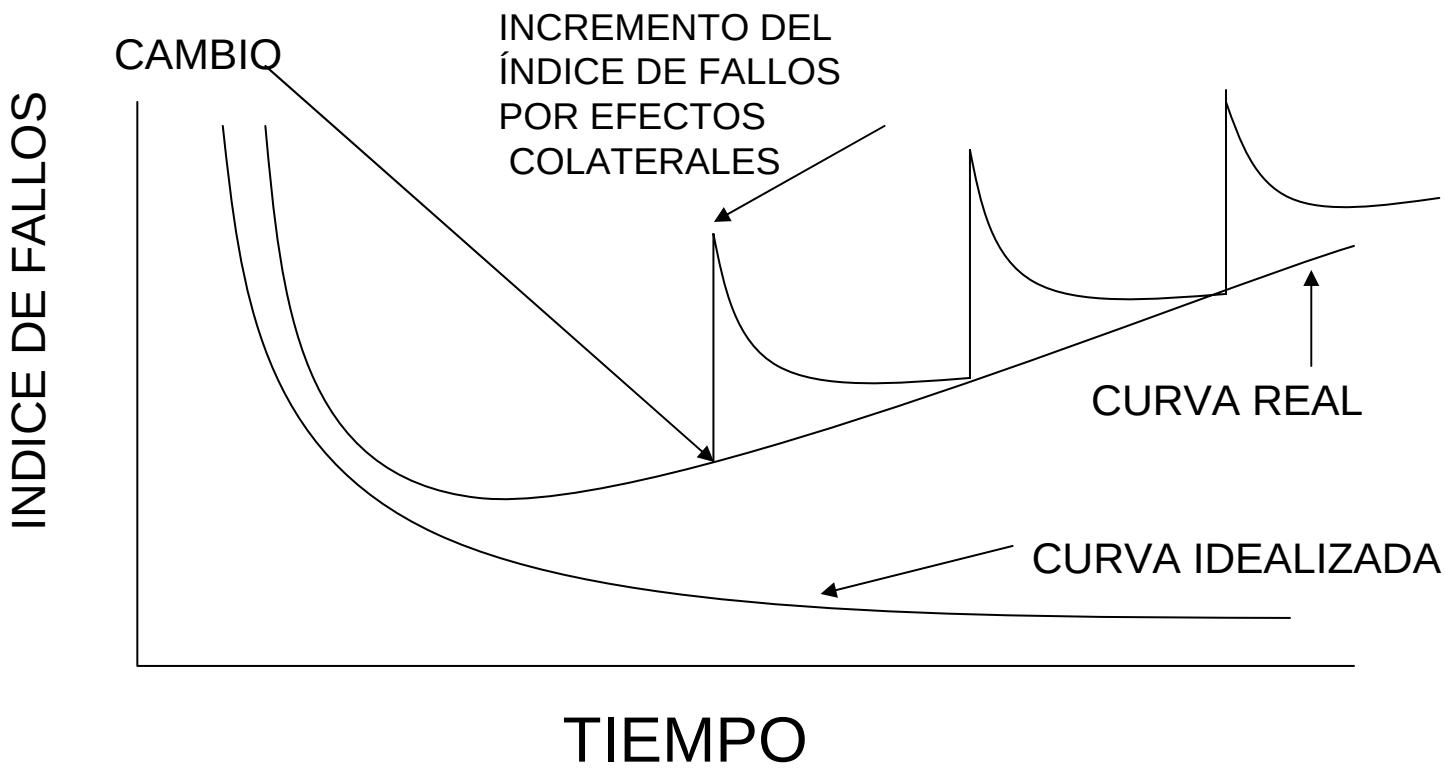
Departamento de Informática
Universidad de Valladolid
Campus de Segovia

