INTRODUCCIÓN AL DESARROLLO DEL SOFTWARE

ENTORNOS DE PROGRAMACIÓN

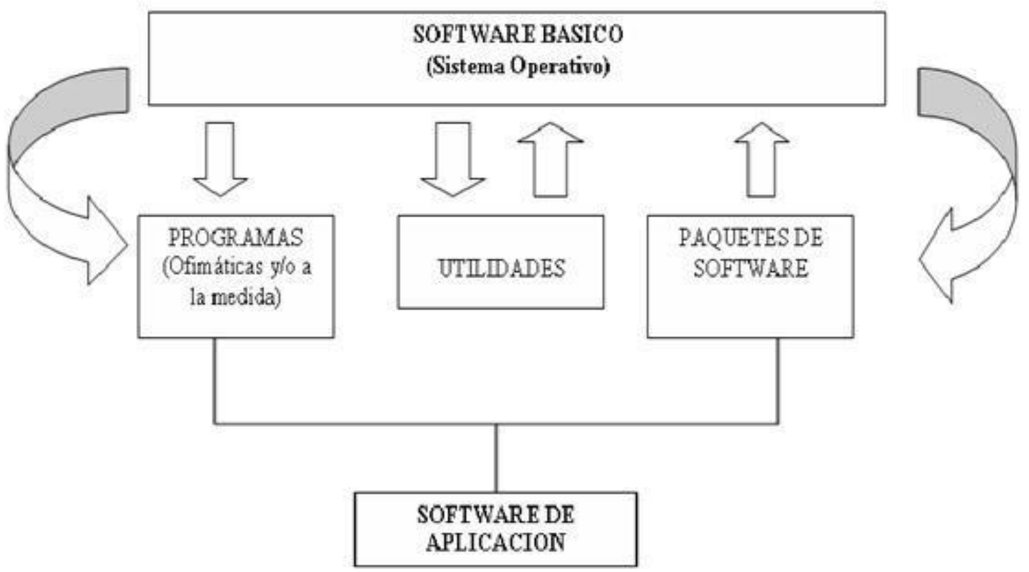
JAKELIN. SAYEL

“Sé menos curioso acerca de las personas y más curioso acerca de las ideas.”

Marie Curie

SOFTWARE

- Es la parte lógica que dota al equipo físico de capacidad para realizar cualquier tipo de trabajo



SOFTWARE

El software de ordenador

- Se conoce como ***software***[1] al soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados *hardware*.
- Clasificación según:
  - Tipo de trabajo que realizan.
  - Método de distribución.
  - Cuenta de licencia.

SOFTWARE

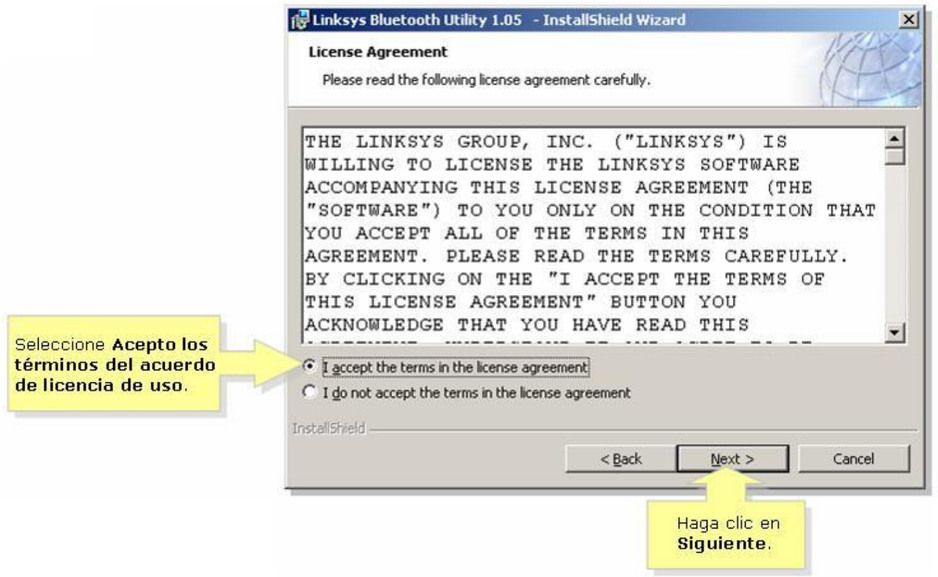
➔ **Clasificación del software según el tipo de trabajo que realiza**

SOFTWARE

Clasificación teniendo en cuenta el tipo de licencia

Licencia: Contrato que se establece entre el desarrollador de un software, sometido a propiedad intelectual y a derechos de autor, y el usuario, en el cual se definen con precisión los derechos y deberes de ambas partes.

- Software libre.
- Software propietario.
- Software de dominio público.



SOFTWARE

➔ **Clasificación del software según el método de distribución**

Entre estos se encuentran los llamados programas enlatados.

- Comercial
- Shareware
- Freeware
- Adware
- Software de uso específico.

