# DESARROLLO DE SOFTWARE

## Unidad Didáctica 1

# Contenidos

Bloques:

I. El software del ordenador
   A. Clasificación del Software
   B. Software de Programación

II. **Fases de desarrollo de una aplicación informática**
   A. **Ciclo de vida del software**
   B. Modelos de ciclo de vida

# Ciclo de vida

El estándar ISO/IEC 12207-1 define **ciclo de vida del software como..**

- Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso

# Ciclo de vida

- El ciclo de vida de un producto software comprende el periodo que transcurre desde que el producto es concebido hasta que deja de estar disponible o es retirado
- Se divide en **etapas** o **fases de desarrollo:**
  - Análisis
  - Diseño
  - Codificación
  - Pruebas
  - Documentación
  - Explotación
  - Mantenimiento

# Análisis

- En esta etapa se debe entender y comprender de forma detallada el problema que se va a resolver
  - El análisis establece QUÉ queremos hacer
- Ojo! Un mal análisis nos conduciría al desastre ya que todo el desarrollo del software depende de saber qué se necesita y cómo se quiere utilizar
- Es fundamental generar una documentación entendible, completa y fácil de verificar y modificar

# Diseño

- Partimos de lo que queremos hacer y en esta etapa estableceremos CÓMO se va a solucionar el problema
- Se deducen estructuras de datos, la arquitectura del software, la interfaz de usuario y procedimientos
  - Por ejemplo, en esta etapa hay que elegir el lenguaje de programación, el Sistema Gestor de Base de Datos, etc.

# Codificación

- Traducimos el diseño a lenguaje procesable por la máquina
- Es decir desarrollamos la arquitectura, procedimientos e interfaz que habíamos diseñado
- La salida de esta fase es código ejecutable

# Pruebas

- Se comprueba que se cumplen criterios de corrección y calidad
- Las pruebas deben garantizar el correcto funcionamiento del sistema

# Mantenimiento

- Esta fase tiene lugar después de la entrega del software al cliente o empresa
- Debemos garantizar que el software puede adaptarse al entorno, a los posibles cambios (cambios de sistema operativo en la empresa, en usuarios, en funcionalidades…)

# Explotación

- En esta etapa se lleva a cabo la instalación y puesta en marcha del producto software en el entorno de trabajo del cliente

# Documentación

- Una tarea fundamental a realizar **en cada etapa** es la documentación
  - Cada etapa tiene como entrada uno o varios documentos procedentes de las etapas anteriores y produce otros documentos de salida

# Análisis

# Análisis

- En esta fase se analizan y especifican los **requisitos** o capacidades que el sistema debe tener porque el cliente así lo ha pedido
- La obtención de los requisitos no es tarea fácil, ya que el cliente puede no tenerlos claros, pueden surgir nuevos requisitos, cambiar lo especificado, malos entendidos o el cliente puede no expresarse de forma clara debido a la falta de conocimientos informáticos, etc.
- Para realizar un proyecto satisfactorio es necesario obtener unos buenos requisitos y para ello es esencial **una buena comunicación** entre cliente y desarrolladores
- Para facilitar esta comunicación se utilizan varias técnicas, algunas de ellas son las siguientes:

# Análisis

- **Entrevistas:** es la técnica más tradicional que consiste en hablar con el cliente
  - Hay que tener sobre todo, conocimientos de psicología
- **Brainstorming:** es un tipo de reuniones cuyo objetivo es generar ideas desde diferentes puntos de vista para la resolución de un problema
  - Su utilización es adecuada al principio del proyecto, pues puede explorar un problema desde muchos puntos de vista
- **Prototipos:** es una versión inicial del sistema que se utilizan para clarificar algunos puntos, demostrar los conceptos,... En definitiva, para enterarse más acerca del problema y sus posibles soluciones
  - Después puede desechar o se puede utilizar como punto inicial del sistema

# Tipos de requisitos

Se especifican dos tipos de requisitos:

- **Requisitos funcionales:** describen con detalle la función que realiza el sistema, cómo reacciona ante determinadas entradas, cómo se comporta en situaciones particulares, etc.
- **Requisitos no funcionales:** tratan sobre las características del sistema, cómo puede ser la fiabilidad, mantenibilidad, sistema operativo, plataforma hardware, restricciones, limitaciones, etc.

