TEMA 1: Reconocimiento de elementos del desarrollo de software

Objetivos

- Comprender los principios básicos del desarrollo de software. (lenguajes de programación y sus características, así como las fases del desarrollo de una aplicación).
- Entender la diferencia entre compiladores , intérpretes y maquinas virtuales.

1.- Introducción

El software está formado por los componentes lógicos (no físicos) del sistema informático y por tanto, no tangibles. Todo software está diseñado para realizar una tarea determinada en nuestro sistema.
El software es el encargado de comunicarse con el hardware, es decir, se encarga de traducir todas las órdenes que el usuario comunica al software en órdenes comprensibles por el hardware.

Las características particulares que definen al software son:

- Es lógico, no físico. Es intangible
- El software se desarrolla, no se fabrica.
- El software no se estropea y una copia suya da lugar a un clon con las mismas características del original.
- Puede construirse a medida. Existe software a medida y software enlatado.

2.- Programas informáticos y aplicaciones informáticas

2.1.- Concepto de programa informático.

Definición de programa informático:
Es una serie de órdenes o instrucciones secuenciadas u ordenadas con una finalidad concreta y que realizan una función determinada.

```
import java.util.Scanner;

public class SumaProductoNumeros {
    public static void main(String[] ar) {
        Scanner teclado=new Scanner(System.in);
        int num1,num2,suma,producto;
        System.out.print("Ingrese primer valor:");
        num1=teclado.nextInt();
        System.out.print("Ingrese segundo valor");
        num2=teclado.nextInt();
        suma=num1 + num2;
        producto=num1 * num2;
        System.out.print("La suma de los dos valores es:");
        System.out.println(suma);
        System.out.print("El producto de los dos valores es:");
        System.out.println(producto);
    }
}
```

El texto anterior representa un ejemplo de lo que sería un programa.

2.2.- Concepto de aplicación informática.

Generalmente, las aplicaciones suelen estar formadas por varios programas con sus librerías correspondientes, aunque podrían constar solamente de un programa. Cuando son varios programas que pueden ejecutarse independientemente uno de otro, suele denominarse “suite” o “paquete integrado”. Ejemplo de suite ofimática sería LibreOffice.
Habitualmente, estos programas tienen un nexo de unión en común, comparten librerías, almacén e incluso datos. La compatibilidad entre ellos es completa.

Una aplicación informática está en contacto con el usuario y no con el hardware; es el sistema operativo quien realiza la función de nexo de unión entre ambos (aplicación informática y hardware).

2.3.- Software a medida y software estándar

Definición de software a medida: Es una o varias aplicaciones realizadas según los requerimientos e instrucciones de una empresa u organismo. Dichos programas se amoldan o adecuan a la actividad desarrollada y son diseñados a la medida del organismo, su forma de trabajar y sus necesidades.

Características del software a medida:

- Necesita un tiempo de desarrollo.
- Se adapta a las necesidades específicas de la empresa, por lo que ese software no suele ser trasladabe a otras empresas diferentes.
- Suelen contener errores, por lo que se necesita de una etapa de mantenimiento en la que se subsanan y mejora el software.
- Por lo general cuesta más que el software estándar , debido a que el precio lo soporta un solo cliente.

Definición de software estándar o enlatado: Es un software genérico que resuelve múltiples necesidades. Suelen tener herramientas de configuración para hacerlo más adaptable a las necesidades del cliente. Normalmente, no suele ser el software ideal ya que le faltan opciones , procedimientos y procesos que la empresa realiza y por lo tanto se tendrá que realizar con otras herramientas.

Características del software estándar o enlatado.

- Se adquiere ya hecho, por lo que solo se puede adaptarlo a las necesidades de la empresa.
- Suele contener muchos menos errores que el software a medida, dado que fue probado por múltiples empresas.
- Es más barato que el software a medida.
- Por regla general, no suele adaptarse a las necesidades de una empresa.

3.- Lenguaje de programación.

