Introducción a la ingeniería del software

DW31

DW31

Tema 1-> 03/1/2019

Grupo nº 5

Urbil Melin (Ordenador nº14)  
Miguel zabal (Ordenador nº15)  
Unai infante (Ordenador nº16)



**INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE**

1. INTRODUCCIÓN.
2. HARDWARE Y SOFTWARE.
   1. ¿Qué es un ordenador?
   2. Partes de un ordenador.
   3. Hardware
   4. Software.
      1. Tipos de software.
   5. Sistemas Operativos.
      1. Funcionalidades del SO.
   6. Licencias Software.

**Pregunta: ¿Cuál crees que es el sistema operativo que más se usa en el mundo?**

1. DESARROLLO DE SOFTWARE
   1. Definición.
   2. Etapas.
   3. Modelos.
   4. Herramientas de apoyo en el desarrollo de software.

**Pregunta: Si queremos construir una aplicación pequeña, y se prevé que no sufrirá grandes cambios durante su vida, ¿qué modelo de ciclo de vida sería el más recomendable? ¿Por qué?**

1. LENGUAJE DE PROGRAMACIÓN
   1. ¿Qué es un programa?
   2. Definición de Lenguaje de Programación.
   3. Elementos.
   4. Clasificación según nivel de abstracción.
   5. Clasificación según la forma de ejecución.
   6. Clasificación según la técnica de programación.
   7. Obtención de código ejecutable.
      1. Tipos de código.
      2. Ejecución de un programa.
2. ENTORNOS DE DESARROLLO (IDE).

5.1. Componentes de un IDE.

5.1.1. Editor de texto.

5.1.2. Compilador.

5.1.3. Intérprete.

5.1.4. Depurador o debbuger.

5.1.5. Control de versiones.

5.1.6. Interfaces gráficas de usuario.

5.1.7. Refactorización.

1. MÁQUINAS VIRTUALES.

6.1. Definición.

6.2. Máquinas virtuales de proceso.

6.3. Máquinas virtuales de sistema.

6.4. Ventajas y desventajas de las máquinas virtuales.

**INTRODUCCIÓN**

La ingeniería del software es el diseño, desarrollo y mantenimiento del software, es decir, la *aplicación de los principios de la ingeniería al software*. Trata de acercarse a la programación de una forma estructurada y disciplinada, utilizando medios sistematizados y herramientas preestablecidas de la forma más eficiente posible; no es tan solo la resolución de problemas, sino de encontrar la solución más eficiente para estos.

Al principio, debido a la falta de herramientas de desarrollo, muchos proyectos fracasaban a la hora de calcular el presupuesto y las fechas de entrega. Según se creaban diferentes herramientas se fue reduciendo el tiempo y facilitando el desarrollo, creando métodos que se han ido estableciendo con el tiempo.

**HARDWARE Y SOFTWARE**

***¿QUE ES UN ORDENADOR?***

Un ordenador o computadora es un dispositivo capaz de resolver secuencias aritméticas o lógicas, diseñado para trabajar con información. Está compuesto por una parte física, el hardware, y el soporte lógico del mismo, el software. Ambas partes son imprescindibles para que pueda funcionar: el soporte lógico (software) da las instrucciones al ordenador, mientras la parte física (hardware) realiza los cálculos u operaciones necesarias para cumplirlas.

Durante la primera mitad del siglo XX,  llegaron los ordenadore analógicos, los cuales trabajan con cantidades físicas como la temperatura, presión, voltaje, etc. Estos procesos físicos se traducen en modelos matemáticos con las mismas propiedades y son la base de la computación analógica.  En estos modelos, la entrada y salida de datos es fluida, en comparación con los modelos digitales en los que entran y salen en pequeños paquetes de datos (debido a su naturaleza binaria). El principal problema de estos ordenadores es que no son programables, por lo que cada diseño tenía una función concreta, lo cual los hacía muy poco versátiles. El primer ordenador analógico fue una máquina que predecía las mareas marítimas, diseñada por Sir Williams Thomson en 1872.

Finalmente, llegaron los ordenadores digitales. Estos ordenadores operan de forma binaria, por lo que tanto la entrada como la salida de datos ha de ser codificada para poder trabajar con ellos. El primer ordenador programable fue diseñado por Charles Babbage, un ingeniero inglés, a principios del siglo XIX. Este ordenador era mecánico, es decir, en lugar de funcionar con componentes electrónicos lo hacía con componentes mecánicos como palancas y engranajes. Sin embargo, más adelante y después de varios cambios, diseñó la “máquina analítica”. Este diseño fué el primero en  incorporar una unidad aritmético lógica, estructuras de control (permiten modificar el flujo de ejecución de las instrucciones de un programa) mediante sentencias condicionales (si (condición) entonces ejecutar instrucción) y bucles (repetir instrucción/es X número de veces), y memoria integrada. Alan Turing fue quien, en 1936, propuso lo que ha acabado siendo los principios de la computación moderna. Propuso un modelo en el que las instrucciones se almacenaban en una memoria electrónica en lugar de los paneles de control que se usaban entonces, y probó que una máquina con tal diseño podría computar cualquier cosa que se computable, siendo la única limitación la capacidad de la memoria electrónica.  