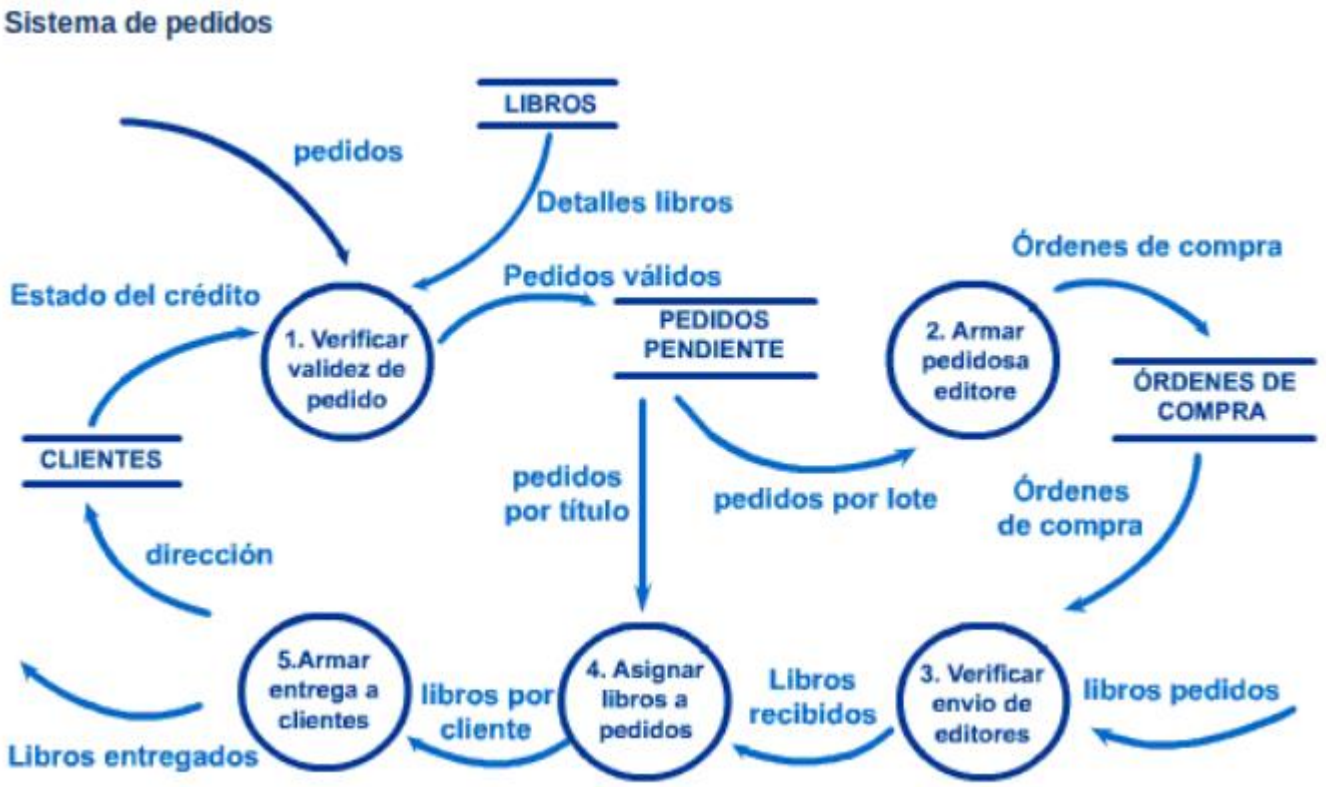
# Tipos de requisitos. Ejemplos

| Requisitos funcionales | Requisitos no funcionales |
| --- | --- |
| El usuario puede agregar un nuevo contacto | La aplicación debe funcionar en Windows y Linux |
| El usuario puede ver lista de contactos | Tiempo de respuesta a consultas, altas, bajas y modificaciones deber ser inferior a 5 segundos |
| El usuario puede eliminar un cliente de la lista | Interfaz debe ser a través de ventanas y debe ser intuitiva |
| El usuario puede modificar cliente de una lista | El manejo de la aplicación se realizará con teclado y ratón |

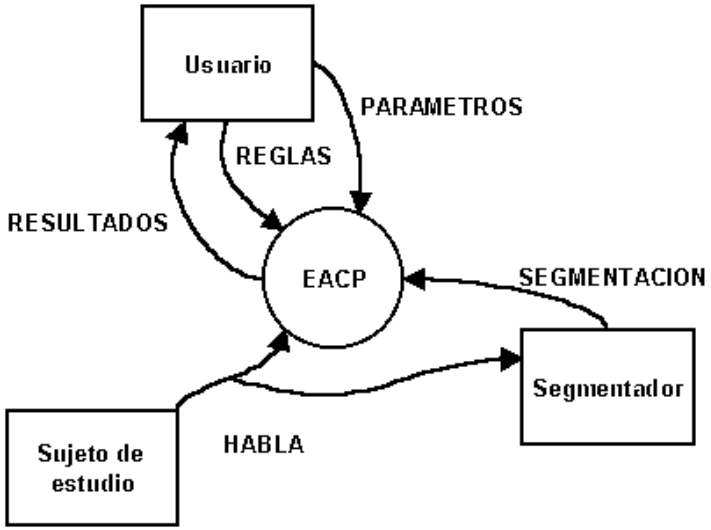
# Representar requisitos

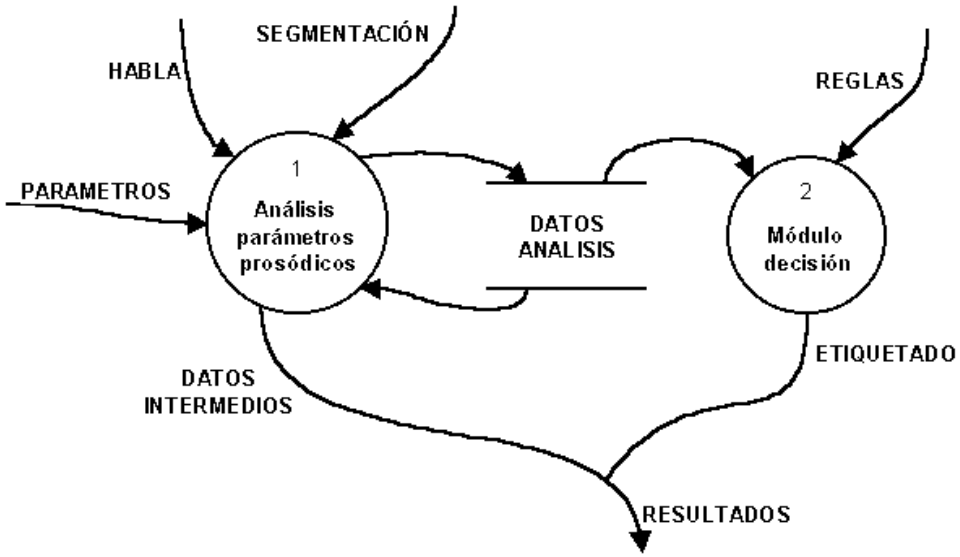
- Para representar los requisitos se utilizan diferentes **técnicas:**
- **Diagramas de flujo de datos, DFD.** Es un diagrama que representa el flujo de datos entre los distintos procesos, entidades externas y almacenes que forman el sistema
  - **Procesos:** identifican funciones dentro del sistema. Se representan mediante burbujas ovaladas o circulares.
  - **Entidades externas:** representan componentes que no forman parte del sistema pero proporcionan datos al sistema o reciben de él. Se representan mediante un rectángulo
  - **Almacenes:** representan los datos desde el punto de vista estático, es decir, representan el lugar donde se almacenan los datos procesados o desde donde se recuperan para apoyar un proceso. Se representan mediante dos líneas horizontales y paralelas
  - **Flujo de datos:** representa el movimiento de datos dentro del sistema. Se representa mediante flechas

