Trabajo Práctico en clase - Modelos y Sistemas

Escuela Técnica N°1 de Muñiz

7° 1º Año de Secundaria

Profesor: Omar Alejandro Bazar

alumno: Pedro Caballero

Fundamentos de Ingeniería de Software: Análisis, Metodologías y Ciclo de Vida

### **Tema 1: Análisis Funcional y Requerimientos del Software**

#### **1. Explica qué es el análisis funcional en el contexto del desarrollo de software y por qué es importante.**

El **análisis funcional** es el proceso de comprender y definir cómo debe comportarse un sistema de software para satisfacer las necesidades de los usuarios. Se centra en las funciones que el software debe realizar, identificando los requisitos, procesos y flujos que el sistema debe soportar. Es importante porque:

* **Asegura que el software cumpla con las expectativas del usuario.**
* **Sirve como base para el diseño y desarrollo del sistema.**
* **Ayuda a identificar posibles riesgos o limitaciones desde el inicio.**

#### **2. Define qué son los requerimientos funcionales y no funcionales. Proporciona un ejemplo de cada uno.**

* **Requerimientos funcionales:** Son las características o funciones que el sistema debe realizar. Se refieren a lo que el sistema debe hacer.
  + **Ejemplo:** "El sistema debe permitir que los usuarios se registren con su nombre, correo electrónico y contraseña."
* **Requerimientos no funcionales:** Son características relacionadas con el rendimiento, usabilidad, seguridad y otros aspectos del sistema que no están directamente relacionados con la funcionalidad.
  + **Ejemplo:** "El sistema debe ser capaz de procesar 1000 transacciones por segundo."

#### **3. Describe el proceso de elicitación de requerimientos. ¿Qué técnicas se pueden utilizar para obtener información de los stakeholders?**

La **elicitación de requerimientos** es el proceso de recopilar información de los stakeholders (clientes, usuarios, expertos) para identificar sus necesidades y expectativas con respecto al sistema. Técnicas comunes de elicitación:

* **Entrevistas:** Conversaciones directas con los stakeholders para entender sus necesidades.
* **Cuestionarios y encuestas:** Herramientas estructuradas para obtener información de forma más amplia.
* **Observación:** Estudio de cómo los usuarios interactúan con sistemas existentes.
* **Talleres o focus groups:** Reuniones de trabajo con múltiples stakeholders para obtener un consenso.
* **Prototipos:** Creación de versiones tempranas del software para obtener feedback directo.

#### **4. ¿Qué es un caso de uso y cómo se utiliza en el análisis funcional? Proporciona un ejemplo sencillo.**

Un **caso de uso** es una descripción detallada de una funcionalidad del sistema desde la perspectiva del usuario. Describe una secuencia de pasos que ilustran cómo un actor (usuario o sistema) interactúa con el software para alcanzar un objetivo específico. En el análisis funcional, los casos de uso ayudan a:

* Definir los requisitos del sistema de manera clara.
* Servir como base para las pruebas y validación del software.
* Asegurar que todas las funcionalidades necesarias estén contempladas.

**Ejemplo de caso de uso:** **Caso de uso:** **Inicio de sesión de usuario**

* **Actor:** Usuario registrado.
* **Objetivo:** El usuario desea iniciar sesión en el sistema.
* **Pasos:**
  1. El usuario ingresa su nombre de usuario y contraseña.
  2. El sistema valida las credenciales.
  3. Si las credenciales son correctas, el sistema muestra la página de inicio. Si son incorrectas, muestra un mensaje de error.

#### **5. Explica la importancia de la priorización de requerimientos en un proyecto de software. ¿Qué factores se deben considerar al priorizar?**

La **priorización de requerimientos** es crucial porque ayuda a enfocar los esfuerzos del equipo en las funcionalidades más importantes y urgentes, maximizando el valor entregado al cliente. Además, ayuda a gestionar el tiempo y los recursos de manera eficiente. Factores a considerar al priorizar:

* **Valor para el usuario:** Qué tan importante es para los usuarios finales.
* **Riesgos:** Qué tan crítico es implementar un requerimiento para el éxito del proyecto.
* **Complejidad y esfuerzo:** Qué tanto esfuerzo y recursos se requieren para implementar el requerimiento.
* **Dependencias:** Qué funcionalidades deben completarse antes de poder implementar otras.

### **Tema 2: Metodologías Ágiles**

#### **6. Define qué son las metodologías ágiles y cómo se diferencian de las metodologías tradicionales de desarrollo de software.**

Las **metodologías ágiles** son un conjunto de enfoques de desarrollo de software que priorizan la colaboración, la flexibilidad, la entrega continua de valor, y la adaptación a los cambios durante el ciclo de vida del proyecto. A diferencia de las metodologías tradicionales (como el modelo en cascada), las ágiles son más iterativas y permiten entregas parciales y frecuentes, lo que facilita la incorporación de cambios en los requisitos y mejora la satisfacción del cliente. Diferencias clave:

* **Iteración continua:** Las metodologías ágiles permiten ajustar el producto en función del feedback recibido en cada iteración.
* **Adaptación a cambios:** Son más flexibles frente a cambios en los requisitos, mientras que las tradicionales siguen un enfoque secuencial rígido.

