UNIVERSIDAD POLITÉCNICA ESTATAL DEL CARCHI

CARRERA DE COMPUTACIÓN

TALLER

1. Datos Informativos

1.1.Módulo: Unidad 4

1.2.Nivel: Séptimo

1.3.Fecha: 05/05/2025

1.4.Tema: Métricas Aplicadas a la Explicabilidad

1.5.Integrantes: Aupas Antony, Baraja Cristian, Basantes Geovanny

2. Objetivos

- Comprender qué son las métricas de productividad en desarrollo de software.
- Identificar métricas útiles y riesgosas.
- Reflexionar sobre el uso ético y estratégico de estas métricas.
- Analizar casos reales y construir criterios de aplicación.

3. Contenido

3.1.Introducción

¿Cómo sabes que eres productivo como desarrollador?

Ser eficaz como programador implica no solo aumentar el número de líneas de código, sino también aportar valor a través de soluciones de alta calidad y efectivas. La eficacia está relacionada con la habilidad de convertir requerimientos en software operante de forma duradera.

¿Por qué medir? ¿Para qué?

Cuantificar la eficacia permite optimizar métodos, detectar puntos problemáticos, hacer elecciones fundamentadas y alinear las metas personales con las de la organización. No obstante, también conlleva riesgos si los datos se interpretan erróneamente o se emplean para controlar en vez de mejorar. Según lo mencionado por Forsgren et al. (2018), las métricas deben funcionar como un recurso para el avance continuo, no como herramientas de supervisión.

3.2.Presentación Teórica

3.2.1. Definición de métricas de productividad

Las métricas de productividad en el ámbito del desarrollo de software son parámetros usados para medir el desempeño, la calidad y la eficacia de los grupos o programadores individuales. Estas métricas pueden fundamentarse en datos cuantitativos (como el número de líneas de código) o cualitativos (como el valor aportado al usuario).

3.2.2. Tipos de métricas

A continuación, se presentan las métricas más comunes:

- **LoC (Líneas de Código):** evalúa la cantidad de código producido, aunque puede ser engañoso si se confunde cantidad con calidad.
- **Commits:** total de envíos al repositorio, útil cuando se analiza en su contexto y contenido.
- **Bugs reportados/resueltos:** indica la calidad del software, especialmente si se tiene en cuenta su gravedad.
- **Velocidad:** utilizada en enfoques ágiles, mide cuántas historias de usuario se finalizan en un sprint.

Como menciona Smith (2020), ninguna métrica determina por sí sola la productividad real; su valor está en la interpretación conjunta y con contexto.

3.2.3. Métricas útiles vs. métricas engañosas

Las métricas beneficiosas son aquellas que facilitan la mejora, mientras que las engañosas pueden ocasionar incentivos erróneos o crear presión innecesaria. Un ejemplo es el medir solo líneas de código, lo que podría motivar a producir código superfluo.

De acuerdo con Kerzner (2017), las métricas efectivas deben ser comprensibles, accionables y estar alineadas con los objetivos estratégicos del equipo.

3.2.4. Buenas prácticas

La incorporación de indicadores tanto numéricos como descriptivos nos ofrece una perspectiva más integral del desempeño de los programadores, puesto que no todo puede expresarse en cifras. Es crucial que estas métricas se utilicen como un recurso para el avance, en lugar de como métodos para sancionar o ejercer presión. Asimismo, cada resultado debe ser analizado en su entorno, tomando en cuenta aspectos como la dificultad del proyecto o los retos superados. Fomentar la claridad y la interpretación compartida de la información permite que cada integrante del equipo se sienta incluido en el proceso y promueve un clima de mejora continua en lugar de rivalidad excesiva.

3.2.5. Herramientas disponibles

Hay varias herramientas que permiten registrar y analizar las métricas de productividad de un desarrolador y se describen a continuación:

- **GitHub/GitLab:** Para commits, problemas y solicitudes de extracción.

- **Jira:** Para medir velocidad y seguimiento de tareas.
- **SonarQube:** Para evaluar la calidad del código.
- **CodeClimate:** Para análisis de mantenibilidad y cobertura de pruebas.