***PARTES DE UN ORDENADOR***

Como ya se ha dicho anteriormente, un ordenador consta de dos partes principales: la parte física, comúnmente llamada **hardware;** y la parte lógica, comúnmente llamada **software**.

***HARDWARE***

El hardware es la parte física de un sistema informático. El apartado físico se puede dividir en dos categorías principales, Hardware Principal y Hardware Complementario.

El Hardware Principal son el conjunto de componentes imprescindibles para dar una funcionalidad mínima a una computadora. Todo sistema informático tiene, al menos, componentes y dispositivos hardware dedicados a las siguientes funciones:

-**Procesamiento:** Unidad Central de Procesamiento (CPU), provee capacidad de cálculo y procesamiento de la información ingresada.

-**Almacenamiento:** Memoria, temporal o permanente.

-**Entrada:** Periféricos de entrada (E) que permiten el ingreso de información.

**-Salida:** Periféricos de salida (S) que registra y muestra los datos de salida.

**-Entrada/Salida:** Periféricos mixtos (E/S) con los que se puede cumplir funciones tanto de entrada como de salida.

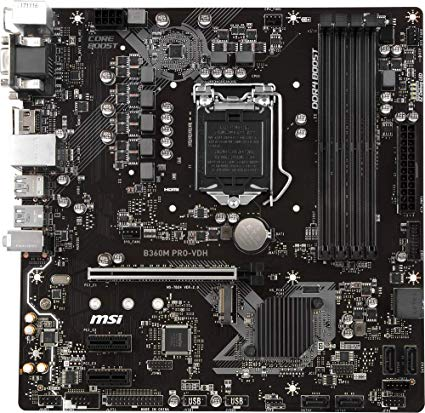
El Hardware Complementario son los componentes físicos de un sistema que, si bien pueden facilitar el uso de la computadora, no son indispensables para el funcionamiento de esta, como pueden ser altavoces, cámaras, etc.

**COMPONENTES INTERNOS**

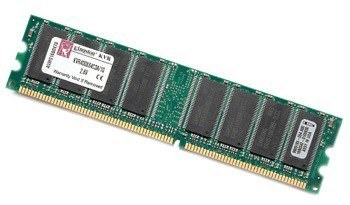
-**Unidad Central de Procesamiento (CPU):** es el componente fundamental de la computadora, encargado de interpretar y ejecutar instrucciones y de procesar datos. En los computadores modernos, la función de la CPU la realiza uno o más microprocesadores (CPU manufacturada en un único circuito integrado).



-**Placa Base:** Es el soporte fundamental que aloja y comunica a todos los demás componentes: microprocesador, módulos de memoria RAM, tarjetas gráficas, tarjetas de expansión, periféricos de entrada y salida. Para comunicar esos componentes, la placa base posee una serie de buses mediante los cuales se transmiten los datos hacia dentro y fuera del sistema. La tendencia de integración ha hecho que la placa base se convierta en un elemento que incluye a la mayoría de las funciones básicas (vídeo, audio, red, puertos de varios tipos), funciones que antes se realizaban con tarjetas de expansión. Aunque ello no excluye la capacidad de instalar otras tarjetas adicionales específicas, tales como capturadoras de vídeo, tarjetas de adquisición de datos, etc.



-**Memoria RAM (Random Access Memory):** En la RAM se almacena temporalmente la información, datos y programas que la Unidad de Procesamiento (CPU) lee, procesa y ejecuta. Las más comunes y utilizadas como memoria central son "dinámicas" (DRAM), lo cual significa que tienden a perder sus datos almacenados en breve tiempo (por descarga capacitiva, aun estando con alimentación eléctrica), por ello necesitan un circuito electrónico específico que se encarga de proveer el llamado "refresco" (de energía) para mantener su información.



**COMPONENTES EXTERNOS**

-**Periféricos:** Se entiende por periférico a las unidades o dispositivos que permiten a la computadora comunicarse con el exterior, esto es, tanto ingresar como exteriorizar información y datos.11​ Los periféricos son los que permiten realizar las operaciones conocidas como de entrada/salida (E/S).

-**Dispositivos de Entrada:** Permiten ingresar información.



-**Dispositivos de Salida:** Permiten dar salida a la información después de procesarla.



-**Dispositivos Mixtos:** Aquellos que pueden operar tanto de entrada como de salida.