CONCEPTO DE PROGRAMA INFORMÁTICO

- Según la wikipedia tenemos las siguientes definiciones:
  - **Programa**: “Secuencia de instrucciones que una computadora puede interpretar y ejecutar.”
  - **Dato:**“Un dato es una representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa. Los datos describen hechos empíricos, sucesos y entidades. Es un valor o referente que recibe el computador por diferentes medios, los datos representan la información que el programador manipula en la construcción de una solución o en el desarrollo de un algoritmo. ”

CONCEPTO DE PROGRAMA INFORMÁTICO

- Un programa informático es un conjunto de instrucciones que se ejecutan de manera secuencial con el objetivo de realizar una o varias tareas en un sistema

PROGRAMA = DATOS + INSTRUCCIONES

CONCEPTO DE PROGRAMA INFORMÁTICO

➔ Para realizar un programa, necesitamos diseñar un **algoritmo** que lo resuelva.

Un **algoritmo** es un conjunto ordenado y finito de operaciones que permiten resolver un problema y que cumplen las siguientes características:

- Tiene un número finito de pasos
- Acaba en un tiempo finito. Si no acabase nunca, no se resolvería el problema
- Todas las operaciones deben estar definidas de forma precisa
- Puede tener varios datos de entrada y como mínimo uno de salida

CONCEPTO DE PROGRAMA INFORMÁTICO

El ejemplo más claro es cualquier receta de cocina en la que hay unos pasos con un orden definido, tienen unos datos de entrada que son los ingredientes y una salida que es el plato preparado.

Ejemplo: Preparar un sandwich mixto a la plancha.

- Datos de entrada: Pan, jamón york y queso
- Datos de salida: Sandwich mixto
- Procedimiento:
  - - Sacar dos rebanadas de pan de la bolsa
  - - Poner jamón york sobre una de las rebanadas
  - - Poner queso sobre la rebanada
  - - Tapar con la otra rebanada
  - - Calentar plancha
  - - Poner el sandwich sobre la plancha durante 30 segundos
  - Girar el sandwich y dejar otros 30 segundos
  - Apagar plancha

CONCEPTO DE APLICACIÓN INFORMÁTICA

- Es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo.

- Esto lo diferencia principalmente de otros tipos de programas como:

  - los sistemas operativos(que hacen funcionar al ordenador),
  - las utilidades (que realizan tareas de mantenimiento o de uso general),
  - y los lenguajes de programación (con el cual se crean los programas informáticos)

CONCEPTO DE APLICACIÓN INFORMÁTICA

- Las aplicaciones informáticas, son aplicaciones desarrolladas «a medida» que ofrecen gran potencia, ya que están exclusivamente diseñadas para resolver un problema específico.
- Conjunto de uno o más programas enlazados o relacionados entre sí, junto con la documentación generado durante el proceso de desarrollo de dicha aplicación.