TEMA 6: INTRODUCIÓN A LA INGENIERÍA DEL SOFTWARE

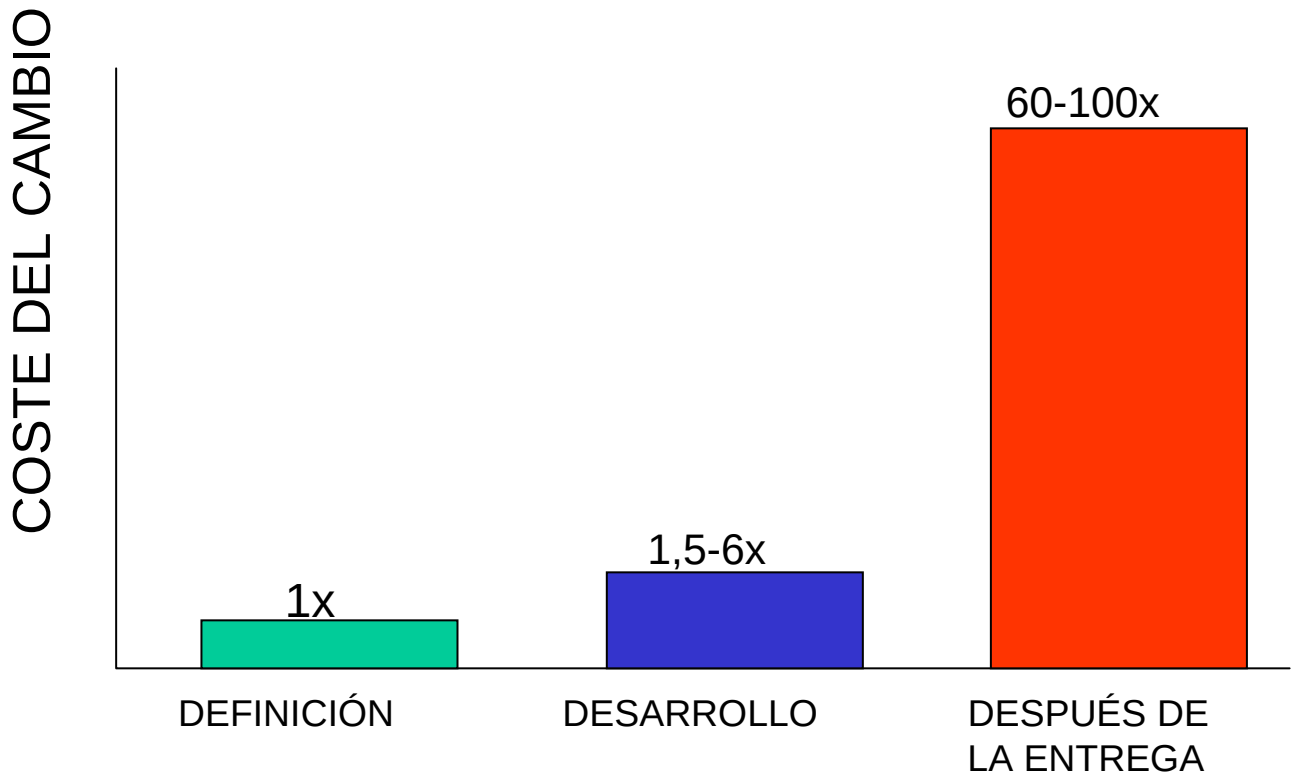
CARACTERÍSTICAS DEL SOFTWARE

- El software se desarrolla, no se fabrica.
- El software no se estropea, se deteriora debido a los cambios.
- El software, hoy por hoy se construye a medida. (no hay apenas reutilización).

CURVA DE FALLOS REAL E IDEALIZADA DEL SOFTWARE



EL IMPACTO DEL CAMBIO SEGÚN EL MOMENTO EN EL QUE SE INTRODUCEN



“ Cuanto más pronto se comience a escribir código, más se tardará en terminarlo”.

INGENIERÍA DEL SOFTWARE. DEFINICIÓN

- Marco de trabajo en el cuál se desarrolla **software de calidad**

- Comprende un proceso, un juego de métodos y un conjunto de herramientas.

EL PROCESO

- El desarrollo del software es un proceso de **aprendizaje iterativo**.

  Para poder definir los pasos que definen un proceso la I.S. Provee de:

  - **Métodos**: Indica como desarrollar el software.
  - **Herramientas**: Proporcionan un enfoque automático o semiautomático para el proceso y los métodos.

FASES GENERALES DEL PROCESO DE DESARROLLO

- **Fase de Definición** (¿Qué hacer?).

  (definición del problema=comunicación entre cliente y analista).
- **Fase de Desarrollo** (¿Cómo hacerlo?).
- **Fase de Mantenimiento** (Cambios).
  - Corrección
  - Adaptación
  - Mejora
  - Prevención

MODELOS

- Lineal Secuencial. Modelo de ciclo de vida básico.
- Construcción de prototipos.
- Modelos evolutivos. El modelo de la espiral.

