Contents

Clase 5: Métricas de Calidad del Software

1. Objetivos de la Clase:

- Comprender la importancia de las métricas para evaluar la calidad del software.
- Identificar diferentes tipos de métricas de calidad del software.
- Aprender a seleccionar y aplicar métricas apropiadas para diferentes contextos.
- Analizar la relación entre las métricas y los atributos de calidad del software.
- Interpretar los resultados de las métricas y utilizarlos para la mejora continua.

2. Contenido Teórico Detallado:

2.1. Introducción a las Métricas de Calidad del Software:

Las métricas de calidad del software son mediciones cuantitativas que permiten evaluar y monitorear diferentes aspectos de la calidad del software. Estas métricas proporcionan información objetiva sobre el estado del software, identifican áreas de mejora y ayudan a tomar decisiones informadas durante el ciclo de vida del desarrollo. Son una herramienta fundamental para garantizar que el software cumple con los requisitos y las expectativas del cliente.

2.2. Tipos de Métricas de Calidad:

Existen diversos tipos de métricas de calidad, que se pueden clasificar según el atributo de calidad que miden o la fase del ciclo de vida en la que se aplican:

- **Métricas de Producto:** Evalúan las características del software en sí, como:
  - **Líneas de Código (LOC):** Mide el tamaño del código fuente. Aunque simple, puede indicar complejidad.
  - **Complejidad Ciclomática (McCabe):** Mide la complejidad del flujo de control dentro de un módulo. Un valor alto sugiere un código más difícil de entender y probar.
  - **Densidad de Defectos:** Número de defectos encontrados por cada mil líneas de código (KLOC). Un valor alto indica una calidad deficiente.
  - **Cobertura de Pruebas:** Porcentaje del código cubierto por las pruebas unitarias. Un mayor porcentaje indica una mayor probabilidad de detectar errores.
- **Métricas de Proceso:** Evalúan la eficiencia y efectividad de los procesos de desarrollo, como:
  - **Tiempo de Ciclo:** Tiempo requerido para completar una iteración o fase del desarrollo.
  - **Esfuerzo:** Cantidad de trabajo invertido en una tarea o proyecto.
  - **Velocidad (Velocity):** En metodologías ágiles, mide la cantidad de trabajo completada en una iteración.
  - **Número de Defectos Encontrados Durante la Pruebas:** Indica la efectividad de las actividades de prueba.
- **Métricas de Proyecto:** Evalúan el progreso y el rendimiento del proyecto en general, como:
  - **Variación del Cronograma:** Diferencia entre el tiempo planificado y el tiempo real consumido.
  - **Variación del Presupuesto:** Diferencia entre el presupuesto planificado y el costo real.
  - **Satisfacción del Cliente:** Medida de la satisfacción del cliente con el producto o servicio.

2.3. Selección de Métricas Apropiadas:

La selección de métricas apropiadas depende del contexto del proyecto, los objetivos de calidad y los atributos de calidad que se desean medir. Es importante considerar lo siguiente:

- **Relevancia:** La métrica debe ser relevante para los objetivos de calidad.
- **Medibilidad:** La métrica debe ser fácil de medir y cuantificar.
- **Interpretación:** Los resultados de la métrica deben ser fáciles de interpretar y comprender.
- **Costo:** El costo de recopilar y analizar la métrica debe ser razonable.
- **Accionabilidad:** Los resultados de la métrica deben permitir tomar acciones correctivas y mejorar la calidad.

2.4. Relación entre Métricas y Atributos de Calidad:

Cada métrica está relacionada con uno o varios atributos de calidad del software. Por ejemplo:

- La complejidad ciclomática está relacionada con la mantenibilidad y la probabilidad de errores.
- La densidad de defectos está relacionada con la confiabilidad y la calidad general.
- La cobertura de pruebas está relacionada con la confiabilidad y la detección temprana de errores.
- El tiempo de ciclo está relacionado con la eficiencia del proceso de desarrollo.

2.5. Interpretación y Uso de los Resultados:

La interpretación de los resultados de las métricas debe realizarse con cuidado y en el contexto del proyecto. Es importante establecer umbrales o rangos de referencia para determinar si los valores de las métricas son aceptables o requieren acciones correctivas.

