Seguridad en el *Software* Aeronáutico

Autor: Coronel Fernando Aguirre Estévez, Dirección de Ingeniería del Mando del Apoyo Logístico del Ejército del Aire.

Palabras clave: certificación, crítico, fallo, fiabilidad, peligro, riesgo, *safety*, seguridad, *software*.

Metas Tecnológicas relacionadas: MT 3.1.1; MT 7.1.1; MT 7.1.2; MT 7.3.1.

Introducción

Recientemente, el Departamento de Defensa Norteamericano reconoció 873 deficiencias de *software* sin resolver en el caza F-35, de las cuales 13 son de clase 1 (consecuencias operacionales en la plataforma). Los resultados son desalentadores, teniendo en cuenta que en el anterior informe el número de deficiencias identificadas fueron 917 [1]; si bien no debe olvidarse que el F-35 es un avión construido alrededor de un grupo de ordenadores extraordinariamente complejos, donde el número de líneas de código ya se cifran en veinticuatro millones en las versiones más avanzadas del caza.

El IEEE (*Institute of Electrical and Electronics Engineers*) define *software* como "un conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación" [2]. Así, el vocablo *software* abarca no solo al ejecutado por los ordenadores donde esté instalado, sino que también alcanza a los datos que necesitan estos programas, al sistema operativo y a los interfaces que interaccionan con el resto de los equipos. La importancia del *software* en las aeronaves es capital, habida cuenta de que un fallo de un programa puede ir desde insignificante, que no afecta a las prestaciones de la plataforma y sus sistemas, hasta el fallo de componentes críticos. Es por ello que se requiere aplicar unos procedimientos de verificación y validación del *software* que garantice que no existe riesgo de que se produzcan condiciones de fallo inseguras.

Aprobación del *software*

Atendiendo a lo indicado en el Reglamento de Aeronavegabilidad de la Defensa (RAD), el Certificado de Tipo hace constar que una aeronave y sus sistemas embarcados han sido diseñados y ensayados siguiendo las normas y procedimientos aprobados y que, por tanto, se considera segura para el vuelo [3]. La aprobación del *software* embarcado forma parte del proceso de certificación de la aeronave y sus sistemas, que busca demostrar que se cumple con los requisitos mínimos que garantice la seguridad en vuelo.

La criticidad de los cometidos encomendados a un paquete *software* será función de la severidad que el fallo pudiera ocasionar. Para establecer una clasificación relativa a dicha criticidad, la FAA (*Federal Aviation Administration*) ha adoptado el estándar DO-178 "*Software Considerations in Airborne Systems and Equipment Certification*" [4] para sistemas embarcados, desarrollado en colaboración con la EUROCAE (*European Organisation for Civil Aviation Equipment*), que en Europa se ha designado ED-12. Este estándar define cinco niveles de seguridad DAL (*Design Assurance Level*), en función de las posibles consecuencias del malfuncionamiento del *software*, graduado desde sin efectos (*No Safety Effect*) hasta catastrófico. Dependiendo del DAL, el programador/desarrollador debe cumplir un conjunto de requerimientos, que el estándar denomina objetivos, y lograr cada objetivo requiere, a su vez, implementar un conjunto de actividades o subobjetivos.

El DO/178/ED/12 permite realizar la verificación y validación del *software* aeronáutico, agilizando los trámites de certificación ante la Autoridad Aeronáutica [5]. Para ello, este documento proporciona una guía a seguir en el ciclo de vida de un paquete *software* con el objeto de verificar su integridad y calidad, alcanzando una mayor portabilidad y reusabilidad a menores costes, donde el fin último es garantizar que cada línea de código esté libre de errores y que los procesos de codificación y ensamblado de este código no adicionen *software* corrupto.

