aprenderaprogramar.com

Calidad del software. Métricas y fiabilidad de aplicaciones (2ª parte) (DV00104A)

Sección: Divulgación

Categoría: Tendencias en programación

Fecha revisión: 2029

Autor: César Krall

Resumen: Este artículo explica cuestiones básicas sobre la calidad del software y sobre qué son las métricas, norma CMMI, etc. Resume y comenta una conferencia impartida por Ramiro Carballo (Ingeniero de la empresa GESEIN) en la Escuela de Ingeniería Informática de la Universidad de Sevilla.

CONTINUACIÓN: CALIDAD DEL SOFTWARE ¿POR QUÉ DEDICAR RECURSOS A LA CALIDAD?

Un error común es pensar que si un proyecto de software con una persona se desarrolla en 12 meses, con dos personas se desarrollará en 6 meses o con tres personas en 4 meses. Esto no es así por muchos motivos, entre otros porque hay fases como las pruebas y el diseño que no pueden solaparse.

La búsqueda del software "perfecto" puede ser contraproducente: dedicar tiempo a eliminar errores muy pequeños o de muy escasa probabilidad de que se produzcan significa que tendremos más gastos. Por tanto lo que interesa es un software con la calidad adecuada para la satisfacción del cliente, no un software perfecto. Si nos planteamos entregar software sin defectos, posiblemente nunca lleguemos a entregar un proyecto. Hay que buscar un equilibrio: ni perfecto, ni demasiado imperfecto…

Cuando se entrega un proyecto de software a un cliente lo normal es que se le ofrezca un periodo de garantía. Durante ese periodo pueden ocurrir varias cosas:

a) Hay fallos pero el cliente no los encuentra. No supone ningún problema.

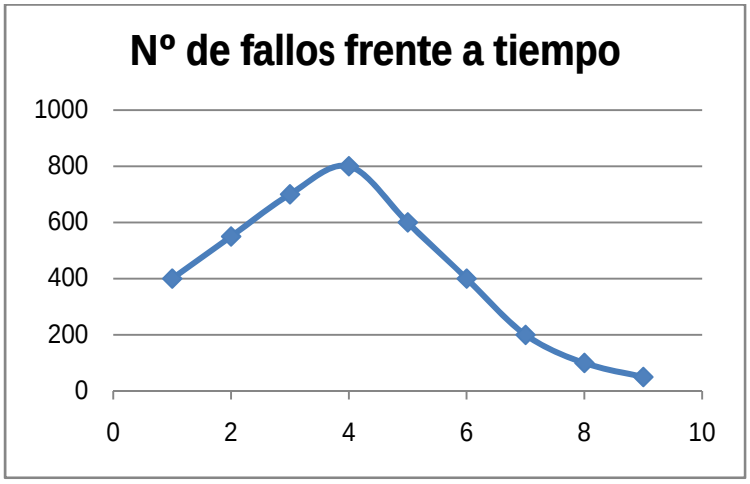
b) El cliente encuentra un número moderado de fallos. Nos supondrá tener que realizar cierto número de correcciones, lo cual debíamos haber contemplado dentro del presupuesto y no suponernos ningún problema.

c) El cliente encuentra un número elevado de fallos o el producto le resulta no satisfactorio. Esta es la situación que resulta problemática (y no tan infrecuente como sería deseable) y que nos obliga a realizar una inversión de tiempo y recursos humanos para hacer una corrección en profundidad. Con el riesgo añadido de que al introducir cambios significativos nos arriesgamos a generar nuevos fallos o problemas que eternicen el proyecto. Evitar estas situaciones es lo que justifica utilizar modelos de procesos como CMMI y dedicar suficientes testers y tiempo a las pruebas. Los "atajos" en la producción de software suelen ser a la larga una mala solución.

