CURSO 2022-2023
CICLO SUPERIOR DE DESARROLLO DE APLICACIONES WEB
IES AGUADULCE

Entornos de Desarrollo

Francisco Javier Sueza Rodríguez

24 de octubre de 2022

Índice general

1 Desarrollo de Software **4**
1.1 Software y Tipos de Software 4
1.2 Relación Hardware-Software 5
1.3 Desarrollo de Software 6
1.3.1 Ciclos de Vida del Software 6
1.3.2 Herramientas de Apoyo al Desarrollo De Software 7
1.4 Lenguajes de Programación 8

Glosario **9**

Bibliografía **10**

Índice de figuras

1.2.1 Arquitectura John Von Neuman 5
1.3.1 Etapas del Desarrollo de Software 6
1.4.1 Evolución de los lenguajes de programación 8

Tema 1

Desarrollo de Software

En esta unidad vamos a realizar una introducción al concepto de software, así como a los diferentes tipos de software que podemos encontrar y al proceso de desarrollo de éste. También hablaremos de los lenguajes de programación, en que consiste y que características tiene. Por último, veremos que son los entornos de desarrollo, cuales su función y su evolución histórica.

1.1 Software y Tipos de Software

Un ordenador se compone de dos partes bien diferenciadas, el **hardware** y el **software**.

El **hardware**, **equipo** o **soporte físico** en informática se refiere a las partes físicas, tangibles de un sistema informático, sus componentes eléctricos, electrónicos y electromecánicos. Los cables, así como los muebles o cajas de todo tipo también se incluyen dentro de esta categoría. [1]

Por otro lado, el **software** es el conjunto de componentes lógicos que hace posible la realización de tareas específicas [2]. Dicho de otra forma, es el conjunto de programas informáticos que actúa sobre el hardware para ejecutar lo que el usuario desee.

Según su funcionalidad, podemos diferencias tres tipos principales de software:

- **Sistema Operativo**: conjunto de programas de un sistema informático que gestiona los recursos hardware y provee servicios a las aplicaciones informáticas para su funcionamiento. Ejemplos de sistemas operativos son: Microsoft Windows, Linux, macOS...
- **Software de Programación**: conjunto de herramientas y utilidades que permiten a los programadores desarrollar programas informáticos.
- **Aplicaciones Informáticas**: programas o conjunto de programas que tienen una aplicación concreta. Algunos ejemplos de aplicaciones informáticas son los procesadores de texto, reproductores multimedia, juegos...

Aunque estos son los tipos principales de software, podemos ampliar esta clasificación incluyendo los siguientes tipos, los cuales son, en mayor medida, subdivisiones de las aplicaciones informáticas [3]:

- **Software de Tiempo Real**: son aquellos programas que se usan en sistemas de tiempo real y que reaccionan con el entorno físico respondiendo a los estímulos del éste en un tiempo limitado. Ejemplos de este software son los sistemas de control de procesos, aplicaciones de robótica,..

- **Software de Gestión**: son programas que utilizan grandes cantidades de información almacenadas en bases de datos para ayudar a la administración y toma de decisiones. Un ejemplo de estos sistemas son los **ERP**.
- **Software Científico o de Ingeniería**: software que se encarga de realizar complejos cálculos numéricos de todo tipo, siendo la corrección y exactitud en estos cálculos uno de los requisitos básicos. También incluye los sistemas de diseño, ingeniería y fabricación asistida por ordenador (CAD, CAE y CAM), simuladores gráficos,..
- **Software Empotrado**: software que por norma general va instalado en memorias ROM y sirve para controlar productos y sistemas de los mercados industriales. Se aplica a todo tipo de productos como neveras, reproductores de vídeo, misiles, sistemas de control de automóviles,...
- **Software de Inteligencia Artificial**: software que basándose en el uso de lenguajes declarativos, sistemas expertos y redes neuronales, para simular comportamientos humanos como el aprendizaje, razonamiento y la resolución de problemas para la realización de forma fiable y rápida operaciones que para el ser humano son tediosas o incluso inabordables. Ejemplos de estas aplicaciones son las aplicaciones de *machine learning*, chatbots,...

