Ponencia del IV Congreso Gallego de la Calidad. Santiago de Compostela, 7 de mayo de 2003.

CÓMO VERIFICAR LA CALIDAD DEL SOFTWARE EN SISTEMAS CRÍTICOS

José Carlos Sánchez Domínguez
Responsable de calidad
SoftWcare S.L

Patricia Rodríguez Dapena
Gerente
SoftWcare S.L.

RESUMEN

En cada vez más áreas de nuestra vida cotidiana se utilizan "sistemas" críticos con un alto contenido en software: automóviles, aviones, sistemas médicos, etc. Existen diferentes técnicas para el tratamiento de la robustez del software crítico que contribuyen a mejorar la seguridad[i] y fiabilidad del software aunque, desgraciadamente, no se utilizan mucho todavía: el estado del arte y su aplicabilidad están aún muy inmaduros, los clientes de estos sistemas no lo exigen, no hay unanimidad en qué método usar en cada caso, etc.

Esta ponencia se centra especialmente en la presentación de un método de "eliminación de fallos de software" y su influencia en el proceso del ciclo de vida de desarrollo del software. Además, se presentan ejemplos de fallos reales encontrados en diferentes sistemas críticos aplicando el método descrito.

INTRODUCCIÓN

La presencia de software en sistemas críticos es cada vez mayor (desfibriladores cardíacos, sistemas de control de aviónica, sistemas de control de las plantas de energía nuclear, sistemas antifrenado de los automóviles, etc.) convirtiéndose en un elemento básico en los sistemas actuales sobre cuya fiabilidad y seguridad descansa la vida de miles o quizás millones de personas.

Cajeros automáticos, sistemas basados en transacciones a través de internet, redes de comunicaciones, etc. son sistemas cuyos fallos, aun sin provocar la pérdida de vidas humanas, suponen un elevado perjuicio en nuestra vida cotidiana y cuya fiabilidad y seguridad también es importante.

Hay numerosos ejemplos de consecuencias catastróficas de estos fallos, que demuestran la importancia que el software está adquiriendo en nuestras vidas. Uno de ellos es el fallo del software de control de ferrys que ocasionó más de una docena de choques contra el muelle en el puerto de Seattle provocando unas pérdidas de más de 7 millones de dólares, aparte de los más de 3 millones de dólares empleados en volver al sistema manual.
No obstante, incluso en los sistemas más costosos, ampliamente probados y certificados de forma independiente pueden ocurrir fallos meses e incluso años después de su puesta en marcha. Los productos software en estas aplicaciones críticas ("safety critical applications") suelen ser grandes y complejos (con operaciones en tiempo real, algoritmos complejos, numerosas interacciones, etc.).

En la industria de la aviación, por ejemplo, cada dos años se duplica el software utilizado en los sistemas en vuelo. Con estas características no es posible tener una confianza plena en el sistema y debe aceptarse la posibilidad de que se produzcan errores en el software.

Esta imposibilidad de garantizar la ausencia total de errores en los sistemas seguirá vigente en la medida en que:

- Sean desarrollados como sistemas nuevos en vez de fomentar la reutilización de productos ya probados reduciendo así la utilidad de los históricos de fallos.
- No existan datos de fallos de los sistemas actuales pues normalmente no se hacen públicos obstruyendo así cualquier posible aprendizaje de la experiencia.
- Las técnicas de tolerancia a fallos de software no estén muy probadas.
- Los métodos existentes de evaluación y pruebas del software no ayuden adecuadamente a descubrir errores.

Sin embargo, existen diferentes técnicas para el tratamiento de la robustez del software crítico que contribuyen a mejorar la seguridad y fiabilidad del software. Desgraciadamente no se utilizan mucho pues el estado del arte de su puesta en práctica está aún inmaduro y su aplicación no se exige por los clientes.

Cada una de estas técnicas presenta importantes líneas de investigación. Esta ponencia se centra especialmente en una de ellas: la eliminación de fallos software y su influencia en el proceso del ciclo de vida de desarrollo del software, además de presentar experiencias prácticas de su utilidad.