Asimismo, la modificación de un producto software después de su entrega es un procedimiento común de mantenimiento, debido a la necesidad de corregir defectos, mejorar el rendimiento u otras propiedades deseables, adaptarlo a un entorno distinto, y/o incorporar nuevas capacidades. Según lo indicado por el RAD [3], mientras que las modificaciones menores en aeronaves y sus sistemas deben ser aprobadas por el Órgano Técnico Competente, las modificaciones mayores, diseñadas y ensayadas por una organización distinta del titular del Certificado de Tipo, requieren de un Certificado de Tipo Suplementario que demuestre el cumplimiento de los requisitos y procedimientos aplicables aprobados. Por su parte, el Ejército del Aire instituyó la Instrucción General 70-12 como documento normalizador del Ciclo de Modificaciones de un Sistema de Armas, especificando las fases de diseño y desarrollo, así como las competencias de cada entidad involucrada.

El DO-178/ED-12 no solo puede aplicarse al *software* desarrollado y certificado previamente con este mismo estándar, sino que también puede usarse para modificaciones de *software* desarrollado con anterioridad bajo otro estándar y que deba ser certificado de nuevo, ya sea bajo el mismo entorno de desarrollo u otro diferente. Además, este estándar también se ha empleado en modificaciones

DAL	CONDICIÓN DE FALLO	PROBABILIDAD DE FALLO
A	Catastrophic	menor de 10^{-9} por hora de vuelo
B	Hazardous/Severe-Major	entre 10^{-7} y 10^{-9} por hora de vuelo
C	Major	entre 10^{-5} y 10^{-7} por hora de vuelo
D	Minor	mayor que 10^{-5} por hora de vuelo
E	No Effect	ninguna

Fig. 1. Categorización de la condición de fallo en función del DAL del *software*. (Fuente: DO-178/ED-12).

PROCESOS DEL DESARROLLO	DAL				
	A	B	C	D	E
Software Planning Process	7	7	7	2	0
Software Development Process	7	7	7	7	0
Verification of Outputs of Software Requirements Process	7	7	6	3	0
Verification of Outputs of Software Design Process	13	13	9	1	0
Verification of Outputs of Software Coding & Integration Processes	7	7	6	0	0
Testing of Outputs of Integration Processes	5	5	5	3	0
Verification of Verification Process Results	8	7	6	1	0
Software Configuration Management Process	6	6	6	6	0
Software Quality Assurance Process	3	3	2	2	0
Certification Liaison Process	3	3	3	3	0
TOTAL	66	65	57	28	0

Fig. 2. Número de objetivos a cumplir en cada proceso en función del DAL. (Fuente: DO-178/ED-12).

de *software* embebido en productos COTS (*Commercial-Off-The-Shelf*).

Software crítico

La utilización del *software*, particularmente en sistemas críticos, se ha multiplicado en los últimos años [5], volviéndose estos dispositivos cada vez más complejos, de tal modo que la incorporación de nuevas capacidades en los Sistemas de Armas pasa indefectiblemente, en mayor o menor medida, por cambios *software* que demandan análisis de riesgos específicos que eviten condiciones de fallo. Si a lo anterior se añade que estas modificaciones no solo afectan a la plataforma aérea sino también al elemento logístico, al *Health Monitoring System*, a la *Combat Cloud*, a la compatibilidad con el simulador de vuelo, etc., se configura un caldo de cultivo donde es preciso hacer uso de protocolos de seguridad *software*.

Un *software* que tiene encomendadas funciones críticas tiene que protegerse de peligros y/o fallos críticos, de tal modo que no causen daños importantes al sistema. Para ello es necesario disponer de dispositivos y/o mecanismos que mitiguen o controlen estos peligros o fallos, siendo preferible que estos mecanismos sean vía *software*. Para que este *software* crítico funcione correctamente se deben dar una serie de principios:

- El *software* no debe fallar al implementar las funciones críticas que tenga asignadas, de modo que conduzca al sistema a un estado peligroso e inesperado.
- El *software* no debe fallar en la detección o corrección del estado peligroso del sistema.
- El *software* no debe ser la causa de que tengan que actuar los mecanismos de mitigación.
- El *software* no debe fallar a la hora de implementar mecanismos de mitigación o reducción de efectos, en caso de que ocurra un estado peligroso.

