RESUMEN AMPLIACIÓN UNIDAD 01 ENTORNOS

3.2 Herramientas de apoyo al desarrollo del software.

La finalidad principal es **automatizar** las tareas y ganar **fiabilidad y tiempo**.

Las herramientas **CASE (*Computer Aided Software Engineering*)** son un conjunto de aplicaciones que se utilizan en el desarrollo de software con el objetivo de **reducir costes y tiempo del proceso**, mejorando por tanto la productividad del proceso.

El desarrollo rápido de aplicaciones o **RAD** es un proceso de desarrollo de software que comprende el **desarrollo iterativo, la construcción de prototipos y el uso de utilidades** CASE.

La tecnología CASE trata de **automatizar las fases del desarrollo de software** para que mejore la calidad del proceso y del resultado final.

En concreto, estas herramientas permiten:

- **Mejorar** la **planificación** del proyecto.
- Darle **agilidad** al proceso.
- Poder **reutilizar** partes del software en proyectos **futuros**.
- Hacer que las aplicaciones **respondan a estándares**.
- **Mejorar** la tarea del **mantenimiento** de los programas.
- **Mejorar** el proceso de **desarrollo**, al permitir **visualizar las fases de forma gráfica.**

CLASIFICACIÓN

Normalmente, las herramientas CASE se clasifican **en función de las fases del ciclo** de vida del software en la que ofrecen ayuda:

- **U-CASE:** ofrece ayuda en las fases de **planificación y análisis** de requisitos.
- **M-CASE:** ofrece ayuda en **análisis y diseño**.

- **L-CASE:** ayuda en la **programación del software**, **detección de errores** del código, **depuración** de programas y **pruebas** y en la generación de la documentación del proyecto.

Ejemplos de herramientas CASE libres son: **ArgoUML**, **Use Case Maker**, **ObjectBuilder**...

4. Lenguajes de programación

Definimos un lenguaje de programación como un idioma creado de forma artificial, formado por un conjunto de símbolos y normas que se aplican sobre un alfabeto para obtener un código, que el hardware puede entender y ejecutar.

La evolución de los lenguajes de programación:

LENGUAJE MÁQUINA > LENGUAJE ENSAMBLADOR > LENGUAJE DE ALTO NIVEL > LENGUAJE VISUAL

Características de los lenguajes de programación:

Lenguaje máquina:

- Sus instrucciones son combinaciones de unos y ceros.
- Es el único lenguaje que entiende directamente el ordenador. (No necesita traducción). Fue el primer lenguaje utilizado.
- Es único para cada procesador (no es portable de un equipo a otro).
- Hoy día nadie programa en este lenguaje.

Lenguaje ensamblador:

- Sustituyó al lenguaje máquina para facilitar la labor de programación.
- En lugar de unos y ceros se programa usando mnemotécnicos (instrucciones complejas).
- Necesita traducción al lenguaje máquina para poder ejecutarse.

- Sus instrucciones son sentencias que hacen referencia a la ubicación física de los archivos en el equipo.
- Es difícil de utilizar.

Lenguaje de alto nivel basados en código:

- Sustituyeron al lenguaje ensamblador para facilitar más la labor de programación.
- En lugar de mnemotécnicos, se utilizan sentencias y órdenes derivadas del idioma inglés. (Necesita traducción al lenguaje máquina).
- Son más cercanos al razonamiento humano.
- Son utilizados hoy día, aunque la tendencia es que cada vez menos.

Lenguajes visuales:

- Están sustituyendo a los lenguajes de alto nivel basados en código.
- En lugar de sentencias escritas, se programa gráficamente usando el ratón y diseñando directamente la apariencia del software.
- Su correspondiente código se genera automáticamente.
- Necesitan traducción al lenguaje máquina.
- Son completamente portables de un equipo a otro

4.1.- Concepto y características.

CONCEPTO

Un lenguaje de programación es el conjunto de:

- **Alfabeto**: conjunto de símbolos permitidos.
- **Sintaxis**: normas de construcción permitidas de los símbolos del lenguaje.
- **Semántica:** significado de las construcciones para hacer acciones válidas.

