TIPOS DE LICENCIA SOFTWARE

* **Software del sistema:** Permite al hardware funcionar.
* **Software de aplicación:** Nos permite realizar tareas específicas de manera automatizada o asistida.
* **Software de programación / desarrollo:** Software que permite crear nuevo software.
* **Software libre:** Gratuito y en ocasiones es posible su modificación.
* **Software propietario:** Uso mediante un pago.

Otras modalidades:

**Shareware→** Se libera de forma gratuita pero su uso está limitado, es decir, algunas funciones están bloqueadas o solo te aceptan usarlo durante cierto tiempo

**Freeware→** Gratuito pero no lo puedes modificar

**Open Source→** El código del programa es libre y habitualmente se podría modificar dependiendo de las licencias

**Dominio Público→** Software el cual todo el mundo puede usarlo, ya que no tiene licencia o se ha caducado o se desconoce su autor

CICLO DE VIDA DEL SOFTWARE

Se divide en:

* **Análisis:** estudiar y analizar las necesidades o los problemas a resolver
* **Diseño:** especificar cómo vamos a resolver esos problemas
* **Codificación:** traducir el problema a lenguaje de programación, ya sea de alto o bajo nivel
* **Pruebas:** comprobar que el software realice la tarea con algunos requisitos de calidad
* **Mantenimiento:** se realizan posteriores para adaptarlo a nuevas necesidades o cambios obligados por errores

MODELOS DE CICLO DE VIDA

MODELO CASCADA

* **Análisis del sistema:**
  + Identificación del problema o necesidades
  + División del sistema en subsistemas
  + Evaluación de la viabilidad técnica y económica
  + Establecimiento de restricciones de coste y plazo de entrega
* **Análisis conceptual:** Diseño de las operaciones y datos del sistema informático, es decir, cómo va a ser, dónde se va a ejecutar, etc
* **Diseño lógico:**
  + Datos
  + Arquitectura
  + Lógica de los procedimientos
  + Lógica de la Interfaz UI
* **Codificación:** Traducir el diseño lógico a físico, con un código ejecutable programado
* **Pruebas:** Con un equipo distinto al de los desarrolladores
* **Lanzamiento y mantenimiento**

Ventajas:

* Fácil de comprender y planificar
* Producto resultante de alta calidad
* Permite trabajar con personal poco cualificado

Desventajas:

* Tienen que estar definidos todos los requisitos desde el principio
* Ninguna de las etapas empieza sin antes haber terminado la anterior
* El producto no estará en funcionamiento hasta el final

MODELO ITERATIVO INCREMENTAL

Se parte de una versión previa a la que se le van añadiendo nuevas funciones y/o requisitos.

**Análisis → Diseño → Codificación → Pruebas → Entregar 1ª versión**

**↓**

**Al finalizar la 1ª entrega**

**↓**

**Análisis → Diseño → Codificación → Pruebas → Entregar 2ª versión**

**↓**

**Al finalizar la 2ª entrega**

**↓**

**Análisis → Diseño → Codificación → Pruebas → Entregar 3ª versión**

Ventajas:

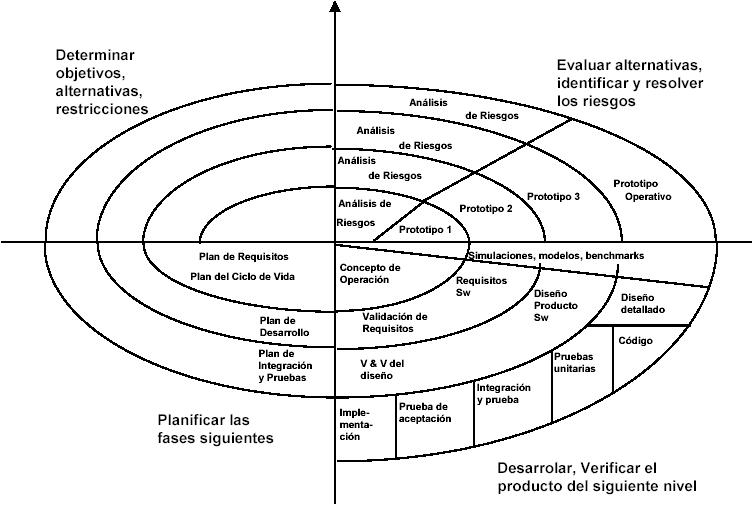
* No se requiere el conocimiento de todos los requisitos desde el comienzo
* Entrega temprana de versiones operativas
* Estas entregas permiten la realimentación para futuras evoluciones