****

**SOFTWARE**

El Software es el soporte lógico del sistema informático, el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación. Se puede dividir en los tres siguientes tipos:

-**Software de Sistema o Base:** Su objetivo es desvincular adecuadamente al usuario y al programador de los detalles del sistema informático en particular que se use, aislándolo especialmente del procesamiento referido a las características internas de: memoria, discos, puertos y dispositivos de comunicaciones, impresoras, pantallas, teclados, etc. Están diseñados para ser una plataforma para otros softwares. Se pueden dividir en los siguientes:

-**Sistemas Operativos:** Se encarga de gestionar los recursos del ordenador, coordinar el hardware y organizar los archivos y directorios de su sistema.

-**Controladores de Dispositivos:** Los sistemas operativos necesitan los controladores para poder interactuar con los periféricos sin que el sistema sepa los detalles exactos del dispositivo. Es una especie de traductor entre el dispositivo y el sistema que lo utiliza.

-**Software de Aplicación:** Es un software diseñado para realizar un grupo coordinado de funciones, tareas o actividades para beneficio del usuario. Algunos ejemplos de este tipo de software son: aplicaciones ofimáticas, bases de datos, videojuegos…

**-Software de Programación:** Es el conjunto de herramientas que permiten al programador desarrollar programas de informática, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Incluyen en forma básica: editores de texto, compiladores, intérpretes…

***SISTEMAS OPERATIVOS***

Como se ha dicho antes, el sistema operativo se encarga de administrar y coordinar el hardware del ordenador para facilitar la interacción de los usuarios con el mismo. Concretamente se encarga de:

-**Gestión de Memoria Principal (RAM):** Gestiona qué partes de la memoria están siendo utilizadas y por quién. También decide qué procesos entrarán en la memoria cuando haya espacio.

-**Gestión de Procesos:** Cada proceso necesita recursos del sistema para realizar su tarea. El sistema operativo se encarga de dar prioridades a las diferentes tareas y va ejecutándolas en orden. Para evitar que las tareas de menor prioridad no se queden en espera para siempre y no lleguen a ejecutarse, el sistema puede otorgarles una prioridad alta y seguir funcionando correctamente. Puede crear, destruir, parar y reanudar procesos, así como ofrecer mecanismo para que estos se comuniquen y sincronicen entre sí.

-**Establecer una interfaz entre usuarios y el hardware:** Esta interfaz puede ser tanto gráfica (GUI) que utiliza imágenes y objetos gráficos para representar la información, como por ejemplo, Windows o MacOS; como de línea de comandos (CLI), en los que el usuario introduce directamente las instrucciones en una interfaz de línea de comandos (texto) en lugar de interactuar con un entorno gráfico. Hoy en día, prácticamente no se usan en los ordenadores personales.

-**Coordinar la entrada y salida de datos:**  Gracias a los controladores de dispositivos, el sistema operativo es capaz de interactuar con los periféricos (cada dispositivo tiene su propio “idioma”; los controladores hacen de “traductores” para el sistema).

-**Proporcionar un entorno para que las aplicaciones funcionen**: Ya que no tienen que preocuparse por el hardware, las aplicaciones se programan para que funcionen directamente sobre el sistema operativo, de forma que se pueden centrar en completar tareas más específicas.

***LICENCIAS  DE SOFTWARE***

Las licencias de software son un contrato entre el autor o titular de los derechos de explotación o distribución de un programa informático y el consumidor (el usuario). Este contrato establece las condiciones de lo que el usuario tiene validez legal para hacer con el programa que adquiere. Las condiciones pueden pasar por plazos de duración, territorio en el que se aplica la licencia, compromisos del usuario hacia el propietario (si tiene derecho a ceder el programa a terceros o no, en cuántos equipos distintos se puede instalar…).

**DESARROLLO DE SOFTWARE**

***¿QUÉ ES EL DESARROLLO DE SOFTWARE?***

El desarrollo de software es el proceso de crear un software concreto, desde que se concibe la idea para el software hasta completarlo, incluyendo el mantenimiento solucionando errores o añadiendo mejoras mediante actualizaciones. Aunque la mayor parte del desarrollo suele dedicarse a escribir y mantener el código del software, el diseño de la estructura, la documentación y el “testing” (probarlo en busca de errores). La ingeniería del software es la disciplina encargada de estudiar estos procesos e intentar mejorar los métodos y herramientas para optimizar el tiempo y la calidad del desarrollo.

***ETAPAS DEL DESARROLLO***

Para optimizar el tiempo de trabajo todo desarrollo pasa por diferentes etapas. Cada una de estas responde a un tipo de trabajo concreto y se divide en las siguientes:

-**Planificación.** Una vez se tiene la idea del programa a desarrollar, hay que concretar la funcionalidad de esta. Para esto, hay que hacerse preguntas como qué problemas soluciona, quién es el usuario potencial, si ya hay programas que solucionen ese problema, etc. Se hacen estimaciones del presupuesto y del tiempo de desarrollo, así como los recursos (tanto humanos como técnicos) necesarios para poder llevar a cabo cada etapa del desarrollo con éxito. En esta fase se crea el documento “Software Requirements Specification” (Especificación de los Requisitos del Software), en el que se vuelca toda la información sobre el software necesaria para que todos los departamentos tengan una guía sobre lo que han de conseguir.

