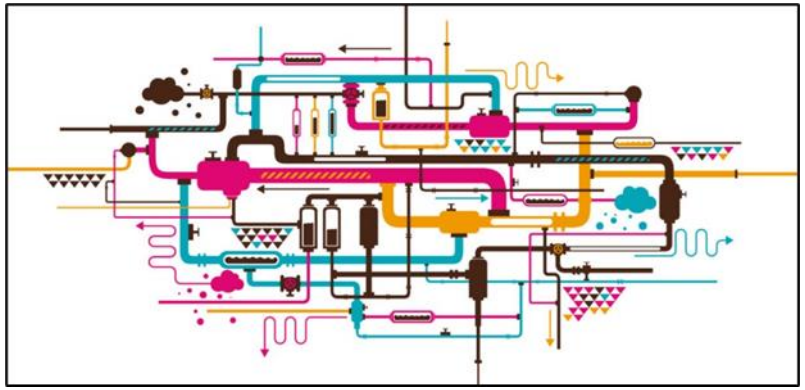
Florida

1º DAM/DAW EDE

U1. Desarrollo de software y Entornos

Introducción y conceptos



Pascual Martínez

Hardware - Software

Partes o elementos de un dispositivo informático:

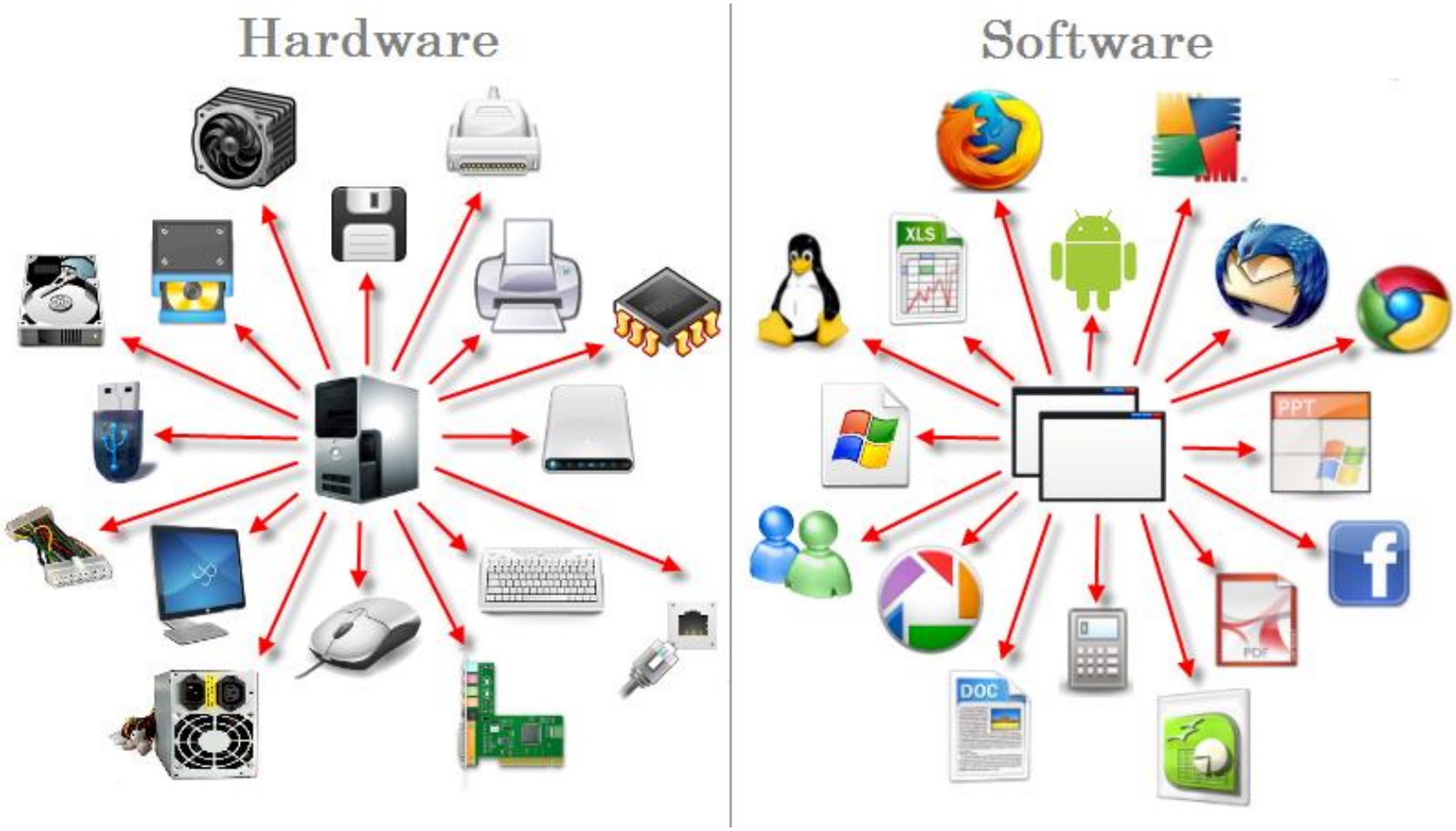
- **Hardware (HW):** conjunto de componentes físicos o materiales que componen un dispositivo o PC.
- **Software (SW):** conjunto de programas y aplicaciones que residen y se ejecutan sobre el HW.

