Instituto Universitario Aeronáutico

El desarrollo de *software* en Sistemas Críticos para la Seguridad

Marcelo Luis Persano

Tesis para optar por el título de Ingeniero en Sistemas del Instituto Universitario Aeronáutico

2015

INSTITUTO UNIVERSITARIO AERONÁUTICO

FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN

Trabajo Final de Grado

Carrera: Ingeniería de Sistemas

El desarrollo de *software* en Sistemas Críticos para la Seguridad

Buenos Aires, Argentina - 2015

Autor: Marcelo Luis Persano

Tutora: Mgter. Ing. Brenda Elizabeth Meloni

Dedicatoria

A mis padres, por darme la vida, a mi padre que partió a la espera del encuentro final en la gloria de Nuestro Señor y a mi madre, ejemplo cabal de grandeza y amor.

A mi Patria, por darme la oportunidad de estudiar una carrera universitaria, para ser más, y en la esperanza que el presente trabajo sea de provecho para la Nación.

A la institución Fuerza Aérea Argentina y a sus inmortales aviadores soldados Héroes de Malvinas.

A mi amada esposa, unidos en sagrado matrimonio para recorrer la aventura de vivir, y a nuestros maravillosos hijos, con cuya llegada Dios nos bendijo con la dicha de formar esta hermosa familia.

Agradecimientos

A todo el personal docente y administrativo del Instituto Universitario Aeronáutico, por su constante apoyo y ayuda.

A la Ing. Brenda Elizabeth Meloni y al Ing. Juan Giró, por su paciencia, dedicación y apoyo en la realización del presente trabajo.

ÍNDICE

CAPÍTULO 1 RESUMEN 6

CAPÍTULO 2 INTRODUCCIÓN 7

CAPÍTULO 3 LOS SISTEMAS CRÍTICOS PARA LA SEGURIDAD 10

SECCIÓN 3.1 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN INFRAESTRUCTURA 15

SECCIÓN 3.2 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN MEDICINA 22

SECCIÓN 3.3 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN ENERGÍA NUCLEAR 27

SECCIÓN 3.4 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN TRANSPORTE FERROVIARIO 33

SECCIÓN 3.5 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN TRANSPORTE AUTOMOTOR 38

SECCIÓN 3.6 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN AVIACIÓN 44

SECCIÓN 3.7 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN ACTIVIDADES ESPACIALES 50

SECCIÓN 3.8 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN DEFENSA 56

SECCIÓN 3.9 SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN INDUSTRIA 62

CAPÍTULO 4 EL DESARROLLO DE *SOFTWARE* EN LOS SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN AVIACIÓN 67

SECCIÓN 4.1 *DO-178B/DO-178C SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT CERTIFICATION* (CONSIDERACIONES DEL *SOFTWARE* EN SISTEMAS AÉREOS Y EN CERTIFICACIÓN DE EQUIPOS) 69

SECCIÓN 4.2 IDENTIFICACIÓN Y EVALUACIÓN DE LAS EXIGENCIAS ESPECÍFICAS Y SU IMPACTO EN UN PROYECTO DE DESARROLLO DE *SOFTWARE* 75

SECCIÓN 4.3 SOLUCIONES PARA LOS PROBLEMAS IDENTIFICADOS O MITIGACIÓN DE SUS CONSECUENCIAS 81

CAPÍTULO 5 EJEMPLOS DE APLICACIONES *SOFTWARE* EN SISTEMAS CRÍTICOS PARA LA SEGURIDAD EN AVIACIÓN 98

SECCIÓN 5.1 SISTEMA DE VISUALIZACIÓN DE AVIÓNICA EN *JETS* PRIVADOS 101
SECCIÓN 5.2 SISTEMA DE INSTRUMENTOS DE VUELO PARA *JETS* COMERCIALES 102
SECCIÓN 5.3 SISTEMA DE CONTROL DE FRENADO EN EL TREN DE ATERRIZAJE 103
SECCIÓN 5.4 UNIDAD DE REFERENCIA INERCIAL DE DATOS PARA EL *JET* COMERCIAL *AIRBUS A350 XWB* 104
SECCIÓN 5.5 *SOFTWARE* PARA AERONAVES MILITARES EN FUERZA AÉREA ARGENTINA. .. 105

SECCIÓN 5.6 HELICÓPTERO MILITAR *EUROCOPTER ARINC 653* 107
SECCIÓN 5.7 PROGRAMA DE MODERNIZACIÓN DEL AVIÓN DE COMBATE *EMBRAER AMX A-1* EN LA FUERZA AÉREA BRASILEÑA 108
SECCIÓN 5.8 SISTEMA DE REABASTECIMIENTO AÉREO *AIRBUS 330 MRTT* 109
SECCIÓN 5.9 SISTEMAS DE A BORDO EN AVIÓN DE COMBATE *EUROFIGHTER TYPHOON* 109
SECCIÓN 5.10 SISTEMAS DEL CAZA POLIVALENTE FURTIVO *LOCKHEED MARTIN F-35 LIGHTNING II* 110
SECCIÓN 5.11 GESTOR DE INTERFAZ Y RADIO CONTROL DEL SISTEMA DE GESTIÓN DE VUELO DE AERONAVE *C-130J SUPER HERCULES* 112
SECCIÓN 5.12 PANTALLA DEL SISTEMA AVANZADO DE INFORMACIÓN DE RADAR *TARDIS* PARA EL AVIÓN DE ATAQUE *PANAVIA TORNADO GR.4* DE LA *RAF* 113
SECCIÓN 5.13 SISTEMAS DEL AVIÓN DE TRANSPORTE Y DE REABASTECIMIENTO AÉREO *BOEING KC-767* 114

CAPÍTULO 6 CONTENIDOS A INCORPORAR EN LA CARRERA DE INGENIERÍA EN SISTEMAS DEL IUA 115

SECCIÓN 6.1 ASPECTOS ESTADÍSTICOS 115
SECCIÓN 6.2 ASPECTOS MATEMÁTICOS 115
Subsección 6.2.1 Métodos Formales 115
Subsección 6.2.2 Métodos Semi-Formales 117
SECCIÓN 6.3 ASPECTOS DE LA INGENIERÍA DE *SOFTWARE* 118
Subsección 6.3.1 Gestión de la Seguridad 118
Subsección 6.3.2 Gestión del Riesgo 118
Subsección 6.3.3 Gestión de la Calidad 119
Subsección 6.3.4 Gestión de las Pruebas 120
Subsección 6.3.5 Gestión de la Configuración 120

CAPÍTULO 7 LA CREACIÓN DE UNA RED DE DESARROLLADORES DE *SOFTWARE* (REDES) 121

SECCIÓN 7.1 EL PARADIGMA DEL *SOFTWARE* DE CÓDIGO ABIERTO (*OSS*) Y EL DESARROLLO COLABORATIVO 121
SECCIÓN 7.2 LA UTILIZACIÓN DE *OSS* Y SU APLICACIÓN EN LOS SISTEMAS CRÍTICOS PARA LA SEGURIDAD 122
SECCIÓN 7.3 LA RED DE DESARROLLADORES DE SOFTWARE (REDES) Y LOS SISTEMAS CRÍTICOS PARA LA SEGURIDAD 124

CAPÍTULO 8 CONCLUSIONES 126

APÉNDICE A LISTADO DE SÍMBOLOS Y CONVENCIONES 128

APÉNDICE B PALABRAS CLAVE 151

REFERENCIAS Y BIBLIOGRAFÍA 152

1. Resumen

Las numerosas actividades que frecuentemente dependen de los sistemas computarizados se han incrementado a través de los años. El *software*, que define a los componentes lógicos necesarios para la realización de las tareas específicas, se convirtió en un elemento fundamental para el desempeño eficaz de los sistemas. La inclusión de actividades calificadas como críticas para la seguridad ha planteado nuevos desafíos para el desarrollo de *software* en forma segura. Las consecuencias de un funcionamiento defectuoso del *software* en dichos sistemas pueden ser tan graves como para llegar a provocar la muerte o heridas a las personas, afectando a la seguridad y el bienestar público.

El presente trabajo de grado comprende una presentación de la temática de los Sistemas Críticos para la Seguridad en distintas actividades para luego profundizar en una de ellas, la Aviación, investigando y evaluando las exigencias específicas que establecen la serie de estándares internacionales *DO-178,* la problemática planteada a partir de su aplicación, su impacto en las distintas etapas y tareas de un proyecto de desarrollo de *software* y la postulación de diversas soluciones para los aspectos mencionados. Se describen además una serie de aplicaciones reales de estos conceptos en una diversidad de sistemas existentes en las áreas relacionadas con la Aviación.

The numerous activities which often rely on computerized systems have increased over the years. The software, which defines the necessary logical components to perform specific tasks, became a key element for the effective performance of the systems. The inclusion of activities classified as safety-critical has raised new challenges for secure software development. The consequences of software malfunction in such systems can be severe enough to eventually cause death or injury to persons, affecting public safety and welfare.

This graduate work includes a presentation of the theme of the Safety-Critical Systems in various activities and then delves into one of them, the Aviation sector, in order to investigate and evaluate the specific requirements that are established by the series of international standards DO-178, and its impact for the different stages and tasks of a software development project. The current applications of these concepts are described in a variety of existing systems, in the areas related to Aviation.

2. Introducción

Los Sistemas Críticos para la Seguridad, como muchas otras actividades modernas, dependen fundamentalmente de sus componentes computarizados. El *software* es un elemento clave de los mismos. En estos sistemas una falla o funcionamiento defectuoso puede provocar, entre otras cosas, la muerte o heridas graves a las personas, la pérdida o un daño considerable tanto en equipos o infraestructura como a la propiedad o daños graves al medio ambiente y a la salud pública.

El desarrollo de *software* para tales sistemas posee ciertas características en particular. La definición de estándares que guían a la ingeniería del *software* ha sido de importancia fundamental para poder ajustar el diseño a los requerimientos específicos acerca de la seguridad.

Este trabajo tiene como objetivos explicar las peculiaridades de los sistemas críticos para la seguridad y su relación con el desarrollo del *software*, analizando los desafíos y problemas que surgen durante dicha actividad y planteando las soluciones para resolver los mismos o mitigar las consecuencias que surgen de ellos.

Este trabajo tiene como **objetivo general** establecer las pautas y criterios para el desarrollo de *software* en Sistemas Críticos para la Seguridad, en particular en lo referido a sistemas aplicados en Aviación, a través del análisis de la serie de estándares *DO-178B/DO-178C* y los desafíos que surgen en la aplicación de los mismos, planteando soluciones que resuelvan o mitiguen los efectos de los distintos problemas, proponiendo los contenidos de una propuesta educativa para incorporar en la carrera de Ingeniería en Sistemas del IUA a fin de ampliar la disponibilidad de recursos humanos con los conocimientos en este tipo de sistemas y postulando la creación de una Red de Desarrolladores de *Software* (REDES) que permita complementar las tareas desarrolladas por las Organizaciones públicas y privadas en caso de una situación de urgente necesidad de recursos capacitados para el desarrollo de *software* en Sistemas Críticos para la Seguridad.

Los **objetivos particulares** de este trabajo son:

- Presentar y analizar los distintos tipos de Sistemas Críticos para la Seguridad y las distintas actividades en las que son utilizados, describiendo el uso y necesidad de estos sistemas en una sociedad moderna

- Presentar y evaluar los desafíos que surgen en un proyecto de desarrollo de *software*, a través del uso de los estándares *DO-178B/DO-178C*, planteando diversas actividades que solucionen o mitiguen los efectos de dichos aspectos

- Describir diversas aplicaciones reales de estos sistemas en aviónica, superficies de control, navegación, etc., tanto en el ámbito de la aviación civil como la militar

- Basado en las características identificadas, postular los contenidos a incorporar en el plan de estudios de la carrera de Ingeniería en Sistemas para cubrir los aspectos estadísticos, matemáticos y de ingeniería de *software*

- Proponer la creación de una Red de Desarrolladores de *Software* que permita ampliar la oferta de recursos humanos familiarizados con los aspectos que presenta el desarrollo de *software* en sistemas críticos para la seguridad y su aplicación en áreas estratégicas para la Nación y el futuro de una sociedad moderna

La tesis está dividida en seis capítulos (3 al 8).

En el capítulo 3, *"Los Sistemas Críticos para la Seguridad"*, investigamos los distintos tipos de Sistemas Críticos para la Seguridad y las distintas actividades en las que son utilizados, describiendo el uso y necesidad de estos sistemas en actividades relacionadas con la Infraestructura Pública, la Medicina, la generación de Energía Nuclear, el Transporte Ferroviario y Automotor, la Aviación, las Actividades Espaciales, la Defensa, la Industria y otras actividades en general.

En el capítulo 4, *"El desarrollo de software en los Sistemas Críticos para la Seguridad en Aviación"*, investigamos y evaluamos los requerimientos y desafíos que plantea uno de los estándares más ampliamente utilizado, la serie *DO-178*, para un proyecto de desarrollo de *software*, analizando su impacto y la problemática que plantea su aplicación y postulando la solución o mitigación de dichos aspectos para las exigencias que demanda un proyecto de desarrollo de *software* de estas características.

En el capítulo 5, *"Ejemplos de aplicaciones software en Sistemas Críticos para la Seguridad en Aviación"*, describiremos aplicaciones reales de estos Sistemas, en las distintas actividades relacionadas con la Aviación.

En el capítulo 6, *"Contenidos a incorporar en la carrera de Ingeniería en Sistemas del IUA"*, describiremos los elementos identificados como claves para ser incorporados en el

programa de estudios de la carrera de Ingeniería en Sistemas que permitan cubrir los aspectos relacionados con el desarrollo de *software* en Sistemas Críticos para la Seguridad.

En el capítulo 7, *"La creación de una Red de Desarrolladores de Software (REDES)"*, postularemos la creación de esta Red, que permitirá ampliar la oferta de recursos humanos familiarizados con el desarrollo de *software* en Sistemas Críticos para la Seguridad, aplicando estos conocimientos en áreas estratégicas para la Nación y el funcionamiento de una sociedad moderna.

Finalmente, en el capítulo 8, *"Conclusiones"*, resumiremos las conclusiones de este trabajo.

3. Los Sistemas Críticos para la Seguridad

"No permitiré que tu pie resbale; jamás duerme el que te cuida"
Salmos 121:3

Existen numerosas definiciones respecto a lo que se entiende por Sistemas Críticos para la Seguridad. En general, todas coinciden en considerar así a aquellos sistemas cuya falla podría resultar en la pérdida de vidas o en un daño significativo a la propiedad o al medioambiente. Podemos ver que las características importantes se refieren a las consecuencias de un fallo. En otras palabras, si la falla del sistema podría producir consecuencias que se consideran inaceptables tanto desde el punto de vista ético y legal como desde el punto de vista económico, entonces el sistema puede considerarse como crítico para la seguridad. Esencialmente, de acuerdo con John C. Knight (2002), "*…es un sistema del cual dependemos para nuestro bienestar…*" [1]

Un servicio de despacho de emergencias médicas, una estación generadora de energía eléctrica, un dispositivo para radioterapia o una cirugía robotizada, una planta de energía atómica, la dirección del tráfico ferroviario mediante señalización, un automóvil moderno, el control del tráfico aéreo, un reactor de combate moderno, un vehículo espacial, un proceso de fabricación industrial automatizado, son todos ejemplos de actividades que basan su funcionamiento en sistemas computarizados. Una falla en cualquiera de estos sistemas puede representar, en forma inmediata, un peligro significativo.

En 1999, el *US PITAC* (*United States President's Information Technology Advisory Committee*) mencionaba que "*…nos hemos vuelto dependientes peligrosamente de sistemas de software grandes cuyo comportamiento no es bien entendido y los cuales a menudo fallan en formas impredecibles…La Nación depende de software que, frecuentemente, es frágil, poco fiable y que es extremadamente difícil y laborioso de desarrollar, probar y evolucionar…Los sistemas basados en software se encuentra ahora entre las más complejas estructuras de ingeniería creadas por el hombre…Debemos comprometernos a desarrollar software que sea confiable, tolerante a las fallas, seguro, capaz de evolucionar, escalable, sencillo de mantener y rentable…*" [2]

Existen una gran cantidad de ejemplos en los que estas fallas, lamentablemente, generaron las consecuencias mencionadas:

- En 1962, la misión *Mariner I* de la *NASA* debió ser destruida poco después del lanzamiento, debido a un error en la trayectoria prevista provocado por un fallo en el *software* de guiado

- El primer vuelo de pruebas del cohete *Ariane 5* falló en 1996, provocando su auto-destrucción 37 segundos después del despegue, debido a una falla en el *software* de control

- El dispositivo de radioterapia *Therac-25* provocó entre los años 1985 a 1987 una serie de sobredosis masivas de radiación en pacientes, debido a un fallo en el *software* de control

- En 1999 las sondas espaciales *Mars Polar Lander* y *Mars Climate Orbiter* se perdieron, debido a sendos errores en el *software* de control de los motores en el descenso y en el sistema de guiado, respectivamente

- En 1992 el servicio de ambulancias de la ciudad de Londres, implementó un sistema computarizado para el despacho de las emergencias. El sistema presentó errores que provocaron numerosas demoras en el despacho de las ambulancias

- De acuerdo con la *Food and Drug Administration* (*FDA* Administración de Medicamentos y Alimentos de los Estados Unidos), los fallos de *software* y seguridad fueron la causa en el 24% de los dispositivos médicos retirados del mercado en 2011

- El 9 de Abril de 2014, el servicio de llamado de emergencias al 911 en los Estados Unidos dejó de funcionar durante 6 horas por un error de *software*, dejando sin servicio a 11 millones de personas

Podemos encontrar Sistemas Críticos para la Seguridad en actividades tales como Infraestructura, Medicina, Energía Nuclear, Transporte Ferroviario, Transporte Automotor, Aviación, Actividades Espaciales, Defensa e Industria, entre otras.

Antes de introducirnos en cada una de estas actividades en detalle y en la investigación de los estándares relacionados con el desarrollo de *software* para las actividades mencionadas, definiremos primeramente (ver Gráfico 1) los conceptos comunes al *software* de Sistemas Críticos para la Seguridad, que serán de utilidad para contextualizar los procedimientos que serán descriptos.

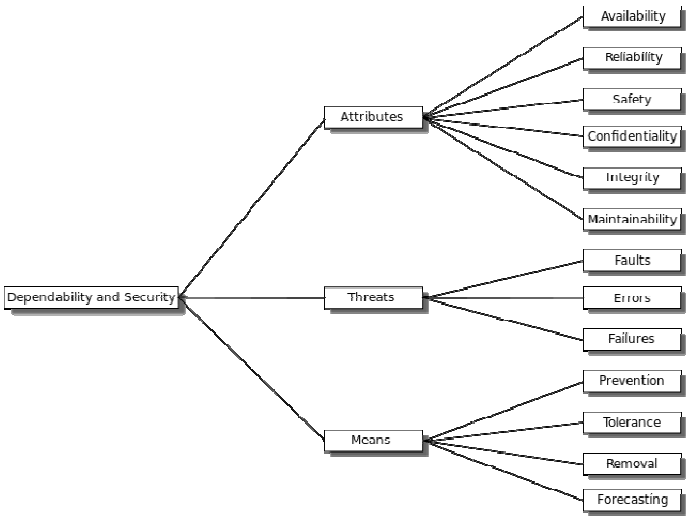


Gráfico 1: Taxonomía que describe la relación entre fiabilidad y seguridad, los atributos, las amenazas y los medios [3]

A la medida de la disponibilidad, la confiabilidad y la capacidad de mantenimiento de un sistema se la conoce, según la *IEC* (1990), como fiabilidad o seguridad de funcionamiento (*dependability*).[4] Entre sus atributos, es decir las cualidades del sistema, que pueden ser evaluados para determinar su fiabilidad general usando medidas cualitativas o cuantitativas, podemos mencionar, como establece J. C. Laprie (1992): [5]

- **Disponibilidad** (*Availability*): el Dr. Andrew J. Kornecki (2014) la define como la probabilidad de funcionar correctamente en cualquier momento dado [6]
- **Confiabilidad** (*Reliability*): es la probabilidad de funcionar correctamente durante un período de tiempo dado, Kornecki (2014) [7]
- **Seguridad** (*Safety*): A. Avizienis et al. (2004), la definen como la ausencia de consecuencias catastróficas para el usuario y el entorno. [8] Podemos agregar también que se trata de no lastimar a las personas ni causar daño a la propiedad, de acuerdo con Kornecki (2014) [9]
- **Confidencialidad** (*Confidentiality*): se trata de prevenir el acceso no autorizado a los datos, puntualiza Kornecki (2014) [10]

- **Integridad** (*Integrity*): A. Avizienis et al. (2004), la definen como la ausencia de alteraciones al sistema en forma incorrecta. [11] Y agrega Kornecki (2014), que se trata de prevenir la corrupción de los datos [12]

- **Facilidad de Mantenimiento** (*Maintainability*): A. Avizienis et al. (2004), la definen como la capacidad de un sistema para someterse a modificaciones y reparaciones. [13] Kornecki (2014), agrega que se trata de proporcionar un mantenimiento eficiente del sistema [14]

Entre las amenazas a la fiabilidad o seguridad de funcionamiento, es decir aquellas cosas que pueden afectar a un sistema y causar una disminución de su fiabilidad general, podemos mencionar, de acuerdo con A. Avizienis et al. (2004): [15]

- **Falla** (*Fault*): es un defecto en el sistema. Su presencia puede o no dar lugar a una avería

- **Error** (*Error*): es una discrepancia entre el comportamiento previsto de un sistema y su comportamiento real, dentro de los límites del sistema. Se trata de la manifestación interna de un fallo

- **Avería** (*Failure*): es un momento en el tiempo en el cual un sistema muestra un comportamiento que es contrario a su especificación. Se trata de la manifestación externa de una falla

Entre los medios que pueden utilizarse para incrementar la fiabilidad general del sistema, podemos mencionar, siguiendo a A. Avizienis et al. (2004): [16]

- **Prevención** (*Prevention*): la prevención de fallas se ocupa de evitar que una falla sea incorporada al sistema

- **Eliminación** (*Removal*): la eliminación de fallas puede darse durante el desarrollo o durante el uso del sistema. En el desarrollo requiere de una verificación para que las fallas puedan detectarse y eliminarse antes que el sistema entre en producción. Una vez que el sistema está en producción, se necesita llevar un registro de las fallas y eliminarlas a través de un ciclo de mantenimiento

- **Pronosticación** (*Forecasting*): la pronosticación de fallas predice las posibles fallas para que puedan ser eliminadas o para que sus efectos puedan ser evitados

- **Tolerancia** (*Tolerance*): la tolerancia a fallos trata del establecimiento de mecanismos que permitan que un sistema pueda continuar con sus actividades, aun ante la presencia de fallas, si bien el funcionamiento puede ser degradado hasta cierto nivel

Dentro de los requisitos para la seguridad de un sistema, podemos mencionar los siguientes conceptos, según Donald G. Firesmith (2004): [17]

- **Objetivo** (*Goal*): es una declaración de la importancia de lograr las metas deseadas respecto a algún factor de seguridad. Está por encima del nivel de una política y no se encuentra lo suficientemente formalizado como para ser verificable

- **Política** (*Policy*): es una decisión estratégica que establece un objetivo deseado, estableciendo un criterio de seguridad

- **Requisito** (*Requirement*): especifica una cantidad mínima y obligatoria para la seguridad, en términos de un criterio de calidad dado y un nivel mínimo de una métrica asociada, y que puede ser visible externamente, siendo verificable y validable. Algunas veces puede también especificarse respecto a una cantidad máxima y un nivel máximo de la métrica asociada

- **Restricción** (*Constraint*): es una decisión de ingeniería que especifica una salvaguardia específica y que ha sido seleccionada para ser impuesta como un requisito

Dentro de las situaciones que afectan a la seguridad, están los siguientes criterios, de acuerdo con Philip Koopman (2014): [18]

- **Peligro** (*Hazard*): es una situación que conlleva un peligro potencial para las personas, el medio ambiente o la propiedad

- **Incidente** (*Incident*): es una situación de peligro real para las personas, el medio ambiente o la propiedad. A veces un peligro da como resultado un incidente

- **Cuasi incidente** (*Near miss*): es una situación que, en otras circunstancias, habría sido un incidente

- **Accidente** (*Accident* o *mishap*): es un evento que causa la muerte o lesiones a las personas, daños al medio ambiente o a la propiedad. Un incidente puede convertirse en accidente

- **Riesgo** (*Risk*): es una combinación de la probabilidad de los riesgos y la gravedad de los resultados probables

La relación entre el *software* y la seguridad puede describirse a través de los siguientes conceptos inicialmente, como lo define Michael F. Siok (2013): [19]

- ***Software* Crítico para la Seguridad** (*Safety-Critical Software*): es una unidad de *software*, componente, objeto o sistema de *software* cuyo reconocimiento, control, rendimiento o tolerancia a fallas apropiada es esencial para el funcionamiento seguro y para el soporte del sistema en el cual se ejecuta

- **Funciones Críticas para la Seguridad** (*Safety-Critical Functions*): es una función o funciones integradas que se encuentran implementadas en el *software* y que contribuyen a comandar, controlar o monitorear funciones críticas para la seguridad a nivel del sistema que son necesarias para operar o soportar en forma segura al sistema en el cual se ejecutan

- **Seguridad del *Software*** (*Software Safety*): es la aplicación de las disciplinas de Ingeniería de Seguridad de Sistemas, Ingeniería de Sistemas e Ingeniería de *Software* para garantizar que se tomen medidas activas a fin de asegurar la integridad del sistema a través de la prevención, eliminación y/o el control de los riesgos que pueden ser causados o inducidos por el *software*

3.1 Sistemas Críticos para la Seguridad en Infraestructura

"El verdadero progreso es el que pone la tecnología al alcance de todos"
Henry Ford

En las décadas recientes, la Infraestructura considerada crítica se ha vuelto dependiente de aplicaciones de *software* complejas para poder llevar a cabo aquellas funciones que son vitales para una sociedad moderna, por ejemplo: la distribución de energía, la banca, el transporte, etc. Una falla en alguno de estos sistemas podría generar no sólo un daño material a la propiedad sino también tener un grave impacto en la salud y la seguridad pública, la economía y la seguridad nacional.

