Desarrollo de Software

Contenido

Software de ordenador

El ordenador se divide en la parte física (hardware) y la lógica (software), que hace referencia a los programas que ejecuta para realizar determinadas tareas sobre el hardware. Se puede ordenar por dos categorías: Por sus tareas y su distribución:

- **Por sus tareas:**
  - **De sistema:** Hace que el hardware funcione, controlan el hardware y lo conecta con los usuarios (Sistemas operativos, drivers, diagnostico, etc)
  - **De aplicación:** Realizan tareas específicas para las que han sido creados, como por ejemplo las herramientas ofimáticas
  - **De programación o desarrollo:** Proporcionan las herramientas para escribir programas informáticos por medio de los lenguajes de programación
- **Por su distribución:**
  - **Shareware:** El usuario evalúa de forma gratuita el producto, pero con características limitadas por un pago (WinRAR (si y no))
  - **Freeware:** El usuario descarga de forma gratuita el programa, pero tiene derechos de autor
  - **Adware:** Software con publicidad incrustada, también se puede llegar a considerar malware

Licencias

Las licencias son un contrato entre un desarrollador y el usuario, se especifican los derechos de ambas partes. El desarrollador elige el tipo de software que distribuye:

- **Software libre:** El autor concede al usuario las libertades de usar el programa con todo tipo de fin, para estudiar su código, para modificarlo, compartir copias, etc. No significa que sea gratis, sino que otorga al usuario ciertas libertades.
- **Software propietario:** No permite al usuario acceder al código fuente y, por ende, limita al usuario a su uso. Este no puede ser redistribuido sin permiso, al igual que no se puede modificar o copiar en varios equipos (MacOS (cabrones de la manzanita)
- **Software de dominio público:** No pertenece a nadie, no tiene licencia y cualquiera puede hacer lo que quiera con él.

Concepto de programa informático

Son fragmentos de software con una serie de instrucciones y procesos con el fin de hacer un objetico concreto. Están programados en un lenguaje de programación concreto y se traducen al lenguaje máquina para ser procesados.

Componentes

Es necesario tener en cuenta el hardware del ordenador, ya que las instrucciones hardware se cargan en la memoria principal y se ejecutan en la CPU.

La arquitectura empleada está basada en la de Von Neumann, donde la unidad de control (UC) conecta con la unidad aritmeticológica (ALU), los dispositivos de entrada y salida y con las memorias de instrucción y datos.

- **Unidad de Control:** Interpreta y ejecuta las instrucciones que se almacenan en la memoria principal, además genera señales de control para ejecutarlas
- **Unidad aritmeticológica (ALU):** Recibe los datos y ejecuta las operaciones matemáticas y lógicas, siendo supervisada por la unidad de control.
- **Registros:** Almacenan la información temporal, es el almacenamiento interno.
    - **Contador de programa:** Tiene la dirección de la siguiente instrucción a realizar. Se actualiza por medio de la CPU
    - **Registro de instrucción:** Contiene el código de la instrucción. Consta de la dirección de memoria y el fragmento de código a ejecutar.
    - **Registro de dirección de memoria:** Tiene asignada una dirección correspondiente a una posición de memoria por medio del bus de direcciones
    - **Registro de intercambio de memoria:** Recibe o envía la información o dato contenido en el registro de dirección.
    - **Decodificador de instrucción:** Extrae y analiza el código de la instrucción del registro de instrucción.
    - **El reloj:** Por medio de impulsos eléctricos, marca los tiempos para ejecutar las instrucciones del decodificador
    - **El secuenciador:** Se sincronizan al reloj para que se ejecuten correctamente

Cuando se ejecuta una instrucción hay varias fases:

- **Búsqueda:** Busca la información en la memoria principal y la lleva a la unidad de control
- **Fase de ejecución:** Ejecuta las acciones de las instrucciones.

Código fuente, objeto y ejecutable

Las instrucciones de los programas que ejecutan los ordenadores se deben escribir en un lenguaje que estos entiendan, el binario. Programar en binario es muy complejo así que se crearon los lenguajes de programación de alto nivel, para traducir el programa escrito en un lenguaje a un programa escrito en binario. En este proceso se distinguen tres tipos de código:

- **Código fuente:** El que hacen los programadores con algún entorno de desarrollo. No se ejecutan directamente en el ordenador
- **Código objeto:** El código que se crea al compilar el código fuente. No se entiende ni para el ordenador ni la persona.
- **Código ejecutable:** Se obtiene tras unir el código con varias librerías y que así lo lea el ordenador.