#### **7. Enumera y describe brevemente los cuatro valores fundamentales del Manifiesto Ágil.**

Los cuatro valores fundamentales del **Manifiesto Ágil** son:

1. **Individuos e interacciones sobre procesos y herramientas:** Valorar más la colaboración directa entre los miembros del equipo que las herramientas y los procesos rígidos.
2. **Software funcionando sobre documentación extensiva:** Priorizar la entrega de software funcional en lugar de dedicar mucho tiempo a la documentación.
3. **Colaboración con el cliente sobre negociación de contratos:** Buscar una relación constante y activa con el cliente para adaptar el producto a sus necesidades, en lugar de seguir estrictamente los términos de un contrato.
4. **Respuesta ante el cambio sobre seguir un plan:** Ser capaz de adaptarse rápidamente a cambios en los requisitos o en el mercado, en lugar de seguir un plan predefinido.

#### **8. Explica qué es Scrum y describe tres de sus elementos principales (por ejemplo, Sprint, Product Backlog, Scrum Master).**

**Scrum** es una metodología ágil que se basa en ciclos de trabajo iterativos y un conjunto de roles, eventos y artefactos para gestionar el desarrollo de software. Tres elementos principales de Scrum:

* **Sprint:** Un ciclo de trabajo de corta duración (generalmente de 1 a 4 semanas) donde se completan tareas del Product Backlog.
* **Product Backlog:** Es una lista priorizada de todas las funcionalidades, mejoras y correcciones que deben implementarse en el producto. Es administrado por el Product Owner.
* **Scrum Master:** Es el facilitador del equipo Scrum, se asegura de que se sigan las prácticas ágiles, elimina obstáculos que pueda enfrentar el equipo y fomenta la colaboración.

#### **9. ¿Qué es un "sprint" en el contexto de las metodologías ágiles y cuál es su propósito?**

Un **sprint** es un ciclo de trabajo corto y bien definido dentro de la metodología Scrum. Durante un sprint, el equipo trabaja en una parte del Product Backlog que se ha priorizado para ese período, y al final del sprint, debe haber entregado un incremento funcional del producto. El propósito del sprint es:

* **Generar entregas parciales y funcionales de valor en cada ciclo.**
* **Permitir la retroalimentación temprana del cliente y de los usuarios.**
* **Facilitar la adaptación continua del producto a cambios y necesidades emergentes.**

#### **10. Describe qué es una "historia de usuario" y cómo se utiliza en el desarrollo ágil de software.**

Una **historia de usuario** es una pequeña descripción de una funcionalidad del sistema escrita desde la perspectiva de un usuario final. Sigue un formato sencillo:

* **Como [tipo de usuario], quiero [funcionalidad] para [objetivo].** Las historias de usuario se utilizan para:
* **Definir requisitos de manera clara y comprensible.**
* **Planificar el trabajo de desarrollo en iteraciones.**
* **Enfocar el desarrollo en el valor que entregará al usuario.**

**Ejemplo de historia de usuario:** "Como usuario registrado, quiero poder restablecer mi contraseña para recuperar el acceso a mi cuenta si olvido mis credenciales."

### 

### **Tema 3: Ciclo de Vida del Software**

#### **11. Define qué es el ciclo de vida del software y enumera sus fases principales.**

El **ciclo de vida del software** es el conjunto de etapas que atraviesa el software desde su concepción hasta su retiro. Asegura que el desarrollo sea estructurado, controlado y eficiente. Fases principales del ciclo de vida:

1. **Planificación:** Se definen los objetivos y el alcance del proyecto.
2. **Análisis de requisitos:** Se recopilan y analizan los requisitos del sistema.
3. **Diseño:** Se define la arquitectura y el diseño detallado del sistema.
4. **Desarrollo:** Se lleva a cabo la programación del sistema.
5. **Pruebas:** Se valida que el sistema cumpla con los requisitos.
6. **Despliegue:** Se entrega el sistema a los usuarios finales.
7. **Mantenimiento:** Se realizan actualizaciones y correcciones tras la entrega.

#### 

#### **12. Explica la diferencia entre el modelo en cascada y el modelo iterativo de desarrollo de software.**

* **Modelo en cascada:** Es un modelo secuencial en el que cada fase del ciclo de vida del software debe completarse antes de pasar a la siguiente. No permite revisiones o cambios fácilmente.
* **Modelo iterativo:** En lugar de ser secuencial, el desarrollo ocurre en ciclos