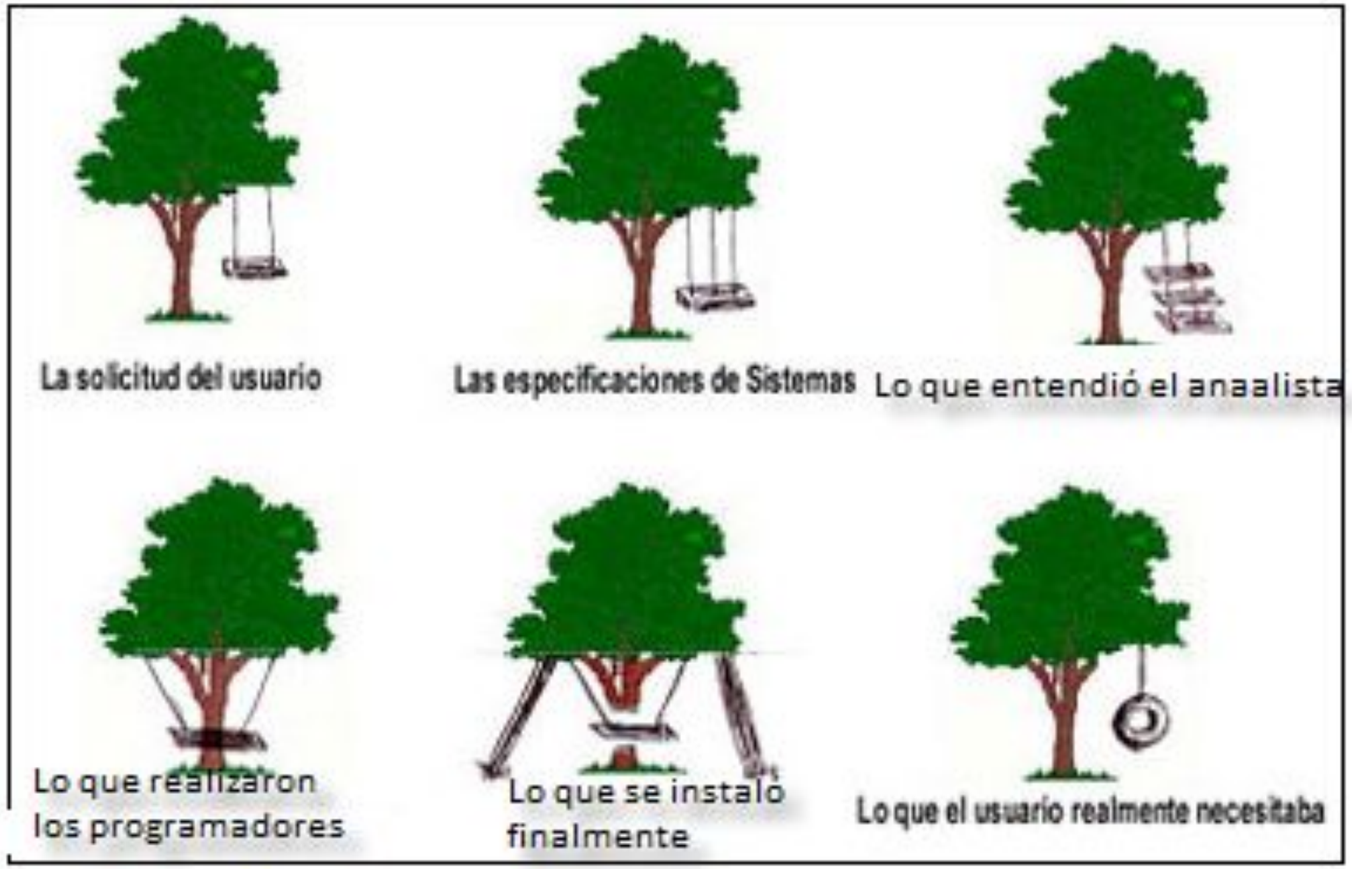
APLICACIONES

Actualmente la informática tiene tantas aplicaciones que prácticamente es inconcebible pensar que exista un campo o área donde la informática no este presente. Algunas de estas son:

- En el área Administrativa
- En la educación
- En la Navegación
- En la Aeronáutica
- En la Ciencia
- En el transporte urbano
- En la industria
- En la vigilancia
- En el campo de la medicina

INGENIERÍA SOFTWARE

- Desde que se detecta la necesidad de construir un sistema de software hasta que este está disponible o es retirado, se identifican varias etapas que en conjunto se denominan el **ciclo de vida del software.**



CICLO DE VIDA DEL SOFTWARE

- Un ciclo de vida es el conjunto de fases por las que un sistemas software pasa desde que surge la idea hasta que muere.
- El proceso de desarrollo del software implica un conjunto de actividades que se tienen que planificar y gestionar de tal manera que aseguren un producto final que dé solución a las necesidades de todas aquellas personas que lo van a utilizar
- En función de cuáles sean las características del proyecto, se configurará el ciclo de vida de forma diferente en cada caso

- De los ciclos de vida destacan ciertos aspectos claves que son: las fases , las transiciones entre fases y las posibles entradas y salidas que surgen en cada transición.

- Los ciclos de vida tradicionales son:
  - Ciclo de vida lineal o en cascada.
  - Ciclo de vida evolutivo

Las etapas principales a realizar en cualquier ciclo de vida

1. Análisis
2. Diseño
3. Codificación
4. Pruebas
5. Documentación
6. Explotación
7. Mantenimiento

Fases de desarrollo del software.

Ciclo de vida:

“Todas las etapas por las que pasa un proyecto de desarrollo de software desde que se inicia hasta que se finaliza y se considera una solución completa, correcta y estable”.

Ingeniería del software.

- QUÉ
  - Se va a desarrollar

1. Estudio del Sistema
2. Planificación del Sistema
3. Especificación de Requisitos
4. Análisis del Sistema

- CÓMO
  - Se va a desarrollar

1. Diseño (arquitectura, estructura de datos, ...)
2. Codificación e implementación
3. Pruebas

- CAMBIO
  - Que se puede producir en el sistema

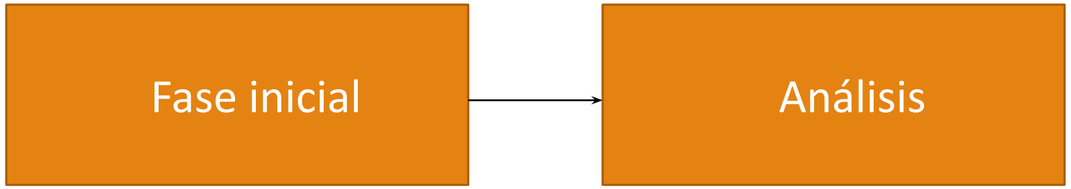
Adaptación o mejora del sistema

Ingeniería del software: Fases de desarrollo.

Fase inicial

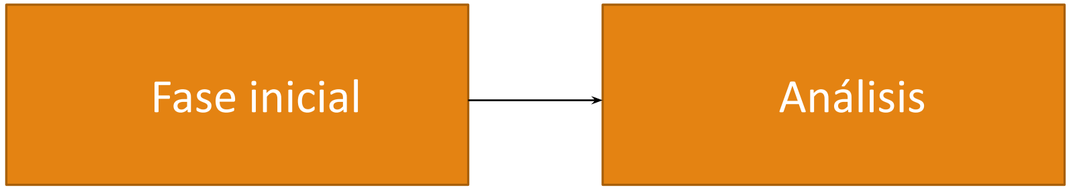
En esta fase se hacen estimaciones de **viabilidad** del proyecto, es decir, si es rentable o no y se establecen las bases de las fases del proyecto.

Fase de Análisis



- Se analiza el problema, es decir, se recopila, examina y formulan los requisitos del cliente.
- Es fundamental, tener claro **QUÉ hay que hacer (comprender el problema)**.
- Se realizan varias reuniones con el cliente, generando documentación en la que se recopilan los requisitos, Especificación de Requisitos del Software (**ERS**) .
- Al finalizar estas reuniones, se genera un acuerdo en el que el cliente se compromete a no variar los requisitos hasta terminar una primera **release**.