La definición de Infraestructura Crítica, tal como lo establece la *U. S. Public Law 107-296* (2002), incluye "*sistemas y activos, ya sean físicos o virtuales, los cuales son tan vitales para la Nación que la incapacidad o la destrucción de los mismos podría debilitar a la seguridad, a la economía nacional o a la salud pública, o a una combinación de todas ellas*". [20]

En los Estados Unidos, el Departamento de Seguridad Nacional (*Department of Homeland Security*) ha designado 18 sectores de la Infraestructura como críticos, de acuerdo con lo especificado por Jacob Olcott (2012): [21]

- Agricultura y Alimentación
- Banca y Finanzas
- Química
- Instalaciones Comerciales
- Comunicaciones
- Fabricación Crítica
- Represas
- Industria para la Defensa
- Servicios de Emergencia
- Energía
- Instalaciones Gubernamentales
- Salud Pública
- Tecnologías de la Información
- Monumentos Nacionales
- Reactores Nucleares
- Correo y Distribución

- Sistemas de Transporte
- Agua

El *National Research Council* (Consejo Superior de Investigaciones Científicas), a través del *Committee on Certifiably Dependable Software Systems* (Comité del *Software* Confiable y Certificable de los Sistemas), advirtiendo el riesgo inherente en la introducción de *software* que controle áreas críticas de la Infraestructura, menciona que "*La creciente omnipresencia y centralidad del software en nuestra Infraestructura cívica es probable que aumente la gravedad y la frecuencia de los accidentes que se pueden atribuir al software. Por otra parte, el riesgo de una catástrofe mayor en el que un fallo de software juegue un papel importante es cada vez más grande, debido a que el incremento de la complejidad y la invasividad de los sistemas de software no está siendo acompañado de mejoras en la fiabilidad*". También, estableció que "*se necesitan mejoras en el desarrollo de software para mantener el ritmo con las demandas de la sociedad. A menos que las mejoras se efectúen, el despliegue más generalizado de software en la Infraestructura cívica podría llevar a consecuencias catastróficas. El software tiene el potencial de traer enormes beneficios a la sociedad, pero no será posible alcanzarlos, especialmente en aplicaciones críticas, a menos que el software se convierta en más confiable*". [22]

En lo que existe un consenso generalizado entre los investigadores es que, sin dudas, el *software* para Sistemas Críticos tendrá un papel preponderante en la sociedad del futuro. De acuerdo con el *High Confidence Software and Systems Coordinating Group* (*HCSS CG*, Grupo de Coordinación para el *Software* y Sistemas de Alta Confiabilidad) del *U.S. National Coordination Office for Networking and Information Technology Research and Development* (*NITRD*, Oficina de Coordinación Nacional de los Estados Unidos para la Investigación y Desarrollo de las Redes y las Tecnologías de la Información) "*…estos sistemas, a menudo integrados en otros sistemas físicos y de tecnologías de información más grandes, son esenciales para el funcionamiento de la Infraestructura crítica del país, la aceleración de la capacidad de los Estados Unidos en cuanto a la competitividad industrial y la optimización de la calidad de vida de los ciudadanos…*" [23]

El *Committee on Certifiably Dependable Software Systems*, según lo mencionado por Daniel Jackson et al. (2007), sugiere como prioritario el focalizarse en las siguientes áreas para poder incrementar la confiabilidad del *software* [24]:

- **Pruebas (*testing*) como evidencia**: Cuando las pruebas se efectúan extensiva y sistemáticamente, son un elemento importante para la confiabilidad. Sin embargo, es difícil asignar un grado de confiabilidad en función de las pruebas realizadas, ya que una prueba exhaustiva de cada línea de código y estado del sistema es muchas veces imposible en la práctica y por otro lado la realización de un conjunto de pruebas seleccionadas podría ser insuficiente para detectar los errores que hayan quedado. Debe continuarse investigando en métodos que creen evidencias para poder asignar un grado de confiabilidad explícito a los sistemas

- **Comprobar el código en función de las propiedades específicas del dominio**: En los últimos años se han realizado avances importantes en las técnicas para la comprobación automática del código. Estas técnicas serán esenciales para el aumento de la confiabilidad, especialmente si son capaces de manejar propiedades del dominio específico en lugar de las propiedades locales del código solamente

- **Lenguajes y herramientas fuertes para la independencia de los argumentos**: El uso de lenguajes inseguros compromete a la modularidad la cual, de otra manera, podría ser un argumento de peso en la independencia de los argumentos. Por ejemplo, en un programa escrito en *C*, un acceso a un *array* fuera de los límites definidos puede sobrescribir estructuras de datos que no son accesibles por su nombre, por lo que uno no podría basarse en el uso de nombres para determinar como un módulo podría interactuar con otro. Es preciso seguir investigando para poder entender si el uso de lenguajes seguros u otras herramientas podrían justificar la independencia y ayudar a la confiabilidad de los argumentos y como construir esta confiabilidad en aquellos casos en que existen razones de peso para el uso de otros lenguajes considerados inseguros

- **Componer casos para la confiabilidad de los componentes**: Con el incremento en la reutilización de componentes y una mayor conciencia sobre los riesgos asociados (especialmente en el caso del uso de sistemas operativos estándares en funciones críticas), el nivel de aseguramiento del componente se convertirá en una actividad esencial, siendo necesario encontrar la manera de componer los argumentos de confiabilidad de los componentes en un argumento que aplique al sistema como un todo

- **Modelado y razonamiento acerca de los entornos**: La confiabilidad de un sistema se basa, generalmente, en suposiciones sobre el comportamiento de los operadores

del sistema y los dispositivos asociados dentro del entorno del sistema y, de una manera más amplia, en la organización humana en la cual el sistema es implementado. El incremento en la confiabilidad debería entonces involucrar el razonar acerca de las interacciones entre el sistema y su entorno

- **Razonando acerca de los sistemas a pruebas de fallas**: Las propiedades críticas de confiabilidad en la mayoría de los sistemas críticos tomarán la forma de "*X nunca debe suceder, pero si lo hace, entonces Y debe suceder*". Los sistemas a pruebas de falla se construyen pensando que ciertas fallas nunca ocurrirán, pero aun así están diseñados para fallar de un modo seguro en caso de que ello ocurra

- **Haciendo argumentos más fuertes de los más débiles**: Los argumentos más débiles pueden combinarse para formar un único argumento más fuerte. El aumento en la confiabilidad involucrará evidencias de clases diferentes, cada una contribuyendo con un cierto grado de confiabilidad a la confiabilidad del todo

También, siguiendo con Daniel Jackson et al. (2007), establece los siguientes postulados y recomendaciones [25]:

- **Se necesitan mejoras en el desarrollo de *software* para mantener el ritmo de las demandas sociales**: La calidad del *software* producido por la industria es muy variable, y existe una supervisión inadecuada en algunas áreas críticas

- **Se necesitan más datos sobre los fallos de *software* y la eficacia de los enfoques usados para su desarrollo**: Se debe prestar más atención a la contribución del *software* en los accidentes y se necesita contar con un repositorio de informes de accidentes, en los cuales el *software* esté involucrado, para poder analizar las tendencias y evaluar los métodos y las tecnologías utilizadas. Sin un esfuerzo concertado para obtener mejores datos, la inversión en investigación y desarrollo de *software* podría ser mal dirigida, permaneciendo las prácticas ineficaces, siendo un obstáculo para la adopción de métodos más eficaces

- **Para los desarrolladores y usuarios de *software*:**

    - **Aprovechar al máximo las tecnologías de desarrollo de *software* y los métodos formales que sean eficaces**: Una variedad de tecnologías modernas como lenguajes de programación seguros, análisis estático y

métodos formales, probablemente reducirán el costo y la dificultad para producir *software* confiable

- **Seguir los principios comprobados para el desarrollo de *software*:**

  - **Utilizando una perspectiva de sistemas**: Una perspectiva de sistemas debe adoptarse en la que la fiabilidad de *software* no es vista en términos de sus propiedades intrínseca (como la incidencia de los errores en el código) sino en términos del sistema en su conjunto, incluyendo las interacciones entre personas, procesos y la tecnología, abarcando el entorno tanto físico como organizativo del sistema. La ingeniería de *software* debe ser dirigida por una consideración de los riesgos y su mitigación, y deben utilizarse las técnicas de análisis y reducción de riesgos que se aplican en otros dominios en forma exitosa (como el análisis de riesgos)

  - **Aprovechar la simplicidad**: Para lograr la confiabilidad a un costo razonable, la sencillez debe convertirse en un objetivo clave, y los desarrolladores y clientes deben estar dispuestos a aceptar los compromisos que conlleva. Un crecimiento sin restricciones en la complejidad de la funcionalidad ofrecida es incompatible con la confiabilidad. La arquitectura del *software* debe reflejar la priorización de requisitos, idealmente de manera que las propiedades críticas se pueden establecer examinando sólo una pequeña parte del *software*, basándose en la independencia de los argumentos para poder dar cuenta de la falta de interferencia de los partes restantes

- **Construir un caso de confiabilidad para un sistema dado y su contexto: evidencia, claridad y experiencia**: Un sistema de *software* debe considerarse como confiable sólo si se presenta evidencia suficiente como para justificar el reclamo de confiabilidad. La evidencia debe tomar la forma de un caso de confiabilidad que explique por qué las propiedades críticas se sostienen e involucrará un razonamiento tanto sobre el código como sobre los supuestos del entorno. En la medida en que este razonamiento pueda ser apoyado por herramientas automatizadas, será más creíble

- **Exigir más transparencia, de modo que los clientes y usuarios puedan efectuar juicios más precisos sobre la confiabilidad**: Los clientes y usuarios podrán tomar mejores decisiones al elegir proveedores y productos sólo si las declaraciones de confiabilidad, los criterios utilizados y la evidencia es transparente

- **Utilizar pero no confiar únicamente en el proceso y las pruebas**: Las pruebas se convertirán en un componente esencial para la confiabilidad, aunque no será suficiente, ya que incluso los mayores conjuntos de pruebas no validan la totalidad del código como para proveer una evidencia de que el *software* es correcto ni tendrán un significancia estadística importante para los niveles de confiabilidad que se desean

- **Basar la certificación en la inspección y el análisis de la confiabilidad declarada y la evidencia suministrada en apoyo de esta**: Dado que las pruebas y el proceso no son suficientes en sí mismos como para asegurar la confiabilidad, se deberá suministrar la evidencia proporcionada por el análisis

- **Incluir consideraciones de seguridad en los casos de confiabilidad**: Las vulnerabilidades en cuanto a seguridad pueden socavar el logro de la confiabilidad deseada, por lo que es necesario tener en cuenta explícitamente los riesgos de seguridad que puedan comprometer otros aspectos, asegurándose que las certificaciones de seguridad sean apropiadas para resistir cualquier ataque. Nuevos certificados de seguridad son necesarios, a través de los supuestos de que un atacante del sistema puede descubrir nuevas vulnerabilidades que sean aprovechadas para efectuar otros ataques

- **Exigir la responsabilidad y explicitarlo**: Se debe explicitar quién es responsable, profesional y legalmente, por el incumplimiento en el logro de la confiabilidad declarada. Ningún *software* debería considerarse confiable si se suministra con una advertencia de que el fabricante no se compromete a garantizar el *software* contra fallos

- **Para las agencias y organizaciones que apoyan la educación e investigación en *software*:**

- **Poner mayor énfasis en la confiabilidad y en sus fundamentos en las escuelas secundarias y en la educación de pre-grado y de grado de los desarrolladores de *software***: Muchos desarrolladores no tienen una adecuada apreciación de las cuestiones de confiabilidad del *software*, no son conscientes de las prácticas de desarrollo eficaces o no son capaces de aplicarlas apropiadamente. La importancia de la confiabilidad del *software* no se destaca adecuadamente en la mayoría de las carreras de grado en los Estados Unidos. Un mayor énfasis debe colocarse en el pensamiento sistémico, en los requerimientos, la especificación y los diseños en gran escala, en la seguridad, en la usabilidad, en el desarrollo de código robusto y resistente, en matemáticas discretas y estadística y en la construcción y análisis de los argumentos de confiabilidad

- **Las agencias federales que apoyan la investigación y el desarrollo en tecnologías de la información, deben dar prioridad a la investigación básica para lograr una confiabilidad mayor, enfatizando una perspectiva de sistemas y la obtención de evidencias**: Hasta que no haya una mejora notable en los métodos, lenguajes de programación y herramientas para el desarrollo de *software*, habrá sistemas que no podrán ser construidos con un nivel de confiabilidad adecuado. Se necesita una investigación que enfatice una perspectiva de sistemas y de las "tres E" (eficiencia, economía y elegancia) y que asigne mayor valor a los avances que puedan tener un impacto en un mundo de sistemas grandes interactuando con otros sistemas y operadores en un entorno físico y un contexto organizacional complejos

3.2 Sistemas Críticos para la Seguridad en Medicina

"La ciencia es el alma de la prosperidad de las naciones y la fuente de vida de todo progreso"
Louis Pasteur

En la actualidad existe una evidencia considerable que indica que los sistemas médicos computarizados pueden aportar un beneficio enorme en el mejoramiento del cuidado de los pacientes. De acuerdo con Kevin Fu (2011), profesor asistente en el Departamento de Ciencias de la Computación de la Universidad de *Massachusets Amherst*, "*…sin el software, muchos tratamientos médicos no existirían…*" [26]. Sin embargo, como con toda la

tecnología aplicada en Medicina, existen también riesgos potenciales involucrados. El consenso entre los desarrolladores e investigadores, agrega John Fox (2001), es que "*…estos sistemas deben diseñarse para que sean seguros y que cualquier efecto secundario o consecuencia negativa derivada de su uso se mantengan en un mínimo absoluto…*" [27]

El *IEEE*, en su Glosario de Estándares de la terminología de Ingeniería de *Software*, establece que la confiabilidad es "*la capacidad de un sistema o componente para desempeñar sus funciones requeridas, bajo las condiciones establecidas, por un período específico de tiempo*". En el caso del *software* para dispositivos médicos, esta definición debe expandirse para incluir los conceptos de seguridad y efectividad, según lo define Richard C. Fries (1997): "*la capacidad de un sistema o componente para desempeñar sus funciones requeridas* ***de una manera segura y efectiva****, bajo las condiciones establecidas, por un período específico de tiempo*". [28]

El concepto principal en esta definición, continuando con Richard C. Fries (1997), es que "*…la confiabilidad, la seguridad y la efectividad son requisitos inseparables en todo software para dispositivos médicos…*" [29]

En los Estados Unidos, la FDA tiene un papel muy activo en regular el *software* que se desarrolla para dispositivos médicos, estableciendo regulaciones y requisitos estrictos que deben cumplirse antes que los dispositivos médicos puedan ser vendidos a los hospitales y puestos a disposición de los médicos, de modo que no puedan provocar daño alguno a los pacientes. Los desarrolladores del *software* deben suministrar evidencias que prueben que el sistema ha sido diseñado, construido y probado de acuerdo con las mejores prácticas implementadas a través de las diversas fases del ciclo de vida, de acuerdo con *Parasoft Corporation* (2011). [30] También, recomienda no sólo que las pruebas incluyan una mezcla de métodos de análisis y prueba, aplicados a través del Ciclo de Vida de Desarrollo del *Software* (*SDLC, Software Development Life Cycle*), sino que también un conjunto amplio de actividades de Gestión del Ciclo de Vida del *Software* y de Gestión de Riesgos sean integradas a través del proceso para garantizar la entrega de *software* seguro y confiable. [31]

Existe actualmente una gran variedad de dispositivos médicos de todas las formas y tamaños que utilizan *software*, desde equipos de radiación enormes hasta pequeños marcapasos implantados y también varían mucho en el carácter crítico de sus funciones previstas. También, un gran volumen de información clínica es generado y procesado, cuyo uso puede hacer una diferencia fundamental para los pacientes, según lo menciona *Critical Software* (2014). [32]

El alcance del *software* aplicado a Sistemas Críticos para la Seguridad en Medicina, tal como lo establece la *Food and Drug Administration* (2002), incluye al: [33]

- *Software* usado como componente, parte o accesorio de un dispositivo médico
- *Software* que sea en sí mismo un dispositivo médico (v.g., *software* utilizado en los centros de donación de sangre)
- *Software* usado en la fabricación de un dispositivo (v.g., controladores lógicos programables (*PLC*) en los equipos de fabricación)
- *Software* usado en la implementación del sistema de calidad del fabricante del dispositivo (v.g., *software* que registra y mantiene el registro histórico del dispositivo)

De acuerdo con la función que cumple, el *software* para Sistemas Críticos para la Seguridad en Medicina, tal como lo menciona la *European Commission DG Health and Consumer Directorate B* (2012), puede clasificarse en: [34]

- *Software* que puede controlar directamente un aparato (v.g., tratamiento de radioterapia)
- *Software* que puede ayudar a tomar una decisión, basado en la información suministrada (v.g., medidores de glucosa en sangre)
- *Software* que proporciona apoyo a los profesionales de la Salud (v.g., interpretación de resultados de electrocardiogramas)

Adicionalmente, excluyendo el caso más representativo del *software* que puede controlar directamente un dispositivo médico, y teniendo en cuenta el entorno en el cual se aplica, el resto del *software* puede clasificarse en: [35]

- *Sistemas de información hospitalaria*: en este contexto, se refiere a los Sistemas que apoyan el proceso de tratamiento de los pacientes. Típicamente, se encuentran dedicados a la admisión de pacientes, a la programación de turnos y para propósitos de la gestión de facturación y seguros médicos
- *Software de apoyo a la decisión*: se refiere a herramientas basadas en computadora las cuales combinan base de datos de conocimientos médicos y algoritmos con datos

específicos del paciente. Están pensados para ayudar a los profesionales de la Salud con recomendaciones para el diagnóstico, pronóstico, seguimiento y tratamiento de pacientes individuales (v.g., Sistemas de planeamiento de tratamientos radioterapéuticos, Sistemas de planeamiento para la dosificación de medicamentos, Sistemas de detección asistidos por computadora, etc.)

- *Sistemas de información*: se refiere a aquellos Sistemas que están dedicados sólo al almacenamiento, archivo y transferencia de datos y que no son considerados en sí mismos como dispositivos médicos. (v.g., Sistemas electrónicos de los registros del paciente, Sistemas de información clínica, Sistemas de gestión de los datos del paciente, Sistemas móviles pre-hospitalarios, Sistemas de información radiológica, etc.) Sin embargo, pueden ser usados en conjunto con ciertos módulos que califiquen para ser considerados como dispositivos médicos, interactuando con ellos

- *Sistemas de comunicación*: son Sistemas dedicados a la transferencia electrónica de información (v.g., recetas, derivaciones, imágenes, registros de pacientes, etc.). Se basan en herramientas destinadas a fines generales, por lo que no se consideran en sí mismos como dispositivos médicos, aunque pueden ser utilizados junto con otros módulos que sí lo sean

- *Sistemas web para el monitoreo de datos*: estos Sistemas típicamente interactúan con un dispositivo médico (v.g., dispositivos implantados o dispositivos de atención domiciliaria) y utilizan un transmisor para el envío de información a través de Internet, una línea telefónica o una red móvil. La información se obtiene y almacena en un servidor web y puede ser accedida por los profesionales de la Salud que se encuentren autorizados o por los mismos pacientes, a través de Internet

- *Software para diagnóstico "in vitro"*: se refiere a Sistemas de Información de Laboratorio (*LIS, Laboratory Information System*) y Gestores de Áreas de Trabajo (*WAM, Work Area Managers*). Son Sistemas que apoyan a estos procesos desde la toma de muestras del paciente hasta la obtención de los resultados

Además de las implicancias negativas potenciales sobre la salud de las personas que los fallos en los Sistemas Críticos para la Seguridad en Medicina pueden provocar, el impacto sobre la economía es también notable. De acuerdo con la agencia *Reuters* un re-llamado para corregir un error de *software* en un dispositivo médico en el año 2012 tuvo un costo de U$S 221 millones y según un reporte de la consultora *McKinsey & Company*, mencionado

por Paul Anderson (2015) "*…el costo total de la calidad en la industria de los dispositivos médicos, incluyendo los costos y la pérdida de ganancias asociada con los fallos, asciende a entre U$S 17.000 millones a U$S 26.000 millones, o el equivalente a entre el 12 y el 18 por ciento de las ganancias de la industria…*" [36]

La necesidad de continuar desarrollando y aplicando los estándares para el desarrollo de *software* en Sistemas Críticos para la Seguridad en Medicina queda reflejada en los siguientes datos (ver Gráfico 2), proporcionados por la *Food and Drug Administration* (2013): [37]

- Un análisis realizado por la *FDA* de los re-llamados por fallos en dispositivos médicos realizados entre 1992 y 1998 reveló que de un total de 3.140 casos, en 242 de ellos (7,7%) la causa estaba relacionada con fallos de *software*
- El mismo análisis realizado entre 2008 y 2012 reveló que de un total de 4.941 casos, en 746 de ellos (15,1%) la causa estaba relacionada con fallos de *software*, afectando a millones de dispositivos en uso en los Estados Unidos

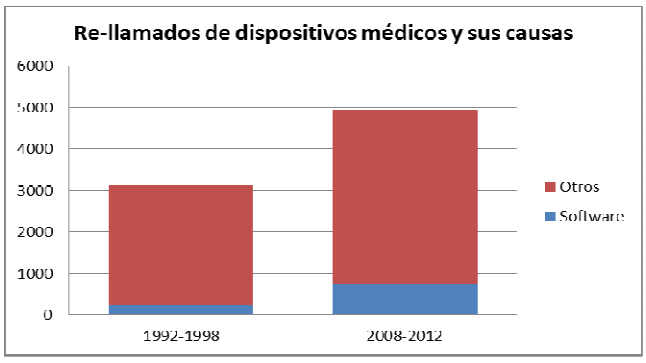


Gráfico 2: Re-llamados de dispositivos médicos y sus causas

Mientras que la cantidad total de re-llamados se incrementó entre dichos períodos en un 57,35%, los casos relacionados con fallos en el *software* se incrementaron en un 208%, duplicando su importancia entre el total de motivos relacionados con los re-llamados.

La *FDA* atribuye estos fallos a inconvenientes para implementar controles adecuados en el diseño del *software*, procedimientos de pruebas insuficientes y, principalmente, a la complejidad creciente en el entorno de uso del dispositivo médico, debido al incremento en la conectividad y la interoperabilidad. [38]

Esto agrega un elemento más dentro de la complejidad de la seguridad en los Sistemas Críticos en Medicina, el cual, según lo definido por el *Board on Health Care Services* (2012), es que "*…la Seguridad es una propiedad emergente de un sistema más amplio que incluye no sólo el software, sino también la forma en que es utilizado por los médicos. Este sistema más grande, a menudo llamado Sistema Socio-Técnico, incluye la tecnología (v.g., software y hardware), las personas (v.g., médicos y pacientes), los procesos (v.g., flujos de trabajo), las organizaciones (v.g., su capacidad y la toma de decisiones sobre cómo aplicar las Tecnologías de la Información) y el ambiente externo (v.g. las regulaciones y la opinión pública). La adopción de una perspectiva socio-técnica reconoce que la Seguridad surge de la interacción entre diversos factores…*" [39]

La creación y consecuente aplicación de los estándares recomendados en el desarrollo de *software* de Sistemas Críticos para la Seguridad en Medicina, sin embargo, no son suficientes por sí mismos. Los estándares son sólo tan eficaces como las personas que los aplican y, de acuerdo con Barry Barber y Ray Rogers (2003), necesitan también del establecimiento de: [40]

- Una cultura de la seguridad para los profesionales en Sistemas de Información aplicados a la Medicina para que coincida con la cultura de la seguridad desarrollada entre los profesionales de la Salud

- Un código de práctica de Ética para los profesionales en Sistemas de Información aplicados a la Medicina

3.3 Sistemas Críticos para la Seguridad en Energía Nuclear

"Muchos países continuarán encontrando en la energía nuclear una opción de futuro y, por eso, tenemos que hacer lo posible para garantizar la seguridad"
Ban-Ki-moon

Los reactores nucleares, al igual que cualquier complejo de plantas industriales, pueden experimentar fallas operacionales o en sus equipamientos, las cuales conllevan potencialmente consecuencias graves. De acuerdo con la IAEA (2014), la Seguridad Nuclear se define como "*el logro de condiciones de operaciones adecuadas, la prevención de accidentes o la mitigación de las consecuencias de un accidente, lo que resulta en la protección de los trabajadores, el público y el medio ambiente de los peligros debidos a la radiación*". [41] Una posible consecuencia de un accidente en un reactor nuclear es la

liberación de radiación o material radiactivo en el medio ambiente y por lo tanto podría tener consecuencias inmensamente más grandes que en otras actividades, ya que por ejemplo, un área de considerable superficie podría tener que ser evacuada y permanecer inhabitable por muchos años. En general, las plantas nucleares cuentan con cuatro tipos de Sistemas de Seguridad, tal como lo detalla la *Atomic Energy of Canada Ltd. and Ontario Power Generation, Inc.* (1999): [42]

- Un Sistema de parada de emergencia con redundancia
- Un Sistema de enfriamiento del núcleo de emergencia
- Un Sistema de contención

Estos Sistemas, encargados de monitorear la seguridad en los reactores nucleares, se denominan Sistemas Clase 1E. De acuerdo con el *IEEE*, según lo mencionado por Warren L. Persons y J. Dennis Lawrence (1993), se llama así a "*…la clasificación de seguridad de los equipos eléctricos y sistemas que son esenciales para la parada de emergencia del reactor, el aislamiento en la contención prevista, el enfriamiento del núcleo del reactor, la contención y eliminación del calor generado por el reactor, u otros que sean esenciales en la prevención de la liberación de material radiactivo al medio ambiente…*" [43]

A mediados de la década del 70, comenzó a implementarse *software* para los Sistemas Clase 1E en reemplazo de los dispositivos analógicos o electromecánicos (*relé*), denominándose a los mismos *Software* Clase 1E. [44]

Una definición más amplia de los que se conocen como Sistemas Críticos para la Seguridad relacionados con la Energía Nuclear es la de Sistemas de Instrumentación y Control (*I&C, Instrumentation and Control*), aportan conceptualmente Nicolas Sannier y Benoit Baudry (2012). En ella, se incluye a "*los instrumentos para monitorear las condiciones físicas de la planta (v.g., temperatura, presión y radiación), los sistemas redundantes para hacer frente a condiciones accidentales (v.g., sistemas de seguridad) y todo el equipamiento para que los operadores puedan controlar el comportamiento de la planta*". [45]

Lo que distingue a los Sistemas Críticos para la Seguridad en Energía Nuclear respecto a otras aplicaciones de Sistemas Críticos, de acuerdo con el *National Research Council* (1997), es la necesidad de establecer los mismos niveles muy altos de confiabilidad y seguridad pero en un rango más amplio de condiciones, de manera que "*los Sistemas de I&C deben ser utilizados para reducir la posibilidad de que incluso los eventos de baja*

probabilidad ocurran". [46] Del mismo modo, se establece la necesidad de separar completamente los Sistemas de Control de los Sistemas de Seguridad para poder asegurar la confiabilidad y la seguridad, reducir la complejidad, facilitar la verificación y proteger al Sistema de Seguridad de interferencias para que pueda realizar sus funciones previstas.

Los Sistemas de I&C (ver Figura 1) son agrupados generalmente en tres categorías: [47]

- Sistemas de monitoreo de la planta: se encarga del monitoreo de variables de la planta y proporciona datos para los otros Sistemas de I&C. Por ejemplo, monitorean y muestran el estado del Sistema de protección contra-incendios, temperatura y presión de los fluidos, etc. Generalmente proporcionan alarmas visuales y auditivas para las estaciones de control, notificando a los operadores sobre alguna tendencia o valor en particular que requiera una acción por parte de los operadores para evitar un problema o emergencia real

- Sistemas de control de la planta: se usan para controlar todas las operaciones normales de la planta que involucran al arranque, la generación de energía, las paradas e inconvenientes en la planta. Están diseñados para ser robustos y cuentan un nivel de redundancia considerable, para poder prevenir fallos individuales y anticiparse a eventos que podrían generar una parada de la planta u otros accidentes que pongan en peligro al personal, el equipamiento o al público. Por ejemplo, los Sistemas de alimentación del suministro de agua y control del vapor, los controles del generador de la turbina y la multitud de otros Sistemas utilizados para el control de disyuntores, bombas, válvulas, etc.

- Sistemas de protección y mitigación de accidentes: constituyen una capa separada de los demás Sistemas de I&C. En caso de detectar, por ejemplo, que los Sistemas de monitoreo de la planta y de control de la planta no logran mantenerla dentro de un conjunto pre-definido de condiciones, tomará acciones en forma inmediata para proceder al apagado de la planta y arrancar los Sistemas que puedan mitigar el problema detectado y volver a restituir la planta a un estado seguro. Están diseñados a prueba de fallos individuales, es decir no permite ningún fallo a nivel del Sistema ni tampoco que un operador, por error, pueda evitar que el Sistema funcione adecuadamente. Poseen redundancia, teniendo en paralelo un conjunto separado de equipamiento y Sistemas para llevar a cabo la misma función. Este tipo de redundancia proporciona protección contra los fallos en los cuales un error individual deshabilita múltiples funciones de seguridad independientes, llamados

modos de averías comunes (*common-mode failures*). Además, poseen otras funcionalidades para poder mejorar la confiabilidad y aumentar la eficacia frente a los riesgos

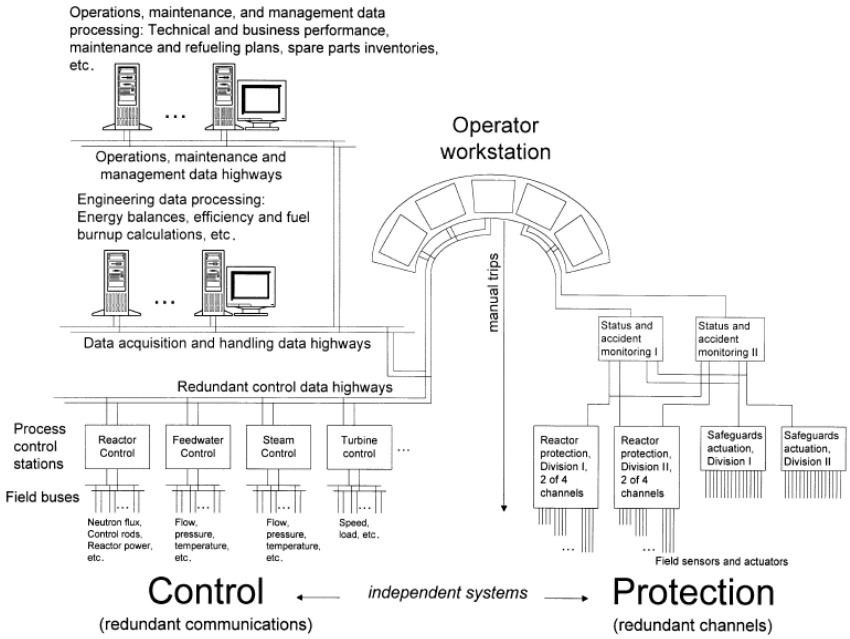


Figura 1: Ilustración de los Sistemas de I&C en una planta nuclear [48]

Para que los Sistemas de I&C sean implementados con éxito en las plantas de energía nuclear, uno de los factores críticos es la consideración del rol de los operadores en cuanto a los factores humanos y a las interfaces hombre-máquina. En las siguientes fotografías (ver figuras 2 a 5) podemos apreciar la evolución en el tiempo de las salas de control en cuatro plantas de energía nuclear en Japón: [49]

Figura 2: Planta *Mihama* en los años ‘70

Figura 3: Planta *Takahama* en los años ‘80

Figura 4: Planta *Ohi* en los años ‘90

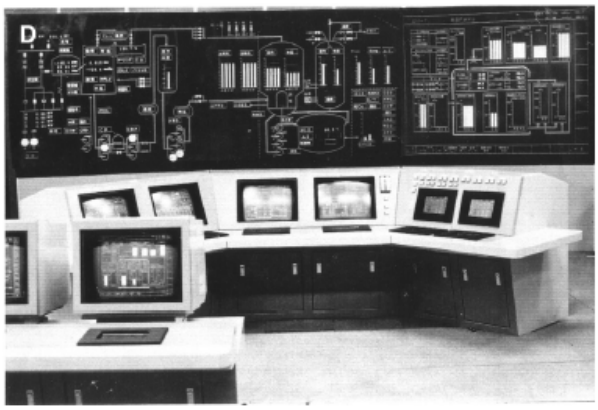


Figura 5: Nueva generación de plantas, construidas por *Kansai Electric Power Co., Inc.*

De esta manera, con un conjunto cada vez menor de dispositivos de asociados a los Sistemas de I&C, según lo establece la *Japan Nuclear Energy Safety Organization* (2007), un operador podría controlar las operaciones de toda una planta nuclear que genera 1.350 Mwe. [50]

Además del principio ya mencionado de *redundancia*, en los Sistemas Críticos para la Seguridad en Energía Nuclear, se utiliza el concepto de *diversidad.* Por ejemplo, según lo detallado por Mark Lawford y Alan Wassyng (2012), un Sistema de parada de emergencia con redundancia (*Shutdown*) está compuesto por dos sistemas distintos en donde cada uno de ellos posee distintos *hardware*, lenguaje de programación utilizado en el *software* y métodos de uso. [51]

Un concepto adicional que es empleado en muchos de los Sistemas Críticos para la Seguridad y en forma notoria en los aplicados a las plantas de Energía Nuclear, de acuerdo con la *International Atomic Energy Agency* (2009), es el de *defensa en profundidad*. Este principio implica "*…la provisión de barreras consecutivas e independientes que protejan contra las amenazas identificadas…la defensa en profundidad ayuda a mitigar los efectos de las CCF (common cause failures)…*" [52]

Una Falla de Causa Común (*CCF*) es definida, de acuerdo con el Consejo de Seguridad Nuclear (2014), como "*la falla de dos o más estructuras, sistemas, canales o componentes debido a un único evento específico o causa*". [53]

3.4 Sistemas Críticos para la Seguridad en Transporte Ferroviario

"Los ferrocarriles constituyen la llave fundamental de una nación. La economía nacional, pública y privada, el equilibrio de las diversas regiones que la integran, la actividad comercial e industrial, la distribución de la riqueza y hasta la política doméstica e internacional están íntimamente vinculadas a los servicios públicos de comunicación y transporte"
Raul Scalabrini Ortiz

Jonathan F. Luedeke (1995), explica que la aplicación de Sistemas Críticos para la Seguridad en la Industria Ferroviaria ha evolucionado desde los sencillos pero vitales relés hasta la implementación de configuraciones más complejas basadas en sistemas informáticos, en particular a partir de la introducción de trenes de alta velocidad y de levitación magnética, siendo utilizados para funciones críticas para la Seguridad tales como determinar la localización de un tren, control de rutas por enclavamiento (*interlocking*), control de la propulsión y el frenado para garantizar una distancia segura, gestión de las comunicaciones entre los trenes, los sistemas de señalización en el trayecto de las vías y el control centralizado, entre otras. [54]

