

CARRERA DE COMPUTACIÓN

1. Datos Informativos

- 1.1. **Módulo:** 2
- 1.2. **Nivel:** 7mo Nivel
- 1.3. **Apellidos y Nombres:** Samira Narváez, Estiven Méndez, Almer Paguay
- 1.4. **Tema:** Taller Métricas de Software
- 1.5. **Fecha:** 19 de noviembre de 2025

2. Objetivo

Entender, de manera simple, qué son las métricas de software y con qué propósito sirven, saber cuáles son sus categorías fundamentales y aprender a hacer mediciones prácticas que nos permitan determinar si un software es verdaderamente bueno, útil y satisface lo que los usuarios requieren.

3. Contenido

Métricas del Software: Categorías Básicas y Mediciones

Conceptos Fundamentales

- **¿Qué son las métricas?**

Las métricas son para el software lo que un termómetro es para la temperatura.

Las métricas, de la misma manera que un termómetro te indica si tienes fiebre, te indican si un proyecto está en buen estado o si hay algún problema.

- **¿Por qué medir el software?**

- **Realizamos mediciones para:**

- Identificar problemas de manera anticipada, antes de que generen retrasos o costos.
- Aumentar la calidad, ya que lo que se mide tiende a mejorar.
- Tomar decisiones basándose en datos, no en conjeturas
- Determinar si el equipo está trabajando eficientemente.
- Comparar versiones, por ejemplo: ¿Es más rápida la aplicación ahora que antes?

- **Categorías básicas:**

- **Producto (lo que entregamos)**

Se centran en el propio software.

Evalúan aspectos tales como:

- La calidad del código
- Cantidad de errores
- Desempeño (tiempo de reacción)
- Facilidad de uso
- Seguridad

Ejemplo: La métrica indica que existen inconvenientes de rendimiento si una aplicación tarda cinco segundos en abrirse.

- **Proceso (Cómo trabajamos)**

Evalúan el desarrollo del software.

Ayudan a optimizar el desempeño del equipo.

- Incluyen lo siguiente:
- Duración del desarrollo
- Rapidez del equipo (Sprints realizados)
- Porcentaje de fallos detectados por fase
- Precio del proceso

Ejemplo: Si los mismos errores se reportan cada semana, hay un fallo en el proceso.

- **Proyecto (Gestión y avance)**

Contribuyen a determinar si el proyecto se encuentra en buen estado, con retraso o excedido de presupuesto.

Evalúan aspectos tales como:

- Progreso real en comparación con el planificado
- Cumplimiento de plazos
- Presupuesto empleado
- Peligros del proyecto

Ejemplo: La métrica avisa que habrá inconvenientes si el proyecto debía estar al 60% y se encuentra al 30%.

- **Qué es medición.**

La medición es el procedimiento para conseguir un valor o una información acerca de algo con el fin de comprenderlo mejor.

En términos más sencillos:

Medir es asignarle un número a una propiedad para determinar su estado.

Ejemplos para comprenderlo:

- Tomarse la temperatura → saber si hay fiebre.
- Medir el tiempo que tarda en abrir una aplicación → determinar si es rápida o lenta.
- Determinar la cantidad de errores que tiene un programa → saber si es estable o si se requieren mejoras.

Sin medición, simplemente "creemos" que algo está correcto.

Con la medición, sabemos cuán bien está.

Actividad práctica grupal

Actividad: Caso de Estudio

• Caso simulado:

“Un equipo ha desarrollado una app educativa. Deben evaluar cómo va el proyecto. Les entrego los siguientes datos”

Datos

- LOC: 4,800
- Número de módulos: 12
- Tiempo promedio para corregir errores: 30 horas
- Costo planeado: \$5,000 / Costo real: \$5,600
- Bugs reportados: 10 en fase de pruebas

Tarea de cada grupo:

- Identificar 3 métricas: una de producto, una de proceso y una de proyecto.

a) **Métrica de Producto:** Norma ISO/IEC 25010 (Calidad del producto software). Métrica: Densidad de defectos.

$$Densidad = \left| \frac{\text{Nº de defectos encontrados en pruebas}}{\text{Tamaño de Producto}} \times 1000 \right|$$

$$\frac{10}{4800} \times 1000 = 2.08 \text{ defectos/KLOC}$$

Interpretación normativa: Según ISO 25010, esto se relaciona con el atributo de calidad Confiabilidad (subcaracterística: tolerancia a fallos). Un valor > 2 defectos/KLOC en etapa de pruebas puede considerarse alto para software crítico educativo si afecta la funcionalidad.

- b) Métrica de Proceso:** Norma ISO/IEC 12207 / CMMI Nivel 4 (Gestión cuantitativa de procesos)
Métrica: Eficiencia de corrección de defectos.