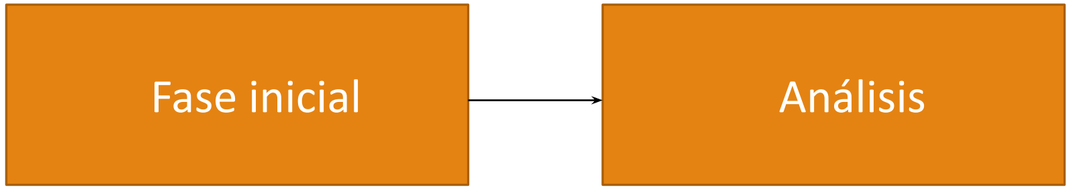
Fase de Análisis



Para facilitar la comunicación con el cliente, se utilizan técnicas como:

- ✔ Casos de uso (técnica **UML**, en la que se representan los requisitos funcionales del sistema).
- ✔ Brainstorming (desde distintos puntos de vista).
- ✔ Desarrollo conjunto de aplicaciones (**JAD**) y Planificación conjunta de requisitos (**JRP**). Apoya en la dinámica de grupos y, respectivamente, los productos generados comprenden los requisitos claves o estratégicos.
- ✔ Prototipos (versión inicial del sistema).

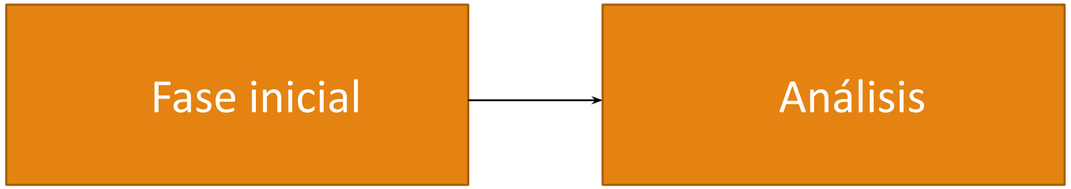
Fase de Análisis



Hay dos tipos de requisitos:

- ✔ **Requisitos funcionales**: Describen con detalle la función que realiza el sistema.
- ✔ **Requisitos no funcionales**: Tratan de las características del sistema como fiabilidad, mantenibilidad, plataforma hardware, S.O, restricciones, etc.

Fase de Análisis



Para su representación, se pueden utilizar:

- Diagrama de Flujo de Datos (**DFD**) de distintos niveles: Representan el flujo de datos entre los distintos procesos, entidades externas y almacenes que forman el sistema.
- Diagrama de Flujo de Control (**DFC**): Similares a los DFD pero muestran el flujo de control.
- Diagramas de transición de Estados (**DTE**): Representa cómo se comporta el sistema como consecuencia de sucesos externos.
- Diagrama Entidad/Relación (**DER**): Se usa para representar los datos y sus relaciones entre ellos.

Fase de Diseño

- Consiste en dividir el problema en partes y se determina la función de cada una de estas. Se decide **CÓMO se hace (CÓMO se resuelve el problema).**
- Se elabora una documentación técnica y detallada de cada módulo del sistema.
- La documentación es realizada por el *analista junto al jefe de proyecto*.
- Los tipos de diseños más utilizados son:
  - **Diseño estructurado (clásico)**, está basado en el flujo de datos a través del sistema.
  - **Diseño Orientado a Objetos**, el sistema se entiende como un conjunto de objetos. Los cuales, tienen propiedades, comportamientos y se comunican entre ellos.

Fase de Codificación

- El programador, codificará el software según los criterios seleccionados en la etapa de diseño.
- Al trabajar en equipo, debe haber unas normas de codificación y estilo, claras y homogéneas
- Cuanto más exhaustivo haya sido el análisis y el diseño, la tarea será más sencilla, pero nunca está exento de necesitar reanálisis o un rediseño si se encuentran problemas al programar.

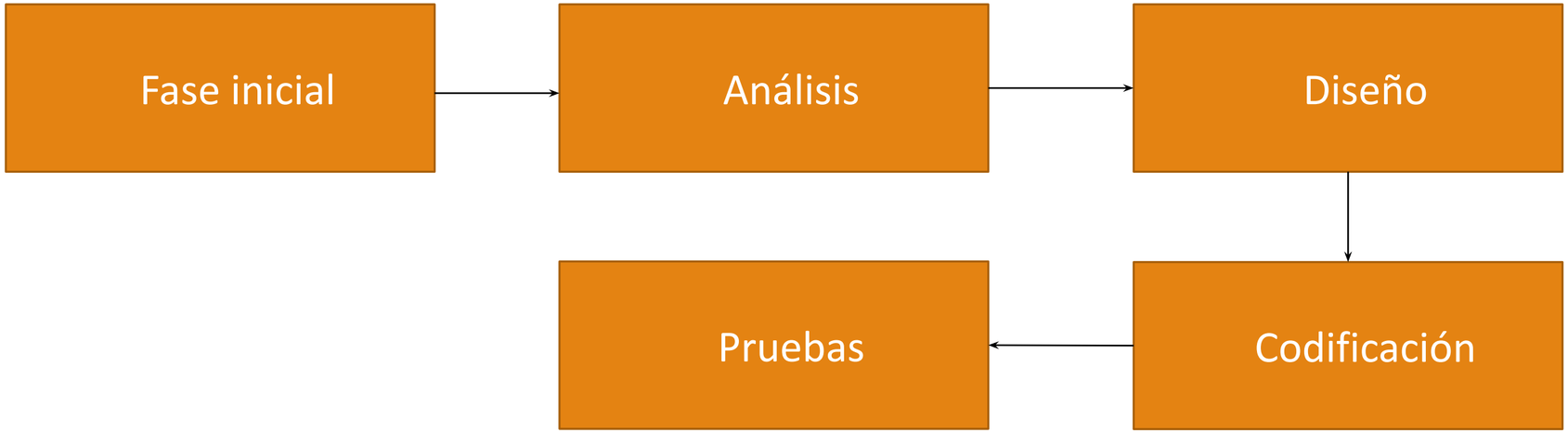
Fase de Codificación

En esta fase, se genera documentación del código que se desarrolla: entradas, librerías, etc.

