Métrica → es un número obtenido como resultado de un proceso de medición. Permite la construcción de indicadores que facilitan la toma de decisiones. Es el grado de presencia de un atributo en función de lo que quiero medir.

Métricas de software en el enfoque tradicional

Son definidas de forma clara por el líder de proyecto, al principio del proyecto. Cómo se van a tomar, quién las va a tomar, de dónde se van a obtener los datos… Las métricas no son perfectas, pero no significa que no haya que tomarlas, ya que nos sirven para obtener visión objetiva. No sirve tomar métricas que después no se vayan a usar, porque es una pérdida de tiempo. (salvo que se automaticen).

El dominio** de las métricas del software en el enfoque tradicional se divide en:

- Métricas de Proceso: son de la organización, públicas. Se utilizan para mejorar el proceso en el contexto de varios proyectos. La mayoría de las métricas de proceso se obtienen de los proyectos, por lo que pueden terminar siendo muy similares -la diferencia está en el ámbito. (Ej.: una métrica de desviación de estimación en un proyecto es, obviamente, del ámbito de proyecto. Así mismo, se puede plantear una métrica de desviación de estimación de la organización, y esta sería de ámbito de proceso)
  Una métrica de eficiencia en la remoción de defectos, por ejemplo, permite descubrir qué tan eficiente es el proceso con el tratamiento de defectos, mientras más rápido se detecten los defectos, más eficiente es el proceso.
- Métricas de Proyecto: es privada, se maneja con los miembros del proyecto. No se publican organizacionalmente sino hasta juntar muchas, promediarlas y despersonalizarlas. Por ejemplo, el esfuerzo que lleva desarrollar una funcionalidad del producto.
- Métricas de Producto: permite mejorar el producto, por lo que son métricas que se enfocan directamente en el software. Un ejemplo de esta métrica (muy mala) es la cantidad de líneas de código; mejores ejemplos son la satisfacción de cliente, calidad… Sin embargo, estos últimos no pueden medirse de forma directa, por lo que se debe medir "algo alrededor" para intentar medirlos. Qué porcentaje de requerimientos se han cumplido, cantidad de casos de uso por complejidad, índice de retención (si es un producto en el mercado), cantidad de defectos, cantidad de clases...

***Todas las métricas se toman **en el ámbito de un proyecto**, por más que el dominio sea proceso, proyecto o producto.*

***No se puede medir la productividad de la gente**, por eso la "P" de persona no existe en este contexto. Las métricas públicas deben despersonalizarse: apuntar a quién tuvo la culpa genera miedo, un mal ambiente de trabajo y daña la transparencia.*

¿Cuáles son las métricas básicas para un proyecto de software en este enfoque?

- Tamaño del producto: casos de uso por complejidad (nunca medir casos de uso solitos).
- Esfuerzo: se mide en horas persona lineales. Existen distintos niveles de granularidad: esfuerzo por actividades, o por workflow de proceso (testing, análisis, diseño, implementación), o por iteración, o por caso de uso…
- Tiempo (calendario): la unidad de medida cambia según el tamaño del proyecto (días, semanas, meses). Tiempo calendario, precisión de estimaciones en el calendario y el esfuerzo.

- Defectos: No sirve contar defectos sin discriminar, entonces es importante clasificarlos. Defectos por severidad, densidad de defectos (defectos x bloque de código -caso de uso, módulo, clase).

La precisión es cara. No tiene sentido medir a milímetros si estamos a millas del destino. Lo importante es mantener las métricas lo más simples posibles, pero que a su vez permitan la mayor transparencia.

Preguntas:
¿Nos da más información de la que tenemos hasta ahora? ¿Es esta información un beneficio práctico?

Tenemos que satisfacer necesidades y expectativas, pero distintas, porque son de distintas personas. Tenemos que elegir métricas que más o menos cubran los 3 aspectos que conforman la calidad que son recursos, tiempo y alcance.

El alcance se relaciona con las métricas del tamaño del producto, los recursos se relacionan con el esfuerzo, el tiempo se corresponde con el tiempo (obviamente) y los defectos son la forma más básica de medir calidad.

Métricas de software en el enfoque ágil

*Intentan ser ligeras, sencillas y pocas, **contemplando solo lo esencial**, porque las métricas no generan valor al cliente. Se basan en el software funcionando. No existen métricas de proceso, porque el agilismo plantea que la experiencia no puede extrapolarse entre un proyecto y otro.*

- Velocidad: cantidad de puntos de historia (PRODUCTO) aceptados por el Product Owner. No se estima, se calcula. Mide software funcionando. Si o si se mide en puntos de historia. Es una métrica de producto.
- Capacidad *(o velocidad estimada)*: permite definir hasta cuándo el equipo se puede comprometer. Es estimada en horas ideales o, los equipos más maduros, en puntos de historia. Para estimar en horas ideales, es necesario desagregar las User Stories en tareas (algo que hacen los equipos menos maduros). NO se miden User Stories en horas, porque las horas son trabajo, no producto. Es una métrica de proyecto.
- Running Tested Features *(RTF)*: no se utiliza más, porque se mide por *features* sin considerar la complejidad o dificultad de las *features* desarrolladas. Es decir, mide solo la cantidad de *features* completas por iteración.

Métricas en Kanban/Lean

Como es un framework de mejora de procesos, las métricas están enfocadas en el proceso.

- Lead time: es la vista del cliente. Es el tiempo desde que el cliente solicita un ítem de trabajo hasta que se le entrega. Se mide en días de trabajo o esfuerzo.
- Cycle time: es la vista del equipo, interna. Es el tiempo desde que empezamos a trabajar en un ítem de trabajo hasta terminarla por completo. Se mide en días de trabajo o esfuerzo, y es el ritmo de terminación.
- Touch time: tiempo en el cual un ítem de trabajo fue realmente trabajado por el equipo. Solo tiene en cuenta el tiempo en que un ítem de trabajo está en una columna de "trabajo en curso", no de estado bloqueado o espera. Es la métrica más pequeña.

- Eficiencia del ciclo de Proceso: Es igual al Touch Time / Lead Time. Lo mejor es que esté más cercano al "1", ya que significaría que hubo poco o nulo desperdicio.

$$\boldsymbol{Touch\ Time} \leq \boldsymbol{Cycle\ Time} \leq \boldsymbol{Lead\ Time}$$

$$\%\boldsymbol{Eficiencia} = \frac{\boldsymbol{Touch\ Time}}{\boldsymbol{Lead\ Time}}$$

- Métricas orientadas a servicio: Se expresan en los SLA – *Service Level Agreements*. Acá es donde se empieza a medir el producto.
    - Expectativa de nivel de servicio que los clientes esperan: establece una combinación entre prioridad y severidad de los defectos, y se fija cuánto tiempo tienen los desarrolladores para solucionar los defectos.
    - Capacidad del nivel de servicio al que el sistema puede entregar
    - Acuerdo de nivel de servicio que es acordado con el cliente
    - Umbral de la adecuación del servicio el nivel por debajo del cual este es inaceptable para el cliente

Hay 2 tipos de desperdicio: Evitables (reuniones, pero son muy útiles) e Inevitables (no se pueden eliminar). Es por los desperdicios inevitables que es complicado que el valor de la eficiencia del ciclo del proceso llegue a 1.

¿Cómo se mide el producto?

Tamaño

- ➔ Líneas de código
- ➔ Requerimientos

Defectos

- ➔ Cobertura: cuántas funcionalidades del sistema hemos probado, sobre el total de funcionalidades del producto.
- ➔ Defectos por severidad
- ➔ Densidad de defectos