CRITICIDAD DE LOS FALLOS Y EVOLUCIÓN DE LOS FALLOS EN EL TIEMPO

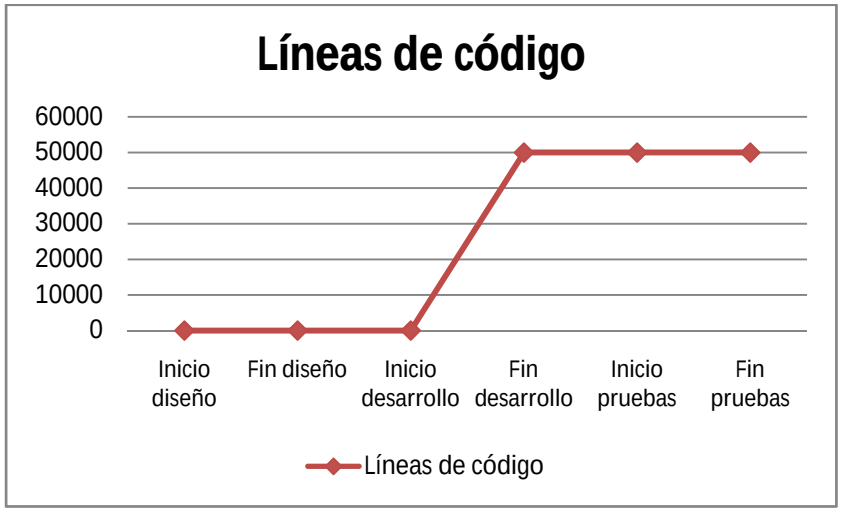
Un fallo en el software puede tener distintos niveles de criticidad. No es lo mismo un fallo en el diseño de la base de datos que obligue a replantear una parte importante del proyecto que un error en la sintaxis del código que no tiene mayores implicaciones para el proyecto en su conjunto. Por tanto cuando realizamos control de calidad del software además de registrar el número de fallos debemos registrar su relevancia.

La evolución habitual de los fallos en los proyectos es la siguiente: al iniciar las pruebas los fallos van en ascenso hasta un nivel máximo a partir del cual descienden. Este descenso nos indica que el código va quedando limpio.



A medida que se van encontrando menos fallos se va retirando personal de pruebas (testers). El criterio para retirar un tester suele ser un indicador del tipo: Retirar tester cuando se produce: Rendimiento ≅ Horas de tester / Fallos encontrados < Umbral. Cuando son necesarias más horas de tester que las económicamente asumibles, se retira el tester.

Si analizamos el número de líneas de código por etapas lo normal es que sea de este tipo:



Durante el diseño y pruebas el código no crece o lo hace mínimamente. Parece que estuviéramos "parados", sin embargo son etapas fundamentales para que un proyecto pueda llegar a buen puerto.

MÉTRICAS, REPOSITORIOS Y PLANES DE PROYECTO

Con todos los datos que vamos acumulando durante el ciclo del software, como número de testers dedicados, tiempo de pruebas, programadores dedicados, número de reclamaciones del cliente, tiempo medio entre reclamaciones del cliente, etc. obtenemos medidas (métricas) que nos permiten extraer conclusiones a partir del análisis estadístico. Estas métricas nos permiten poder evaluar cómo se va a comportar un proyecto en el futuro: cuánto personal va a consumir, número de errores previsible, satisfacción del cliente esperable, etc.