EL MODELO DE CICLO DE VIDA BÁSICO

PLANIFICACIÓN

ANÁLISIS → DISEÑO → CÓDIGO → VALIDA-CIÓN

MANTENIMIENTO

EL MODELO DE CICLO DE VIDA BÁSICO. PLANIFICACIÓN

- Comunicación cliente-analista.
- Obtención y establecimiento de los requisitos.(Una buena definición evita problemas).
- Definición de las necesidades del producto.
- Valoración de los recursos humanos y técnicos que se necesitan.

EL MODELO DE CICLO DE VIDA BÁSICO. ANÁLISIS.

- Análisis de:
  - funciones que debe cumplir la aplicación.
  - La integración de los diferentes módulos que conforman la aplicación.
  - Determinación del sistema de pruebas que se van a emplear para validar la aplicación.
- Como resultado de este trabajo se redactan las especificaciones detalladas de funcionamiento: (Documentación).

EL MODELO DE CICLO DE VIDA BÁSICO. DISEÑO.

- Diseño del:
  - Conjunto de bloques que conforman la aplicación.
  - Se dividen por partes y se asignan a los grupos de programadores.
- Se elabora cada parte en lenguaje algorítmico.

EL MODELO DE CICLO DE VIDA BÁSICO. CODIFICACIÓN.

- Escritura de los algoritmos en el lenguaje de programación elegido.

(Este paso debe ser casi automático).

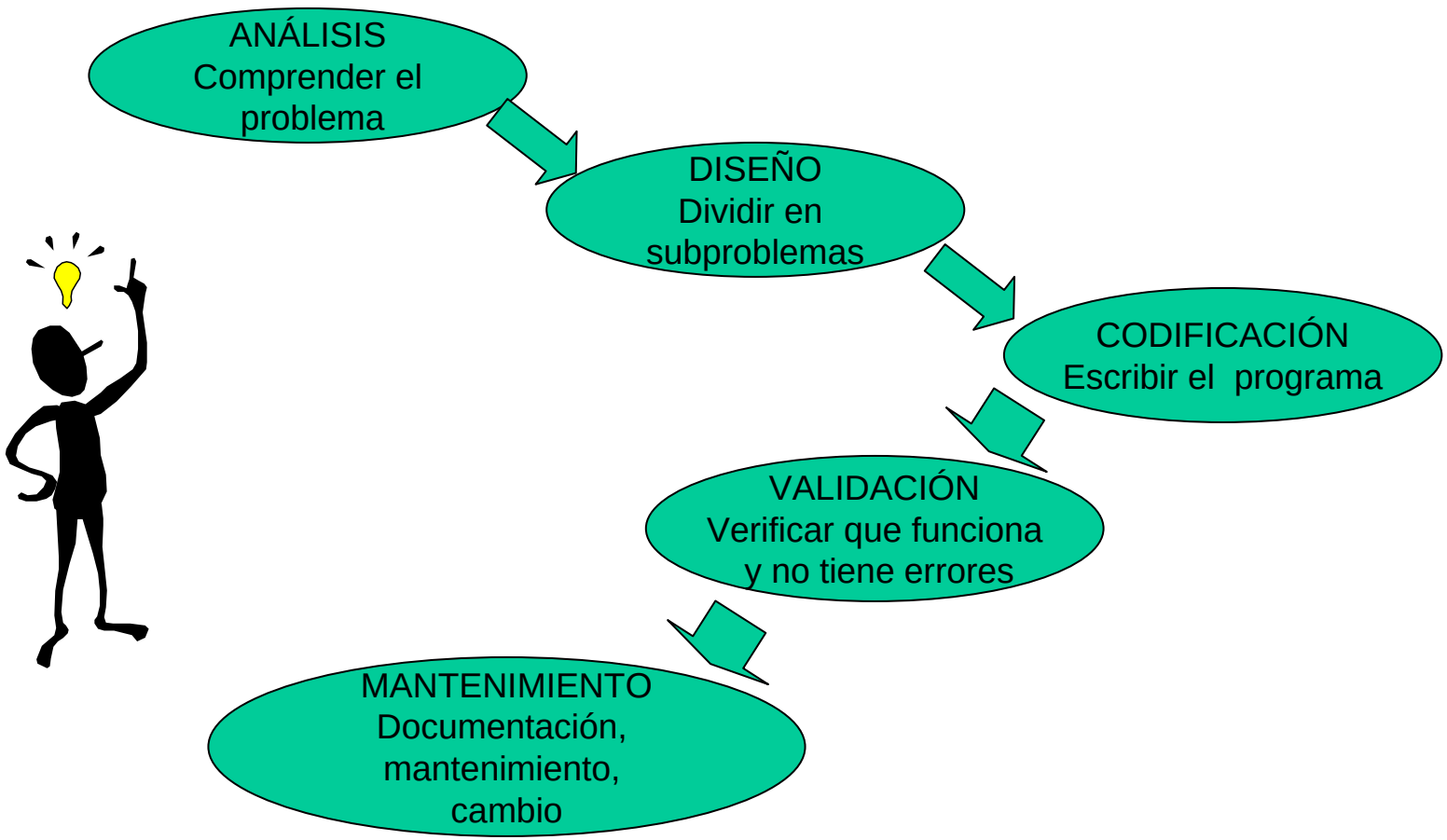
EL MODELO DE CICLO DE VIDA BÁSICO. VALIDACION.

