**TEMA 4 - EL SOFTWARE**

¿Qué es el software?

Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

• En lugar de manipular interruptores o cables, los programadores escriben programas y los introducen en la memoria del computador, donde se almacenan como secuencias de 0 y 1.

• El concepto de programa almacenado apareció en 1945, y se le atribuye a John Von Neumann.

**Tipos de software: SEGÚN SU FIN**

**Software de aplicación**

• Programas y utilidades que cumplen una tarea específica.

• Producto final.

• Permiten que una máquina de uso general se convierta en un sistema de uso específico

Ej. Word, Google chrome, Photoshop

**Software de programación**

• Utilidades digitales que permiten a los programadores desarrollar programas informáticos utilizando diferentes lenguajes de programación o bases de datos.

• Ej. Editores de texto, compiladores, intérpretes, enlazadores, depuradores, …

**Software del sistema**

• Permite al usuario usar la interfaz de SO.

• Conecta las aplicaciones con los recursos de HW

• Ej. Controladores, herramientas de diagnóstico, SO, …

**Tipos de software: ENTORNO**

De escritorio, app, web, “en la nube”

**Tipos de Software: Licencia/derechos de autor**

Cuando se adquiere un programa se adquiere en realidad el derecho a usarlo, es decir, una licencia.

El acuerdo que se acepta al utilizar un programa se denomina EULA (End User License Agreement)

Según el Proyecto GNU

o Freeware. Software gratuito, protegido por derechos de autor.

o Shareware. Uso con limitaciones.

o Software libre. Permite copiarse, modificarse y distribuirse.

o Software de código abierto. Propiedad intelectual compartida.

o Software privativo. No es libre, pertenece a una empresa.

o Software comercial. Su finalidad es generar ganancias económicas.

**Creación de Software**

Metodología

• Describe cómo es el proceso de elaboración del Software, desde la idea o necesidad, hasta la entrega del producto final.

• INGENIERÍA DEL SOFTWARE.

Tecnología

• Conjunto de herramientas (técnicas) utilizadas en cada etapa para producir los resultados esperados.

• DESARROLLO Y PROGRAMACIÓN.

Creación de Software. Ciclo de vida clásico.

Antes de empezar viabilidad técnica y económica

**Análisis de requisitos ¿Qué se quiere hacer?:**

• En esta fase se analizan las necesidades del cliente y de los usuarios finales del software para determinar qué objetivos debe cubrir.

• El resultado de esta fase son dos documentos que contienen la especificación completa de lo QUÉ se debe hacer:

– URD (User Requirement Document)

– SRD (Software Requirement Document)

• Es importante señalar que los requisitos deben estar bien definidos y recoger todas las necesidades del sistema, dado que son el punto de partida para el resto del proceso.

**Diseño del Sistema. ¿Cómo se hará?**

• Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo.

• El resultado de esta fase es el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

• Es conveniente distinguir entre

– Diseño de alto nivel o arquitectónico: define la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. (ADD)

– Diseño detallado: define los algoritmos empleados y la organización del código para comenzar la implementación. (DDD)

Diseño de:

• Interfaz de usuario

• Base de datos

• Procesos

**Programación.**

• Es la fase en donde se implementa el código fuente, haciendo uso de prototipos así como de pruebas y ensayos para corregir errores.

Dependiendo del lenguaje de programación y su versión se crean las bibliotecas y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

**Pruebas**

• La fase de pruebas es crítica en un sistema, dada la dificultad de construir sistemas software sin fallos. Las pruebas pueden ser llevadas a cabo por

– Los propios equipos de desarrollo

– Equipos específicos de prueba

– Usuarios finales

Herramientas - Compiladores, Editores, SDK, …

Lenguajes y paradigmas - C, C++, Java, C#, PHP, HTML, …

**Documentación y mantenimiento**

• El proceso de documentación es una tarea transversal a todos los pasos, aunque se culmina en este punto.

• Una de las etapas más críticas, (se destina un elevado porcentaje de los recursos), es el mantenimiento del Software, ya que al utilizarlo como usuario final puede ser que no cumpla con todas nuestras expectativas.

• Implica tareas, durante toda la vida útil del producto, de

– Monitorización.

– Evaluación.

– Reparación.

– Mejora del sistema.

**Metodología**

Existen dos grupos principales:

• Metodologías tradicionales.

– Los ciclos de desarrollo son poco flexibles.

– No se permiten realizar cambios.

– Organización del trabajo lineal.

• Metodologías ágiles.

– Son flexibles y ágiles.

– Altamente utilizadas en la actualidad.

– Metodologías incrementales.

– Creación de equipos de trabajo autónomos.

Destacan:

• **Waterfall** (Cascada). Construcción secuencial con revisión del producto en cada etapa.

• **Prototipado**. Basada en la construcción de un prototipo funcional para la recogida de feedback.

**• Espiral**. Combinación de las dos anteriores. Añada análisis de riesgo.

• **Incremental**. Construcción progresiva del proyecto.

**• Diseño rápido de aplicaciones (RAD)**. Permite desarrollos de alta calidad en un periodo corto de tiempo, con un alto coste.

• **Kanban**. Se dividen las tareas en porciones mínimas y se organizan en un tablero como tareas pendientes, en curso y finalizadas.

• **Scrum.** Metodología incremental. Se itera sobre bloques de tiempo cortos y fijos con resultados completos en cada bloque.

**• Lean**. Pequeños equipos de trabajo muy capacitados y versátiles realizan las tareas en poco tiempo. El tiempo y el coste permanecen en un segundo plano.

• **Programación Extrema (XP)**. Basada en relaciones interpersonales. Creación de buen ambiente de trabajo y feedback constante del cliente.

Niveles de programación.

**Lenguaje máquina:**

• Lenguaje más primitivo de programación, basado en la interpretación binaria (0/1).

**Lenguajes de bajo nivel:**

•Fáciles de interpretar, pueden variar según la máquina en la que se esté programando. Ejemplo: Ensamblador.

**Lenguajes de alto nivel**.

• Lenguajes más utilizados, dividido en generaciones cronológicas. Ejemplos: C++,Python, C#, SQL,PHP,…

Compiladores vs. intérpretes

• Su función es la convertir el código de software que se ha escrito a un formato ejecutable y legible por máquina.

**Compiladores**

• Traduce código fuente a lenguaje intermedio, produce un archivo objeto

**Interpretes**

• Traduce y ejecuta un programa

Interfaz de usuario gráfica

Descripción generada automáticamente