Se pueden distinguir dos tipos:

- **Documentación del proceso**: Comprende las fases de desarrollo y mantenimiento.
- **Documentación del producto**: Describe el producto que se está desarrollando, se divide en dos tipos: documentación del usuario y documentación del sistema.

Fase de Pruebas



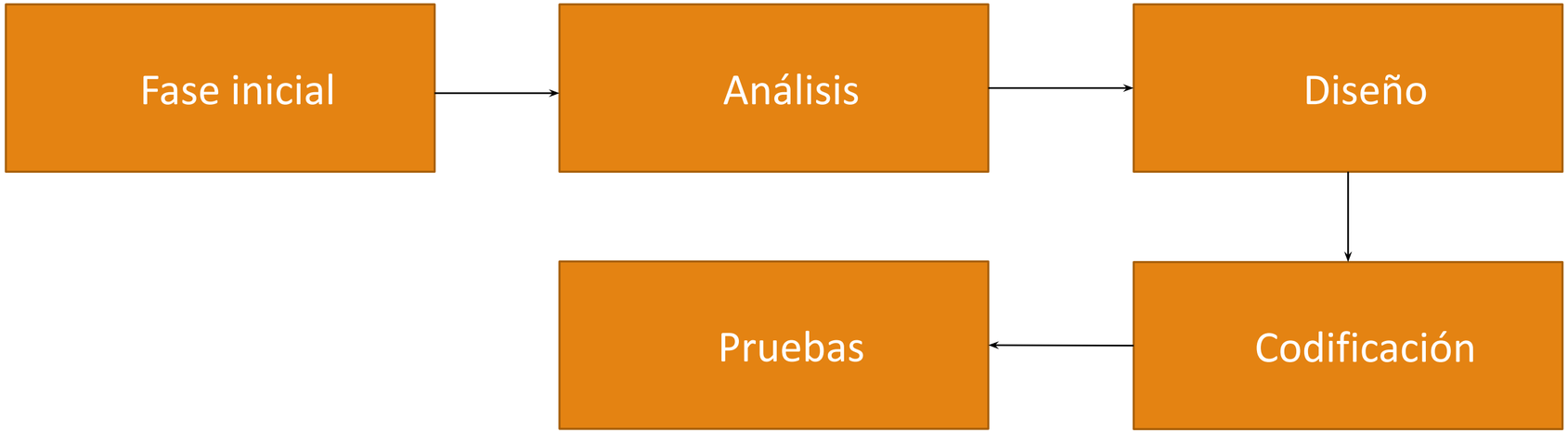
Se realizan pruebas para verificar que el programa se ha realizado acuerdo las especificaciones originales, es decir, se comprueba lo que **DEBE HACER**.

Existen varios tipos de pruebas:

1) **Funcionales**: Se validan las especificaciones establecidas por el cliente y, posteriormente, será validada por este.

2) **Estructurales**: Se realizan pruebas técnicas sin necesidad del consentimiento del cliente. Se realizarán cargas reales de datos.

Fase de Pruebas

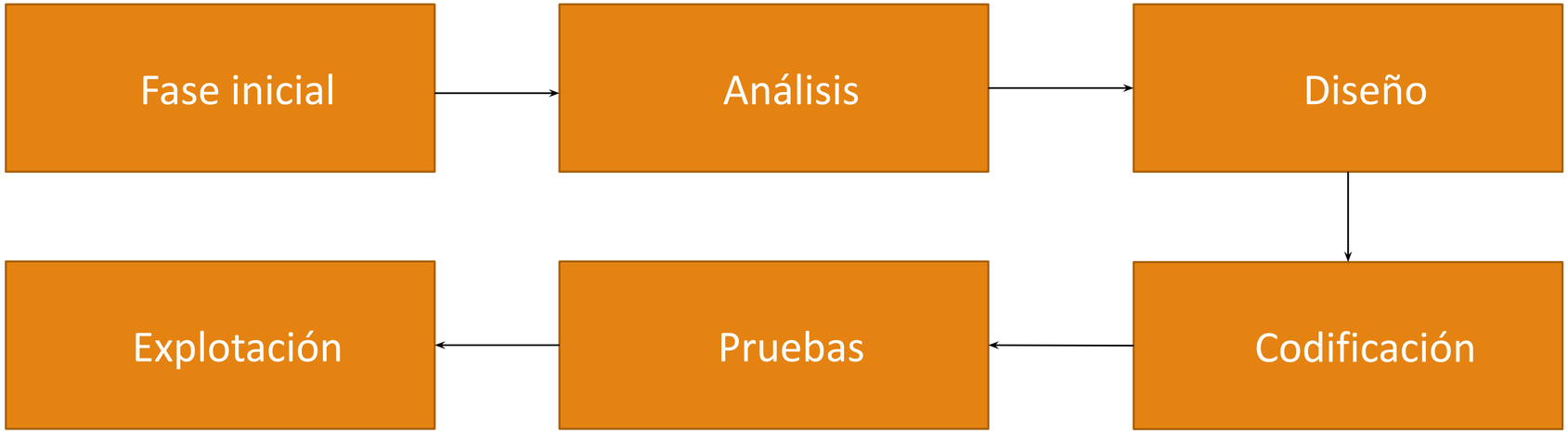


Esta fase comprende las siguientes tareas:

- **Verificación**: Se trata de comprobar si se está construyendo el software correctamente, es decir si implementa correctamente la función.
- **Validación**: Trata de comprobar si el software construido se ajusta a los requisitos del cliente.