Ambos elementos **se complementan** para que un dispositivo pueda funcionar.

Tipos de Software

Según la función:

- **Sistemas operativos (S.O.):** es el software base que ha de estar instalado y configurado en un ordenador para que las aplicaciones puedan ejecutarse y funcionar.
- **Software de programación:** herramientas para desarrollar otros programas informáticos. Ejemplo: **entornos de desarrollo**.
- **Aplicaciones:** son un conjunto de programas con finalidad más o menos concreta.


Hardware
Software
XLS
PPT
DOC
PDF

Desarrollo de Software

- Es el **proceso** desde que se detecta una necesidad o se concibe una idea, hasta que se dispone de una herramienta software en funcionamiento que la resuelve, o más bien, hasta que esta herramienta o solución deja de utilizarse:
  - **Inicio**: se detecta una necesidad o se concibe una idea.
  - **Objetivo**: desarrollar e implantar una **solución software**, que resuelva la necesidad y cumpla los acuerdos previstos (tiempo, forma y coste).
  - **Final**: la herramienta o solución deja de utilizarse y mantenerse.

Fases clásicas de un proyecto de desarrollo de SW

- **Análisis:** se recopilan los requerimientos y se analizan en profundidad.
- **Diseño:** se definen en detalle las funcionalidades necesarias para cubrir los requerimientos.
- **Desarrollo:** se codifica la solución en base al diseño realizado.
- **Pruebas:** se verifican y testean de los resultados del desarrollo de la solución.
- **Implantación:** etapa de puesta en marcha de la solución. Instalaciones. Capacitación de usuarios.
- **Mantenimiento y evolución:** esta etapa puede durar toda la vida útil de la solución.

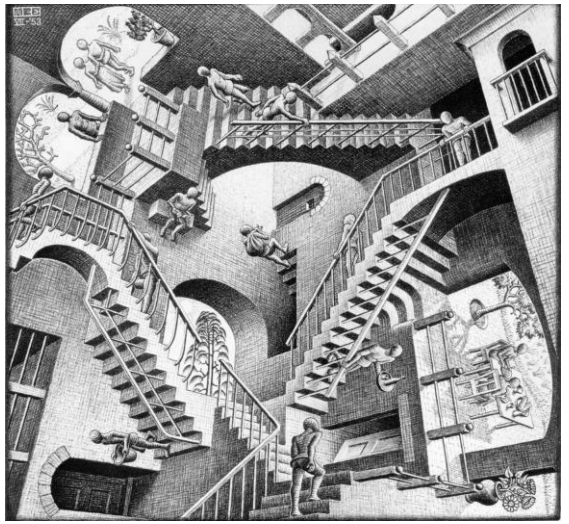
Fases clásicas de un proyecto de desarrollo de SW

Fases clásicas de un proyecto de desarrollo de SW

- **Análisis**:
  - Es la **primera fase** de un proyecto. **Las demás dependen de su precisión.**
  - **¿Qué hay que hacer en esta fase? Objetivos:**
    - Definir los **requerimientos**, funcionales o no funcionales.
    - Definir una **estrategia**.
    - Definir una **planificación** de tareas e hitos.
  - La **comunicación e implicación** de **todo el personal implicado** en el proyecto es un factor clave para definir **¿qué debe realizar la solución?**.

