Introducción a la Ingeniería del Software

Algunas definiciones:

Sistema

> "Un conjunto de elementos que operan conjuntamente para cumplir un propósito, meta u objetivo común"

Información

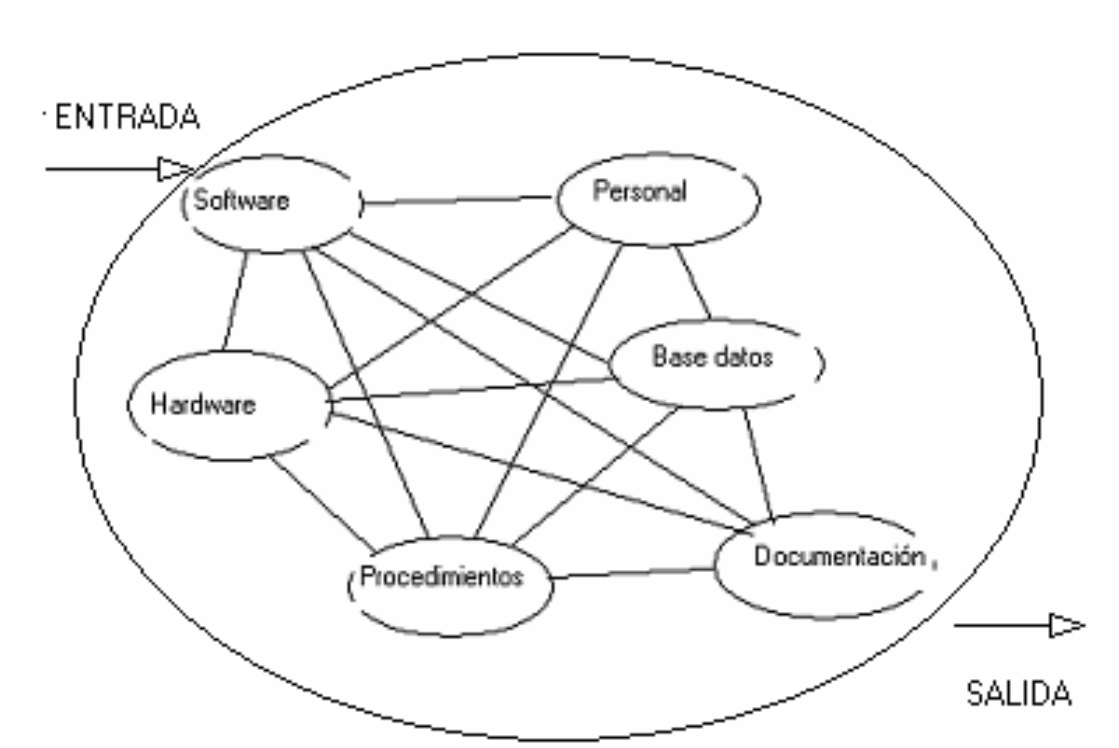
> "Todo elemento de conocimiento susceptible de ser tratado y utilizado"

Sistema de información

> "Sistema de información es un conjunto de procedimientos y medios materiales y humanos que trabajando de una forma ordenada recopilan, almacenan, tratan y recuperan datos para reducir la incertidumbre en la toma de decisiones al suministrar información en tiempo oportuno"

Sistema informático.

> "Un conjunto (u ordenación) de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento automático de información"



- ✓**Hardware**.- Dispositivos electrónicos y electromecánicos que proporcionan la capacidad de computación y dispositivos electromecánicos que proporcionan las funciones del mundo exterior.

- ✓**Personal**.- Individuos que son usuarios y operadores de software-hardware.

- ✓**Base Datos..-** Colección de información (grande) que accede mediante software y que es una parte integral del funcionamiento del sistema.

- ✓**Documentación**.- Manuales impresos e información descriptiva que explica el uso y/o la operación del sistema.

- ✓**Procedimientos**.- Pasos que definen el uso específico de cada elemento del sistema o el contexto procedimental en que reside el sistema.

- ✓**Software.-** Programa de computadora, estructura de datos y la documentación asociada para realizar algún método lógico, procedimiento o control requerido.

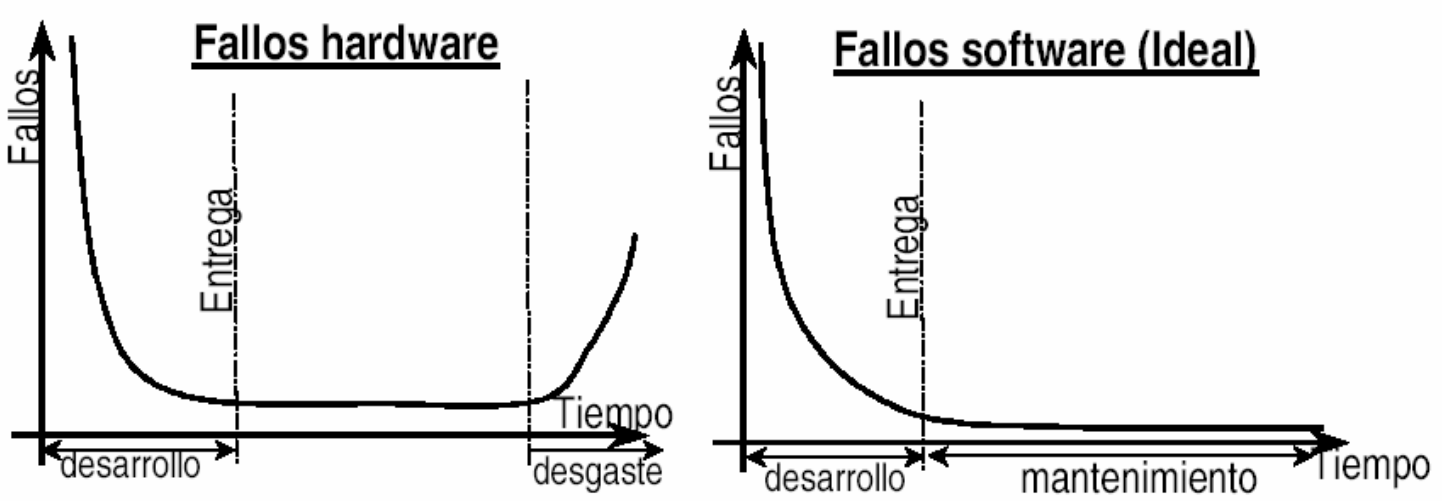
El software son tres elementos:

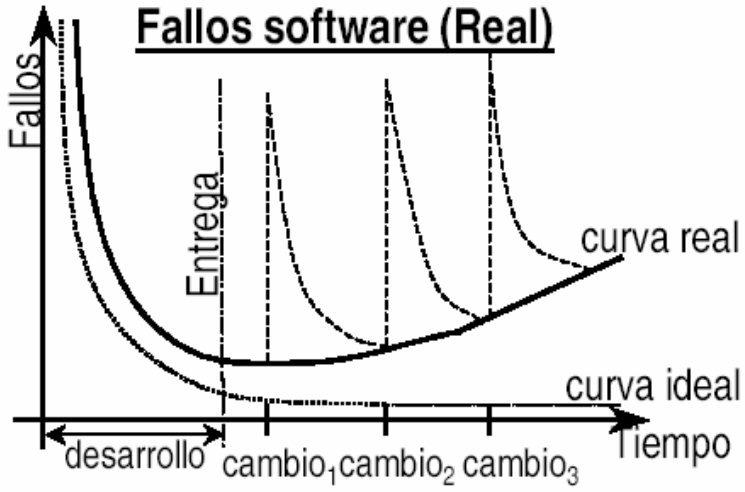
1. **Instrucciones** que, cuando se ejecutan, proporcionan la funcionalidad deseada.
2. **Estructuras de datos** que facilitan a las instrucciones manipular adecuadamente la información.
3. **Documentos** que describen el desarrollo, uso, instalación y mantenimiento de los programas.