# DFD. Ejemplo


Sistema de pedidos
LIBROS
pedidos
Detalles libros
Estado del crédito
Pedidos válidos
Órdenes de compra
1. Verificar validez de pedido
PEDIDOS PENDIENTE
2. Armar pedidosa editore
ÓRDENES DE COMPRA
CLIENTES
pedidos por título
pedidos por lote
Órdenes de compra
dirección
5.Armar entrega a clientes
libros por cliente
4. Asignar libros a pedidos
Libros recibidos
3. Verificar envio de editores
libros pedidos
Libros entregados

# DFD. Ejemplo

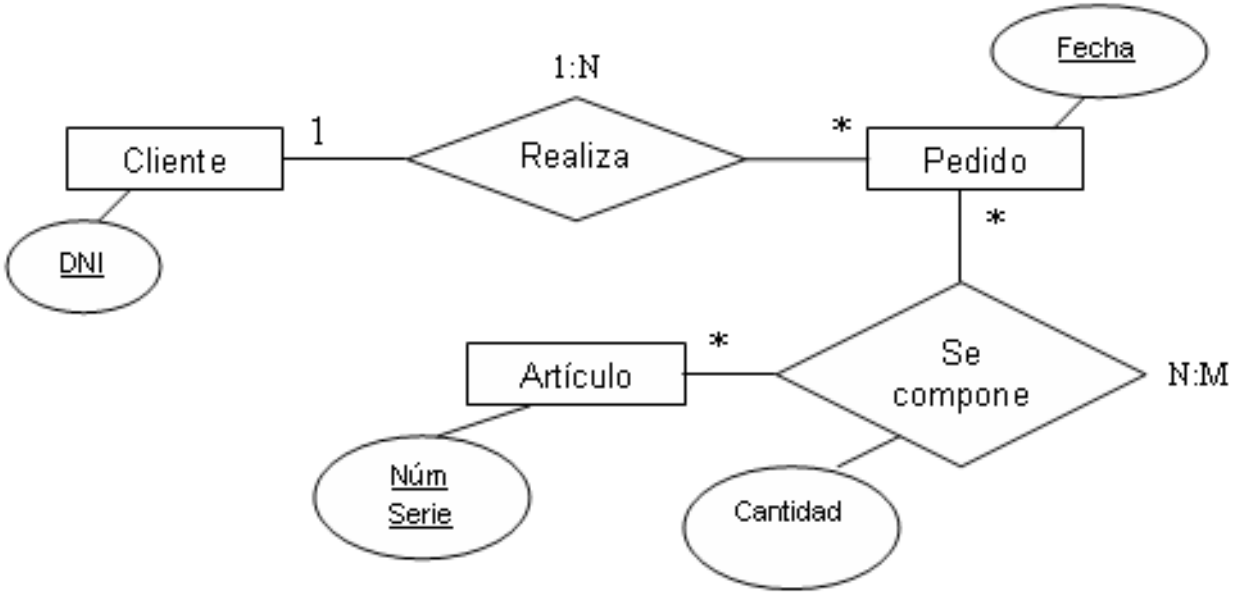

Usuario
PARAMETROS
REGLAS
RESULTADOS
EACP
SEGMENTACION
Segmentador
Sujeto de estudio
HABLA