TÉCNICAS DE ELIMINACIÓN DE FALLOS

Las técnicas de prevención, tolerancia, eliminación y previsión de fallos software[ii] contribuyen a la seguridad y fiabilidad del software. A las técnicas de prevención y eliminación de fallos software se les suele denominar como técnicas de evitación de fallos, aunque preferiremos la utilización del término “eliminación” en esta ponencia, considerando que la aplicación de las técnicas de eliminación en las fases más tempranas de un proyecto tiene un sentido igualmente preventivo.

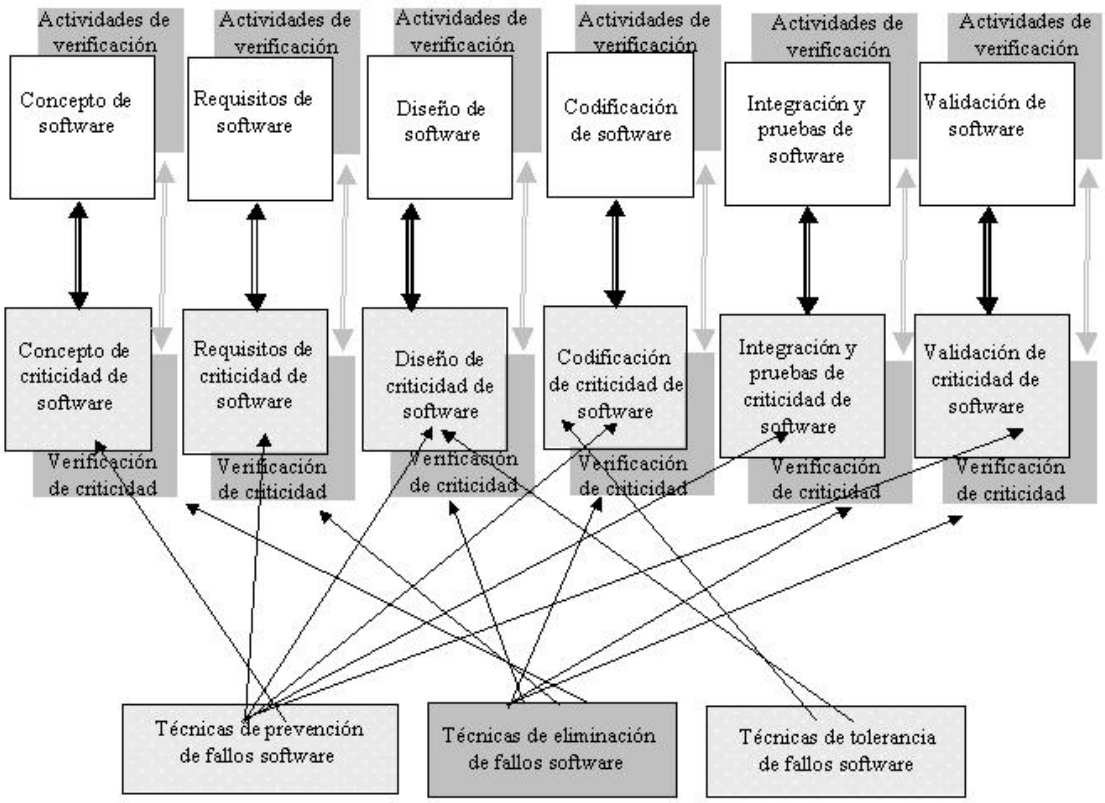

Figura 1. Técnicas de robustez de software

Las técnicas anteriores son igualmente importantes para el análisis de los sistemas críticos y por ello se exigen en diferentes estándares ([2], [3], [4]). Es cierto que los aspectos de seguridad en dominios como la aviónica o los sistemas nucleares no son nuevos. Existen estándares internacionales que regulan el proceso de desarrollo y evaluación de estos aspectos del software, pero sólo en limitadas ocasiones proponen técnicas prácticas y eficaces. La realidad es que las pocas técnicas que se mencionan o bien son muy generales, sin guías prácticas para su aplicación, o son las mismas técnicas que se utilizan para los subsistemas hardware, que no ahondan adecuadamente en las características específicas de los productos software haciéndolas así inaplicables (los fallos potenciales del software no son los mismos que los del hardware u otros mecanismos: el software no se desgasta, etc.; no existen datos estadísticos de fallos en productos software y muchas técnicas hardware se basan en ellos, etc.)

