Ingeniería de Software

Prof. Carlos A. Fernández y Fernández
Instituto de Electrónica y Computación
Universidad Tecnológica de la Mixteca

¿Qué es la Ingeniería de Software?

La **ingeniería de software** es el término que cubre la totalidad del proceso de desarrollo (o producción) de un software.

Incluye áreas como:

- Administración de Proyectos
- Organización de Equipos
- Interacción Humano Máquina
- Programación, etc

¿Qué es la Ingeniería de Software?

- La Ingeniería de Software es la producción de un software **profesional**
- La producción es la punta del iceberg. El producto debe de:
  - Cumplir los requerimientos del cliente
  - Tener calidad
  - Estar probado
  - Documentado
  - Mantenido

Crisis del software.

El software no ha llevado el mismo nivel de crecimiento del hardware y la demanda de software se ha incrementado; por lo tanto su costo no ha disminuido sino aumentado con el paso del tiempo

Crisis del software: Productividad.

No se construye software a la velocidad requerida.

- Atención de la demanda. Hacia 1970 se estimaba que la demanda de software aumentaría entre el 11.5 y 17% anual, no pudiendo ser atendida y arrojando un déficit anual del 6 al 9.5% en E.U. (en realidad entre 1975 y 1985 crecio entre 21 y 23% anual)

Crisis del software: Productividad

Efectividad del esfuerzo.

En la década de los setenta, el Departamento de Defensa de los Estados Unidos, el mayor usuario de computadoras en el mundo informo de sus pagos en proyectos de cómputo:

- Gasto el 48% de los recursos destinados a la contratación de proyectos de software que nunca recibió.
- El 27% por sistemas que le fueron entregados pero que nunca utilizó.
- El 21% por productos que le fueron entregados con errores importantes, y que tuvo que abandonar o reelaborar (14%) o modificar (7%).

Crisis del software: Productividad

Parálisis debido al mantenimiento.

Yourdon -entre otros- ha considerado que en promedio las empresas gastan el 79% de su presupuesto para desarrollo en el mantenimiento del software existente.

Crisis del software: Costo.

El costo del software se ha incrementado al contrario del precio del hardware.

- En 1955, el costo del software era menor al 10% del costo del hardware
- En 1975 la diferencia del costo era tres veces mayor que el hardware.
- Para 1985 la diferencia se incremento 9 veces.

Crisis del software: Confiabilidad.

El software es uno de los productos construidos por el hombre más susceptibles a fallas.

- Si la industria del software fuera como la industria automotriz, la mayor parte de las compañías de software estarían hundidas en demandas.
- En junio de 1962, el Mariner I se salió de curso y tuvo que ser destruido: el problema - que costo 18.5 millones de dólares-, se debió a un error en uno de los programas que guiaban la nave.

Crisis del software: Confiabilidad

- El 15 de enero de 1990, el sistema de larga distancia de la AT&T sufrió una falla catastrófica que dejó paralizada la mayor parte de la red telefónica nacional de Estados Unidos durante nueve horas. El problema fue en el software de enrutamiento.
- El diseño deficiente de la interfaz con el usuario fue el factor principal de la identificación incorrecta de una imagen de radar, que resultó en el abatimiento del vuelo iraní y la muerte de sus 290 pasajeros.

Complejidad del software.

"La complejidad del software es una propiedad inherente, no accidental". Fred Brooks.

- La complejidad del dominio del problema.
- La dificultad de administrar el proceso de desarrollo de software.
- La naturaleza abstracta del software.

Complejidad: El límite de Miller.

La capacidad de los sistemas de software frecuentemente exceden la capacidad del intelecto humano.

El psicólogo George Miller dice que el ser humano solamente puede manejar, procesar o mantener la pista de aproximadamente siete objetos, entidades o conceptos a la vez.

En problemas con múltiples elementos, arriba de entre 7 y 9 elementos los errores en los procesos crecen desorbitadamente.

