Contents

Clase 11: Métricas de Calidad del Software

1. Objetivos de la Clase:

- Comprender la importancia de las métricas de calidad en el desarrollo de software.
- Identificar y definir las principales métricas de calidad del software.
- Aplicar métricas de calidad para evaluar y mejorar la calidad del software.
- Interpretar los resultados de las métricas para tomar decisiones informadas.

2. Contenido Teórico Detallado:

Las métricas de calidad del software son medidas cuantitativas que permiten evaluar y monitorear las características de calidad de un producto de software o de su proceso de desarrollo. Estas métricas proporcionan información valiosa para la toma de decisiones, la identificación de áreas de mejora y la verificación del cumplimiento de los objetivos de calidad.

2.1. Importancia de las Métricas de Calidad:

- **Medición Objetiva:** Proporcionan una base objetiva para evaluar la calidad, en lugar de depender de opiniones subjetivas.
- **Identificación de Problemas:** Permiten identificar áreas problemáticas en el software o en el proceso de desarrollo que requieren atención.
- **Seguimiento del Progreso:** Facilitan el seguimiento del progreso en la mejora de la calidad a lo largo del tiempo.
- **Toma de Decisiones Informada:** Ayudan a tomar decisiones informadas sobre la asignación de recursos, la priorización de tareas y la implementación de cambios.
- **Comunicación:** Facilitan la comunicación clara y concisa sobre la calidad del software a todas las partes interesadas.

2.2. Clasificación de las Métricas de Calidad:

Las métricas de calidad se pueden clasificar en varias categorías, según el aspecto de la calidad que miden:

- **Métricas de Producto:** Miden las características del producto de software en sí mismo.
  - **Tamaño:** Líneas de código (LOC), puntos de función.
  - **Complejidad:** Complejidad ciclomática de McCabe, complejidad de Halstead.
  - **Acoplamiento:** Acoplamiento entre clases (CBO).
  - **Cohesión:** Cohesión entre métodos de una clase (LCOM).
  - **Defectos:** Densidad de defectos (defectos por KLOC), tasa de defectos.
- **Métricas de Proceso:** Miden las características del proceso de desarrollo de software.
  - **Esfuerzo:** Horas-persona gastadas en el proyecto.
  - **Tiempo:** Tiempo de ciclo de desarrollo, tiempo de respuesta a los cambios.
  - **Productividad:** Puntos de función por hora-persona.
  - **Eficiencia:** Costo por punto de función.
  - **Estabilidad del proceso:** Variación en el esfuerzo y el tiempo de ciclo.
- **Métricas de Recursos:** Miden la utilización y eficiencia de los recursos utilizados en el desarrollo.
  - **Utilización de recursos:** Uso de la CPU, memoria, espacio en disco durante las pruebas.
  - **Eficiencia de recursos:** Defectos encontrados por hora de prueba.
  - **Costo de recursos:** Costo por hora de prueba.
- **Métricas de Satisfacción del Cliente:** Miden la percepción del cliente sobre la calidad del software.
  - **Encuestas de satisfacción:** Puntuación de satisfacción del cliente (CSAT), Net Promoter Score (NPS).
  - **Quejas de clientes:** Número de quejas recibidas.

- **Solicitudes de cambio:** Número de solicitudes de cambio iniciadas por el cliente.

2.3. Ejemplos de Métricas Específicas:

- **Densidad de Defectos:** Número de defectos encontrados por cada mil líneas de código (KLOC). Una densidad alta puede indicar problemas de calidad en el código.
- **Complejidad Ciclomática:** Medida de la complejidad del flujo de control en un programa. Un valor alto puede indicar código difícil de entender y mantener, y propenso a errores.
- **Acoplamiento entre Clases (CBO):** Número de clases a las que una clase está acoplada. Un alto acoplamiento puede indicar una falta de modularidad y dificultar la reutilización y el mantenimiento del código.
- **Cohesión entre Métodos de una Clase (LCOM):** Medida de la relación entre los métodos de una clase. Una baja cohesión puede indicar que la clase tiene demasiadas responsabilidades y debería dividirse en clases más pequeñas.
- **Tiempo de Respuesta:** Tiempo que tarda el sistema en responder a una solicitud del usuario. Un tiempo de respuesta lento puede afectar la usabilidad y la satisfacción del usuario.
- **Puntuación de Satisfacción del Cliente (CSAT):** Medida de la satisfacción del cliente con el producto o servicio.