Características del Software

Para poder comprender que es el software es importante examinar las características que lo diferencian de las otras cosas que los hombres pueden construir. El software es un elemento del sistema que es lógico en lugar de físico. El software tiene unas características considerables distintivas a las del hardware:

- ✓ **El** software **se desarrolla, no se fabrica en un sentido clásico**
- ✓ **El** software **no se estropea**
- ✓ **La mayoría del** software **se construye a medida, en vez de ensamblar componentes existentes. (**Objetivo-→ conseguir la reutilización (reusabilidad))





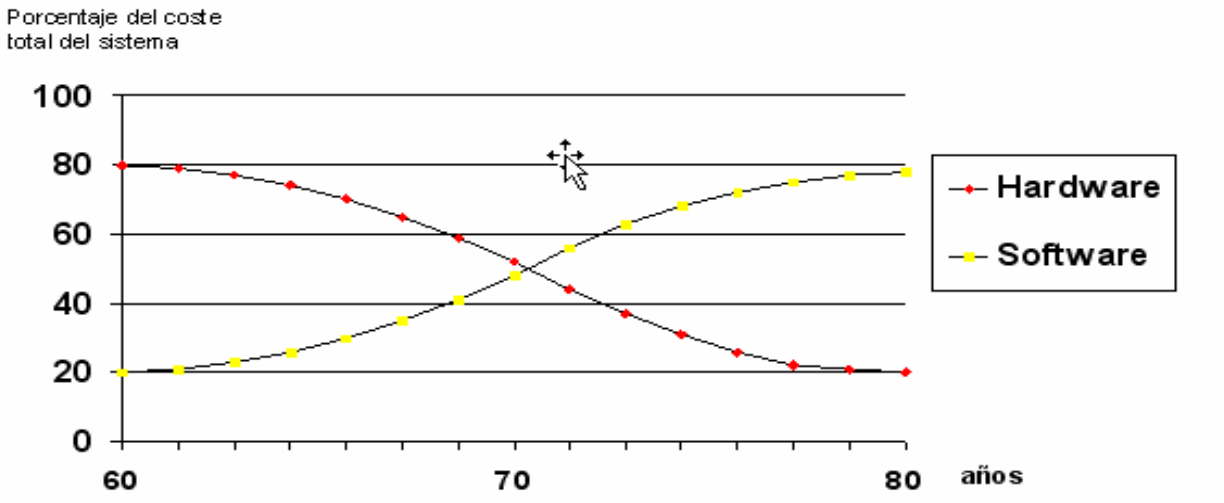
Problemas del software (“*crisis del software*”)

Estos problemas son causados por las propias características del software y por los errores cometidos por quienes intervienen en su producción. En 1968 se acuño el término *crisis del software* para definir la situación.

1. **La planificación y la estimación de costes son muy imprecisas.**
2. **La productividad es baja.**
3. **La calidad es mala.**
4. **El cliente queda insatisfecho.**

Algunas causas

- ✓Naturaleza “no física” de la programación.
- ✓Problemas derivados de la intervención de grupos.
- ✓Problemas de comunicación con los clientes.
- ✓Poco esfuerzo en el análisis y el diseño.
- ✓Herramientas comerciales poco adecuadas.
- ✓Problemas de gestión.
- ✓*Planificaciones optimistas, plantillas poco cualificadas...*
- ✓A veces, el software debe “solucionar” los problemas del sistema global.
- ✓Difusión limitada de las nuevas técnicas, métodos y herramientas.
- ✓...industria pendiente de su ‘revolución industrial’ ¿???



Definición de Ingeniería del Software

No existe una fórmula mágica para solucionar estos problemas, pero combinando métodos aplicables a cada una de las fases del desarrollo de software, construyendo herramientas para automatizar estos métodos, utilizando técnicas para garantizar la calidad de los productos desarrollados y coordinando todas las personas involucradas en el desarrollo de un proyecto, podremos avanzar mucho en la solución de estos problemas. De ello se encarga la disciplina llamada ***Ingeniería del Software.***

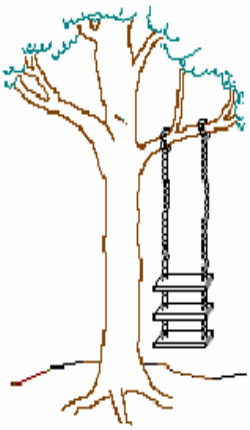
Una de las primeras definiciones que se dio de la ingeniería del software es la siguiente:

> **"El establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico, que sea fiable y funcione de manera eficiente sobre máquinas reales"**

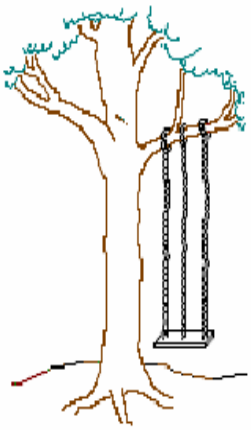
- ✓Los ***métodos*** indican cómo construir técnicamente el software, y abarcan una amplia serie de tareas que incluyen la planificación y estimación de proyectos, el análisis de requisitos, el diseño de estructuras de datos, programas y procedimientos, la codificación, las pruebas y el mantenimiento.

- ✓Las ***herramientas*** proporcionan un soporte automático o semiautomático para utilizar los métodos. Estas herramientas se denominan CASE (Computer Assisted Software Engineering).

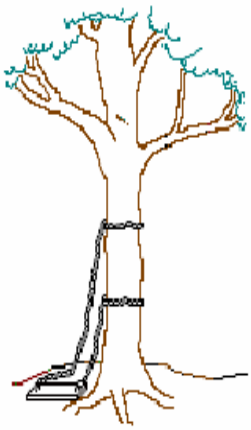
- ✓Los ***procedimientos*** definen la secuencia en que se aplican los métodos, los documentos que se requieren, los controles que permiten asegurar la calidad y las directrices que permiten a los gestores evaluar los progresos.



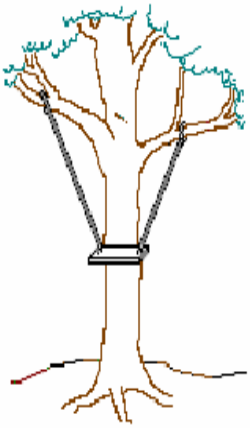
1. Lo que el director desea.



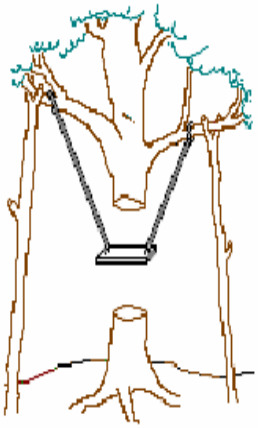
2. Como lo define el director de proyecto.



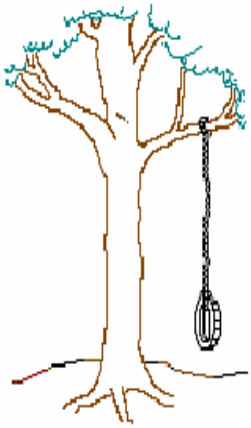
3. Como se diseña el Sistema.



4. Como lo desarrolla el programador.



5. Como se ha realizado la instalación.



6. Lo que el usuario quería.

✓ Se trata de definir una disciplina que garantice la producción sistemática y mantenimiento de los productos software desarrollados en al plazo fijado y dentro del coste estimado.