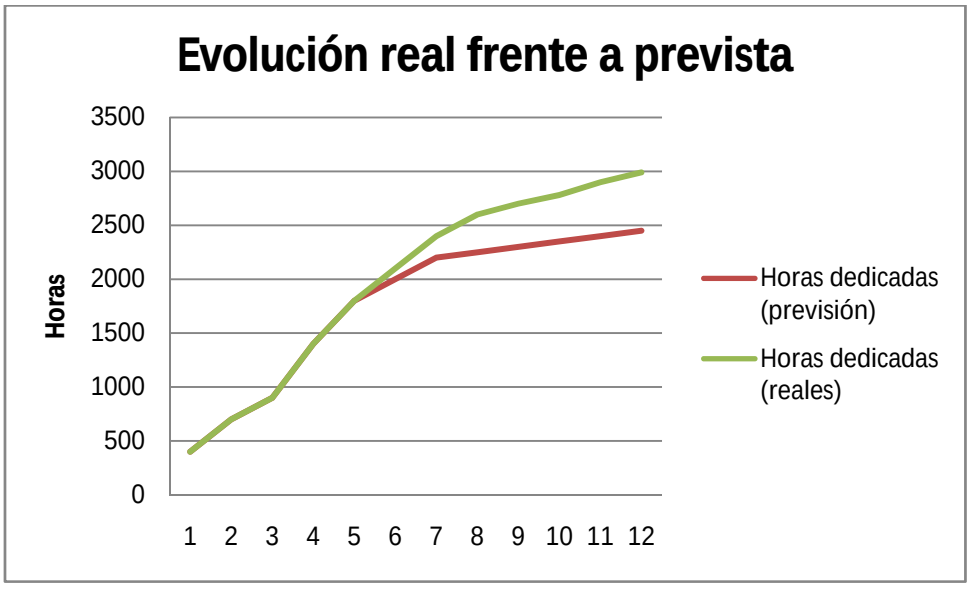
Es importante mantener bien almacenados en una base de datos los parámetros que definen los proyectos que hayamos hecho. Estas bases de datos se llaman repositorios. Existen repositorios públicos pero los realmente potentes son los propios de una empresa porque reflejan fielmente lo que sucede en ella.

En un repositorio podemos citar algunos datos fundamentales por proyecto como:

a) Tamaño. Medido por ejemplo como número de líneas de código o número de puntos función.
b) Esfuerzo en recursos humanos: horas de trabajo por categorías.
c) Duración del proyecto.
d) Defectos encontrados durante el proyecto.
e) Tipo de proyecto incluyendo datos relativos al lenguaje utilizado, sector del cliente (gestión, telecomunicación, tiempo real, etc.)
f) Presión del cliente en cuanto a plazos.
g) Relación entre parámetros en el tiempo (productividad).
h) Número de fases en que se desarrolló el proyecto.

Con los datos disponibles en el repositorio de nuestra empresa podremos analizar el encargo de un cliente y hacer un plan de proyecto donde plasmamos lo que se puede prever que va a suponer ese proyecto: cuánto va a durar el desarrollo, cuántas líneas de código vamos a tener cada mes durante el desarrollo, cuánto personal va a haber por fases y cuáles van a ser los gastos previstos en el tiempo.

Una vez en marcha el proyecto, compararemos las previsiones contempladas en el plan de proyecto con lo que realmente esté sucediendo. Los resultados pueden ser de este tipo:



El jefe de proyecto deberá manejar gráficos y tablas de datos de este tipo para conocer en todo momento la marcha del proyecto y poder tomar las decisiones oportunas. Con un panel de gráficos y tablas podrá conocer en cualquier momento cómo evoluciona el proyecto respecto a lo previsto en sus diferentes parámetros: nivel de gastos, nivel de fallos en el código, satisfacción del cliente, etc.

Con CMMI se consigue optimizar el proceso de creación y entrega de software. A la larga tiene diferentes ventajas como ser más eficiente, lograr mayores niveles de satisfacción del cliente y obtener beneficios económicos para la empresa.

La implantación de CMMI debe ser gradual. Lo normal es empezar con el nivel 1, el más sencillo, y progresivamente ir implantando niveles hasta llegar al nivel 5, el más exigente.

REFERENCIAS Y MÁS INFORMACIÓN

Este artículo resume y comenta la conferencia pública impartida por Ramiro Carballo, Ingeniero Informático de la empresa GESEIN, en el marco de las "Jornadas Imaginática: La informática del futuro", que tuvieron lugar en la Escuela Técnica Superior de Informática de la Universidad de Sevilla (España) y a las que tuvimos la oportunidad de asistir.

Gesein es una empresa con sede en Madrid que cuenta con unos 250 empleados. Cuentan con una sección de consultoría para la optimización del desarrollo del software. Trabajan en la implantación de modelos de procesos como ISO 9001, CMMI, etc.

Para más información: Asociación Española para la calidad: www.aec.es; www.gesein.com.