Para aplicar estas técnicas cuando se desarrolla un producto software, debe establecerse una relación entre ellas y (a) cada etapa de desarrollo, (b) los métodos y las técnicas utilizadas en el desarrollo de software y, (c) con las diferentes arquitecturas del producto.

EL MÉTODO SOFTCARE

El método SoftCare pretende identificar los fallos software que pudieran tener consecuencias graves en un sistema (pérdida del propio sistema o de vidas humanas, etc.), ayudando al aseguramiento de esta forma de la seguridad y fiabilidad del sistema, mediante su aplicación en las diferentes fases de desarrollo del producto.
La definición del método se basa en la idea de que a través de la comprobación sistemática y estática de los fallos potenciales del software durante el desarrollo de un producto de software crítico, es posible reducir drásticamente los riesgos de fallo del sistema debidos a problemas en el software, antes de su puesta en operación.

Las técnicas tradicionales para el análisis estático de la seguridad y la fiabilidad de los sistemas, tales como SFMEA[iii] (Análisis de los Modos de Fallo del Software y sus Efectos) y SFTA[iv] (Análisis del Árbol de Fallos del Software) ya se aplican a nivel sistema paralelamente a otras técnicas dinámicas, pero no a nivel de los subsistemas software en sistemas con un alto contenido en software.

Su mayor ventaja, cuando se aplica para analizar un producto software, se obtiene cuando se utilizan conjuntamente: SFMEA se centra en la identificación de la severidad y la criticidad de los fallos y SFTA en identificar las causas de estos fallos, y en orden inverso a su utilización normal a nivel sistema.

La elección de estas técnicas viene determinada por los siguientes factores:

- Su uso desde las etapas iniciales del proyecto, ayudando a descubrir y eliminar fallos potenciales con una mayor anticipación.
- Su integración con las demostraciones de fiabilidad a nivel de sistema.
- Ambas cuentan con una gran aceptación como técnicas de análisis de seguridad y fiabilidad, como demuestra su inclusión en los estándares.
- No requieren de una infraestructura especial para su aplicación.

Con todo, SFMEA y SFTA no son una solución infalible, pero pueden aplicarse de forma sencilla en el software, con ciertas adaptaciones que deberán ser consideradas de forma específica para cada etapa de desarrollo del software en las que se apliquen.

La combinación de las técnicas SFMEA Y SFTA, así como las adaptaciones específicas necesarias para el software, definen el método SoftCare, donde:

- El análisis SFMEA identifica los modos de fallo funcionales, analizando sus efectos e identificando las causas potenciales y,
- El análisis SFTA identifica en detalle los fallos de software causantes de los modos de fallo.

Tras la aplicación de SFMEA y SFTA debe realizarse una evaluación de resultados que incluya las recomendaciones necesarias para eliminar los fallos del software que pudieran ser potencialmente desastrosos para el sistema.