Diferencias con otras ingenierías:

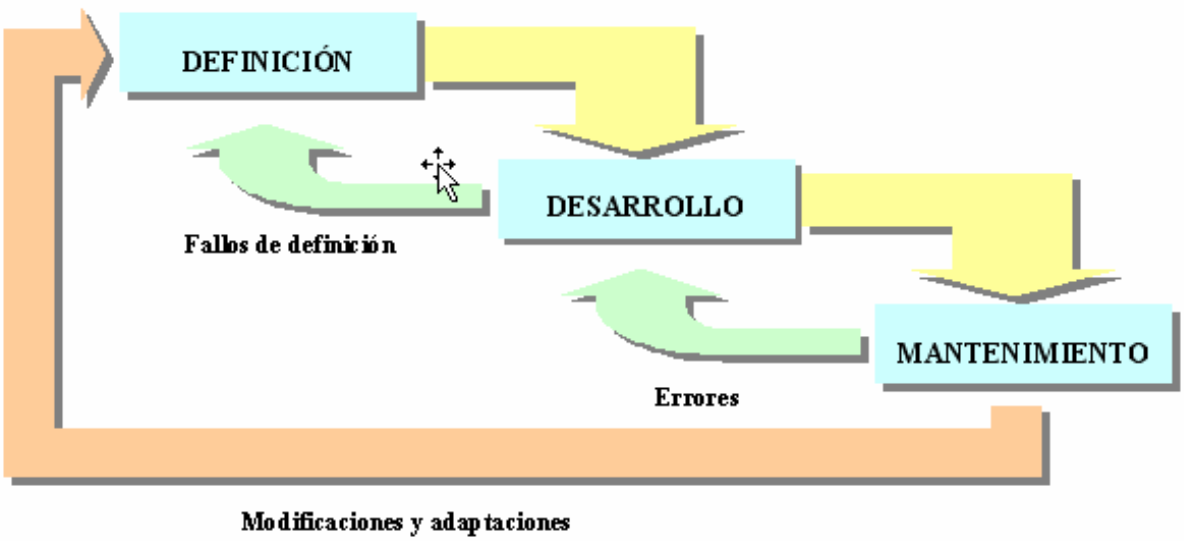
Ingenierías:

✓"Construyen instrumentos (sistemas artificiales) que imitan, aumentan, ayudan, facilitan o sustituyen capacidades físicas del ser humano".

Ingeniería del Software:

✓"Construyen instrumentos que imitan, aumentan, ayudan, facilitan o sustituyen capacidades psíquicas del ser humano (imitan y ayudan al ser humano en su capacidad de resolución de problemas)".

Visión genérica de la Ingeniería del Software.



Definición. ¿Qué? ¿Cuánto? ¿Cuando?

1. Análisis del sistema.
    a. Establecer el ámbito del software.
2. Análisis de requisitos del sistema software.
    a. Definición detallada de la función del software.
3. Planificación.
    a. Análisis de riesgos.
    b. Asignación de recursos.
    c. Definición de tareas.
    d. Estimación de costes.

La fase de definición se centra en el qué.

✓qué información es la que tiene que ser procesada.
✓qué función y rendimiento son los que se esperan.
✓qué restricciones de diseño existen.
✓qué interfaces deben utilizarse.
✓qué lenguaje de programación, sistema operativo y soporte hardware van a ser utilizados.
✓qué criterios de validación se necesitan para conseguir que el sistema final sea correcto.

Desarrollo. ¿Cómo?

1. Diseño.
    a. Arquitectura de la aplicación.
    b. Estructura de los datos.
    c. Estructura interna de los programas.
    d. Diseño de las interfaces.
2. Codificación.
3. Pruebas.

La fase de definición se centra en el cómo.

✓cómo ha de ser la arquitectura de la aplicación.
✓cómo han de ser las estructuras de datos.
✓cómo han de implementarse los detalles procedimentales de los módulos.
✓cómo van a ser las interfaces.
✓cómo ha de traducirse el diseño a un lenguaje de programación.
✓cómo van a realizarse las pruebas.

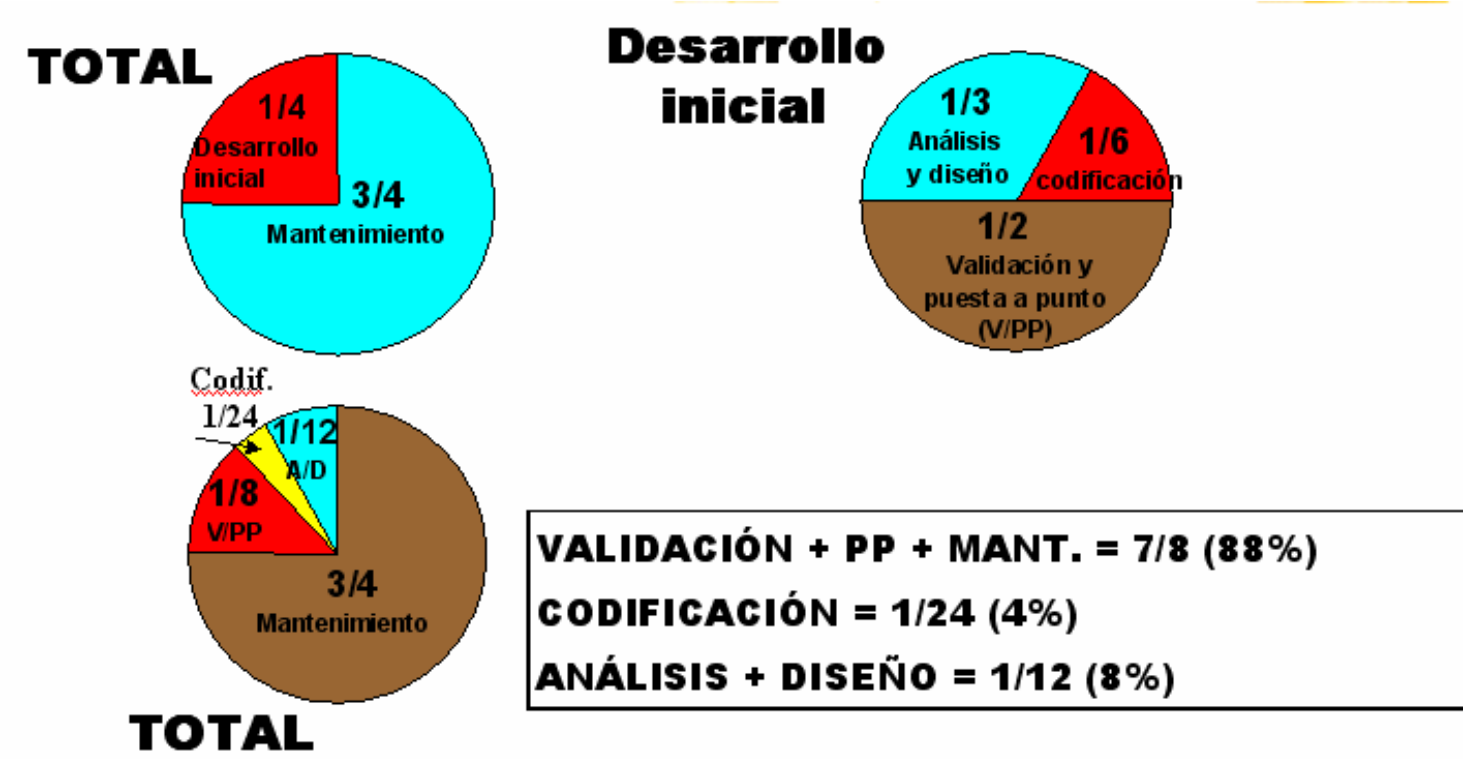
Mantenimiento. El cambio.

a. Corrección de errores.
b. Cambios en el entorno.
c. Cambios en los requisitos.

