Uso de Métricas en la Ingeniería de Software

El proceso de planificación del desarrollo de cualquier sistema debe hacerse partiendo de una estimación del trabajo a realizar. Sólo a partir de ello es factible conocer los recursos necesarios y el tiempo necesario para su realización.

Una **métrica** es una medida efectuada sobre algún aspecto del sistema en desarrollo o del proceso empleado que permite, previa comparación con unos valores (medidas) de referencia, obtener conclusiones sobre el aspecto medido con el fin de adoptar las decisiones necesarias. Con esta definición, la definición y aplicación de una métrica no es un objetivo en sí mismo sino un medio para controlar el desarrollo de un sistema de software.

Métricas sobre el producto

Las métricas sobre el producto están orientadas a estimar las características del mismo antes de su desarrollo. Estas estimaciones se basan en el conocimiento que los desarrolladores adquieren a partir de datos obtenidos de proyectos anteriores.

A) Tamaño estimado del código

La forma más obvia y la que se ha utilizado históricamente para estimar el tamaño es contar el número de líneas de código. Con ciertas normas para determinar qué es lo que se cuenta (líneas de comentario, código incluido, etc.) y siempre referido a un lenguaje concreto, lo que los valores nos dan es un valor para, comparando con otros casos, poder estimar el esfuerzo necesario en futuros desarrollos.

Los resultados obtenidos (estimaciones y valores reales) alimentan la base de datos históricos que es el fundamento para posteriores estimaciones. Boehm desarrolló una técnica empleando el método Delphi para mejorar las estimaciones con múltiples opiniones de expertos. La idea de emplear el método Delphi es asegurar en dos o tres pasos de convergencia que las estimaciones son aceptadas por los expertos.

Ha sido muy criticada la tendencia en estimar el esfuerzo en base a las líneas de código. Una de las críticas se centra en que la complejidad del desarrollo no está directamente ligada al tamaño cuando nos movemos hacia el dominio de los sistemas concurrentes, distribuidos o de tiempo real. En ellos, las medidas deben referirse a estimaciones del mismo tipo de productos.

Otro problema surgido recientemente con la proliferación de generadores de código es que no importa demasiado el número de líneas de código generadas (excepto por problemas derivados del tamaño de la memoria para sistemas embebidos) sino el número de líneas de especificación que las han generado, porque la complejidad del problema de mantenimiento depende de ello.

B) Complejidad estimada .

Con el fin de superar el problema de las estimaciones del tamaño de código, se ha prestado recientemente atención a medidas de complejidad no basadas en estimaciones de número de líneas.

Albrecht definió en 1979 un método conocido como de **puntos de función** que está teniendo cada vez más aceptación. Su método se basa en el empleo de factores normalizados para juzgar la importancia relativa de varios requisitos funcionales.

Parte de cinco funciones básicas que suelen aparecer en muchos sistemas:

- Entradas: Pantallas o formatos empleados para introducir datos a un programa.

- Salidas: Pantallas o informes empleados para utilizarlos con otros programas o para lectura directa .

- Consultas: Mecanismos para pedir ayuda o dar órdenes de ejecución.

- Ficheros de datos: Conjuntos lógicos de información empleados por una aplicación (ya sean tablas en memoria como ficheros de disco) junto con los procedimientos de acceso a los mismos.

- Interfaces: Ficheros compartidos con otras aplicaciones.

La idea básica del método consiste en definir unas estimaciones de complejidad para cada una de estas funciones (en forma de pesos relativos) y estimar, dadas las especificaciones del sistema, cuántos elementos de cada tipo van a ser necesarios.

El problema con los puntos de función es que no son realmente medidas sino valoraciones subjetivas y no tienen en cuenta diferencias en la implementación (al fin y al cabo, el esfuerzo del desarrollo depende también del lenguaje utilizado o del dominio de aplicación y eso no se tiene en cuenta). De nuevo, la comparación con sistemas similares permite “calibrar” las decisiones tomadas.

C) Robustez .

Por **robustez** de un programa se entiende la ausencia de fallos en su ejecución con diferentes datos de entrada durante intervalos de tiempo predeterminados.

La robustez de un programa está ligada a la aparición de problemas durante su ejecución. Generalmente, el número de fallos encontrados durante la fase de prueba y, posteriormente, durante el mantenimiento del sistema constituye una medida de la calidad del producto de software e indirectamente, de la calidad del proceso de desarrollo.

La importancia de conocer el número de fallos encontrados en un intervalo de tiempo no reside únicamente en obtener un valor global de la calidad del producto sino en los beneficios derivados de su análisis.

Las medidas estadísticas de fiabilidad (tiempo medio entre fallos encontrados durante la ejecución, reducción del número de recopilaciones necesarias, etc.) sirven para alimentar el proceso de desarrollo.

Métricas sobre el proceso

Las métricas mencionadas anteriormente estaban orientadas a conocer la complejidad del producto (con algún valor indirecto como el tamaño) para poder estimar los recursos necesarios para su realización. Hemos mencionado también que, según se vayan acumulando datos y se analicen estadísticamente, las estimaciones serán cada vez mejores. Esto nos servirá para planificar mejor futuros desarrollos.

Existen otros tipos de datos que se pueden tomar durante el desarrollo de un producto de software y que no están ligados al producto sino a los procesos implicados. El análisis de cómo estos procesos se realizan a partir de medidas tomadas en el desarrollo es la base para su posterior mejora.

Algunos de los elementos a medir son:

• Distribución del esfuerzo en cada una de las fases con objeto de poder estimar los recursos necesarios. Obsérvese que esta medida es complementaria a las de tamaño mencionadas anteriormente; aquella nos permitía conocer los recursos globales necesarios; de lo que se trata aquí es de obtener medidas reales y extrapolarlas a futuros proyectos.

• Productividad medida en número de líneas de código documentadas que es capaz de producir una persona en una unidad de tiempo. Podemos decir que los valores típicos de productividad por persona (empleando tecnologías de desarrollo convencional) están entre 30 y 50 líneas de código por día de trabajo.