Fases clásicas de un proyecto de desarrollo de SW

- **Diseño**:
  - **¿Cómo se van a resolver los requerimientos?**. Dependerá de la naturaleza de los requerimientos, de la estrategia y de las circunstancias del proyecto.
  - **¿Qué hay que hacer en esta fase?**
    - **Análisis técnico de los requerimientos**: mediante gráficos, diagramas, algoritmos, textos, etc.
    - **Decisiones para organizar y cuantificar recursos**: personas, infraestructura, Entornos de desarrollo, lenguaje/s de programación, gestor de bases de datos.



Fases clásicas de un proyecto de desarrollo de SW

- **Desarrollo o codificación:**

  - **¿Qué hay que hacer en esta fase? Objetivos:**
    - **Codificar** mediante uno o varios lenguajes de programación.
    - **El código puede pasar por diferentes estados:**
      - **Código fuente:** lo generan l@s **programador@s**. Habitualmente mediante el uso de lenguajes de programación de **alto nivel**.
      - **Código objeto:** código binario resultado de **compilar** el código fuente. **Código intermedio**, no es inteligible por las personas, y todavía no es ejecutable por un sistema operativo.
      - **Código ejecutable o código máquina:** código binario resultante de **enlazar o linkar** el código objeto con rutinas y bibliotecas utilizadas. El sistema operativo lo podrá cargar en memoria RAM y **ejecutarlo directamente**.

Fases clásicas de un proyecto de desarrollo de SW

- **Pruebas:**
  - Conforme se va disponiendo de elementos desarrollados, es crucial realizar pruebas.
  - **¿Qué hay que hacer en esta fase? Objetivos:**

    - Se distinguen **múltiples tipos** de pruebas:
      - **Unitarias:** se testean pequeños elementos aislados.
      - **Pruebas de integración:** se testea que los elementos se integran correctamente.
      - **Pruebas sistema:** se testea el funcionamiento de un proceso completo.
    - El resultado de las pruebas puede provocar que algunos elementos retrocedan, incluso a la fase de análisis.

Fases clásicas de un proyecto de desarrollo de SW

- **Implantación:**
  - Cuando una solución alcanza cierta fiabilidad, se prepara el escenario donde se usará.
  - **¿Qué hay que hacer en esta fase?:**
    - **Instalación** de la solución en la infraestructura donde se ejecutará.
    - **Configuración** de la solución.
    - **Capacitación** de usuarios.
    - **Explotación/producción**.

Fases clásicas de un proyecto de desarrollo de SW

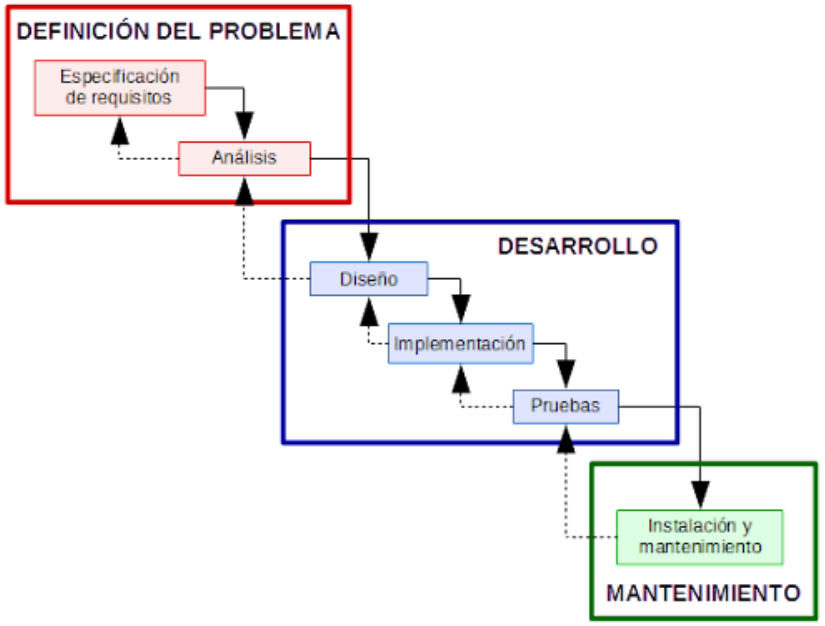
- **Mantenimiento y evolución:**
  - La naturaleza del software es cambiante, necesitará actualizarse, adaptarse y evolucionar.
  - **¿Qué hay que hacer en esta fase?:**
    - El **mantenimiento** de una solución software puede durar toda la vida del producto.
    - Existen varios tipos de mantenimiento:
      - **Correctivo:** corrección de errores y fallos de funcionamiento.
      - **Adaptativo:** modificaciones y actualizaciones de la funcionalidad existente.
      - **Evolutivo:** nuevas funcionalidades, nuevas versiones.