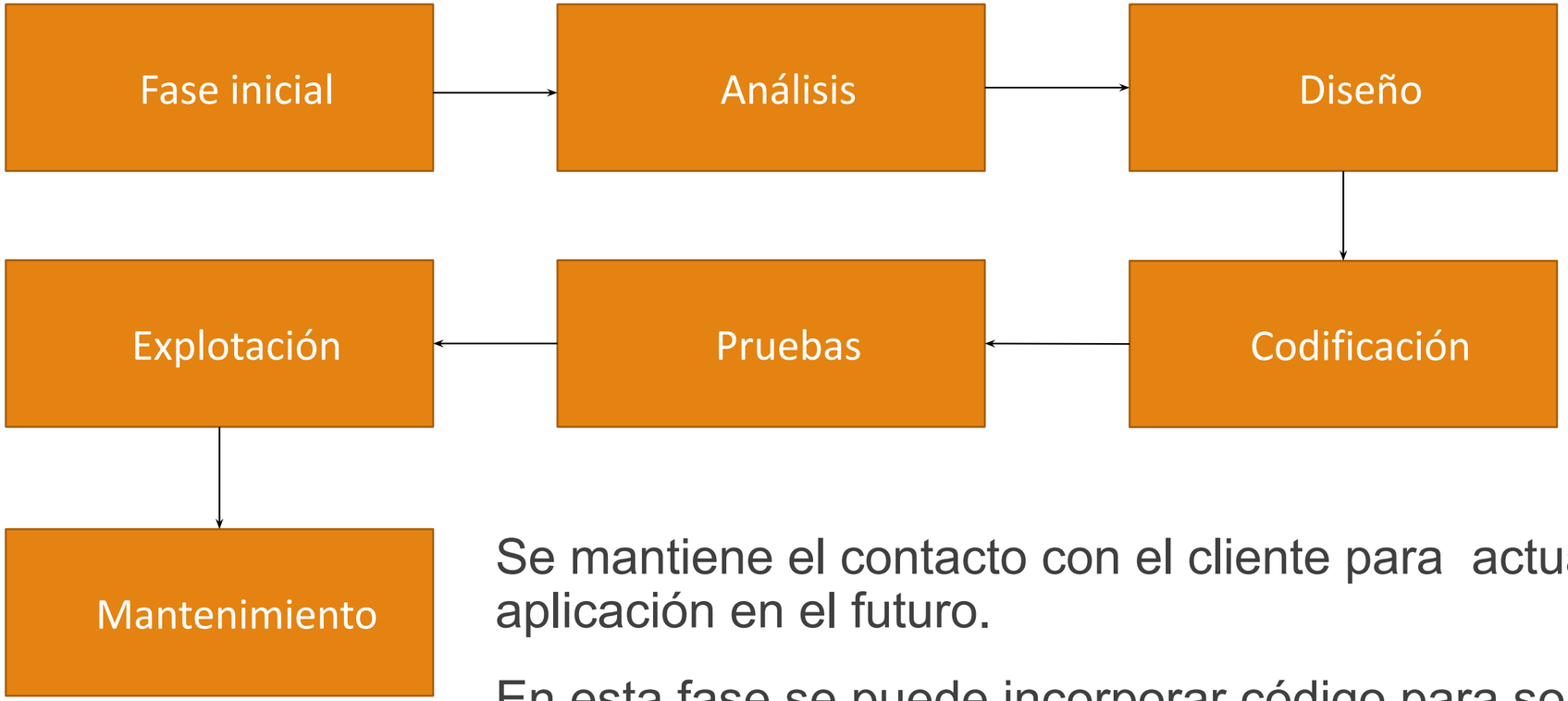
En general, las pruebas las debería realizar, personal diferente al que codificó, con una amplia experiencia en programación.

Fase de Explotación



- Consiste en la instalación y puesta en marcha del software en el entorno de trabajo del cliente, es decir, el software es instalado y utilizado en un entorno real de uso cotidiano.
- Es la fase más larga, ya que suele conllevar múltiples incidencias (**bugs**) y nuevas necesidades.
- En esta etapa se realizan las siguientes tareas: Estrategia para la implantación, pruebas de operación, uso operacional del sistema y soporte al usuario.

Fase de Mantenimiento



Se mantiene el contacto con el cliente para actualizar y modificar la aplicación en el futuro.

En esta fase se puede incorporar código para soluciones a problemas, ampliar la funcionalidad para un cliente, …

Ciclo de Vida en Cascada

➔ Ciclo de vida en el que las distintas fases se van encadenado las etapas de manera consecutiva. Es decir, la siguiente no empieza hasta que no haya terminado la anterior.