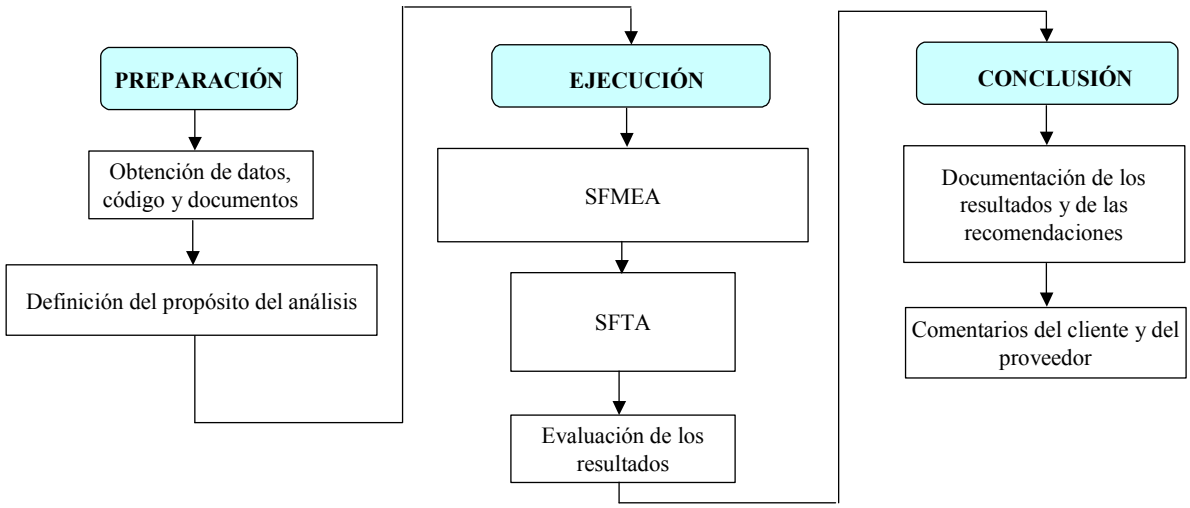

Figura 2. Método SoftCare

A continuación se introducirán brevemente las diferentes etapas o actividades para la utilización del método SoftCare.

Preparación del método

Inicialmente, la ejecución del método SoftCare como la de cualquier otra técnica de análisis requiere una recolección previa de información y una definición del alcance del análisis.

Recolección de información.

En esta etapa deberán considerarse diferentes aspectos incluyendo tanto la información disponible del producto software como los conocimientos requeridos por parte del evaluador.

- *Información disponible*. La información y documentación disponible para el análisis del producto debería ser tanto a nivel software como del sistema: requisitos, diseño, manuales de usuario, código, resultados de análisis previos del sistema total de criticidad y de actividades de verificación y validación. En función del alcance del análisis y de su profundidad, parte de la información anterior podría ser irrelevante o incluso no estar disponible.
- *Conocimientos requeridos*. Es importante que el equipo de análisis tenga un buen conocimiento del dominio del problema al que pertenece el producto software o la parte del producto que va a ser analizada.

Definición del alcance.

En esta etapa se pretende identificar y preparar los elementos que deben ser analizados. Para ello se requiere:

- *Identificar los elementos que deben ser analizados*, revisando cualquier posible condicionamiento que pudiera restringir el alcance del análisis a una parte del mismo.

- *Definir el nivel de profundidad del análisis*, determinando si se aplica a las etapas de identificación de los requisitos, al diseño o a la codificación (dependiendo de factores tales como el ciclo vida utilizado para el desarrollo del sistema, la funcionalidad que debe ser analizada, etc.)
- *Familiarizarse con el sistema*. Dada la complejidad que pueden tener los sistemas a analizar, el proceso de familiarización con el sistema podría requerir la obtención e incorporación de cierto conocimiento especializado del sistema en los resultados del análisis y, consecuentemente, conllevará un grado de esfuerzo a tener en cuenta en la planificación total del análisis.
- *Caracterizar el entorno desde el cual se obtendrá la información disponible para el análisis*. Esta caracterización consistirá en la definición de una serie de elementos a partir de los tres ejes existentes en un entorno de desarrollo: el proceso de desarrollo, la arquitectura y la tecnología empleada.

Ejecución del método

La ejecución del análisis de criticidad se realiza en tres pasos secuenciales.

Análisis de los Modos de Fallo del Software y sus efectos (SFMEA).

En primer lugar, debe realizarse el *Análisis de los Modos de Fallo del Software y sus efectos (SFMEA)*, determinando así los modos de fallo funcionales como punto de partida para un análisis posterior más profundo. Por modos de fallo se entienden las potenciales desviaciones del funcionamiento del sistema respecto a lo esperado. Por ejemplo, que una función del sistema se realice a destiempo, erróneamente o que no se realice.