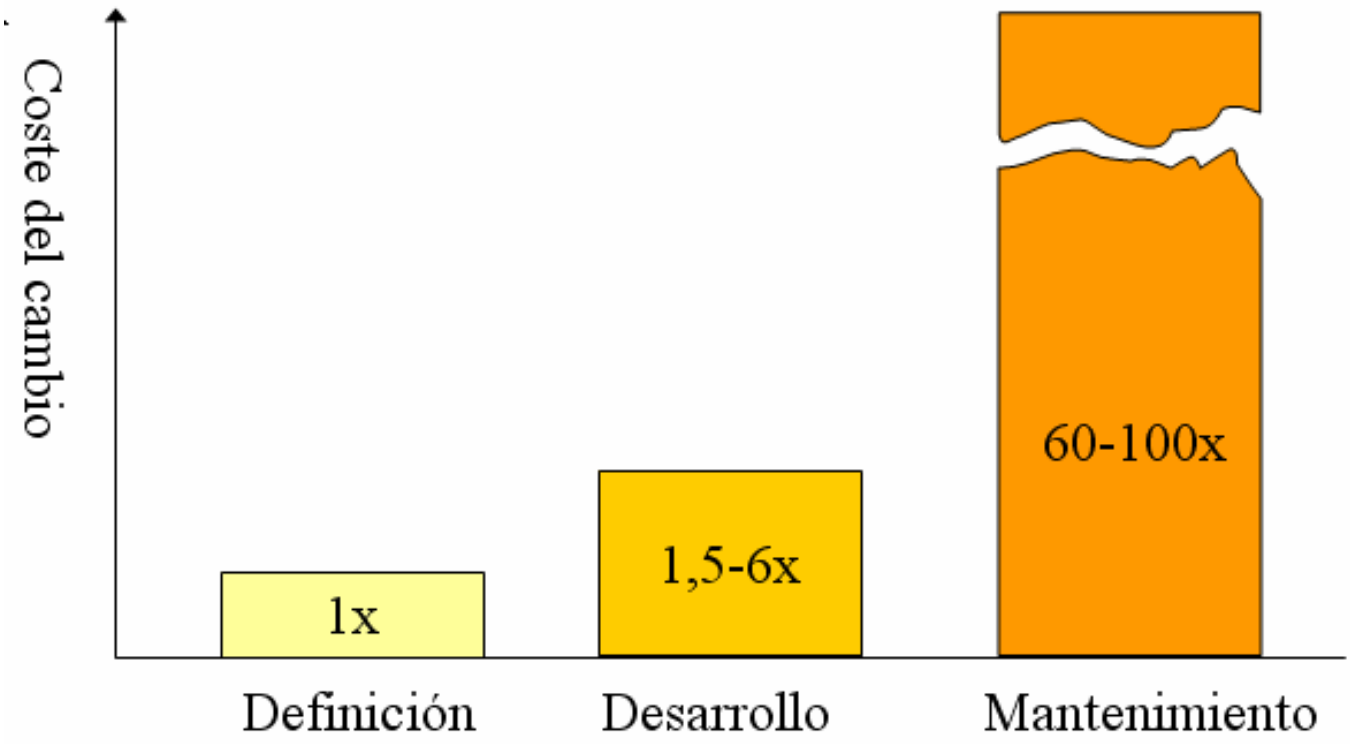
La fase de mantenimiento vuelve a aplicar los pasos de las fases de definición y de desarrollo, pero en el contexto de un software ya existente y en funcionamiento.

✓Comienza una vez construido el sistema, cuando se empieza a utilizar.
✓Se centra en el ***cambio***.
✓El software es sometido a reparaciones y modificaciones cada vez que se detecta un fallo o se necesita cubrir una nueva necesidad de los usuarios.
✓En esta fase recae el mayor porcentaje del coste de un sistema.