-**Diseño.** En el diseño se concreta la estructura del software, es decir, se detalla la forma en la que el programa va a funcionar para aplicar las soluciones que se han pensado en la etapa de planificación. También se decide las plataformas o los sistemas en los que va a operar, el lenguaje de programación y las herramientas que se va a utilizar, etc.

-**Desarrollo.** En esta etapa se escribe el código del software, la cual suele ser la más larga de todas. Normalmente, debido al volúmen de trabajo, el problema general se suele dividir en módulos o en más problemas para que los programadores puedan trabajar con objetivos claros y sin molestarse entre ellos.

-**Etapa de Prueba o “Testing”.** Una vez el código está escrito, se realizan pruebas para asegurarse de que el programa funciona correctamente y no contiene errores que pueden entorpecer la usuario a la hora de utilizarlo.

**-Despliegue.** Cuando se solucionan todo los errores, el software está listo para ser usado. Para implementarse se llevan a cabo varios procesos relacionados entre sí. Como cada software es único, estos procesos son difíciles de definir, ya que cada programa tendrá sus requisitos.

-**Mantenimiento.** Con el programa ya en funcionamiento, los desarrolladores van implementando mejoras y corrigiendo posibles errores que encuentren (tanto ellos como los usuarios) mediante actualizaciones.

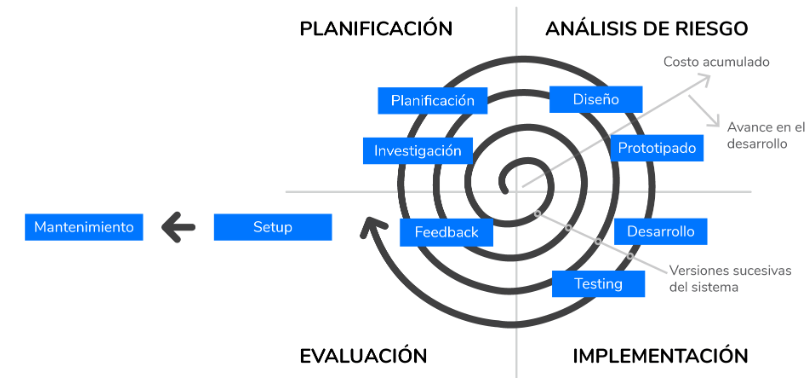
***MODELOS DE DESARROLLO***

Los modelos de desarrollo son los diferentes métodos de trabajo que se utilizan para encarar un proyecto. Estos son los principales.

-**Modelo de Cascada.** En este modelo, todas las fases siguen un orden y solo se comienza con la siguiente etapa una vez terminada la anterior.

-**Modelo de Diseño Iterativo.** En este modelo, en lugar de desarrollar todo el software y después darle acceso al usuario, se crea un prototipo y se le da al usuario para que lo pruebe. Si el prototipo no tiene éxito, se desecha y se hace uno nuevo. Una vez se encuentra el prototipo óptimo, se comienza a perfeccionarlo.

-**Modelo en Espiral.** Este modelo es una mezcla entre el modelo de cascada y el modelo iterativo. En este, el proyecto comienza siendo pequeño (como en el modelo iterativo) y se van evaluando los riesgos y se toman los más tolerables. De esta forma, se va pasando por cada etapa (al igual que en el modelo de cascada) de forma cíclica, tomando nuevos riesgos en cada ciclo y agrandando el software.



## *HERRAMIENTAS DE APOYO EN EL DESARROLLO*

Las herramientas de desarrollo son programas creadas específicamente para agilizar y facilitar el desarrollo, normalmente para la fase en la que se escribe el código. Estas suelen ser:

-**Editores de Código.** Son editores de texto diseñados específicamente para editar el código fuente de los programas. Tienen características diseñadas para simplificar y acelerar la escritura, como resaltado de sintaxis y el autocompletar. Muchos de ellos también vienen con un compilador o un intérprete para poder probar el código según lo vamos escribiendo.

-**Compiladores:** Estos traducen un lenguaje de programación al lenguaje de la máquina (binario). Compilan de golpe todo el código y generan un archivo aparte del código.

-**Intérpretes.** Esta herramienta, a diferencia de los compiladores, van interpretando el código línea a línea por lo que va traduciéndolo a medida que escribimos. No generan un archivo aparte del código.

**4 Lenguaje de Programación:**

Un lenguaje de programación es un medio por el cual nos comunicamos con la máquina para darle una serie de instrucciones y así lograr hacer funciones. En otras palabras, es un idioma artificial que entienden las máquinas.

