POR CUAUHTÉMOC VÉLEZ MARTÍNEZ

Cápsula TI NO. 30

ERRORES DE SOFTWARE

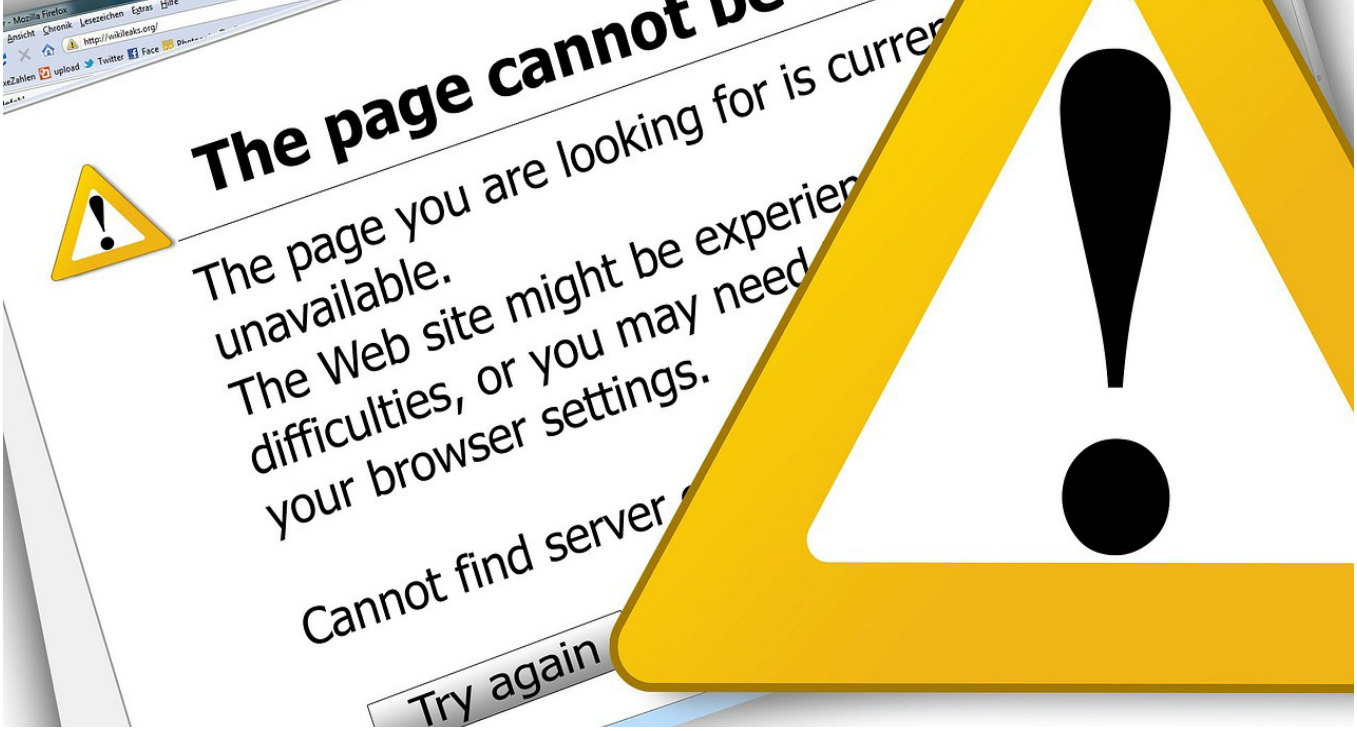
Es innegable la importancia que los sistemas de información juegan en la actualidad. La diversidad de aplicaciones es tan amplia que resulta poco probable encontrar áreas del quehacer humano donde no haya intervenido una computadora, por tanto, el *software.* Aunque tenemos la idea generalizada que el *software* es infalible, lo cierto es que los errores y fallas son mucho más frecuentes de lo que pensamos. En este sentido, el mal diseño de un sistema, la falta de estándares adecuados en el desarrollo de programas o las malas prácticas de programación, conllevan no solo fallas en los resultados de uno o más procesos sino millonarias pérdidas económicas, tecnológicas y hasta humanas.

Como es bien sabido en el ámbito de la ingeniería de *software*, los errores en un sistema de información pueden aparecer en distintas fases de su desarrollo: toma de requerimientos, análisis, diseño, programación, pruebas o implementación; así pues, entre más tarde se detecte la falla, más costoso será corregirla. Por ejemplo, un error de omisión de un campo en las últimas fases del desarrollo (la descripción de un producto, la clave de un empleado o las placas de un vehículo no incluida en la base de datos), representará darlo de alta en el diccionario de datos, modificar todos los reportes y pantallas de captura donde intervenga y reprogramar los módulos de algunos, o quizá, de todos los procesos. Lo anterior implica retrasos importantes en la entrega del sistema, aumento en costos y seguramente el enfrentamiento de múltiples errores colaterales. Por otra parte, los errores pueden ser tan sutiles como reemplazar un símbolo por otro (emplear el signo menos por el más en una operación) pero que origina graves errores en los resultados de un programa. De acuerdo a una encuesta realizada en 2015 por *Statista,* en el Reino Unido se reportaron problemas con los sistemas de información debido a tres causas fundamentales: fallas de *software* (23%), errores humanos (28%) y ataques informáticos (49%). Asimismo, una encuesta realizada en Estados Unidos durante ese año, mostró que 43% de las causas que originan la baja calidad en los datos de un sistema de información son por errores en el *software,* cifras muy elevadas si hablamos del costo que representa desarrollar un sistema más la infraestructura que gira a su alrededor.

