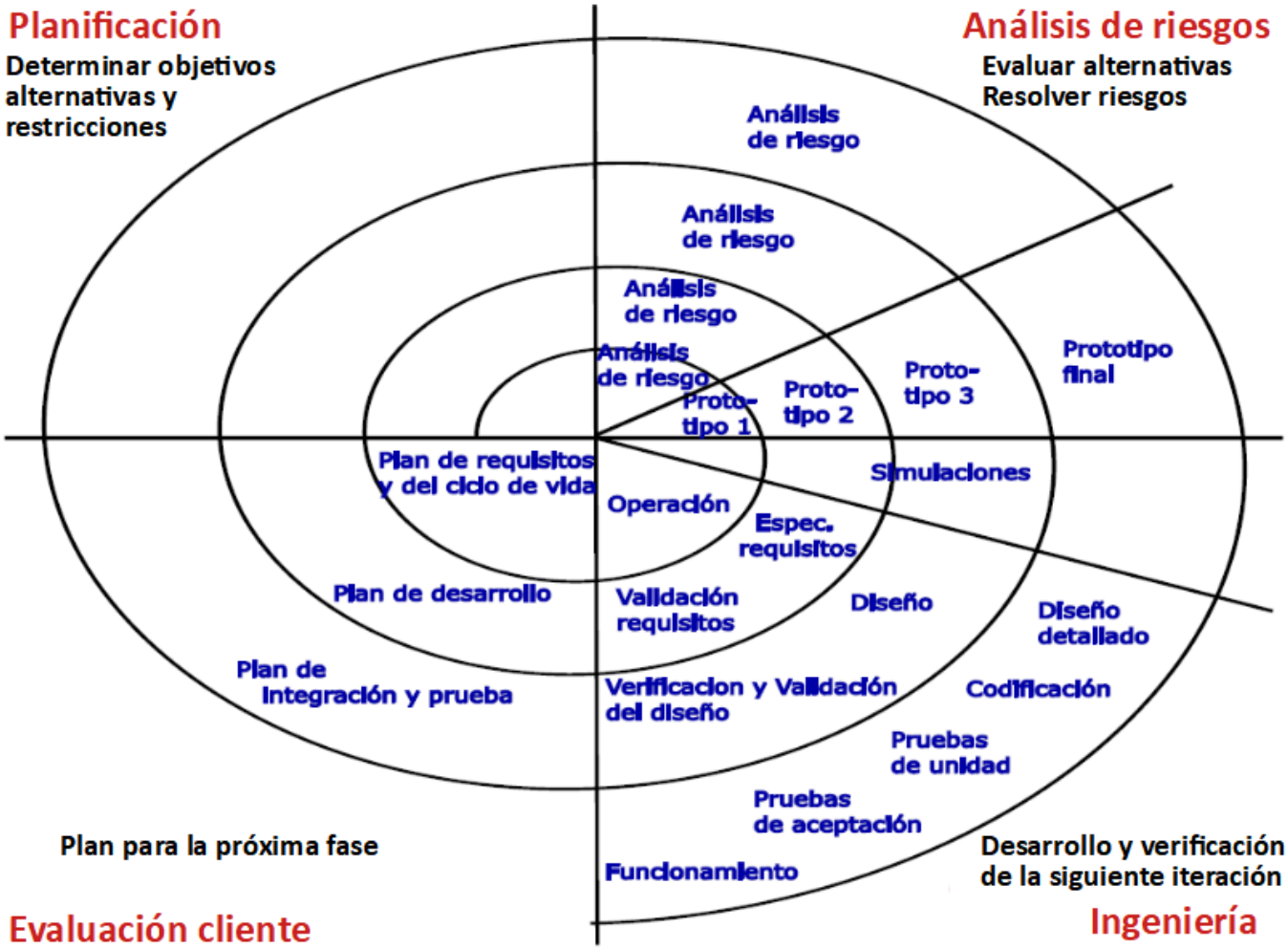
Modelo en Espiral: Cuadrantes

- **Planificación**: Determinación de objetivos, alternativas, restricciones, y elaboración del plan de desarrollo para el ciclo actual.
- **Análisis de riesgos**: Evaluación de las alternativas, identificación y resolución de riesgos. Se decide si se sigue o no con el proyecto.
- **Ingeniería**: Desarrollo del producto siguiendo un modelo: del ciclo de vida o cascada, prototipo, etc.
- **Evaluación por el cliente**: Valoración de resultados.

Modelo en Espiral: Aplicaciones

- **Dominio de aplicación**: Proyectos complejos, dinámicos, innovadores, ambiciosos, llevados a cabo por equipos internos (no necesariamente de software).
- **Dominios de aplicación inapropiados**: Dominio de problemas fáciles: si el dominio del problema está bien entendido y no hay mayores riesgos, es difícil y consume tiempo buscar riesgos donde no los hay.