Fases clásicas de un proyecto de desarrollo de SW

Modelos del ciclo de vida del software

En función de la **secuencia** que se establezca entre las fases o etapas y de las **reglas** que establezcan entre ellas, podemos encontrar diferentes **modelos**:

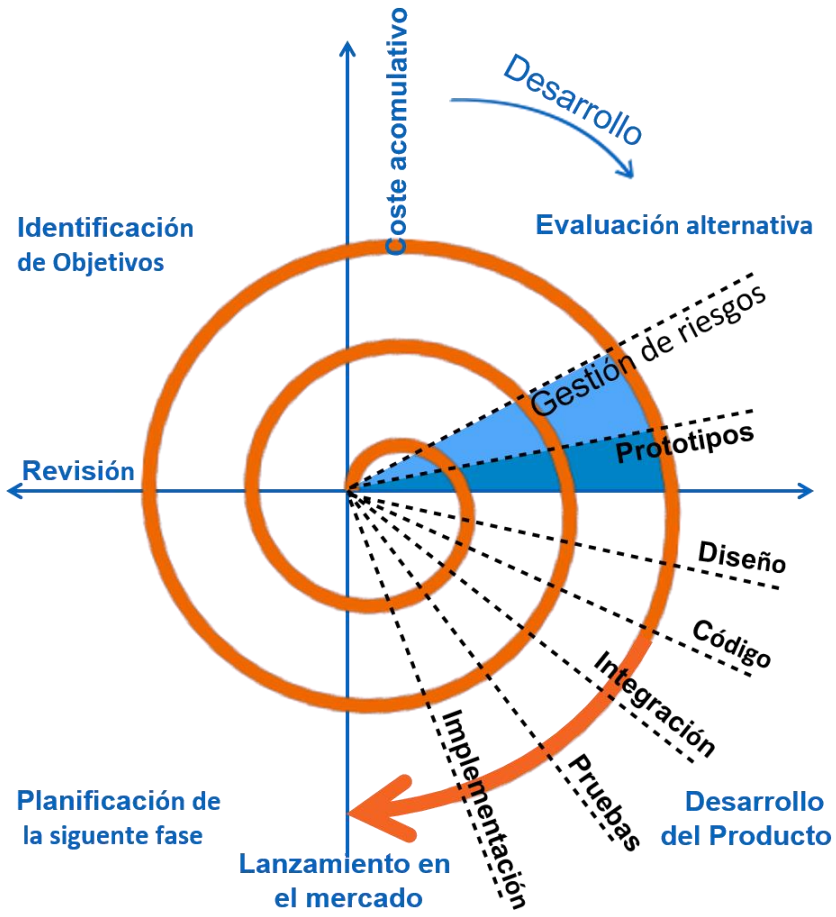
- **En cascada:**
  - Es el modelo más clásico y menos flexible.
  - Una **variante** muy interesante y flexible es el **modelo en cascada con retroalimentación.**



Modelos del ciclo de vida del software

Modelos más actuales, que tienen en cuenta la **naturaleza altamente cambiante del SW**, son los **modelos evolutivos**:

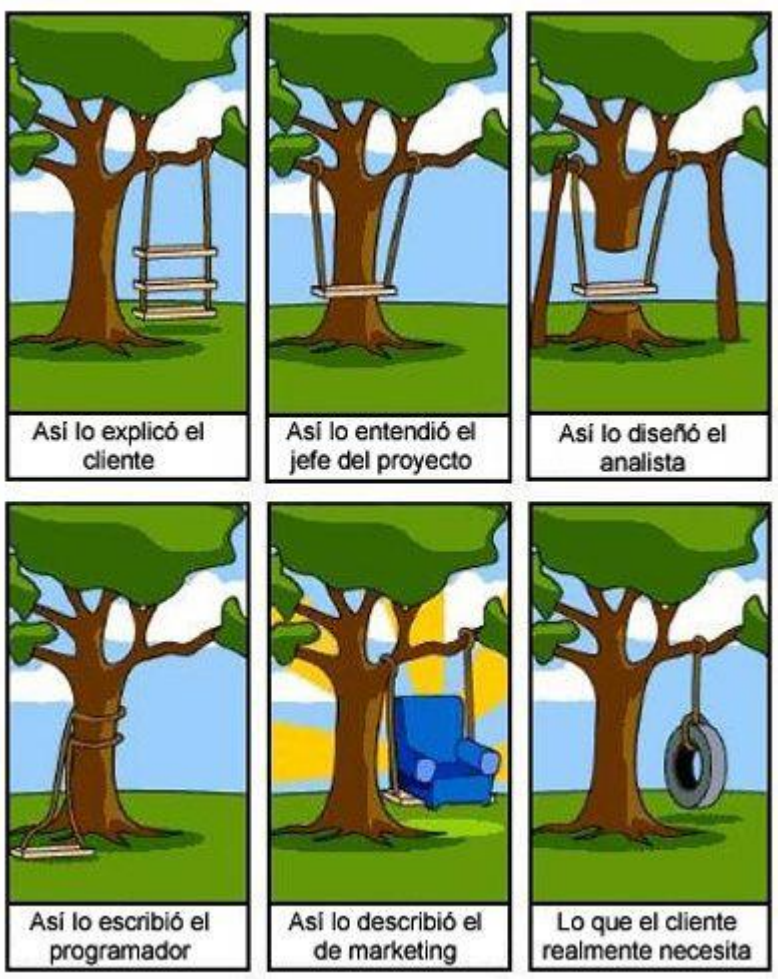
- **En espiral:**
  - Es un modelo flexible.
  - El software se va construyendo **repetidamente** en forma de **versiones mejoradas**.
  - Se apoya en las fases clásicas.
  - Puede ser **complejo** en su implementación.
  - Permite **reducir los riesgos** en cada versión.



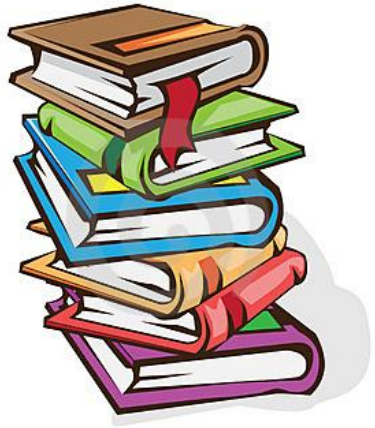
Modelos del ciclo de vida del software

- Existen más modelos (modelo en V, modelo por prototipos, etc.), y más que surgirán…
- No existen modelos “buenos” y “malos”, dependerá de las circunstancias del proyecto..
- Es habitual utilizar una combinación de modelos, en mayor o menor medida, a la hora de desarrollar un proyecto.

Desarrollo de Software, una pincelada de humor



Documentación



- La documentación es una labor muy importante.
- Se puede considerar como una fase clásica más.
- Desde la antigüedad, la documentación ha permitido **universalizar** los detalles de un proyecto.

Mejora e integración continua

- **Mejora continua:**
  - Consiste en repetir este procedimiento de forma permanente:
    - **Analizar procesos** y estudiarlos en detalle.
    - **Realizar adecuaciones** para mejorarlos y minimizar los errores.

- **Integración continua:**
  - Consiste en **integrar las mejoras** de varios equipos o desarrolladores (contribuidores) en un único proyecto de software, de forma periódica.

Gestión ágil de proyectos

- Consiste en realizar **entregas** de **forma continua e iterativa**.
- Las **personas** y su interacción se posicionan como **núcleo** del proyecto. Por encima de las herramientas y los procedimientos.
- La estrecha **colaboración y comunicación** entre los componentes del proyecto es una necesidad primordial para obtener éxito.

Gestión ágil de proyectos

- La **unidad básica** de ejecución es la **iteración**.
- En cada iteración se debe marcar una serie de objetivos, de forma que **cada iteración va aportando valor a la solución**.