Entre las preocupaciones más importantes respecto a la Seguridad en el uso de Ferrocarriles, de acuerdo con Alexei Iliasov y Alexander Romanovsky (2015), se destacan: [55]

- Evitar alguna colisión entre formaciones: una colisión sucede cuando dos trenes se encuentran en la misma parte de un tramo. El mecanismo básico de seguridad es el de bloquear o permitir el paso por el tramo, por medio de señales
- Evitar los descarrilamientos: un descarrilamiento se da cuando un tren se sale de sus carriles o rieles
- Asegurar que la velocidad de la formación sea la adecuada para el tramo que está recorriendo: en una curva, por ejemplo, si circulara a una velocidad insegura, la combinación de las fuerzas gravitacionales, centrípetas y centrífugas podrían provocar un descarrilamiento
- Asegurar que las puertas de la formación permanezcan cerradas y se abran sólo en las estaciones o en situaciones de emergencia

- Proporcionar seguridad al público en los pasos a nivel

En general, los Sistemas que controlan estos eventos se componen de cuatro partes, según lo detalla B. Ning (2010): [56]

- Un Sistema de Control Centralizado (*CCS, Central Control System*)
- Los Sistemas de Control de las Estaciones (*SCS, Station Control System*) y Sistemas de Señalización en el recorrido de las Vías (*BCS, Block Control System*)
- Los Sistemas de Control de a Bordo en las formaciones (*OCS, Onboard Control System*)
- La Red de Comunicaciones (*CNS, Communication Network System*), incluyendo las comunicaciones móviles

Al mismo tiempo, los Sistemas de Señalización cumplen con un rol fundamental en dirigir el tráfico de los ferrocarriles en forma segura y así prevenir colisiones. En su forma moderna, estos sistemas (ver Gráfico 3) son conocidos como Sistemas *CBTC* (*Communications-Based Train Control*). El *IEEE* define a estos Sistemas como "*…un Sistema de control de trenes automático y continuo que utiliza determinación en alta resolución de la ubicación de un tren, independiente de los circuitos de vías, con una comunicación de datos continua, de alta capacidad y bidireccional entre el tren y los sensores al costado de las vías, y con procesadores tanto en el tren como en las vías capaces de implementar funcionalidades de protección y opcionalmente funcionalidades de control y supervisión…*" [57]

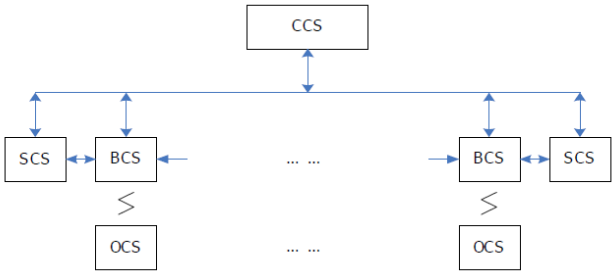


Gráfico 3: Configuración de un Sistema *CBTC* [58]

El Sistema de Control de Estación (*SCS*), establecen N. Bin et al. (2010), es un sistema de enclavamiento (*interlocking*) el cual controla los interruptores, señales, cambios de aguja

(desvíos), pasos a nivel y rutas en las estaciones o en determinadas áreas. *SCS* se comunica con el Sistema de Control Centralizado (*CCS*), con el Sistema de Señalización (*BCS*) y con el Sistema de Control de a Bordo (*OCS*). [59] Un enclavamiento es un dispositivo que permite controlar la circulación en una estación de ferrocarril, siendo capaz de manejar las señales, los desvíos, los calces y las semi-barreras. Además, impide el cambio de los elementos anteriores si la nueva posición se encuentra en una configuración incompatible con la de otro elemento. La tecnología de enclavamientos, siguiendo a Andrew Ryan (2010), puede clasificarse en: [60]

- Enclavamiento por bloques fijos: divide la red ferroviaria en bloques fijos los cuales se encuentran separados por señales. Sólo un tren puede encontrarse en el bloque al mismo tiempo y puede ingresar cuando el próximo bloque está liberado

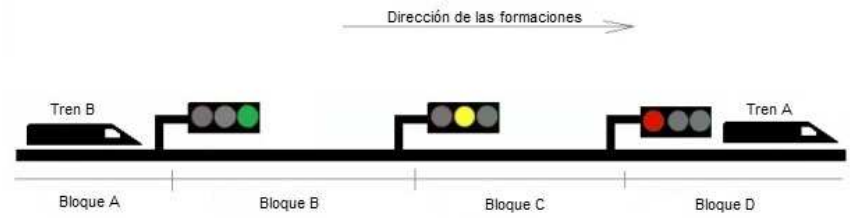


Gráfico 4: Enclavamiento por bloques fijos

- Enclavamiento por bloques móviles: en lugar de dividir la red en bloques fijos, cada tren está asociado con un bloque móvil, el cual está definido por la distancia que necesita el tren para frenar completamente

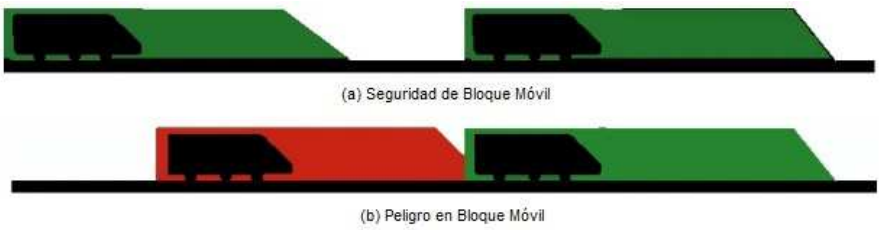


Gráfico 5: Enclavamiento por bloques móviles

Los tipos de enclavamientos, de acuerdo con M. Antoni y N. Ammad (2008), pueden dividirse en: [61]

- Enclavamiento de aproximación: impide el cambio de la ruta cuando el tren se encuentra en la zona de aproximación
- Enclavamiento de tránsito: asegura que todos los desvíos estén en la posición adecuada antes y durante el paso del tren
- Enclavamiento de dirección: impide la apertura simultánea de las señales de dos rutas que tengan direcciones convergentes

El Sistema de Control de Bloqueo (*BCS*) controla la operación segura de los ferrocarriles a través de los bloqueos, organizando la asignación de tramos de vía en la circulación de trenes. El objetivo es impedir la colisión entre trenes reservando cada tramo de vía para un único tren, evitando que dos trenes circulen en sentido contrario por el mismo tramo de vía, que un tren alcance al precedente o que un tren circule sobre aparatos de vía que no hayan sido configurados correctamente.

El Sistema de Control de a Bordo (*OCS*) se encuentra en las locomotoras de las formaciones y su función principal es controlar la velocidad del tren, como también la aceleración, la desaceleración, la velocidad de crucero y el frenado. Los datos relativos a la posición del tren y a su velocidad son enviados al Sistema de Control de Bloqueo (*BCS*) y se reciben por su parte los datos relativos a las velocidades autorizadas y a la Autorización de Movimiento (*MA*) que permite al tren proceder con seguridad hasta cierto punto en el que tiene que detenerse. Estos puntos de conocen como Puntos de Conflicto (*CP*), esto es una localización puntual en el recorrido de las vías más allá de la cual no le está permitido al tren avanzar. Este punto puede ser estático, es decir que su localización en el recorrido es fija, o dinámico, es decir que su localización es otro tren en movimiento, de acuerdo con L. Lindqvist y R. Jadhav (2010). [62]

Casi todos los dispositivos que intervienen en estas funciones poseen redundancia doble y en algunos casos hasta triple, establecen T. Matsuki et al. (2010), a fin de mejorar la fiabilidad del Sistema. [63]

Controlando todas estas actividades, se encuentra el Sistema de Control Central (*CCS*) cuyo núcleo son los algoritmos de despacho que constituyen el *software*. Su tarea es la de generar gráficos de los trenes de acuerdo con los requisitos de la planificación y, si es necesario, restaurar el funcionamiento normal de los trenes cuando el plan de operaciones se encuentre interrumpido, continuando con N. Bin et al. (2010). [64]

El Modelo de Control de Operaciones de los trenes es fundamental para garantizar que las operaciones se realicen en forma segura, ya que los tramos de vías que se habilitan para un tren son calculados en tiempo real, denominándose a estos tramos como *intervalos de trenes*.

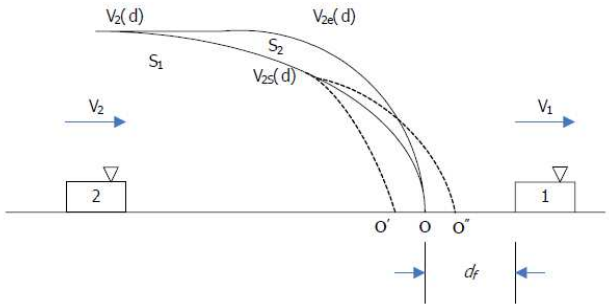


Gráfico 6: Control de Intervalos de Trenes [65]

Los elementos más importantes en el control mostrado en la figura anterior son la *distancia de protección de seguridad* d_f, esto es el intervalo entre dos trenes que corresponde a la distancia de frenado segura, la *curva de velocidad* del segundo tren $V_2(d)$, la *curva de frenado de emergencia* del segundo tren $V_{2e}(d)$, la *curva de frenado de servicio* del segundo tren $V_{2s}(d)$, el *punto de detención* calculado para el segundo tren O y los *puntos de detención real* del segundo tren O' y O''. De esta manera se asegura que delante de cada tren en movimiento hay una distancia suficiente que es igual a la distancia de frenado total, como mínimo. El tren que sigue a otro debe adaptar permanentemente su velocidad en función de la del tren que está delante, con el fin de asegurar que es capaz de detener la formación, evitando una colisión.

Por otro lado, C. Braban y P. Charon (2010) mencionan que los sistemas *CBTC* pueden asimismo funcionar sin un conductor (*driverless*). Para ello, la disponibilidad de la Red de Comunicaciones (*CNS*) es esencial. [66] Estos sistemas son utilizados para toda la variedad de tráfico de trenes, desde los de alta velocidad y líneas convencionales de pasajeros hasta los de líneas de carga.

La constante demanda de más y mejores servicios de ferrocarriles para atender a la cantidad creciente de pasajeros, en la mayoría de los países, plantea nuevos desafíos respecto a la seguridad. Por ejemplo, los algoritmos de despacho y los Sistemas de Control de Bloqueos (*BCS*) deben considerar, para el cálculo de los *intervalos de trenes*, otros elementos adicionales: un tren no debe quedar detenido con una parte dentro y otra fuera de las

estaciones, tampoco dentro de túneles o viaductos, algunos trenes se detienen en ciertas estaciones mientras que otros no, el tiempo de detención puede variar entre formaciones, etc. De esta manera, de acuerdo con W. A. M. Barter (2010), los intervalos deben ser calculados con una longitud mínima más larga que la que en teoría podría utilizarse. [67]

La capacidad de la red de ferrocarriles queda definida entonces como "*…el número de trenes que se pueden incorporar en una planificación que esté libre de conflictos, que sea rentable comercialmente, que cumpla con los requisitos de las regulaciones y que pueda ser operada teniendo en cuenta los niveles anticipados de retrasos para poder cumplir con los objetivos de desempeño acordados…*". [68]

Tanto en los Sistemas de Control de Estación (*SCS*) como en los Sistema de Señalización (*BCS*) se utiliza el concepto de *redundancia orientada a la Seguridad* en configuración redundante triple "*2 de 3*", en la cual se exige el acuerdo de al menos dos de los computadores, aplicándose el mecanismo de *decisión mayoritaria*. La compañía Enclavamientos y Señalización Ferroviaria S.A. (2010) explica que esta configuración ofrece una alta disponibilidad, ya que un fallo simple no afecta a la continuidad del servicio, asegurada en ese caso por los dos computadores restantes sin menoscabo alguno de la seguridad. [69]

Otro concepto que se introduce respecto a la Seguridad en estos Sistemas Críticos es el de *degradación gradual*, en la cual un sistema complejo tiene la facilidad de cambiar a otro modo de funcionamiento si ocurre alguna falla o se reduce la disponibilidad. Una vez que se aísla la falla y se confirma su origen, de acuerdo con la compañía *Railtrack PLC* (1998), el sistema puede continuar funcionando y si es necesario reconfigurarse en un modo degradado. [70]

3.5 Sistemas Críticos para la Seguridad en Transporte Automotor

"Aquel que quiere viajar feliz, debe viajar ligero"
Antoine de Saint-Exupery

La expansión en la demanda de nuevas mejoras para las prestaciones, el confort y la seguridad en los vehículos automotores ha dado lugar a un crecimiento rápido y acelerado en la cantidad y la sofisticación de la electrónica del automóvil, llegando a controlar funciones de los vehículos que son esenciales, como el frenado y la dirección. Sin dudas, han proporcionado mejoras sin precedentes en cuanto a la seguridad, pero también tienen la

capacidad potencial de crear problemas relacionados con la misma cuando no funcionan correctamente, nos explican Barbara J. Czerny et al. (2003). [71]

Hay varios aspectos específicos que se dan en el Sistemas Críticos para la Seguridad en Automotores. En primer lugar, los proveedores de la industria automotriz trabajan con clientes de diversas partes del mundo, cada uno de los cuales puede requerir la aplicación de diferentes estándares relacionados con la seguridad. Otro aspecto es que el desarrollo de *software* para estos Sistemas Críticos en la Industria Automotriz está basado casi exclusivamente en el lenguaje de programación *C*. [72]

Con una producción anual de 70.000.000 de vehículos en todo el mundo, el uso de *software* en los automóviles ha crecido, desde el año 1990, a una tasa del 10% anual. El tamaño del *software* utilizado para el *Peugeot* Modelo CX pasó de 1,1 KB en 1980 a 2 MB para el Modelo 607, en el año 2000. Actualmente, calculan Nicolas Navet y Francoise Simonot-Lion (2009), el costo de la electrónica en una automóvil de alta gama puede representar un 35% del costo total del vehículo [73], utilizando más de 100 Unidades de Control Electrónico (*ECU*) ejecutando, agrega Dave Higham (2013), alrededor de 100 millones de líneas de código. [74]

En un automóvil moderno, estos sistemas controlan el funcionamiento del motor, los limpiaparabrisas, las puertas, la suspensión, la transmisión, el frenado (*ABS*), los dispositivos multimedia, las bolsas de aire (*airbags*), los espejos, la emisión de gases, el consumo de combustible, la estabilidad del vehículo (*ESP*), entre otras funciones, y además interactúan con el conductor y los pasajeros. Por ejemplo, un vehículo puede determinar mediante sus sensores cuando un accidente causa que se activen las bolsas de aire y utilizar al receptor del Sistema de Posicionamiento Global (*GPS*) para que transmita a un equipo de rescate la posición exacta del vehículo. Los fabricantes de automóviles, continuando con Nicolas Navet y Francoise Simonot-Lion (2009), estiman que actualmente más del 80% de la innovación y valor agregado en los modelos se obtiene a partir de los sistemas electrónicos [75]

Una de las principales características de estos sistemas es el uso intensivo de la *multiplexión*, cuya principal ventaja es una reducción significativa en los costos de cableado como también otorgar más flexibilidad a los diseñadores. [76]

El impacto de los fallos de estos sistemas es altamente significativo. En el año 2003, casi el 50% de las averías en automóviles en Alemania se debieron a fallos de dichos sistemas. [77]

Los sistemas pueden clasificarse en función del *dominio* al que corresponden las distintas funcionalidades, restricciones y modelos. De acuerdo con la organización europea *ITEA* (*Information Technology for European Advancement*) se define a un *dominio* como "*una esfera de conocimiento, influencia y actividad con la cual uno o más sistemas deben tratar*". En el caso de los Sistemas aplicados a los Automotores, se han definido seis dominios: [78]

- Motor y Transmisión: representa al sistema que controla al motor de acuerdo con los requerimientos del conductor y los de otras partes de los sistemas integrados. En este dominio interactúan tanto sistemas continuos como muestreados y discretos. En el despliegue de estos sistemas existen algunas características complejas que deben tenerse en cuenta. Por ejemplo, si por razones de costos la implementación debe realizarse sobre un microprocesador que no cuenta con un coprocesador de coma flotante, debe prestarse atención a la precisión de los valores para asegurar que cumplan con los requerimientos. Otro desafío es programar eficientemente las actividades cíclicas, ya que algunas de ellas tienen períodos constantes, mientras que en otras son variables, de acuerdo con los ciclos del motor. Esto significa que su programación depende de diferentes relojes lógicos

- Chasis: comprende a los sistemas cuyo objetivo es controlar la interacción del vehículo con el camino, teniendo en cuenta los requerimientos del conductor, las características del camino y las condiciones ambientales, debiendo asegurar tanto el confort como la seguridad del conductor y los pasajeros. Este dominio incluye sistemas tales como *ABS, ESP, ASC* y *4WD*. Sus características son también similares a las del dominio Motor y Transmisión, es decir controles con variables múltiples, períodos de muestreo diferentes y limitaciones de tiempo estrictas

- Carrocería: contiene todas las funciones que no están directamente relacionadas con la dinámica del vehículo como limpiaparabrisas, luces, puertas, ventanillas, asientos y espejos, entre otras. Interactúa en gran medida con otros sistemas y contiene un subsistema central que asegura que los mensajes sean enviados entre los diferentes sistemas o dominios. En general, estas funciones están asociadas con eventos discretos y su diseño y validación se basan principalmente en *máquinas de estados*. En este contexto, uno de los desafíos es ser capaz de desarrollar un análisis exhaustivo de los diagramas de transición de estados y asegurar que la aplicación respete las limitaciones de seguridad y tolerancia a los fallos

- Multimedia, Telemática e Interacción Hombre-Máquina (*HMI*): La Telemática en los vehículos incluye a los sistemas que soportan el intercambio de información entre vehículos o con la infraestructura del camino. Los sistemas *HMI* soportan la interacción del vehículo con el conductor y los pasajeros. Para estos sistemas el desafío no sólo es considerar la calidad, prestaciones y confort de los servicios ofrecidos, sino también considerar el impacto de esta tecnología respecto a la seguridad. En este sentido, se han comenzado a implementar los llamados *HUD* (*Head-up display*) lo cual permite mostrar información importante sobre el vehículo directamente en el parabrisas, en la línea directa de visión del conductor, evitando así potenciales distracciones respecto al camino. El tamaño óptimo de estos sistemas constituye un desafío ya que por ejemplo una plataforma integrada de telemática y multimedia puede estar compuesta por dos procesadores en los cuales se ejecutan unos 250 hilos (*threads*) en una Máquina Java o sobre un sistema operativo multitarea

- Seguridad Activa y Pasiva: Para la industria automotriz, el desafío consiste en cumplir con los requerimientos de seguridad, minimizando los costos. La Seguridad Activa advierte antes de la ocurrencia de un incidente mientras que la Seguridad Pasiva actúa cuando el incidente ha sucedido. Los sistemas de Seguridad Activa interpretan las señales que provienen de diversos sensores y otros sistemas para asistir al conductor en el control del vehículo e interactuar con los demás sistemas. Los sistemas de Seguridad Pasiva, como por ejemplo los cinturones de seguridad o las bolsas de aire, contribuyen a mitigar los efectos de un accidente y han alcanzado actualmente un buen nivel de madurez

- Diagnóstico: La complejidad de las arquitecturas electrónicas en los automóviles actuales contiene funciones que están desplegadas en varios microprocesadores, los cuales interactúan intensivamente entre ellos. El diagnóstico se ha convertido así en una función vital para la vida útil del vehículo. Un sistema de Diagnóstico (*OBD*) puede recolectar información y determinar el diagnóstico apropiado, reportándolo.

La aplicación de tecnologías de guiado electrónico, llamadas *drive-by-wire*, está siendo implementada en los automóviles. El propósito de estas tecnologías es la de asistir al conductor y a su vez disminuir los costos de producción y mantenimiento. Sin embargo, distinto a lo que ocurre, por ejemplo, en los Sistemas Críticos para la Seguridad en Aviación, no pueden dejarse los elementos mecánicos o hidráulicos originales como

elementos redundantes, debido a restricciones de costos, espacio y peso; por lo tanto necesitan desarrollarse soluciones específicamente tolerantes a los fallos.

De acuerdo con la *ISO*, en su norma 26262, la seguridad funcional en los vehículos queda asegurada cuando "*…una función del vehículo no lo coloque en un estado de peligro intolerable, como resultado de un error de especificación, implementación o realización, por una falla durante la operación o por errores operacionales o mal usos que sean razonablemente previsibles…*". [79]

A fin de disminuir la increíble y lamentable cifra de 1 millón de muertes por accidentes de tránsito, en todo el mundo, muchos países han establecido objetivos para reducir las tasas de mortalidad y en algunos casos, como Suecia, llevarlas incluso a cero. Sin embargo, muchos accidentes siguen ocurriendo debido a errores de los conductores. En consecuencia, se están desarrollando nuevos sistemas que son capaces de controlar el automóvil en lugar del conductor, en determinadas circunstancias, para minimizar sus errores. Algunos de los inconvenientes que se presentan en este proceso son: [80]

- Las limitaciones en cuanto a costo y espacio restringen el uso de redundancia
- La infraestructura vial se encuentra totalmente abierta
- La mayoría de los conductores no son "profesionales"

En forma distinta a, por ejemplo, un tren que en caso de fallas se detiene y como vimos anteriormente el Sistema de Señalización impide que otro tren acceda a ese bloque del trayecto, o a un avión que en caso de fallas posee sistemas redundantes que reemplazan al que genera la falla, en el caso de los automóviles no puede simplemente detenerse al vehículo, ya que esto, en general, sería visto como un obstáculo peligroso para el resto de los automóviles. Por ello, los llamados Sistemas Avanzados de Asistencia al Conductor (*ADAS*) se están desarrollando, los cuales pueden detectar obstáculos en el camino y ayudar en las maniobras para evitar una colisión. Si, no obstante, el vehículo pasa a un estado conocido como de *colisión inevitable* (*ICS*), estos sistemas colaboran en mitigar el daño incluso antes que actúen los sistemas de Seguridad Pasiva.

Con la mejora en las tecnologías de comunicación, conocidas como de vehículo a vehículo (*V2V*) y de vehículo a Infraestructura (*V2I*), se están desarrollando nuevas aplicaciones cuyo objetivo es mejorar la seguridad, colaborando a que otros vehículos eviten pasar a un estado de *ICS*, por ejemplo: [81]

- Un vehículo detenido puede difundir una señal de advertencia a los demás vehículos dentro de un alcance determinado
- El accidente puede ser informado al sistema centralizado de monitoreo del tráfico y retransmitirse a los vehículos que ingresan en esa área
- Adicionalmente, y dependiendo del ancho de banda disponible, cada vehículo podría difundir sus trayectorias para ofrecer una percepción redundante del entorno

También, se establece la necesidad de crear *Autodiagnósticos Inteligentes*, a fin de incrementar las aptitudes en cuanto a detección de fallas.

A fin de ser implementado en un futuro inmediato, se están desarrollando nuevas formas de transporte, basadas en la automatización del tráfico por las rutas, estableciendo redes jerárquicas (por ejemplo, los programas *CyberCars* y *CityMobil* en Europa).

Figura 6: *CyberCars*, de la empresa francesa *Robosoft* [82]

Asimismo, nuevos conceptos como vehículos híbridos o eléctricos, impulsados por la legislación que protege el medio ambiente, representan nuevos desafíos para el desarrollo de Sistemas Críticos para la Seguridad. El desarrollo de *software* para estos nuevos tipos de vehículos es más largo que los de vehículos convencionales, de acuerdo con Masaru Kurihara (2010), ya que hay que realizarlos desde cero, en cambio los otros avanzan mediante desarrollos funcionales incrementales. [83]

Al igual que en los demás Sistemas Críticos para la Seguridad mencionados, existe en este caso la posibilidad que el *software* sea comprometido en forma malintencionada por

terceros. Se han desarrollado experimentos, desarrollando una serie de vectores de ataque para socavar en forma sustancial los sistemas de seguridad de un modelo comercial, siendo capaz de impedir el frenado del vehículo, falsificando las lecturas del velocímetro y deteniendo el motor, de acuerdo con la compañía *GrammaTech* (2011). [84]

3.6 Sistemas Críticos para la Seguridad en Aviación

"Los ángeles pueden volar porque se toman a sí mismos a la ligera"
Gilbert Keith Chesterton

Los Sistemas Críticos para la Seguridad en Aviación, tanto los que se encuentran a bordo de las aeronaves como los que están en los aeropuertos y en los centros de control del tráfico aéreo, postula William S. Greenwell (2003), realizan funciones esenciales en las cuales deben confiar tanto las tripulaciones como los controladores de tráfico aéreo a fin de poder operar y dirigir a las aeronaves en forma segura. [85]

La mayoría de los sistemas de a bordo en una aeronave (*FMS*), como los controles de vuelo, la aviónica o los controles del motor, son ejemplos típicos de Sistemas Críticos para la Seguridad que utilizan *software* en forma intensiva. Usualmente, establece la *Federal Aviation Administration* (2007), operan en ambientes con diversos rangos de temperatura, humedad, presión atmosférica, vibración y movimiento, y están sujetos al accionar del tiempo, el mantenimiento y las condiciones climáticas. [86]

Una proporción cada vez mayor de las funciones de los Sistemas de Navegación Aérea (*ANS*), de acuerdo con la compañía *SESM Finmeccanica Company* (2011), están implementadas mediante *software*. [87]

En un vuelo comercial actual, la aeronave es dirigida mayormente por el piloto automático, dejando las tareas más delicadas como el despegue y el aterrizaje a cargo de los pilotos. Las tareas de los pilotos han cambiado de mantener la aeronave en vuelo y en el rumbo correcto a programar y monitorear los sistemas de a bordo, conocidos como *aviónica*, e interactuar con el Control de Tráfico Aéreo (*ATC*). Los sistemas llamados "*fly-by-wire*", permitieron reemplazar los controles de vuelo analógicos por los electrónicos digitales. En estos, en lugar de transmitir los comandos de vuelo a las superficies de control a través de enlaces mecánicos, neumáticos o hidráulicos, se transmiten señales a las computadoras, las cuales estimulan los actuadores de las superficies de control necesarias para la ejecución de los comandos basándose, entre otros según lo establecen Ian Moir y Allan Seabridge (2001), en los parámetros suministrados por el Sistema de Referencia Inercial y de Datos Aéreos

(*ADIRS*). [88] Estas computadoras, continuando con William S. Greenwell (2003), cuentan con un Sistema de Protección de la Envolvente de Vuelo, el cual puede rechazar los comandos de los pilotos que pudieran comprometer la seguridad excediendo la envolvente operacional de la aeronave. [89]

En un sentido más amplio, podemos considerar a las actividades aeronáuticas (ver Figura 7) dentro del contexto de un Sistema Nacional del Espacio Aéreo (*NAS*), que involucra tanto a los sistemas propios de las aeronaves como también a los de las estaciones meteorológicas, los radares de control aéreo, las ayudas para la navegación, las estaciones de control del tráfico aéreo, las torres de control de los aeropuertos, etc.

Entre las preocupaciones más importantes respecto a la Seguridad, dentro de las actividades de los subsistemas del *NAS*, figuran:

- Mantener la comunicación entre las estaciones de control y las aeronaves
- Suministrar en forma precisa la información necesaria para el guiado de aproximación por instrumentos
- Suministrar los reportes meteorológicos sobre clima adverso en las rutas de vuelo
- Controlar el tráfico aéreo, evitando riesgos de colisiones

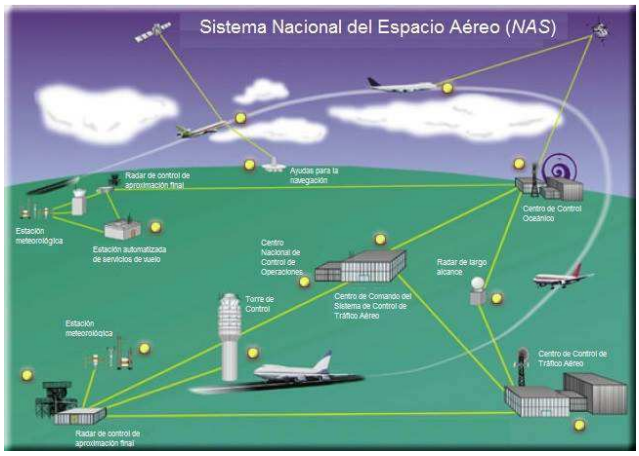


Figura 7: Elementos del Sistema Nacional del Espacio Aéreo [90]

Una explicación precisa de los objetivos que se deben cumplir en el funcionamiento del *NAS*, y que pone a la seguridad entre las principales tareas, es la definición de los objetivos

de una agencia como la *FAA*: "*…su misión principal es asegurar que el transporte aéreo se realice en forma segura, ordenada y eficiente a través de los Estados Unidos. La capacidad para cumplir con esta misión depende de la exactitud y confiabilidad del Sistema Nacional de Control de Tráfico Aéreo (ATC), una vasta red de computadoras, software y equipos de comunicaciones que proveen información a los controladores de tráfico aéreo y a las tripulaciones de vuelo para asegurar el movimiento seguro y expeditivo de las aeronaves. La red ATC* (ver Figura 8) *de la FAA es un enorme y complejo conjunto de sistemas interrelacionados que incluye a los Sistemas de Navegación, Vigilancia y Meteorología, y al procesamiento y visualización de la información en forma automatizada que se encuentran o están asociadas con cientos de instalaciones de la red. Una compleja red de comunicaciones, que transmiten en forma separada tanto voz como datos digitales, interconectan a estos sistemas e instalaciones…*" [91]

Entre los sistemas de las estaciones de control de tráfico aéreo dedicados a la gestión de tráfico (*ATM*), los más destacados son los Sistemas de Radares Automatizados, los cuales actualmente pueden soportar hasta 10.000 rastreos en simultáneo proporcionando información para 200 pantallas de los controladores de vuelo. Además, continuando con William S. Greenwell (2003), incluyen alertas de conflicto por colisiones y alertas de altitud mínima de seguridad (*MSAW*). [92]

Figura 8: Posición de Control de Tráfico Aéreo (*ATC*) [93]

Entre los sistemas de a bordo, en los aviones modernos se destacan especialmente los Sistemas Electrónicos de Instrumentos de Vuelo (*EFIS*), en los que la tecnología de visualización utilizada es electrónica en lugar de electromecánica, constando normalmente

de Pantalla Principal de Vuelo (*PFD*), pantallas multifunción (*MFD*) y una pantalla para el Sistema de Indicación de Motor y Aviso a la Tripulación (*EICAS*).

