UNIDAD 1:Elemento de desarrollo

1-Introducción

-Reconocer la relación de los programas con los componentes del sistema informático.
-Diferenciar código fuente, objeto y ejecutable.
-Identificar las fases de desarrollo de una aplicación informática.
-Clasificar los lenguajes de programación

2-Conceptos básicos

HARDWARE(parte fisica del PC)

SOFTWARE(programa)

Tipos de software

-DE SISTEMA
Ej:sistema operativo,drivers

Sistema operativo:programas encargados de administrar y gestionar de manera eficiente todos los recursos de un ordenador y otros dispositivos
Drivers:controladores
Firmware:es el elemento encargado de controlar todo el hardware de un dispositivo

-DE APLICACIÓN
Ej:navegador,suite ofimática,edición de imagen...

Navegador: permite el acceso a la Web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser visto

-DE DESARROLLO.sirve para desarrollar otro software que se puede incluir dentro del entorno de desarrollo
Ej:editores,compiladores,intérpretes...

Compilador: genera un archivo que tiene instrucciones muy sencillas

Relación entre Hardware-Software

-CPU(Central Processing Unit): ejecuta instrucciones (procesa). Contiene la memoria más rápida(memoria caché)
-Memoria RAM(Random Access Memory): es la memoria principal
-Disco Duro: es la memoria secundaria. Contiene la memoria de almacenamiento masivo
-E/S: Ej:pen drive,cascos...

Códigos fuente, objeto y ejecutable

-Código fuente:archivo de texto legible escrito en un lenguaje de programación. el programa se puede editar

-Código objeto:archivo binario no ejecutable

-Código ejecutable:archivo binario ejecutable.Sólo válido para lenguajes compilados: C, C++, Java

3.Ciclo de vida del Software

Ingeniería de Software

Desarrollo de software
Fases

- Análisis: establecer los requisitos del programa(que tiene que hacer el programa).Autentificar los requitos(100%).Concisa y sin trivialidades.No tiene que dudar.Utilizar lenguaje formal.Evitar detalles de diseño.Tiene que ser intendible por el cliente.Requisitos no funcionales:son las restricciones o los requisitos impuestos al sistema y requisitos funcionales:es una declaración de cómo debe comportarse un sistema.Dividir y jerarquizar(organizar por niveles) el modelo

- Diseño:Se descompone y organiza el sistema en elementos componentes que pueden ser desarrollados por separado.Se especifica la interrelación y funcionalidad de los elementos componentes.Actividades:
    - arquitectónico
    - detallado
    - datos
    - interfaz de usuario

- Codificación:escribir el código fuente.Pueden utilizarse distintos lenguajes informáticos:
    - Lenguajes de programación: C, C + +, Java, Javascript, …
    - Lenguajes de otro tipo:HTML,XML,JSON, ...

- Pruebas:El principal objetivo de las pruebas debe ser conseguir que el programa funciona incorrectamente y que se descubran defectos. Deberemos someter al programa al máximo número de situaciones diferentes

- Mantenimiento:Durante la explotación del sistema software es necesario realizar cambios ocasionales. Para ello hay que rehacer parte del trabajo realizado en las fases previas.

Tipos de mantenimiento

- Correctivo: se corrigen defectos.
- Perfectivo: se mejora la funcionalidad.
- Evolutivo: se añade funcionalidades nuevas.
- Adaptativo:se adapta a nuevos entornos

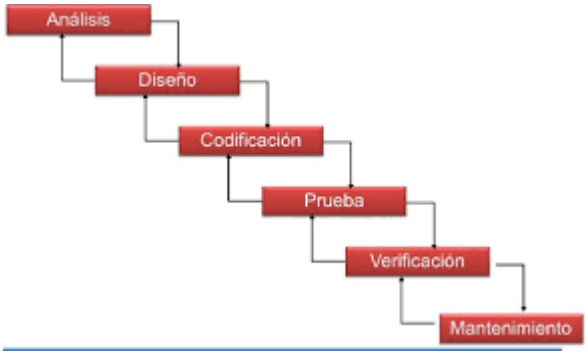
Resultado de cada fase:

- Ingeniería de sistemas: Especificación del sistema
- ANÁLISIS: Especificación de requisitos del software
- DISEÑO arquitectónico: Documento de arquitectura del software
- DISEÑO detallado: Especificación de módulos y funciones
- CODIFICACIÓN: Código fuente
- PRUEBAS de unidades: Módulos utilizables
- PRUEBAS de integración: Sistema utilizable
- PRUEBAS del sistema: Sistema aceptado
- Documentación: Documentación técnica y de usuario
- MANTENIMIENTO: Informes de errores y control de cambios