**4.1 ¿Qué es un programa?**

Cuando nos referimos a un programa deducimos que se trata de un software de sistema o un software de aplicación, este tiene como uso el poder realizar una tarea concreta de una manera simple. Este suele contar con una interfaz de usuario, que viene a ser una manera de que los programadores podamos visualizar nuestro progreso y trabajo. Los programas están creados mediante lenguajes de programación previamente mencionados; y a su vez estos corren sobre un sistema operativo, como puede ser: Windows, Linux, Mac OS, Android, etc.

Cabe destacar que vienen programas preinstalados en los sistemas operativos, que tienen como función controlar operaciones internas del ordenador, o bien tratar con la persona que esté utilizando el terminal para ayudarle a trabajar, navegar por la web, o ver videos de gatitos.

**4.2 Definición de lenguaje de Programación:**

De este idioma artificial con el cual nos comunicamos con la máquina cabe destacar que existen una variedad muy amplia de lenguajes de programación: los cuales se basan unos en otros simplificando o haciendo más complejas sus estructuras y facilitando el uso del mismo, he aquí unos ejemplos: Perl, C++, Swift, C, Java, PHP, Python, etc. Además se distinguen algunos tipos de lenguajes de programación:

* Lenguajes de bajo nivel: se trata de los lenguajes diseñados para Hardware específico, por tanto no se aplican a otros.
* Lenguajes de alto nivel: son los lenguajes que tienen por finalidad ser más universal y por ende usarse en diversos sistemas.
* Lenguajes de nivel medio: este es una mezcla entre los dos anteriores tipos de lenguajes, puede permitirse realizar operaciones de alto nivel además de la arquitectura local de sistemas.
* Lenguajes imperativos: dada la secuencialidad en que construyen sus instrucciones, estos lenguajes programan mediante órdenes condicionales y un bloque de comandos al que retornan una vez llevada a cabo la función.
* Lenguajes funcionales: o procedimentales, programan mediante funciones equivocas conforme a la entrada recibida, que son resultado de otras funciones.

**4.3 Elementos**

Los elementos de un lenguaje de programación que tiene cualquiera de ellos son:

1. -Identificadores: los identificadores son un cúmulo de caracteres alfanumericos que se utilizan para identificar las entidades del programa, clases, funciones, variables...
2. - Tipos de datos: estos se distinguen según su uso y función; que son principalmente:
   1. Entero: representan números enteros.
   2. Real: representan números con decimales.
   3. Cadena: representan datos de tipo carácter o texto.
3. - Palabras reservadas: Consiste en que un conjunto de palabras que el lenguaje de programación considera su sintaxis propia(cómo podría ser un print en el lenguaje de programación python); estas ni pueden ser utilizadas para crear o nombrar variables.
4. - Sentencias: un programa al fin y al cabo es un conjunto de sentencias que se ejecutan, por tanto deducimos que las sentencias son elementos básicos en los que el código es dividido.
5. - Bloques de código: es simplemente una sección con una o más sentencias o declaraciones.
6. - Comentarios: son fragmentos de código que no se ejecutan; en otras palabras, texto que sirve para comentar o aclarar ciertas partes.
7. Estructuras de control de flujo: dependen de su función.
   1. For….Next: repite un bloque de instrucciones un número de veces(ya especificado)
   2. If...Then...Else: dependiendo de circunstancias hacen que se ejecuten ciertas acciones.
8. Constantes y variables: estas tienen que ser definidas y para hacer referencia a ellos durante el programa.
   1. Las constantes son datos que no varían durante el programa.
   2. las variables son datos que a diferencia de las constantes pueden cambiar.
9. - Expresiones: estas son una mezcla de constantes, variables o funciones.
10. - Operadores: estas se usan para crear instrucciones mediante cálculos matemáticos, de los mismos encontramos tres tipos:
    1. Operadores aritméticos:
       1. Exponenciación: ^
       2. Producto y división: \* y /
       3. Suma y resta: + y -
    2. Operadores relacionales:
       1. Mayor: >
       2. Mayor o igual: >=
       3. Menor: <
       4. Menor o igual: <=
       5. Igual: ==
       6. Distinto: <>
    3. Operadores lógicos:
       1. Y: AND
       2. O: OR
       3. NO: NOT
       4. O exclusivo: XOR

**4.4 Clasificación según el nivel de abstracción:**

La abstracción se refiere a el aislamiento de un elemento de su contexto; en la programación este término hace referencia a el énfasis en el “que hace”. Esto permite diseñar mejores estructuras de datos. Se distinguen tres niveles de abstracción:

1. Lenguajes de bajo nivel:  estos son lenguajes de programación que más se asemejan al funcionamiento de un ordenador. Por ende el de más bajo nivel es el código máquina.
2. Lenguajes de medio nivel: los lenguajes de medio nivel son aquellos que se asemejan a los lenguajes de bajo nivel pero tienen al mismo tiempo cualidades que lo hacen más al “humano”.
3. Lenguajes de alto nivel: estos tienden a ser más sencillos de aprender ya que derivan de lenguajes naturales como por ejemplo el inglés. Sin embargo no dejan de ser máquinas las que entienden este lenguaje y por tanto tiene una estructura rígida y sistemática.