Complejidad: ¿cómo se enfrenta?

¿Si los sistemas de software son complejos, y nuestra capacidad de hacer frente a cuestiones complejas es limitada, como hemos podido construir software?

- Por medio de la **descomposición**.
  - La técnica para enfrentar la complejidad es conocida desde los tiempos antiguos: *divide y vencerás.*

Descomposición

La descomposición se puede ver de dos formas:

- Descomposición algorítmica
- Descomposición orientada a objetos.

Descomposición

Descomposición algoritmica = Análisis y diseño estructurado.

Descomposición Orientada a Objetos = Análisis y diseño Orientado a Objetos

Descomposición Algorítmica.

- La descomposición algorítmica se aplica para descomponer un gran problema en pequeños problemas.
- La unidad fundamental de este tipo de descomposición es el subprograma.
- El programa resultante toma la forma de un árbol, en el que cada subprograma realiza su trabajo, llamando ocasionalmente a otro programa.

Descomposición Orientada a Objetos

- El mundo es visto como un conjunto de entidades autónomas, que al colaborar muestran cierta conducta.
- Los algoritmos no existen de manera independiente, estos están asociados a los objetos.
- Cada objeto exhibe un comportamiento propio bien definido, y además modela alguna entidad del mundo real.

Principales Metodologías Orientadas a Objetos

- **OOSE/Objectory** (Object Oriented Software Engineering) [Jacobson 92]
- **OOA&D/ROSE** (Object-Oriented Analysis & Design) [Booch 94]
- **OOA&D** (Object-Oriented Analysis & Design) [Coad y Yourdon 91]
- **OMT** (Object Modeling Technique) [Rumbaugh 91]
- Surge de Jacobson, Booch y Rumbaugh la metodología **UML**: Unified Modeling Language

Comparación entre Análisis Estructurado y Orientado a Objetos

- El análisis y diseño estructurado se concentra en especificar y descomponer la funcionalidad del sistema total. Ambas metodologías cuentan con modelos similares: objetos, dinámico y funcional.
- OMT está dominado por el modelo de objetos, en el modelo estructurado domina el funcional y el de objetos es el menos importante.

Comparación entre Análisis Estructurado y Orientado a Objetos

- El modelo estructurado se organizan alrededor de procedimientos. Los modelos OO se organizan sobre el concepto de objetos.
- Comúnmente los requisitos cambian en un sistema y ocasionan cambios en la funcionalidad más que cambios en los objetos.

Comparación entre Análisis Estructurado y Orientado a Objetos

- En el modelo estructurado la descomposición de un proceso en subprocesos es bastante arbitraria.
- El enfoque orientado a objetos integra mejor las bases de datos con el código de programación.