En este tema, nos centraremos en las **aplicaciones informáticas**, como se desarrollan y cuales son las fases de este desarrollo. También veremos el **ciclo de vida de una aplicación informática** y los diferentes tipos de **lenguajes de programación** y sus características.

1.2 Relación Hardware-Software

Como hemos comentado en el punto anterior, un sistemas informático se compone de hardware y software. Existe una relación indisoluble entre estos dos componentes ya que se necesita que estén instalados y configurados correctamente para que el ordenador funcione.

El primer modelo de arquitectura de hardware con programa almacenado fue propuesto por **John Von Neumman** en 1946. A continuación se muestra una imagen con las diferentes partes de esta arquitectura:

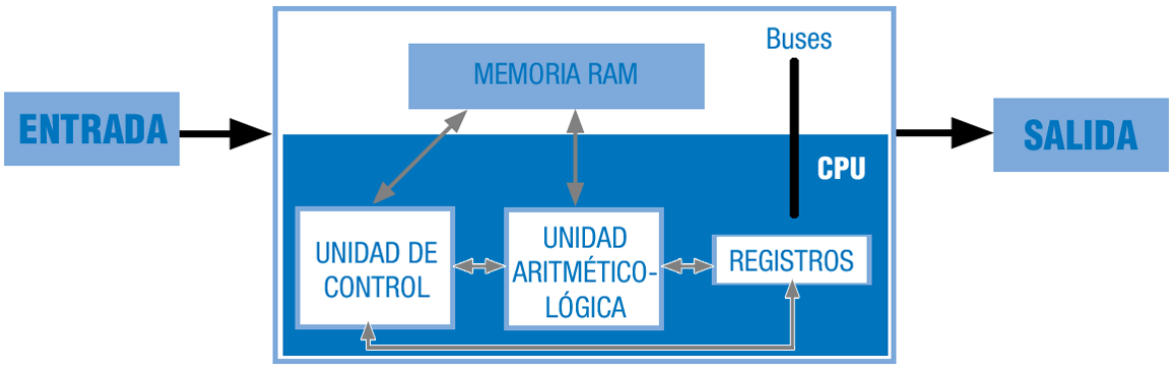

Figura 1.2.1: Arquitectura John Von Neuman

Esta relación software-hardware la podemos analizar desde dos puntos de vista diferentes:

a **Desde el punto de vista del sistema operativo**: el sistema operativo es el encargado de coordinar el funcionamiento del ordenador, actuando entre este y las aplicaciones que están corriendo en ese momento, gestionando los diferentes recursos hardware que requieren estas aplicaciones (CPU, RAM, interrupciones, dispositivos de E/S..).

b **Desde el punto de vista de las aplicaciones**: una aplicación solo es un conjunto de programas escritos en algún lenguaje de programación. Hay multitud de lenguajes de programación, con la característica de que todos están escritos en un idioma que el ser humano puede entender. El hardware por otro lado solo es capaz de interpretar señales eléctricas que se traducen en secuencias de 0 y 1 (código binario). Para que estás aplicaciones puedan ejecutarse en el hardware debe darse un proceso de "traducción"que veremos mas adelante.

1.3 Desarrollo de Software

Entendemos por **Desarrollo de Software** todo el proceso desde que se concibe la idea hasta que el programa está implementado y funcionando en el ordenador. Aunque en principio pueda parecer una tarea simple, consta de una series de pasos de obligado cumplimiento, pues solo así podemos garantizar que las aplicaciones creadas son eficientes, seguras, fiables y responden a las necesidades de los usuarios finales.

Como veremos más adelante en la unidad, esta serie de pasos se le suele denominar **Etapas** en el desarrollo de software. Según el orden y la forma en la que se llevan a cabo estas etapas hablaremos de diferentes ciclos de vida del software.

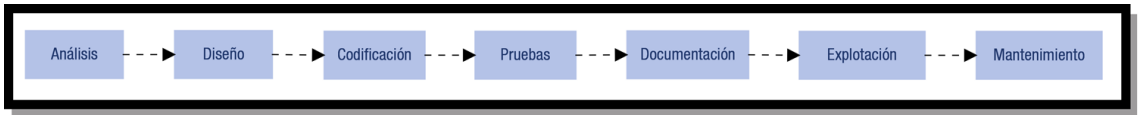

Figura 1.3.1: Etapas del Desarrollo de Software

Cabe destacar que la construcción de software es un proceso muy complejo y que requiere de una gran coordinación y disciplina del grupo de trabajo que lo desarrolle.