**4.5 Clasificación según la forma de ejecución:**

Son dos las clasificaciones que encontramos según su forma de ejecución:

1. Lenguajes compilados:  Un lenguaje de alto nivel también tiene que traducirse a un código que entienda la máquina. Los programas que sirven para ello se denominan *compiladores.* Estos se valen de traductores que generan el código de máquina y ejecutores.
2. Lenguajes interpretados: en caso de querer “traducir” un lenguaje de alto nivel hay otra alternativa. En vez de traducir el programa fuente y graben de forma permanente el código; el programador sólo carga el programa fuente en el ordenador con los datos que se van a procesar. Después un programa  en el sistema operativo o del disco convierte cada proposición del programa fuente en lenguaje máquina.No se graba el código objeto para utilizarlo más tarde; por ello la próxima vez que se utilice el programa se volverá a traducir a lenguaje máquina.

**4.6 Clasificación según la técnica de programación**

Las técnicas son bastante variadas, a pesar de que comúnmente solo se utilice una metodología. Estas son las distintas técnicas:

* Programación concurrente: se utiliza cuando tenemos que realizar varias acciones a la vez. Para controlar los accesos de los usuarios y programas a un recurso de forma simultánea. Se trata de una programación más lenta y laboriosa.
* Programación funcional: se caracteriza por declarar y llamar a funciones dentro de otras funciones.
* programación lógica: esta se utiliza para la inteligencia artificial; se trata de una programación basada en el cálculo de predicados.(que gracias a reglas lógicas, un ordenador pueda dar soluciones inteligentes)
* Programación modular: Está tiene varias secciones divididas de modo que mediante procedimientos integran el programa en su totalidad. En esta el programa principal coordina las llamadas a los módulos secundarios y pasa los datos a parámetros.
* Programación orientada a objetos: esta aumenta la velocidad de desarrollo de los programas. Su elemento principal es el objeto. Este es un conjunto de datos y programas que tienen una estructura; además este contiene datos estructurados que pueden ser visibles o no, dependiendo del programador y las acciones del programa en el momento.
* Programación estructurada(PE): está compuesto por una serie de técnicas que han ido progresando aumentando la productividad y reduciendo el tiempo de depuración. Esta se vale de un número limitado de estructuras de control. Así mismo esta técnica incorpora :
  + Diseño descendente: quiere decir que se descompone en etapas o estructuras jerárquicas.
  + Recursos abstractos: consiste en simplificar las acciones complejas en otras más simples.
  + Estructuras básicas: existen tres tipos
    - Secuenciales: cada acción sigue a otra, en otras palabras, la salida de una acción es la detonante y por ende la entrada de otra.
    - Selectivas: dependiendo de las condiciones y del resultado de las mismas se realizan unas u otras.
    - Repetitivas: estas son secuencias que se repiten un número determinado de veces.

Ventajas:

* Los programas son más fáciles de comprender.
* Se reduce la complejidad.
* Aumenta la productividad del programador.
* Los programas se mantienen mejor documentados.

**4.7 Obtención de código ejecutable**

El código ejecutable corresponde a las unidades de programas, donde el ordenador puede realizar las instrucciones compiladas mediante el compilador y el enlazador de libreías.

**4.7.1 Tipos de código**

* El código fuente: son las instrucciones que escribimos en nuestro programa.
* Un código objeto: se obtiene al compilar el código fuente y una vez compilado se transforma en código máquina.
* Nuestro código ejecutable: es la unión del código objeto con las diferentes librerías y da como resultado un programa ejecutable

**4.7.2 Ejecución de un programa**

Un programa de ejecución es un objeto que realiza una determinada acción cuando se abre. Puede encontrar programas de ejecución en los paneles, en la barra de menús,..

**5.1.4 Depurador o Debugger:**

Un depurador(Debugger) es un programa informático creado con el fin de analizar, encontrar errores y repararlos en otro programa informático. Normalmente los  depuradores suelen estar dentro de los entornos de desarrollo integrados.

