

Herramientas de desarrollo de software

1.- Metodología de desarrollo del software

Definimos metodología como un conjunto de pasos y procedimientos, que deben seguirse para el desarrollo del software.

Los enfoques metodológicos han ido evolucionando a lo largo del tiempo. Inicialmente se identifica un período de **desarrollo convencional**, durante el cual las prácticas de desarrollo eran totalmente artesanales y en el que no había metodologías definidas, lo que acarrea multitud de problemas.

Posteriormente surge el **desarrollo estructurado** partiendo de la programación estructurada y que sigue con los métodos de análisis y diseño estructurado, hasta llegar a metodologías estructuradas que cubren el ciclo de vida completo.

En la actualidad aparece el **paradigma de la orientación a objetos**, como un nuevo enfoque en la ingeniería del software.

- **Desarrollo Convencional:** en los años 50, no existían metodologías de desarrollo. Las personas que desarrollaban los sistemas eran programadores más enfocados en la tarea de codificar, que en la de recoger y comprender las necesidades de los usuarios. Estos, a menudo, no quedaban satisfechos con el sistema, porque sus necesidades no estaban definidas con claridad en una fase de análisis previo. Ante esta perspectiva se vió la importancia del análisis y del diseño en el desarrollo de un sistema. Ahora se empieza a hablar de analistas programadores y analistas de sistemas.

- **Desarrollo Estructurado:** el nacimiento de las técnicas estructuradas se puede considerar un punto de partida en el que se pasa de la construcción de programas de forma artesanal, a una que sigue unos métodos de ingeniería, sentando las bases para un desarrollo automatizado (herramientas CASE).

- **Programación Estructurada:** el enfoque de desarrollo estructurado comenzó con la programación para determinar como se debía ver un programa de forma que fuera lo más comprensible posible. Consiste en establecer normas para la aplicación de las estructuras de datos y de control.

Tipos de estructuras de control: secuencial, repetitiva (bucle), y alternativas.

Tipos de estructuras de datos: vectorial, matricial, listas,...

- **Diseño Estructurado:** a mediados de los 70, el enfoque estructurado se extiende a la fase de diseño. Se define un nivel de abstracción más amplio utilizando el módulo de programa como componente básico de construcción.

- **Análisis Estructurado:** hasta la aparición de los primeros conceptos sobre el análisis estructurado, en la mayoría de los proyectos se hacía una especificación narrativa de los requisitos tal y como los percibía el analista. Estas especificaciones tenían varios problemas:

1- **Eran monolíticas**, es decir había que leer la especificación de requisitos completamente para poder entenderla.

2- **Eran redundantes**, es decir, frecuentemente se repetía la misma información en partes diferentes del documento.

3- **Eran ambiguas**, es decir, un informe detallado de los requisitos se interpretaba de forma diferente por usuarios, analistas y diseñadores, por esto se produce un movimiento gradual hacia las especificaciones funcionales **gráficas** (aquellas compuesta por variedad de diagramas, apoyados con técnicas textuales detalladas, que sirven de referencia a la especificación. Ej: diagrama de flujo de datos), **particionadas** (aquellas que permiten que se puedan leer porciones independientes de la especificación) y **mínimamente redundantes**.

Este enfoque se conoce como análisis estructurado o también análisis descendente o Top-Down.

-Desarrollo Orientado a Objetos: el paradigma orientado a objetos, a diferencia del enfoque estructurado trata los procesos y los datos de forma conjunta, es decir, modulariza tanto la información como el procesamiento. La orientación a objetos empieza con los lenguajes de programación orientados a objetos; en estos lenguajes los problemas del mundo real se representan como un conjunto de objetos para los que se adjuntan un conjunto de operaciones. Ej: C++, java.

En España, la metodología utilizada en la Administración pública es METRICA V3 (<http://administracionelectronica.gob.es>). También existen otras como MERISSE y SSADM.

