Diez métricas de calidad del software que debes conocer

1. Rotación total de códigos
2. Tasa de fallas
3. Disponibilidad del sistema
4. Densidad de defectos
5. Tiempo medio de detección (MTTD)
6. Tiempo medio entre fallas (MTBF)
7. Tiempo medio de resolución (MTTR)
8. Satisfacción del cliente (CSAT)
9. Tiempo medio para remediar una vulnerabilidad
10. Cobertura de código

1. Rotación total de códigos

Es la cantidad de líneas de código agregadas, modificadas o eliminadas de una base de código durante un período de tiempo.

Aunque desarrollar y mantener software consiste en cambiar una base de código varias veces, hacerlo con demasiada frecuencia puede indicar problemas de calidad. Esto se debe a que, como regla general, cuantos más cambios, mayor será la probabilidad de introducir errores en los productos.

No mide la calidad per se, pero la alta rotación podría correlacionarse con otros problemas de métricas de calidad.

Esta métrica está formada por los siguientes datos:

- Líneas de código agregadas = lc_a
- Líneas de código modificadas = lc_m
- Líneas de código eliminadas = lc_e
- Periodo de tiempo de la medición (por lo general por Sprint) = t

$$Rotación\ total = \frac{lc_a + lc_m + lc_e}{t}$$

Ejemplo: 10,000 líneas de código en tres sprints.

2. Tasa de fallas

La tasa de fallas es la frecuencia con la que falla un producto de software.

Y cuando eso sucede, los usuarios finales simplemente no pueden operar el producto en absoluto.

Los fallos son errores inesperados, como un defecto que los testers no detectaron o una sobrecarga repentina del sistema. Ocurren mientras el software se está ejecutando y finalizan su ejecución abruptamente. A veces, los datos se pierden en el proceso.

Las empresas de sistemas críticos, como las que desarrollan sistemas de control de tráfico aéreo o sistemas de bolsa, se centran mucho en esta métrica.

Esta métrica está formada por los siguientes datos:

- Número de fallas = N_f
- Periodo de tiempo de la medición (por lo general es por días) = t

$$Tasa\ de\ fallas = \frac{N_f}{t}$$

Ejemplo: una caída del sistema por cada 30 días.

3. Disponibilidad del sistema

Evalúa la cantidad de tiempo que un sistema está en funcionamiento. Compara el tiempo de actividad del sistema con su tiempo de inactividad durante un período de tiempo. Y considera que un sistema que está disponible es tan rápido como se espera y no falla a menos que ocurran condiciones imprevistas.

Otra forma de decirlo: la disponibilidad del sistema es la probabilidad de que un producto de software no deje de estar disponible cuando los usuarios lo necesiten.

Una alta disponibilidad significa más tiempo de actividad que de inactividad, lo que significa que los usuarios rara vez pierden el acceso al sistema. En ese caso, el producto lanzado no contiene errores o solo algunos de ellos, lo que requiere pocas correcciones en su código base.

Esta métrica está formada por los siguientes datos:

- Tiempo de actividad (medido en horas) = T_a
- Tiempo de inactividad (medido en horas) = T_i

$$\text{Disponibilidad} = \frac{T_a}{T_a + T_i} * 100$$

Ejemplo: un producto de software que está activo nueve horas de cada diez tiene una disponibilidad del sistema del 90 %.

4. Densidad de defectos

Es la cantidad de defectos en un producto de software en comparación con su tamaño. Esto significa que es un número relativo.

Los defectos son errores encontrados por los evaluadores antes del lanzamiento del producto. Representan necesidades de usuario insatisfechas. Y si los evaluadores no los detectan a tiempo, los defectos originan fallas en manos de los usuarios finales.

Esta métrica mide la calidad del código y le brinda información para estimación de los esfuerzos de corrección del software. Un código de alta calidad requiere pocas correcciones y es más fácil de mantener, escalar y evolucionar.

Esta métrica a menudo se expresa como la cantidad de defectos por cada 1000 líneas de código y está formada por los siguientes datos:

- Número de defectos = N_d
- Número total de líneas de código = N_c

$$Densidad\ de\ defectos = \frac{N_d}{\frac{N_c}{1000}}$$

Ejemplo: Diez defectos en un total de 20,000 líneas de código equivalen a una densidad de defectos de 0,5 por 1,000 líneas de código.

5. Tiempo medio de detección (MTTD)

Es el tiempo promedio que le toma a tu equipo detectar errores en un producto de software.

Un MTTD bajo indica que, normalmente, su equipo encuentra errores rápidamente. Y cuanto más rápido los encuentren, más rápido corregirán esos errores. Como resultado, el sistema no está inactivo por mantenimiento durante demasiado tiempo, lo que minimiza el impacto de los defectos o fallas en los usuarios finales.

Este indicador está formado por los siguientes datos:

- Tiempo total para detectar todos los errores durante un período de tiempo = T_d
- Número total de errores = N_e
- Tiempo para detectar un error = T_{d1}
- Día y hora en que se detectó = D_d
- Día y hora en que ocurrió = D_o

$$MTTD = \frac{T_d}{N_e}$$

Ejemplo: en agosto de 2023, una aplicación produjo 20 errores con un tiempo total de detección de 1000 minutos, lo que equivale a un MTTD de 50 minutos

6. Tiempo medio entre fallas (MTBF)

Es el tiempo promedio entre dos fallas del sistema. Se trata de errores que se encuentran tras el lanzamiento del producto y que se deben, por ejemplo, a un defecto no detectado.