2.4. Proceso de Definición y Aplicación de Métricas:

1. **Definir Objetivos:** Establecer claramente los objetivos de calidad que se quieren alcanzar.
2. **Seleccionar Métricas:** Elegir las métricas que mejor se adapten a los objetivos definidos. Considerar el contexto del proyecto y los recursos disponibles.
3. **Definir Procedimientos de Medición:** Establecer cómo se van a recolectar y calcular las métricas.
4. **Recolectar Datos:** Recolectar los datos necesarios para calcular las métricas.
5. **Analizar Datos:** Analizar los resultados de las métricas para identificar tendencias, patrones y áreas de mejora.
6. **Tomar Acciones:** Tomar acciones basadas en los resultados del análisis para mejorar la calidad del software.
7. **Monitorear Resultados:** Monitorear los resultados de las acciones tomadas para verificar su efectividad.
8. **Ajustar Métricas:** Ajustar las métricas según sea necesario para asegurar que sigan siendo relevantes y útiles.

3. Ejemplos o Casos de Estudio:

- **Caso de Estudio 1: Medición de la Densidad de Defectos en un Proyecto de Desarrollo Web:** Un equipo de desarrollo web está trabajando en un nuevo proyecto. Para medir la calidad, deciden monitorizar la Densidad de Defectos (defectos/KLOC) durante las pruebas. Después de la primera fase de pruebas, encuentran 15 defectos en 5000 líneas de código. La densidad de defectos es $15/5 = 3$ defectos/KLOC. Si el umbral establecido por la empresa es de 2 defectos/KLOC, el equipo debe investigar las causas de esta alta densidad de defectos y tomar medidas correctivas (revisión de código, pruebas adicionales) antes de continuar.

- **Caso de Estudio 2: Uso de la Complejidad Ciclomática para Refactorizar Código:** Un desarrollador identifica una función con una complejidad ciclomática de 25. Esto indica que la función es muy compleja y difícil de entender. El desarrollador decide refactorizar la función dividiéndola en funciones más pequeñas y simples. Después de la refactorización, la complejidad ciclomática de cada función es menor a 10. Esto mejora la legibilidad y mantenibilidad del código, y reduce la probabilidad de errores.

- **Caso de Estudio 3: Monitoreo del Tiempo de Respuesta para Mejorar la Experiencia del Usuario:** Una empresa de comercio electrónico monitorea el tiempo de respuesta de su sitio web. Descubren que el tiempo de respuesta promedio es de 5 segundos, lo que está afectando la experiencia del usuario y las ventas. El equipo de desarrollo optimiza el código y la infraestructura del sitio web. Después de las optimizaciones, el tiempo de respuesta promedio se reduce a 2 segundos, lo que mejora la satisfacción del usuario y aumenta las ventas.

4. **Problemas Prácticos o Ejercicios con Soluciones:**

- **Ejercicio 1:** Un proyecto de software tiene 10,000 líneas de código y se han encontrado 50 defectos durante las pruebas. Calcule la densidad de defectos.
  - **Solución:** Densidad de Defectos = 50 / (10,000/1000) = 5 defectos/KLOC
- **Ejercicio 2:** Una función tiene la siguiente estructura de control:

  ```
  Si (condición 1) entonces     Si (condición 2) entonces         // Bloque 1     Sino
  // Bloque 2     Fin Si Sino     // Bloque 3 Fin Si
  ```

  Calcule la complejidad ciclomática de la función. * **Solución:** La complejidad ciclomática se puede calcular como el número de regiones en el grafo de flujo de control, que en este caso es 4 (Bloque 1, Bloque 2, Bloque 3 y la región exterior). También se puede calcular como el número de decisiones (2) + 1 = 3. La complejidad es 4 o 3, dependiendo de la convención. El uso más común es la segunda, donde la complejidad ciclomática es 3.
- **Ejercicio 3:** Una empresa está utilizando la puntuación de satisfacción del cliente (CSAT) para medir la satisfacción del cliente con su producto de software. En una encuesta reciente, la puntuación promedio de CSAT fue de 4 sobre 5. ¿Qué significa esto? ¿Qué acciones debería tomar la empresa?
  - **Solución:** Una puntuación de CSAT de 4 sobre 5 indica que los clientes están generalmente satisfechos con el producto de software. Sin embargo, la empresa debería investigar por qué algunos clientes no están completamente satisfechos (puntuación de 3 o menos) y tomar medidas para mejorar su satisfacción. También, se debe mantener el monitoreo de la métrica.

5. **Materiales Complementarios Recomendados:**

- **Libros:**
  - "Metrics and Models in Software Quality Engineering" de Stephen H. Kan.
  - "Software Metrics: A Rigorous and Practical Approach" de Norman E. Fenton y Shari Lawrence Pfleeger.
- **Artículos:**
  - "A Systematic Review of Software Metrics for Quality Assessment" (Buscar en IEEE Xplore o ACM Digital Library).
- **Sitios Web:**
  - Software Engineering Institute (SEI): https://www.sei.cmu.edu/ (Buscar información sobre métricas de software).
  - IEEE Computer Society: https://www.computer.org/ (Buscar artículos y publicaciones sobre métricas de software).
- **Videos:**
  - Buscar videos en YouTube sobre "Software Metrics" y "Quality Metrics".