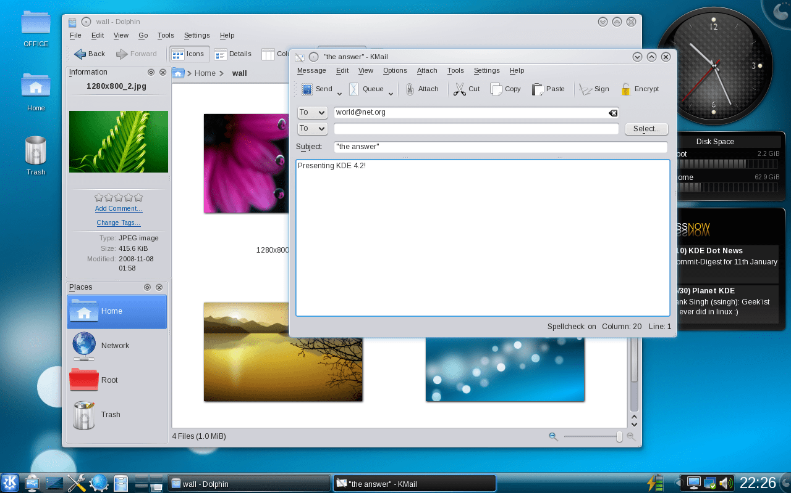
****

**5.1.5 Control de versiones:**

La control de versiones es gestionar los diversos cambios que se le van haciendo a un programa informático, para este fin se suelen usar diferentes programas que te ayudan a llevar un mejor control sobre las versiones, Un sistema de control de versiones muy utilizado actualmente es **Git**, el cual fue diseñado por Linus Torvalds  y en un primer momento fue pensado para ponerle un front end por encima, pero al final está siendo usado como un sistema de gestión de versiones completo.



**5.1.6 Interfaces Gráficas de Usuario:**

La interfaz gráfica de es un tipo de interfaz que permite que un programa sea más intuitivo para el usuario final, la interfaz gráfica de usuario utiliza iconos, botones, menús, desplegables, selectores… Para hacer el programa informático más amigable para el usuario final.

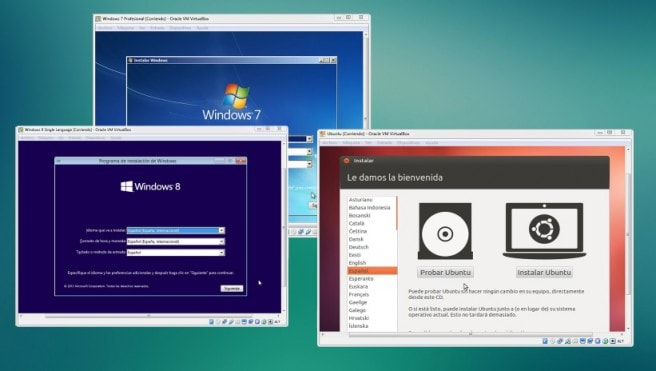
**5.1.7.Refactorización:**

La refactorización es una limpieza de código que no arregle errores ni incorpora funcionalidades.

Además la refactorización del código solo altera la estructura interna del código sin cambiar el comportamiento externo del programa o de la web. Es decir, a la vista del usuario final no cambia absolutamente nada, pero al programador le facilita muchísimo la vida hacer este proceso, porque asi el codigo se vuelve muchísimo más entendible por una persona humana.

**Objetivos de la refactorización:**

* Facilitar la comprensión del código.
* Cambiar estructura y diseño del código.
* Eliminar código muerto.
* Facilitar el mantenimiento en un futuro.

**6. Máquinas Virtuales: **

**6.1.Definición:**

Una máquina virtual es un software que simula un ordenador con todo su hardware y su software dentro de otro ordenador físico y puede ejecutar programas tal y como se haría en un ordenador físico, esto tiene muchos usos y muy variados algunos de ellos pueden ser probar varios sistemas operativos dentro de un ordenador para decidirte por cual instalar como nativo, este es el uso doméstico que se les da a las máquinas virtuales, para lo cual suelen usarse las plataformas VMWare Workstation o ESXi

Cada vez son más las empresas que usan sistemas de virtualización en sus servidores, como podían ser VMWare ESXi y Proxmox. Estos sistemas permiten ejecutar muchos servidores con múltiples sistemas operativos diferentes en una misma máquina física, y esto es muy útil, al poder tener un controlador de dominio de windows y un servidor http en un ubuntu server por ejemplo en un mismo equipo físico.

****

**6.2.Máquinas virtuales de proceso**

Una máquina virtual de proceso, se ejecuta como si fuese un programa normal por encima de un sistema operativo, por ejemplo la máquina virtual de java, esto nos permite ejecutar aplicaciones que están hechas precisamente para ser utilizadas de esa manera.

Por ejemplo todas las versiones de android trae una máquina virtual de java, ya que las aplicaciones android están programadas en java, android suele traer preinstalada una de estas dos máquinas, dalvik o ART. Estas máquinas virtuales hacen posible que android sea capaz de ejecutar aplicaciones.

**6.3. Máquinas virtuales de sistema**

Una máquina virtual de sistema es una máquina virtual en la cual podemos instalar un sistema operativo completo por encima, el el cual instalar aplicaciones y ejecutarlas como si de un ordenador real se tratase.

Estas máquinas virtuales suelen ser usadas para probar sistemas operativos antes de que los instalemos de forma nativa en nuestro equipo, también suelen usarse para instalar virus y así poder estudiar su comportamiento en un entorno aislado y seguro.