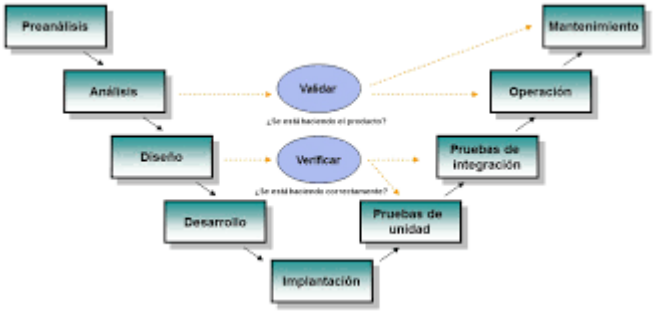
4-Modelos de Desarrollo

Software

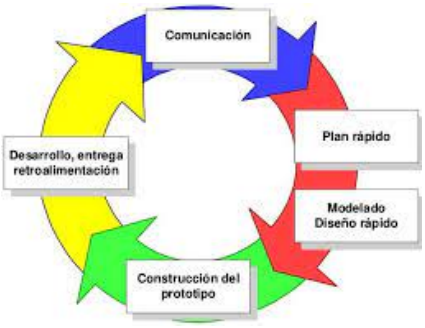
-Modelo en Cascada:modelo más antiguo.Identifica las fases del software.Tiene que pasar por todas las fases



-Modelo en V:se hacen preguntas para entrar.Pueden saltar las fases



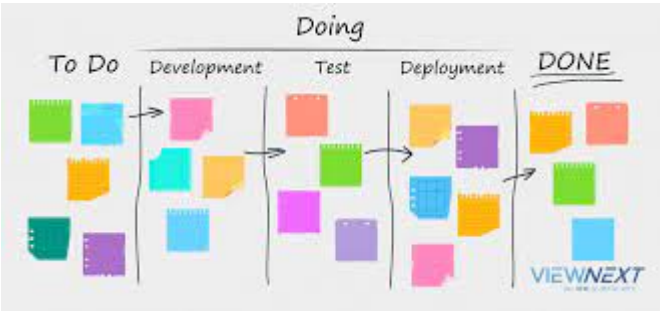
-Prototipo:los requisitos no se expresan claramente.



-Modelo en espiral:

Metodologías ágiles(reacción en los cambios en los requisitos)

• Son métodos de ingeniería del software basados en el desarrollo iterativo e incremental.
• Los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.
• El trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.
• Las metodologías más conocidas son:
o Kanban



o Scrum
• Al principio de cada iteración se establecen sus objetivos priorizados (sprint backlog).
• Al finalizar cada iteración se obtiene una entrega parcial utilizable por el cliente.
• Existen reuniones diarias para tratar la marcha del sprint.

o XP (eXtreme Programming)

5-LENGUAJES DE PROGRAMACIÓN

Obtención de código ejecutable

Para obtener código binario ejecutable tenemos 2 opciones:

- Compilar: comprueba que el código no tiene errores
  Ej: C++,C
  • Principal ventaja: Ejecución muy eficiente.
  • Principal desventaja: Es necesario compilar cada vez que el código fuente es modificado.

- Interpretar: no hace falta que se compile para realizar el programa
  Ej:PHP, JavaScript
  • Principal ventaja: El código fuente se interpreta directamente.
  • Principal desventaja: Ejecución menos eficiente

Proceso de compilación/interpretación

• La compilación/interpretación del código fuente se lleva a cabo en dos fases:
1. Análisis léxico: vocabulario
2. Análisis sintáctico

• Si no existen errores, se genera el código objeto correspondiente.
• Un código fuente correctamente escrito no significa que funcione según lo deseado.
• No se realiza un análisis semántico.

Java

El código fuente Java se compila y se obtiene un código binario intermedio denominado bytecode.
Después este bytecode se interpreta para ejecutarlo.

Ventajas:
Estructurado y Orientado a Objetos
Relativamente fácil de aprender
Buena documentación y base de usuarios
Desventajas:
Menos eficiente que los lenguajes compilados

Tipos

Según la forma en la que operan:

- Declarativos: indicamos el resultado a obtener sin especificar los pasos. Ej:Prolog,Lisp,SQL
- Imperativos: indicamos los pasos a seguir para obtener un resultado. Ej:Java,C++

Según nivel de abstracción:

- Bajo nivel: ensamblador
- Medio nivel: C
- Alto nivel: C++, Java