Figura 9: *EFIS* en avión comercial Airbus A380 [94]

La *PFD* muestra toda la información crítica para el vuelo, incluida la velocidad aerodinámica, altitud, rumbo, actitud, velocidad vertical y guiñada. Está diseñada para mejorar la conciencia de la situación de un piloto, mediante la integración de esta información en una sola pantalla en lugar de seis diferentes instrumentos analógicos, reduciendo la cantidad de tiempo necesario para controlar los instrumentos. También aumenta la conciencia situacional de la tripulación del avión al alertar de condiciones inusuales o potencialmente peligrosas, por ejemplo, de baja velocidad, alta tasa de descenso, etc., cambiando el color o la forma de la pantalla o alertas de audio.

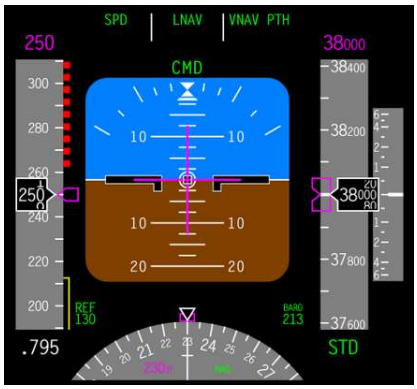


Figura 10: *PFD*, con Indicador de Situación Horizontal (*HSI*), rumbo, velocidad y altitud [95]

Las Pantallas Multifunción (*MFD*) muestran la navegación y la información en tiempo real de múltiples sistemas (ver Figura 11). Están pensadas como cartas digitales, donde las tripulaciones pueden superponer informaciones diferentes sobre un mapa o un gráfico, por ejemplo el plan de ruta actual de la aeronave, información de la meteorología provista por el radar de a bordo o bien por los radares meteorológicos en tierra, el espacio aéreo restringido, el tráfico de otras aeronaves, etc. También pueden mostrar información acerca de los sistemas de la aeronave, tales como el combustible y los sistemas eléctricos, pudiéndose cambiar el color o la forma de los datos para alertar a la tripulación de situaciones peligrosas.

Uno de los sistemas integrados más importantes desde el punto de vista de la seguridad es el Sistema de Alerta de Tráfico y Evasión de Colisión (*TCAS*), el cual puede no sólo alertar sobre la proximidad de tráfico en la zona sino también proponer una corrección inmediata en la trayectoria.

Figura 11: *MFD* con carta de navegación e indicación de tráfico aéreo en avión comercial *Boeing 747-400* [96]

Otro sistema muy importante también integrado es el Sistema de Alerta de Proximidad al Suelo (*GPWS*), el cual alerta a la tripulación si la aeronave se encuentra en un peligro inmediato de colisión contra el suelo u otro obstáculo.

El *EICAS* muestra información sobre los sistemas de la aeronave, incluyendo el combustible, los sistemas eléctricos e información del motor. Generalmente, están diseñados para imitar a los indicadores convencionales analógicos incorporando lecturas digitales de los parámetros, permitiendo que la tripulación pueda visualizar en forma gráfica la información considerada crítica para la seguridad del vuelo y alertando acerca de situaciones inusuales o peligrosas. Puede contener información de, por ejemplo, la

temperatura y presión del aceite en los motores, las revoluciones por minuto, distribución y consumo del combustible, temperatura y prestaciones de las turbinas, indicador de las superficies de mando primarias y secundarias, información del estado de los sistemas hidráulicos, neumáticos, eléctricos, deshielo, etc. informando asimismo de errores detectados y sugiriendo la acción a tomar por parte de la tripulación.

Figura 12: *EICAS* en avión *jet* ejecutivo *Cessna Citation X* [97]

Como en otros Sistemas Críticos para la Seguridad, la redundancia y diversidad están presentes tanto en el *hardware* como en el *software* utilizados. Las técnicas de *votación* se utilizan para detectar discrepancias o desacuerdos entre los canales de información.

Al igual que en otros casos, los sistemas están diseñados para ser a prueba de fallos, lo cual para las aeronaves significa que el sistema debe ser capaz de detectar los fallos y reaccionar en consecuencia, asegurando que ante una falla en un régimen crítico del vuelo, como es el despegue por ejemplo, permitirá igualmente que el piloto pueda continuar con el mismo en forma segura. Por esta razón, por ejemplo según Ian Moir y Allan Seabridge (2001), si las válvulas de combustible fallaran, lo hacen quedando en la posición abierta. [98]

3.7 Sistemas Críticos para la Seguridad en Actividades Espaciales

"El dominio del espacio por el hombre es la mayor aventura y la más inspiradora empresa"
Wernher von Braun

Desde el comienzo de la era espacial en 1957, con el lanzamiento del satélite *Sputnik-1* por parte de la entonces Unión Soviética, y durante la década de 1960 en los Estados Unidos, se comenzó a utilizar a las computadoras para controlar los sistemas aplicados a las actividades de exploración espacial, construyéndose, según detalla el *National Research Council* (1993), sistemas en tiempo real para la gestión de las complejas naves espaciales y sus sistemas de soporte en programas tales como *Mercury*, *Gemini*, *Apollo* y el Transbordador Espacial (*Space Shuttle*), entre otros, por parte de la *NASA*. [99]

La exploración espacial constituye uno de los hitos más importantes de la humanidad, desde el punto de vista de la complejidad de la ingeniería aplicada. Nada de esto podría haber sido posible sin los sistemas de control sumamente sofisticados que gobiernan las prestaciones de las naves espaciales y el desarrollo de *software* para el desempeño de las misiones en forma segura y eficiente. Aun así, el espacio sigue siendo, de acuerdo con Robert Dewar (2002), "*…un ambiente duro e inflexible donde la más mínima falla puede conducir a la muerte…*" [100]

El *software* de las naves espaciales, satélites y sondas de exploración interplanetarias, controla la mayoría de los aspectos relacionados con el ascenso, descenso, navegación espacial y operaciones en órbita de las mismas (*AOCS*), según lo detalla el *National Research Council* (1993), en tiempo real y tomando las acciones apropiadas en fracciones de segundo. [101]

Las características propias de los sistemas y computadoras de las naves espaciales, establecen R. H. Pierce et al. (1997), incluyen una arquitectura restringida y recursos muy limitados, con un complicado mantenimiento correctivo del *software*, necesitándose la supervisión y control de numerosas actividades de a bordo y contando con requerimientos de tiempos muy restringidos en donde la exactitud y puntualidad de la ejecución es crítica para la correcta ejecución de los sistemas. [102] Adicionalmente, agrega Jens Eickhoff (2012), deben ser capaces de permitir un control interactivo de la nave espacial, tanto en forma remota como mediante controles autónomos y automatizados e incluir las rutinas para el manejo de la carga útil transportada. [103]

Además de las actividades propias de las naves espaciales controladas por *software*, existe otro aspecto en el cual el mismo es vital para la seguridad y éxito de las operaciones, el cual es el *software* implementado en los Centros de Control de Misión (*Mission Control Center*). Estos sistemas, de acuerdo con Ljerka Beus-Dukic (2001), se encargan del apoyo operativo a las misiones espaciales, incluyendo generalmente una base de datos principal de configuración de las aplicaciones (por ejemplo, la Estación Espacial Internacional cuenta con unos 2 millones de componentes bajo control de la configuración), entornos para el desarrollo de *software*, soporte para simulación y pruebas, monitoreo y control técnico de la misión, control de actitud y órbita, monitoreo de la carga útil, etc. [104]

Figura 13: Centro de Control de Misión *Mercury*, *NASA*, año 1962 [105]

Figura 14: Centro de Control de Misión *CryoSat-2*, Agencia Espacial Europea, año 2010 [106]

Desde los programas desarrollados en lenguaje *Assembler*, para las primeras misiones (*Gemini, Mariner, Apollo*), en los cuales debían aplicarse técnicas que aprovecharan al

máximo los pocos recursos disponibles de las *CPU* originales, pasando por los ejemplos posteriores en cuanto a la codificación de *software* (*Space Shuttle*, *Pioneer*, *Voyager*), hasta los ejemplos más recientes como la misión *Curiosity*, el *software* ha evolucionado hacia lenguajes de más alto nivel, tales como *Ada*, *C* y *C++*, entre otros, que aprovechan las mejoras en el *hardware* disponible y permiten gestionar la complejidad creciente en los requerimientos de las misiones.

El *software* utilizado se clasifica, en general, según lo detallan Malcolm Macdonald y Viorel Badescu (2014), en cuatro categorías: [107]

- *Software* de arranque (*boot*): el *software* de las naves espaciales se ejecuta típicamente en un ordenador de placa reducida (*SBC*) con un procesador, *hardware* de a bordo y periféricos. El *software* de arranque configura al *SBC* para que ejecute una aplicación de *software*, incluyendo la inicialización de los registros del procesador, la preparación de la memoria y la configuración de los dispositivos periféricos

- Comando y manejo de datos (*C&DH*): controla el flujo de datos en la nave espacial, proveyendo las entradas a través del procesamiento de comandos y las salidas en forma de información telemétrica para el personal del Centro de Control de Misión. También maneja el direccionamiento de los datos hacia el resto de los subsistemas tanto en la nave espacial como en áreas externas a través de *bus* de datos. Otras funcionalidades incluyen el registro y recupero de datos en dispositivos de estado sólido (*SSR*) y ejecutar las operaciones para la gestión de fallos que garanticen la seguridad de la nave espacial

- Guiado, Navegación y Control (*GN&C*): sus funciones primarias son mantener la actitud de la nave espacial, ejecutar las maniobras de propulsión necesarias para controlar su trayectoria y proveer una función de navegación que permita el conocimiento de la posición dentro de un marco de referencia dado. Dado que es un sistema de control, la ejecución de sus algoritmos se encuentra estrechamente controlada y se ejecutan a una frecuencia fija y conocida. Así, por ejemplo la frecuencia de 20 Hz es usada para el control y la de 1 Hz para el guiado y la navegación, los cuales son sincronizados con algún tiempo estándar, como por ejemplo el Tiempo Transcurrido de la Misión (*MET*) o el Tiempo Universal Coordinado (*UTC*). El *software* toma los datos desde los sensores del subsistema *GN&C* en coordinación con el subsistema *C&DH*, utilizándolos para ejecutar el

código y direccionar las acciones destinadas a los actuadores (ruedas de reacción, barras de torsión magnética y propulsores de control de actitud). Los sensores incluyen, entre otros, a la Unidad de Medición Inercial (*IMU*), el Rastreador Estelar (*ST*) y el Sensor Solar (*SS*)

- Carga Útil (*Payload Software*): En general, para el manejo de la carga útil transportada se cuenta con un procesador dedicado, con el *software* integrado en la arquitectura general, siendo responsable por la operación de los instrumentos que operan la carga útil como también de enviar y recibir datos desde y hacia la computadora principal de la nave

Además de las categorías mencionadas, de acuerdo con Daniel L. Dvorak (2009), podemos incluir otras que se han incorporado recientemente, debido al uso de los modernos astromóviles (*rover*) para la exploración planetaria (*Pathfinder*, *Spirit*, *Opportunity*, *Curiosity*, etc.): [108]

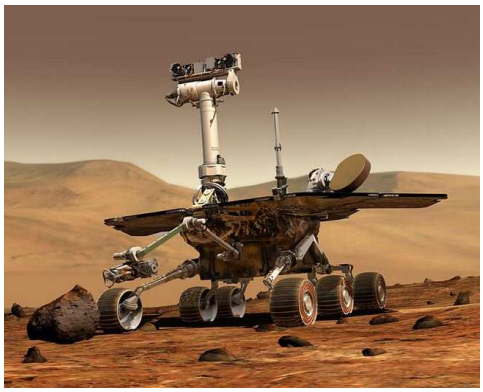


Figura 16: *Mars Exploration Rover* [109]

- Entrada, Descenso y Aterrizaje (*EDL*): Controla las operaciones desde que la nave espacial alcanza la atmósfera del planeta a explorar hasta que la astronave se encuentra sobre la superficie del planeta, incluyendo el uso de sensores atmosféricos

- Movilidad y navegación en la superficie (*SM*): Incluye la conducción del astromóvil, medición del movimiento real, interpretación del terreno y selección de objetivos de interés, entre otras funciones

- Operaciones científicas en la superficie (*SO*): Incluye la recolección y análisis de muestras de suelo y rocas, búsqueda de evidencias químicas y orgánicas,

caracterización del clima y la geología, gestionando el uso de cámaras, espectrómetros, detectores de radiación y sensores ambientales

La complejidad del *software* utilizado y su evolución en el tiempo (ver Gráfico 7) junto con el costo asociado puede ser ejemplificado con algunas de las características del *software* desarrollado para el proyecto del Transbordador Espacial, por parte de la *NASA*. Contenía originalmente unas 400.000 líneas de código (*HAL/S*), en más de 1.500 unidades compilables, con un *software* de respaldo (*backup*) con 90.000 líneas de código, insumiendo unos U$S 100 millones anualmente en su mantenimiento y actualización. Durante unos 9 años, desde 1983 hasta 1991, el *software* recibió 14 actualizaciones, las cuales modificaron un total de 152.000 líneas de código, aproximadamente. Cada actualización, llamadas Incrementos Operacionales (*OI*), insumía un proceso de desarrollo de *software* de alrededor de 28 meses, con un costo de U$S 6.000 por línea de código, en promedio, según lo detalla el *National Research Council* (1993). [110] Un proyecto mucho más reciente como el de la nave espacial *Orion* (*MPCV*), continuando con Daniel L. Dvorak (2009), sólo en su sistema de *GN&C* contiene más de 1 millón de líneas de código. [111]

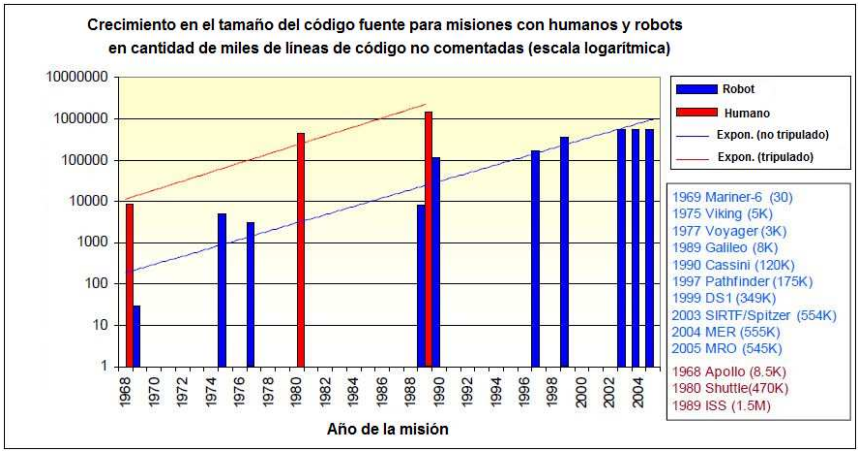


Gráfico 7: Evolución del tamaño del *software* en actividades espaciales [112]

Se prevé que adicionalmente el *software* pueda gestionar en el futuro otras tareas críticas tales como: [113]

- Encuentro y acoplamiento (*Rendezvous and docking*)
- Ascenso guiado

- Razonamiento basado en modelos

La seguridad del *software* crítico en las actividades espaciales requiere que sea capaz de gestionar, principalmente, dos tipos de incidentes:

- Reconocer y manejar adecuadamente problemas relacionados con un funcionamiento defectuoso del *hardware* utilizado para las funciones de control
- Evitar que el *software* pueda generar respuestas incorrectas o respuestas extemporáneas que pudieran contribuir a que el sistema alcanzase un estado peligroso

Para ello, como en otros Sistemas Críticos, se utiliza el concepto de redundancia y diversidad, por ejemplo en la forma de *redundancia modular múltiple* (*NMR*), en ciertas operaciones críticas como durante el despegue y la reentrada, contando con hasta 4 computadoras ejecutando los procesos correspondientes y una quinta computadora ejecutando una versión diferente del *software* (*BFS*).

Otra técnica utilizada es la del *análisis de peligros* sobre el Sistema (*system-hazard analysis*), mediante la cual los peligros que se identifiquen pueden ser trazados hasta el requerimiento de *software* en particular, y desde allí hasta el módulo de *software* correspondiente, para efectuarles un análisis y tareas de prueba especiales a fin de proveer una confiabilidad adicional.

De acuerdo con Malcolm Macdonald y Viorel Badescu (2014), con el incremento de las actividades de exploración espacial hacia lo que se conoce como el espacio profundo (*deep space*), con misiones como *Voyager*, *Galileo*, *Magellan* y *Cassini*, se hizo necesario el desarrollo más intensivo de sistemas *software* autónomos dado que las restricciones de las misiones requerían que las sondas espaciales fueran capaces de tomar decisiones críticas a distancias que hacen imposible la intervención humana en tiempo real. [114]

La tecnología espacial cuenta hoy con múltiples usos y aplicaciones: [115]

- Civil: meteorología, búsqueda y salvamento, demostración tecnológica, actividades educativas, detección de objetos cercanos a la Tierra, etc.
- Comercial: comunicaciones satelitales y transmisiones, sistemas de posicionamiento global (*GPS*), observación de recursos naturales, turismo espacial, etc.

- Científico: astronomía y ciencias del espacio, ciencias de la Tierra, etc.
- Defensa: comunicaciones militares, vigilancia, reconocimiento electrónico, sistemas antimisiles, etc.

Un concepto que utiliza el *software* considerado crítico para las Actividades Espaciales es el de *perennidad*, agrega Ljerka Beus-Dukic (2001), que es la capacidad de un producto para durar por el tiempo de vida completo de un proyecto espacial (por ejemplo, 15 años). [116]

Las misiones espaciales requieren de un enfoque de mantenimiento y soporte del *software* a largo plazo. En algunas misiones, es extremadamente importante la posibilidad de efectuar actualizaciones del código posterior al lanzamiento a fin de corregir cualquier problema que surja en la fase de operación o para realizar cambios debido a variaciones en los requerimientos del entorno y mejorar ciertas funciones. [117]

3.8 Sistemas Críticos para la Seguridad en Defensa

"Lo único que puedes hacer con un F-22 sin software, es sacarle una foto"
De un General de la USAF [118]

Al igual que en otros sectores de la sociedad, el uso de *software* se ha convertido en algo habitual en todos los aspectos de las Organizaciones Militares. Actualmente, de acuerdo con la publicación especializada *PR Newswire* (2006), las actividades relacionadas con la Defensa están atravesando una rápida transformación dado que los dispositivos utilizados requieren niveles crecientes de conectividad, interoperabilidad y confiabilidad en combinación con un menor tamaño, peso y consumo de energía. [119]

El *software* se ha convertido en algo esencial para un amplio rango de operaciones y capacidades de los Sistemas Militares, incrementándose su escala, su complejidad y el rol protagónico que tiene en la capacidad funcional. Por ejemplo, en un avión de combate de la década del '60, como el *McDonnell Douglas* F-4 *Phantom*, el porcentaje de funciones ejecutadas por el *software* de control era de sólo un 8% y en un avión de combate de la década del '80, como el *General Dynamics* F-16 *Fighting Falcon* ese porcentaje se había incrementado hasta un 45%. En un avión de combate moderno como el *Lockheed Martin* F-22 *Raptor*, de acuerdo con William Scherlis et al. (2011), esa cantidad se incrementa hasta un 80% del total de las funciones del sistema. [120] En las aeronaves de combate modernas, aun las actividades básicas relacionadas al vuelo no podrían ser posibles sin la sofisticación

de los sistemas computarizados, debido a que están diseñadas para ser aerodinámicamente inestables.

Figura 17: Lockheed Martin F-22 *Raptor* [121]

El *software* incrementa la capacidad, la integración y la agilidad en los Sistemas de Defensa, teniendo el potencial de poder adaptarse a distintos tipos de amenazas, entornos operativos y plataformas tecnológicas. El incremento de las capacidades provisto por el *software* se ha convertido en una ventaja fundamental que proporciona una superioridad estratégica única para las operaciones militares, siendo un componente fundamental del poderío militar.

Una definición amplia de lo que se entiende por *software* para la Defensa, nos detalla Capers Jones (2002), incluye a una cantidad de subclases: [122]

- *Software* asociado con los sistemas de armas (misiles, torpedos, artillería, aeronaves de combate, etc.)
- Sistemas de Comando, Control, Comunicaciones y Computación (C4)
- Aplicaciones para la logística

El *software* de los Sistemas de Armas, según el *Department of Defense* (1990), comprende al *software* que está contenido en: [123]

- Los Sistemas de Armas en sí mismos
- El equipamiento para pruebas y mantenimiento

- El equipamiento para entrenamiento

El principal atributo que distingue al *software* para la Defensa de otros tipos de *software* es la adhesión de los mismos a ciertos estándares definidos por los Departamentos de Defensa correspondientes.

Las actividades relacionadas con la Defensa fueron el principal motor en el desarrollo de las aplicaciones de computación, en particular a partir de la década del ’50. El Departamento de Defensa de los Estados Unidos, establece William Farr (2003), pasó de utilizar una cantidad de líneas de código fuente de una magnitud del orden de 100.000 (10^5 *LOCS*) en la década del ’50 a unos 1.000 billones (10^{15} *LOCS*) actualmente. [124]

La siguiente lista de Sistemas para las actividades de la Defensa, de acuerdo con el *Department of Defense* (1998) y *NATO* (1997), puede ser considerada representativa de la criticidad para la seguridad de las mismas: [125, 126]

- Cualquier Sistema relacionado con el uso de munición que controle o influya directamente en la preparación, armado, lanzamiento o disparo, trayectoria de vuelo o detonación de la munición, incluyendo la identificación, selección y designación del objetivo

- Cualquier Sistema que controle o influya directamente en el movimiento de los montajes de las armas, lanzadores u otros equipos, especialmente con respecto a la seguridad en la precisión y disparo de dicho equipamiento

- Cualquier Sistema de combate basado en computadora

- Cualquier Sistema que controle o influya directamente en el movimiento de las municiones y/o materiales peligrosos

- Cualquier Sistema que supervise el estado de otro Sistema por razones de seguridad

- Cualquier Sistema que controle, regule o contenga fuentes de energía potencialmente peligrosas

- Cualquier Sistema utilizado para calcular datos críticos para la seguridad (incluyendo el *software* de aplicaciones que podrían no estar conectadas o controlando directamente a un sistema de *hardware* crítico para la seguridad, como los programas de análisis de resistencia)

- Cualquier Sistema que recopile, almacene, manipule y muestre o genere reportes de datos que pueden ser críticos para la seguridad
- Cualquier Sistema que controle total o parcialmente el movimiento de un vehículo (por ejemplo, una nave, aeronave, vehículos terrestres, objetos guiados por radar, etc.)
- Cualquier Sistema que controle total o parcialmente el movimiento de piezas de equipos que sean potencialmente peligrosas para las personas en sus proximidades
- Cualquier Sistema que detecte peligros y/o muestre información relativa a la protección del Sistema
- Cualquier unidad de *software* o módulo que maneje o responda a las detecciones de fallos y restaure la seguridad del Sistema
- Cualquier *software* para la interrupción del procesamiento, para la interrupción de los esquemas y rutinas de prioridades, los cuales activen o desactiven las interrupciones
- Cualquier unidad de *software* o módulo que tenga control autónomo sobre el *hardware* crítico para la seguridad
- Cualquier unidad de *software* o módulo que controle o influya directamente en el movimiento de componentes de *hardware* potencialmente peligrosos o que inicien acciones críticas para la seguridad

Los sistemas complejos basados en *software* poseen una ubicuidad única en las actividades relacionadas con la Defensa. Por ejemplo, de acuerdo con Siddhartha R. Dalal et al. (2003), el Sistema de Combate *AEGIS*, un sistema de armas integrado que controla el funcionamiento de las armas de ataque y defensa de los buques de diversas marinas, entre otras Estados Unidos, Japón, España, Noruega, Australia y Corea del Sur, posee un total de más de 2.000 *KLOC* (1.200 *KLOC* para interfaces de visualización, 385 *KLOC* para pruebas, 266 *KLOC* para el armamento, 110 *KLOC* para entrenamiento y 337 *KLOC* para comando y toma de decisiones). [127]

Figura 18: Centro de Información de Combate (*CIC*)-Sistema *AEGIS*, destructor clase *Arleigh Burke*, *U.S.Navy* [128]

A partir de los requerimientos específicos de la industria para la Defensa, el Departamento de Defensa de los Estados Unidos encargó la creación de dos lenguajes de programación que fueron fundamentales para el desarrollo del *software* para la Defensa:

- *JOVIAL*, un acrónimo de "*Jules Own Version of the International Algorithmic Language*", un lenguaje de alto nivel similar a *ALGOL* desarrollado en 1959, pero especializado para el desarrollo de sistemas embebidos en aeronaves de combate, satélites de comunicaciones, misiles de crucero, naves de combate, sistemas de control y defensa aérea, guiado de armas, radares, etc.

- *Ada*, un lenguaje multipropósito, orientado a objetos y concurrente, diseñado con la seguridad como prioridad y dirigido a la reducción de errores comunes y problemáticos de descubrir, desarrollado en 1979 como ganador de un concurso público encargado por el *DoD*, el cual se convirtió en un estándar para el *DoD* y la *OTAN* hasta el año 1997 en que comenzaron a utilizarse además productos *COTS*

Al incrementarse la dependencia de los sistemas de Defensa en el *software*, surgen otras preocupaciones, no sólo respecto a la Seguridad, sino también a los costos asociados con la complejidad y tamaño de las aplicaciones. El avión de combate de quinta generación *Lockheed Martin* F-35 *Lightning II* posee unos 24 millones de líneas de código, unos 9 millones de líneas de código mayor a lo esperado, es decir un 60% más de lo planeado originalmente. El *software* en este avión es tan crítico y fundamental, que se convirtió en el eje de una disputa entre los Estados Unidos y potenciales compradores del avión como el

Reino Unido, Australia y Turquía a raíz de la negativa estadounidense en proporcionar el código fuente como parte de la transferencia de tecnología.

Figura 19: *Lockheed Martin* F-35 *Lightning II* [129]

El *software* del *JSF* (*Joint Strike Fighter*), es uno de los proyectos más grandes y complejos en la historia del *DoD*, proporcionando las capacidades esenciales de esta aeronave, pero el crecimiento en el tamaño y complejidad del *software* ha generado retrasos y sobrecostos importantes. Es posible que, de acuerdo con Michael J. Sullivan (2012), las complejidades asociadas con este tipo de *software* crítico de última generación no sean aun correctamente estimadas en todos los proyectos ya que "*…el crecimiento del tamaño del software del JSF no es muy diferente al de otras adquisiciones recientes para la Defensa, las cuales han experimentado incrementos en el tamaño del software de entre el 30% al 100%…*" [130]

Los llamados Sistemas Ciber-Físicos para las actividades de la Defensa, esto es aquellos sistemas que integran los procesos físicos e informáticos en tiempo real, presentan grandes desafíos en la actualidad. Estos sistemas distribuidos en tiempo real, con millones de líneas de código y controlados por una gran cantidad de sensores que informan en una variedad de escalas de tiempo para distintos sistemas de armas y protocolos de control de dispositivos, tienen en común, detallan Joan D. Winston y Lynette I. Millett (2007), dos características que constituyen todo un reto: la *incertidumbre* y la *escala*: [131]

- Gran escala, con decenas de miles de requerimientos funcionales y de rendimiento, millones de líneas de código y cientos de elementos para la configuración del *software*
- Requerimientos de funcionamiento simultáneos que entran en conflicto, procesamiento en tiempo real y recuperación de errores limitada

- La diversidad que existe en la implementación, con diferentes lenguajes de programación, sistemas operativos, arquitecturas complejas, ciclos de vida del sistema de entre 20 y 40 años y normas de certificación rigurosas

- Complejidad de la arquitectura, con el concepto de sistemas de sistemas, redes cableadas mixtas, inalámbricas o satelitales, servidores de varios niveles, dispositivos personales y estaciones de trabajo y configuraciones del sistema múltiples

Estos sistemas son desafiantes, complejos y costosos. Frecuentemente, los desafíos pueden simplificarse difiriendo o eliminando capacidades, lo cual puede ayudar a limitar los costos y las fechas de entrega, pero aun así los sobrecostos y demoras siguen siendo comunes. La Oficina de Responsabilidad Gubernamental de los Estados Unidos (*GAO*) informó que, durante el período fiscal del año 2003, el *DoD* gastó 21.000 millones de dólares en investigación, desarrollo, pruebas y evaluación (*RDT&E*) para los nuevos sistemas de combate, de este gasto cerca del 40% fue utilizado en retrabajos sobre el *software* para corregir los problemas relacionados con la calidad. [132]

Los problemas identificados están relacionados con el uso de "*…enfoques que no están basados en el conocimiento o que no son evolutivos, intentando desarrollar capacidades revolucionarias en un solo paso, causando incertidumbres tecnológicas y de diseño significativas y generando grandes sobrecostos y demoras…*" [133]

3.9 Sistemas Críticos para la Seguridad en Industria

"La prevención de accidentes no debe entenderse como una regulación exigida por la ley, sino como un precepto de la responsabilidad humana y de la economía "
Werner Von Siemens, 1880

La sociedad actual está basada en una industria moderna en la cual la automatización industrial es un factor clave. La seguridad de estos sistemas automáticos ha evolucionado en forma constante, debido fundamentalmente a los avances logrados en la microelectrónica. Además, la opinión pública exige nuevos requerimientos para lograr procesos de fabricación y productos más seguros, existiendo nuevas legislaciones para la protección del medio ambiente y la seguridad en la producción.

La globalización de la industria lleva a un incremento en la competencia y la exigencia de lograr buenos resultados económicos por parte de las empresas, resultando imperativo el incremento de la eficiencia y la seguridad en la producción.

