# UNIVERSIDAD PRIVADA DEL NORTE

## FACULTAD DE INGENIERÍA

**CARRERA:** Ingeniería de sistemas computacionales

**CURSO:** Introducción a la ingeniería de sistemas computacionales

**DOCENTE:** Ing. José Castillo Zumarán

**TEMA:** Software e Ing. de Software (metodologías).

## Índice de contenidos

# Software

## 1. Definición de Software

El software es un conjunto de instrucciones y datos que controlan el hardware y permiten realizar tareas específicas. Existen diferentes clasificaciones, como software de sistema, software de aplicación y software de desarrollo, cada uno diseñado para desempeñar un rol específico.

## 2. Tipos de Software

-> Software de Sistema: Maneja los recursos del hardware. Ejemplos incluyen sistemas operativos como Windows, Linux y MacOS. Es la interface que permite los usuarios manejaren los hardwares.

-> Software de Aplicación: Dirigido a usuarios finales para tareas específicas, como aplicaciones de productividad, videojuegos, herramientas de diseño, Word, Excel, PowerPoint, etc.

-> Software de Programación o Desarrollo: Utilizado para crear otros programas, como compiladores, editores de texto y entornos de desarrollo (IDE), el visual estudio code por ejemplo.

-> Software Integrado (Embebido): Diseñado para controlar dispositivos y sistemas integrados, por ejemplo, el software que se encuentra en electrodomésticos o sistemas automotrices.

## 3. Ciclo de Vida del Software

El ciclo de vida del software se refiere a las etapas por las que pasa el desarrollo de un programa. Los modelos más comunes son:

-> Análisis de Requisitos: Identificación de necesidades y expectativas de los usuarios.

-> Diseño: Creación de la arquitectura del sistema, definiendo su estructura y comportamiento.

Implementación: Proceso de codificación en lenguajes de programación como C#, Java, Python, etc.

-> Pruebas: Validación de funcionalidad, detección de errores y evaluación de la calidad.

-> Mantenimiento: Corregir errores, mejorar el rendimiento y actualizar la funcionalidad según nuevas necesidades.

-> Documentación: Hacer la documentación de un software es una parte fundamental del desarrollo, ya que permite que otros (y tú mismo en el futuro) entiendan cómo funciona el programa, cómo instalarlo, cómo usarlo y cómo contribuir con él.

## 4. Metodologías de Desarrollo de Software

Las metodologías de desarrollo ayudan a gestionar el proceso de creación de software y pueden ser:

### 🎓 METODOLOGÍAS TRADICIONALES

### 1. Cascada (Waterfall)

Explicación:

Es el modelo secuencial más clásico. Cada fase debe completarse antes de pasar a la siguiente.

Fases:

->Requisitos

->Diseño

->Implementación

->Pruebas

->Mantenimiento

Ventajas:

->Fácil de gestionar y documentar.

->Ideal para proyectos con requisitos bien definidos desde el principio.

Desventajas:

->Poco flexible ante cambios.

->No permite ver resultados tempranos.

### 2. Modelo en V (V-Model)

Explicación:

Extiende el modelo en cascada relacionando cada fase de desarrollo con una fase de pruebas correspondiente, en forma de "V".

Ventajas:

->Mejora la verificación y validación.

->Aumenta el control de calidad.

Desventajas:

->Rigidez similar al modelo en cascada.

->Costoso si hay cambios tardíos.

## 3. Modelo Incremental

Explicación:

El sistema se construye en partes o incrementos funcionales, cada uno de los cuales se desarrolla por separado.

Ventajas:

->Entregas tempranas y funcionales.

->Más flexible que cascada.

Desventajas:

->Puede generar dependencias mal gestionadas entre incrementos.

## 4. Modelo Espiral

(Boehm, 1986)

Explicación:

Combina elementos del modelo en cascada y el enfoque iterativo, con énfasis en la evaluación de riesgos. Es como una mescla entre el modelo Cascada y el modelo V.