Coste del software



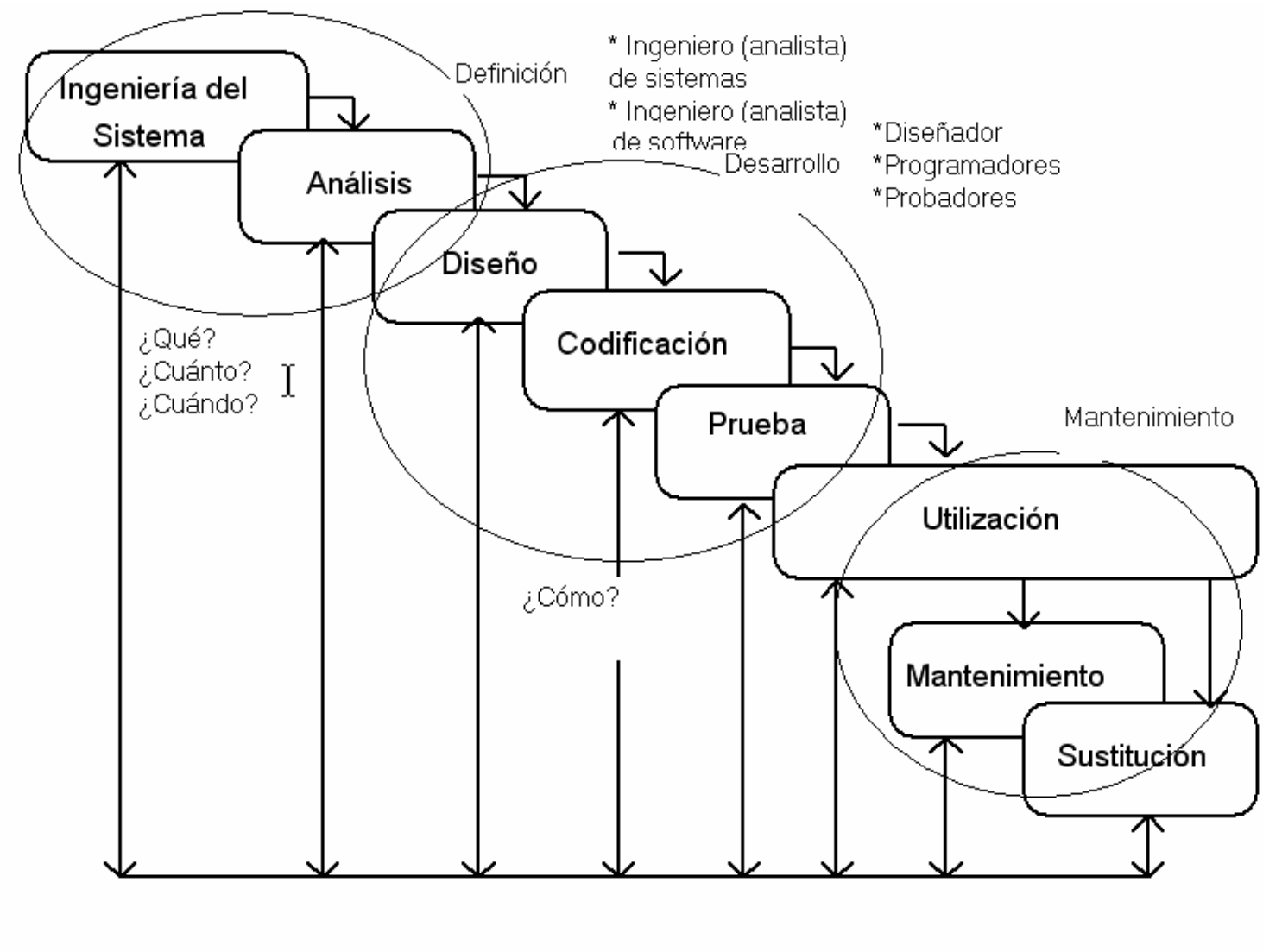
Impacto del cambio

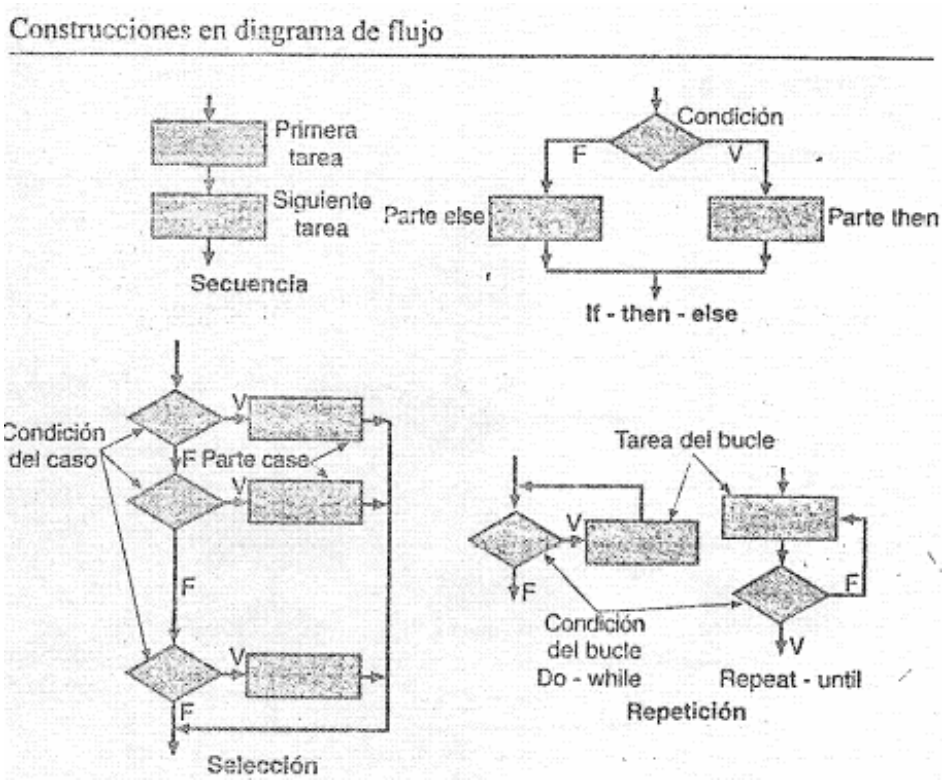


Procedimientos de la Ingeniería del Software

La Ingeniería del Software esta compuesta por una serie de pasos que abarcan los métodos, las herramientas y los procedimientos. Estos pasos se denominan frecuentemente **Paradigmas de Ingeniería del Software**

Ciclo de Vida Clásico o modelo en cascada





Diagramas Nassi-Scheneiderman- Diagramas NS

Identificar el mayor y menor de dos números

<table>
<tr><td colspan="3">Inicio</td></tr>
<tr><td colspan="3">Entero: n1, n2</td></tr>
<tr><td colspan="3">Leer n1, n2</td></tr>
<tr><td>Si</td><td colspan="2">n1 = n2
No</td></tr>
<tr><td rowspan="3">Escribir
"Números iguales"</td><td>Si</td><td>n1 > n2
No</td></tr>
<tr><td>Escribir
n1, "Mayor"</td><td>Escribir
n2, "Mayor"</td></tr>
<tr><td>Escribir
n2, "Menor"</td><td>Escribir
n1, "Menor"</td></tr>
<tr><td colspan="3">Fin algoritmo</td></tr>
</table>

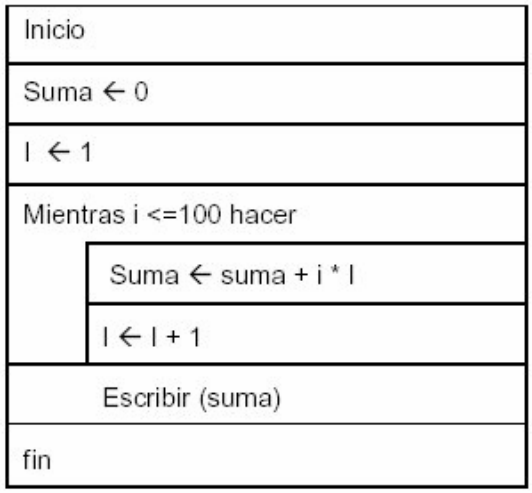
Pseudocódigo

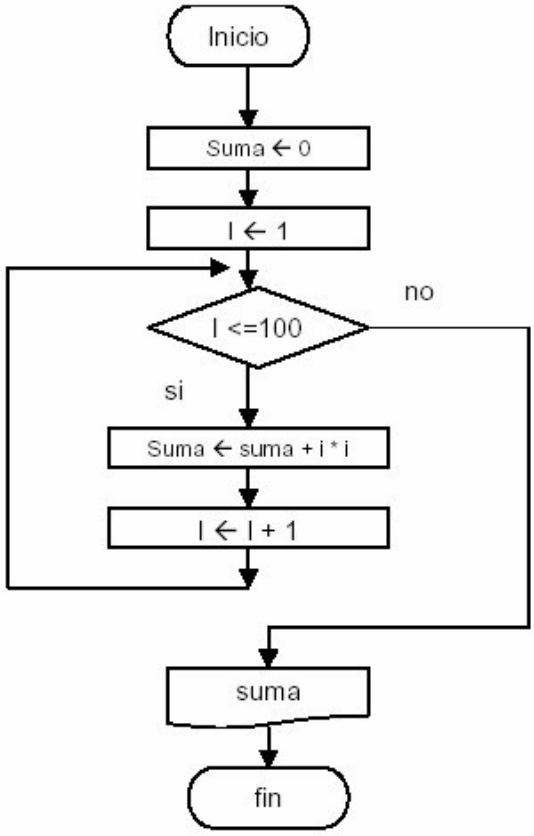
Calcular la suma de los cuadrados de los primeros 100 números enteros y escribir el resultado.

Hagamos el algoritmo:

```
Inicio
  Suma ← 0
  I ← 1
  Mientras I < = 100 hacer
                Suma ← suma + I * I
                I ← I + 1
  Fin_mientras
  Escribir (suma)
Fin
```

Flujograma:





Modelo en cascada. Críticas

- ✓ Carácter secuencial. El ciclo de vida clásico requiere la definición inicial de todos los requisitos y no es fácil acomodar en él las incertidumbres que suelen existir al comienzo de todos los proyectos.
- ✓ 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 la validación y mantenimiento recae, en su mayor parte, sobre el código fuente.
- ✓ El software va deteriorándose y resulta cada vez más difícil de mantener.

Los paradigmas alternativos al "**modelo en cascada**" buscan esencialmente corregir el carácter secuencial de este. Introduciendo iteraciones más rápidas entre el análisis y la codificación

Una frase conocida….

"***Caminar sobre las aguas y desarrollar programas a partir de especificaciones es fácil, …….si ambas están congeladas***"
(E.V. Berard).