Así pues, una falla de *software* por minúscula que sea, tiene implicaciones diversas; tal es el caso de la sonda espacial MCO (*Mars Climate Orbiter*) lanzada en 1998 y cuyo objetivo era rastrear el clima de Marte y transmitir los datos obtenidos a la Tierra; sin embargo, justo al realizar las maniobras de inserción a la órbita marciana, se perdió toda comunicación con el satélite. La junta de investigación de accidentes determinó que el origen de esta millonaria pérdida fue debido a que los datos generados por el programa que calculaba el rendimiento del propulsor estaban en unidades inglesas y que a su vez, los resultados de este proceso eran utilizados por otro módulo que determinaba la desaturación del movimiento angular, pero los requería en unidades MKS. Esta incompatibilidad de unidades originó pequeños errores traducidos en una trayectoria seguida por la nave 170 kms más baja que la planeada, dando como resultado la destrucción del orbitador en la atmósfera de Marte.

Asimismo, durante 2003, el noreste y medio oeste de Estados Unidos así como la provincia de Ontario en Canadá, sufrieron un apagón de enormes dimensiones que duró alrededor de 7 horas y en otros lugares hasta una semana, afectando a más de 55 millones de personas. Una de las causas que contribuyeron a este problema fue un error de *software.* Semanas después de haberse presentado el percance, los ingenieros de *General Electric* realizaron una auditoría a su sistema. Después de analizar varios millones de líneas de código, detectaron que un error en la programación provocó una falla en el sistema de alarma del centro de control de energía de la empresa *FirstEnergy* propiciando con ello el apagón.

En años más recientes los errores de *software* no han dejado de existir. Durante 2014 el Banco Real de Escocia fue multado con 56 millones de libras esterlinas luego que, por un error de *software*, fueran cerradas durante varias semanas las cuentas de sus más de 6.5 millones de clientes impidiendo realizar operaciones bancarias. Por otro lado en 2015, *Nissan* hizo un llamado a más de un millón de sus vehículos debido



a que identificó problemas de *software* vinculados con las bolsas de aire pues el programa no detectaba la presencia de un adulto ubicado en el asiento del acompañante. También en diciembre de ese año, el espacio aéreo londinense fue cerrado por un malfuncionamiento del *software* encargado de rastrear y planear el despegue y aterrizaje de aviones de uno de los aeropuertos más congestionados del mundo ocasionando que cientos de vuelos fueran cancelados, demorados o desviados.

Estos y muchos casos más, no significan que los errores sean una característica intrínseca en los sistemas de información, sino más bien, que los procesos de control y auditoría en las fases de desarrollo no se aplican o son inadecuadas. Si bien es cierto, la industria del *software* ha relajado considerablemente sus controles de calidad (es una de las razones por las que existen tantas actualizaciones), la realidad es que para muchas empresas, hacer programas sin errores no es rentable pues las pruebas y controles que deben aplicar representan costos y retrasos en la liberación de sus productos. Esto impide comprender las consecuencias de adquirir software defectuoso y después esperar a que el proveedor lo corrija, pero nadie en su sano juicio compraría una casa nueva sin cimientos o con paredes agrietadas aun si el vendedor se comprometiera a reparar las fallas. La diferencia es que el *software* es intangible y los errores no se ven sino hasta el momento en que el programa se ejecuta y se dan las condiciones que evidencian la falla manifestándose por lo general en bloqueos del programa, errores de datos o pérdidas de información.

Si bien es cierto ningún sistema de información es infalible, la tasa de errores puede disminuir considerablemente en la medida que se apliquen adecuadamente los procesos y controles señalados en la ingeniería de *software*; esto dará como resultado, sistemas de mejor calidad y altamente confiables.

Referencias

-NASA. (1999). Mars Climate Orbiter Mishap Investigation Board Phase I Report. NASA. Tomado de: ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf

-Poulsen. K. (2004). Software Bug Contributed to Blackout. SecurityFocus. Tomado de: http://www.securityfocus.com/news/8016

- Statista. (2015). Distribution of the root causes of data breach in the United Kingdom (UK) in 2015. Tomado de: http://www.statista.com/statistics/483047/root-cause-of-data-breach-uk/

- Statista. (2015). What are the main causes for poor data quality? Tomado de: http://www.statista.com/statistics/518069/north-america-survey-enterprise-poor-data-quality-reasons/

- Tricentis. (2015). Software Failures of 2015: Quarter One. Tricentis. Tomado de: http://www.tricentis.com/blog/2015/05/05/software-failures-of-2015-quarter-one/

Revisión técnica: Ing. Julio Alfonso de León Razo