CARACTERÍSTICAS

Podemos clasificar los distintos tipos de lenguajes en base a distintas características:

- Según lo cerca que esté del lenguaje humano

    - **Lenguajes de Programación De alto nivel:** por su esencia, están más próximos al razonamiento humano.
    - **Lenguajes de Programación De bajo nivel:** están más próximos al funcionamiento interno de la computadora:
        - **Lenguaje Ensamblador**
        - **Lenguaje Máquina**
- Según la técnica de programación utilizada:
    - **Lenguajes de Programación Estructurados:** Usan la técnica de programación estructurada. Ejemplos: Pascal, C, etc.
    - **Lenguajes de Programación Orientados a Objetos:** Usan la técnica de programación orientada a objetos. Ejemplos: C++, Java, Ada, Delphi, etc.
    - **Lenguajes de Programación Visuales:** Basados en las técnicas anteriores, permiten programar gráficamente, siendo el código correspondiente generado de forma automática.

4.2.- Lenguajes de programación estructurados.

La programación estructurada se define como una técnica para escribir lenguajes de programación que permite sólo el uso de tres tipos de sentencias o estructuras de control:

- Sentencias secuenciales.
- Sentencias selectivas (condicionales).
- Sentencias repetitivas (iteraciones o bucles).

La programación estructurada fue de gran éxito por su sencillez a la hora de construir y leer programas. Fue sustituida por la programación modular, que permitía dividir los programas grandes en trozos más pequeños.

A su vez, triunfaron los lenguajes orientados a objetos y de ahí la programación visual.

VENTAJAS DE LA PROGRAMACIÓN ESTRUCTURADA

- Los programas son fáciles de leer, sencillos y rápidos.

- El mantenimiento de los programas es sencillo.
- La estructura del programa es sencilla y clara.

INCONVENIENTES

- Todo el programa se concentra en un único bloque
- No permite reutilización eficaz de código, ya que todo va "en uno".

4.3.- Lenguajes de programación orientados a objetos.

Los lenguajes de programación orientados a objetos tratan a los programas no como un conjunto ordenado de instrucciones sino como un conjunto de objetos que colaboran entre ellos para realizar acciones.

No es una programación tan intuitiva como la estructurada.

El 55% del software producido por las empresa se hace usando esta técnica

Razones:

- El código es reutilizable
- Es más fácil de localizar y depurar un objeto que un programa entero en caso de error.
- Los objetos tienen una serie de atributos que los diferencian entre sí
- Se define clase como una colección de objetos con características similares
- Mediante los llamados métodos, los objetos se comunican con otros produciéndose un cambio de estado de los mismos
- Son como unidades individuales e indivisibles que forman la base.

Principales lenguajes orientados a objetos: Ada, C++, VB.NET, Delphi, Java, PowerBuilder, etc.

5.- Fases en el desarrollo y ejecución del software.

Etapas:

1. ANÁLISIS DE REQUISITOS.

Se especifican los requisitos funcionales y no funcionales del sistema.

2. DISEÑO.

Se divide el sistema en partes y se determina la función de cada una.

3. CODIFICACIÓN.

Se elige un Lenguajes de Programación y se codifican los programas.

4. PRUEBAS.

Se prueban los programas para detectar errores y se depuran.

5. DOCUMENTACIÓN.

De todas las etapas, se documenta y guarda toda la información.

6. EXPLOTACIÓN.

Instalamos, configuramos y probamos la aplicación en los equipos del cliente.

7. MANTENIMIENTO.

Se mantiene el contacto con el cliente para actualizar y modificar la aplicación el futuro.

5.1.- Análisis.

Primera fase del proyecto, cuando se finalizada pasamos a la siguiente (diseño)

Es la fase de mayor importancia en el desarrollo del proyecto y todo lo demás dependerá de lo bien detallada que esté.

También es la más complicada, ya que no está automatizada y depende en gran medida del analista que la realice.

Se especifican y analizan los requisitos funcionales y no funcionales del sistema.

Requisitos:

- Funcionales: Qué funciones tendrá que realizar la aplicación. Qué respuesta dará la aplicación ante todas las entradas. Cómo se comportará la aplicación en situaciones inesperadas.

- No funcionales: Tiempos de respuesta del programa, legislación aplicable, tratamiento ante la simultaneidad de peticiones, etc.

Es fundamental la comunicación entre cliente y analista.

La culminación de esta fase es el documento ERS (Especificación de Requisitos Software).

En este documento quedan especificados:

La planificación de las reuniones que van a tener lugar.