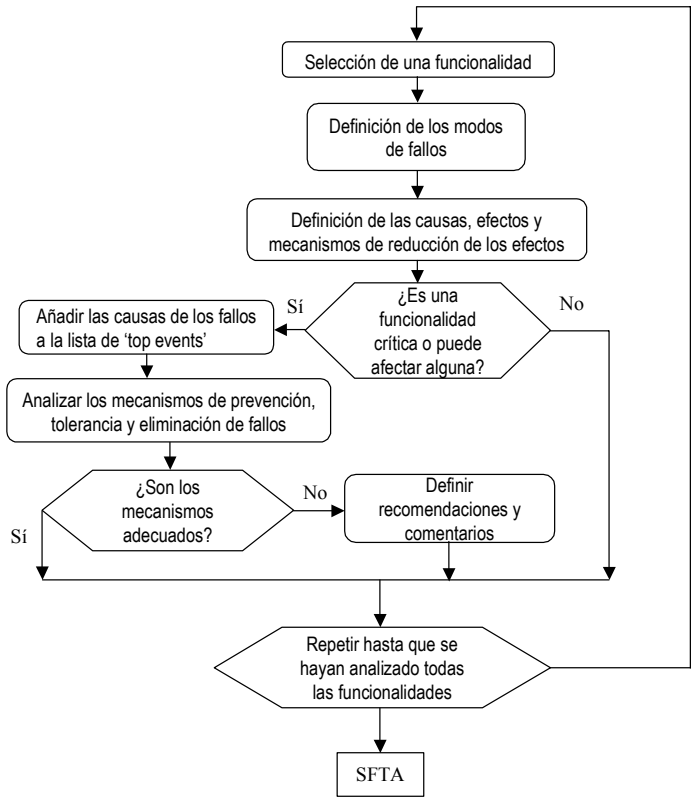

Figura 3. Procedimiento SFMEA

El SFMEA deberá utilizarse para identificar las funciones críticas a partir de la definición (o requisitos) del software proporcionado. El nivel de criticidad de los modos de fallo y, por tanto, su inclusión en un posterior análisis de criticidad, vendrá determinado tanto por la severidad en las consecuencias del modo de fallo como por los requisitos aplicables.

En esta etapa se elaborarán recomendaciones para la utilización de otras técnicas de prevención, eliminación o tolerancia de fallos que puedan reducir la criticidad de los modos de fallo.

Análisis del Árbol de Fallos del Software (SFTA)

Tras el SFMEA, deberá realizarse un *Análisis del Árbol de Fallos del Software (SFTA)*, que deberá identificar los fallos software que provocan estos modos de fallo. Se trata de una técnica descendente que determina el origen del fallo crítico, a partir de la información proporcionada por el diseño y descendiendo hasta los módulos de código.

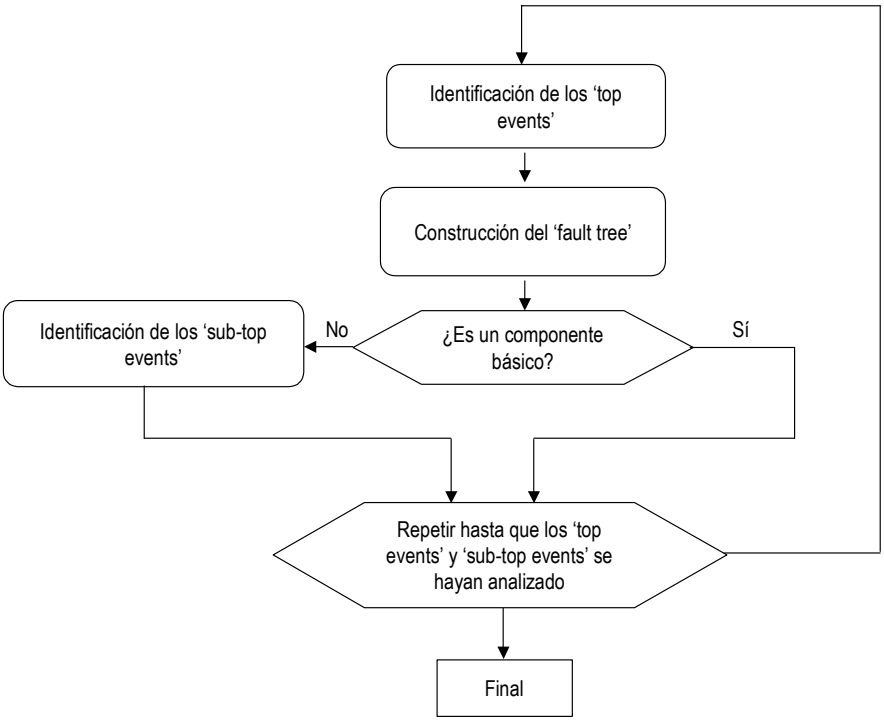

Figura 4. Procedimiento SFTA

El SFTA deberá confirmar la verdadera existencia de los modos de fallo anteriormente definidos (como una salida para el SFMEA) cuando se analice el diseño y el código (desde la fase de definición de los requisitos y hasta las fases de diseño e implementación), ayudando así a:

- Recomendar la corrección de errores en el diseño o en el código que causan el modo de fallo, o recomendar la utilización de técnicas de tolerancia a fallos para reducir el efecto del modo de fallo si no se puede evitar;
- Definir los casos de prueba que deberían incluirse en el conjunto de casos de prueba para la validación del software.