La prevención de fallos intenta evitar que se introduzcan fallos en el sistema antes de que entre en funcionamiento, mientras que si el sistema continúa funcionando aunque se produzcan fallos es lo que se denomina tolerancia al fallo, siendo la redundancia una solución a este problema como factor mitigador. Qué duda cabe que al emplear componentes o *softwares* añadidos que detectan el fallo y recuperan el comportamiento correcto, se induce una mayor complejidad en el sistema y también se pueden producir fallos adicionales, dándose dos situaciones: en la redundancia estática, los elementos redundantes están siempre activos mientras que en la redundancia dinámica los elementos redundantes entran en funcionamiento al actuar los mecanismos de mitigación cuando se detecta un fallo.

De cara a abordar los análisis de riesgos, el Anexo 19 de la OACI (Organización de Aviación Civil Internacional) define el riesgo como "la probabilidad y la severidad previstas de las consecuencias o resultados de un peligro", mientras que para entender el concepto de peligro se puede acudir al *Safety Management International Collaboration Group,* que concibe el peligro como "la condición que puede causar o contribuir a un incidente o accidente de la aeronave". Asimismo, el IEEE define la fiabilidad como "la probabilidad de que un sistema desempeñe sus cometidos, bajo condiciones específicas, en periodos de tiempo determinados", la cual aumenta con la tolerancia al fallo. La probabilidad de fallo de un *software* depende esencialmente de dos factores: los errores en el código, propiamente dicho, y/o la entrada de valores erróneos para los cuales el sistema no tiene respuesta. Ciertamente, un *software* fiable y robusto sigue cumpliendo su misión a pesar de recibir variables erróneas. Finalmente, la seguridad (*safety*) es la probabilidad de que no ocurra ningún evento que provoque un fallo (ISO/IEC 15026), que aumenta con una adecuada prevención de fallos.

Una vez identificados los riesgos, el análisis de los mismos se encarga de su categorización, dependiendo de su severidad y probabilidad, para a continuación intentar evitarlos o, si no se puede, incluir medidas mitigadoras incorporando mayores redundancias entre otras opciones. Además, es necesario planificar y programar criterios de cancelación de la misión FTS (*Flight Terminate System*) en caso de que se materializase un fallo crucial. Para la gestión de riesgos son de aplicación diferentes procedimientos como el estándar MIL-STD-882E "*System Safety*", el STANAG 7160 "*Aviation Safety*" o el Doc 9859 AN/474 OACI "Manual de gestión de la seguridad operacional" [6], por citar los más relevantes.

Singular preeminencia cobran en estas circunstancias todos aquellos procesos de comprobación y análisis, simulaciones, tanto en bancos de pruebas como en vuelo, explorando los distintos modos de fallo así como las limitaciones inducidas y los criterios FTS. Estos procesos permitirán asegurar que el *software* se desarrolla de acuerdo a sus especificaciones y satisface los requisitos (verificación), desde un punto de vista teórico, mientras que en la validación se deben cumplir los requisitos del usuario en un análisis práctico.

Desarrollo *software*

El fin último de los procesos de desarrollo *software* es minimizar la probabilidad de fallos durante la creación del sistema. Así, es de destacar que tras la introducción del DO-178B en la década de los 90, no ha ocurrido un solo incidente letal que pueda ser atribuido a un fallo en el desarrollo del *software* certificado bajo este estándar.

La última versión DO-178C mantiene los principios establecidos por sus versiones anteriores, DO-178A y DO-178B. Así, aunque el DO-178C tiene muchos cambios menores respecto al DO-178B, estos son en su mayoría aclaratorios. De hecho, el *software* existente que ha sido aprobado previamente bajo el DO-178B también puede aprobarse bajo el DO-178C [7].