Los resultados de las métricas se pueden utilizar para:

- Identificar áreas de mejora en el código, los procesos o el proyecto.
- Monitorear el progreso de la calidad a lo largo del tiempo.
- Tomar decisiones informadas sobre el diseño, la implementación y las pruebas.
- Comunicar el estado de la calidad a las partes interesadas.
- Mejorar continuamente la calidad del software.

3. Ejemplos y Casos de Estudio:

- **Caso de Estudio 1: Proyecto de Desarrollo de una Aplicación Web:** Se utilizan métricas como líneas de código, complejidad ciclomática y densidad de defectos para monitorear la calidad del código. Se observa un aumento en la complejidad ciclomática en un módulo específico, lo que indica la necesidad de refactorización.
- **Caso de Estudio 2: Proyecto de Mantenimiento de un Sistema Legado:** Se utilizan métricas como el número de defectos reportados por los usuarios y el tiempo de resolución de los defectos para evaluar la mantenibilidad del sistema. Un aumento en el número de defectos y el tiempo de resolución indica la necesidad de mejorar la documentación y la capacitación del equipo de mantenimiento.
- **Ejemplo:** En un proyecto ágil, la velocidad del equipo (cuántas historias de usuario completan por sprint) se utiliza como una métrica de proceso para planificar sprints futuros y mejorar la eficiencia del equipo. Si la velocidad disminuye, el equipo analiza las posibles causas (cuellos de botella, falta de habilidades, etc.) y toma medidas para solucionarlas.

4. Problemas Prácticos y Ejercicios:

1. **Ejercicio 1:** Calcula la complejidad ciclomática de la siguiente función:

   ```
   python def calcular_descuento(precio, es_cliente_premium):     if es_cliente_premium:
   if precio > 100:             descuento = 0.20         else:             descuento =
   0.10     else:         descuento = 0.05     return precio * (1 - descuento)
   ```

   Solución: La complejidad ciclomática es 4. Hay tres ”if” que aumentan la complejidad, más el camino base.

2. **Ejercicio 2:** Un proyecto tiene 5000 líneas de código y se han encontrado 25 defectos durante las pruebas. Calcula la densidad de defectos.

   Solución: La densidad de defectos es 25 / 5 = 5 defectos por KLOC.

3. **Ejercicio 3:** Investiga diferentes herramientas para medir la complejidad ciclomática en código Java o Python.

   Solución: Algunas herramientas populares son SonarQube, PMD, y Radon.

4. **Ejercicio 4:** En un proyecto, la cobertura de pruebas unitarias es del 60%. ¿Qué implicaciones tiene esto para la calidad del software? ¿Qué acciones se pueden tomar?

Solución: Una cobertura del 60% significa que el 40% del código no está siendo probado por las pruebas unitarias. Esto implica un mayor riesgo de que existan defectos no detectados en ese código. Se pueden tomar las siguientes acciones:

- **Analizar el código no cubierto:** Identificar las razones por las que el código no está cubierto por las pruebas unitarias.
- **Escribir nuevas pruebas unitarias:** Desarrollar pruebas unitarias adicionales para cubrir el código no cubierto.
- **Refactorizar el código:** Simplificar el código para facilitar la escritura de pruebas unitarias.

5. Materiales Complementarios Recomendados:

- **Libros:**
  - "Software Metrics: A Rigorous and Practical Approach" by Norman Fenton and Shari Lawrence Pfleeger.
  - "Metrics and Models in Software Quality Engineering" by Stephen H. Kan.
- **Artículos:**
  - Buscar artículos académicos sobre métricas de calidad del software en bases de datos como IEEE Xplore o ACM Digital Library.
- **Herramientas:**
  - SonarQube (análisis estático de código)
  - JaCoCo (cobertura de código Java)
  - Coverage.py (cobertura de código Python)

Esta clase proporciona una base sólida para comprender y aplicar las métricas de calidad del software. En las clases siguientes, exploraremos cómo utilizar estas métricas en diferentes fases del ciclo de vida del desarrollo y cómo integrarlas en un proceso de mejora continua de la calidad.