



Los costos de subestimar los defectos del software

Cada año se pierden miles de millones de euros a causa de los defectos del software. Sin embargo, muchas empresas no son conscientes de los riesgos que conllevan el desarrollo y la modificación de aplicaciones informáticas. Para solucionar este problema, primero se han de identificar y medir estos riesgos, para después pasar a neutralizarlos mediante un proceso de pruebas estructurado.

El software no sólo está cada vez más presente en nuestras vidas cotidianas (casi sin que nos demos cuenta de ello), sino también se vuelve cada vez más complejo y extenso. Prácticamente todos los sistemas críticos, como aparatos médicos, centrales nucleares o sistemas de guiado de trenes y aviones, hacen un uso intensivo del software. Además, el diseño e implementación de software es un proceso cada vez más complejo, en el cual -como demuestran los ejemplos a continuación- cualquier fallo puede tener gravísimas consecuencias.

Ejemplo:

En 1996, la definición incorrecta de un solo campo del software que guiaba el cohete Ariane 5 convirtió su lanzamiento en un fracaso. Al cabo de 36,7 segundos, el cohete tuvo que ser destruido en pleno vuelo tras haberse desviado demasiado de su curso programado. Con el cohete se perdieron 4 satélites, lo que acarrió unos costes totales de 500 millones de dólares.^[1]

Ejemplo:

Las ventas de una empresa multinacional de artículos de deporte cayeron en más de 100 millones de dólares después de que un error en el sistema de planificación de recursos provocara retrasos en los pedidos y un exceso de inventarios. Al día siguiente del anuncio de los problemas por parte de la empresa, el valor de sus acciones cayó en 10 dólares por acción.^[2]

Impacto económico de los defectos del software

Aún cuando no faltan ejemplos de problemas ocasionados por los defectos del software, muchas organizaciones siguen subestimando sus riesgos y posibles consecuencias.

Esto resulta aún más sorprendente cuando consideramos el impacto económico de los defectos del software. Según cálculos del Instituto Nacional de Estándares y Tecnología (NIST) de los Estados Unidos, en el año 2002 las pérdidas por errores del software en el mercado norteamericano rondaron los 67,5 mil millones de

euros[†], lo que equivale al 0,6% del Producto Interior Bruto del país.^[3] Si aplicamos este porcentaje al PIB español de 2007, 1050 mil millones de euros^[4], resultaría en unos costes anuales de 6,3 mil millones de euros, o 17,3 millones de euros al día.

Utilicemos el siguiente ejemplo práctico para ilustrar los costes de un defecto del software: debido a un error en el batch nocturno, los sistemas no pueden arrancar en la mañana por lo que no se puede comenzar con las actividades diarias. Imaginemos que dicho problema afecta a una empresa de mil empleados, y su resolución tarda una hora. Una vez resuelto el problema, habrá de esperar media hora más hasta que todos los sistemas vuelvan a ser operativos y puedan reanudarse las actividades con normalidad. Entonces, la pérdida de productividad directa será de 1,5 horas * 1.000 * 40 euros / hora^[5] = 60.000 euros. A este importe hay que sumar varios otros costes, como los relacionados con la corrección del error en la aplicación, las ventas perdidas y las horas extra necesarias para recuperar el trabajo acumulado. Entonces, en este caso los daños totales fácilmente superarán los 100.000 euros.

En el caso de instituciones financieras o empresas basadas en Internet, los daños producidos por errores del software pueden llegar a ser mucho mayores, por ejemplo cuando impiden las transacciones o limitan el acceso de los clientes a sus servicios.

Ejemplo:

En 1999, la página web de una empresa destinada a la subasta de productos a través de Internet sufrió un fallo técnico que perduró 21 horas. El resultado fue un total de 1,2 millones de clientes perjudicados, pérdidas económicas por valor de 5 millones de dólares y una reducción del 11% del valor en bolsa de la empresa.^[6] Asimismo, se dañó la imagen de la empresa.

Pero tenemos garantía ¿verdad?

Una suposición frecuente en relación con los defectos del software es que los perjuicios ocasionados están cubiertos por la garantía del proveedor. Sin embargo, ésta sólo incluye la corrección de los errores en la aplicación, no la indemnización por los daños adicionales que éstos han ocasionado:^[7]

- Pérdida de ventas;
- Imagen deteriorada;
- Pérdida de productividad; y
- Reclamaciones por daños y perjuicios.