Para cualquier fallo encontrado que pudiera considerarse como causa de los correspondientes modos de fallo identificados en el SFMEA, deberá proporcionarse una recomendación para su eliminación y control.

La principal aportación del método SoftCare no es ya sólo el orden de aplicación de las técnicas SFMEA y SFTA, sino también las taxonomías de los modos de fallo y de los fallos software que las acompañan.

El uso de taxonomías para los diferentes modos de fallo y para los fallos software mejora notablemente el rendimiento de cualquier análisis de seguridad y fiabilidad, haciéndolo más sistemático, completo y objetivo.

La naturaleza del sistema o subsistema analizado supondrá diferentes tipos de fallos en el software. Así, por ejemplo, podríamos adoptar la siguiente taxonomía:

1. Los fallos hardware que producen un fallo del software son fallos físicos que tienen su origen en un dispositivo físico.
2. Los demás fallos tienen un origen humano y se pueden subdividir en fallos de diseño (incluyendo los de codificación), de interacción e intencionados.

Evaluación de resultados.

La evaluación de los resultados se realiza tras los dos pasos anteriores (los análisis SFMEA y SFTA) para destacar las discrepancias potenciales y preparar las medidas correctivas a modo de recomendación. Además, se podrán proporcionar recomendaciones a las pautas de diseño y codificación seguidas.

El SFTA podría resultar en un árbol de grandes dimensiones, siendo difícil su reducción manual. Estas reducciones podrían realizarse por medio de una herramienta. SoftCare evalúa el árbol SFTA, unificando ramas comunes del árbol. Se informará de todos los fallos del software que puedan causar uno de los modos de fallo identificados a nivel funcional, acompañándolos de una recomendación en cada caso para evitarlos o eliminarlos reduciendo su impacto en el sistema.

Conclusiones del análisis

Como conclusión del análisis realizado se elabora un informe en el que se documentan las actividades realizadas en el análisis, incluyendo como parte esencial del mismo tanto los fallos encontrados como las recomendaciones propuestas para su eliminación, considerando el nivel de criticidad en cada caso.

Asimismo, se recomienda siempre obtener la opinión del cliente (y del desarrollador del software, en su caso) respecto de los hallazgos del análisis y las recomendaciones que en él se proponen.

APLICACIÓN PRÁCTICA

Las características y la relevancia de los resultados obtenidos en la aplicación práctica del método varían según la etapa del ciclo de desarrollo del software en la que se aplica.

Ya se ha comentado anteriormente en la presentación del método que una de sus ventajas es la capacidad para anticipar resultados desde las etapas más tempranas del desarrollo del software. Así, la localización de los modos de fallo en los requisitos permite prevenir y evitar que estos se propaguen hacia el diseño y la codificación. La existencia de incoherencias, ambigüedades, formulaciones incorrectas o falta de información en los requisitos suelen ser causas de fallo habituales en esta etapa, y éstos son detectados por el método SoftCare. Así, por ejemplo, la utilización incorrecta de valores correspondientes a sistemas métricos diferentes hizo fracasar la misión del *Mars Climate Orbiter* de la NASA en 1999. Según el informe elaborado por la propia NASA, entre las causas de este fallo habría que destacar un 'inadecuado proceso de verificación y validación de los requisitos'.

La localización de fallos en el diseño permite igualmente anticipar errores en la construcción del software. En ocasiones es necesario llevar a cabo un proceso de conocimiento profundo del sistema que, por medio de la ingeniería inversa y con ayuda de la documentación existente, permita completar una arquitectura del diseño a alto nivel que no siempre está disponible. La localización de fallos en el diseño suele incluir requisitos que no fueron implementados en el diseño o que fueron implementados incorrectamente.