Desventajas:

* No se conoce el precio final correcto
* No recomendable para sistemas de alta disponibilidad o altos requisitos de calidad

MODELO EN ESPIRAL

Es también incremental, pero parte de otro prototipo distinto al cómo será el producto al final. Normalmente, las primeras etapas suelen ser en papel.



**Inicio → Análisis de riesgos → Prototipo → Concepto de operaciones → Requisitos del ciclo de vida → Análisis de objetivos → Análisis de riesgos → Prototipo → Pruebas → Requisitos Software → Validación de estos →…→ Prototipo funcional → Pruebas →... → Implementación →…**

**Ventajas:**

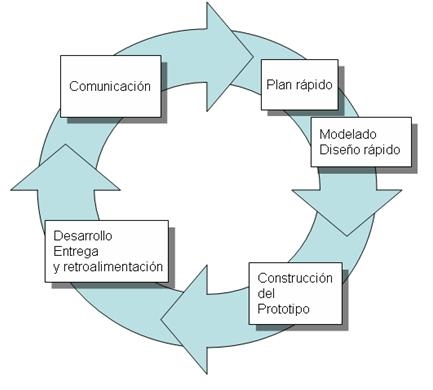
* No requiere conocimiento de todos los requisitos desde el principio
* Análisis de riesgos en cada ciclo
* Reduce riegos y se asegura que sea de calidad

**Desventajas:**

* El coste final es difícil de evaluar
* El coste aumenta en cada ciclo
* Depende de la fase del análisis de riesgos

Este modelo se recomienda para proyectos de gran tamaño, con constantes cambios y aquellos en los que el factor de riesgos sea muy importante

MODELO DE PROTOTIPADO

Usado mucho hoy en día, se saca un prototipo y liberan el software para que lo prueben, es decir, son los propios usuarios los que hacen la fase de pruebas.

Ventajas:

* Útil cuando el cliente conoce los objetivos generales para el software, pero no los requisitos de entrada, procesamiento o salida.
* Mejor enfoque cuando el desarrollador está inseguro de la eficacia del algoritmo.
* Se puede reutilizar el código.

Desventajas:

* Tiende a crearse unas expectativas cuando ve el prototipo

DESARROLLO ÁGIL DE SOFTWARE

Se saca el software de manera rápida, con fallos, para ser el primero en salir. WhatsApp, FaceBook, etc han usado esta metodología y han triunfado.

FASES DE DESARROLLO DE SOFTWARE

ANÁLISIS

Diálogo entre la empresa desarrolladora y el cliente. Entender y comprender el problema y sus requisitos. Técnicas:

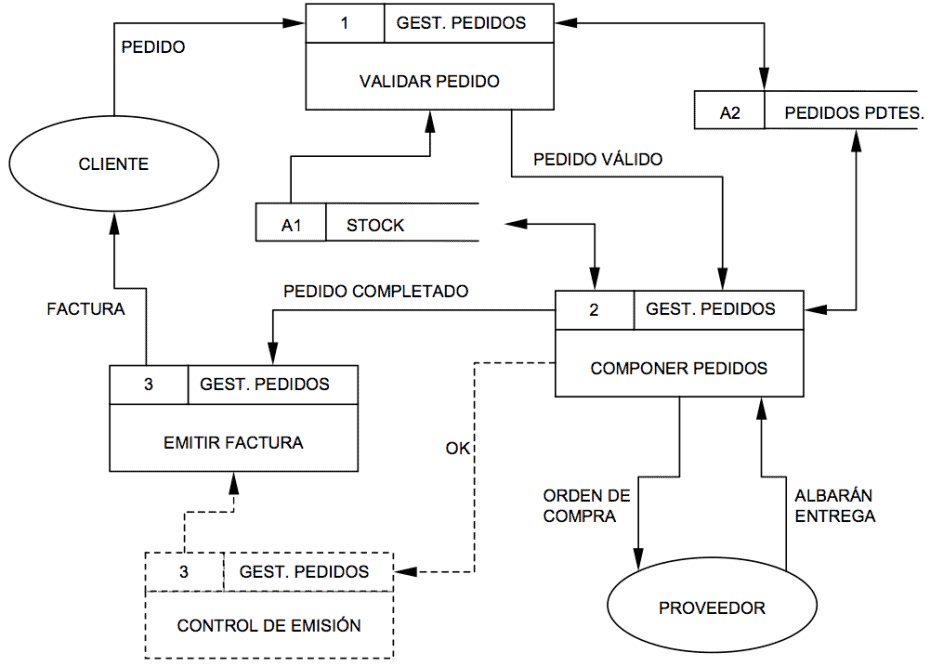
* **Entrevistas**
* **Desarrollo conjunto de aplicaciones (JAD)**
* **Planificación conjunta de requisitos (JRP)**
* **Brainstorming**
* **Prototipos**
* **Casos de uso (modelos UML)**