- En esta fase se aplica el **sistema de pruebas** a:
  - cada módulo.
  - a las conexiones entre ellos (pruebas de integración).
  - y por último a toda la aplicación (pruebas de nivel superior).
- El propósito es buscar errores para la posterior **depuración** de la aplicación.

EL MODELO DE CICLO DE VIDA BÁSICO. MANTENIMIENTO.

- En esta fase se redacta la documentación tanto para su posterior mantenimiento como para el usuario.
- Se detectan y corrigen nuevos errores.
- Seguimiento de la aplicación de cara a posibles futuras modificaciones.

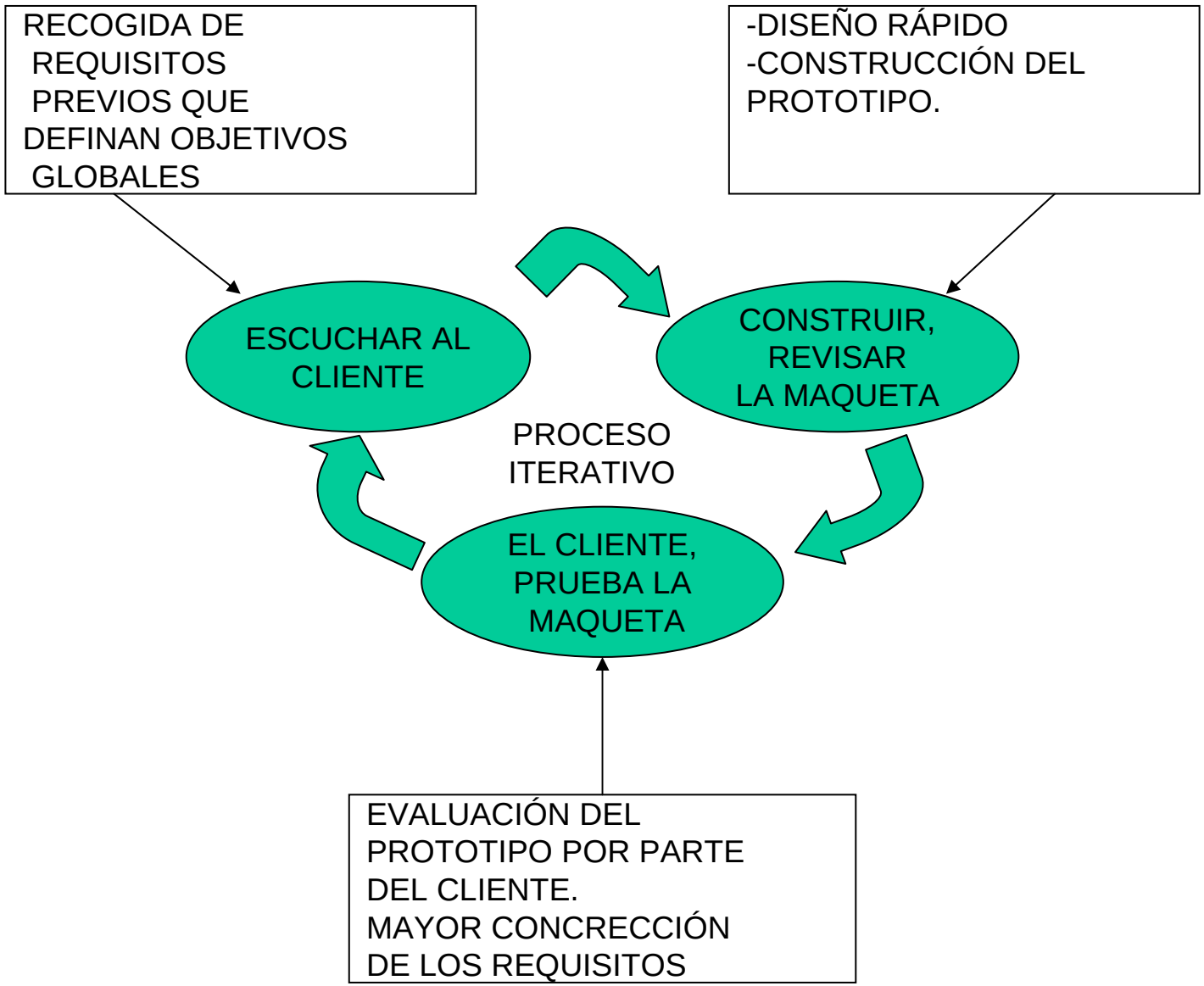
RESUMEN


ANÁLISIS
Comprender el problema
DISEÑO
Dividir en subproblemas
CODIFICACIÓN
Escribir el programa
VALIDACIÓN
Verificar que funciona y no tiene errores