- Relación de los objetivos del usuario cliente y del sistema.
- Relación de los requisitos funcionales y no funcionales del sistema.
- Relación de objetivos prioritarios y temporización.
- Reconocimiento de requisitos mal planteados o que conllevan contradicciones, etc.

5.2.- Diseño.

Se divide el sistema en partes y se establece qué relaciones habrá entre ellas, se decide qué hará exactamente cada parte

En definitiva, debemos crear un modelo funcional-estructural de los requerimientos del sistema global.

Se deben tomar decisiones importantes:

- Entidades y relaciones de las bases de datos.
- Selección del lenguaje de programación que se va a utilizar.
- Selección del Sistema Gestor de Base de Datos.
- Etc.

5.3.- Codificación. Tipos de código.

Consiste en elegir un determinado lenguaje de programación, codificar toda la información anterior y llevarlo a código fuente.

Las características deseables de todo código son:

- **Modularidad:** que esté dividido en trozos más pequeños.
- **Corrección**: que haga lo que se le pide realmente.
- **Fácil de leer:** para facilitar su desarrollo y mantenimiento futuro.
- **Eficiencia**: que haga un buen uso de los recursos.
- **Portabilidad:** que se pueda implementar en cualquier equipo.

El código pasa por diferentes estados:

- **Código fuente:** escrito por los programadores en algún editor de texto, se usa lenguaje de alto nivel.
- **Código objeto:** código binario resultante de compilar el código fuente. La compilación es la traducción de una sola vez del programa, la interpretación es la traducción y ejecución simultánea del programa línea a línea. Código intermedio no inteligible por humano o computadora.
- **Código ejecutable:** Código binario resultante de enlazar los archivos del código objeto con ciertas rutinas y bibliotecas necesarias. El SO será el encargado de cargarlo en la RAM y ejecutarlo. También conocido como código máquina y directamente inteligible por la computadora.

5.4.- Fases en la obtención de código.

5.4.1.- Fuente.

Conjunto de instrucciones que la computadora deberá realizar, escritas por los programadores en algún lenguaje de alto nivel.

No directamente ejecutable por la máquina

Un aspecto importante en esta fase es la elaboración previa de un algoritmo, se diseña en pseudocódigo.

Para obtener el código fuente de una aplicación informática:

- Se debe partir de las etapas anteriores de análisis y diseño.
  Se diseñará un algoritmo que simbolice los pasos a seguir para la resolución del problema.

- Se elegirá una Lenguajes de Programación de alto nivel apropiado para las características del software que se quiere codificar.
- Se procederá a la codificación del algoritmo antes diseñado.

Un aspecto importante a tener en cuenta es su licencia:

- **Código fuente abierto**. Es aquel que está disponible para que cualquier usuario pueda estudiarlo, modificarlo o reutilizarlo.
- **Código fuente cerrado**. Es aquel que no tenemos permiso para editarlo.

5.4.2.- Objeto.

Código intermedio

Resultado de traducir código fuente a código equivalente formado por unos y ceros que aún no puede ser ejecutado directamente por la computadora.

Código resultante de la compilación del código fuente.

Consiste en un bytecode (código binario resultante de la traducción de código de alto nivel que aún no puede ser ejecutado) que está distribuidso en varios archivos.

Proceso de traducción:

- Compilación: El proceso de traducción se realiza sobre todo el código fuente, en un solo paso. Se crea código objeto que habrá que enlazar.
- Interpretación: El proceso de traducción del código fuente se realiza línea a línea y se ejecuta simultáneamente. No existe código objeto intermedio.

5.4.3.- Ejecutable.

Resultado de enlazar los archivos de código objeto, consta de un único archivo ejecutable por la computadora.

Se enlazan todos los archivos del código objeto a través de un software llamado linker (Enlazador, pequeño software encargado de unir archivos para generar un programa ejecutable) y obtener así un único archivo que ya sí es ejecutable por el usuario.

- A partir de un editor, escribimos el lenguaje fuente con algún Lenguaje de programación
- A continuación, el código fuente se compila obteniendo código objeto o bytecode
- Ese bytecode, a través de la máquina virtual, pasa a código máquina, ya directamente ejecutable por la computadora.

5.5.- Máquinas virtuales.

Una máquina virtual es un tipo especial de software cuya misión es separar el funcionamiento del ordenador de los componentes hardware instalados.

Con el uso de máquinas virtuales podremos desarrollar y ejecutar una aplicación sobre cualquier equipo, esto garantiza la portabilidad de las aplicaciones.