Las empresas tienen diversas razones económicas para tener sistemas de producción que sean seguros, por ejemplo, de acuerdo con la compañía *Siemens AG* (2011): [134]

- Evitar los costos directos e indirectos asociados a un incidente que puede causar heridas al personal, mejorando los resultados económicos (*ROI*)
- Mejora de la productividad por la mayor disponibilidad de la maquinaria de producción (*OEE*)
- Prolongar la vida útil de los sistemas

La seguridad en la automatización, requiere la recolección de información, su procesamiento y, en función de los resultados, actuar sobre los procesos para que estos ejecuten las tareas previstas en forma correcta y segura, de acuerdo con Josef Börcsök (2004), utilizando un conjunto de sensores, actuadores y buses de red. [135]

Los Sistemas Críticos para la Seguridad en las actividades industriales se encuentran generalmente asociados a los llamados Sistemas de Fabricación Integrados (*IMS*), un conjunto de máquinas que realizan una función de fabricación específica, por ejemplo, menciona Robin Carver (2010): líneas de embotellado, líneas de envasado, líneas de empaquetado, líneas de montaje de componentes, sistemas de paletización, etc. [136] Al *software* de aplicación, agregan Johan Hedberg et al. (2011), se lo conoce como *Software* de Aplicación Relacionado con la Seguridad (*SRASW*). [137]

Los sistemas programables que controlan a este conjunto de maquinaria, de acuerdo con Timo Malm y Marita Hietikko, pueden clasificarse en: [138]

- Controladores Lógicos Programables seguros (*PLCs*), buses de seguridad y redes de comunicación de seguridad
- Sistemas distribuidos con controladores, por ejemplo: microcontroladores, *PLCs*, *PC* industriales, matrices de puertas programables (*FPGA*), circuitos integrados para aplicaciones específicas (*ASICS*)

- *PLC* para control y supervisión y un sistema cableado redundante para funciones específicas como, por ejemplo, una parada de emergencia

- Sistemas de control con programación de parámetros, específicos para maquinarias automatizadas, como por ejemplo: máquinas-herramientas, prensas, máquinas de papel, etc.

- Sistemas de control de la maquinaria con un entorno de *software* de aplicación, específicos para *robots* industriales y máquinas-herramientas

- Sistemas de control para máquinas autónomo como, por ejemplo, robots con generación de código automatizada a partir de una fotografía o de una imagen de Diseño Asistido por Computador (*CAD*), vehículos guiados automáticos (*AGV*) con tareas automatizadas

Las arquitecturas de *software* utilizadas en estos Sistemas Críticos para la Seguridad, además de los ya vistos conceptos de *redundancia* y *diversidad*, utilizan dos formas sencillas de implementar la tolerancia a fallos como son la *comprobación de validez* (*sanity check*) y los mecanismos de *reintento de recuperación de fallos* (*re-try*). La comprobación de validez permite verificar la validez o verosimilitud en la salida de un componente específico y se utiliza para detectar fallos de valores. El reintento de recuperación de fallos permite que, ante la detección de una falla, el sistema reintente la ejecución del requerimiento.

Una mecanismo de seguridad adicional son los conocidos como circuitos de vigilancia (*watchdog circuits*), los cuales pueden generar un reinicio del sistema como una medida de recuperación de fallos, en caso que el sistema se encuentre bloqueado.

Los Sistemas Críticos para la Seguridad en la Industria ejecutan una serie de funciones de seguridad a través de los siguientes subsistemas (ver Figura 20), de acuerdo con la compañía *Siemens AG - Digital Factory* (2015): [139]

- Detección (interruptor de posición, parada de emergencia, dispositivos de detección de presencia *light curtains*, etc)

- Evaluación (control de seguridad *fail-safe*, dispositivo de seguridad *relay*, etc)

- Reacción (contactor, convertidor de frecuencia, etc)

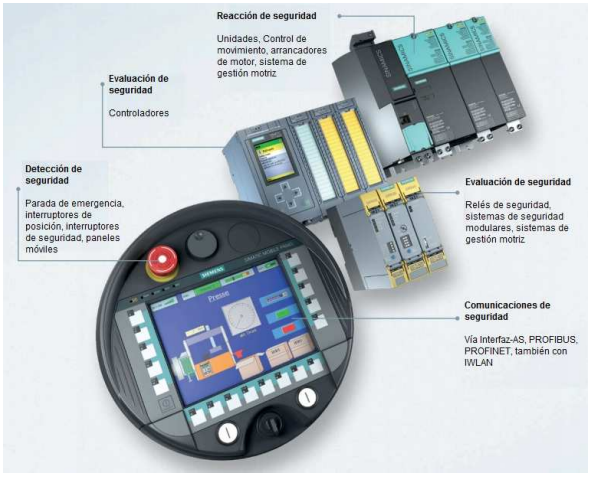


Figura 20: Seguridad Integrada dentro del Sistema de Automatización, *Siemens AG* [140]

Las funciones de Seguridad pueden clasificarse en las siguientes categorías: [141]

- Funciones para efectuar un apagado seguro de la máquina sin necesidad de desconectar el suministro de energía:
    - Desconexión segura del par de fuerza motriz (*STO*): esta función previene cualquier liberación del par de fuerza en el eje motor, posterior al apagado
    - Parada de Operación Segura (*SOS*): en esta función, al contrario que la *STO*, no previene la liberación posterior del par de fuerza pero la máquina permanece en posición y es controlada hasta su detención completa
    - Parada de seguridad Tipo 1 (*SS1*): esta función activa los frenos de la máquina, antes de la activación de la función *STO*. Las unidades con una energía cinética alta pueden ser detenidas rápidamente, en caso de peligro
    - Parada de seguridad Tipo 2 (*SS2*): similar a la función *SS1*, esta función activa los frenos de la máquina, sin embargo en este caso la función *SOS* es ejecutada hasta la detención completa
- Funciones para la vigilancia de los movimientos de la máquina:
    - Velocidad limitada de seguridad (*SLS*): esta función controla si la máquina excede las velocidades máximas especificadas

- Control de velocidad segura (*SSM*): esta función controla si la máquina funciona por debajo de la velocidad especificada.

- Dirección segura (*SDI*): esta función controla el cumplimiento de la dirección seleccionada del movimiento o rotación

- <u>Funciones para la vigilancia de la posición de la máquina</u>:

  - Posición limitada de seguridad (*SLP*): esta función previene que la máquina exceda un rango especificado de posiciones. Facilita la realización de un área de trabajo sobre ejes específicos, delimitando una zona de protección

  - Posición segura (*SP*): esta función transfiere los valores de la posición segura de la máquina a un control superior (*PLC*) donde se puede establecer un área de seguridad y control (*SCA*) determinada, enviando una señal de seguridad una vez que la máquina está posicionada dentro del rango de posición especificado

4. El desarrollo de *software* en los Sistemas Críticos para la Seguridad en Aviación

Los estándares utilizados en el desarrollo de *software* para Sistemas Críticos para la Seguridad agrupan generalmente una colección de documentos (reglas, guías, regulaciones, instrucciones, etc.) los cuales contienen información que será utilizada por los analistas y desarrolladores. El principal objetivo de estos estándares es la eliminación de inconsistencias, malentendidos y desacuerdos entre los distintos participantes durante la definición y desarrollo de un sistema, facilitando la concreción de las metas respecto a la Seguridad.

Cuando los primeros Sistemas Críticos para la Seguridad comenzaron a ser implementados, pronto surgieron inquietudes sobre cómo podríamos confiar en la Seguridad de un *software* sin poder realmente comprobar que es confiable o aún correcto. Bajo el auspicio de distintas organizaciones, comenzaron a definirse las técnicas y mejores prácticas para poder alcanzar una Seguridad apropiada en un Sistema basado en *software* y en cómo aplicar estos conceptos en forma metodológica para poder utilizarlos en diferentes circunstancias. En la actualidad los estándares, tanto nacionales como internacionales, son generalmente requeridos por la legislación vigente o por requerimientos contractuales, durante el desarrollo y certificación de los Sistemas.

El uso de determinados estándares permite que los Sistemas sean certificados por la autoridad competente, confirmando que el *software* ha sido desarrollado utilizando un modelo estructurado y aplicando los estándares correspondientes.

Las principales compañías fabricantes de aeronaves, como *Boeing* y *Airbus*, proyectaron un crecimiento geométrico en el tamaño y complejidad del *software* para la aviónica y se esperaba que, según lo establecido en la *Sixth Annual Conference on Software Assurance* (1991), los proveedores de *software* deberían lograr tasas de defectos de *software* cercanas a cero mediante el uso de nuevas tecnologías y metodologías. [1]

Crecimiento estimado de las líneas de código *software* (*SLOC*) en aeronaves

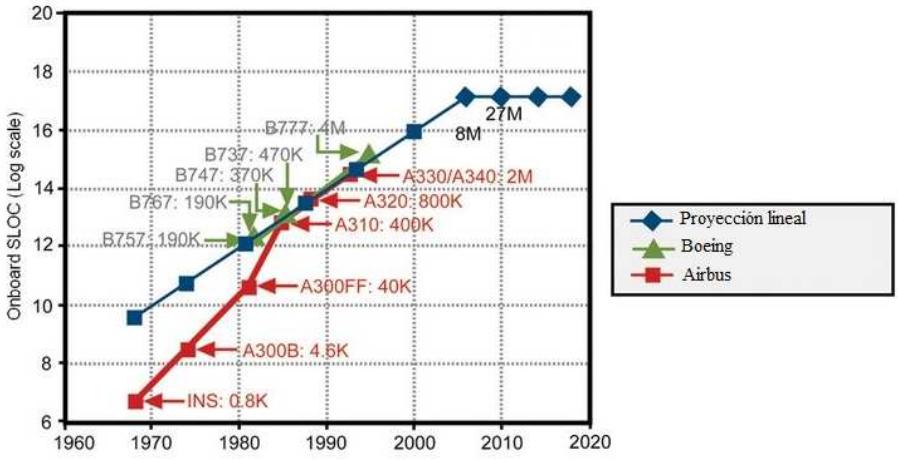


Gráfico 8: Crecimiento estimado de las líneas de código *software* en aeronaves

Focalizándonos en los estándares utilizados para los Sistemas Críticos para la Seguridad en Aviación, podemos mencionar entre las organizaciones que gobiernan estos estándares a:

- Comisión Radiotécnica para la Aeronáutica (*RTCA*): es una organización fundada originalmente en el año 1935 en los Estados Unidos, con sede en la ciudad de *Washington D.C.*, con carácter de comisión federal consultiva (*FAC*) y auspiciado por la *FAA*, que realiza recomendaciones para la comunicación, navegación y monitoreo de la gestión del tráfico aéreo a través de los sistemas *CNS/ATM*. Nuestro país es miembro de la *RTCA* a partir del año 2012, a través de la DIGAMC

- Instituto Estadounidense para la Aeronáutica y la Astronáutica (*AIAA*): es una sociedad profesional en el ámbito de la ingeniería aeroespacial fundada en los Estados Unidos en 1963, contando en la actualidad con más de 30.000 miembros y tiene su sede en la ciudad de *Reston, Virginia*. Nuestro país se relaciona con el *AIAA* a través de la AATE

- Sociedad de Ingenieros de Automoción (*SAE*): es una asociación profesional, fundada originalmente en 1905 en los Estados Unidos, que establece estándares para la ingeniería profesional en diversas industrias. Con sede en la ciudad de *Warrendale, Pennsylvania,* cuenta en la actualidad con más de 138.000 miembros en todo el mundo

- Organización Europea para la Seguridad de la Aeronavegación (*Eurocontrol*): es una organización europea civil y militar, fundada en 1960 e integrada actualmente por 41 estados miembros, que tiene como objetivo el desarrollo de sistemas seguros, eficaces y coordinados para el tráfico aéreo europeo. Tiene su sede central en la ciudad de Bruselas, Bélgica

- Autoridad de la Aviación Civil del Reino Unido (*CAA*): es una corporación estatutaria que supervisa y regula todos los aspectos de la aviación civil en el Reino Unido. Fundada en 1972, tiene su sede en el municipio de *Camden*, Londres

4.1 *DO-178B/DO-178C Software Considerations in Airborne Systems and Equipment Certification* (Consideraciones del *Software* en Sistemas Aéreos y en Certificación de Equipos)

Publicado originalmente en 1992 por la *RTCA, DO-178B* si bien es una norma de lineamientos técnicos, ha sido de hecho ampliamente utilizado durante muchos años como estándar para el desarrollo del *software* en los sistemas encargados de la aviónica de las aeronaves. En la actualidad, ha sido reemplazado a partir del año 2012 por la norma *DO-178C.*

Este estándar es utilizado principalmente como una herramienta para determinar si el *software* funcionará de manera confiable en un medio aeronáutico. Como en muchos otros estándares en diversas industrias, es del tipo descriptivo más que prescriptivo, es decir describe los objetivos de los procesos en lugar de establecer los métodos que deben utilizarse.

Los Niveles del *Software*, llamados Niveles de Garantía del Diseño (*DAL*), son determinados a través de un proceso de evaluación de la seguridad y análisis de peligros, examinando las consecuencias potenciales de una avería en el sistema. Las averías son categorizadas en función de sus efectos sobre la aeronave, la tripulación y los pasajeros, según detalla la compañía *AdaCore* (2014): [2]

Nivel	Condición de Avería	Consecuencias	Ejemplo	Tasa de Averías
A	Catastrófica	Puede causar que la aeronave se estrelle. Errores o pérdida de funciones críticas requeridas para el vuelo y aterrizaje en forma segura	Sistemas "*fly-by-wire*"	0,000000001 / hora
B	Peligrosa	Averías que tienen un gran impacto negativo en la seguridad o el rendimiento, o que reducen la habilidad de la tripulación para poder operar la aeronave debido a malestar físico o a una gran carga de trabajo, o que causen heridas serias o fatales entre los pasajeros	Gestión del combustible	0,0000001 / hora
C	Mayor	Averías significativas, pero con un impacto menor que las consideradas peligrosas y que pueden incrementar significativamente la carga de trabajo de la tripulación	Comunicaciones entre el piloto y *ATC*	0,00001 / hora
D	Menor	Averías detectadas, pero con un impacto menor que las consideradas mayores	Registro de datos del vuelo	0,001 / hora
E	Sin efecto	Averías que no tienen impacto en la seguridad, operaciones o en la carga de trabajo de la tripulación	Sistema de entretenimiento	N/A

Tabla 1: Categorización de averías

El enfoque de este estándar está basado en la formulación de objetivos apropiados y en la verificación que estos objetivos se hayan cumplido. La estructura de sus actividades es una jerarquía de procesos, existiendo tres niveles superiores de grupos de procesos, de acuerdo con la compañía *Esterel Technologies SA* (2012): [3]

- El proceso de planificación del *software*, que define y coordina las actividades del desarrollo e integración del mismo para un proyecto
- Los procesos de desarrollo de *software*, que producen el producto. Estos procesos son el proceso de requerimientos del *software*, el proceso de diseño del *software*, el proceso de codificación del *software* y el proceso de integración
- Los procesos integrales, que aseguran la corrección, control y confiabilidad en los procesos del ciclo de vida del *software* y sus salidas. Estos procesos son el de verificación del *software*, el de gestión de configuración del *software*, el de aseguramiento de la calidad del *software* y el proceso de certificación. Estos procesos son ejecutados concurrentemente con los procesos del desarrollo del *software*, a través del ciclo de vida del mismo

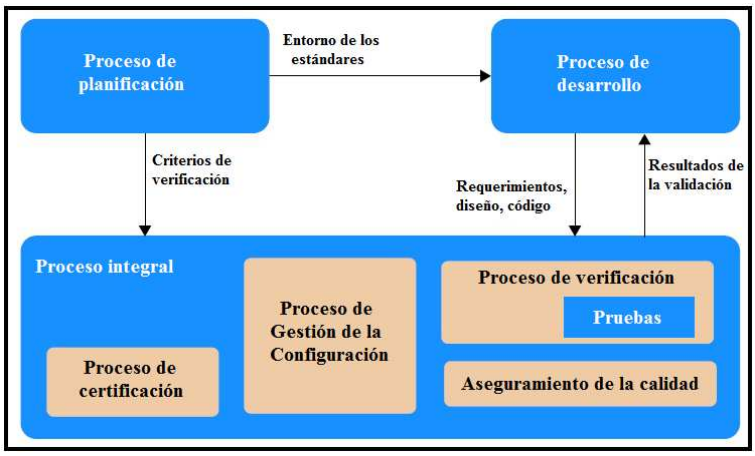


Gráfico 9: Estructura del proceso del ciclo de vida [4]

El proceso de desarrollo del *software* (ver Gráfico 10) está compuesto por:

- El proceso de requerimientos del *software*, el cual produce los requerimientos de alto nivel (*HLR*). Estos incluyen las especificaciones operacionales y funcionales, las restricciones de tiempo y memoria, las interfaces con el *software* y el *hardware*, los requerimientos de monitoreo de seguridad y detección de averías, como también los requerimientos de particiones

- El proceso de diseño del *software*, el cual produce los requerimientos de bajo nivel (*LLR*) y la arquitectura del *software* a través de una o más refinaciones de los *HLR*. Los *LLR* incluyen descripciones de las entradas y salidas, el flujo de datos y de control, las limitaciones en cuanto a recursos, los mecanismos de cronogramas y comunicaciones como también los componentes *software*

- El proceso de codificación del *software*, el cual produce el código fuente y el objeto. A través de este proceso, los *LLR* son implementados como código fuente

- El proceso de integración, el cual produce el código objeto ejecutable y lo agrega al sistema integrado o equipamiento

En todas estas etapas la trazabilidad es requerida: entre los requerimientos del sistema y los *HLR*, entre los *HLR* y los *LLR*, entre los *LLR* y el código fuente y también entre las pruebas y los requerimientos.

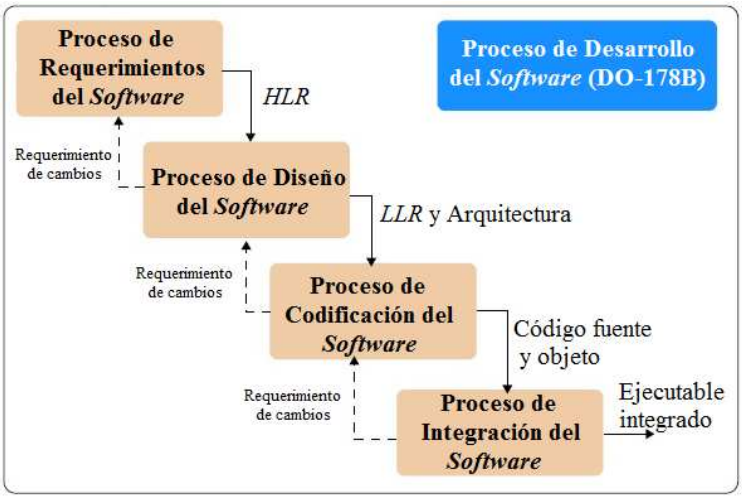


Gráfico 10: Proceso de Desarrollo del *Software*

Los procesos de prueba del *software* para la aviónica de las aeronaves tienen dos objetivos complementarios. El primero es demostrar que el *software* satisface sus requerimientos. El segundo objetivo es demostrar, con un alto nivel de confiabilidad, que todos los errores que podrían llevar a una condición de avería inaceptable han sido removidos.

Hay tres tipos de actividades de pruebas:

- Pruebas de bajo nivel: para verificar la implementación de los *LLR*
- Pruebas de integración del *software*: para verificar la interrelación entre los requerimientos del *software* y los componentes y para verificar su implementación dentro de la arquitectura del *software*
- Pruebas de integración con el *hardware*: para verificar la correcta operación del *software* en el ambiente de la computadora de destino

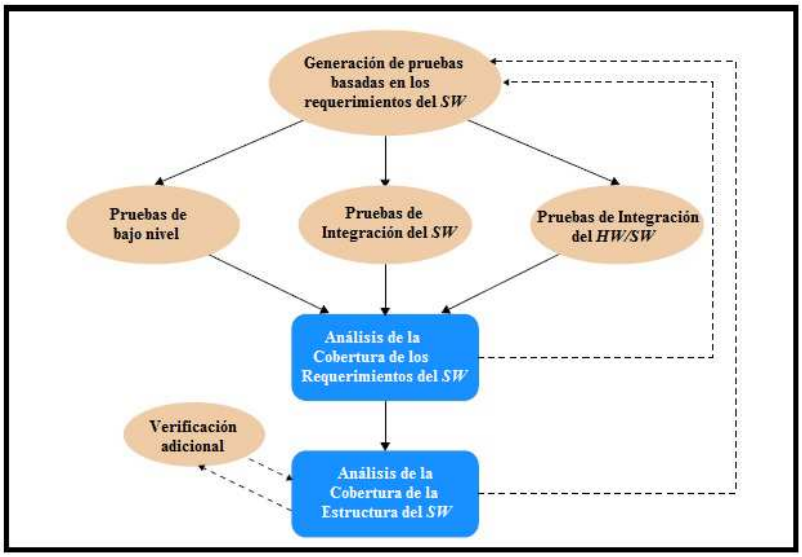


Gráfico 11: Procesos de Prueba

La versión *DO-178C* del estándar, publicado originalmente en el 2012, surge como una revisión y actualización de su predecesor *DO-178B* (ver Gráfico 12). Las razones para estos cambios son las siguientes, detallan Ben Brosgol y Cyrille Comar (2010): [5]

- Considerar las nuevas tecnologías respecto a *software*
    - Programación orientada a objetos
    - Diseño basado en modelos y generadores automáticos de código
    - *COTS* y otras herramientas, incluyendo sistemas operativos en tiempo real
- Reconocer que la verificación del *software* incluye más actividades, además de las pruebas
    - Métodos formales
    - Interpretación abstracta

Este estándar es utilizado como documento obligatorio para que las autoridades de certificación, tales como *FAA*, *EASA* y el Departamento de Transporte de Canadá (*Transport Canada*), aprueben todos los sistemas aeroespaciales comerciales basados en *software*.

Una de las diferencias respecto a *DO-178B* se refiere a la trazabilidad. En esta nueva versión, se requiere que la trazabilidad sea bidireccional.

Gráfico 12: Suplementos incluidos en *DO-178C* [6]

El suplemento de Métodos Formales permite el desarrollo de aviónica para las aeronaves utilizando pruebas formales como un medio adicional de verificación. Si bien la verificación matemática es considerada adecuada en teoría, este estándar continua recomendando la realización de pruebas sobre el destino, a fin de asegurar que el código funciona correctamente en el *hardware* de destino y evitar falsos positivos a partir de los métodos formales.

El desarrollo Basado en Modelos proporciona soporte para el desarrollo a niveles más altos de abstracción que el código fuente, ayudando a manejar así el enorme crecimiento del *software*. Las investigaciones indican que la creación de prototipos de los requerimientos de *software* en etapas tempranas, utilizando un modelo ejecutable, efectivamente elimina los defectos en los niveles del requerimiento y del diseño. Además, es posible generar código fuente en forma automática a partir del modelo ejecutable.

El suplemento de Orientación a Objetos y sus tecnologías relacionadas, se focaliza en lenguajes tales como *C++*, *Java* y *Ada 2005*, considerando en particular el asunto de la verificación de subtipos. Este estándar recomienda el uso de Pruebas de Verificación del Diseño (*DVT*), efectuado a nivel de las clases, el cual demuestra que todas las funciones miembro cumplen con el contrato de la clase con respecto a la precondiciones, pos-condiciones y las invariantes del estado de la clase.

4.2 Identificación y evaluación de las exigencias específicas y su impacto en un proyecto de desarrollo de *software*

Las principales agencias que certifican y aprueban la introducción de nuevos sistemas de aviónica o modificaciones a los sistemas existentes, como la *FAA*, exigen que los desarrollos y productos entregables se adapten a las normas requeridas. Esto generalmente introduce una preocupación por los incrementos en los costos que se generan en un proyecto de desarrollo de *software*, especialmente en lo relacionado con las actividades de V&V, ya que se exigen tasas de defectos cercanas a cero, respaldadas por una serie de documentos y demostraciones requeridas por el proceso de certificación.

La flexibilidad que otorga la serie de estándares *DO-178* al no ser prescriptivas, introduce también una dificultad en su implementación inicial, debido a que diversos aspectos son esencialmente abstractos y no existe un grupo de actividades base a partir de las cuales pueda avanzarse.

Generalmente, se incorpora también en un proyecto de desarrollo de *software* un rol muy importante que es el llamado Representante de Ingeniería Designado (*DER*), el cual se encarga de aprobar o recomendar la aprobación de los datos técnicos que surgen del proyecto de desarrollo de *software*.

Asimismo, todas las herramientas utilizadas para el desarrollo del *software*, deben a su vez ser parte del proceso de certificación, es decir no sólo se controla que el *software* desarrollado cumpla con los requisitos exigidos, sino que también las herramientas usadas durante el desarrollo cumplan con las normas establecidas.

Un factor de complejidad adicional es la exigencia de niveles completos respecto a la trazabilidad de los requerimientos.

Una organización dedicada al desarrollo de *software* que quiera crear un marco de trabajo adecuado para el desarrollo de sistemas de aviónica que luego deban ser certificados por normas *DO-178*, según la compañía *Wind River Systems, Inc.* (2015), debería efectuar las siguientes evaluaciones respecto a sus procesos actuales en cuanto a la planificación, desarrollo y verificación: [7]

- Planificación:
    - Revisión y análisis de los estándares y herramientas de desarrollo y pruebas

- Revisión y análisis del plan de aseguramiento de la calidad del *software*
- Revisión y análisis del plan de gestión de la configuración del *software*

- Desarrollo:
  - Revisión y análisis de la documentación de los requerimientos del sistema y su trazabilidad
  - Revisión y análisis de la definición de las arquitecturas del *software*
  - Revisión y análisis de la implementación de los requerimientos en el código fuente
  - Revisión y análisis de los procesos para informar sobre errores y cómo son corregidos
  - Revisión y análisis del manejo y gestión de la documentación del código fuente
  - Revisión y análisis de las métricas obtenidas respecto a funcionalidades implementadas en el código fuente, cantidad de líneas de código y construcciones del lenguaje utilizado
  - Identificación de librerías especiales utilizadas por las aplicaciones (por ejemplo, librerías estándares del lenguaje *C* o librerías matemáticas) y su documentación

- Verificación:
  - Revisión y análisis de las herramientas y procedimientos usados durante la V&V
  - Revisión y análisis de los procedimientos de V&V cuando se efectúan adaptaciones al sistema
  - Revisión y análisis de la generación y documentación de casos de prueba
  - Determinar la viabilidad del compilador y enlazador para soportar los requerimientos de *DO-178B/DO-178C*
  - Revisión y análisis del ambiente de pruebas utilizado

Las características particulares demandadas por *DO-178B/DO-178C* naturalmente incrementan los costos en comparación a un desarrollo de *software* tradicional. La problemática principal para las organizaciones que quieran desarrollar *software* para aviónica es entonces poder limitar estos incrementos de costos, al mismo tiempo que se incrementa la productividad. Los mayores costos surgen debido a diversas causas, entre las principales de ellas se encuentran la falta de entendimiento del proceso completo establecido por *DO-178B/DO-178C*, la falta de un involucramiento temprano de la autoridad de certificación o el *DER* y una baja productividad debido a herramientas y procesos ineficientes para un proyecto de desarrollo de *software* de estas características, postula Mehmet Kerim Çakmak (2013). [8]

Un proyecto de *software* dirigido a ser certificado por *DO-178B/DO-178C* frecuentemente incrementa, de acuerdo con Vance Hilderman y Tony Baghi (2007), en un 20 a 40 % el costo del proyecto. [9]

Utilizando los métodos tradicionales, la productividad de un proyecto de *software* basado en la serie *DO-178* es cuatro veces menos que la de un proyecto de desarrollo de *software* para aplicaciones no críticas. Por lo tanto, menciona Bernard Dion (2007), la necesidad de aplicar nuevas herramientas específicas para las etapas de diseño, codificación y V&V es notoria. [10]

Dado que *DO-178B/DO-178C* no recomiendan un lenguaje de programación específico, el uso de lenguajes de programación muy difundidos como *C* o *C++* plantea inconvenientes adicionales ya que muchas de sus características, si no son controladas y verificadas apropiadamente, pueden resultar en un código *software* que sea no determinístico, no utilizado o difícil de verificar y cuya configuración puede cambiar dependiendo del estado del sistema en tiempo de ejecución, siguiendo a David Beberman y Joe Wlad (2010). [11]

El alcance de la cobertura de las pruebas requeridas es también un factor importante en cuanto a complejidad e incremento de costos ya que se requiere una cobertura de código estructural para el *software* categorizado como nivel C, una cobertura de decisión verificando cada bifurcación para el *software* categorizado como nivel B, y una Condición Modificada/Cobertura de Decisión (*MC/DC*) para el *software* categorizado como nivel A. Podemos ejemplificarlo, continuando con Mehmet Kerim Çakmak (2013), mediante el siguiente código básico: [12]

If (a || b && c)

x++;

Una cobertura de código estructural requerirá una prueba única que ejecute el caso por verdadero, por ejemplo a = b = c = 1. Una cobertura de decisión requerirá dos casos de prueba, uno por verdadero y otro por falso, por ejemplo a = b = c = 1 y a = b = c = 0. Una cobertura *MC/DC* requerirá generalmente N + 1 casos de prueba, donde N es el número de condiciones en la lógica, en este caso serán cuatro casos de prueba, por ejemplo a = b = c = 0, a = 1 con b = c = 0, a = c = 0 con b = 1 y b = c = 1 con a = 0.

Una pregunta que surge naturalmente respecto a un proyecto de desarrollo de *software* de estas características, teniendo en cuenta sus peculiaridades, es ¿cuál es la cantidad mínima de personas que se necesitarán para el proyecto y cuáles son sus roles? Podemos responder a ello, de acuerdo con Vance Hilderman y Tony Baghi (2007), con el siguiente detalle de roles necesarios: [13]

- Un experto en seguridad
- Un encargado de redactar los requerimientos del sistema
- Un encargado del sistema de gestión de la configuración
- Un encargado de Aseguramiento de la Calidad
- El *DER*
- Un arquitecto para el *software*
- Al menos un desarrollador y un *tester*

Las tareas relacionadas con el proceso de pruebas implican costos superiores respecto a un proyecto convencional. Las pruebas de *software* para aviónica tienen dos objetivos complementarios. El primero, naturalmente, es demostrar que el *software* satisface los requerimientos. El segundo objetivo es demostrar, con un alto grado de confianza, que todos los errores que podrían llevar a una condición de falla inaceptable, tal como está determinada por el proceso de evaluación de seguridad del sistema, han sido removidos.