2.- Herramientas utilizadas en programación

Para llevar a cabo la codificación y prueba de los programas se suelen utilizar entornos de programación . Estos entornos nos permiten realizar diferentes tareas:

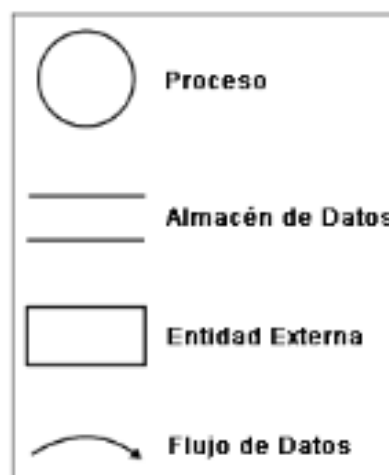
- Crear , editar y modificar el código fuente del programa.
- Compilar , montar y ejecutar el programa.
- Examinar el código fuente.
- Ejecutar el programa en modo depuración.
- Realizar pruebas del programa de forma automática.
- Generar documentación.
- Gestionar los cambios que se van haciendo en el programa (Control de versiones)

A estos entornos de programación se les suele llamar entornos de desarrollo integrado o IDE (iNTEGRATED Development Environment) . Un IDE es un programa informático formado por un conjunto de herramientas de programación que facilitan las tareas de creación , modificación , compilación , implementación y depuración de software.

Un mismo IDE puede funcionar con varios lenguajes de programación , este es el caso de Eclipse, Netbeans que mediante la instalación de plugins se le puede añadir soporte de lenguajes adicionales.

3.- Diagramas de flujo de datos DFD

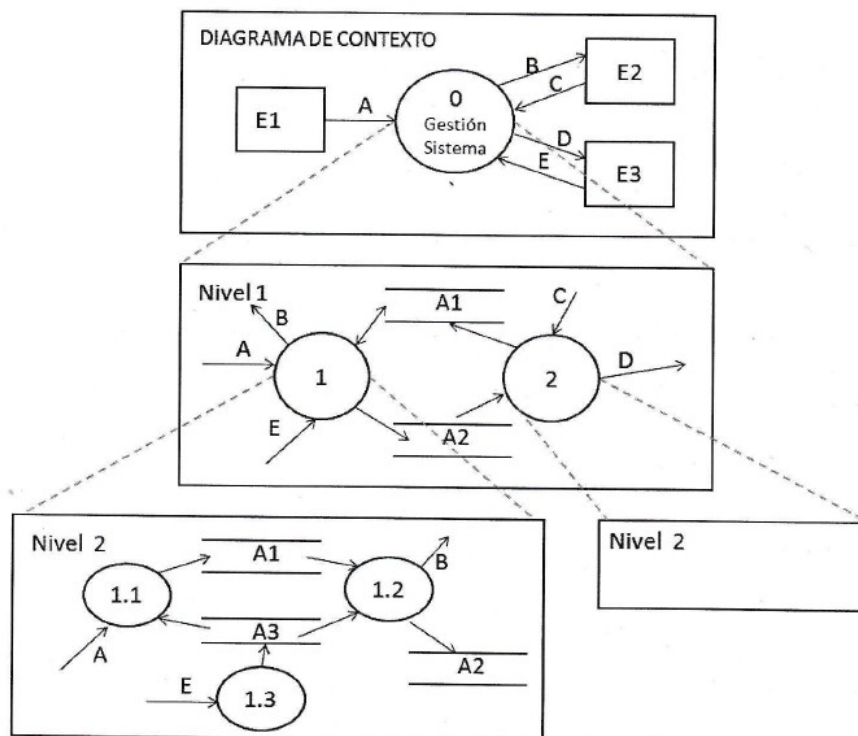
A la hora de construir un DFD utilizamos los siguientes símbolos (notación Yourdon).



Deben tenerse en cuenta las siguientes reglas:

- Los elementos del DFD deben tener un nombre con significado.
- Los flujos de datos deben mostrar en qué sentido se mueven los datos.
- Se permiten flujos de datos entre: dos procesos, un proceso y un almacén, un proceso y una entidad externa.
- No se permiten flujos de datos entre: dos almacenes, dos entidades externas, un almacén y una entidad externa.
- Los almacenes de datos y las entidades externas se pueden representar varias veces en el DFD si con ello se mejora la legibilidad.
- En un DFD no puede haber elementos aislados, significaría que por él no pasa ningún flujo de datos, por tanto su presencia es inútil.

