

# FUNDAMENTOS DEL DESARROLLO DE SOFTWARE (UD01)

## El Proceso y el Algoritmo

- **Naturaleza del Desarrollo:** El Desarrollo de Software es una **disciplina de Ingeniería** que aplica un enfoque **sistemático, disciplinado y cuantificable** a la producción y mantenimiento de software. Su objetivo es la creación de artefactos **fiables, eficientes y seguros**.
- **Algoritmo:** Una secuencia **finita, bien definida y no ambigua** de instrucciones para alcanzar un objetivo específico o resolver una clase de problemas.
  - **Algoritmo:** Especificación de la **lógica de negocio** o del problema (**el qué**).
  - **Programa:** La **implementación** del algoritmo en un lenguaje de programación específico (**el cómo**).

**Ejemplo:** La especificación de la **fórmula** para el **cálculo del promedio aritmético** de  $n$  elementos constituye el algoritmo. Su transcripción en **código fuente** (e.g., Python) es el programa.

---

## 2 Ciclo de Vida del Software (SDLC)

El SDLC es el marco que describe las fases de vida de un sistema de software, desde su concepción hasta su retiro. Se compone de seis fases fundamentales:

1. **Planificación: Estudio de viabilidad** y definición de objetivos, **alcance del proyecto** y asignación de recursos.
2. **Análisis de Requisitos (Fase Crítica):** Proceso de **determinación, documentación y gestión de los requisitos del usuario y del sistema**. Se produce la **Especificación de Requisitos del Software (ERS)**.
3. **Diseño:** Se define la **arquitectura del sistema**, el diseño de la base de datos y la interfaz de usuario. Se produce el **Modelo de Diseño**.
4. **Implementación (Codificación):** Traducción del modelo de diseño a **código fuente** funcional.
5. **Pruebas (Testing):** Verificación y validación para asegurar que el sistema cumple con la ERS y que está **libre de defectos (defect-free)**.
- 6.
7. **Mantenimiento:** La fase más larga del SDLC, incluye **mantenimiento correctivo** (corrección de *bugs*), **adaptativo** (cambios de entorno) y **perfectivo** (mejoras).

funcionales).

---

### 3 Roles en el Equipo de Desarrollo

- **Arquitecto de Software:** Define la **estructura de alto nivel** del sistema, las tecnologías y los **patrones de diseño** (*Design Patterns*).
  - **Analista de Sistemas/Negocio:** Sirve de **enlace** entre el cliente y el equipo técnico, transformando las **necesidades funcionales** en **requisitos técnicos** detallados.
  - **Programador/Desarrollador:** **Construye y prueba** los módulos de código, siguiendo las especificaciones de diseño.
  - **Tester (Ingeniero de Calidad):** Diseña y ejecuta **casos de prueba** (*test cases*) y **reporta defectos** (*bug reporting*).
- 

## ESTRATEGIAS Y METODOLOGÍAS DE DESARROLLO

### 1 Modelo Clásico: Cascada (Waterfall)

- **Naturaleza:** Un **modelo predictivo y secuencial lineal**. La finalización de una fase es un **prerrequisito estricto** para el inicio de la siguiente.
  - Fases (las del SDLC)
  - **Debilidad:** Presenta **alta rigidez**. La **gestión del cambio** (*Change Management*) es costosa, ya que el *feedback* del cliente es **tardío** (al final del proceso).
- 

### 2 Metodología Ágil y Scrum

- **Naturaleza:** Un enfoque de desarrollo **iterativo e incremental** que promueve una entrega rápida y continua de valor.
- **Ventajas Principales:**
  - **Adaptación al cambio** sobre el seguimiento de un plan.
  - Entrega de **software funcional frecuente**.
  - **Colaboración continua** con el cliente.
  - **Reducción del riesgo** mediante ciclos cortos (**Sprints**).
- **Manifiesto Ágil (Principios Clave):**
  - **Individuos e interacciones** sobre procesos y herramientas.
  - **Software funcional** sobre documentación exhaustiva.
  - **Colaboración con el cliente** sobre negociación contractual.

- **Respuesta al cambio** sobre seguir un plan.
- 

## LENGUAJES, PARADIGMAS Y TRADUCCIÓN

### 1 Paradigmas de Programación

- **Imperativo:** Se basa en **órdenes explícitas** que modifican el **estado del programa** a través de sentencias de asignación y control de flujo. (Ejemplo: C).
  - **Orientado a Objetos (POO):** Centrado en la encapsulación de datos y comportamiento en **objetos** definidos por **clases**, promoviendo **herencia**, **polimorfismo** y **abstracción**. (Ejemplo: Java).
  - **Declarativo:** Se especifica **el resultado deseado** (*qué*) sin detallar el algoritmo exacto (*cómo*). (Ejemplo: SQL).
- 

### 2 Clasificaciones Clave

- **Sistema de Tipos:**
    - **Estático y Fuerte (e.g., Java):** La **verificación de tipos** se realiza en **tiempo de compilación**, lo que previene errores en tiempo de ejecución.
    - **Dinámico y Débil (e.g., JavaScript):** La verificación ocurre en **tiempo de ejecución** (*runtime*), ofreciendo flexibilidad a costa de potencialmente más *bugs*.
  - **Nivel de Abstracción:**
    - **Bajo Nivel:** Lenguajes **orientados a la máquina**, directamente relacionados con la **arquitectura del procesador** (e.g., Ensamblador).
    - **Alto Nivel:** Lenguajes más **cercanos al lenguaje humano**, que **abstraen** la complejidad del *hardware* (e.g., Python, Java).
- 

### 3 Proceso de Traducción y Portabilidad

- **Fases del Compilador:**
  - **Análisis Léxico:** El código fuente se descompone en **tokens** (unidades mínimas con significado).
  - **Análisis Sintáctico:** Se verifica que la secuencia de *tokens* cumpla con la **gramática** formal del lenguaje (construcción del **Árbol de Sintaxis Abstracta - AST**).
  - **Análisis Semántico:** Se comprueba el **significado lógico** y la **coherencia de**

**tipos.**

- **Portabilidad en Java:** El compilador Java genera **Bytecode**, un **código intermedio** que es interpretado por la **Máquina Virtual de Java (JVM)**. Esto garantiza la **portabilidad** entre diferentes sistemas operativos.
  - **Lema:** *"Write Once, Run Anywhere"* (Compila una vez, ejecuta en cualquier lugar).

PREGUNTAS EXAMEN:

1.DESCRIBE LOS DIFERENTES ROLES EN EL EQUIPO DE DESARROLLO

2.DESCRIBE LAS FASES DEL MODELO CASCADA

3.NOMBRA 5 SOFTWARES DE USUARIO, 4 SOFTWARES DE PROGRAMACIÓN Y 3 SOFTWARES DE SISTEMAS

4.ESCRIBE TODO LO QUE SEPAS DE SCRUM

5.¿En qué se diferencia una página web de una aplicación web? / ¿En qué se diferencia una aplicación web de una aplicación instalable o de escritorio?