Dado que es bastante difícil probar la completa ausencia de errores de *software*, el principal objetivo del DO-178/ED-12 es demostrar la calidad del proceso de desarrollo desde los comienzos hasta el final, teniendo siempre presente la necesidad de minimizar la creación de errores. Así, la filosofía de los estándares DO-178/ED-12 requieren que se realicen una gran cantidad de pruebas de *software* basadas en requisitos, análisis de seguridad del sistema, análisis de *software*, revisiones de *software* y pruebas formales, las cuales se emplean en soportar y dar confianza en todo el proceso de desarrollo.

Software Development Plan

El *Software Development Plan* (SDP) concreta los lenguajes de programación utilizados, el estándar de codificación, los métodos de pruebas, las herramientas de *debugging,* los procedimientos de desarrollo y diseño del *software*, así como el *hardware* usado en el desarrollo y ejecución del *software*. El SDP tiene como principal objetivo el limitar los errores introducidos y, en segundo lugar, detectar aquellos errores que pudieran adicionarse mediante métodos de verificación. Para ello, es necesario disponer de compiladores, linkadores, además de herramientas para verificación y validación. El *Software Development Plan* contiene cronogramas de eventos y entregables, debiendo someterse a revisiones periódicas que tengan

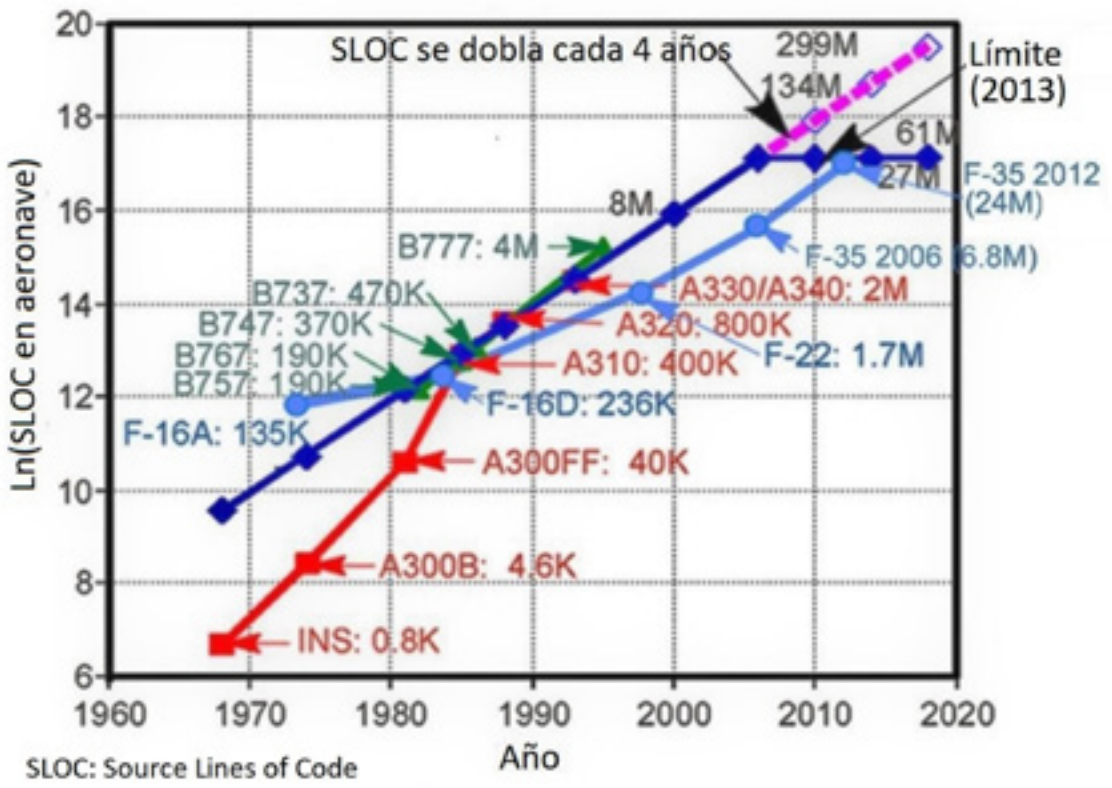

Fig. 3. Crecimiento del número de líneas de código *software* en aeronave (Fuente: Boeing Airbus).