Fases:

->Planificación

->Análisis de riesgos

->Desarrollo y prueba

->Evaluación del cliente

Ventajas:

->Ideal para proyectos grandes y críticos.

->Se adapta bien al cambio.

Desventajas:

->Complejo y costoso.

->Requiere experiencia en gestión de riesgos.

## ⚙ METODOLOGÍAS ÁGILES

(Agile Manifesto, 2001)

Las metodologías ágiles enfatizan:

->Colaboración con el cliente.

->Entregas frecuentes

->Adaptación al cambio.

->Equipos autoorganizados.

### 1. Scrum

Explicación:

Marco de trabajo que divide el desarrollo en iteraciones llamadas sprints [1] (2-4 semanas).

Roles:

->Product Owner [2]

-> Scrum Master [3]

->Equipo de desarrollo

Artefactos:

->Product Backlog [4]

->Sprint Backlog [5]

->Incremento

Ventajas:

->Alta adaptabilidad.

->Entregas frecuentes de valor.

Desventajas:

->Requiere disciplina y compromiso del equipo.

[1] Sprints: "sprint" (o ciclo de sprint) se refiere a un período de tiempo fijo, generalmente de 2 a 4 semanas, durante el cual un equipo trabaja para completar una porción de trabajo y generar un incremento de valor. Los sprints son fundamentales para la entrega iterativa y el trabajo en ciclos de desarrollo, permitiendo una mayor flexibilidad y adaptación a los cambios.

[2] Product Owner: Un Product Owner (Propietario del Producto) es un rol esencial en proyectos ágiles, particularmente en Scrum, que se encarga de maximizar el valor del producto al definir la visión del producto, priorizar tareas y gestionar el Product Backlog. Su función es actuar como nexo entre las partes interesadas, el equipo de desarrollo y los usuarios finales, asegurando que el producto entregado sea el más valioso posible.

[3] Scrum Master: Un Scrum Master es un líder de equipo en metodologías ágiles, específicamente Scrum. Se enfoca en asegurar que el equipo entienda y aplique los principios de Scrum, facilita la colaboración, elimina obstáculos y ayuda al equipo a alcanzar sus objetivos.

[4] Product Backlog: El "product backlog" es una lista dinámica y priorizada de tareas, características y requisitos que guían el desarrollo de un producto. En esencia, es una guía para el equipo de desarrollo, que evoluciona constantemente a medida que el proyecto avanza.

[5] Sprint Backlog: El backlog de sprint es una lista de tareas que un equipo Scrum se compromete a realizar durante un sprint específico, una iteración corta en el marco ágil Scrum. Es una parte del Product Backlog (lista de trabajo para el proyecto completo) y se crea durante la planificación del sprint.

## 2. Kanban

Explicación:

Método visual que gestiona el flujo de trabajo mediante tableros y tarjetas. No usa iteraciones.