Compilación

Es el proceso a través del cual un programa se transforma al lenguaje máquina (binario). En función del lenguaje de programación, se compila o se interpreta. La compilación se

realiza por medio de dos programas: Un **compilador** (que gestiona los errores y el análisis de este), y un **enlazador**, que inserta en el código objeto las librerías para crear un programa ejecutable.

Pasos que pasa el compilador:

- **Léxico:** Lee el código obteniendo tokens
- **Sintáctico:** Recibe los tokens, ejecuta el análisis de la estructura del programa y comprueba si está todo estructurado
- **Análisis semántico:** Revisa que las declaraciones, expresiones, etc, sean correctas.
- **Generación de código intermedio:** Crea una representación intermedia entre código fuente y objeto
- **Optimización de código:** Se mejora el código para que sea más fácil de interpretar
- **Generación de código:** Genera el código objeto

Tipos de lenguajes de programación

Son los conjuntos de caracteres, reglas y acciones combinadas y consecutivas que un equipo debe ejecutar. Los lenguajes constan de los siguientes elementos:

- Alfabeto: símbolos permitidos
- Sintaxis: Reglas para realizar las construcciones con los símbolos
- Semántica: Reglas que determinan el significado de la construcción.

Se clasifican según su nivel de abstracción, su ejecución y su paradigma.

- **Según su abstracción:**
  - **Alto nivel:** Es más fácil a la hora de aprender, ya que usan el lenguaje natural (normalmente inglés) para ejecutar lo que se escribe.
  - **Bajo nivel:** Es el lenguaje máquina, un lenguaje de programación que el equipo entiende perfectamente sin pasar por traducciones
  - **Nivel medio:** Se suele usar para crear sistemas operativos, tiene un poco de ambos.
- **Por su ejecución**
  - **Compilados:** Necesitan un compilador para traducir al lenguaje máquina. Traducen por bloques de código y devuelven errores si está mal escrito
  - **Interpretados:** Sigue las instrucciones del programa fuente con las entradas proporcionadas por su traductor. Es más lento que el compilado, pero evita tener que compilar todo el rato los programas que se prueban
- **Por su paradigma de programación**
  - **Imperativo:** Sentencias que marcan cómo debe manipularse la información digital. Se establece el orden de ejecución y el flujo del programa por medio de **estructuras de control**.
  - **Funciona:** Se forman por definiciones de funciones con sus argumentos. Las variables almacenan definiciones y expresiones
  - **Lógicos:** Basados en el concepto de razonamiento. Por medio de una base de datos el sistema razona las relaciones y las propiedades de las entidades

- **Estructurados:** Utilizan las funciones lógicas anteriores y resultan fáciles de leer. El código está centrado en un solo bloque, por lo que es difícil tratar con sus problemas.
- **Lenguajes orientados a objetos:** Son los lenguajes definidos por un conjunto de objetos y módulos. A su vez, estos están formados por una estructura de datos (atributos) y una colección de métodos para usar dichos datos. Las clases son la plantilla para crear objetos, y tienen de ventaja su fácil reutilización, teniendo de inconveniente la subjetividad del programador

Ciclos de vida del software

Un ciclo de vida determina el orden en el que se hacen las tareas de un proceso. Consisten en la obtención de productos e indicar las características para satisfacerlos.

Hay distintos modelos:

- **Modelo en cascada:** Para entrar en una etapa, hace falta finalizar la anterior. Se establece un protocolo de revisión al completar cada fase. El modelo permite reiterar, es decir, permite hacer cambios en la realización del programa, aunque supondrá hacer pruebas de vuelta
- **Modelo en cascada con retroalimentación:** Es como el modelo anterior, pero este tiene una retroalimentación, es decir, si se detectan fallos en una etapa, se vuelve a la anterior para hacer ajustes.
- **Modelo iterativo incremental:** Se basa en varios modelos retroalimentados repetidamente. Divide el software en partes pequeñas pero utilizables (incrementos). Los incrementos se realizan sobre un incremento ya terminado. Es más versátil y barato. Permite la entrega temprana de software se recomienda al probar nuevas tecnologías, pero es difícil estimar el esfuerzo que supone hacerlo.
- **Modelo en espiral:** Combina el modelo en cascada con el iterativo. Es una espiral donde, en cada ciclo (formado por cuatro fases) se desarrolla una parte de software. Cada ciclo se compone de estas fases:
  - **Determinar riesgos:** Identifica los objetivos y alternativas para llegar al objetivo
  - **Análisis de riesgos:** Evalúan las alternativas con las limitaciones, además de cómo resolverlos
  - **Desarrollo y prueba:** Se desarrolla la solución y se verifica si es aceptable
  - **Planificación:** Se revisa y evalúa para decidir si se continua.