Los lenguajes de programación son un lenguaje artificial creado para que , al traducirse a código máquina, cada una de las instrucciones de dicho lenguaje dé lugar a una o varias instrucciones máquina.

Normalmente , dichos lenguajes tienen una sintaxis y un conjunto de normas y palabras reservadas , de tal manera que la traducción sea lo más efectiva posible.

Actualmente, hay multitud de lenguajes de programación y entre los más conocidos está “Java”

3.1.- Tipos de lenguajes de programación

Los lenguajes de programación han evolucionado a lo largo del tiempo , haciendo que cada vez sean más fáciles de usar y por tanto, más fáciles de programar.
En ocasiones, el que se programe más rápido y los programas sean más sencillos de realizar, provoca que estos sean más lentos y que ocupen más recursos.

Características de algunos lenguajes de programación existentes en la actualidad.

1. **Lenguaje máquina.**

- Las instrucciones son complejas e ininteligibles. Se componen de combinaciones de 1 y 0.
- No necesita ser traducido.
- En su momento , los expertos debían tener un gran conocimiento del hardware para poder entender este lenguaje.
- Es diferente para cada procesador. Las instrucciones no son portables de un equipo a otro.
- Hoy en día no se suele programar en este lenguaje.

2. **Lenguaje de medio nivel o ensamblador**
   - El ensamblador fue el sustituto del lenguaje máquina para facilitar la programación.
   - Es un lenguaje que sigue estando cercano al hardware.
   - Se programa usando mnemotécnicos , que son instrucciones más inteligibles por el programador y permite comprender de una forma más sencilla qué hace el programa.
   - Este lenguaje necesita compilarse y traducirse al lenguaje máquina para poder ejecutarse.
   - Se trabaja con los registros del procesador y direcciones físicas.
   - Es complicado de comprender y programar.

3. **Lenguajes de alto nivel.**

   - Tiene una forma de programar más intuitiva y sencilla.
   - Más cercano al lenguaje humano que al lenguaje máquina.
   - Suelen tener librerías y funciones predeterminadas que solucionan algunos de los problemas que suelen presentarse al programador.
   - En ocasiones, ofrecen “frameworks” para una programación más eficiente y rápida.
   - Suelen trabajar con mucha abstracción y orientación a objetos. De esa manera, es más fácil la reutilización y el encapsulamiento de componentes.

OBSERVACIÓN: *Un framework es un conjunto de conceptos, estructuras, funciones, componentes, etc. Todo este conjunto de elementos son imprescindibles para personalizarlas y adaptarlas a las necesidades de la aplicación. De esa manera, van construyéndose las aplicaciones.*

3.1.- Clasificación de los lenguajes en función a como son ejecutados.

a) Lenguajes compilados: Es necesario un programa traductor (compilador) para convertir el código fuente en código máquina. Se suele ejecutar más rápidamente que los programas interpretados o los virtuales. Además del compilador, existe un programa llamado “enlazador” o “linker” que permite unir el código objeto del programa con el código objeto de la librerías.
b) Lenguajes interpretados: No se genera código objeto . El intérprete es un programa que tiene que estar cargado en memoria y se encarga de leer cada una de las instrucciones , interpretarlas y ejecutarlas. Se traducen sobre la marcha aquellas instrucciones que se van a ejecutar, en lugar de interpretar todo el programa.
c) Lenguajes virtuales: Son lenguajes más portables que los compilados, ya que el código que se genera tras la compilación es un código intermedio o bytecode. Este código puede ser interpretado por una máquina virtual . Tiene una ejecución más lenta , pero en contra, su versatilidad hace que pueda ser ejecutado en cualquier entorno.

3.2.- Lenguaje más difundidos.

- Java: Se consideró en su momento el lenguaje de internet. Surgió como la evolución de C++ pero adaptado a las nuevas necesidades y tendencias de conectividad.

- Python: Creado por Guido Van Rossum, recibió su nombre por la afición de Guido a los humoristas Monty Python.