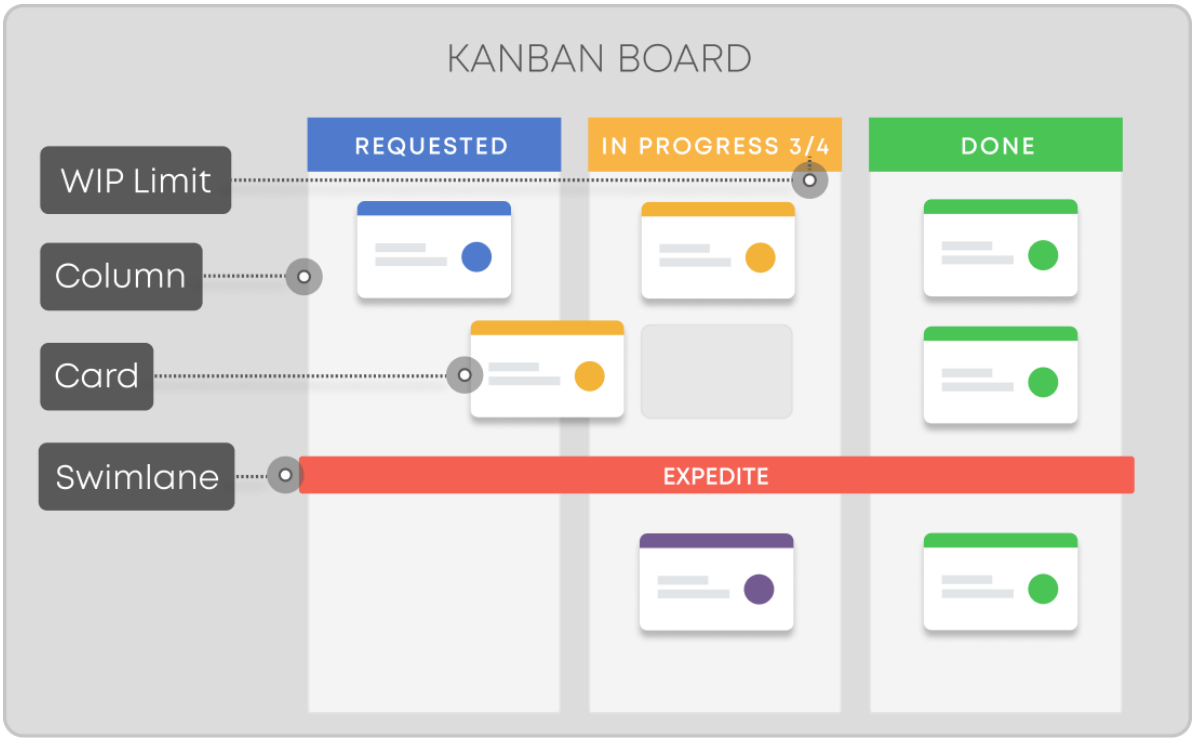
Ventajas:

->Mejora la visibilidad del trabajo.

->Reduce el trabajo en curso.

Desventajas:

->Puede carecer de estructura si no se controla bien.




## 3. Extreme Programming (XP)

(Kent Beck, 1999)

Explicación:

Enfocado en mejorar la calidad del software y la capacidad de respuesta ante requisitos cambiantes.

Prácticas:

->Programación en pareja.

->Desarrollo dirigido por pruebas (TDD).

->Integración continua.

->Simplicidad en el diseño.

Ventajas:

->Código más limpio y testeado.

->Mayor comunicación entre desarrolladores.

## 4. Lean Software Development

(Basado en principios de Toyota) Desarrollo del software en línea.

Principios:
->Eliminar desperdicios.
->Ampliar el aprendizaje.
->Empoderar al equipo.