SEGMENTACIÓN
HABLA
REGLAS
PARAMETROS
1
Análisis parámetros prosódicos
DATOS ANALISIS
2
Módulo decisión
ETIQUETADO
DATOS INTERMEDIOS
RESULTADOS

# Representar requisitos

- **Diagramas de flujo de control, DFC:** Similar a los DFD pero con la diferencia de que muestra el flujo de control, en lugar de datos
- **Diagrama Entidad Relación, DER:** Usado para representar los datos y la forma en la que se relacionan entre ellos
- **Diccionario de datos, DD:** Es una descripción detallada de los datos utilizados por el sistema que gráficamente se encuentran representados por los flujos de datos y almacenes presentes en el conjunto de DFD

# DER. Ejemplo


Fecha
1:N
Cliente
1
Realiza
*
Pedido
DNI
*
Artículo
*
Se
compone
N:M
Núm
Serie
Cantidad

# DD. Ejemplo

| Empleado | | | | | |
|---|---|---|---|---|---|
| Llave | Nombre | Campo | Tipo | Tamaño | Descripciòn |
| PK | Còdigo Empleado | cdep | Numerico | 4 | Almacena còdigo del empleado |
| | Nombres | nbep | Texto | 50 | Almacena nombre del empleado |
| | Apellidos | apep | Texto | 50 | Almacena apellidos del empleado |
| | Telèfono | tlep | Numerico | 10 | Almacena telèfono del empleado |
| | Gènero | gnep | Texto | 10 | Almacena Gènero del empleado |
| | E_mail | emep | Hipervinculo | 100 | Almacena correo electrònico del empleado |
| | Direcciòn | drep | Memo | 100 | Almacena direcciòn del empleado |
| FK | Còdigo barrio | cdbrr | Numerico | 4 | Almacena còdigo del barrio |
| FK | Còdigo ciudad | cdcdd | Numerico | 4 | Almacena còdigo de la ciudad |
| FK | Còdigo tipo contrato | cdtc | Numerico | 4 | Almacena còdigo del tipo de contrato |
| FK | Còdigo localidad | cdl | Numerico | 4 | Almacena còdigo de la localidad |

# Documentación

- Todo lo realizado en esta fase debe quedar reflejado en el documento de ***Especificación de Requisitos del Software (ERS)***. *(IEEE 830)*
- Este documento:
  - no debe tener ambigüedades,
  - debe ser completo,
  - consistente,
  - fácil de verificar y modificar,
  - fácil de utilizar en la fase de explotación y mantenimiento y
  - fácil de identificar el origen y las consecuencias de los requisitos
- Sirve como entrada para la siguiente fase

# Diseño

# Diseño

- Una vez identificados los requisitos es necesario componer la forma en que se solucionará el problema
- En esta etapa se traducen los requisitos funcionales y no funcionales en una representación de software
- Vemos dos tipos de diseño:
  - **Diseño estructurado:** que está basado en el flujo de los datos a través del sistema
  - **Diseño orientado a objetos:** donde el sistema se entiende como un conjunto de objetos que tienen propiedades y comportamientos, además de eventos que activan operaciones que modifican el estado de los objetos

# Diseño Estructurado

Pasamos ahora a ver con detalle Algoritmos y Pseudocódigo para un Diseño Estructurado

- Documentos: U01.-Algoritmos

# Diseño Orientado a Objetos

- Para el análisis y diseño OO se utiliza **UML *(Unified Modeling Language* –** *Lenguaje de Modelado Unificado)*
- Es un lenguaje de modelado basado en diagramas que sirve para expresar modelos
- Se ha convertido en el estándar de facto de la mayoría de las metodologías de desarrollo OO que existen hoy en día