Entre los principales desafíos que una organización enfrenta cuando se desarrolla *software* crítico para la seguridad para Aviación basado en *DO-178B/DO-178C*, podemos mencionar, según lo detalla la compañía *Esterel Technologies SA* (2012), los siguientes: [14]

- Evitar múltiples descripciones del *software*: dado que el desarrollo del *software* está dividido en varias fases con sus respectivas salidas (*HLR*, *LLR* y el código fuente), es importante evitar reescribir la descripción del *software* ya que es costoso y propenso a errores, con un mayor riesgo de inconsistencias entre las diferentes descripciones. Esto implica un esfuerzo significativo para verificar el cumplimiento de cada nivel respecto a su nivel previo. El propósito de muchas de las actividades es detectar errores introducidos durante las transformaciones de una versión a otra

- Prevenir la ambigüedad y la falta de precisión en las especificaciones: las especificaciones de los requerimientos y del diseño están redactadas generalmente en algún lenguaje natural, posiblemente complementado por alguna notación informal (como por ejemplo *UML*). La ambigüedad inherente de un lenguaje natural puede llevar a diferentes interpretaciones, dependiendo del lector. Esto cierto especialmente con cualquier texto que describa un comportamiento dinámico. Por ejemplo, ¿cómo interpretaría uno la combinación de fragmentos de varias secciones de un documento, tales como "A genera a B", "si ambos B y C ocurren, entonces establezca D", "si D o Z están activos, entonces reestablecer A"?

- Evitar los errores en los requerimientos de bajo nivel y la codificación: la codificación toma como entrada la última formulación del lenguaje natural o pseudocódigo. Dado que los programadores tienen generalmente un conocimiento limitado del sistema, las ambigüedades en las especificaciones afectan a la codificación

- Permitir una implementación eficiente del código en el *hardware* destino: el código debe ser simple, determinístico y eficiente. Debe requerir los menores recursos que sean posibles, en términos de memoria y tiempo de ejecución, para la *CPU*. Debe ser posible reubicarlo fácil y eficientemente en otro procesador

- Encontrar errores en la especificación y diseño tan pronto como sea posible: muchos de estos errores son detectados durante las pruebas de integración del *software*. Uno de los motivos es que la especificación de los requerimientos y el diseño es frecuentemente ambigua y sujeta a interpretación. Otra causa es que es difícil para

un lector el entender los detalles respecto a un comportamiento dinámico sin ser posible de verlo en ejecución. En un proceso tradicional, frecuentemente, la primera vez que se ejecuta el *software* es durante la integración, lo cual es muy tarde para el proceso. Si un error en las especificaciones puede ser detectado sólo durante la fase de integración del *software*, el costo de corregirlo es mucho mayor que si hubiera sido detectado durante la fase de especificación

- Disminuir la complejidad de las actualizaciones: existen muchas fuentes de actualizaciones para el *software*, yendo desde el arreglo de errores a mejoras de sus funciones o la introducción de nuevas funciones. Cuando hay que efectuar una actualización, todos los productos del ciclo de vida del *software* tienen que ser actualizados en forma consistente y todas las actividades de verificación deben ser ejecutadas

- Mejorar la eficiencia de la verificación: el nivel de verificación para el *software* crítico para la seguridad en Aviación es mucho mayor que para otros *softwares* no críticos. Para un *software* categorizado como nivel A, el costo total de la verificación y pruebas puede alcanzar hasta un 80 por ciento del presupuesto del proyecto. La verificación suele ser también un cuello de botella para la finalización de un proyecto, por lo tanto cualquier cambio en la dinámica o el costo de las verificaciones tiene un impacto directo en el costo y duración del proyecto

- Proporcionar una manera eficiente de resguardar la propiedad intelectual: actualmente, una parte significativa del conocimiento desarrollado por las compañías fabricantes de aeronaves o equipamiento para las mismas reside en el *software*. Es de la mayor importancia entonces proporcionar métodos y herramientas para poder resguardar la propiedad intelectual

Hemos visto hasta aquí que los desafíos y exigencias que plantea el desarrollo de *software* crítico para la seguridad en Aviación no podrían ser resueltos exitosamente con los métodos y herramientas tradicionales. Revisaremos ahora algunas de las soluciones que se encuentran disponibles y han sido utilizadas exitosamente para los problemas planteados.

4.3 Soluciones para los problemas identificados y mitigación de sus consecuencias

Hemos visto en la sección anterior los principales desafíos que presenta el desarrollo de *software* en sistemas críticos para la seguridad, los cuales son principalmente en mi opinión, fundamentada en la investigación realizada, los siguientes:

- Lograr que tanto los procesos como los productos resultantes del proyecto desarrollo de *software* reciban la certificación de la entidad correspondiente respecto al cumplimiento de los requisitos exigidos para un sistema crítico para la seguridad. Esto es fundamental para lograr la autorización para incorporar el *software* dentro de los sistemas de una aeronave y poder así comercializar la misma en el mercado y acceder a nuevos segmentos

- Lograr que los costos y tiempos del proyecto de desarrollo de *software* se mantengan dentro de márgenes que garanticen la consecución satisfactoria del mismo, empleando técnicas y herramientas disponibles en el mercado, desarrolladas específicamente para este tipo de sistemas

Por ejemplo, el costo de efectuar las pruebas con una cobertura de la totalidad del código fuente, utilizando técnicas tradicionales, haría que fuera prohibitivo el desarrollo de este tipo de *software*. Por otro lado, la complejidad de desarrollar una herramienta específica desde cero, también sería poco práctico ya que, recordemos, debemos asegurarnos que tanto el producto final *como las herramientas utilizadas para obtenerlo* cumplan con los requisitos establecidos por los estándares internacionales respecto a la seguridad.

El enfoque más adecuado en estos casos es utilizar alguna herramienta ya desarrollada y con un amplio historial en el mercado, la cual a su vez cuente con una certificación respecto al estándar seguido en el proyecto de desarrollo de *software*, permitiéndonos así poder emplear métodos ya probados, realizar *tests* con una cobertura del código apropiada y mantener los costos y tiempos de desarrollo dentro de los valores planificados.

A continuación, analizaremos y evaluaremos algunas de las más utilizadas habitualmente en proyectos de desarrollo de *software* de estas características.

CodeSonar* y *CodeSurfer* de *GrammaTech

La empresa *GrammaTech* desarrolló dos herramientas de análisis estático, utilizadas para el proceso de verificación del *software*, llamadas *CodeSonar* y *CodeSurfer* para dar apoyo a las actividades de desarrollo de *software* basadas en *DO-178B/DO-178C*. *CodeSonar* ejecuta un análisis interprocedural completo del código fuente desarrollado en *C* y *C++*, identificando errores de programación que pueden resultar en caídas del sistema, corrupción de la memoria y otros problemas. Incluye numerosas características de automatización del flujo de trabajo, incluyendo *API* para integraciones adaptadas al cliente. *CodeSurfer* es una herramienta para el entendimiento de los programas a través de un análisis del código fuente, calculando una variedad de representaciones que pueden ser exploradas a través de *GUI* o accedidas a través de *API*. Según la compañía *GrammaTech Inc.* (2010), *CodeSonar* busca los errores en forma automática, mientras que *CodeSurfer* hace que las revisiones manuales del código sean más fáciles y rápidas. [15]

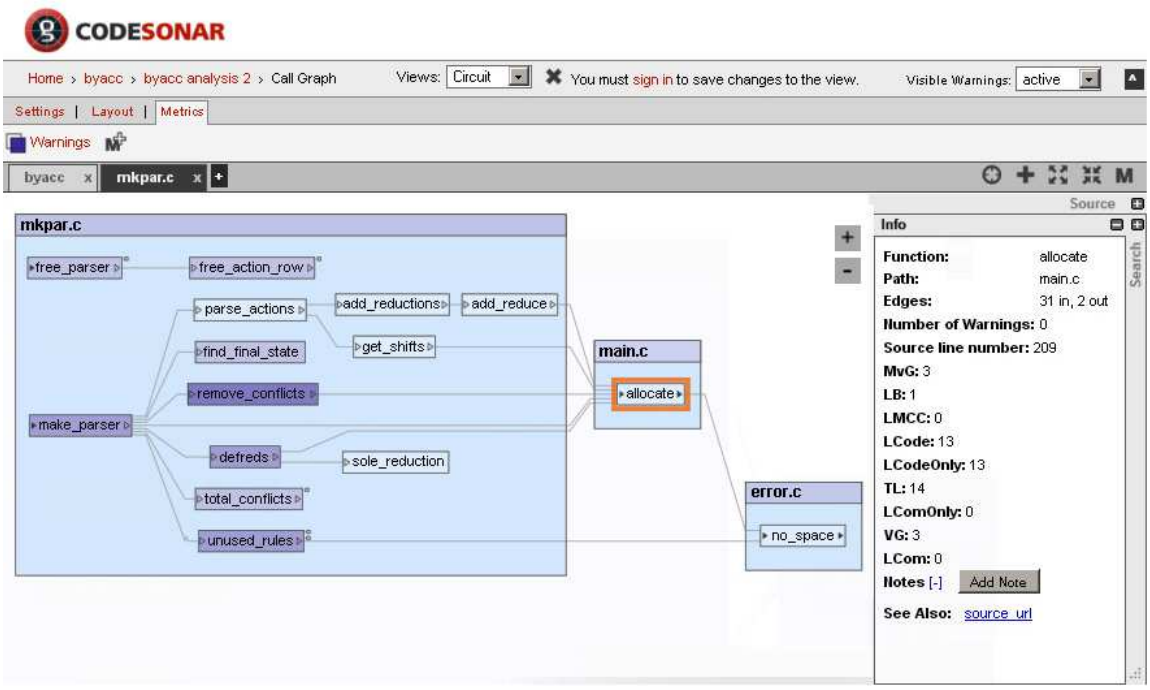


Figura 22: *CodeSonar* de *GrammaTech*

Respecto a la arquitectura del *software*, ambas herramientas proporcionan soporte adicional para la revisión de la misma. *CodeSurfer* proporciona visualización y una funcionalidad de

query sofisticada, mientras que *CodeSonar* posee opciones de verificación y extensión que permiten que diversos problemas de la arquitectura puedan ser señalados automáticamente.

Respecto a la compatibilidad con los requerimientos de alto nivel, *DO-178B* establece que ”*…el objetivo es asegurar que la arquitectura del software no entra en conflicto con los requerimientos de alto nivel, especialmente con las funciones que aseguran la integridad del sistema, por ejemplo, esquemas de particionamiento…*”. *CodeSurfer* puede usarse para inspeccionar las propiedades estructurales del *software* en muchos niveles de detalle. El *GUI* permite observar en forma interactiva los elementos estructurales, mientras que la *API* proporciona las capacidades para automatización. Ambas poseen funciones *queries*, tales como rebanamiento estático (*slicing*), que pueden usarse para verificar los esquema de particionamiento.

En referencia a la consistencia, *DO-178B* establece que “*…el objetivo es asegurar que una relación correcta existe entre los componentes de la arquitectura del software. Esta relación existe a través de los flujos de datos y control…*”. *CodeSurfer* permite examinar ambos flujos desde varias perspectivas, entre ellas los gráficos de flujo de control y gráficos de llamadas. Los resultados de estos gráficos pueden ser sobreimpuestos para facilitar el reconocimiento de las dependencias involucradas.

Respecto a la compatibilidad con la computadora destino, *DO-178B* establece que “*…el objetivo es asegurar que no existan conflictos, en especial relacionados con la inicialización, operaciones asíncronas, sincronización e interrupciones, entre la arquitectura del software y las características de la computadora de destino respecto a software y hardware…*”. *CodeSonar* puede detectar problemas potenciales de compatibilidad a través de sus verificaciones *Uninitialized Variable*, *Double Lock*, *Double Unlock*, *Try-lock that will never succeed*, y *Shift Amount Exceeds Bit Width*.

En cuanto a la verificabilidad, *DO-178B* establece que “*…el objetivo es asegurar que la arquitectura del software pueda ser verificada, por ejemplo, que no hayan algoritmos recursivos sin límites…*”. Si bien la demostración en forma completa de la ausencia de recursión sin límites es probablemente imposible de lograr, *CodeSonar* ofrece varias verificaciones que ayudan a reducir la probabilidad que ocurra una recursión infinita.

Respecto a cumplir con los estándares, *DO-178B* establece que “*…el objetivo es asegurar que se han seguido los estándares para el diseño de software y que cualquier desvío de los estándares ha sido debidamente justificado…*”. *CodeSurfer* calcula diversas métricas de

complejidad estándares, cuyos valores pueden ser comparados contra los límites establecidos por los estándares de diseño del *software*.

Con respecto a la integridad de particionamientos, *DO-178B* establece que "*…el objetivo es prevenir o aislar las violaciones de los particionamientos…*". *CodeSurfer* permite seguir los flujos tanto de control como de datos a través de las diversas ejecuciones del programa, pudiendo obtenerse respuestas a preguntas tales como "¿pueden los datos en la región A influir en la ejecución de la región B?", "¿hay alguna camino de ejecución entre las regiones C y D?" o "¿quiénes llaman a la función F?".

En cuanto a la precisión y consistencia, *DO-178B* establece que "*…el objetivo es determinar la corrección y consistencia del código fuente, incluyendo uso de la pila, desbordamientos aritméticos, contención de recursos, peor caso del tiempo de ejecución, manejo de excepciones, uso de variables sin inicializar, variables o constantes no utilizadas y corrupción de datos debido a conflictos de tareas o interrupciones…*". *CodeSonar* efectúa una gran variedad de verificaciones de corrección y consistencia, incluyendo aquellas para las siguientes clases de alertas: *Excessive Stack Depth, Buffer Overrun, Buffer Underrun, Type Overrun, Type Underrun, Cast Alters Value, Integer Overflow of Allocation Size, Uninitialized Variable, Unused Value, Deadlock* y *File System Race Condition.*

Respecto al análisis de la cobertura estructural, *DO-178B* establece que el mismo "*…puede revelar estructuras de código que no fueron ejecutadas durante las pruebas. La resolución requerirá actividades de verificación de software adicionales. El código sin ejecutar puede ser el resultado de código muerto o código desactivado…*". *CodeSonar* detecta código inalcanzable como parte de su conjunto de funciones de análisis estándar, con distintas clases de alertas tales como: *Unreachable Call, Unreachable Computation, Unreachable Conditional, Unreachable Control Flow*, y *Unreachable Data Flow.* CodeSurfer proporciona información complementaria acerca de la accesibilidad del código en varios niveles de detalle, pudiendo determinar si hay funciones que nunca son llamadas (*call graph*).

La combinación de ambas herramientas, *CodeSonar* y *CodeSurfer*, hacen de esta opción una de las más adecuadas ya que permite realizar tanto un análisis estático del código automatizado como también una análisis estático eficiente en forma manual. La habilidad del producto para poder determinar los errores reales de aquellos que son falsos positivos durante la tarea de detección, un aspecto negativo al fijar características de búsqueda que

son necesariamente exhaustivas, es una funcionalidad que marca la diferencia en este tipo de herramientas, ya que facilita enormemente la tarea de los analistas y *testers*.

SCADE* de *Esterel Technologies

Esterel Scade es una línea de productos de *Esterel Technologies SA* (2012) que incluye a: [16]

- *SCADE Display* para el diseño de pantallas gráficas embebidas
- *SCADE Suite* para el diseño de pantallas lógicas
- *SCADE System* para el diseño de sistemas
- *SCADE LifeCycle* para la gestión del ciclo de vida de las aplicaciones

SCADE son las iniciales de "*Safety-Critical Application Development Environment*". Ha sido usado para el desarrollo del *software* para control de vuelo de *Airbus* y *Eurocopter*, entre otros proyectos.

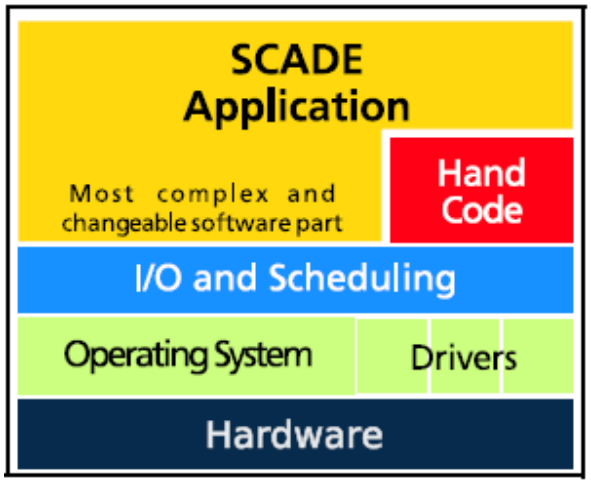


Gráfico 13: Aplicaciones de *SCADE*

Los ingenieros de desarrollo de controles y los ingenieros de *software* usan en general notaciones y conceptos diferentes. Estas diferencias hacen que la transición entre las especificaciones de los controles hacia las especificaciones de *software* sea compleja, costosa y sujeta a errores. Para solucionar este problema *SCADE Suite* proporciona

construcciones de *software* rigurosas que reflejan las construcciones de los ingenieros de controles.

Para modelar comportamientos, *SCADE Suite* recurre a la familiaridad y precisión de dos formalismos de especificaciones: las máquinas de estados, para especificar modos y transiciones en una aplicación, y los diagramas de flujo de datos para especificar los algoritmos de control. Un bloque de construcción básico en *SCADE* se conoce como operador, el cual puede representarse tanto gráficamente como textualmente.

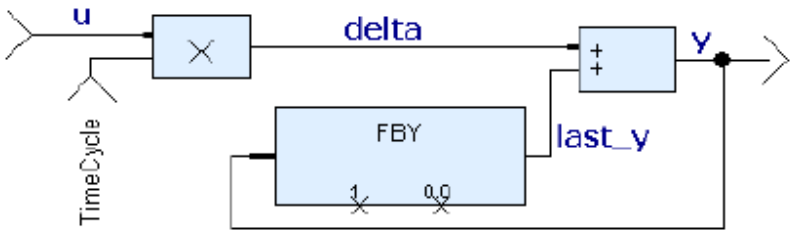


Gráfico 14: Representación gráfica de un operador

Component	Textual Notation for an Integrator Operator[a]	Graphical Notation
Formal interface	`node IntegrFwd(U: real ; hidden TimeCycle: real)` `  returns (Y: real) ;`	Arrows

Local variables declarations	`var`
	`delta : real ;`
	`last_y : real;`
		Named wires

Equations	`delta = u * TimeCycle ;`
	`y = delta + last_y ;`
	`last_y = fby(y , 1 , 0.0) ;`
		Network of operator calls

Gráfico 15: Representación textual de un operador

SCADE Suite posee diagramas de flujos de datos para un control continuo, tales como los sensores que actúan a intervalos de tiempos regulares. Se representa gráficamente utilizando diagramas de flujo de datos.

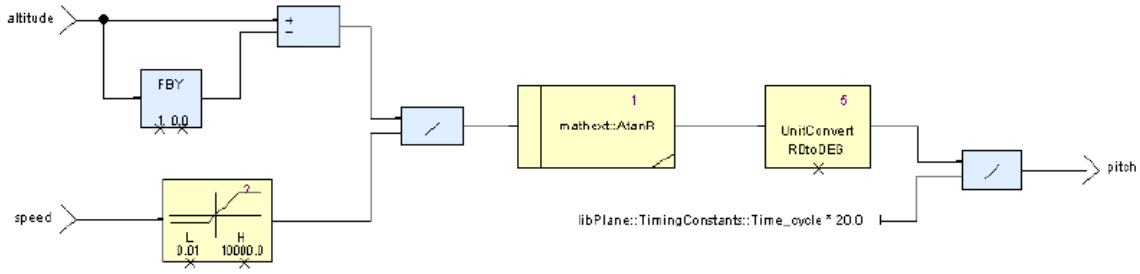
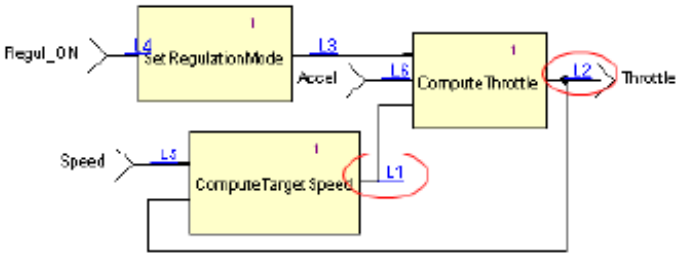


Gráfico 16: Ejemplo del modelado por *SCADE* de un flujo de datos de un sistema de control de vuelo

Los operadores en *SCADE* son completamente jerárquicos: los operadores a un nivel de descripción pueden a su vez estar compuestos de pequeños operadores interconectados por flujos locales, siendo. *SCADE* posee modularidad, el comportamiento de un operador no varía de un contexto a otro.

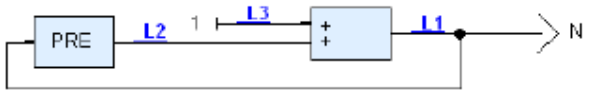
El lenguaje utilizado por *SCADE* es de tipos fuertes, cada flujo de datos posee un tipo y su consistencia es verificada. *SCADE* hace posible también el manejo de la sincronización y la causalidad. Si el dato X depende del dato Y, entonces Y tiene que estar disponible antes que el cálculo de X comience. Por ejemplo, un circuito de datos recursivo plantea un problema de causalidad, como se ilustra a continuación, en donde la salida del acelerador depende de sí mismo a través de los operadores *ComputeTargetSpeed* y *ComputeThrottle*. La verificación semántica de *SCADE Suite* detecta este tipo de errores y los señala como una salida con una definición recursiva.



Category	Code	Message
Semantic Error	ERR_400	**Causality error at** Package2::Operator1/ L1= the definition of flow _L1 depends on flow _L2 ; (Package2::Operator1/ L2=) the definition of flow L2 depends on flow L1 ;

Gráfico 17: Detección de un problema de causalidad en *SCADE Suite*

Otra característica importante de *SCADE Suite* es la relacionada con la inicialización de flujos. En ausencia de una inicialización explícita utilizando el operador -> (Init), la verificación semántica genera un error, como se muestra en el siguiente gráfico:



Category	Code	Message
Semantic Error	ERR_300	**Initialization error at** Package3::Operator4/ L2= The operator "pre" expects a well-initialized argument (Package3::Operator4/ L2=) The pre operator caused a delay

Gráfico 18: Detección de un problema de inicialización de flujos en *SCADE Suite*

Las máquinas de estados en *SCADE* son jerárquicas y los estados pueden ser estados simples o estados macros.

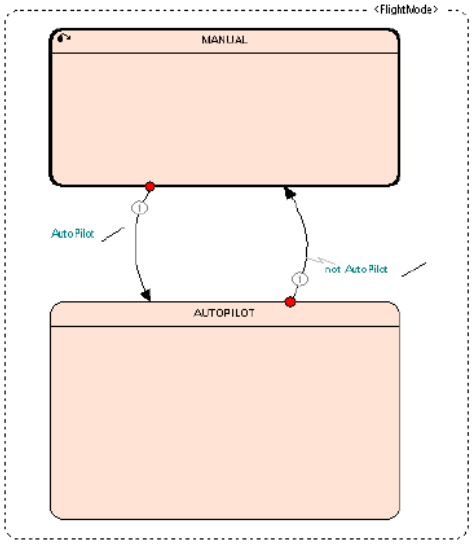


Gráfico 19: Máquinas de estados en *SCADE Suite*

SCADE Suite también permite combinar y anidar flujos de datos y control, tal como en el ejemplo siguiente de control de vuelo:

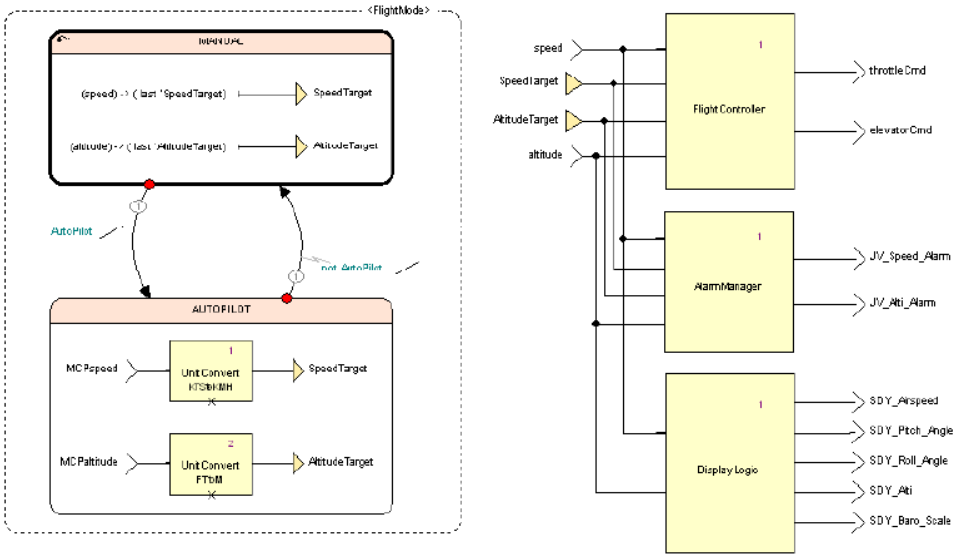


Gráfico 20: Combinación de flujos de datos y control del control de vuelo en *SCADE Suite*

SCADE Suite es, además, un entorno de desarrollo que soporta el paradigma de desarrollos basado en modelos:

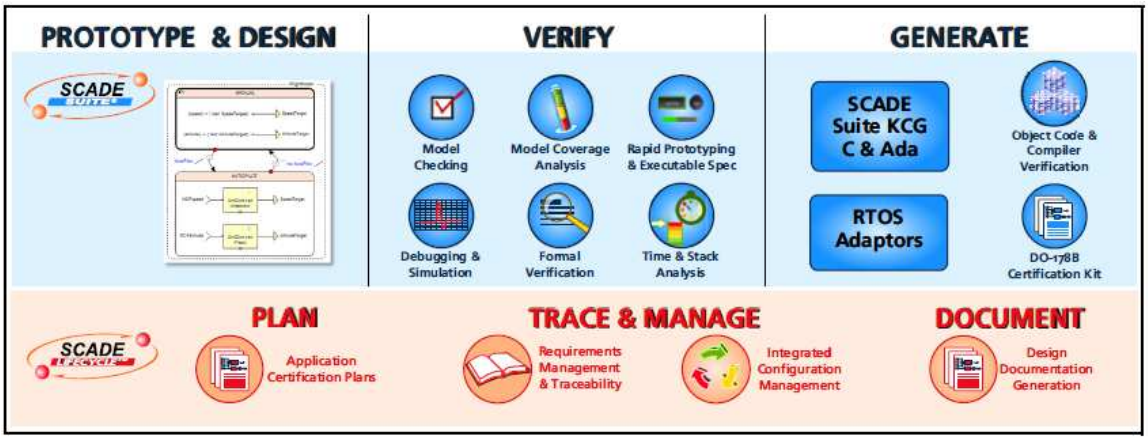


Gráfico 21: Desarrollo basado en modelos en *SCADE Suite*

SCADE Suite posee un generador de código automático llamado *KCG Code Generator*, implementando no sólo un esqueleto del código sino el comportamiento dinámico en forma completa. El código generado tienen las siguientes características:

- Es portable

- La estructura del código refleja la arquitectura del modelo para las partes de flujo de datos. Para las partes de flujos de control, se asegura la trazabilidad entre los nombre de estados y el código en *C*
- El código es legible y posee trazabilidad hacia el modelo de entrada
- La alocación de memoria es totalmente estática, no se permite alocación dinámica de la memoria
- No existen llamadas recursivas
- Sólo se permite lazos debidamente limitados
- Se limita el tiempo de ejecución
- No se efectúan cálculos de direcciones dinámicos
- No existen conversiones implícitas
- No se pasan las funciones como argumentos

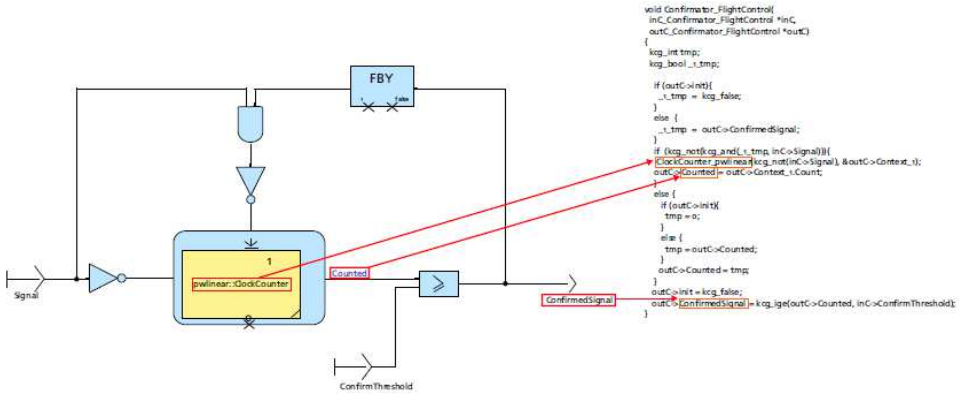


Gráfico 22: Flujo de datos y trazabilidad al código generado en *C* en *SCADE Suite*

SCADE Suite permite realizar una simulación dinámica del comportamiento del modelo a través de SCADE Suite Simulator:

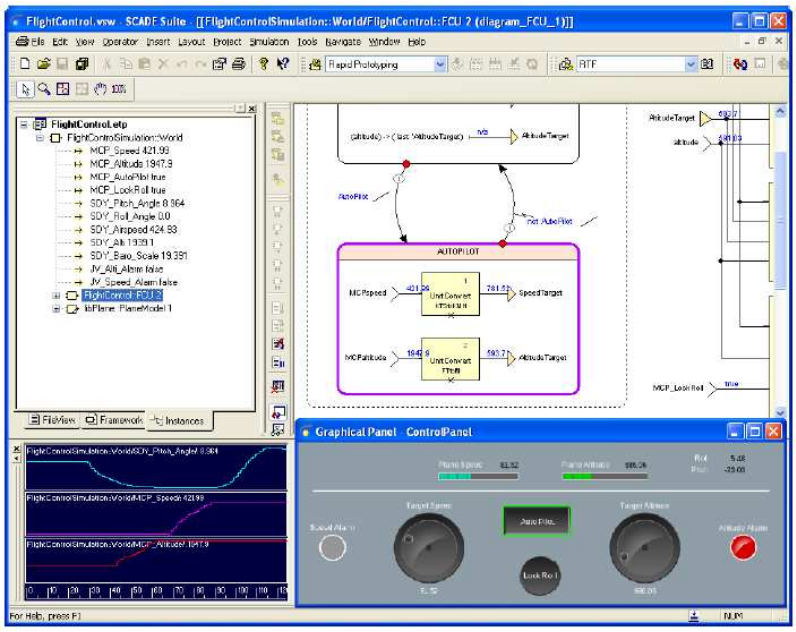


Figura 23: La simulación permite la visualización en tiempo de ejecución de las especificaciones de *software* en *SCADE Suite*

La funcionalidad que permite el desarrollo de código en forma automática, basada en el modelo desarrollado, es una de las grandes ventajas de *SCADE*, ya que permite grandes ahorros respecto a los costos del proceso de certificación. Además, su uso extendido en numerosos proyectos críticos para la seguridad y la disponibilidad de una *suite* que integra las distintas soluciones, hacen de *SCADE* una de las soluciones con más aceptación en el mercado.