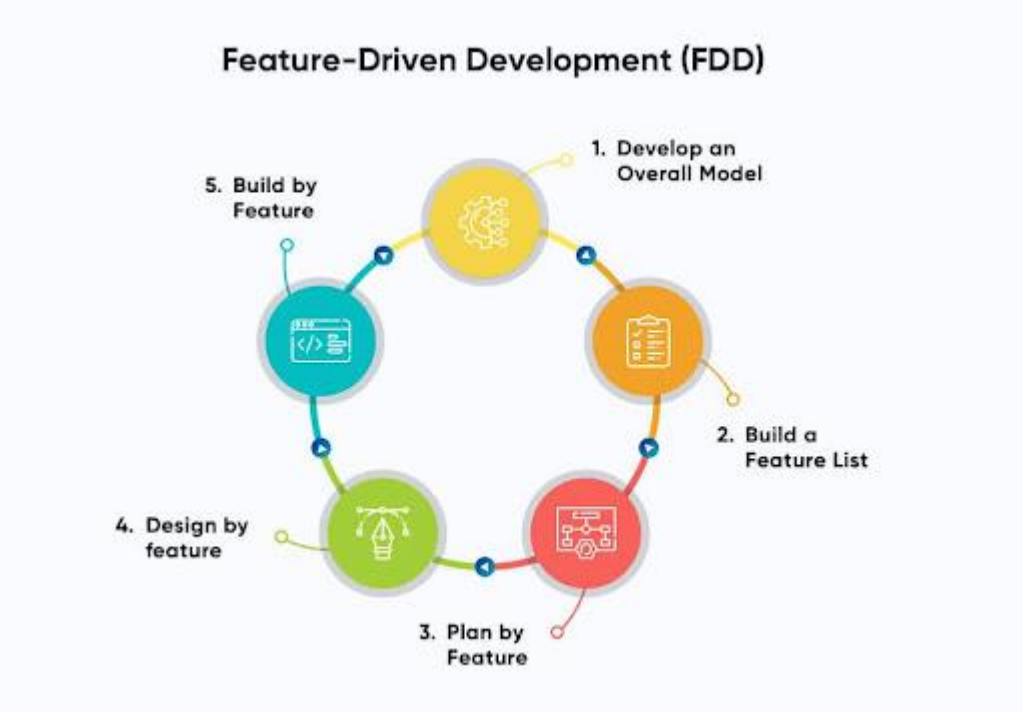
Ventajas:
->Reduce costos y tiempos.
->Enfocado en valor para el cliente.

## 5. Feature-Driven Development (FDD)

Explicación:
Desarrollo orientado a funcionalidades específicas y medibles.

Fases:
->Desarrollo del modelo general.
->Lista de funcionalidades.
->Planificación por funcionalidad.
->Diseño.
->Construcción por funcionalidad.

Ventajas:
->Buena para proyectos grandes.
->Orientada al cliente.



## 🔀 ENFOQUES HÍBRIDOS Y MODERNOS

### 1. DevOps

Explicación:

Integra desarrollo (Dev) y operaciones (Ops) para entregar software más rápido y confiable.

Prácticas:

->Automatización de CI/CD (Integración y entrega continua).

->Monitorización y retroalimentación constante.

Ventajas:

->Mejora la calidad y la velocidad de entrega.

->Fomenta la colaboración interdepartamental.

### 2. Modelo Ágil Escalado (SAFe, LeSS, Nexus)

Explicación:

Extiende principios ágiles a equipos grandes y organizaciones enteras.

Ventajas:

->Aplica ágil a escala empresarial.

->Mejora la coordinación entre equipos.

## 📄 Resumen en Tabla

| Metodología | Tipo | Iterativo | Flexible | Enfoque |
|---|---|---|---|---|
| Cascada | Tradicional | No | No | Secuencial |
| V-Model | Tradicional | No | No | Verificación |
| Incremental | Tradicional | Parcial | Medio | Modular |
| Espiral | Tradicional | Sí | Sí | Riesgos |
| Scrum | Ágil | Sí | Sí | Iterativo |
| Kanban | Ágil | Fluido | Sí | Flujo |
| XP | Ágil | Sí | Sí | Calidad |
| Lean | Ágil | Sí | Sí | Valor |
| FDD | Ágil | Sí | Medio | Funcionalidad |
| DevOps | Híbrido | Sí | Sí | Automatización |
| SAFe/LeSS/Nexus | Ágil Escalado | Sí | Sí | Coordinación |

**Bibliografía**

Pressman, R. S., & Maxim, B. R. *Engenharia de Software Moderna*. McGraw-Hill, 2020. https://www.javier8a.com/itc/bd1/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Sommerville, I. *Engenharia de Software: Uma Abordagem Profissional*. Pearson Education do Brasil, 2011. https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf

Pressman, R. S., & Maxim, B. R. (2014). Ingeniería del software: Un enfoque práctico (8.ª ed.). McGraw-Hill. https://www.academia.edu/44082567/INGENIERÍA_DEL_SOFTWARE_UN_ENFOQUE_PRÁCTICO#loswp-work-container