3.2.6. Reflexión: ¿Qué métricas ha usado o te han aplicado en prácticas, trabajo o clases?

En el ámbito educativo, he utilizado diversas medidas para evaluar mis habilidades. Las calificaciones obtenidas en tareas y proyectos han sido las más frecuentes, lo que indica mi cumplimiento de las expectativas y la calidad de mi trabajo. Mis contribuciones en clase y mi asistencia constante también han influido en la percepción de mi dedicación a la materia. La puntualidad en la entrega de trabajos ha sido otro factor en revisión, siendo crucial el cumplimiento de los plazos establecidos. Además, en algunos cursos, se han utilizado evaluaciones tanto mías como de mis compañeros, lo que ha facilitado una evaluación minuciosa de mi desempeño, junto con el de los demás. Finalmente, las entregas mejoradas o revisadas han recibido en ocasiones reconocimiento favorable, lo que promueve el aprendizaje y el progreso continuos.

3.3.Análisis de Casos

Caso 1

Enfoque orientado a resultados en una empresa de desarrollo ágil

Empresa de desarrollo de software que implementa metodologías ágiles para crear app web para clientes del sector financiero.

Métricas utilizadas:

- **Velocidad del equipo:** Miden cuántos puntos de historia completan por sprint

(promedio de 40 puntos por sprint de 2 semanas)

- **Tiempo de ciclo:** Rastrean cuánto tarda una tarea desde que se inicia hasta que se completa (objetivo: menos de 3 días)
- **Calidad del código:** Utilizan SonarQube para mantener una cobertura de pruebas superior al 80%
- **Satisfacción del cliente:** Implementan encuestas NPS (Net Promoter Score) después de cada entrega
- **Frecuencia de despliegue:** Realizan despliegues a producción 2-3 veces por semana

Beneficios identificados:

- Los desarrolladores se centran en entregar valor real al cliente
- El equipo tiene autonomía para decidir cómo resolver problemas
- La calidad del código se mantiene alta gracias a las revisiones por pares
- Los clientes perciben mejoras continuas y respuestas rápidas a sus necesidades
- La motivación del equipo es alta porque ven el impacto de su trabajo

Problemas identificados:

- Ocasionalmente hay presión para completar más puntos de historia, lo que puede afectar la calidad
- La estimación de puntos de historia puede variar entre equipos, dificultando comparaciones
- Algunos desarrolladores se sienten presionados cuando su velocidad individual es menor.

Mejoras propuestas

Para mejorar tanto la calidad del desarrollo como el crecimiento del equipo, es esencial realizar revisiones de código más a menudo, ya que ayudan a identificar errores de manera oportuna y a compartir buenas prácticas. Asimismo, implementar un sistema de mentoría sería muy beneficioso para ayudar a aquellos desarrolladores que avanzan a un ritmo más pausado, fomentando el aprendizaje colaborativo. También es crucial reservar un tiempo en cada sprint para enfocarse en la reducción de la deuda técnica, ya que esta tiende a acumularse y puede afectar el rendimiento a largo plazo. Finalmente, no es suficiente con solo evaluar resultados cuantitativos; es necesario complementar estas métricas con análisis cualitativos que tomen en cuenta el esfuerzo, la creatividad y la colaboración dentro del equipo.

Caso 2

Enfoque de control estricto por métricas cuantitativas en una consultora de software

Empresa de creación de programas que cobra por cada hora a sus clientes y que se esfuerza por incrementar la eficacia cuantificable de sus programadores.

Métricas utilizadas:

- **Líneas de código producidas:** Objetivo de 200-300 líneas de código por día
- **Número de commits diarios:** Expectativa de 5-8 commits diarios
- **Horas facturables:** Mínimo de 7 horas facturables de las 8 horas laborales
- **Tiempo de respuesta a tickets:** Menos de 2 horas para comenzar a trabajar en un ticket asignado

- **Bugs reportados por desarrollador:** Penalización si superan cierto umbral mensual

Problemas identificados:

- Alta rotación de personal (más del 30% anual)
- Código inflado y redundante para cumplir con las métricas de líneas de código
- Fragmentación excesiva de commits para alcanzar los objetivos diarios
- Competencia negativa entre desarrolladores que daña la colaboración
- Baja calidad del código a largo plazo, generando más bugs y mantenimiento costoso
- Burnout en el equipo por la presión constante de cumplir métricas

Beneficios identificados:

- Visibilidad clara para la gerencia sobre la actividad diaria
- Facilidad para justificar costos a los clientes
- Identificación rápida de desarrolladores con baja actividad

Mejoras propuestas:

- Reemplazar métricas de líneas de código por métricas de calidad y valor entregado
- Implementar revisiones de código colaborativas en lugar de individuales
- Establecer tiempo protegido para aprendizaje y mejora técnica
- Medir la satisfacción del equipo y correlacionarla con la productividad

- Crear un sistema de reconocimiento basado en la calidad y el impacto, no en la cantidad
- Implementar retrospectivas regulares para identificar y eliminar obstáculos.