Este ciclo de vida no comprendía la retroalimentación entre fases y la posibilidad de volver atrás.

Por lo tanto, al llevarlo a la práctica lo que ocasionó que no se supiera reaccionar a errores ocurridos en fases avanzadas.

Ventajas:

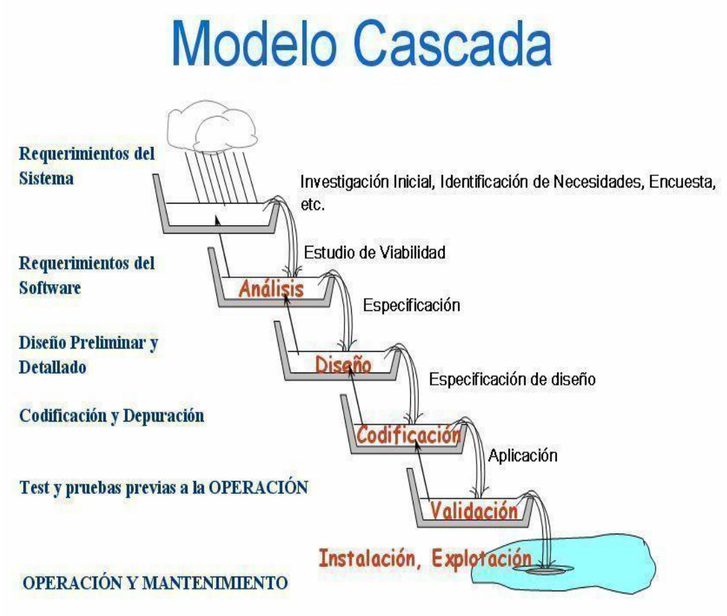
- ✔ Fácil de comprender, planificar y seguir.
- ✔ La calidad del producto resultante es alta.
- ✔ Permite trabajar con personal poco cualificado.

Inconvenientes:

- × Necesidad de tener todos los requisitos definidos desde el principio (pueden surgir [illegible]os)
- × Es difícil volver atrás si se cometen errores en una etapa.
- × El producto no está disponible para su uso hasta que no está totalmente terminado.

Ciclo de Vida en Cascada con retroalimentación

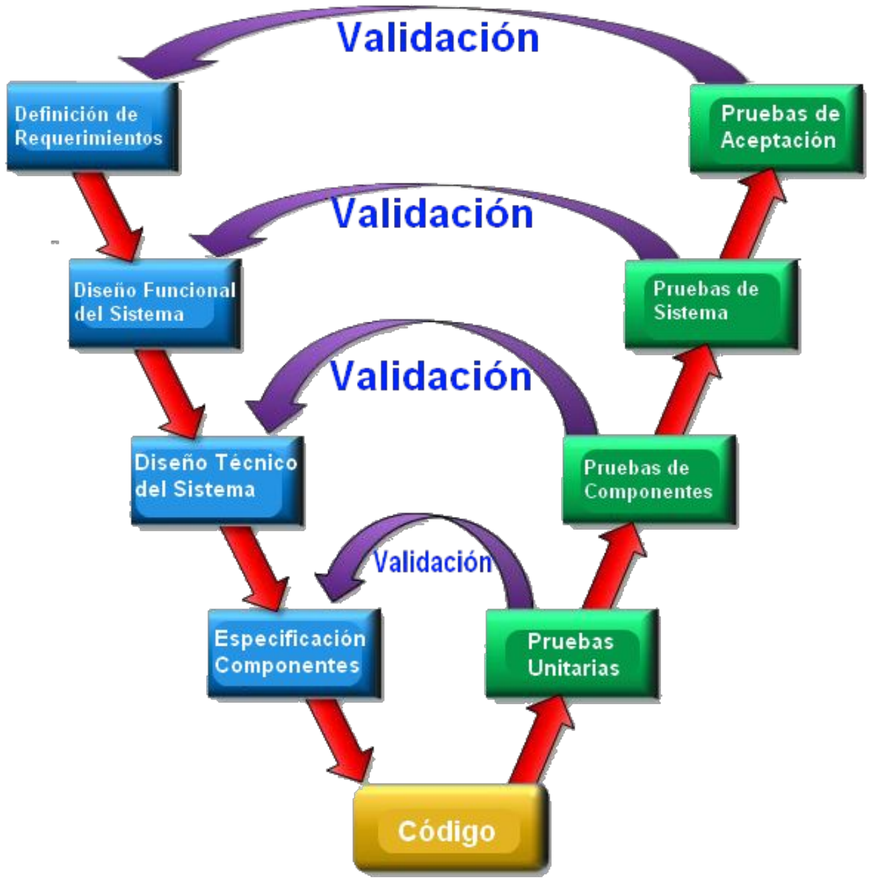
- Se recomienda (dada la necesidad de tener todos los requisitos desde el principio y la dificultad de volver atrás) cuando:
  - El proyecto es similar a alguno que ya se haya realizado con éxito anteriormente
    - Los requisitos son estables y están bien comprendidos.
    - Los clientes no necesitan versiones intermedias.



Ciclo de Vida en V

- El modelo en V viene para corregir los fallos del modelo en cascada, incluyendo explícitamente la retroalimentación.

- El problema es que los fallos detectados en fases tardías de un proyecto obligaban a empezar de nuevo, lo cual supone un gran sobrecoste.