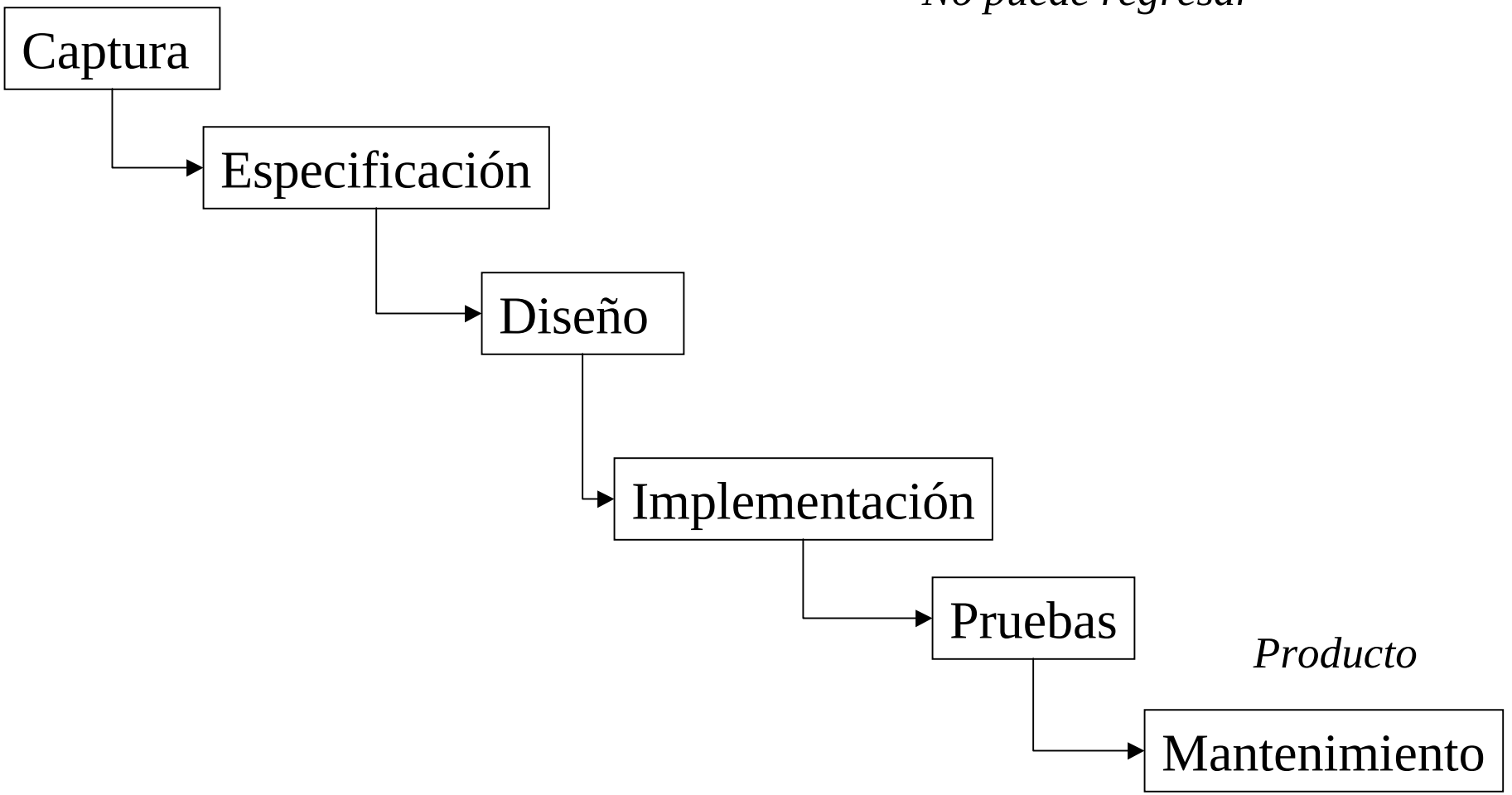
Ciclos de vida del software

La elección del ciclo de vida es independiente de la metodología de desarrollo del software.

- Método de Cascada
- Método de Cascada con retorno
- Método iterativo
- Modelo en espiral