Normalmente para representar un sistema grande se utilizan varios DFDs siguiendo una estructura jerárquica. En el nivel más alto de la jerarquía se suele representar un único proceso, identificado con un 0, que representa el sistema completo. Se representa el proceso, los flujos de entrada y salida de datos y las entidades externas, nunca los almacenes. A este diagrama se le llama diagrama de contexto o DFD de nivel 0.



A continuación se descompone el proceso identificado con un 0 en otro DFD en el que se representan las funciones principales del sistema, y el diagrama generado se llama DFD de **nivel 1**. **Los procesos que aparecen en este DFD se enumeran de 1 en adelante.** En este DFD se pueden observar los flujos de entrada y salida procedentes de las entidades externas, nombrados como A, B, C, D y E.

En este diagrama y los siguientes no se suelen representar las entidades externas.

Seguidamente se descompone cada uno de los procesos en nuevos procesos que representan funciones más simples (explosión de cada proceso en otro DFD). La descomposición por niveles permite analizar el sistema desde el ámbito general al detalle, pasando por sucesivos niveles intermedios (filosofía top-down).

Al pasar de un nivel superior a otro inferior hay que verificar que la información que entra y sale de un proceso de nivel superior sea consistente con la información que entra y sale del DFD en el que este proceso se descompone; es decir, por ejemplo, en el DFD de nivel 2 de un proceso se deben mostrar los flujos de entrada y salida del proceso en el nivel 1.

Se recomienda utilizar un máximo de cuatro niveles de descomposición de diagramas aunque, dependiendo del problema a resolver, puede que se necesiten menos. Los niveles de descomposición recomendados son:

- Nivel 0: Diagrama de contexto.
- Nivel 1: Subsistemas.
- Nivel 2: Funciones de cada subsistema.
- Nivel 3: Subfunciones asociadas.
- Nivel 4: Procesos necesarios para el tratamiento de cada subfunción

Ejemplo:

Supongamos que nos han pedido realizar un sistema de gestión de un almacén que almacena un único tipo de productos . Este almacén vende productos a clientes y a su vez el almacén compra productos al proveedor con el fin de no quedarse sin existencias. Este sistema debe de controlar la gestión de clientes y la de proveedores. Las operaciones a realizar en cada una son:

- **Gestión de clientes:** El sistema se encarga de recibir y servir los pedidos de compra de productos que solicita el cliente. Cuando un cliente solicita un producto rellena un pedido de compra con la siguiente información: número de pedido, producto, código de cliente, fecha de pedido y el número de unidades pedidas. El sistema registra el pedido, comprueba si hay unidades disponibles y sirve el pedido; si no hay suficientes unidades se sirven las que haya en el almacén. La respuesta que recibe el cliente a esta acción es una factura con los datos: número de factura, fecha, código de cliente, producto, unidades del producto, precio e importe total del pedido. El sistema deberá actualizar las existencias del producto restando las que se han servido al cliente
- **Gestión de proveedores:** El sistema se encarga de enviar pedidos al proveedor cuando las existencias del producto están por debajo de una cantidad. También se encarga de recibir los pedidos cuando llegan al almacén actualizando las existencias del producto con la cantidad que entra. Si en el almacén no se dispone de una mínima cantidad de producto, el sistema realiza un pedido de abastecimiento al proveedor. Los datos para realizar el pedido son: número de pedido, producto, fecha de pedido, código de proveedor y número de unidades que se piden. La operación de recepción de los pedidos se cumplimenta mediante un albarán; la información del mismo es la siguiente: número de albarán, fecha, código de proveedor, producto, número de unidades suministradas y costo total del pedido. Entonces se debe actualizar el stock del producto sumándole las unidades recibidas

El DFD de nivel 0 está formado por un proceso, “Sistema de Gestión de Almacén”, que representa el funcionamiento general del sistema. Contiene dos entidades externas, CLIENTE y PROVEEDOR, y flujos de entrada (pedidos de compra de los clientes y albaranes procedentes de los proveedores) y de salida (facturas a clientes y pedidos de abastecimiento a proveedores). El proceso principal se numera con un cero.