Tiempo promedio de corrección= 30 horas/error

Interpretación normativa: Según CMMI, se debe establecer un objetivo de rendimiento de proceso (Process Performance Baseline). 30 horas indica posible falta de estandarización en el proceso de corrección (análisis, asignación, solución, verificación).

- c) Métrica de Proyecto:** Norma ISO 21500 (Gestión de proyectos) / PMI
Métrica: Índice de Rendimiento de Costos (CPI)

$$CPI = \left| \frac{\text{Valor Ganado}(EV)}{\text{Costo Real (AC)}} \right|$$

Suponiendo que el valor ganado = costo planeado (se completó el scope planeado):

$$EV = 5000, AC = 5600$$

$$CPI = \frac{5000}{5600} \approx 0.89$$

Interpretación normativa: CPI < 1 indica sobrecostos. Según ISO 21500, se deben tomar acciones correctivas cuando el CPI se desvía significativamente de 1.

Proponer una mejora basada en los datos.

- a) **Mejora de la métrica del producto:** Optimizar el proceso de corrección de errores.

Problema Identificado: 30 horas promedio por error es muy alto.

Acciones:

1. Implementar revisiones de código para detectar bugs temporalmente.
2. Utilizar herramientas de análisis estático como SonarQube para identificar problemas antes de las pruebas.
3. Crear bases de conocimiento documentando errores comunes y sus soluciones.
4. Capacitar al equipo en técnicas de debugging efectivo.

- b) **Mejora de la métrica de proceso:** Control presupuestario y gestión de costos.

Problema identificado: Sobrecosto del 12% (cpi = 0.89).

Acciones:

1. Problemas en el proceso de debugging.
1. Implementar seguimiento semanal de gastos vs. Presupuesto
2. Identificar las causas del sobrecosto (¿son las 30 horas por bug?)
3. Si los bugs están generando el sobrecosto: priorizar calidad en desarrollo para reducir trabajo correctivo
4. Establecer un buffer presupuestario del 10-15% para contingencias

- c) **Mejora para la métrica de proyecto:** Aumentar la cobertura de pruebas

Problema identificado: 10 bugs en fase de pruebas sugieren testing insuficiente.

Acciones

1. Implementar pruebas unitarias con objeto de 80%+ cobertura.
2. Establecer pruebas automatizadas para los 12 módulos
3. Realizar pruebas de integración entre módulos antes de la fase de pruebas formal.

4. Aplicar testing continuo durante el desarrollo (shift-left testing),

Conclusiones

Frase:

"Medir es el primer paso para controlar y mejorar.

Beneficio esperado según normativas:

- Reducción de retrabajo (mejora en CPI en próximos proyectos).
- Cumplimiento de ISO 25010 en confiabilidad.
- Proceso de mantenimiento correctivo más predecible (CMMI Nivel 4)

4. Conclusión

Finalmente, la medición de software no es sólo un trabajo técnico; También es una forma de comprender realmente el progreso del proyecto, si el equipo está trabajando correctamente y si el producto final beneficiará a los usuarios. Las mediciones nos ayudan a dejar de especular y empezar a tomar decisiones utilizando información real. Nos permiten detectar problemas tempranamente, optimizar la calidad del código, eliminar retrasos y garantizar que el trabajo en equipo se refleje verdaderamente en un software rápido, confiable y funcional. Cuando nos entrenamos para medirnos, aprendemos a preocuparnos por el proyecto, protegernos como equipo y entregar algo realmente útil. Porque al final del día, el "buen" software no es sólo software que funciona, es software que hace lo que promete, se mantiene estable y proporciona una experiencia de usuario agradable. El primer paso para la mejora es la medición, y lo que nos permite desarrollar software de calidad con un propósito es precisamente la mejora.

La experiencia del taller confirma el principio básico de que "la medición es el primer paso para controlar y mejorar". Sin métricas, los equipos de desarrollo operan a ciegas y toman decisiones basadas en la intuición, lo que puede resultar costoso. Por el contrario, la medición sistemática proporciona evidencia objetiva que permite tomar decisiones informadas, predice riesgos y optimiza los recursos. El uso de estándares internacionales como ISO/IEC 25010 para la calidad del producto, ISO/IEC 12207 para procesos e ISO 21500 para gestión de proyectos proporciona un marco estandarizado que facilita no sólo la evaluación interna, sino también la comparación con las mejores prácticas de la industria. Después de todo, el éxito de un proyecto de software depende no



Sólo de las habilidades técnicas del equipo, sino también de su capacidad para medir, analizar y actuar sobre los datos que revelan el verdadero estado de desarrollo.

Calle Antisana y Av. Universitaria |
Telf: (06) 2980837 - 2984435
info@upec.edu.ec
www.upec.edu.ec
Tulcán - Ecuador |