3.4.Dinámica: Construcción de una métrica ideal

Construcción de una Métrica Ideal para Equipos Ágiles

Fundamentado en estudios y enfoques óptimos, sugiero un conjunto de indicadores balanceado que mide la productividad de manera justa y eficiente:

Métrica: Índice de Impacto del Desarrollador (DII)

¿Qué mide?

El DII evalúa la participación efectiva del creador al integrar tres aspectos fundamentales:

- **Valor entregado (40%):** Impacto de las funcionalidades completadas en los objetivos de negocio y satisfacción del cliente
- **Calidad técnica (30%):** Sostenibilidad y excelencia del código producido
- **Colaboración y aprendizaje (30%):** Contribución al equipo y crecimiento profesional

¿Cómo se calcula?

DII = (0.4 × Valor) + (0.3 × Calidad) + (0.3 × Colaboración)

Donde:

- Valor = Promedio de (Historias completadas ponderadas por impacto + Satisfacción del cliente con las funcionalidades)

- Calidad = Promedio de (Cobertura de pruebas + Reducción de deuda técnica + Tasa de defectos)
- Colaboración = Promedio de (Contribuciones a revisiones de código + Transferencia de conocimiento + Participación en retrospectivas).
- Cada componente se normaliza en una escala de 0-10.

¿Qué problemas evita?

- Evita enfocarse solo en métricas cuantitativas como líneas de código o número de commits
- Previene la competencia tóxica entre miembros del equipo
- Reduce el riesgo de manipulación de métricas individuales
- Evita ignorar aspectos cualitativos importantes como la colaboración
- Previene la desalineación entre productividad y objetivos de negocio

¿Cómo se usa éticamente?

1- **Transparencia:** Los desarrolladores conocen exactamente cómo se calcula la métrica
2- **Contextualización:** Siempre se analiza en el contexto del equipo, no de forma aislada
3- **Mejora continua:** Se utiliza principalmente para identificar áreas de mejora, no para castigos
4- **Personalización:** Los pesos de cada componente pueden ajustarse según las prioridades del proyecto
5- **Complementariedad:** Se usa junto con retrospectivas y conversaciones

cualitativas

6- **Privacidad:** Los resultados individuales solo se comparten con el desarrollador, mientras que los datos agregados del equipo son público.

4. Conclusiones

Las mediciones de productividad en la creación de software deben balancear componentes tanto cuantitativos como cualitativos, centrándose no solo en la cantidad de tareas completadas, sino en el verdadero valor proporcionado a la empresa y al cliente. Un enfoque integral que una las evaluaciones de valor, la calidad técnica y la cooperación permite una valoración más equitativa y exhaustiva del rendimiento, evitando los impactos adversos de métricas unidimensionales como el número de líneas de código o de compromisos. La ejecución ética de las métricas demanda claridad, contextualización y un énfasis en la mejora continua más que en la evaluación punitiva. Las prácticas óptimas incluyen la participación de los desarrolladores en la creación de las métricas, ajustando los indicadores a las necesidades particulares del proyecto, y empleándolos como instrumentos para el desarrollo profesional y la optimización de procesos, en lugar de usarlos como herramientas de control que puedan fomentar rivalidades destructivas o comportamientos contraproducentes en el grupo.

5. Referencias bibliográficas

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution.

Kerzner, H. (2017). *Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance* (3rd ed.). Wiley.

Smith, T. (2020). *Productivity in Software Engineering: Beyond the Numbers*. IEEE Software, 37(2), 13–17. https://doi.org/10.1109/MS.2020.2967194

Wiggins, A. (2022). *Ethical Use of Metrics in Software Development*. Communications of the ACM, 65(4), 24–27. https://doi.org/10.1145/3514188