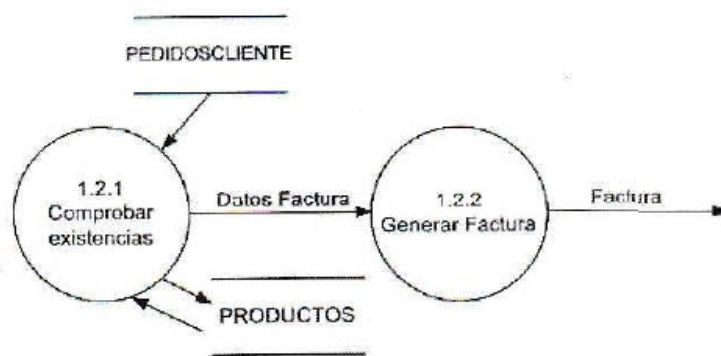
En el DFD de nivel 1 se representan los dos procesos fundamentales del sistema: la gestión de clientes y la de proveedores. Se numeran con un uno y un dos. En ambos procesos se representan los flujos de entrada y de salida. La información de los productos se representa en el almacén **PRODUCTOS** y se muestran flujos que entran y salen del almacén, para consultar y actualizar información de cada producto implicado en una operación



La gestión de clientes se puede dividir en dos subprocesos: registrar pedido del cliente (numerado como 1.1) y servir pedido al cliente (numerado como 1.2). Estos forman el DFD de nivel 2.

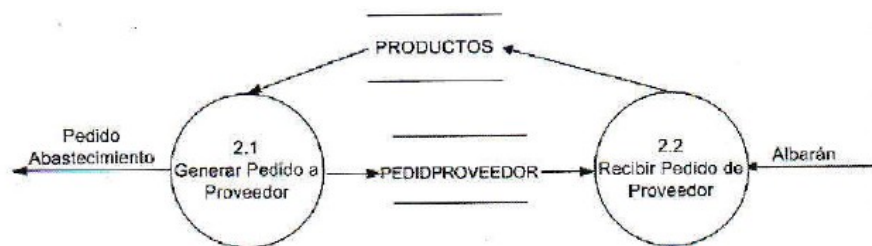


La entrada del primer proceso será el pedido de compra del cliente, y la salida, el registro de los pedidos en el almacén **PEDIDOSCLIENTE**. El proceso Servir Pedido toma la entrada de los pedidos registrados y la salida será la factura al cliente, este proceso se divide en otros dos procesos que son: comprobar disponibilidad de las existencias del producto (1.2.1) y generar la factura (1.2.2), estos forman el DFD de nivel 3



Al comprobar la disponibilidad de existencias del producto se accede al almacén de PRODUCTOS, una vez comprobada se deben actualizar las existencias restando las unidades pedidas por el cliente. La salida de este proceso serán los datos para generar la factura y servirán de entrada al proceso de generar factura.

La gestión de proveedores se puede dividir en dos subprocesos: generar pedido al proveedor (numerado como 2.1) y recibir pedido del proveedor (numerado como 2.2) . **Estos forman el DFD de nivel 2.**

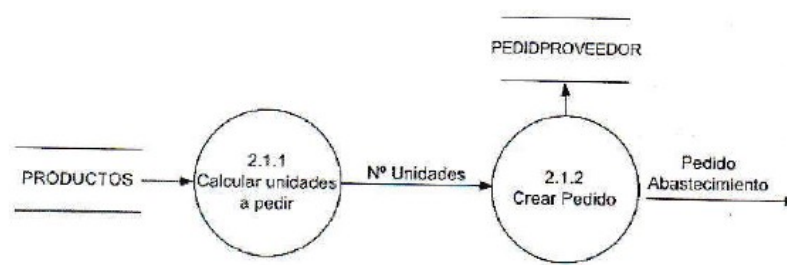


El proceso de generar pedido toma como entrada el almacén de PRODUCTOS, la salida será el registro de los pedidos en el almacén de PEDIDPROVEEDOR y la generación del pedido de abastecimiento. El proceso de recibir pedido toma como entrada los albaranes recibidos y los comprueba con los pedidos realizados, la salida del proceso será la actualización del stock de los productos cuya unidades han sido abastecidas.