Fig. 4. Gestión de riesgos de seguridad (Fuente: Doc 9859 AN/474 OACI).

en cuenta modificaciones y desviaciones, contemplando el ciclo de vida en el desarrollo del *software*.

Software Development Process

A partir de las especificaciones iniciales se generan los requisitos de alto nivel, que deben ser desarrollados para generar los requisitos de usuario, los cuales tienen que ser traducidos a requisitos *software*. El proceso de desarrollo *software* convierte estos requisitos en arquitectura *software* que, mediante un proceso secuencial, son transformados en código. El conjunto de los requisitos *software* es lo que el DO-178C denomina requisitos del sistema, los cuales no solo incluyen aquello que debe hacer el *software* sino también otro tipo de requisitos, como la evaluación de seguridad del sistema o el rendimiento. La validación es el proceso final donde se determina que los requisitos de *software* son correctos, están libres de errores y están completos. El DO-178C no da una guía como tal para las pruebas de validación, ya que se parte de la base de que si la verificación del *software* es correcta, no deberían aparecer problemas de validación en las pruebas de integración y sistema. El *Software Development Process* considera el desarrollo *software* como un ciclo de vida que comienza con la planificación y desarrollo, de acuerdo a los requisitos *software*, continúa con la verificación y validación, finalizando con la implantación mediante la carga en flota y su posterior mantenimiento.

Software Verification Process

La verificación del *software* busca obtener una evidencia de que es correcto contra los requisitos. Así, este proceso consiste en revisiones de requisitos, revisiones de diseño, de estado, de arquitectura, de código, análisis y pruebas, que van desde la implementación del *software* hasta la entrada de datos, pasando por la inyección de fallos controlados que identifican fuentes de errores y/o estadística de probabilidades de eventos, examinando la trazabilidad de las salidas de los procesos.

Especial relevancia cobran en este proceso las pruebas de no regresión que garantizan que, al modificar un *software*, no se introducen errores adicionales y continua cumpliendo sus funcionalidades iniciales.

Certificación del software

Mediante la certificación del *software*, los programadores/desarrolladores dan evidencia a la Autoridad Aeronáutica de que es seguro. Técnicamente, la certificación se refiere a la evaluación de la conformidad que asegura que un producto, proceso o sistema de gestión cumple unos requisitos específicos.

De acuerdo al estándar DO-178/ED-12, el proceso de certificación se alcanza cuando se demuestra ante la Autoridad Aeronáutica el cumplimiento del *Plan for Software Aspect of Certification* (PSAC), plan que ha sido previamente aprobado por dicha Autoridad. Asimismo, el *Software Accomplishment Summary* (SAS) da evidencias que demuestran el cumplimiento del PSAC, describiendo el sistema de forma general en cuanto a arquitectura, funciones y características de seguridad, entre otras.

El PSAC enlaza con los cuatro planes restantes que exige el DO-178/ED-12: el SDP, ya visto anteriormente, el *Quality Assurance Plan* (QAP), el *Configuration Management Plan* (CMP) y el *Software Verification Plan* (SVP). El QAP especifica cómo se llevaran a cabo las auditorías del *software* y de los procesos que se ven involucrados. El CMP define el control de configuración del *software*, el control de revisiones, la trazabilidad y los informes requeridos. El SVP es similar al SDP pero concretando el proceso de revisión (interno, externo, por pares,…), actividades de verificación del *software*, de requisitos, de diseño,

tipos de herramientas para verificación, revisión de procedimientos de prueba, etc.