PARASOFT test C/C++

Parasoft test C/C++ es una solución integrada para la automatización de la verificación y validación de la calidad de los procesos y del *software*, de acuerdo a lo especificado en *DO-178B/C*, incluyendo análisis estático, análisis estático de flujo de datos, análisis de métricas, revisión de código, pruebas unitarias y detección de errores en tiempo de ejecución, mencionan Adam Trujillo et al. (2012). [17]

En el análisis automatizado del código, se verifica que los estándares de codificación y las políticas de desarrollo definidas sean cumplidos, pudiendo personalizarse las distintas reglas.

El análisis de flujo simula los caminos de ejecución posibles y determina si esto caminos podrían conducir a distintas categorías de errores específicos del tiempo de ejecución. Esto es especialmente útil para los sistemas embebidos, en donde un análisis detallado en tiempo de ejecución es muchas veces imposible.

La revisión de código consta de módulos automáticos que preparan, notifican y realizan el seguimiento de las mismas. Cuando se combina con el análisis estático de código *C/C++*, virtualmente elimina la necesidad de una inspección línea por línea.

La función de detección de errores en tiempo de ejecución, monitorea continuamente a la aplicación detectando problemas tales como pérdidas de memoria, punteros nulos, memoria sin inicializar y desbordamiento de memorias intermedias.

En las pruebas unitarias y de integración con análisis de la cobertura, la herramienta prueba automáticamente en forma completa a la aplicación, validando el comportamiento de la aplicación y su funcionalidad y verificando la respuesta ante entradas inesperadas.

Pueden configurarse, además, todo un conjunto de reportes en diversos formatos (*HTML, PDF,* etc.) que informan sobre qué archivos fueron probados y analizados además de proporcionar los resultados de las pruebas y cobertura del código, automatizándose el envío de los mismos a los desarrolladores correspondientes junto con los gerentes y líderes de equipos.

La herramienta proporciona también un ruteo automatizado de tareas en la cual los defectos potenciales detectados son asignados automáticamente al desarrollador responsable por el código.

Parasoft test puede integrarse con la aplicación *Parasoft Concerto*, una plataforma para la gestión del desarrollo del *software* (ver Figura 24) que proporciona trazabilidad para todos los artefactos de un proyecto, requerimientos, defectos o mejoras y tareas, permitiendo establecer las políticas relacionadas con el cumplimiento y estandarización de los procesos.

Algunas de las características más destacadas respecto al cumplimiento de lo establecido por *DO-178B/DO-178C* son las siguientes:

- Actividades del procesos de codificación del *software*: se establecen reglas que refuerzan el cumplimiento de las mejores prácticas establecidas por la industria,

pudiendo seleccionarse librerías completas basadas en estándares y reglas individuales o crear reglas personalizadas

- Revisión y análisis de la arquitectura del *software*: pueden configurarse reglas para reforzar las políticas establecidas, incluyendo reglas que se basen en las especificaciones de la computadora de destino. Se incluyen las funcionalidades para análisis estático, pruebas unitarias, análisis de cobertura del código, revisión del código y detección de errores en tiempo de ejecución

- Revisión y análisis del código fuente: además de las características anteriormente mencionadas, los casos de prueba pueden ser generados automáticamente de acuerdo con las definiciones del usuario, para asegurar que el código cumple con los requerimientos de bajo nivel establecidos

- Ambiente de pruebas: puede usarse con una variedad de arquitecturas y sistemas operativos (ver Figura 25) embebidos mediante una compilación cruzada de la librería en tiempo de ejecución proporcionada para el ambiente en tiempo de ejecución requerido

- Métodos de pruebas basados en requerimientos: la funcionalidad *Data Source Wizard* ayuda en la parametrización de los casos de pruebas y *stubs*, permitiendo incrementar el alcance y cobertura de las pruebas con un esfuerzo mínimo. El análisis y generación de *stubs* se facilita mediante la opción *Stub View*, la cual presenta todas las funciones usadas en el código y permite la creación de *stubs* para cualquiera de dichas funciones que no estén disponibles para el alcance de las pruebas definido o incluso alterar las funciones existentes para propósitos específicos de las pruebas. El módulo avanzados de análisis estático entre procedimientos permite simular la factibilidad de los caminos de ejecución de la aplicación y determina si dichos caminos pueden llevar a errores específicos en tiempo de ejecución

- Análisis de cobertura estructural: la aplicación ofrece un analizador de cobertura de las pruebas con métricas múltiples, incluyendo cobertura de sentencias, saltos, caminos y *MC/DC*. Se dispone de trazabilidad desde los elementos establecidos en la cobertura hasta los casos de prueba correspondientes, permitiendo el análisis de los resultados de las pruebas y extender la cobertura en forma eficiente

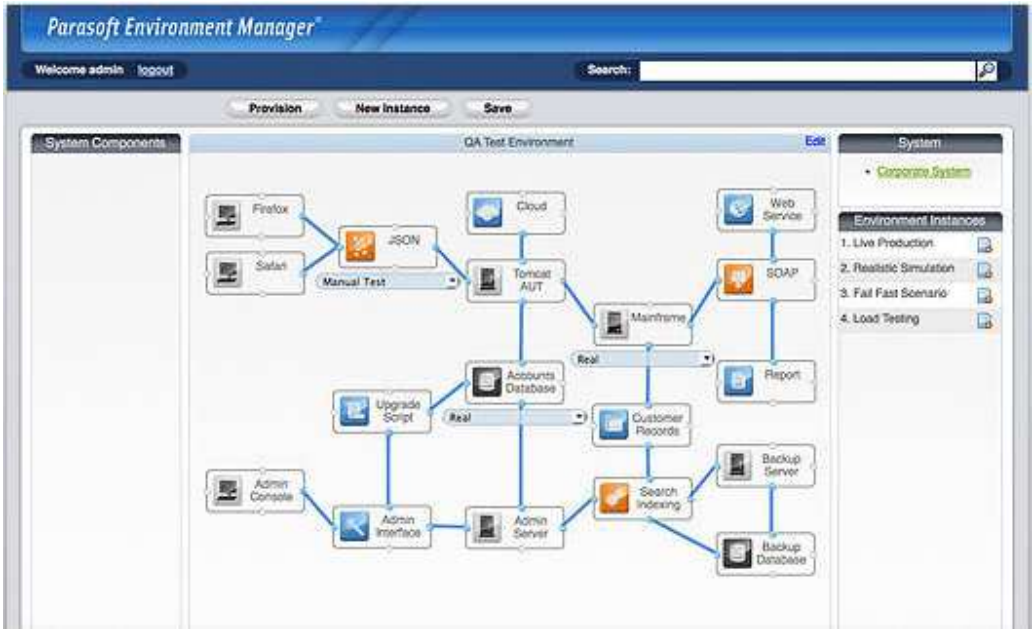


Figura 24: Gestión de Ambiente de *Parasoft*

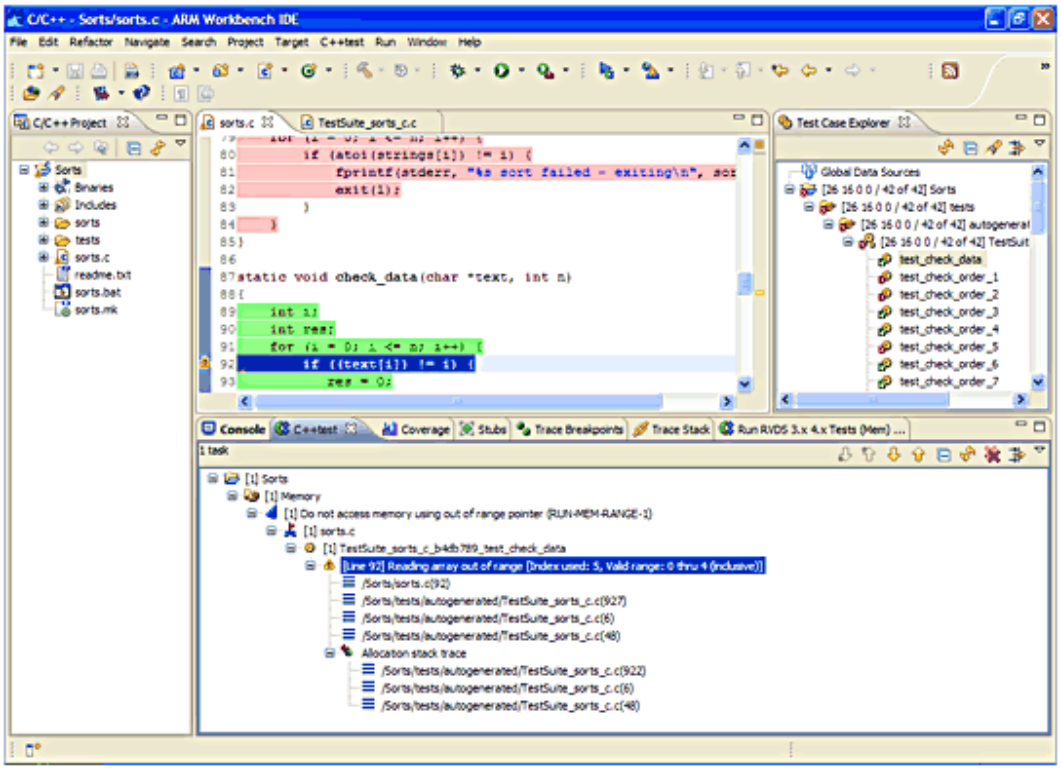


Figura 25: *Parasoft test C/C++*

Una de las principales ventajas de *Parasoft* es que está específicamente adaptado para cumplir con los requerimientos de *DO-178B/DO-178C* e incluye, además de *C* y *C++*, la posibilidad de utilizar otros lenguajes tales como *Java* y *.NET*.

GNAT Pro Safety-Critical

GNAT Pro es un entorno ideal para el desarrollo de aplicaciones embebidas críticas para la seguridad con una confiabilidad alta, siendo especialmente adaptada para el cumplimiento de distintos estándares internacionales, entre otros *DO-178B/DO-178C*.

Entre sus características más destacadas, según la compañía *AdaCore* (2014), podemos mencionar a: [18]

- Librerías configurables en tiempo de ejecución: permite especificar cualquier nivel de soporte para las características dinámicas del lenguaje *Ada*, en todas sus versiones. Las unidades incluidas en la librería pueden ser cualquier subconjunto de

las unidades estándares provistas con *GNAT Pro*, o pueden ser adaptadas específicamente. Esta capacidad es muy útil, por ejemplo, si uno de los perfiles predefinidos provee casi todas las características necesarias para adaptar un sistema existente a los nuevos requerimientos críticos para la seguridad, en donde los costos de adaptación sin las características adicionales son excesivamente altos

- Implementación en forma completa de la versión *Ada 2012*: incluye todas las revisiones y nuevas características incluidas en *Ada 2012*

- Análisis estático avanzado: entre otras características, a través del módulo *GNATStack*, permite calcular el espacio máximo de la pila requerido por cada tarea de una aplicación. Los límites calculados pueden ser usados para asegurar que el espacio suficiente es reservado, garantizando la seguridad y la ejecución predecible con respecto al uso de la pila

- Simplificación de los esfuerzos de certificación: pueden restringirse las características del lenguaje que, aunque no requieran de una librería en tiempo de ejecución, sin embargo podrían complicar la parte del análisis de cobertura de las pruebas como parte del esfuerzo para la certificación. Por ejemplo, puede prohibirse el uso de construcciones que podrían resultar en código con lazos y condicionales implícitos (*slice assignment*)

- Trazabilidad: a través de cambios en el compilador, puede generarse una versión de bajo nivel del programa fuente que revele las decisiones de implementación pero que permanezca, básicamente, independiente de la máquina. Esto ayuda a cumplir con los requerimientos de trazabilidad y puede usarse como punto de referencia para la verificar que el código objeto concuerde con el código fuente. Otras opciones de compilación generan detalles relacionados con otras representaciones de los datos (tamaños, definición de registros, etc.) lo cual también es de mucha ayuda para las actividades relacionadas con la trazabilidad

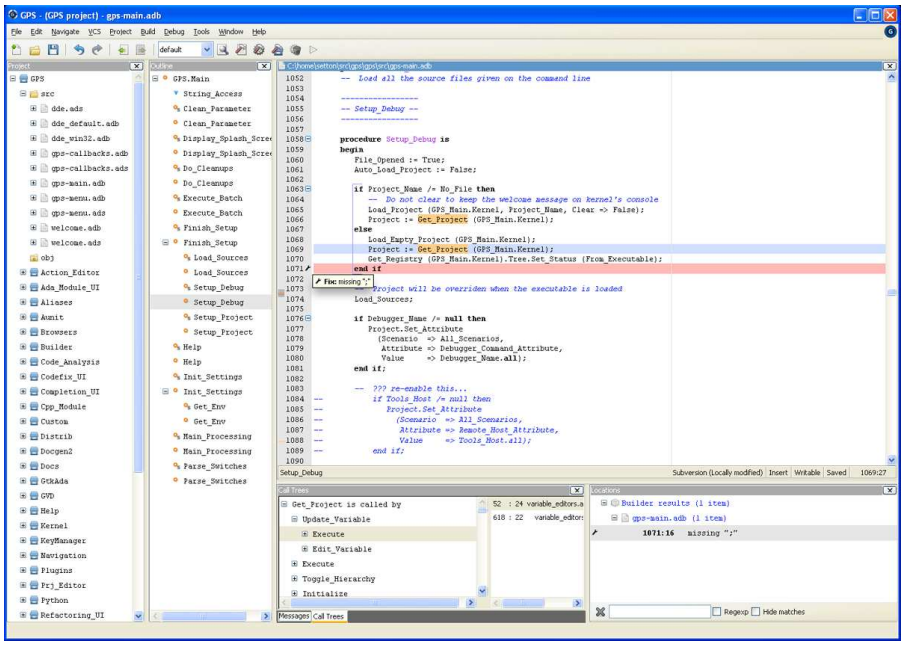


Figura 26: *GNAT Pro Safety-Critical*

Una de sus mayores ventajas es ofrecer una suite en donde se integran todas las funcionalidades ofrecidas por este entorno de desarrollo y su adaptación especial para el uso del lenguaje *Ada*, si bien esto último puede ser considerado también una limitación.

5. Ejemplos de aplicaciones *software* en Sistemas Críticos para la Seguridad en Aviación

En este capítulo describiremos aplicaciones reales de estos Sistemas en la Aviación. Entre las entidades y empresas involucradas en el desarrollo y uso de Sistemas Críticos para la Seguridad en Aviación, podemos mencionar a:

- *Rockwell Collins, Inc.*: compañía estadounidense proveedora de sistemas de información y tecnología en aviónica, fundada en 1933. Tiene su sede central en *Cedar Rapids, Iowa, USA*, cuenta con 20.000 empleados e ingresos anuales por 4.600 millones de dólares. La Fuerza Aérea Argentina ha seleccionado a *Rockwell Collins* para la modernización del sistema de aviónica de las aeronaves *C-130 Hércules*

- *Barco N.V.*: compañía multinacional de origen belga, fabricante especializado en pantallas, visores y *software*, fundada en 1934. Tiene su sede central en *Kortrijk, Bélgica*, cuenta con 3.250 empleados e ingresos anuales por 1.050 millones de euros

- *Boeing Company*: compañía norteamericana fabricante de aeronaves, vectores y satélites, fundada en 1916. Es la principal compañía exportadora de los Estados Unidos, tiene su sede central en *Seattle, Washington, USA*. Cuenta con 162.700 empleados e ingresos anuales por 91.000 millones de dólares

- *UTC Aerospace Systems*: compañía norteamericana proveedora de la industria aeroespacial y de defensa, creada en el 2012. Posee su sede central en *Charlotte, North Carolina, USA*. Cuenta con 42.000 empleados e ingresos anuales por 14.000 millones de dólares

- *AgiLynx, Inc.*: pequeña compañía estadounidense diseñadora y fabricante de componentes electrónicos para la aviación comercial y militar. Tiene su sede en *Billerica, Massachusetts, USA* y cuenta con 10 empleados e ingresos anuales por 500.000 dólares

- *PTC, Inc.*: compañía estadounidense de *software*, fundada en 1985. Tiene su sede central en *Needham, Massachusetts, USA* y cuenta con 6.200 empleados e ingresos anuales por 1.400 millones de dólares

- *Esterel Technologies*: compañía francesa de *software*, fundada en 1999. Con sede central en *Elancourt*, Francia, cuenta con 200 empleados e ingresos anuales por 15 millones de euros

- *Crane Aerospace & Electronics*: compañía estadounidense proveedora de componentes críticos para la industria aeroespacial y de defensa. Fundada en 1999, tiene sus oficinas centrales en *Burbank, California, USA*, cuenta con 2.800 empleados e ingresos anuales por 3.000 millones de dólares

- *Airbus Group SE*: compañía multinacional europea fabricante de aeronaves y componentes para la defensa. Fundada en el 2000, tiene su sede central en *Leiden*, Holanda, cuenta con 138.600 empleados e ingresos anuales por 60.700 millones de euros

- *DDC-I, Inc.*: compañía estadounidense de *software*, fundada en 1986. Tiene su sede central en *Phoenix, Arizona, USA*, cuenta con 35 empleados e ingresos anuales por 10 millones de dólares

- *Sikorsky Aircraft Corporation*: compañía estadounidense fabricante de aeronaves, fundada en 1925. Tiene su sede central en *Stratford, Connecticut, USA*, cuenta con 16.000 empleados e ingresos anuales por 10.000 millones de dólares

- *Embraer (Empresa Brasileira de Aeronáutica S.A.)*: compañía aeronáutica brasileña fundada en 1969. Es la tercera compañía fabricante de aviones del mundo, sólo por detrás de *Boeing* y *Airbus*. Su sede central se encuentra en *São José dos Campos, São Paulo, Brasil*, cuenta con 17.000 empleados e ingresos por 10.400 millones de dólares. Nuestro país cuenta con convenios de colaboración y asociación en la construcción de modelos de aeronaves (KC-390)

- *Thales Group*: compañía multinacional francesa fundada en el año 2000 que diseña y construye sistema eléctricos, proporcionando servicios diversas áreas como aeroespacial, defensa, transporte y seguridad. Con oficinas centrales en *Neuilly-sur-Seine*, Francia, cuenta con 63.700 empleados e ingresos anuales por 13.000 millones de euros. La empresa cuenta con oficinas en nuestro país, localizadas en la ciudad de Buenos Aires

- *Raytheon Company*: corporación industrial estadounidense para la industria de la defensa, fundada en 1922. Tiene su sede central en *Waltham, Massachusetts, USA*, cuenta con 63.000 empleados e ingresos anuales por 24.000 millones de dólares

- *BAE Systems plc*: compañía británica para las industrias de la defensa, seguridad y aeroespacial, fundada en 1999. Tiene su sede central en *Farnborough, Reino Unido*, cuenta con 84.600 empleados e ingresos anuales por 16.600 millones de libras esterlinas

- *Martin-Baker Aircraft Co. Ltd.*: fabricante británico de asientos eyectores y equipamiento de seguridad para la aviación, fundada en 1934. Tiene su sede central en *Denham, Buckinghamshire*, Reino Unido. La empresa cuenta con oficinas comerciales en nuestro país localizadas en la ciudad de Buenos Aires

- *Teledyne Technologies*: compañía estadounidense proveedora de instrumentos tecnológicos, imágenes digitales y *software*, electrónica para la defensa y actividades espaciales e ingeniería de sistemas. Fundada en 1953, tiene su sede central en *Thousand Oaks, California, USA*. Cuenta con 9.600 empleados e ingresos anuales por 2.400 millones de dólares

- *Texas Instruments Inc.*: compañía estadounidense de electrónica, fundada en 1951. Tiene su sede central en *Dallas, Texas, USA*, cuenta con 31.000 empleados e ingresos anuales por 13.000 millones de dólares

- *Sikorsky Aircraft Corporation*: compañía estadounidense fabricante de aeronaves, fundada en 1925. Tiene su sede central en *Stratford, Connecticut, USA*, cuenta con 16.000 empleados e ingresos anuales por 10.000 millones de dólares

- *Lockheed Martin*: compañía estadounidense nacida de la fusión de *Lockheed Corporation* y *Martin Marietta* en 1995, con presencia global en las áreas aeroespacial, defensa, seguridad y tecnologías de avanzada. Cuenta oficinas centrales en *Bethesda, Maryland, USA*, y emplea a unas 112.000 personas, teniendo ingresos del orden de los 45.000 millones de dólares anuales

- *AgustaWestland S.p.A.*: compañía angloitaliana fabricante de helicópteros, creada en el año 2000. Tiene su sede central en *Samarate, Italia*, cuenta con 13.000 empleados e ingresos anuales por 4.200 millones de euros

- *General Electric Corporation*: conglomerado multinacional de origen estadounidense con presencia en múltiples actividades económicas y tecnológicas. Fundado en 1892, tiene dos sedes principales en *Schenectady, New York, USA* y en *Fairfield, Connecticut, USA*, respectivamente. Cuenta con 305.000 empleados e ingresos anuales por 148.000 millones de dólares. Presente en nuestro país desde 1920, cuenta con oficinas en Buenos Aires y Rosario y plantas en Provincia de Buenos Aires, Chubut y San Luis, empleando a 2.000 personas

- *Selex ES*: compañía angloitaliana de electrónica y tecnologías de la información, creada en el 2013. Tiene sus sedes en *Basildon*, Reino Unido y en Roma, Italia. Cuenta con 17.000 empleados e ingresos anuales por 3.500 millones de euros

- *Indra Sistemas*: compañía española de consultoría en las áreas de transporte, defensa, energía, telecomunicaciones, servicios financieros y sector público, creada en 1993. Tiene su sede central en Alcobendas, España, cuenta con 42.000 empleados e ingresos anuales por 1.500 millones de euros. Con presencia en nuestro país desde 1993, emplea actualmente a más de 1.000 personas y tiene sus oficinas en las ciudades de Buenos Aires, Córdoba, Rosario y San Luis

- *Green Hills Software*: compañía estadounidense de *software*, creada en 1982. Con sede central en *Santa Bárbara, California, USA*, cuenta con 200 empleados e ingresos anuales por 70 millones de dólares

- *GrammaTech, Inc.*: compañía mediana estadounidense de *software*, creada en 1988. Tiene su sede central en *Ithaca, New York, USA*, cuenta con 60 empleados e ingresos anuales por 10 millones de dólares

5.1 Sistema de visualización de aviónica en *jets* privados

La compañía *Barco* ha desarrollado un sistema de visualización de aviónica para jets comerciales, utilizando el entorno de desarrollo de *AdaCore, GNAT Pro Ada*. El sistema operativo utilizado es *VxWorks 653 RTOS* de *Wind River*. El sistema, de acuerdo con la

compañía *AdaCore* (2010), cumple con los máximos requerimientos de seguridad especificados en el estándar *DO-178B*. [1]

La unidad de visualización ejecuta hasta siete tipos de aplicaciones críticas para la misión, incluyendo *pfd*, *mfd*, mapas, video, mantenimiento y cartas y está basado en la infraestructura modular abierta *MOSArt*.

Figura 27: Sistema *display* de aviónica de *Barco N.V.*

5.2 Sistema de instrumentos de vuelo para *jets* comerciales

Rockwell Collins ha adoptado la plataforma *GNAT Pro High-Integrity for DO-178B* para implementar su Sistema Electrónico de Instrumentación de Vuelo e Indicación de Motores y su Sistema de Alerta de la Tripulación. Para este proyecto se utilizó una librería en tiempo de ejecución llamada *Zero-Footprint*, que corresponde a un subconjunto del lenguaje *Ada* que elimina tanto las características no determinísticas como las características complejas del lenguaje, por ejemplo, el uso de la semántica dinámica de *Ada*, de esta forma, de acuerdo con la compañía *AdaCore* (2010), se reducen los costos y tiempos para la certificación del sistema. [2]

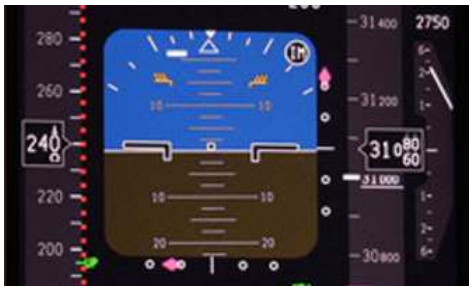


Figura 28: Sistema de *display* de aviónica avanzado de *Rockwell Collins*

5.3 Sistema de control de frenado en el tren de aterrizaje

Los sistemas de control de frenado de *Crane Aerospace & Electronics* son críticos para asegurar la seguridad de los pasajeros durante los aterrizajes rutinarios como también en los despegues abortados. Según detalla la firma *ANSYS Advantage* (2013), el sistema cumple con las especificaciones del estándar *DO-178B.* [3]

Para el desarrollo del *software* se ha utilizado *SCADE Suite*. El sistema es una combinación de *software*, actuadores controlados electrónicamente e interfaces de comunicación digitales de alta velocidad con los demás sistemas de a bordo. Por ejemplo, en caso de un desperfecto eléctrico u otro evento inesperado, el *software* no sólo asegura el correcto rendimiento del sistema de frenado sino que también alerta a la tripulación sobre el desperfecto y envía automáticamente un alerta para el personal de mantenimiento.

Figura 29: El sistema de control de frenado incluye a una cantidad de componentes mecánicos e hidráulicos, todos controlados por miles de líneas de código *software* crítico para la seguridad

A través del uso de *SCADE Suite* se han logrado beneficios significativos en cuanto al costo, velocidad y eficiencia tanto en las tareas del desarrollo de *software* como en las de verificación y validación. El entorno de desarrollo genera automáticamente el código y permite su prueba con respecto a miles de diferentes entradas.

Desde el comienzo de su uso, *SCADE Suite* ha identificado más de 180 defectos en los requerimientos de *software* correspondientes. Cuando los errores son detectados, *SCADE Suite* permite a los ingenieros de *software* la creación de escenarios *what-if* para poder ver rápidamente el impacto final que cualquier cambio en el diseño tiene sobre la confiabilidad y el rendimiento del sistema de frenado.

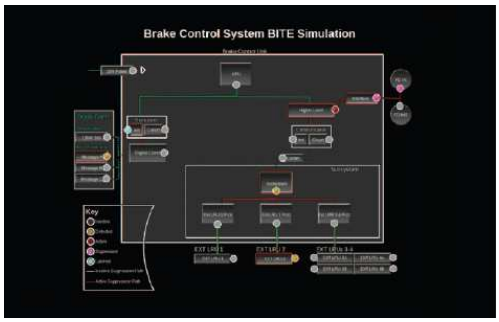
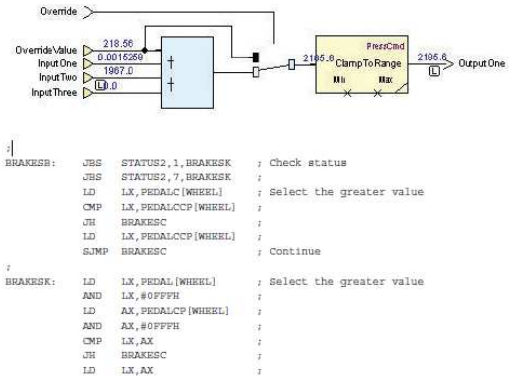


Figura 30: Simulación del subsistema de detección de fallas

SCADE Suite convierte las líneas de código en gráficos intuitivos, permitiendo mostrar la lógica subyacente de una forma más visual.




```
;
BRAKESB:    JBS    STATUS2,1,BRAKESK    ; Check status
            JBS    STATUS2,7,BRAKESK    ;
            LD     LX,PEDALC[WHEEL]     ; Select the greater value
            CMP    LX,PEDALCCP[WHEEL]   ;
            JH     BRAKESC              ;
            LD     LX,PEDALCCP[WHEEL]   ;
            SJMP   BRAKESC              ; Continue
;
BRAKESK:    LD     LX,PEDAL[WHEEL]      ; Select the greater value
            AND    LX,#0FFFH            ;
            LD     AX,PEDALCP[WHEEL]    ;
            AND    AX,#0FFFH            ;
            CMP    LX,AX                ;
            JH     BRAKESC              ;
            LD     LX,AX                ;
```

Gráfico 23: Simulación del subsistema de detección de fallas

5.4 Unidad de referencia inercial de datos para el *jet* comercial *Airbus A350 XWB*

Thales utilizó *GNAT Pro High-Integrity Edition for DO-178B* y la versión *Ada 2005* del lenguaje para construir la *ADIRU* usada en el *A350 XWB*. La unidad proporciona información precisa sobre la posición del vuelo. De acuerdo con la compañía *AdaCore* (2009), el sistema operativo utilizado es el *MACS2*. [4]

En el proyecto se utilizaron además técnicas de programación ágiles y las características seguras de la programación orientada a objetos. Otras herramientas usadas han sido *Qualified Code Standard Checker* y *Coverage*, las cuales permiten adoptar un enfoque

innovador a través de la provisión de información sobre Condición Modificada/Cobertura de Decisión (*MC/DC*), utilizando un simulador *PowerPC*.

Figura 31: *Airbus A350 XWB*

Figura 32: Pantallas de la aviónica del *Airbus A350 XWB*

5.5 *Software* para aeronaves militares en la Fuerza Aérea Argentina

La Dirección General de Investigación y Desarrollo de la FAA (DIGID) tiene como misión la investigación y el desarrollo tecnológico para el crecimiento del poder aeroespacial. Se encuentra conformada por una serie de centros e institutos (por ejemplo, el Centro de Investigación y Desarrollo de Tecnologías Aeronáuticas, CITeA) dedicados a diversos proyectos de cuya administración se encarga la DIGID, según el Grupo de Investigación de Defensa (s.f.). [5]

Los proyectos en los que se trabaja son de dos clases: los de desarrollo, que generalmente terminan en prototipos que deben ser homologados y certificados, y los de investigación.