En este tipo de máquinas virtuales nos encontramos Virtualbox y vmware workstation.

**6.4.Ventajas y desventajas de las máquinas virtuales:**

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| Posibilidad de instalar varios sistemas operativos dentro de una misma máquina física. | Cuando hay muchas máquinas virtuales ejecutándose al mismo tiempo en una misma máquina física estas pueden presentar un rendimiento inestable. |
| La máquina virtual puede ofrecer una arquitectura diferente a la máquina real | La máquina virtual no es tan eficiente como una verdadera a la hora de acceder al hardware |
| Mantenimiento sencillo. | Precio |
| Centralizar la administración de red | Complejidad |
| Reducir la dependencia del hardware |  |
| Ideal para experimentar |  |

**Bibliografía**

1.

<https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software>

2.1:

<https://en.wikipedia.org/wiki/Computer>

<https://www.brighthubengineering.com/diy-electronics-devices/97571-difference-between-analog-and-digital-computing/>

2.2:

<https://es.wikipedia.org/wiki/Hardware>

2.3:

<https://es.wikipedia.org/wiki/Hardware>

2.4:

<https://en.wikipedia.org/wiki/Software>

2.4.1:

<https://en.wikipedia.org/wiki/Software>

<https://www.masadelante.com/faqs/sistema-operativo>

2.5:

<https://en.wikipedia.org/wiki/Operating_system#Components>

2.5.1:

<https://en.wikipedia.org/wiki/Operating_system#Components>

<http://www.cavsi.com/preguntasrespuestas/funciones-sistema-operativo/>

3:

<https://en.wikipedia.org/wiki/Software_development#Software_development_activities>

3.1:

<https://en.wikipedia.org/wiki/Software_development#Software_development_activities>

3.2:

<https://en.wikipedia.org/wiki/Software_development#Software_development_activities>

<https://www.linkedin.com/pulse/what-software-development-life-cycle-sdlc-phases-private-limited>

3.3:

<https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software>

<https://aspgems.com/metodologia-de-desarrollo-de-software-iii-modelo-en-espiral/>

<https://aspgems.com/metodologia-de-desarrollo-de-software-ii-modelo-de-diseno-iterativo/>

4.

<https://blog.dinahosting.com/los-10-lenguajes-de-programacion-mas-usados/>

<https://es.wikipedia.org/wiki/Programa_inform%C3%A1tico>

4.1:

<https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-programa/>

<https://concepto.de/programa-informatico/>

4.2:

<https://concepto.de/programa-informatico/>

<https://concepto.de/lenguaje-de-programacion/>

4.3:

<http://teleformacion.edu.aytolacoruna.es/PASCAL/document/senten.htm>

<http://dis.um.es/~lopezquesada/documentos/IES_1415/IAW/curso/UT3/ActividadesAlumnos/java7/paginas/pag2.html>

<https://es.calameo.com/books/00348279277f53453d702>

4.4:

<https://es.m.wikipedia.org/wiki/Abstracci%C3%B3n_(inform%C3%A1tica)>

4.5:

<http://lprogra.galeon.com/productos1671617.html>

4.6:

<https://es.wikiversity.org/wiki/T%C3%A9cnicas_de_programaci%C3%B3n/Programaci%C3%B3n_estructurada>

<https://es.slideshare.net/marfonline/clasificacin-de-los-datos-tcnicas-de-la-programacin-operadores>

<https://desarrolloweb.com/articulos/2477.php>

4.7:

<https://www.clasesdeinformaticaweb.com/entornos-de-desarrollo/obtencion-de-codigo-ejecutable/>

4.7.2:

<https://docs.oracle.com/cd/E37929_01/html/E36713/glcfm.html>

5.1.4:

<https://www.glosarioit.com/Depurador>

<https://es.wikipedia.org/wiki/Depurador>

5.1.5:

<https://es.wikipedia.org/wiki/Control_de_versiones> <https://es.wikipedia.org/wiki/Programas_para_control_de_versiones>

<https://es.wikipedia.org/wiki/Git>

5.1.6:

<https://www.zorraquino.com/diccionario/marketing-digital/que-es-interfaz-grafica-de-usuario.html>

<https://es.wikipedia.org/wiki/Interfaz_gr%C3%A1fica_de_usuario>

5.1.7:

<http://www.labroma.org/blog/2015/08/18/que-es-la-refactorizacion-de-codigo-algunos-ejemplos/>

<https://es.wikipedia.org/wiki/Refactorizaci%C3%B3n>

6.

<https://es.wikipedia.org/wiki/M%C3%A1quina_virtual>

6.2.

<http://erick.over-blog.com/2016/04/tipos-de-maquinas-virtuales.html>

6.3.

<https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas>

6.4.

<http://blog.hostdime.la/maquinas-virtuales-ventajas-y-desventajas/>