Modelo en Espiral: Cuadrantes



Modelo en Espiral: Ventajas

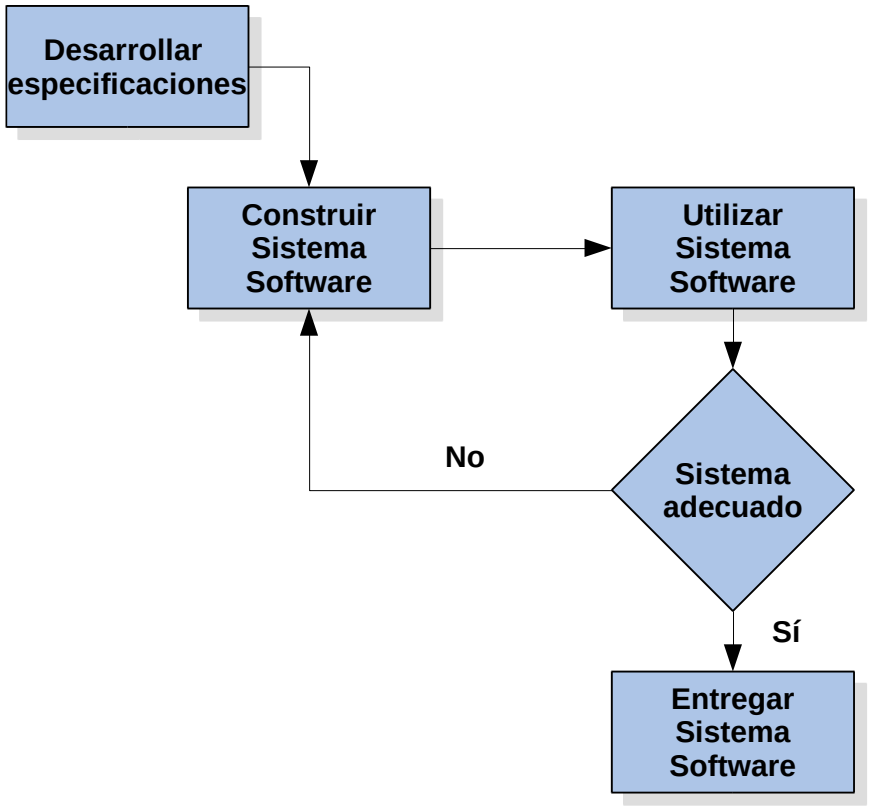
- Evita las dificultades de los modelos existentes a través de un acercamiento conducido por el riesgo.
- Intenta eliminar errores en las fases tempranas.
- Es el mismo modelo para el desarrollo y el mantenimiento.
- Provee mecanismos para la aseguración de la calidad del software.
- Trabaja bien en proyectos complejos, dinámicos e innovadores.
- La reevaluación después de cada fase permite cambios en las percepciones de los usuarios, avances tecnológicos o perspectivas financieras.
- La focalización en los objetivos y limitaciones ayuda a asegurar la calidad.

Modelo en Espiral: Inconvenientes

- Falta un proceso de guía explícito para determinar objetivos, limitaciones y alternativas.
- Provee más flexibilidad que la conveniente para la mayoría de las aplicaciones.
- La pericia de tasación del riesgo no es una tarea fácil. El autor declara que es necesaria mucha experiencia en proyectos de software para realizar estas tareas exitosamente.

Programación Exploratoria

- Basado en el desarrollo de una implementación inicial, exponiéndola a la opinión del usuario y luego refinándola a través de muchas etapas hasta obtener un sistema adecuado.
- Sistemas en los que es difícil (o imposible) establecer una detallada especificación (ejemplo Inteligencia Artificial).
- Forma de validación no medible, convirtiéndose en una apreciación subjetiva por parte del cliente. No puede ser utilizado en el desarrollo de grandes sistemas.



Transformaciones Formales

- Definición formal de los requerimientos del sistema (matemáticamente), ir desarrollando metódicamente hasta llegar al sistema definitivo.
- Se puede demostrar la validación de los requerimientos (que es diferente de los requerimientos del cliente).
- Ejemplo de aplicación: desarrollo de nuevos sistemas operativos, dispositivos médicos, desarrollo de aviónica, etc.
- La ambigüedad, lo incompleto y la inconsistencia se descubren y corrigen más fácilmente, mediante la aplicación del análisis matemático.

Actividades del Proceso

- Hay 4 actividades básicas: Especificación, Desarrollo, Validación y Evolución.
- Se pueden organizar de forma distinta en diferentes procesos del desarrollo:
  - Enfoque en cascada: Se organizan en secuencia.
  - Enfoque de desarrollo evolutivo: Se entrelazan.
- En la práctica real:
  - No hay forma correcta o incorrecta de organizar estas actividades. Depende: del tipo de software, de las personas y de la estructura organizacional implicada.