Funciones principales:

- Conseguir que las aplicaciones sean portables
- Reservar memoria para los objetos que se crean y liberar memoria no utilizada
- Comunicarse con el sistema donde se instala la aplicación (huésped) para control de dispositivos hardware implicados en los procesos
- Cumplimiento de lal normas de seguridad de las aplicaciones

CARACTERÍSTICAS DE LA MÁQUINA VIRTUAL

Para ejecutar el bytecode en cualquier máquina se requiere tener independecia respecto al hardware que se vaya a utilizar, para ello la mv aísla la aplicación de los detalles físicos del equipo.

Funciona como una capa de software de bajo nivel y actúa como puente entre el bytecode de la aplicación y los dispositivos físicos del sistema.

La MV verificael bytecode antes de ejecutarlo y protege direcciones de memoria

5.5.1.- Frameworks.

Un framework es una estructura de ayuda al programador donde desarrollar proyectos sin partir de cero.

Están definidos programas soporte, bibliotecas, lenguaje interpretadom etc.

Ventajas:

- Desarrollo rápido
- Reutilización de partes de código
- Diseño uniforme del software
- Portabilidad de aplicaciones

Inconvenientes:

- Gran dependencia del código respecto al framework utilizado
- La instalación e implementación del framework consume bastantes recursos del sistema

Ejemplos: .NET (windows) y Spring de Java

5.5.2.- Entornos de ejecución.

Un entorno de ejecución es un servicio de MV que sirve como base software para la ejecución de programas.

Es un conjunto de utilidades que permiten la ejecución de programas.

Se encarga de:

- Configurar la memoria principal disponible
- Enlazar los archivos del programa con bibliotecas existentes y subprogramas creados
- Depurar los programas: comprobar la existencia de error semánticos del lenguaje

Funcionamiento del entorno de ejecución:

Está formado por la MV y los API ´s, se suelen distribuir conjuntamente porque necesitan ser compatibles entre sí.

El entorno funciona como intermediario entre el lenguaje fuente y el SO, y consigue ejecutar aplicaciones.

Sin embargo no es suficiente para desarrollar nuevas aplicaciones.

5.5.3.- Java runtimeenvironment.

JRE Java RuntimeEnvironment (entorno en tiempo de ejecución java)

Se compone de conjunto de utilidades que permitirá la ejecución de programas java sobre cualquier tipo de plataforma.

Está formado por:

- Una JMV o JVM que es el programa que interpreta el código de la aplicación escrito en java.
- Bibliotecas de clase estándar que implementan el API de java.
- JMV y API son consistentes entre sí, son distribuidas conjuntamente.

5.6.- Pruebas.

Estas se realizan sobre un conjunto de datos de prueba, consisten en un conjunto seleccionado y predefinido de datos límite a los que la aplicación es sometida.

Se distinguen:

PRUEBAS UNITARIAS

Probar una a una las diferentes partes del software y comprobar su funcionamento

PRUEBAS DE INTEGRACIÓN

Se realizan una vez las unitarias tienen éxito y comprueba el funcionamiento del sistema completo con todas sus partes interrelacionadas

La prueba final se denomina Beta Test y se realiza sobre el entorno de producción donde el software va a ser utilizado por el cliente

El período es pactado con el cliente normalmente

5.7.- Documentación.

Se documenta para dar toda la información a los usuarios de nuestro software y poder cometer futuras revisiones.

Se documenta en cada fase del mismo para pasar de una a otra de forma clara y definida.

Se distingue entre

Guía técnica: Para mantenimiento futuro

Guía de uso: Para los usuarios finales

Guía de instalación: Para la implementación de la aplicación

5.8.- Explotación.

La explotación es la fase en que los usuarios finales conocen la aplicación y comienzan a utilizarla.

En el proceso de instalación, los programas son transferidos al computador del usuario cliente y posteriormente configurados y verificados.

5.9.- Mantenimiento.

Es la más larga de todo el ciclo de vida del software

Se define como el proceso de control, mejora y optimización del software.

Tipos de cambios:

- **Perfectos**: Mejorar la funcionalidad
- **Evolutivos**: Nuevas necesidades, modificaciones, expansiones o eliminaciones de código
- **Adaptivos**: Actualizaciones para adaptarse a nuevas tendencias
- **Correctivos**: Errores futuros