MANTENIMIENTO
Documentación, mantenimiento, cambio

INCONVENIENTES DEL MODELO

- Los proyectos raras veces siguen una evolución secuencial.
- No todos los requisitos son expuestos, al principio, de forma explícita como requiere este modelo.
- El cliente debe tener paciencia, ya que la aplicación sólo estará disponible en un estado muy avanzado del proyecto.

CONSTRUCCIÓN DE PROTOTIPOS



INCONVENIENTES DEL MODELO

- El cliente puede pensar que el prototipo es una versión acabada.
- Las herramientas elegidas pueden ser inadecuadas.

- **LA CLAVE DEL ÉXITO DE ESTE MODELO CONSISTE EN DEFINIR BIEN, DESDE EL PRINCIPIO,LAS REGLAS DEL JUEGO.**

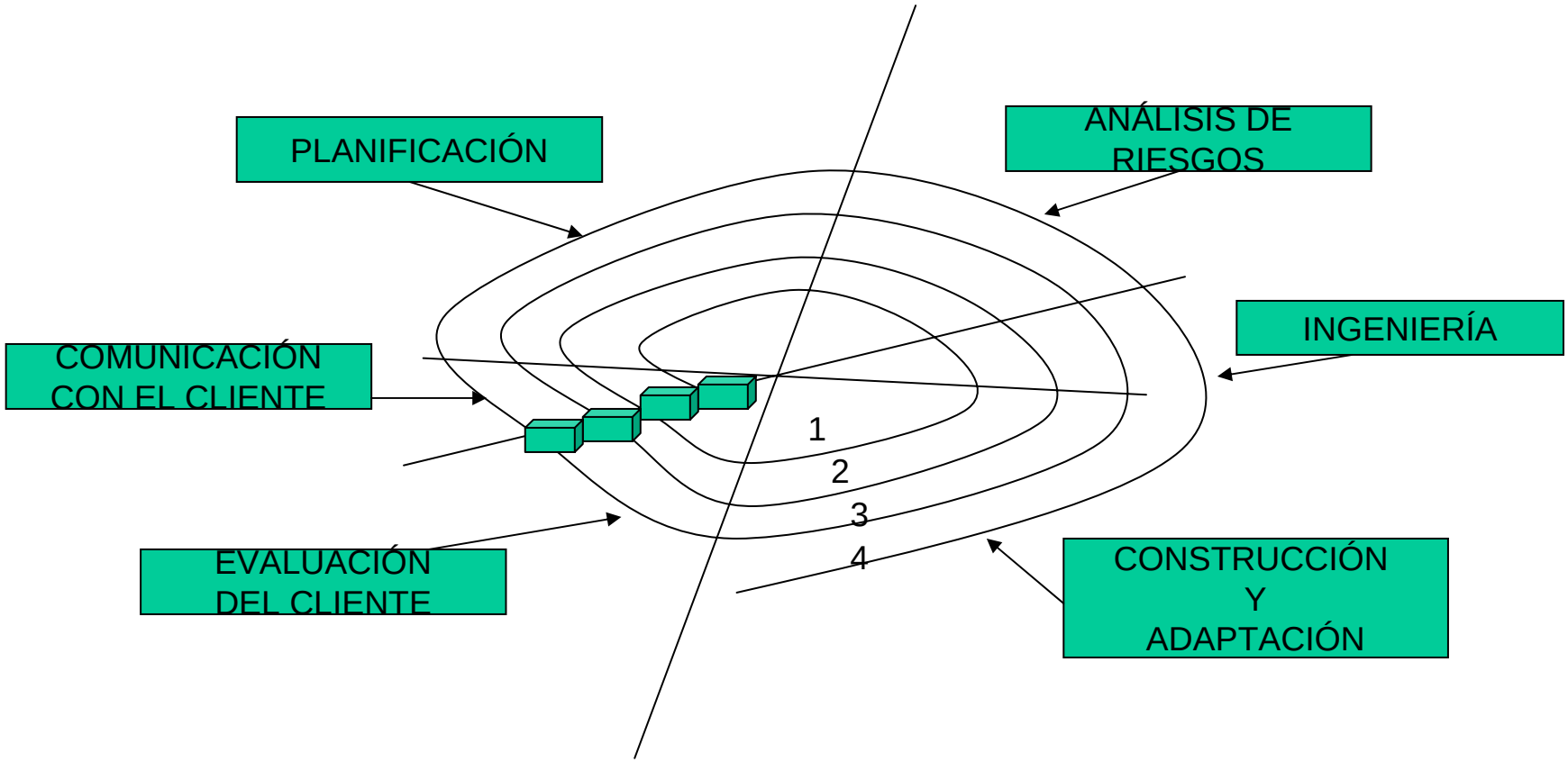
(DIALOGO CLIENTE-PROGRAMADOR).

