1. **Datos Informativos Módulo:** Unidad 2

Nivel: Séptimo

Fecha: 05/05/2025

Tema: Métricas de productividad del desarrollador y rendimiento del software en sistemas de NLP

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?**

Sé que soy productivo como desarrollador cuando cumplo con las tareas asignadas en los plazos establecidos, entrego código funcional y de calidad, recibo pocas correcciones en las revisiones de código, colaboro efectivamente con mi equipo y veo avances claros en los objetivos del proyecto.

- **Breve respuesta ¿Por qué medir? ¿Pa qué?**

Medir es crucial para entender el estado actual, identificar áreas de mejora y evaluar el progreso hacia los objetivos deseados. En esencia, medimos para mejorar y tomar decisiones informadas.

3.2. Presentación Teórica Usar las diapositivas:

- **Definición de métricas de productividad**

Las métricas de productividad son indicadores cuantitativos que evalúan el rendimiento individual o del equipo de desarrollo. Estas métricas evalúan aspectos como eficiencia, calidad, entrega y colaboración, y no deben usarse de forma punitiva sino para la mejora continua y toma de decisiones.

- **Tipos de métricas (LoC, commits, bugs, velocity, etc.)**

1- Líneas de código (LoC)

2- Commits por día/semana

3- Número de tareas completadas

4- Velocidad de entrega (velocity)

5- Tasa de errores o bugs

6- Revisión de código (pull requests)

7- Tiempo de ciclo (cycle time)

- **Métricas útiles vs. métricas engañosas Métricas engañosas:**

- Líneas de código ≠ calidad
- Muchos commits no significan productividad real - Medición sin contexto puede ser contraproducente **Métricas útiles:**
- Aquellas que combinan aspectos cuantitativos y cualitativos
- Las que se adaptan al contexto específico del equipo

- **Buenas prácticas**

- Combinar métricas cuantitativas y cualitativas
- Evaluar en conjunto con revisión de código y colaboración
- Adaptar las métricas al contexto del equipo
- Enfocarse en el rendimiento del equipo más que en individuos aislados
- Usar métricas para aprendizaje y no solo para control

- **Herramientas disponibles** - GitHub Insights

- Jira / Azure DevOps / Trello
- SonarQube (calidad de código)
- GitPrime / LinearB / Code Climate

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

En mi experiencia académica y profesional, solo he utilizado métricas relacionadas con Git para seguimiento de contribuciones y el cumplimiento de plazos de entrega para evaluar mi desempeño y el del equipo, lo que ha permitido mantener un ritmo constante de trabajo y asegurar que los proyectos se completen dentro de los tiempos establecidos.

3.3. Análisis de Casos

Caso 1

Enfoque orientado a resultados en una empresa de desarrollo ágil

Contexto: Empresa de desarrollo de software que implementa metodologías ágiles para crear aplicaciones 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:

Implementar revisiones de código más frecuentes para mantener la calidad

Establecer un sistema de mentoría para desarrolladores con menor velocidad

Incluir tiempo dedicado para reducir la deuda técnica en cada sprint

Complementar las métricas cuantitativas con evaluaciones cualitativas del trabajo

Caso 2

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

Contexto: Consultora de desarrollo de software que factura por hora a sus clientes y busca maximizar la productividad medible de sus desarrolladores.

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

Basado en la investigación y mejores prácticas, propongo un conjunto de métricas equilibrado que evalúa la productividad de forma ética y efectiva:

Métrica: Índice de Impacto del Desarrollador (DII) ¿Qué mide?

El DII mide la contribución real del desarrollador combinando tres dimensiones clave:

- **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 públicos

4. Conclusiones

- Las métricas de productividad en desarrollo de software deben equilibrar aspectos cuantitativos y cualitativos, enfocándose no solo en la cantidad de trabajo realizado sino en el valor real entregado al negocio y al cliente. Un enfoque holístico que combine mediciones de valor, calidad técnica y colaboración permite una evaluación más justa y completa del desempeño, evitando los efectos negativos de métricas unidimensionales como líneas de código o número de commits.
- La implementación ética de métricas requiere transparencia, contextualización y un enfoque en la mejora continua más que en la evaluación punitiva. Las mejores prácticas incluyen involucrar a los desarrolladores en la definición de las métricas,

Referencias:

Machuca Villegas, L. E. (2020). Factores sociales y humanos que influyen en la productividad del desarrollo de software.

Scott, M. E. (2016). *Un enfoque adaptativo para entrenar desarrolladores de software* (Doctoral dissertation, Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA)).

Zamudio, M. A., Palacio, R. R., & Castro, L. A. (2017). Diseño de métricas basadas en estresores laborales para desarrolladores de software. *Res. Comput. Sci.*, *136*, 9-19.