  Algunas ventajas es que se analiza el riesgo en todas las etapas y no requiere una definición completa, pero tiene una difícil evaluación de riesgos y el costo del proyecto avanza en función de las iteraciones de la espiral.

Modelo en V:

Es un proceso que representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto. Describe las actividades y resultados que se dan en el proceso de crear un proyecto. Es un modelo muy sencillo donde el cliente está involucrado, aunque sus pruebas son costosas y el cliente debe tener paciencia.

Es muy similar al de cascada, la forma en v viene de las **etapas**, que van desde el diseño de pruebas hasta la aplicación de estas.

Metodologías ágiles

Son métodos de gestión que permiten adaptar la forma de trabajo a la naturaleza del proyecto, teniendo en cuenta las exigencias. Tiene una serie de ventajas:

- Mejora la satisfacción del cliente
- Ahorra en costes y tiempos
- Mejoran la calidad del producto
- Elimina lo innecesario
- Alerta rápidamente los errores y los problemas

También tiene problemas a la hora de aplicar las metodologías:

- Los clientes están sometidos a presiones ya que tienen que ocuparte de las gestiones
- No todos los componentes del equipo siempre tienen la misma personalidad
- Los cambios con muchos participantes son difíciles
- Mantener la línea de trabajo es difícil por temas como los plazos de entrega y los ritmos
- Las grandes compañías tienen muchos cambios todo el tiempo, por lo que trabajar con esto es un infierno.

SCRUM

Se basa en el trabajo iterativo. Tiene tres partes que aportan un desarrollo incremental

- **Planificación:** Se establecen los objetivos generales del proyecto
- **Ciclos:** Se desarrolla una iteración o un incremento
- **Documentación:** Se desarrolla la ayuda y los manuales de usuario.

Una de sus bases es el ciclo de vida iterativo e incremental, donde se va liberando el producto por partes, poco a poco, incrementando su funcionalidad.

Además, tienen una reunión diaria para ver qué se hizo el día anterior y qué problemas se han encontrado. Tienen también **reuniones de revisiones** al final de cada sprint para analizar si se ha completado o no.

Programación extrema (XP)

Potencian las relaciones interpersonales, promoviendo el trabajo en equipo. Es la retroalimentación continua entre el cliente y los desarrolladores. La XP es adecuada para proyectos de requisitos imprecisos

Los requerimientos se expresan como escenarios y se basan en la programación a pares (trabajo colectivo). Se valora las interacciones del equipo de desarrollo sobre el proceso. Se trata de desarrollar un software que funcione, y propone que haya una colaboración constante además de una alta habilidad de responder a los cambios que haya a lo largo del proyecto.

Kanban

Se utiliza para controlar el avance del trabajo en una línea de producción. Visualiza el trabajo en bloques, y tiene varias barras de progreso donde la tarea va avanzando según su estado.

Se consigue así una visión global de todas las tareas y permite hacer cálculos con las tareas realizadas y las similares que haya que realizar, haciendo así una idea del tiempo que puede tardar y limitar el WIP (work in progress)

RUP (Rational Unified Process)

Estructura y organiza el desarrollo de software por los casos de uso

Se centra en la arquitectura del software. Se centra enn vistas a las fases de análisis, diseño e implementación, analizando el software como uno solo y por partes.

Se divide en proyectos donde se va incrementando su funcionalidad

La estructura es dinámica (fases sobre el tiempo de desarrollo), estática (muestran las actividades), y prácticas (muestran las buenas prácticas)

Proceso de obtención de código

El código fuente pasa por una etapa de codificación para ser generado. Se pasa por varios estados:

- **Código fuente:** El escrito por los programadores por medio de un lenguaje de alto nivel
- **Código objeto:** El resultante de compilar el código fuente, no se entiende
- **Código máquina o código ejecutable:** Lenguaje que entiende la máquina a la perfección (binario)

Herramientas de obtención de código ejecutable

- **Librerías:** Contienen funcionalidades distintas, y pueden ser llamadas desde cualquier aplicación del sistema
- **Editor:** Crea el código fuente, marcando su sintaxis y realizando tareas de autocompletado
- **Compilador:** Proporciona el lenguaje máquina entendible por el ordenador
- **Enlazador:** Coge los objetos que se crean en los primeros pasos de la compilación, creando un ejecutable.