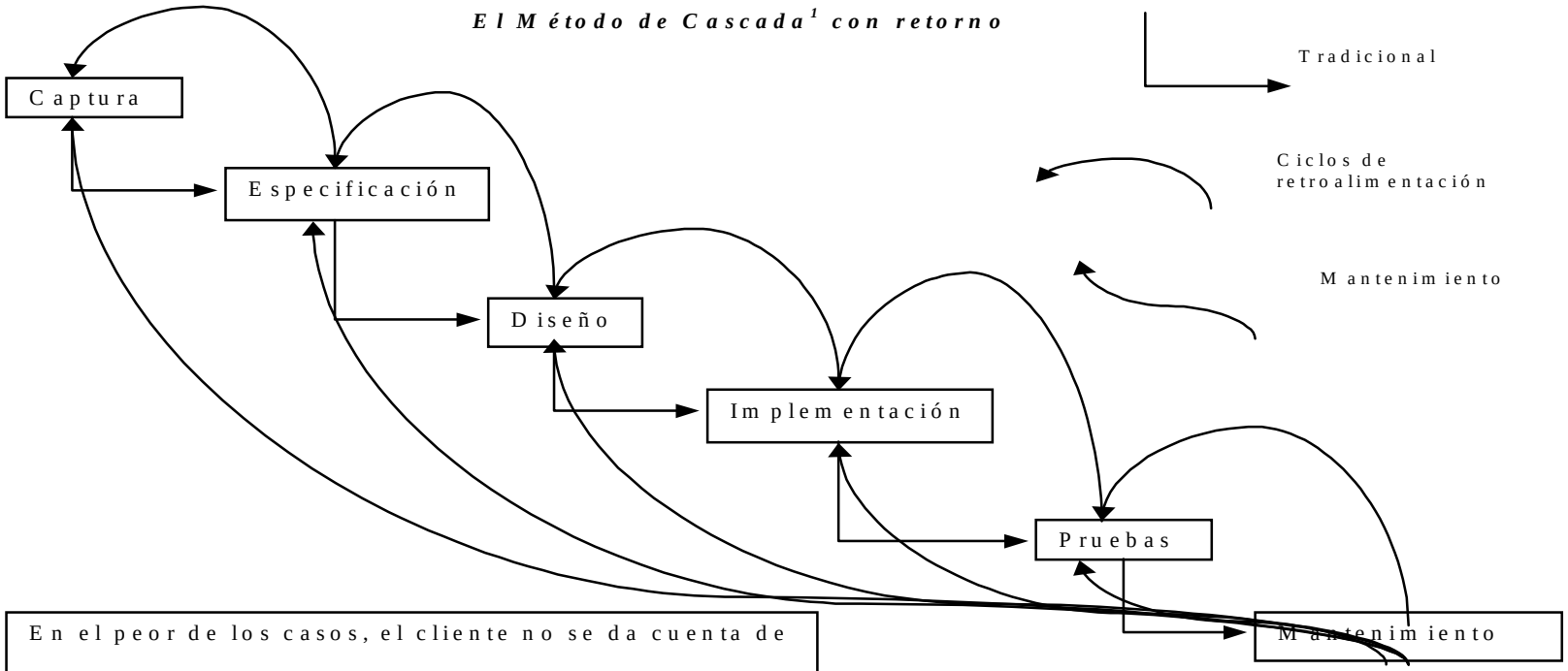
El Método de Cascada

No puede regresar



Producto

Método de Cascada con retorno



En el peor de los casos, el cliente no se da cuenta de que su sistema erróneo se está construyendo hasta que el software se libera.

Se debe de realizar en cada etapa la verificación y validación, y el resultado será utilizado dentro del proceso de desarrollo.