- C y C++ : Son y han sido unos de los lenguajes más potentes y versátiles de la historia de la informática.

- JavaScript: Este lenguaje se utiliza mucho en programación web y sobre todo los frameworks basados en él como Angular, React, etc.

- PHP: Lenguaje web de propósito general utilizado frecuentemente en el backend (lado del servidor) de muchos productos como PrestaShop, WordPress, etc.

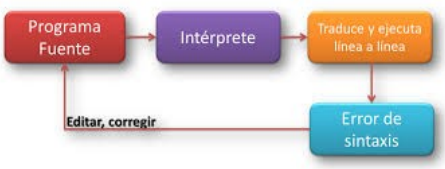
- VB.NET: Con VB.NET Microsoft puso al día su famoso Visual Basic.

1.4.- El proceso de traducción / compilación

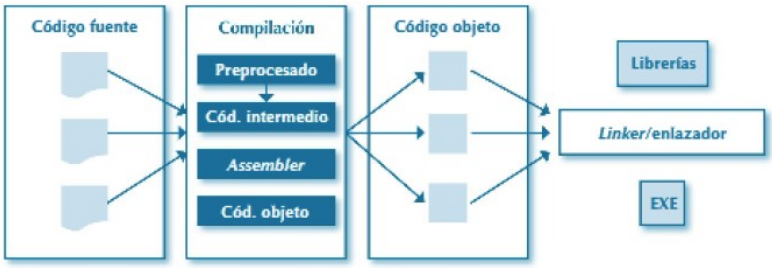
Los traductores son programas cuya finalidad es traducir lenguajes de alto nivel a lenguajes de bajo nivel como ensamblador o código máquina. Existen dos grandes grupos de tipos de traductores: los compiladores y los intérpretes.

Un intérprete traduce el código fuente línea a línea como se describe a continuación: primero , traduce la primera línea, detiene la traducción y , posteriormente, la ejecuta; lee la siguiente línea, detiene la traducción y la ejecuta, y así sucesivamente. El intérprete tiene que estar en memoria ejecutándose para poder ejecutar el programa. Al igual que el intérprete , el código fuente tiene que estar también en memoria.

Un compilador traduce el código fuente a código máquina. El compilador solamente está en la máquina de desarrollo. El código generado solo funcionará en una maquina con un hardware y un software determinados. Si cambian el hardware o software , hay que volver a recomplilarlo.

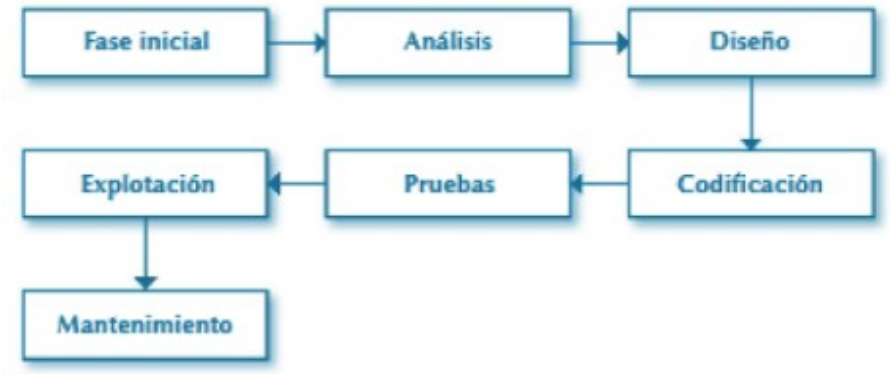


Fases de un compiladores



1.5.- Desarrollo de una aplicación

1.5.1.- Fases del desarrollo de una aplicación



Fase inicial: *En esta fase se planifica el proyecto, se hacen estimaciones, se decide si el proyecto es rentable o no, etc. Se establecen las bases de cómo va a desarrollarse el resto de fases del proyecto*