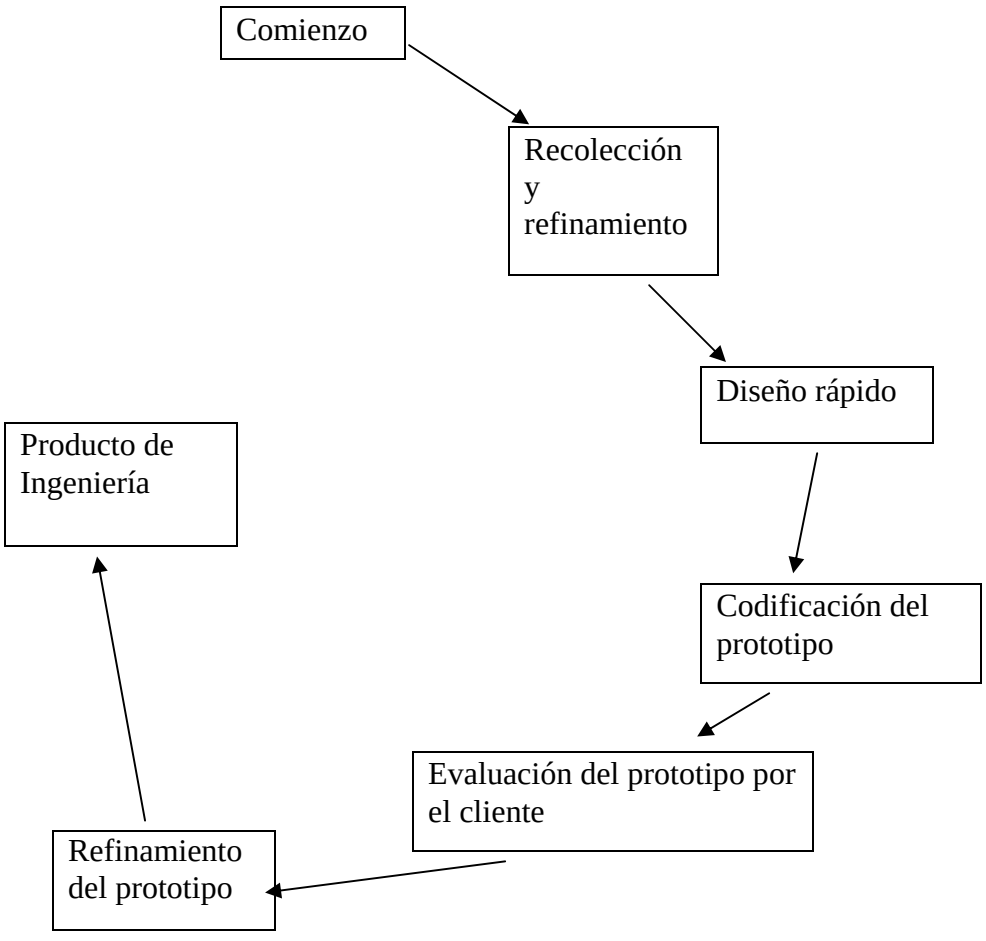
Construcción de prototipos

☞♦ un proceso que facilita al programador la creación de un modelo de software a construir. El modelo tomará una de las siguientes tres formas:

- ✓ Un prototipo en papel que describa la interacción hombre – maquina de forma que facilite al usuario la comprensión de cómo se producirá tal interacción.

- ✓ Un prototipo que implemente algunos subconjuntos de la función requerida del programa deseado.

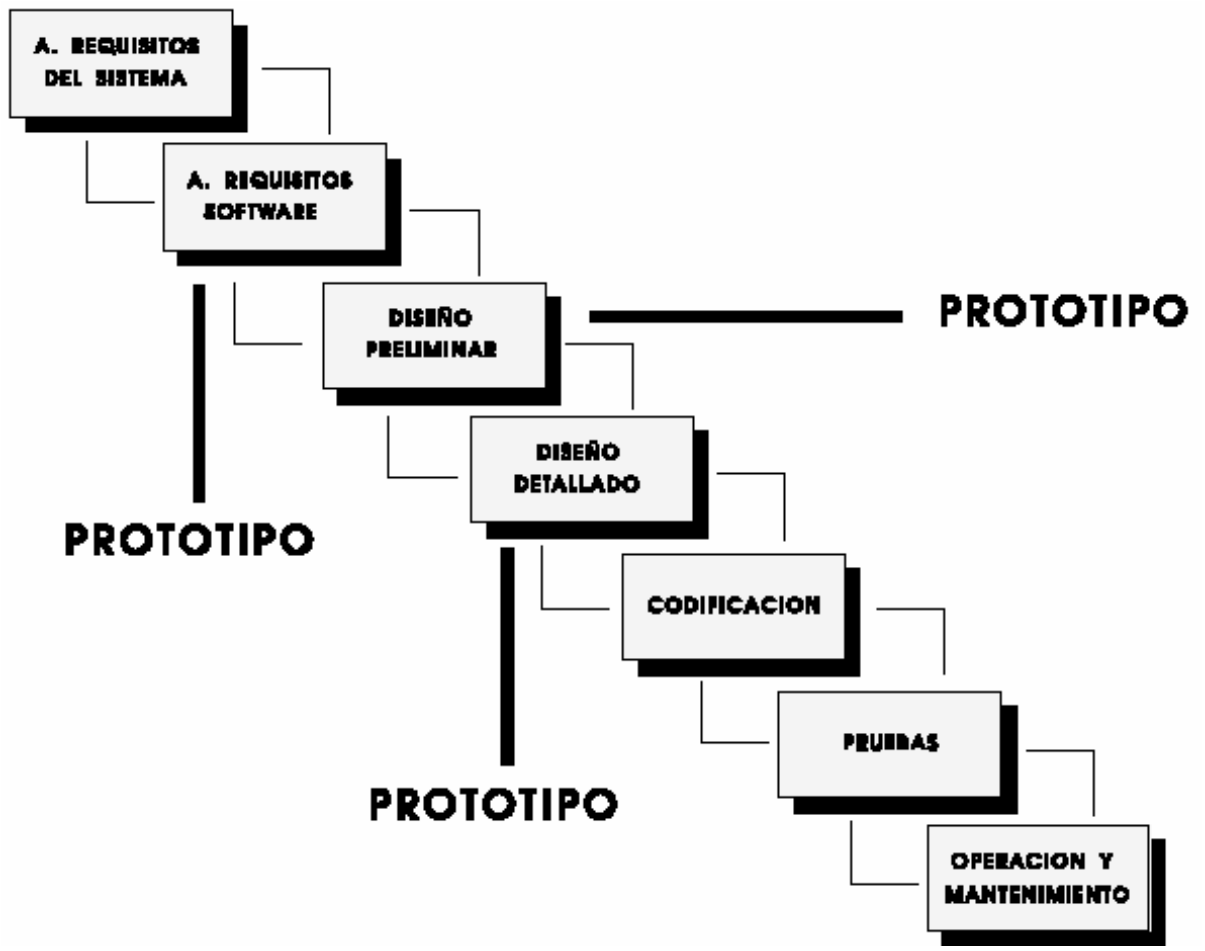
- ✓Un programa existente que ejecute parte de toda la función deseada pero que tenga otras características que deba ser mejorada en el nuevo trabajo de desarrollo.



Identifican todos los requisitos conocidos y perfilan las áreas en donde sea necesario una mayor definición.