1.3.1 Ciclos de Vida del Software

Ya hemos visto que para crear software debemos seguir un número de pasos conocidos como ciclo de vida del software. Más adelante en esta unidad veremos en que consiste cada paso mas detalladamente. Por ahora, vamos a centrarnos en ver los diferentes ciclos de vida del software que existen atendiendo a como se desarrollan dichos pasos.

Aunque podemos encontrar diferentes clasificaciones sobre los distintos tipos de ciclos de vida del software los mas conocidos y utilizados son los siguientes:

1. **Modelo en Cascada**: es el modelo de vida clásico del software. Actualmente es prácticamente imposible de utilizar, salvo para desarrollos muy pequeños, ya que es necesario conocer todos los requisitos con antelación y las etapas pasan de una a otra sin retorno, presuponiendo que no ha habido errores en la etapa anterior.
2. **Modelo en Cascada con Realimentación**: es uno de los modelos más utilizados. Proviene del modelo anterior, pero se introduce una realimentación entre etapas, permitiendo que podamos volver hacia atrás en cualquier momento para corregir, depurar o modificar algún aspecto. Es el modelo perfecto si el proyecto es es rígido (pocos cambios y poco evolutivo) y los requisitos están claros. No obstante, si se prevén muchos cambios durante el desarrollo no es el modelo más idóneo.
3. **Modelos Evolutivos**: son los modelos más modernos y tienen en cuenta el aspecto cambiante y evolutivo del software. Dentro de este modulo podemos encontrar dos variantes:

3.1. **Modelo Iterativo Incremental**: ésta basado en el modelo en cascada con realimentación, donde las fases se repiten y refinan, propagando su mejora a las fases siguientes. El proyecto se realiza en pequeñas porciones (**incrementa**) en sucesivas iteraciones (**sprints**) al final de las cuales se ve lo que se ha desarrollado, pudiendo hacer correcciones o modificaciones antes de la siguiente iteración o incluso añadir nuevos requerimientos (**adaptativo**). Cada sprint debe proporcional un resultado completo preparado para entregárselo al clientes.

3.2. **Modelo en Espiral**: es una combinación del modelo anterior con el modelo en cascada. En éste, el software se va construyendo repetidamente en forma de versiones que son cada vez mejores, debido a que se va incrementando la funcionalidad. Es un modelo muy complejo.

En la actualidad, la metodologías de desarrollo ágil, enmarcadas dentro de los modelos evolutivos, están en auge, y metodologías como **Scrum**, **TDD** o **BDD**, así como metodologías híbridas basadas en estas, se están convirtiendo en el estándar de la industria. [4]

1.3.2 Herramientas de Apoyo al Desarrollo De Software

En la práctica, vamos a contar con un conjunto de herramientas que nos van a facilitar llevar a cabo las diferentes etapas del ciclo de vida de desarrollo, automatizando las tareas, ganando con ello fiabilidad y tiempo, y permitiéndonos centrarnos en los requerimientos del sistema y el análisis del mismo.

Las herramientas **CASE** son un conjunto de aplicaciones que se usan en el desarrollo de software con el propósito de reducir costes y tiempo de proceso, aumentando la productividad. Estas herramientas pueden ayudarnos prácticamente en cualquier etapa del proceso de desarrollo.

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 herramientas CASE. Hoy en día se usa para referirnos al desarrollo rápido de interfaces de usuario o entornos de desarrollo integrados (**IDE**) completos.

La tecnología CASE trata de automatizar el proceso de desarrollo para que mejore las calidad del proceso y el resultado final. En concreto, estas herramientas permiten:

- Mejorar la planificación del proyecto.
- Darle agilidad al proceso.
- Generar software mas reutilizable.
- Creación de aplicaciones más estandarizadas.
- Mejorar las tareas de mantenimiento de las aplicaciones.
- Permiten visualizar todo proceso de desarrollo de forma gráfica.

Las herramientas CASE se pueden clasificar según su funcionalidad o la función que desempeñan dentro del proceso de desarrollo.