REQUISITOS

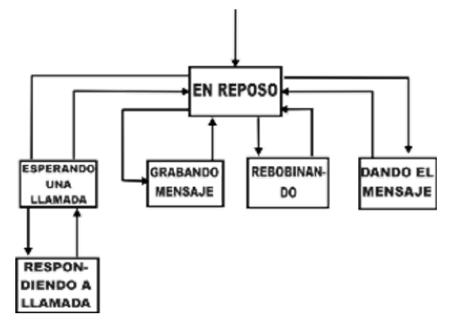
* **Requisitos funcionales**, funcionalidades que va poseer el desarrollo y cómo se va a abordar.
* **Requisitos no funcionales,** entorno de operación del desarrollo (hardware)

Estos se pueden representar mediante:

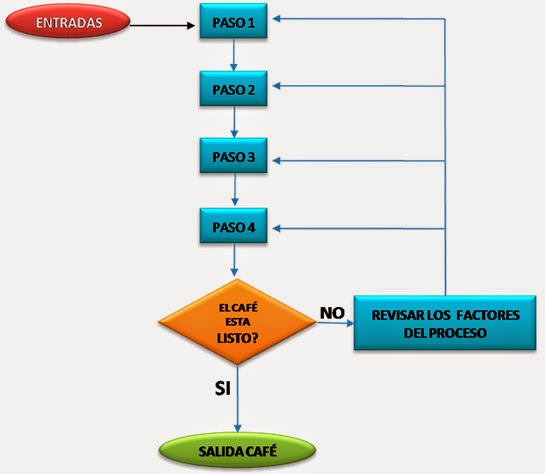
* **Diagramas de flujo de datos (DFD)**



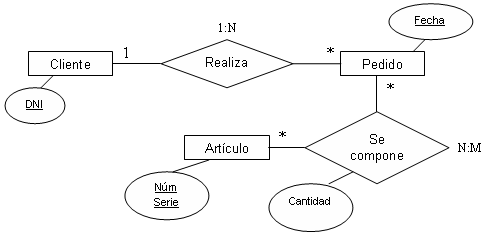
* **Diagramas de transición de estados (DTE)**



* **Diagramas de flujo de control (DFC)**



* **Diagramas Entidad/Relación (DER)**



* **Diccionario de Datos (DD):** Se siguen estándares para crear documentos

DISEÑO

DISEÑO ESTRUCTURADO O CLÁSICO

Cuatro fases que apoyan unas a otras.

* **Diseño de datos**
* **Diseño arquitectónico**
* **Diseño de interfaces**
* **Diseño a nivel de componentes o procedimental:** Usa la programación estructurada. Hay 3 tipos:
  + **Secuencial**
  + **Condicional**
  + **Repetitiva**

Para la ayuda en este tipo de programación se usan recursos como:

* + **Diagramas de flujo**
  + **Diagramas de caja**
  + **Tablas de decisión**
  + **Pseudocódigo**

DISEÑO ORIENTADO A OBJETOS

Se requiere un análisis orientado a objetos, en el cual se definen clases, con sus atributos y métodos, así como las relaciones entre las distintas clases. Existen 4 capas de diseño:

* **Subsistema:** Se implementan las funciones principales.
* **Clases y objetos:** Requisitos.
* **Mensajes:** Colaboración entre objetos.
* **Responsabilidades:** Operaciones y atributos que caracterizan a cada clase.