Representación de los aspectos del Software visibles al usuario

MODELO DE PROTOTIPO

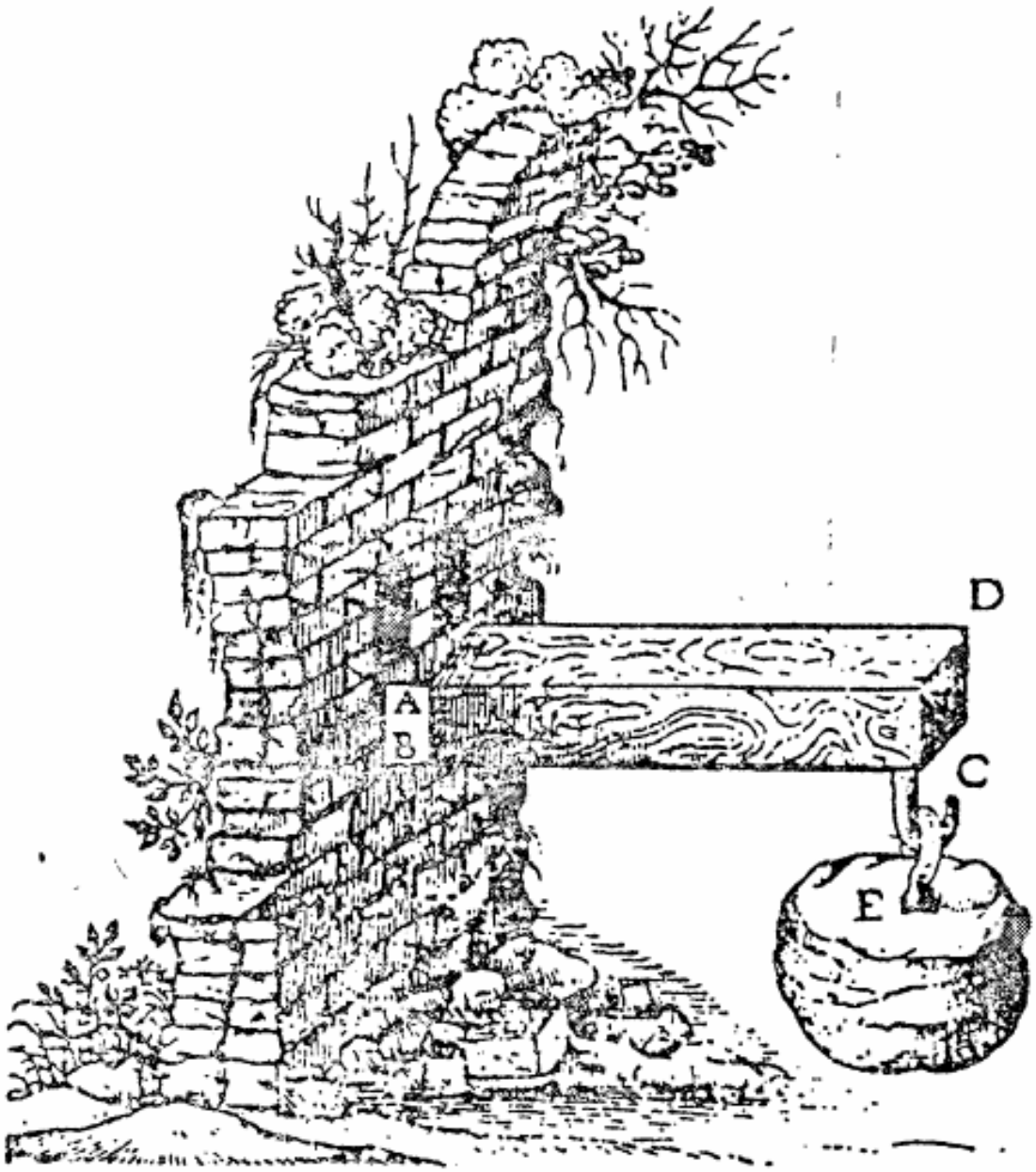


- ✓ No modifica el flujo del ciclo de vida
- ✓ Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios
- ✓ Reduce costos y aumenta la probabilidad de éxito
- ✓ Exige disponer de las herramientas adecuadas
- ✓ No presenta calidad ni robustez
- ✓ Una vez identificados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

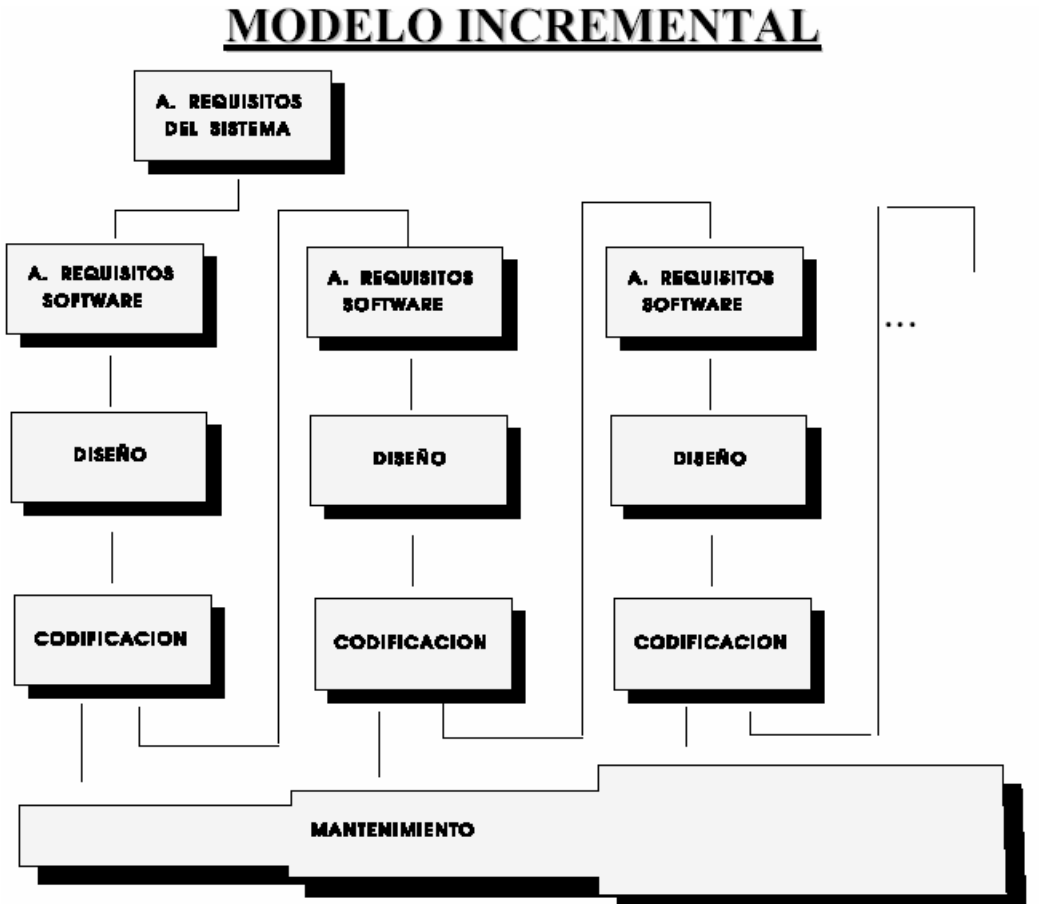
La representación

Dibujo tomado de un trabajo de Galileo. Se utiliza junto como suplemento de un texto que describe el método de calcular la resistencia de una viga. Es un ejemplo de una buena representación de una especificación.

Representación de una especificación (Fuente: *Discorsi e Dimostrazioni Matematiche ubtorno a due nuovo science*, Leyden, 1638; de J. D. Bernal, Science in History, London, Watts, 1969.)