Las fallas pueden ser tan graves como un bloqueo, pero también pueden deberse a que el software no hace lo que los usuarios finales esperan que haga.

Por supuesto, cuanto mayor sea el MTBF, mejor. Significa que el producto es más fiable, lo cual es esencial en industrias como la sanitaria y la aeronáutica.

Abordar una tasa MTBF problemática puede ser un poco más complejo.

Este indicador está formado por los siguientes datos:

- Tiempo total de ejecución medido en horas= T_e
- Número total de fallas = N_f

$$MTBF = \frac{T_e}{N_f}$$

Ejemplo: un software que se ha estado ejecutando durante 3000 horas y ha fallado 15 veces tiene un MTBF de 200 horas, lo que significa que falla en promedio una vez cada 200 horas.

7. Tiempo medio de resolución (MTTR)

Es el tiempo promedio que le toma a tu equipo resolver un error en un producto de software después de que alguien lo descubre.

Esto se calcula como la cantidad de horas o minutos que transcurren entre que se detecta el problema y se resuelve. Generalmente, sólo se cuentan los minutos u horas que caen dentro del horario laboral normal (es decir, no contaría el tiempo transcurrido en las noches o los fines de semana).

Regularmente, un MTTR bajo significa que tu equipo repara los errores rápidamente.

Pero, por supuesto, depende de la gravedad del error y de la experiencia de tus desarrolladores.

Este indicador está formado por los siguientes datos:

- Tiempo total desde el descubrimiento hasta la resolución en minutos = T_d
- Número total de reparaciones = N_r

$$MTTR = \frac{T_d}{N_r}$$

Ejemplo: si transcurrieron 2,880 minutos en total entre el descubrimiento y la resolución de 96 errores en un producto durante los últimos tres meses, entonces el MTTR de ese producto es de 30 minutos para el trimestre.

8. Satisfacción del cliente (CSAT)

Es un número que representa la forma en que los clientes experimentan tu producto de software. Y se llega a ello recopilando y analizando los datos de las encuestas de satisfacción de tus clientes sobre el funcionamiento del software. No se incluyen datos sobre la calidad del servicio como por ejemplo el soporte en tu mesa de ayuda.

Por lo general, una pregunta en esas encuestas es sobre la satisfacción general de los clientes con tu producto. Te recomiendo que den una respuesta en una escala de cinco puntos, desde “extremadamente satisfecho” hasta “extremadamente insatisfecho”.

Esta métrica de calidad te brindará la percepción de los usuarios sobre la calidad general de tu producto. Cuanto mayor sea el CSAT, mejor será esa percepción de calidad.

Este indicador está formado por los siguientes datos:

- Número total de clientes satisfechos = C_s , siendo los clientes satisfechos aquellos que calificaron su satisfacción como “extremadamente satisfecho” y “satisfecho”.
- Número total de respuestas a la encuesta de clientes = N_r

$$CSAT = \frac{C_s}{N_r}$$

Ejemplo: si 53 de 100 clientes calificaron su satisfacción como "extremadamente satisfecho" y "satisfecho", entonces el CSAT de su producto es 53% o solo 53.

9. Tiempo medio para remediar una vulnerabilidad

Es el tiempo promedio que le toma a tu equipo reparar las vulnerabilidades de ciberseguridad en tu producto de software.

La causa principal de esas vulnerabilidades suele ser el código de baja calidad. Y un número bajo significa que los usuarios finales están menos expuestos a posibles violaciones de datos, pérdidas financieras e interrupciones repentinas del servicio.

De manera similar al MTTR, el tiempo medio para remediar una vulnerabilidad depende de qué tan bien tu equipo implemente las mejores prácticas de codificación. Pero en este caso, las mejores prácticas que siempre deben monitorear, incluso después del lanzamiento del producto, son las de ciberseguridad.

Este indicador está formado por los siguientes datos:

- Tiempo total entre el descubrimiento y la corrección de vulnerabilidades = T_t
- Número total de vulnerabilidades = N_v
- Periodo de tiempo = t , generalmente en días.

$$Tiempo\ medio\ para\ remediar\ una\ vulnerabilidad = \frac{T_t}{N_v}$$

Ejemplo: si un equipo de desarrollo pasó 20 horas reparando dos vulnerabilidades en un producto el último trimestre, entonces el tiempo medio del producto para remediar una vulnerabilidad es de 10 horas por trimestre.

10. Cobertura del código

Evalúa la cantidad de código fuente de tu producto para el cual existe una prueba unitaria.

Se trata de pruebas realizadas por programadores en el entorno de desarrollo que utilizaron para crear el producto. Y el objetivo es encontrar errores en el código base lo antes posible y evitar fallas del sistema en el futuro.

Para ello, los programadores deben cubrir cada línea de código con una prueba unitaria. Por ejemplo, las pruebas unitarias deben cubrir todas las condiciones de los casos de uso del producto.

Este indicador está formado por los siguientes datos:

- Número de líneas de código probadas = L_p
- Número total de líneas de código en el código base = L_t

$$Cobertura\ del\ código = \frac{L_p}{L_t}$$

Ejemplo: si una base de código tiene 10 000 líneas de código y las pruebas unitarias cubren 9500 de esas líneas, entonces su producto tiene una cobertura de código del 95%.