Carlos A. Fernández

[1] según Mark Dunlop en Introducing Software Engineering, 1994

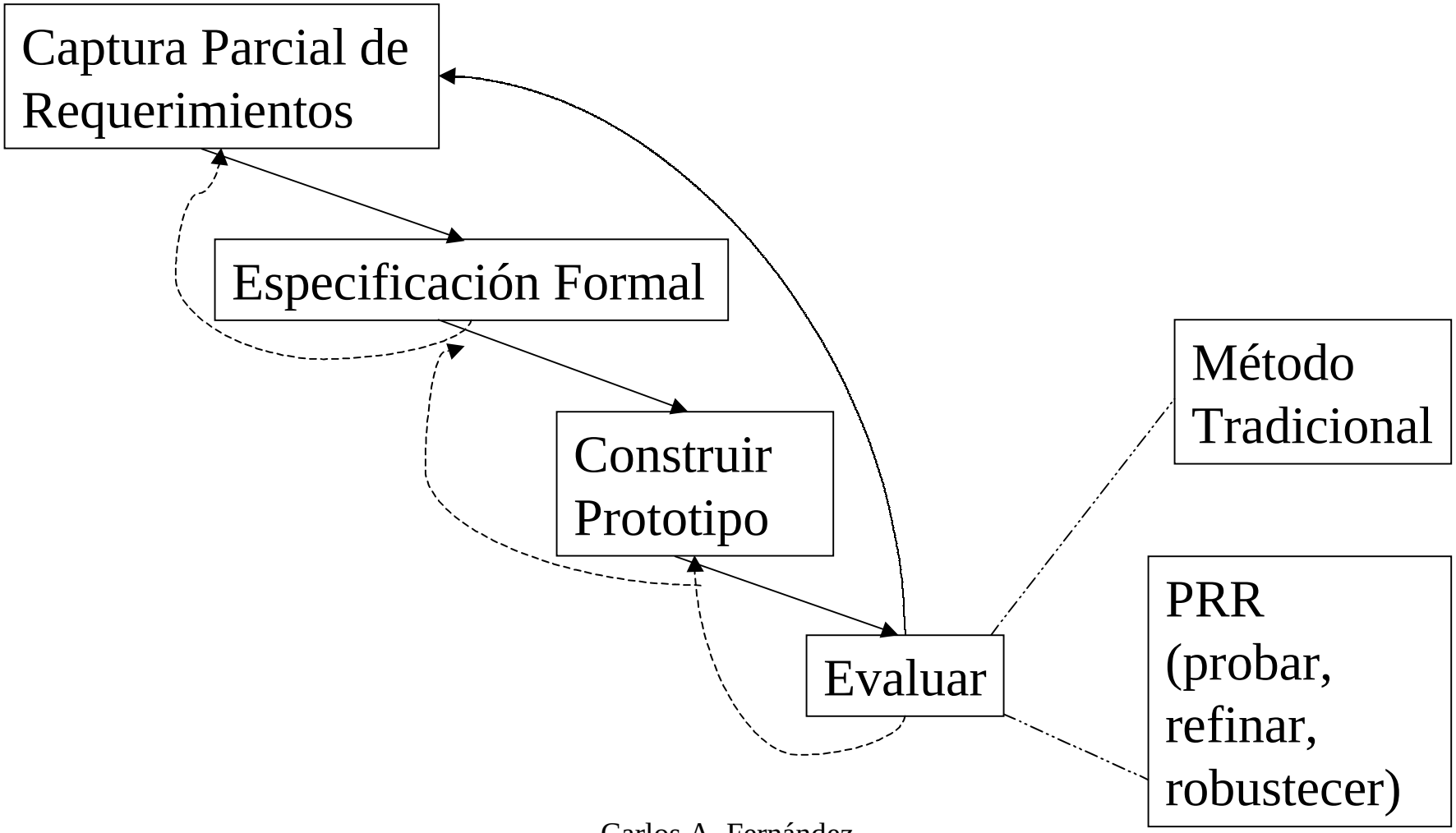
Mejoramiento Iterativo

- Reconocer cuándo realizar iteraciones
- El ciclo de vida de los **Prototipos**

Ventajas:

- capturas y especificaciones parciales
- mostrar el producto para su consideración
- prototipos **evolucionarios**
- prototipos **revolucionarios**