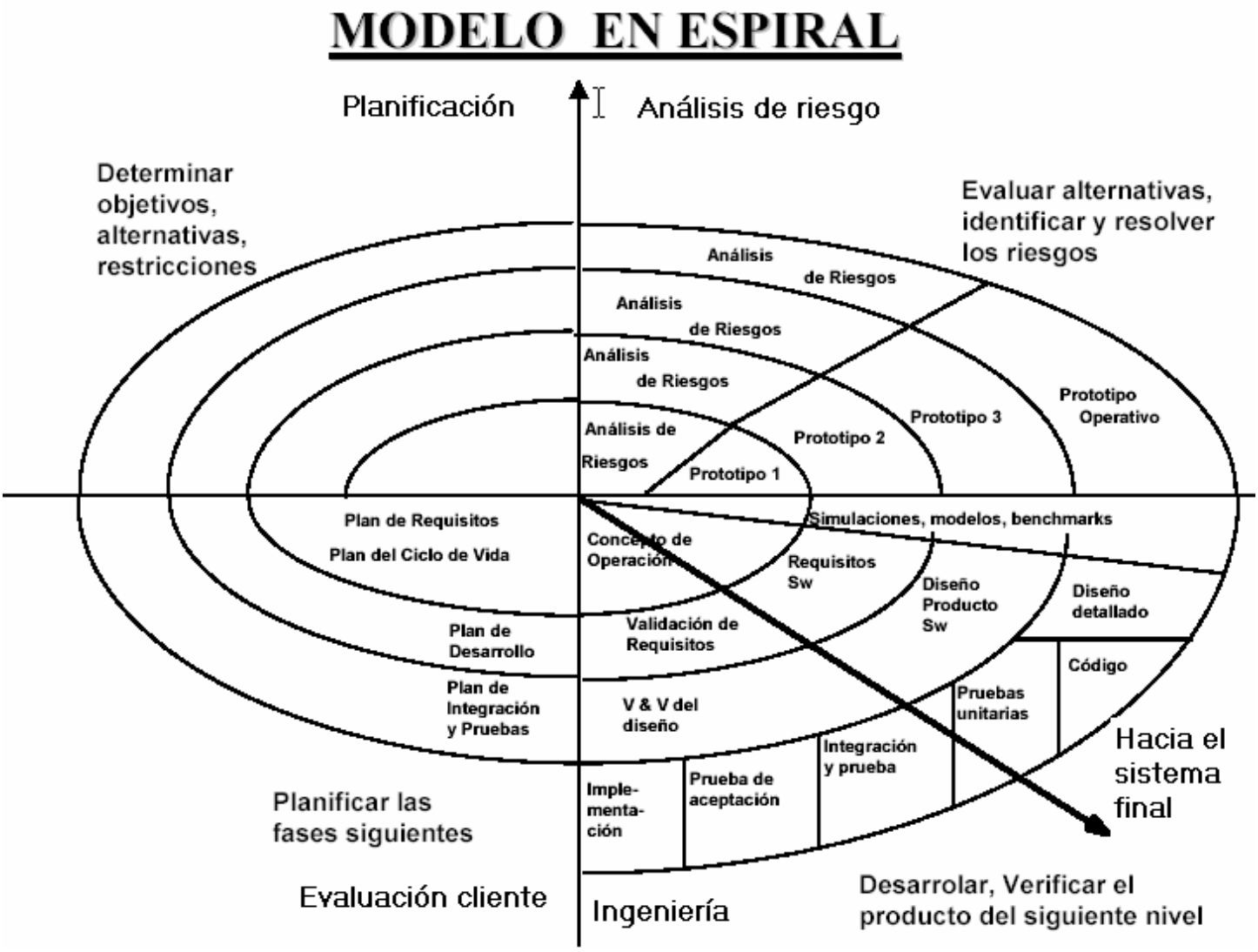


MODELO INCREMENTAL



- ✓ 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

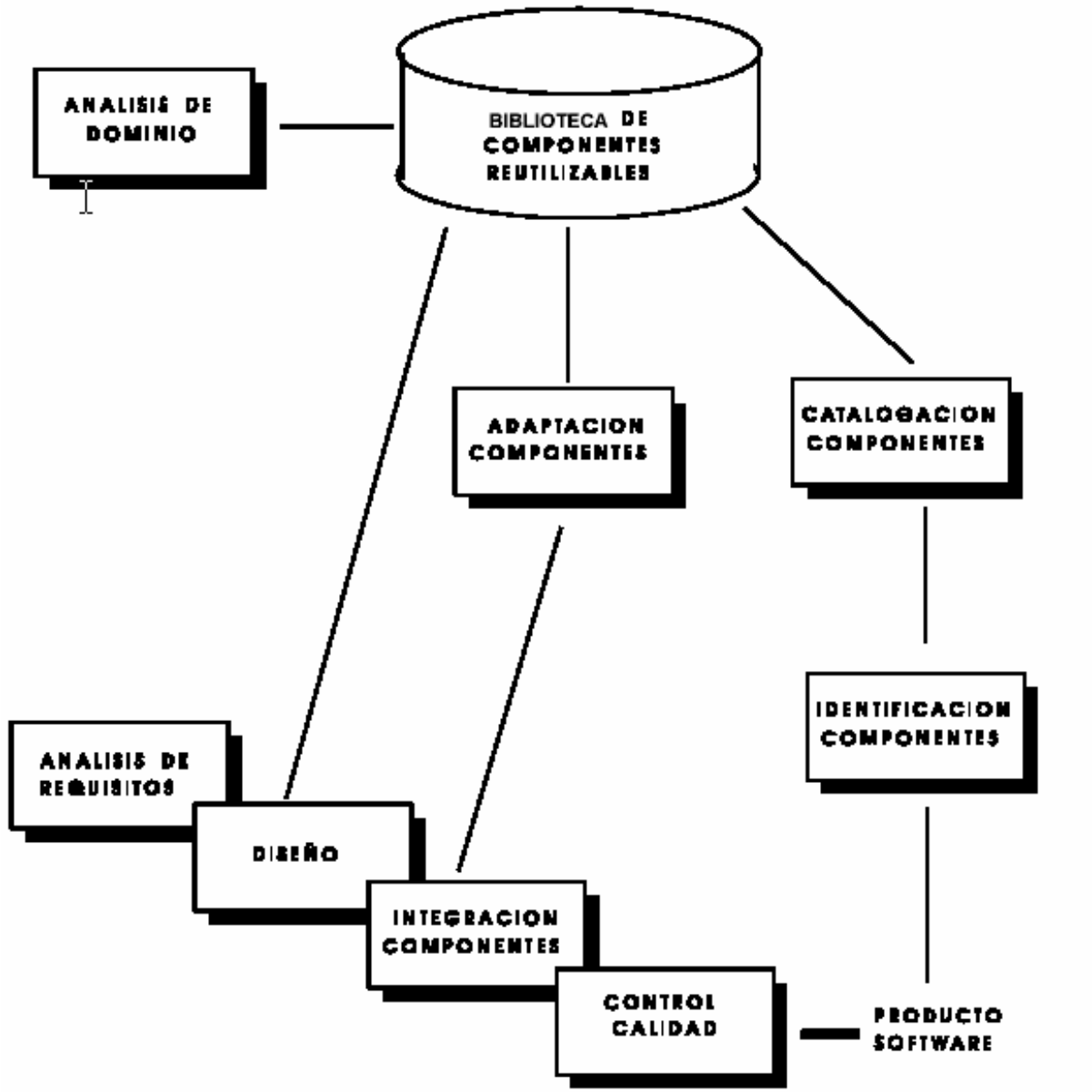
MODELO EN ESPIRAL



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

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

1. Reduce tiempos y costes de desarrollo
2. Aumenta la fiabilidad
3. Dificultad para reconocer los componentes potencialmente reutilizables
4. Dificultad de catalogación y recuperación
5. Problemas de motivación
6. Problemas de gestión de configuración