**Fundamentos de Ingeniería de Software: Análisis,**

**Metodologías y Ciclo de Vida**

Trabajo Práctico en clase - Modelos y Sistemas

Escuela Técnica N°1 de Muñiz

7° 3º Año de Secundaria

Profesor: Omar Alejandro Bazar

Alumno: Nicolás Sosa 7°3

# Tema 1: Análisis funcional y requerimiento del software.

1. Análisis funcional:

En el desarrollo de software, el análisis funcional es el proceso de definir y documentar las funcionalidades y comportamientos que debe tener un sistema para cumplir con los objetivos del cliente. Es importante porque permite entender qué debe hacer el sistema, ayudando a reducir errores en fases posteriores.

2. Requerimientos funcionales y no funcionales:

Los requerimientos funcionales describen qué debe hacer el sistema (por ejemplo, “el sistema debe permitir al usuario iniciar sesión con su correo electrónico y contraseña”).

Los requerimientos no funcionales se refieren a la calidad del sistema o restricciones (por ejemplo, “la respuesta al inicio de sesión no debe tardar más de 2 segundos”).

3. Proceso de elicitación de requerimientos:

Consiste en obtener los requerimientos del sistema a través de la interacción con los stakeholders (partes interesadas). Las técnicas comunes incluyen entrevistas, encuestas, observación, y talleres, entre otros, para entender las necesidades y expectativas de los usuarios.

4. Caso de uso:

Un caso de uso describe una interacción entre un usuario y el sistema para lograr un objetivo específico. Se usa en el análisis funcional para identificar y definir funcionalidades. Ejemplo: “Un usuario busca un producto en una tienda en línea, lo añade al carrito y procede a la compra”.

5. Priorización de requerimientos:

Es fundamental para asegurar que el equipo de desarrollo enfoque sus esfuerzos en las funcionalidades de mayor valor. Factores clave incluyen el impacto en el negocio, la viabilidad técnica, el costo y la urgencia de cada requerimiento.

# Tema 2: Metodologías Ágiles

6. Metodologías ágiles::

Las metodologías ágiles son un enfoque de desarrollo de software basado en iteraciones cortas, colaboración constante entre equipos y clientes, y adaptación a los cambios. A diferencia de las metodologías tradicionales como el modelo en cascada, donde todo el proyecto se planifica y documenta antes de comenzar, las metodologías ágiles priorizan entregas frecuentes y ajustables.

7. Manifiesto ágil:

* + **Individuos e interacciones sobre procesos y herramientas:** La colaboración directa entre personas es más importante que depender exclusivamente de herramientas o procesos.
  + **Software funcional sobre documentación extensa:** Se da prioridad a entregar software que funcione, en lugar de crear demasiada documentación.
  + **Colaboración con el cliente sobre negociación de contratos:** Es preferible trabajar en equipo con los clientes y ajustar según sus necesidades en lugar de ceñirse estrictamente a un contrato.
  + **Respuesta ante el cambio sobre seguir un plan:** En lugar de seguir un plan rígido, las metodologías ágiles aceptan y se adaptan al cambio durante el desarrollo.

8. Scrum y elementos principales:

Scrum es un marco ágil utilizado para gestionar proyectos, basado en iteraciones llamadas "Sprints".

* + **Sprint:** Ciclo de trabajo corto (generalmente 2-4 semanas) en el que se desarrolla un conjunto de tareas específicas.
  + **Product Backlog:** Lista priorizada de tareas o características que el equipo debe implementar.
  + **Scrum Master:** Persona encargada de facilitar el trabajo del equipo, eliminando obstáculos y asegurando que se sigan las prácticas de Scrum.

9. Sprint:

Un Sprint es un período de tiempo fijo en el que un equipo trabaja para completar un conjunto de tareas previamente planificadas. Su propósito es proporcionar incrementos pequeños pero funcionales de un producto, permitiendo revisiones y ajustes continuos.

10. “Historia de Usuario”:

Una historia de usuario es una descripción breve y simple de una funcionalidad desde la perspectiva del usuario final. Se utiliza para identificar qué necesita el usuario, generalmente escrita en formato: "Como [tipo de usuario], quiero [función] para [objetivo]." Facilita la priorización y el diseño de soluciones.

# Tema 3: Ciclo de Vida del Software

11.Ciclo de vida del software:

El ciclo de vida del software es el proceso estructurado que sigue un proyecto de software desde su inicio hasta su finalización o mantenimiento.

* + **Fases principales:**
    1. **Recolección y análisis de requisitos:** Identificar lo que el software debe hacer.
    2. **Diseño:** Crear una arquitectura y especificaciones técnicas.
    3. **Implementación:** Programar el software.
    4. **Pruebas:** Verificar que el software cumpla con los requisitos y no tenga errores.
    5. **Implementación:** Desplegar el software en un entorno real.
    6. **Mantenimiento:** Realizar actualizaciones y resolver problemas.

12. Diferencia entre el modelo en cascada y el modelo iterativo de desarrollo de software:

* + **Modelo en cascada:** Es un enfoque secuencial donde cada fase (requisitos, diseño, implementación, pruebas) debe completarse completamente antes de pasar a la siguiente. Es rígido y no permite cambios durante el proceso.
  + **Modelo iterativo:** Divide el desarrollo en ciclos repetitivos (iteraciones), donde cada iteración resulta en un incremento funcional del producto. Es más flexible, permitiendo ajustes basados en retroalimentación constante.

13. Fases de pruebas:

La fase de pruebas consiste en verificar y validar que el software desarrollado cumple con los requisitos especificados y no contiene errores significativos. Incluye pruebas funcionales, de integración, de rendimiento y de usuario.  
Es crucial porque asegura la calidad del software, detecta y corrige errores antes del despliegue y garantiza que el producto funcione de manera confiable en entornos reales, lo que reduce costos y riesgos asociados a fallos.

14. Mantenimiento del Software:

El mantenimiento del software implica actualizar, mejorar y corregir el sistema después de su implementación. Puede incluir la corrección de errores, adaptación a nuevos entornos o mejoras de funcionalidad.  
Es importante porque el software debe evolucionar para seguir siendo útil y relevante, adaptarse a los cambios en los requisitos de los usuarios o del entorno, y garantizar la continuidad del negocio.

15. "Fase de análisis de requisitos" :

La fase de análisis de requisitos consiste en recopilar, documentar y analizar las necesidades de los stakeholders (usuarios y clientes) para definir qué debe hacer el software.  
Es fundamental porque una comprensión clara y completa de los requisitos asegura que el producto final satisfaga las expectativas del cliente, reduce riesgos de malentendidos y evita retrasos y sobrecostos en fases posteriores.