CODIFICACIÓN

Una vez terminado el diseño se procede a la codificación en el que se crea un conjunto de instrucciones llamado código fuente. Este debe seguir unas normas de codificación para facilitar su futura lectura y mantenimiento.

PRUEBAS

* **De caja negra:** Se centra en validar los requisitos funcionales sin fijarse en el funcionamiento interno del código.
* **De caja blanca:** Se realizan las pruebas conociendo el código fuente del programa.

El flujo a seguir en esta fase se resume en los siguientes pasos:

* **Generación de un plan de pruebas a partir de documentación del proyecto**
* **Diseño de pruebas a partir del plan de pruebas**
* **Generación de los casos de pruebas**
* **Definición de los procedimientos pruebas**
* **Ejecución de pruebas**
* **Evaluación, realizando un informe de pruebas.**
* **Depurar y localizar posibles errores**
* **Análisis de errores**

DOCUMENTACIÓN

Todas las etapas de desarrollo deben quedar documentadas:

* **Documentación del proceso**
* **Documentación del producto**
  + **De usuario**
  + **Del sistema**

Todos estos documentos deben tener una estructura ordenada según unas pautas mínimas.

EXPLOTACIÓN

Es la instalación y puesta en marcha del producto software. Hay varias tareas:

* **Estrategia de implementación del proceso**
* **Pruebas de operación**
* **Uso operacional del sistema**
* **Soporte del usuario**

MANTENIMIENTO

* **Adaptativo:** Se adapta el software a los cambios del hardware y software.
* **Correctivo:** Se arreglan errores que ha encontrado el cliente y no han podido ser localizados en la fase de pruebas.
* **Perfectivo:** Se añaden nuevas funcionalidades.
* **Preventivo:** Modificaciones o mejoras al software tanto para las funcionalidades del software como para el mantenimiento de este.

Se deben realizar las siguientes tareas:

* Implementación del proceso
* Análisis de problemas y modificaciones
* Implementación de las modificaciones
* Revisión/Aceptación del mantenimiento
* Migración
* Retirada del software

PROGRAMA

Es un conjunto de instrucciones escritas en un lenguaje de programación. Los equipos actuales utilizan mayoritariamente la estructura Von Neumann que consta de:

* **Fase de búsqueda de las instrucciones**
* **Fase de ejecución de las instrucciones**

LENGUAJE DE PROGRAMACIÓN

Consta de:

* **Un alfabeto/vocabulario/léxico**
* **Una sintaxis**
* **Una semántica**

Según la abstracción del lenguaje puede ser de:

* **Bajo nivel**
* **Medio nivel**
* **Alto nivel**

Según la forma de ejecución:

* **Compilado:** Se lee el código y se traduce todo a código máquina
* **Interpretado:** Se traduce el código con cada instrucción

Según su paradigma de programación:

* **Imperativo**
* **Declarativo**

OBTENCIÓN DEL EJECUTABLE

* **Análisis del léxico:** Se lee secuencialmente todo el código.
* **Análisis sintáctico:** Se comprueba la estructura del programa.
* **Análisis semántico:** Se comprueba que las declaraciones son correctas, así como las expresiones, operaciones, etc.
* **Generación de código intermedio**
* **Optimización de código**
* **Generación de código**

MÁQUINAS VIRTUALES

Software que es capaz de ejecutar programas en un entorno virtual como si se tratase de una máquina real.

* **De sistema:** Se pueden tener varias máquinas virtuales que actuarán como una máquina independiente más en una misma máquina física.
* **De proceso:** Se ejecuta un proceso en un entorno de ejecución independiente del hardware o software.

HERRAMIENTAS DE CREACIÓN DE CÓDIGO

* **Diagramas de flujo:** Nos permite modelar la estructura procedimental de un programa
* **Pseudocódigo:** Podemos escribir los procedimientos de un programa usando un lenguaje más común con instrucciones clave que hacen referencia a sintaxis habituales en los lenguajes de programación.