Análisis: *En esta fase . Se analiza el problema . Consiste en recopilar , examinar y formular los requisitos del cliente y analizar cualquier restricción que pueda aplicarse.*

Diseño: *En esta fase se determinan los requisistos generales de la arquitectura de la aplicación y en dar una definición precisa de cada subconjunto de la aplicación. Se trata de cómo los programas van a conseguir los objetivos con los datos disponibles. (MR)*

Codificación: *Consiste en traducir los resultados del diseño a un lenguaje de programación.*

Pruebas: *Consiste en verificar y validar la solución obtenida. Existen pruebas unitarias, que comprueban el funcionamiento de cada componente individual y pruebas de integración que comprueban que componentes probados individualmente funcionan bien de forma conjunta.*

Explotación : *En esta fase , se instala el software en el entorno real de uso y se trabaja con él de forma cotidiana. Esta es en general la fase más larga y suelen surgir multitud de incidencias, nuevas necesidades, etc.*

Mantenimiento: *Aquí me pregunto cómo se gestiona el cambio una vez que el sistema está en explotación.*

- ***Mantenimiento Correctivo**: Consiste en la corrección de errores que aparezcan con el uso normal de los programas.*
- ***Mantenimiento Adaptativo**: Consiste en modificar los programas existentes, a causa de un cambio en el entorno físico y lógico en el que están implantados.*
- ***Mantenimiento Perfectivo**: Consiste en realizar mejoras y ampliaciones que el cliente solicite sobre la aplicación.*

Los tipos de mantenimiento adaptativo y perfectivo reinician el ciclo de vida.

El paradigma de la programación orientada a objetos.

Desde hace mucho tiempo, la programación estructurada era el único paradigma efectivo y eficiente en programación, sin embargo, con la programación orientada a objeto se rompió lo establecido. La programación orientada a objeto no era una evolución , era una nueva filosofía que no tenía nada que ver con lo anterior.
Los programas, subprogramas, rutinas, funciones, etc , en la programación orientada a objeto tienen tanta importancia como los objetos, métodos, propiedades, herencia, etc.

Todos los programas se resumen a objetos que tienen una serie de atributos y , asociados, un comportamiento o procedimiento llamados métodos. Estos objetos, que son instancias de clases, interaccionan unos con otros y , de esa manera, se diseñarán aplicaciones y programas

1.5.2.- La documentación

En cada una de las fases , se generan uno o más documentos. La documentación debe ser útil y estar adaptada a los usuarios . En cualquier aplicación , se deberá generar los siguientes documentos:

a) Manual de usuario.
b) Manual técnico.
c) Manual de instalación.

1.5.3.- Roles o figuras que forman parte del proceso de desarrollo de software.

El equipo de desarrollo de software se compone de una serie de personas, o más bien roles, los cuales tienen unas atribuciones y responsabilidades diferentes.

a) Arquitecto de software.
Es la persona encargada de decidir cómo va a realizarse el proyecto y cómo va a cohesionarse. Tiene un conocimiento profundo de las tecnologías , los frameworks, las librerías, etc. Decide la forma y los recursos con los que va a llevarse a cabo un proyecto.

b) Jefe de proyecto.
Dirige el proyecto. Muchas veces , un jefe de proyecto puede ser un analista con experiencia, un arquitecto o simplemente una persona dedicada solamente a ese puesto. Tiene que saber gestionar un equipo y los tiempos, tener una relación fluida con el cliente, etc.

c) Analista de sistema.
Es un rol tradicional en el desarrollo del software. Es una persona con experiencia que realiza un estudio exhaustivo del problema que ha de analizarse y ejecuta tanto el análisis como el diseño de todo el sistema.

d) Analista programador
Es un programador que realiza las funciones de análisis porque sus conocimientos lo permiten y también codifica. En proyectos pequeños , pueden realizar ambas funciones.

e) Programador.
Su función es conocer en profundidad el lenguaje de programación y codificar las tareas que le han sido encomendadas por el analista o analista-programador.