Atendiendo a la función que desempeñan en cada fase del proceso, las herramientas CASE pueden ser:

- **U-CASE**: ofrecen ayuda en las fases de planificación y análisis de requisitos.
- **M-CASE**: ofrecen ayuda en el análisis y diseño.
- **L-CASE**: ofrecen ayuda en la programación del software, detección de errores en código, depuración de programas y pruebas, y en la generación de la documentación.

Si tenemos en cuenta su funcionalidad, se pueden diferenciar algunas como:

- Herramientas de generación semiautomáticas de código.
- Editores **UML**.
- Herramientas de refactorización de código.
- Herramientas de mantenimiento, como los sistemas de control de versiones.

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

1.4 Lenguajes de Programación

Podemos definir un **lenguaje de programación** como un idioma creado artificialmente, que se compone de un conjunto de símbolos y normas que se aplican sobre un alfabeto para obtener un código que el hardware de la computadora pueda entender y procesar. Es el instrumento que tenemos para que el ordenador realice las tareas que necesitamos, dicho de otra forma.

Hay multitud de lenguajes cada uno con su estructura y símbolos. Además cada lenguaje está enfocado en la realización de tareas o áreas determinadas. Por ello, es muy importante la elección del lenguajes o lenguajes de programación en un proyecto.

Los lenguajes de programación han ido evolucionando a través de la historia, podemos ver esta evolución de forma simplificada en la siguientes figura:

Figura 1.4.1: Evolución de los lenguajes de programación

Las principales características de estos lenguajes de son las siguientes:

- **Lenguajes Máquina**:
  - Sus instrucciones están compuestas por unos y ceros.
  - Es el único lenguaje que entiende el ordenador, no necesita traducción.
  - Fue el primer lenguaje utilizado.
  - Es único para cada procesador, es decir, no es portable de un equipo a otro.
  - Hoy en día nadie lo usa.
- **Lenguaje Ensamblador**:
  - Sustituyó el lenguaje máquina para facilitar la programación.
  - Se programa usando mnemotécnicos (instrucciones complejas).
  - Necesita traducción al lenguaje máquina para poder ejecutarse.
  - Sus instrucciones hacen referencia a la ubicación física de los archivos y registros.
  - Es difícil de utilizar.
- **Lenguajes de alto nivel**:

- Sustituyeron al ensamblador para facilitar la programación.
- Se utilizan sentencias y órdenes derivadas el inglés. Necesita traducción al lenguaje máquina.
- Son más cercanos al razonamiento humano.
- Son los más utilizados hoy en día.

- **Lenguajes Visuales**:
  - Están sustituyendo a los lenguajes de alto nivel basados en código.
  - Se programa gráficamente usando 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.

Glosario

CASE Computer Aided Software Engineerig. 7, 8

Desarrollo de Software Conjunto de procesos desde que nace una idea hasta que se convierte en software. 6, 8

ERP Enterprise Resource Planning o Sistema de Planificación de Recursos Empresariales, son programas que se usan para la gestión empresarial y que se hacen cargo de distintas operaciones internas, desde la producción, la distribución o incluso los recursos humanos. 5, 8

IDE Integrated Development Environment. 7, 8

Linux Sistema Operativo tipo Unix compuesto por software libre y de código abierto que sirve como base para multitud de distribuciones como Debian, Slackware, Ubuntu, Gentoo.... 4, 8

macOS Sistema Opertativo basado en Unix, más concretamente en FreeBSD, y que está desarrollado y comercializado por Apple desde 2001. 4, 8

Microsoft Windows Sistema Operativo desarrollado por Microsoft que actualmente se encuentra en la versión 11. 4, 8

RAD Rapid Application Development. 7, 8

UML Unified Modeling Language. 8

Bibliografía

[1] Wikipedia - Hardware. https://es.wikipedia.org/wiki/Hardware.

[2] Wikipedia - Software. https://es.wikipedia.org/wiki/Software.

[3] Joaquin Marquéz y Sergio Ernesto Gastón Perez. Tipos de software. https://es.slideshare.net/susahhreyyhha/tipos-de-software-15979630.

[4] Amna Batool. A survey of key challenges of agile in global software development: A case study with malaysia perspective. *International Journal of Industrial and Manufacturing Engineering, Vol13, nº10.*