En 1996, el cohete Ariane 5 explotó 40 segundos después de su lanzamiento en su vuelo inaugural. Según el informe elaborado por la ESA en relación al accidente, una decisión incorrecta en el diseño del sistema contribuyó de forma importante al fracaso de la misión. Además, se indica que las pruebas realizadas no incluyeron un análisis adecuado del sistema que pudiera haber detectado el fallo potencial. La aplicación del método SoftCare al análisis del software de un satélite permitió identificar un error en el diseño del mismo, de forma que la no utilización de tareas ejecutándose concurrentemente impedía que el subsistema de control de errores del satélite cumpliera con las funciones previstas.

Finalmente, la revisión del código permite detectar las incoherencias entre este y el diseño y los requisitos, además de identificar errores sintácticos, falta de robustez, ausencia de legibilidad, etc. que pueden afectar igualmente a la disponibilidad del sistema.

Conviene destacar que las inconsistencias existentes entre los requisitos definidos para el sistema, la documentación y el código son hallazgos habituales en los análisis realizados.

La aplicación práctica del método SoftCare en sistemas de software críticos ha permitido obtener importantes conclusiones en la evaluación del software de abordo en automóviles o en satélites.

Así, por ejemplo, en el análisis de un producto software de abordo en automóviles se identificó un problema relacionado con el apagado automático del motor cuando se producían determinados fallos software. En el sector aeroespacial, se descubrió que en el envío de datos a un determinado satélite desde tierra, el envío del valor 0 en algunos casos provocaba el fallo general del software de control de abordo.

Alguno de los resultados proporcionados por el método SoftCare en estos análisis se obtuvieron en las etapas más tempranas de desarrollo del software (incluso en el diseño), con las lógicas ventajas que ello supone.

Sin embargo, la aplicación de estas técnicas al software todavía puede ser mejorada ya que se generan estructuras de análisis (tablas, árboles) muy grandes e inmanejables sin el uso de una herramienta CASE de soporte. Además, esta falta de automatización en un análisis de estas dimensiones no permite demostrar el 100% de completitud de análisis, o sea, que el producto evaluado pueda ser 100% fiable. Con todo, la literatura indica que este tipo de análisis encuentra 10 veces más fallos que utilizando otro tipo de pruebas.

Como resultado de la aplicación práctica del método se plantean una serie de mejoras, algunas de las cuales ya están siendo llevadas a cabo:

- La automatización del método en aquellas tareas más sistematizables permitiría una reducción del esfuerzo invertido en el manejo de grandes estructuras generadas por el método, facilitando el análisis. En este sentido, el desarrollo de la herramienta SoftCare está ayudando a mejorar este aspecto (proyecto TIC financiado por la Xunta).
- La utilización del método en combinación con otras técnicas, que ayudaría a mejorar la fiabilidad de los resultados obtenidos y observar el rendimiento proporcionado por el método en función de la técnica o técnicas utilizadas conjuntamente. Hasta el momento, se han realizado diversos análisis en combinación con técnicas dinámicas de inyección de fallos, análisis de trazabilidad, etc. proporcionando resultados de interés tanto para los clientes finales como para los propios participantes en la elaboración del análisis.
- Una mayor experiencia práctica del método permitirá obtener nuevas conclusiones de su aplicación en condiciones diferentes (entornos, equipos de trabajo, etc.). Aunque ya se cuenta con una experiencia práctica importante en este sentido, sería interesante la aplicación del método a una mayor variedad de proyectos, incluyendo productos de software no críticos.

Los estándares existentes hoy día no coinciden respecto a qué métodos se deben aplicar a la hora de desarrollar software como parte de sistemas críticos. Además, la utilidad final de algunos de estos métodos está aún por demostrar y preguntas como las siguientes están aún por responder: ¿Es eficiente imponer el 100% de cobertura en la fase de pruebas del software en vez de complementar estas pruebas con otras técnicas de análisis estáticos, dejándolas por ejemplo al 80%? [13] ¿Con qué otras técnicas, en otras fases del ciclo de desarrollo, se pueden complementar mejor las pruebas para la demostración de la seguridad del software dentro del sistema?

Lo que sí se puede asegurar es que el método SoftCare ayuda en gran medida a mejorar la seguridad y fiabilidad (y en definitiva, la robustez) del software en cualquier tipo de sistema.