Ciclo de Vida de los Prototipos

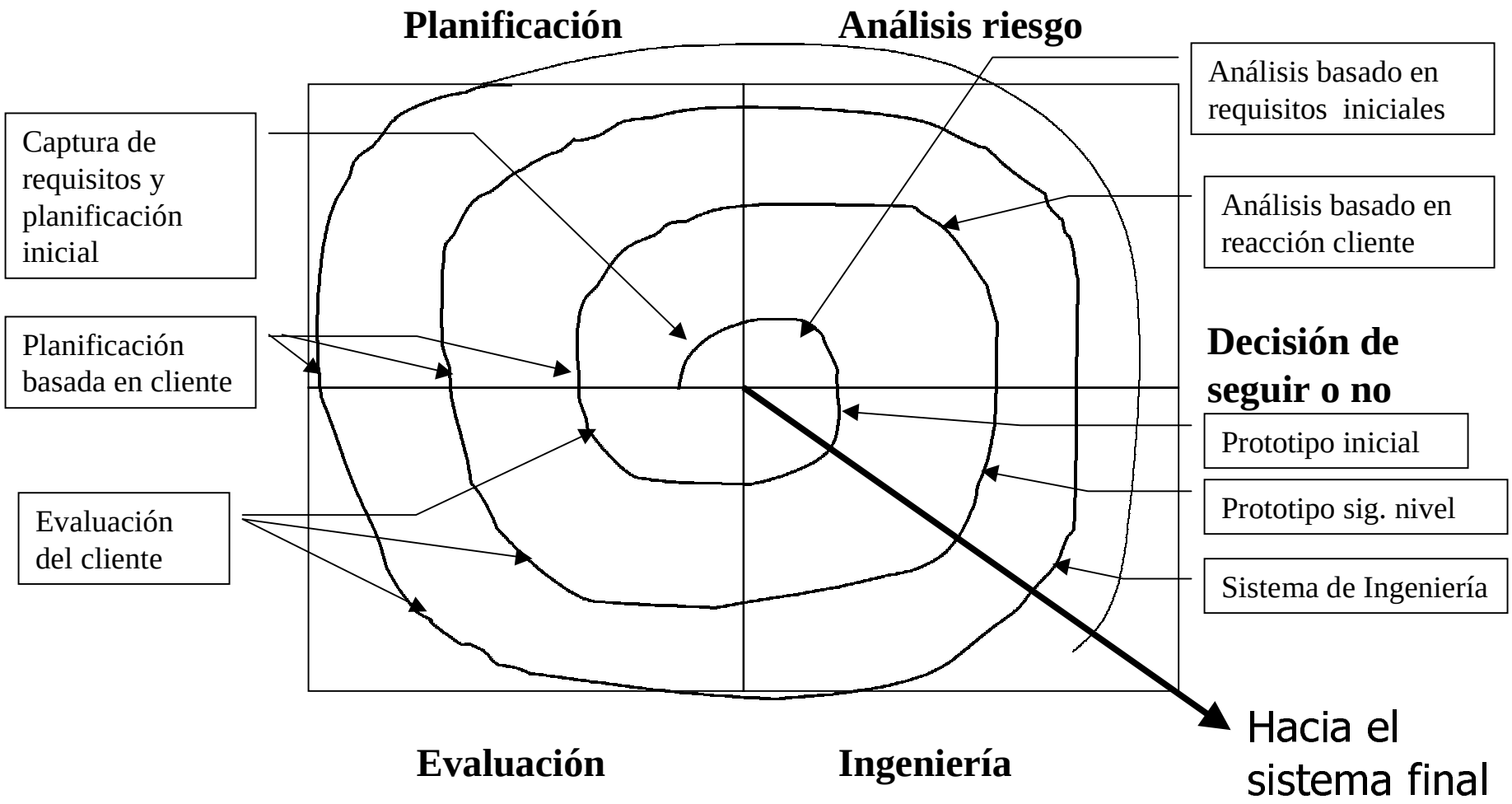


Modelo en Espiral

- Método iterativo que toma las mejores características del método clásico y el de prototipos
- Agrega un nuevo elemento: el **Análisis de Riesgo**

- Tiene cuatro actividades principales
- **Planificación**
- **Análisis de Riesgos**
- **Ingeniería**
- **Evaluación del Cliente**

El Modelo de Espiral



Conclusiones.

La metodología de análisis que se use debe depender de:

- Tipo de software a desarrollar.
- Experiencia en la metodología.
- Lenguaje en que se va a desarrollar

Conclusiones

Independientemente de la metodología de análisis, el ciclo de vida se debe elegir en base a las prioridades que se tengan.

Un método más lineal -como el de cascada- implica un desarrollo más rápido.

Pero un método menos lineal puede resultar más eficiente y confiable.