Consorcio FACE

La carga de trabajo que suponen los protocolos de seguridad *software* junto al imparable aumento en el empleo del mismo ha provocado tal incremento de costes que, si se continúa con la actual tendencia, la futura generación de Sistemas de Armas no podrá abordarse con los presupuestos actuales. Por ello, los Estados Unidos han tomado cartas en el asunto en un esfuerzo de estandarización y normalización en diversas áreas, lanzando en 2010 el *Consorcio Open Group* FACE (*Future Airborne Capability Environment*) [8] al objeto de definir un entorno de programación de *software* abierto para todo tipo de plataformas militares aerotransportadas. Este entorno de aplicación funcional y compartible, busca disminuir costos y tiempos, a la par que intensificar la seguridad, promoviendo la reutilización, portabilidad, modularidad e interoperabilidad del *software* mediante la utilización de principios de diseño y la definición de una arquitectura común de referencia. Esponsorizado por el US *Army*, la US *Navy* y la USAF (*United States Air Forces*), así como las compañías Boeing, Lockheed Martin y Collins, actualmente el FACE es un grupo de proveedores de la industria, clientes, centros y usuarios, con una amplia trayectoria en el desarrollo *software*, cuyo número sigue creciendo, acercándose al centenar [9].

El enfoque adoptado por el FACE es una estrategia nacional, industrial y comercial estadounidense, para dotarse de sistemas *software* a costes asequibles, que promueven la innovación e integración rápida de capacidades entre distintos programas. El Consorcio FACE proporciona un foro de intercambio, donde industria y gobierno trabajan conjuntamente para el desarrollo y consolidación de estándares *software* abiertos, guías de trabajo, estrategias de innovación, entornos de desarrollo, herramientas de verificación y validación, etc., con el objetivo de patrocinar y fomentar:

- la portabilidad de aplicaciones a través de múltiples proveedores adheridos al consorcio FACE
- los estándares que proporcionen una arquitectura robusta de *software* de calidad
- el empleo de interfaces que permitan la reutilización de capacidades
- un amplio catálogo de aplicaciones para su uso en todo el espectro de sistemas a través de un entorno operativo común
- el dotarse de mayores capacidades que lleguen al cliente más rápidamente
- las aproximaciones a estándares abiertos dentro de los diferentes sistemas de aviónica
- la adquisición de productos del estándar FACE
- la disminución de costes de los sistemas FACE
- la innovación y competencia dentro de la industria de *software* de aviónica.

Para cumplir con los objetivos, el FACE parte de un estándar técnico que emplea una arquitectura patrón de referencia, la cual evoluciona un desglose conceptual de funcionalidades, promoviendo la reutilización de paquetes *software*, compartiendo capacidades entre sistemas diversos. Esta arquitectura define interfaces estandarizados para permitir que los paquetes *software* desarrollados por compañías diferentes puedan instalarse en distintos sistemas. Los interfaces estandarizados siguen una arquitectura de datos homogénea que garantice la comunicación entre paquetes instalados en componentes diferenciados.

El origen del FACE parte de los programas de arquitectura abierta desarrollados por la NAVAIR (*US Naval Air Systems Command*) que surgieron en un intento de mejorar la interoperabilidad y la portabilidad del *software* de aviónica en las plataformas de aviación naval, al cual se unieron posteriormente el US *Army* y la USAF. El objetivo en sí no deja de ser ambicioso, reducir el ciclo típico de desarrollo de nuevas capacidades

Probabilidad del riesgo	Severidad del riesgo				
	Catastrófico A	Peligroso B	Mayor C	Menor D	Insignificante E
Frecuente 5	5A	5B	5C	5D	5E
Ocasional 4	4A	4B	4C	4D	4E
Remoto 3	3A	3B	3C	3D	3E
Improbable 2	2A	2B	2C	2D	2E
Extremadamente improbable 1	1A	1B	1C	1D	1E

Fig. 5. Matriz de riesgos (Fuente: Doc 9859 AN/474 OACI).