Cada uno de estos factores puede llegar a poner en riesgo la continuidad de la organización:

Ejemplo:

A finales de los años ochenta, un fallo en el software de un acelerador lineal para la administración de radiación radioterapéutica provocó la muerte de varios pacientes a consecuencia de una sobredosis de radiación letal.^[8]

En consecuencia, es necesario disponer de herramientas

[†] Calculado en base al tipo de cambio dólar/euro al 31-12-2002 (Fuente: BCE)

tas que permitan cuantificar este valor en riesgo y, sobre todo, reducirlo.

Riesgo producto y testing estructurado

En las grandes industrias como la aeronáutica o la química, el control del riesgo producto es una práctica estandarizada. Es más: Ud. probablemente no se subiría a un avión si le dijeran que éste no ha sido sometido a un proceso de pruebas minucioso. Y sin embargo, se sigue ignorando la necesidad de controlar el riesgo producto del software, incluso cuando dependemos cada vez más de ello y se vuelve cada vez más complejo – y por ende más susceptible a los fallos.

El riesgo producto del software se define de la siguiente manera:^[9]

$$R_{\text{producto}} = P_{\text{error}} * F_{\text{uso}} * \Sigma \text{daño esperado}$$

R_{producto} = riesgo producto

P_{error} = probabilidad de que el producto contenga un fallo

F_{uso} = frecuencia de uso

$\Sigma \text{daño esperado}$ = daño esperado en el caso de que se produzca un fallo

Entonces, el riesgo producto depende de la probabilidad de que el producto contenga un fallo, la frecuencia de uso de ese producto y el daño esperado en caso de que se produzca un fallo.

El testing estructurado (del que hablaremos en más detalle en el artículo siguiente) es la solución indicada para reducir este riesgo producto. Su metodología consiste en someter la aplicación a un proceso de pruebas estructurado, el cual simula un uso muy intensivo en un espacio de tiempo reducido con el fin de identificar posibles fallos para poder corregirlos a tiempo. Su objetivo, por lo tanto, no es influir en la frecuencia de uso ni en el daño esperado en caso de un fallo, sino reducir la probabilidad de que se produzca un fallo.

El testing estructurado casi nunca conseguirá eliminar el 100% de los fallos de una aplicación (no suele ser necesario un margen de error de cero), pero permite identificar, y controlar, los riesgos producto del software. En este sentido, el testing es una inversión en el control de riesgos.

Por último, en lo referente al impacto económico de aplicar el testing estructurado, el estudio del NIST al que hicimos referencia previamente concluye señalando que, aunque no se puedan eliminar todos los errores del software, se puede ahorrar más del 37% de los costes calculados de estos errores. Por lo tanto, siguiendo con el ejemplo anterior del mercado americano, la introducción de una mejor infraestructura de pruebas que permita la detección y eliminación más rápida y eficaz de errores, podría reducir las pérdidas económicas por errores de software en 25 mil millones de euros (67,5 mil millones * 0,37). Si aplicamos estas cifras una vez más a la situación en el mercado español, el resultado sería un ahorro anual de 3,4 mil millones de euros (6,3 * 0,37).

En el artículo siguiente

En el siguiente artículo trataremos con más detalle las pruebas estructuradas, las cuales, en varios países, ya se han convertido en best practice para remediar los riesgos en el desarrollo de aplicaciones de software.

[1] ESA Plain Text Press Release; N° 33-1996: Ariane 501 - Presentation of Inquiry Board report

[2] Reuters News Report, 26 de febrero de 2001

[3] Software Errors Cost U.S. Economy \$59.5 Billion Annually; National Institute of Standards and Technology (NIST); 28 de junio de 2002

[4] Eurostat:

<http://epp.eurostat.ec.europa.eu/tgm/table.do?tab=table&language=en&pcode=daa10000&plugin=0&tableSlection=2&footnotes=yes&labeling=labels>

[5] Coste por hora de personal. Programadores a precio de mecánicos. 29 de junio de 2005

[6] Dataquest 2005

[7] TMap@NEXT (EN), T. Koomen e.a., 2006, Sogeti Nederland BV, página 26

[8] Kritieke software en maatschappelijke verantwoordelijkheid: De Therac-25 ongelukken; Walter Hop; TU Delft, Faculteit Elektrotechniek, Wiskunde en Informatica; 11 de abril de 2006

[9] TMap@NEXT (NL), T. Koomen e.a., 2006, Sogeti Nederland BV, página 472

Sobre los autores:

Casper Prins y Martin Blokpoel son Senior Test Engineer y Service Manager, respectivamente, en la unidad Software Control & Testing de Sogeti España en Bilbao.