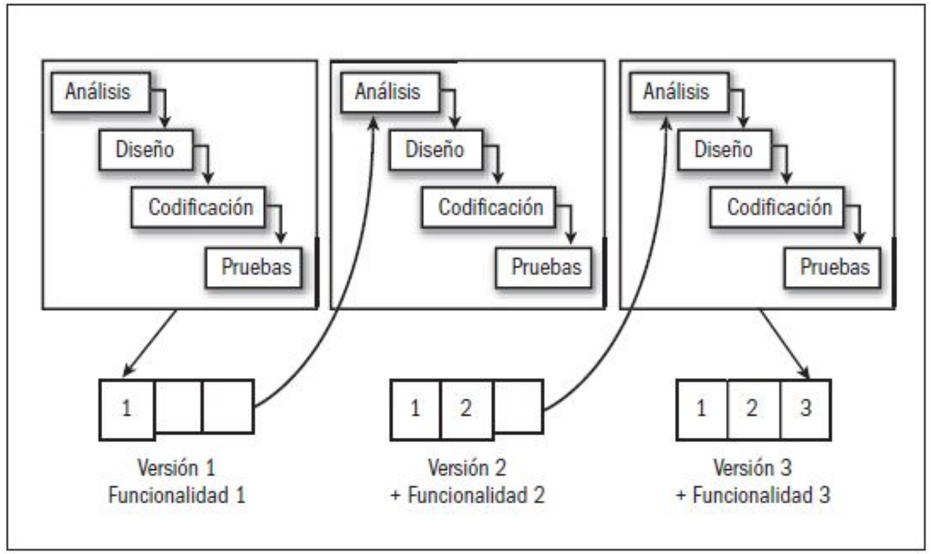


Ciclo de Vida Incremental

- Este modelo busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de solicitud de requerimientos.
- El ciclo se compone de iteraciones, las cuales, están compuestas por las fases básicas.
- Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto.

El cliente al finalizar cada iteración, evalúa el producto y lo corrige o propone mejoras.

Estas iteraciones se repetirán hasta que se desarrolle un producto que satisfaga las necesidades del cliente.



Ciclo de Vida Incremental

- **Ventajas del desarrollo iterativo:**
  - No se necesitan conocer todos los requisitos desde el comienzo.
  - Permite la entrega temprana al cliente de partes operativas del software.
  - Permite separar la complejidad del proyecto, gracias a su desarrollo por parte de cada iteración o bloque.
  - Las entregas facilitan la retroalimentación de los próximos entregables.

- **Debilidades de este modelo de desarrollo:**
  - Es difícil estimar el esfuerzo y el coste final necesario.
  - Se tiene el riesgo de no acabar nunca.
  - No es recomendable para sistemas en tiempo real, de alto índice de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos

Ciclo de Vida Incremental: Modelo en Espiral

MODELO ESPIRAL

CONCEPTO

El modelo en espiral del proceso del software que originalmente fue propuesto por Boehm (1988) .El modelo en espiral es una de las mas recomendables para el desarrollo y creación de un programa, ya que consta de pocas etapas o fases, las cuales se van realizando en manera continua y cíclica.

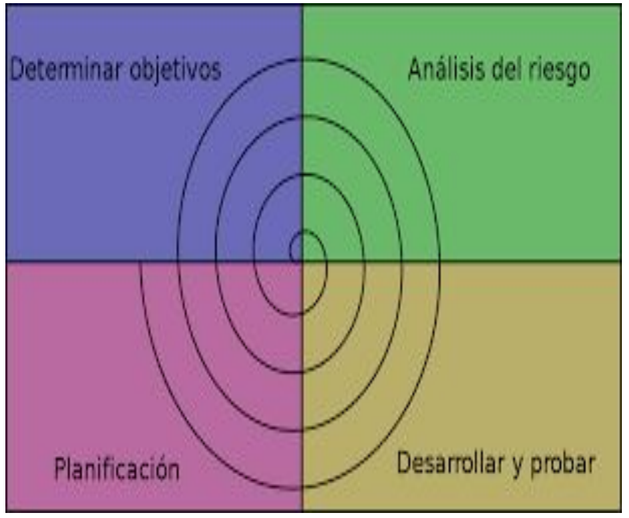
Barry Boehm
Es un ingeniero informático estadounidense y también es profesor emérito de esta materia en el departamento de ciencias tecnológicas en la Universidad del Sur de California. Es conocido por sus múltiples aportes a este campo.

Ciclo de Vida Incremental: Modelo en Espiral

➔ El proceso de desarrollo de software se representa como una espiral, donde en cada ciclo se desarrolla una parte del mismo.

Cada ciclo está formado por cuatro :

- **Determinar objetivos:** Se identifican los objetivos, se ven las alternativas para alcanzarlos y las restricciones impuestas a la aplicación (costos, plazos, interfaz …).
- **Análisis de riesgos:** Un riesgo es por ejemplo, requisitos no comprendidos, mal diseño, errores en la implementación, etc. Utiliza la construcción de prototipos como mecanismo de reducción de riesgos.
- **Desarrollar y probar:** Se desarrolla la solución al problema en este ciclo y se verifica que es aceptable.
- **Planificación:** Revisar y evaluar todo lo realizado y con ello decidir si se continúa, entonces hay que planificar las fases del ciclo siguiente.



Lenguajes de programación.
CONTINUARÁ

SEGUIMOS!!