⇨ Más adelante estudiaremos el Diseño Orientado a objetos

# Codificación

# Codificación

- Una vez realizado el diseño se realiza el proceso de codificación
- En la etapa de codificación el programador recibe las especificaciones del diseño y las transforma en un conjunto de instrucciones escritas en un lenguaje de programación
- A este conjunto de instrucciones se le llama **código fuente**
- El programador debe conocer la sintaxis del lenguaje de programación utilizado

# Codificación

- En cualquier proyecto en el que trabaja un grupo de personas debe haber unas **normas de codificación y estilo, claras y homogéneas**
- Estas normas facilitan las tareas de corrección y mantenimiento de los programas, sobre todo cuando se realizan por personas que no los han desarrollado

⇨ Lee el fichero **ConvencionesJava.pdf** que describe detalladamente las **Convenciones de código para el lenguaje de programación Java**

# Pruebas

# Pruebas

- En esta etapa ya se dispone del software y se trata de encontrar errores
  - de codificación
  - de especificación
  - de diseño

# Pruebas

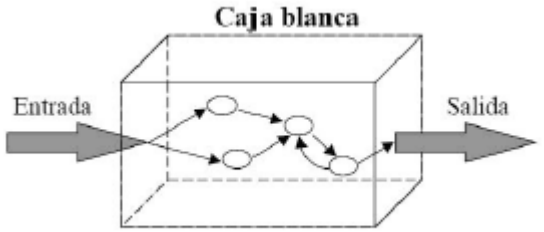
- El **objetivo** de esta etapa es planificar y diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y esfuerzo
  - Una prueba ***tiene éxito*** si descubre un error no detectado hasta entonces
- Un **caso de prueba** es un documento que especifica los valores de entrada, salida esperada y las condiciones previas para la ejecución de la prueba

# Pruebas

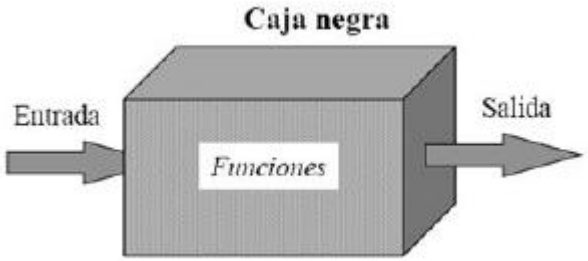
- Las recomendaciones para las pruebas son:
  - Cada prueba debe definir los resultados de salida esperados
  - Es necesario revisar los resultados de cada prueba en profundidad
  - Las pruebas deben incluir datos de entrada válidos y esperados, así como no válidos e inesperados
  - Centrar las pruebas en dos objetivos:
    - Comprobar si el software no hace lo que debe hacer
    - Comprobar si el software hace lo que no debe hacer
  - Evitar hacer pruebas que no estén documentadas ni diseñadas con cuidado
  - No planear pruebas asumiendo que no se encontrarán errores
  - La probabilidad de encontrar errores en una parte del software es proporcional al número de errores ya encontrados
  - Las pruebas son una tarea creativa

# Técnicas diseño de pruebas

- **Pruebas de caja blanca**
  - Se centran en validar la estructura interna del programa (requiere conocer el funcionamiento interno)




- **Pruebas de caja negra**
  - Se centran en validar los requisitos funcionales sin fijarse en funcionamiento interno



# Pruebas con JUnit

- Junit es una herramienta para realizar pruebas unitarias automatizadas

Lo estudiaremos próximamente

# Documentación

# Documentación

- Todas las etapas de desarrollo deben quedar perfectamente documentadas
- En esta etapa será necesario reunir todos los documentos generados y clasificarlos según el nivel técnico de sus descripciones

# Documentación

La documentación presentada se divide en dos clases:

- **La documentación del proceso**
  - Estos documentos registran el proceso de desarrollo y mantenimiento
  - Se indican planes, estimaciones y horarios que se utilizan para predecir y controlar el proceso de software. Informan sobre cómo usar los recursos durante el proceso de desarrollo. Describen normas de cómo se ha de implementar el proceso, etc…
- **La documentación del producto**
  - Describe el producto que está siendo desarrollado
  - Define dos tipos de documentación:
    - **la documentación del sistema** que describe el producto desde el punto de vista técnico orientado al desarrollo y mantenimiento
    - **la documentación del usuario** que ofrece una descripción del producto orientada a los usuarios que utilizarán el sistema. Hay que tener en cuenta diferentes usuarios:
      - Usuarios finales
      - Administradores del sistema

# Documentación del Usuario

1. **Descripción funcional del sistema:**

   Proporciona una descripción general de las funciones del sistema

2. **Documento de instalación del sistema:**

   Destinado a los administradores del sistema

   Describe cómo instalar el sistema en un entorno particular

   Debe contener una descripción de los ficheros que componen el sistema, su configuración, el hardware mínimo requerido, cómo iniciar el sistema, etc.

3. **Manual introductorio:**

   Proporciona explicaciones sencillas de cómo empezar a utilizar el sistema. Para facilitar las explicaciones se deben usar ejemplos. Los usuarios principiantes cometen errores, se debe describir la manera de recuperarse antes de ellos

# Documentación del Usuario

**4. Manual de referencia del sistema:**

Proporciona la descripción detallada de la instalación del sistema y su uso

Debe proporcionar una lista completa de mensajes de error y debe describir cómo recuperarse ante los errores detectados

Debe ser completo, formal y descriptivo

**5. Guía del administrador del sistema:**

Para algunos sistemas donde hay que introducir comandos de control se deben indicar las tareas del operador, se deben describir los mensajes que se producen y las respuestas que el operador tiene que dar al sistema

**6. Tarjeta de referencia rápida:**

Sirven para hacer búsquedas rápidas.

Por ejemplo, un sistema de ayuda en línea que contiene información breve sobre alguna parte del sistema, permite al usuario ahorrar el tiempo invertido en consultar manuales (aunque no debe ser visto como sustituto para la documentación más completa)

# Documentación del Sistema

1. **Fundamentos del sistema:**

   Describe los objetivos de todo el sistema

2. **El análisis y la especificación de requisitos:**

   Proporciona información exacta sobre la especificación de los requisitos, según lo acordado entre las partes interesadas (usuario, cliente, desarrollador)

3. **Diseño:**

   Describe la arquitectura del sistema: cómo se aplican los requisitos del sistema, la forma en que se descompone el sistema en un conjunto de unidades de programas que interactúan y la función de cada unidad de programa

4. **Implementación:**

   Proporciona la descripción de la forma en que se expresa el sistema detallado en algún lenguaje de programación formal y las acciones del programa en forma de comentarios dentro de los programas

# Documentación del Sistema

**5. Plan de pruebas:**

Proporciona una descripción de cómo las unidades de programa son evaluadas individualmente y como todo el sistema se prueba después de la integración

**6. Plan de pruebas de aceptación:**

Se describen las pruebas que el sistema debe pasar antes de que los usuarios las acepten

**7. Los diccionarios de datos:**

Contiene las descripciones de todos los términos que se relacionan con el sistema de software en cuestión

# Explotación

# Explotación

- En esta etapa se lleva a cabo la instalación y puesta en marcha del producto software en el entorno de trabajo del cliente

Se realizan las siguientes tareas:

- **Se define la estrategia para la implementación del proceso:**
  - Se desarrolla un plan donde se establecen las normas para la realización de las actividades y tareas de ese proceso