Gestión ágil de proyectos

- Existen **definiciones** específicas de **pautas de trabajo** basadas en la gestión ágil de proyectos, como, por ejemplo:
  - Scrum.
  - Kanban.
  - …

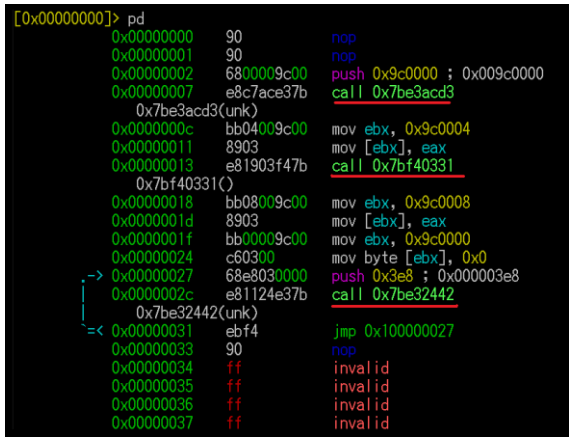
- También existen **herramientas** orientadas a este tipo de gestión:
  - Trello.
  - Jira.
  - …

Lenguajes de programación

- Son **idiomas** creados artificialmente para que una máquina los pueda ejecutar.
- Formados por un conjunto de **símbolos y** unas **normas** aplicables.

Lenguajes de programación

- **Clasificación** de los **lenguajes**:
  - **Según la proximidad con el lenguaje humano:**
    - **Alto nivel**: próximos al lenguaje humano. Los usados habitualmente.
    - **Bajo nivel**: más próximos al funcionamiento interno de una máquina:
      - **Ensamblador.**
      - **Código máquina.**

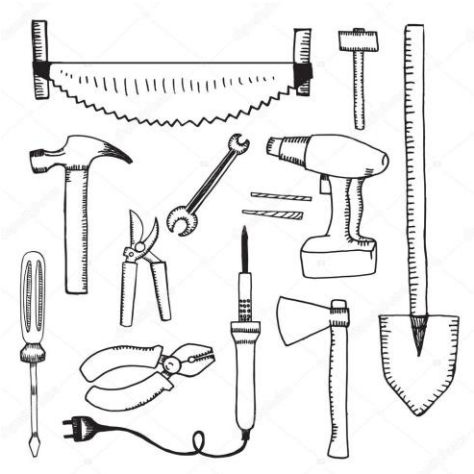


Lenguajes de programación

- **Clasificación** de los **lenguajes**:
  - **Según la técnica de programación utilizada:**
    - **Estructurados:** conjunto ordenado de sentencias con inicio y final (Pascal, C, Fortran, Cobol, BASIC, etc.).
    - **Modulares:** permiten dividir un código en fragmentos o módulos reutilizables (Pascal, C, Cobol, RPG, etc.).
    - **Orientados a objetos:** los programas son un conjunto de objetos que interactúan entre sí. Es el paradigma más extendido (C++, C#, Delphi, Java, JavaScript, PHP, etc.).
    - **Visuales:** permiten generar programas mediante la manipulación de elementos gráficos (Scratch, Bolt, MakeCode, etc.).

Entornos de desarrollo

- En inglés **Integrated Development Environment (IDE).**
- Aplicación con un conjunto de **herramientas para facilitar el desarrollo de software**.
- **Características:**
  - Diseñados para **maximizar la productividad**.
  - Ofrecen **utilidades** para la creación y depuración de software.
  - Facilitan la **reutilización** de componentes.
  - Ayudan a **reducir la configuración**.
  - Algunos soportan **múltiples lenguajes** de programación.
  - Permiten **personalización** mediante la activación o desactivación de herramientas y componentes.



Entornos de desarrollo

- **Ventajas:**
  - **Facilita** el aprendizaje.
  - Dispone de mecanismos de **ayuda avanzada**, como auto completar.
  - Proporciona **mensajes de alerta o error** durante todo el proceso.
  - **Incrementa la productividad** del desarrollador mediante herramientas y plantillas.
- **Inconvenientes:**
  - Suelen **consumir** muchos **recursos**.
  - Generan **dependencia**, cierta cautividad.

Entornos de desarrollo

- **Ejemplos:** en el mercado podemos encontrar múltiples opciones. Por ejemplo:
  - **Visual Studio:**
    - Desarrollado por **Microsoft** y comercializado **bajo licencia privativa**.
    - Es compatible con lenguajes como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP, etc.

  - **Eclipse:**
    - Actualmente lo gestiona una fundación que fomenta el **código abierto**.
    - Es **multiplataforma** (Windows, Linux y Mac).
    - Es posible desarrollar usando Java, C, C++, JSP, perl, Python, Ruby, PHP, etc.

  - **NetBeans:**
    - Desarrollado por **Apache (Oracle)**, **basado en código abierto**.
    - Es el “oficial” de Java, también podemos desarrollar en otros lenguajes como PHP, C, C++, etc.