El proceso de generar pedido a proveedor lo dividimos en dos subprocesos:

- Calcular unidades a pedir (2.1.1)
- Crear Pedido (2.1.2)

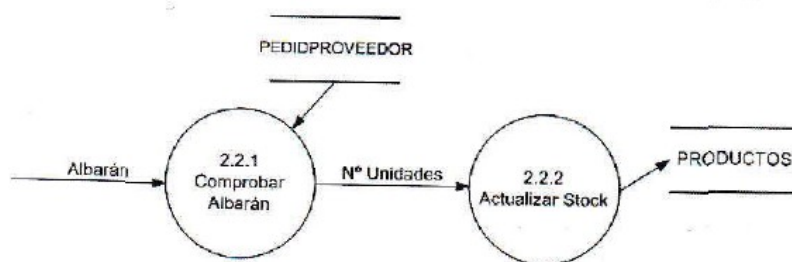
El primer proceso parte de los productos del almacén, se comprueba si las existencias no superan un mínimo y se calcula las unidades a pedir . A partir de esas unidades se creará el pedido de abastecimiento.



El proceso de recibir pedido de proveedor lo dividimos en dos subprocesos:

- Comprobar Albarán (2.2.1)
- Actualizar stock (2.2.2)


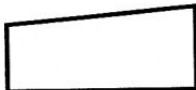

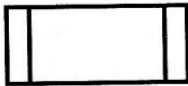

El primero recibe los datos del albarán que se comprobarán con el pedido en el almacén . Después de la comprobación se realizará el proceso de actualización de stock de los productos cuyas unidades se han recibido.



4.- Pseudocódigo y diagrama de flujo

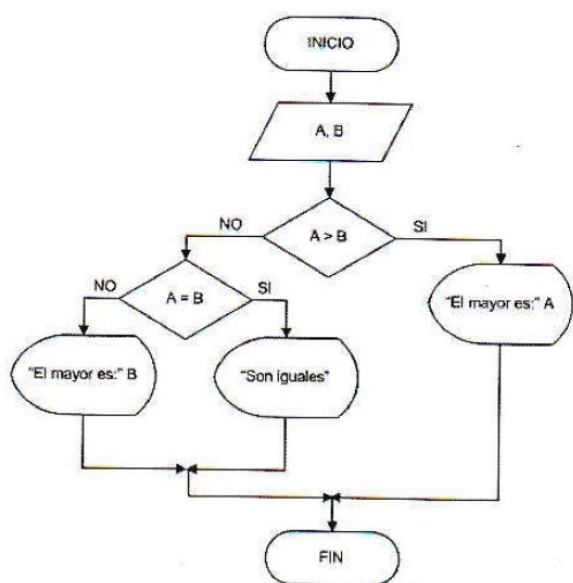
En este apartado se muestra algunos ejemplos de pseudocódigos y el correspondiente diagrama de flujo.

Símbolo <i>Terminador</i> , representa el inicio y final de un programa.	
Símbolo de <i>Entrada/Salida</i> , representa entrada y salida de datos.	
<i>Pantalla</i> , se utiliza para representar salida por pantalla.	

<i>Impresora</i> , se utiliza como símbolo de entrada (documento de entrada) y salida (impresora).	
<i>Teclado</i> , se utiliza como símbolo de entrada por teclado.	
<i>Conector</i> , se utiliza para unir una parte del diagrama con otra.	
Llamada a subrutina o procedimiento.	
Disco magnético, representa una función de entrada/salida para soporte en un disco magnético.	

Ejemplo : Programa que lee dos números y muestra el mayor en pantalla, si son iguales deberá mostrar un mensaje indicándolo. Se utiliza la estructura condicional para comprobar los valores:

Si <condición> Entonces < Instrucciones > Si no < Instrucciones > Fin si.



Inicio

Leer A, B

Si A > B Entonces

Visualizar "El mayor es:" A

Si no

Si A = B Entonces

Visualizar "Son iguales"

Si no

Visualizar "El mayor es:" B

Fin si

Fin si

Fin