CONCLUSIONES

El alto contenido de software de los sistemas críticos que utilizamos cada día crea la necesidad de definir cuál y cómo es la contribución del software en la seguridad (‘safety’) y fiabilidad de estos sistemas.

La aplicación de métodos de análisis estático del software desde las primeras fases de desarrollo del software, como complemento de las pruebas finales que nunca son al 100%, permiten asegurar y demostrar en mayor medida (pero aún no al 100%) el cumplimiento de las características de seguridad atribuidas al software en estos sistemas críticos.

Además, se sabe que la calidad del proceso de desarrollo también influye en la calidad del producto final, aunque aún no hay medidas cuantitativas que lo confirmen, por lo que las evaluaciones de los procesos de desarrollo de software también complementan estas demostraciones de seguridad.

Existen aún muchos aspectos que verificar acerca de esta contribución del software a la certificación de seguridad de estos sistemas críticos, siendo necesario abrir más líneas de investigación y desarrollo y poner en práctica realmente los métodos existentes para la consolidación de los mismos.

En cuanto a la experiencia práctica, la definición y, principalmente, la aplicación de estas técnicas estáticas de análisis, conviene destacar que han proporcionado importantes resultados a la industria del software crítico cuando se han llevado a cabo, como complemento a las técnicas tradicionales de prueba.

En su aplicación, el método ha proporcionado resultados fundamentales en periodos reducidos de tiempo y con un bajo coste que, de no haber sido identificados hubieran conducido a los sistemas evaluados a situaciones catastróficas.

Esta contribución ha sido reconocida por todos los participantes en los análisis y, especialmente, por parte de los responsables de los sistemas críticos evaluados. Probablemente, la mayor contribución al método sea la confianza que en él han depositado algunos clientes de gran relevancia en la industria europea del software crítico.

REFERENCIAS

[01] Software Safety Verification in Critical Software Intensive Systems.

Rodriguez-Dapena, P. Technische Universiteit Eindhoven, 2002.

[02] ECSS (European Cooperation for Space Standardisation) series of standards (http://www.ecss.nl/).

[03] DO-178B/ED-12B Software Considerations in Airborne Systems and Equipment Certification, RTCA, EUROCAE, December 1992.

[04] IEC 61508 – Functional safety: safety-related systems. Parts 1-7. IEC 1999.

[05] Defence Standard 00-55 (PART 1 and 2) / Issue 2.

Requirements for safety related software defence equipment. UK MoD. 1/08/97 (http://www.dstan.mod.uk/).

[06] Defence Standard 00-56 (PART 1) / Issue 2.

Safety management requirements for defence systems Part 1: Requirements. UK MoD, 13 December 1996. (http://www.dstan.mod.uk/)

[07] IEC 60300. Dependability management (parts 1 to 3). IEC 1997.

[08] Software Safety. NASA-STD-8719.13A NASA Technical Standard.

September 15, 1997 Replaces NSS 1740.13 dated February 1996

(http://satc.gsfc.nasa.gov/assure/nssstd.html)

[09] EN50128 Railway Applications: The specification and demonstration of dependability, re-liability, availability, maintainability and safety (RAMS). CENELEC.

[10] National Aeronautics and Space Administration (NASA). Mars Climate Orbiter Mishap Investigation Board. Phase I Report. November 10, 1999.

[11] European Space Agency (ESA). Ariane 501 Inquiry Board Report (July 1996).

[12] Computer Related Risks. P. G. Neumann (http://catless.ncl.ac.uk/Risks).

[13] Streamlining Software Aspects of Certification. Workshop I, II and III.

SSAC Technical Program Manager. NASA Langley Research Center. (http://shemesh.larc.nasa.gov/ssac/)

[i] El término seguridad se referirá normalmente al término en inglés “safety”, en contraposición a “security”.

[ii] En la literatura, a estas técnicas se les denomina habitualmente en inglés como “fault prevention” (prevención de fallos), “fault tolerance” (tolerancia de fallos), “fault removal” (eliminación de fallos) y “fault forecasting” (previsión de fallos).

[iii] Software Failure Modes and Effects Analysis.

[iv] Software Fault Tree Analysis