# Explotación

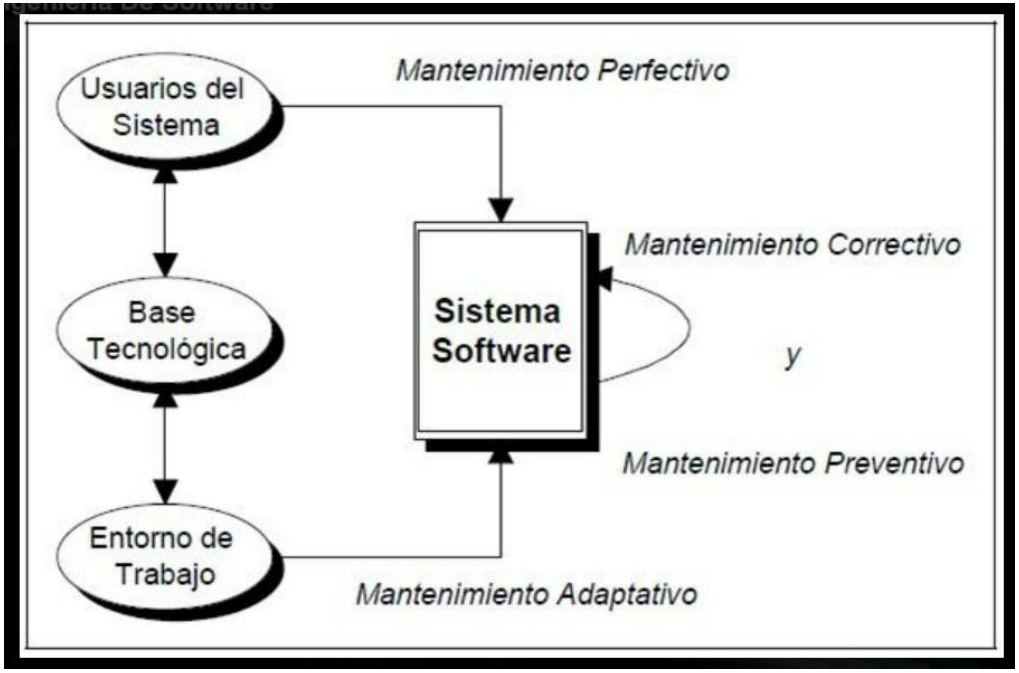
- **Pruebas de operación**
  - Para cada lanzamiento del producto software, se llevarán a cabo pruebas de funcionamiento y tras satisfacerse los criterios especificados, se libera el software para su uso operativo
- **Uso operacional del sistema**
  - El sistema entrará en acción en el entorno previsto de acuerdo con la documentación del usuario
- **Soporte al usuario**
  - Se deberá proporcionar asistencia y consultoría a los usuarios cuando la soliciten. Estas peticiones y las acciones subsecuentes se deberán registrar y supervisar

# Mantenimiento

# Mantenimiento

- El mantenimiento se define como la modificación de un producto de software después de la entrega al cliente para corregir fallos, mejorar el producto, o adaptar a un entorno modificado

- Podemos distinguir cuatro tipos:
  - Adaptativo
  - Correctivo
  - Perfectivo
  - Preventivo



# Mantenimiento

- **Mantenimiento Adaptativo**
  - Con el paso del tiempo se producirán cambios en entorno, empresa… para el que se desarrolló el software. El mantenimiento Adaptativo tiene como objetivo la modificación del producto para adaptarse a esos cambios tanto en hardware como en software.
  - Es el tipo de mantenimiento más habitual
- **Mantenimiento Correctivo**
  - Tras la entrega el cliente es probable que encuentre errores o defectos, este mantenimiento tiene como objetivo corregir los fallos descubiertos

# Mantenimiento

- **Mantenimiento perfectivo**
  - Conforme el cliente utiliza el sw puede descubrir nuevas funcionalidades que mejoran el producto. Este tipo de mantenimiento tiene como objetivo modificar el producto para incorporar nuevas funcionalidades y mejoras
- **Mantenimiento Preventivo**
  - Consiste en modificar el producto sin alterar las especificaciones del mismo con el fin de mejorar y facilitar el mantenimiento
  - Ejemplos: Añadir nuevos comentarios, modificar entornos de desarrollo.