MODELOS EVOLUTIVOS. EL MODELO ESPIRAL

- Propuestos originalmente por Boehm. [BOE 88]*
- Modelo evolutivo que conjuga:
  - la naturaleza iterativa de construcción de prototipos.
  - Aspectos controlados y sistemáticos del modelo lineal secuencial.
- Primeras iteraciones: modelo en papel o prototipo.
- Últimas iteraciones: Versión más completa de la aplicación.

*Boehm, B. “A Spiral Model for Software Development and Enhancement”, Computer, vol. 21, nº 5 Mayo 1988.

EL MODELO ESPIRAL



1. DESARROLLO DE CONCEPTOS
2. DESARROLLO DE NUEVOS PRODUCTOS
3. MEJORA DE PRODUCTOS
4. MANTENIMIENTO DE PRODUCTOS

CALIDAD DEL SOFTWARE. FIABILIDAD Y CORRECCIÓN

- **CALIDAD DE SOFTWARE:** Esto lleva asociado que la implementación (programa) sea correcta y fiable.

CORRECCIÓN

- **Un algoritmo es correcto parcialmente** si se garantiza que para el juego de datos contemplados en la especificación el algoritmo llega a un conjunto de resultados cuando para, aunque no podamos establecer si para, también establecidos en las especificaciones.

- **Un algoritmo es totalmente correcto** si lo es parcialmente y además sabemos que para en un tiempo finito.

- La demostración de la corrección de un algoritmo se establece a través de la **verificación de algoritmos**.

VERIFICACIÓN vs PRUEBAS DE PROGRAMAS

- PRUEBAS DE PROGRAMAS:
  - Comprobación del buen funcionamiento de un programa para unos pocos juegos de datos.
  - Esto no descarta que el programa este libre de errores.

- La verificación establece de una manera formal la corrección total pero sin embargo es más compleja de elaborar que la prueba de programas.

FIABILIDAD DEL SOFTWARE

- “Se define como la probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado y durante un tiempo específico” [MUS 87]*.
- Se define como fallo cualquier falta de concordancia con los requisitos del software:
  - Leves
  - Catastróficos.
- **¡OJO!.** La corrección de un fallo puede, a su vez, introducir nuevos errores.

*Musa. J.D, A. Iannino y K. Okumoto. *Engineering and Managing Software with Reliability Measures*. Mc Graw-Hill 1987.