La DIGID ha puesto en marcha un Sistema de Investigación y Desarrollo de la FAA (SIDFA) que comprende una serie de Verticales de Investigación y Desarrollo, entre ellas podemos mencionar a las siguientes:

- Área espacial: trabaja en proyectos de micro satélites, control y guiado de vectores, motores de plasma y eléctricos para el mantenimiento de orbitas satelitales

- Área de aviónica: desarrollo y mantenimiento de *software* de navegación y ataque en nivel correctivo, modificativo, adaptativo. En este último punto la FAA ha alcanzado un grado avanzado, ya que se cuenta con la capacidad de desarrollar *software* para los sistemas de armas permitiendo estar actualizados respecto a los sistemas de navegación y ataque (reemplazo de computadora de misión para Sistema de Armas A-4AR con arquitectura *ARINC 653*, Sistema de Adiestramiento de Combate IA-63 “Pampa”)

Figura 33: IA-63 “Pampa”

Figura 34: *McDonnell Douglas* A-4AR *Fightinghawk*

- Satélites de órbita baja: adquisición de un *software* para el diseño de órbitas y la complementación con las tareas de los *UAV*

- Simuladores: se utiliza el *software X-plane* para el desarrollo de distintos tipos de simuladores (pruebas de desorientación espacial para el INMAE, entrenador avanzado FAS 4040 para IA-58 “Pucará”)

Figura 35: Simulador para pruebas de desorientación espacial (INMAE) [6]

Figura 36: Entrenador avanzado FAS 4040 para IA-58 “Pucará”

- Juegos de guerra y toma de decisiones: conjunto de simuladores y aplicaciones para la Escuela Superior de Guerra Aérea, utilizando *Visual Studio/.Net Framework/C##* y *.NET*, *Windows Server System*, *SQL Server* y *Microsoft Office SharePoint Server*

5.6 Helicóptero militar *Eurocopter ARINC 653*

Eurocopter ha utilizado *GNAT Pro High-Integrity Edition for DO-178B* para el desarrollo de un demostrador tecnológico para helicópteros militares, el cual proporciona las interfaces militares y las funciones operacionales dentro de una arquitectura *ARINC-653*. Los objetivos del proyecto, según la compañía *AdaCore* (2011), son la demostración de las capacidades de la Aviónica Modular Integrada (*IMA*) y proporcionar una plataforma *ARINC-653* para la captura de los requerimientos técnicos y de procesos. [7]

Esta versión de *GNAT Pro* incorpora características tales como verificador estándar de codificación y analizador del tamaño de la pila estática, las cuales reducen el costo del desarrollo y certificación de los sistemas.

Figura 37: Helicóptero *Eurocopter Tiger*

5.7 Programa de modernización del avión de combate *Embraer AMX A-1* en la Fuerza Aérea Brasileña

Embraer utiliza *GNAT Pro* como su herramienta principal para el desarrollo del Programa de Vuelo Operacional del programa de modernización del avión de combate *AMX*, junto con el sistema operativo *VxWorks* de *Wind River*.

El objetivo de este programa es mantener la flota de 53 *AMX A-1* de la Fuerza Aérea Brasileña (FAB) en activo por otros 20 años, incorporando mejoras de vanguardia en la aviónica del sistema como también en su armamento y sensores. El programa debe cumplir con los niveles de seguridad definidos por el estándar *DO-178B.*

Las principales ventajas del lenguaje *Ada* usadas en este proyecto son su tipo de datos fuerte, sus mecanismos de modularidad (*packages*), verificación en tiempo de corrida, procesamiento en paralelo y manejo de excepciones, de acuerdo con la compañía *AdaCore* (2012). [8]

Figura 38: Avión de ataque *Embraer AMX A-1* de la FAB

5.8 Sistema de reabastecimiento aéreo *Airbus 330 MRTT*

Airbus ha certificado exitosamente el Sistema de Pértiga de Reabastecimiento en Vuelo (*ARBS*) para la versión militar de la aeronave *A330* Transporte Reabastecedor Multi Rol (*MRTT*). El proceso de certificación fue simplificado ya que para el desarrollo del *software* se utilizó la herramienta *GNATcheck* a fin de verificar el cumplimiento por parte del *software* de los estándares de codificación requeridos por el estándar *DO-178B.*

El *ARBS* está equipado con un avanzado sistema automático de vaciado de la carga de combustible y una desconexión automática entre el cisterna y la aeronave reabastecida.

GNATcheck permite, de acuerdo con la compañía *AdaCore* (2011), que los desarrolladores definan un estándar de codificación como un conjunto de reglas, por ejemplo, para definir un subconjunto de características del lenguaje permitidas. [9]

Figura 39: *Airbus A330 MRTT* reabasteciendo a un *F-35A*

5.9 Sistemas de a bordo en avión de combate *Eurofighter Typhoon*

El *Eurofighter Typhoon* es considerado el avión de combate multirol más avanzado del mundo. Fruto de una colaboración entre Alemania, Italia, España y el Reino Unido, tiene un ciclo de vida estimado de 25 años.

Los sistemas de la aeronave están desarrollados en *Ada*, siendo el proyecto más importante de Europa en este lenguaje, con más de 500 desarrolladores dedicados. El primer tramo del proyecto contiene 1.500.000 líneas de código.

La empresa *BAE Systems* seleccionó a *GNAT Pro* para las áreas del proyecto críticas para la seguridad y la misión y a *VxWorks* de *Wind River*. El proyecto se dividió en la entrega de

dos tramos, el segundo de ellos contiene la capacidad completa de las características de ataque aire-tierra así como mejoras al *software* de control de vuelo y los sistemas de defensa, con un tiempo esperado de desarrollo de 3 años más un período de mantenimiento amplio de 15 años, según lo menciona la compañía *AdaCore* (2006). [10]

Figura 40: Avión de combate *Eurofighter Typhoon* de la Fuerza Aérea Austríaca

5.10 Sistemas del caza polivalente furtivo *Lockheed Martin F-35 Lightning II*

El avión de combate multipropósito con tecnología furtiva *F-35 Lightning II* es considerado la aeronave de combate de quinta generación más avanzada tecnológicamente, como también el programa de sistemas de armas más costoso de la historia.

La mayor parte del *software* está escrita en *C* y *C++* y una parte del código en *Ada83* es reusada a partir del *software* del avión de combate *F-22 Raptor*. La elección de *C* y *C++* se debió, de acuerdo con la firma *Lockheed Martin Corporation* (2005), a la creencia que la disponibilidad de programadores capacitados en este lenguaje sería un factor importante para el ciclo de desarrollo. [11]

El *software* corre bajo el sistema operativo en tiempo real *Integrity DO-178B* de la empresa *Green Hills Software*, utilizando procesadores *Freescale PowerPC*.

Con un estimado final de 24 millones de líneas de código, el ciclo de desarrollo ha sufrido repetidas demoras y sobrecostos.

Con un primer vuelo realizado en el 2006, se espera que entre en operaciones durante el 2015 para el *U.S. Marines Corps*, en el 2016 para la *USAF* y en el 2018 para la *U.S. Navy*. Otros países usuarios serán Australia, Israel, Italia, Japón, Holanda, Noruega, Corea del Sur, Turquía y el Reino Unido.

Figura 41*: Lockheed Martin F-35 Lightning II*

Figura 42: Cabina del *Lockheed Martin F-35 Lightning II*

Figura 43: Sistema de *display* integrado en el casco del piloto desarrollado para el *Lockheed Martin F-35 Lightning II*

5.11 Gestor de interfaz y radio control del sistema de gestión de vuelo de aeronave *C-130J Super Hercules*

La empresa *Lockheed Martin* utiliza *GNAT Pro High-Integrity Edition* para el bloque 7 de actualización del *software* del *C-130J*. Este bloque incluye un Gestor de interfaz y radio control para el sistema de gestión de vuelo. El *C-130J*, respecto a las versiones anteriores del *Hercules*, puede transportar 33% más de carga, utiliza la mitad de la tripulación de vuelo, consume menos combustible y es más rápido, con mayor autonomía y techo de servicio. El *software* corre bajo el sistema operativo *VxWorks 653* de *Wind River* y utiliza procesadores *PowerPC*, según lo detallado por la compañía *AdaCore* (2009). [12]

El *software* crítico está compuesto por 23 Unidades de Línea Reemplazables (*LRU*) con aproximadamente 500.000 líneas de código desarrolladas por 18 distintos proveedores utilizando lenguajes tales como *Ada, C, LUCOL, PL/M* y Ensamblador, entre otros. Fue desarrollado, de acuerdo con K. J. Harrison (2003), para cumplir con los requisitos definidos por el estándar *DO-178B.* [13]

Figura 44: *U.S. Air Force Lockheed Martin C-130J Super Hercules*, 2010

Figura 45: Cabina del *Lockheed Martin C-130J Super Hercules*, 2014

5.12 Pantalla del Sistema Avanzado de Información de Radar *TARDIS* para el avión de ataque *Panavia Tornado GR.4* de la *RAF*

La empresa *BAE Systems* utiliza la plataforma de propósito general *VxWorks Edition 5.5* de *Wind River* para el sistema *TARDIS* en los aviones de ataque de la *RAF Tornado GR.4*. El contrato, menciona Paul Parkinson (2004), comprende la entrega de 128 sistemas integrados con el radar por un monto de 70 millones de dólares. [14]

El sistema, detalla la compañía *Wind River* (2005), proporciona características sumamente avanzadas en cuanto a *software* para procesamiento de radar, gráficos y generación de mapas. [15]

TARDIS utiliza pantallas de cristal líquido de matriz activa, proporcionando mapas e imágenes digitales tanto para el piloto como para el navegador, utilizando librerías gráficas *OpenGL*.

El *software*, codificado en *C* y *C++*, fue desarrollado a partir del diseño utilizando *UML*.

Figura 46: Pantallas del navegador - *TARDIS* - *RAF Tornado GR.4*

Figura 47: *RAF Panavia Tornado GR.4*

5.13 Sistemas del avión de transporte y de reabastecimiento aéreo *Boeing KC-767*

La empresa *Smiths Aerospace*, actualmente parte de *GE Aviation Systems*, utilizó la plataforma crítica para la seguridad *ARINC 653* de *Wind River* para los sistemas del *Boeing KC-767*. El Sistema de Control de Misión (MCS) comprende los módulos de guiado en vuelo, navegación y comunicaciones.

La plataforma, de acuerdo con la compañía *Wind River* (2005), proporciona un entorno de desarrollo para poder implementar la Aviónica Modular Integrada (*IMA*) y facilita el proceso de certificación del estándar *DO-178B.* [16]

Figura 48: *Boeing KC-767* de la Fuerza Aérea Italiana reabasteciendo un *Boeing B-52H* de la *U.S. Air Force*, 2007

6. Contenidos a incorporar en la carrera de Ingeniería en Sistemas del IUA

En este capítulo, basados en las características identificadas respecto a los estándares y metodologías en los capítulos anteriores, postularemos que temas podrían ser incorporados en la carrera de Ingeniería en Sistemas del IUA para cubrir dichos aspectos (estadísticos, matemáticos y de ingeniería de *software*).

6.1 Aspectos Estadísticos

Estadística Bayesiana: en la gestión de proyectos de *software* en donde existe un factor de complejidad que involucra a un alto grado de incertidumbre, puede aplicarse la estadística Bayesiana, por ejemplo, para la evaluación de hipótesis múltiples.

Análisis de Modos y Efectos de las Averías (*FMEA*): el *FMEA* es un procedimiento de análisis de averías potenciales, en un sistema de clasificación determinado por la gravedad o por el efecto de los fallos en el sistema. Inicialmente desarrollado para la industria de la defensa, actualmente es utilizado en un gran número de industrias. Su finalidad es eliminar o reducir los fallos, comenzando por aquellos con una prioridad más alta. También puede ser usado para evaluar las prioridades de la gestión del riesgo.

Análisis Preliminar de Peligros (*PHA*): el *PHA* es usado para obtener una evaluación inicial de riesgos de los peligros del sistema. Los peligros son evaluados en función de su severidad, probabilidad y consecuencias operativas.

6.2 Aspectos Matemáticos

Respecto a los métodos a utilizar en las etapas de definición de requerimientos y diseño, los llamados **Métodos Formales** y **Métodos Semi-Formales** son fundamentales para diversas actividades tales como las relacionadas con V&V y contribuyen a reducir el costo y la dificultad para producir *software* confiable.

6.2.1 Métodos Formales

Los **Métodos Formales** involucran formalismos matemáticos y lógicos para representar rigurosamente y sin ambigüedad los documentos iniciales del sistema, incluyendo los requerimientos y el diseño. Estas representaciones pueden ser sometidas al razonamiento deductivo formal (algunas veces automatizado) a fin de detectar anomalías o defectos.

Petri nets: las redes de *Petri* son una representación matemática o gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuído, paralelo o concurrente. Entre otra áreas, son utilizadas para el análisis de datos, diseño de *software*, fiabilidad, flujo de trabajo y programación concurrente.

Las redes de *Petri* son usadas, entre otros, en los siguientes casos:

- Funciones para la detección de fallas y autodiagnóstico
- Creación de la especificación de requerimientos del *software*
- Validación y verificación estática
- Modelado de la arquitectura funcional
- Análisis de flujo de sincronización crítica
- Análisis de peligros del *software*

Lenguaje *B*: *B* es un lenguaje que se representa como un sistema sintáctico y semántico, incluyendo reglas de ambos tipos, y se basa en el análisis de la lógica de predicados.

Lenguaje *Z*: *Z* es un lenguaje formal utilizado en ingeniería de *software* para la especificación formal de un sistema de cómputo, como una fase previa al desarrollo del código fuente para el mismo en un lenguaje de programación determinado. Está basado en la teoría de los conjuntos, el cálculo lambda y la lógica de primer orden.

Lenguaje *Esterel*: *Esterel* es lenguaje sincrónico para el desarrollo de sistemas reactivos complejos, con un estilo de programación imperativo que permite expresar en forma sencilla tanto el paralelismo como la multitarea apropiativa.

Método de Desarrollo de Viena: es un formalismo matemático discreto para definir rigurosamente los procesos de especificación semántica, basado en especificaciones formales. Posee una semántica formal, permitiendo la demostración de las propiedades de un modelo con un alto grado de confiabilidad.

Lógica de Orden Superior (*HOL*): es una forma de lógica de predicados que se distingue de la llamada lógica de primer orden por tener cuantificadores adicionales y una semántica más fuerte.

Sistema de Verificación de Prototipos (*PVS*): es un lenguaje de especificación integrado con herramientas de soporte y con un demostrador de teoremas automatizado, cuyo núcleo consta de un tipo de lógica de orden superior.

6.2.2 Métodos Semi-Formales

Los **Métodos Semi-Formales** comprenden técnicas cuyos métodos de uso normales, forzados o prescritos restringen efectivamente a los usuarios en su especificación, requerimientos o diseños, de manera que diversos problemas de expresión y elaboración pueden ser evitados o reducidos. Tales problemas incluyen aspectos de ambigüedad, falta de completitud, inconsistencia, contradicción y malformaciones. Estas técnicas, frecuentemente basadas en formalismos lógicos y matemáticos, no requieren explícitamente que el usuario especifique o use estos formalismos. Típicamente están embebidas en entornos ricos en funciones, basados en computadora, los cuales proporcionan representaciones gráficas sofisticadas de las entradas del usuario y frecuentemente admiten que las especificaciones del usuario sean simuladas o animadas para permitir la evaluación de las características de tiempo y rendimiento. Un ejemplo de este tipo de métodos es nuestro conocido *UML*.

Método de *Hatley-Pirbhai*: técnicas desarrolladas por las compañías *Boeing* and *Lear*, enfatizando los diagramas de flujos de control, con una buena capacidad de modelamiento de la arquitectura. Está basada en el modelo *IPO*, utilizando cinco componentes básicos: entradas, salidas, interfaz de usuario, mantenimiento y procesamiento.

IBM Rational Statemate: es una solución de prototipos, simulación y diseño gráfico para el desarrollo rápido de sistemas complejos embebidos. Permite la detección de errores desde el principio del ciclo de vida del desarrollo y la creación de prototipos de soluciones.

Metodología de Ingeniería de Requerimientos del Sistema (*SREM*): un lenguaje de especificación para *hardware* y *software* que describe tanto los flujos de datos como los flujos de control del sistema, aplicando reglas de descomposición funcional para reconocer las secuencias de tiempo de las entradas y salidas de las funciones, pudiendo aprovechar las oportunidades del procesamiento de datos en paralelo.

Lenguaje de Enunciado de Problemas/Analizador de Enunciado de Problemas (*PSL/PSA*): es una representación de requerimientos y especificaciones en lenguaje natural restringido con soporte automatizado. Basado en el modelo *ERA*, permite hasta cuatro tipos de entidades participando en una relación. *PSA* es el *software* que gestiona los enunciados

PSL, con un mecanismo sencillo para cargar una base de datos, recuperar la información de la misma y presentar los resultados en la forma de reportes.

6.3 Aspectos de la Ingeniería de *Software*

Entendiendo a la Ingeniería del *Software* como la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del *software*, integrando definiciones matemáticas, de ciencias de la computación y prácticas de ingeniería, enunciaremos a continuación diversos aspectos relacionados con el desarrollo de *software* crítico para la seguridad

6.3.1 Gestión de la Seguridad

Además de los aspectos estadísticos ya mencionados (*FMEA* y *PHA*), podemos incluir como elementos importantes respecto a la gestión de la seguridad a los siguientes:

- Plan de Seguridad del *Software* (*SSP*)
- Ciclo de vida de Seguridad del *Software*
- Evaluación Preliminar de la Seguridad del Sistema (*PSSA*)
- Análisis de Árbol de Fallos del *Software (SFTA)*
- *Software* para la Detección, Aislamiento y Recuperación de Averías (*FDIR*)
- Análisis de Peligros
- Matriz de criticidad de seguridad del *software*
- Plan para el Programa de Seguridad del *Software (SWSPP)*
- Flujo de proceso de seguridad del *software*
- Requerimiento de Seguridad del Sistema *(SSR)*

6.3.2 Gestión del Riesgo

Podemos incluir como elementos importantes respecto a la gestión del Riesgo a los siguientes:

- Análisis de riesgos
- Gestión de riesgos
- Plan de Gestión del Riesgo
- Análisis de Gestión del Riesgo
- Sistema de Gestión de Riesgos
- Proceso de Gestión de Riesgos
- Clasificación de los riesgos
- Análisis de Peligros y Evaluación de Riesgos (*HARA*)
- Análisis de riesgos particulares (*PRA*)
- Índice de riesgos del sistema
- Evaluación de riesgos
- Matriz de evaluación del riesgo

6.3.3 Gestión de la Calidad

Podemos incluir como elementos importantes respecto a la gestión de la Calidad a los siguientes:

- Aseguramiento de la Calidad del *Software* (*SQA*)
- Plan de Aseguramiento de la Calidad del *Software* (*SQAP*)
- Sistema de Gestión de la Calidad
- Registros de Aseguramiento de la Calidad del *Software (SQAR)*
- Programa de Aseguramiento de la Calidad del *Software*

6.3.4 Gestión de las Pruebas

Podemos incluir como elementos importantes respecto a la gestión de las Pruebas a los siguientes:

- Procedimientos y Reporte de las Pruebas Unitarias, de Integración del Sistema y de Validación
- Analizador de pruebas de ejecución
- Cobertura de las pruebas
- Especificación de las pruebas de requerimientos del *software*
- Planificación de las Pruebas

6.3.5 Gestión de la Configuración

Podemos incluir como elementos importantes respecto a la gestión de la Configuración a los siguientes:

- Plan de Gestión de la Configuración del *Software* (*SCMP*)

7. La creación de una red de desarrolladores de *software* (REDES)

En este capítulo, revisaremos en primer lugar el concepto del *software* de código abierto y las implicancias del desarrollo colaborativo, para luego plantear la utilización del llamado *OSS* en aplicaciones críticas para la seguridad.

Finalmente, elaboraremos la idea de la creación de una Red de Desarrolladores de *Software*, que permita ampliar la oferta de recursos humanos familiarizados con los desafíos que presenta el desarrollo de sistemas críticos para la seguridad y su aplicación en áreas estratégicas para la Nación y el futuro de una sociedad moderna.

7.1 El paradigma del *software* de código abierto (*OSS*) y el desarrollo colaborativo

El *software* de código abierto es aquel cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de *software* compatible con la denominada *Open Source Definition* o bien forman parte del dominio público. Esto permite que los usuarios utilizen, cambien, mejoren y redistribuyan el *software*, ya sea en su forma modificada o en su forma original. Generalmente, el *software* se desarrolla de manera colaborativa.

La más difundida de las licencias de *OSS* compatible con la *Open Source Definition* es la *GNU* Licencia Pública General (*GPL*).

El concepto básico del *OSS* es que cuando los programadores pueden leer, modificar y redistribuir el código fuente de un programa, éste evoluciona, se desarrolla y mejora. Los usuarios los adaptan a sus necesidades, corrigen sus errores con un tiempo de espera menor a la aplicada en el desarrollo de *software* convencional o cerrado, dando como resultado la producción de un mejor *software*, incrementando su seguridad y estabilidad.

El desarrollo exponencial en las aplicaciones de *software* ha sido acompañado por el incremento de recursos humanos con las habilidades necesarias para desarrollar el *software*. Esto implica que una inmensa cantidad de talento potencial, en la figura de millones de desarrolladores, se encuentra diseminada globalmente y en los diferentes sectores de las sociedades modernas. El aprovechar ese talento disponible en forma masiva es una de las grandes metas del desarrollo colaborativo.

Otra de las ventajas importantes es la gran reducción de costos por el uso del *software* ya que no debe pagarse por compras o renovaciones de licencias, ni se encuentra el usuario ante una situación de un propietario monopólico que pueda imponer condiciones comerciales excesivamente onerosas.

Sin embargo, existen también diversos inconvenientes que surgen a partir de la creciente tendencia al desarrollo y uso del *OSS*.

Un *software* cerrado o privativo tiene un propietario de una licencia, el cual generalmente provee un soporte a los usuarios respecto a cualquier inconveniente con el *software* y, por lo tanto, conlleva también la responsabilidad legal ante cualquier malfuncionamiento o problemas generados por ese *software*. En el *software* abierto, esa responsabilidad se diluye en la forma de un grupo de desarrolladores que bien pueden encontrarse en distintos países, sometidos a distintas legislaciones y responsabilidades implícitas.

Asimismo, al ser el desarrollo de *OSS* en general una actividad sin un fin de lucro, puede suceder que el *software* desarrollado deba ser abandonado por falta de financiación, y en definitiva dependerá del mayor o menor interés de la comunidad abierta de desarrolladores en mantenerlo y mejorarlo.

7.2 La utilización de *OSS* y su aplicación en los sistemas críticos para la seguridad

Numerosos países, en la forma de su administración pública y actividades de carácter gubernamental, y compañías privadas se vuelcan cada vez más a la implementación y uso de *OSS*. Sin duda, sus ventajas tienen un gran impacto en la reducción de costos de los presupuestos tanto públicos como privados.

Sin embargo, debemos plantearnos la cuestión de la factibilidad de aplicar *OSS* en sistemas críticos para la seguridad. En una primera aproximación, podríamos plantear que siendo una de las principales ventajas del desarrollo colaborativo y abierto el contar con un *software* mejorado, naturalmente sería deseable contar con una característica semejante en sistemas críticos para la seguridad, en donde se incremente la misma y la estabilidad del *software*. Podríamos pensar, por ejemplo, en diversas aplicaciones para los sectores de Infraestructura, Medicina, Transporte e Industria, las cuales serían de enorme beneficio para la sociedad.

Ahora bien, debemos preguntarnos ¿ qué ocurre con los sectores considerados estratégicos por un país ? Un *software* aplicado, por ejemplo, al área de la Defensa, el cual conlleva una ventaja estratégica significativa ¿ es factible colocarlo como de dominio público ? O, con un criterio más amplio ¿ es conveniente para los intereses estratégicos de una Nación determinada distribuir el *software* en forma libre, con la posibilidad que dicho *software* termine en manos de otra Nación que pueda ser un rival potencial ?

Pongamos por caso lo visto respecto al avión de combate de quinta generación *Lockheed Martin* F-35 *Lightning II* cuyo *software* es tan crítico y fundamental, que se convirtió en el eje de una disputa entre los Estados Unidos y potenciales compradores del avión a raíz de la negativa estadounidense en proporcionar el código fuente como parte de la transferencia de tecnología.

Vemos entonces que, independientemente de la factibilidad y conveniencia del uso de *OSS*, existen algunos sectores en los sistemas críticos para la seguridad en los cuales la aplicación de las políticas de *software* abierto y libremente distribuído no sería recomendable. Claro que ellos dependerá de la percepción de la importancia del *software* en cuestión como un logro tecnológico significativo que implique una ventaja estratégica para una Nación.

No obstante lo mencionado, existen diversas iniciativas que abogan por el uso irrestricto de *OSS*, aun en aplicaciones militares. El *DoD* de los Estados Unidos publicó en el año 2006 un documento [1] en el cual se establece que "…*OSS y las metodologías de desarrollo de código abierto son importantes para la seguridad nacional y los intereses nacionales de los Estados Unidos debido a las siguientes razones:*

- *Mejora la agilidad de las industrias de tecnologías de la información para efectuar adaptaciones y cambios más rápidamente, en función de las capacidades que necesitan los usuarios*
- *Fortalece a la industria relacionada al permitir la competencia y no basarse en productos cerrados*
- *Permite al DoD asegurar la infraestructura e incrementar la seguridad mediante el entendimiento de qué es lo que hay realmente en el código fuente del software instalado en sus redes*

- *Responder rápidamente a las acciones de los adversarios como también actualizar la tecnología utilizada…*"

Como en muchos otros sectores, se trata de obtener un balance adecuado en la creación de *OSS* versus el uso de *software* cerrado. Ciertamente, puede desarrollarse *software* crítico para la seguridad utilizando para ello *OSS*, pero existirán casos particulares en los cuales el producto final o ciertos algoritmos del mismo no podrán ser de dominio público.

La clave consistirá entonces en poder aprovechar la ventaja de la filosofía de los *OSS*, especialmente en lo referido al desarrollo colaborativo, mientras que se mantiene un producto final fuera del dominio público.

7.3 La Red de Desarrolladores de *Software* (REDES) y los sistemas críticos para la seguridad

Los desafíos implícitos respecto al desarrollo colaborativo de *OSS* es poder lograr la madurez del *software* y un desarrollo y mejora del mismo contínuo en el tiempo, manteniendo el compromiso y participación del grupo de desarrolladores. Dado que no existe un fin de lucro en el desarrollo de esta actividad, se necesita de un alto entusiasmo y vocación de participar por parte de los desarrolladores.

Diversos factores pueden contribuir a ello: el obtener prestigio profesional, el hallarse actualizado respecto a nuevas tecnologías y tendencias, la necesidad de participar en un grupo de profesionales con intereses comunes, etc. Sin embargo, existe un factor muy importante y que no podemos soslayar: el amor por la Patria, el deseo de dar lo mejor profesionalmente por el bien de la Nación y de la Comunidad.

La Fuerza Aérea Argentina tiene un ejemplo de un accionar semejante en la figura de la ROA (Red de Observadores del Aire) la cual convoca a civiles con vocación de servicio y conocimientos técnicos en comunicaciones y electrónica para formar parte de la misma. De destacadísima e importante labor durante la Guerra de las Malvinas, la ROA puede ser convocada en circunstancias de conflicto externo u otras necesidades operativas.

En el caso de la propuesta REDES, podría estar integrada por estudiantes y profesionales de las carreras de tecnologías de la información de las Universidades Nacionales, bajo la órbita del Ministerio de Defensa u otra organización gubernamental. A los conocimientos metodológicos comunes, será necesario incorporar los aspectos específicos descriptos sobre

los Sistemas Críticos para la Seguridad, los cuales presentan puntos en común con una especialización de la carrera como podría ser la de sistemas embebidos.

Existen ejemplos de agrupaciones similares en diversos países, por ejemplo Estados Unidos y China:

- *Mil-OSS* es una organización estadounidense, descripta como una "*…comunidad patriótica activa…*", que conecta y alienta a desarrolladores de *software*, tanto civiles como militares, en el uso y mejora de *OSS* a través del *DoD* de los Estados Unidos. Sus objetivos son mejorar la seguridad del *software*, acotar los costos de desarrollo e incrementar la innovación a través del desarrollo colaborativo [2]

- Fundada en el 2004, la Red de Desarrolladores de *Software* China (*CSDN*), proporciona foros *Web*, *blogs*, noticias de *IT*, capacitación y otros servicios a más de 10 millones de usuarios registrados, en donde predomina una cultura de compartir el conocimiento y ayudar a los miembros de la Red.

8. Conclusiones

Hemos visto en los capítulos anteriores las características del *software* crítico para la seguridad y sus aplicaciones en las distintas áreas de una sociedad moderna junto con la importancia de los estándares para la certificación del cumplimiento de las normas correspondientes. Posteriormente hemos profundizado particularmente en una de esas áreas, la Aviación, y en el desarrollo de *software* crítico para la seguridad que cumpla con uno de los principales estándares, la serie *DO-178B/DO-178C*.

Hemos enunciado los desafíos y problemas que surgen a partir del desarrollo de *software* con las características mencionadas y la necesidad de aplicar nuevos métodos y herramientas para poder resolver o mitigar los efectos negativos respecto a costos y productividad que se generan.

Para ello, hemos analizado una variedad de herramientas y aplicaciones que están especialmente diseñadas para aumentar la productividad, reducir los costos involucrados especialmente con las actividades de V&V y facilitar los procesos de certificación de los estándares internacionales. Adicionalmente, hemos mostrado varios ejemplos de implementación exitosa de sistemas críticos para la seguridad en Aviación, a partir de los estándares, métodos y herramientas mencionados.

Debemos preguntarnos ahora ¿qué garantiza que un *software*, cuyo fallo podría tener consecuencias catastróficas, se encuentra libre de errores? Podríamos decir, con fundamento, que si el 100% de las líneas de código han sido probadas, podemos asegurar que se encuentra libre de errores. Sin embargo, debemos recordar que las pruebas demuestran la presencia de defectos pero no pueden demostrar que no los hay. Las mismas reducen la probabilidad de que existan defectos ocultos en el *software*, pero aunque no hayamos detectado ninguno, no constituye una evidencia de corrección.

Podríamos entonces contar con las mejores herramientas y métodos de V&V, pero esto sólo por sí mismo no bastaría para garantizar que el *software* cumplirá con los requisitos de seguridad necesarios. Pues bien, ¿qué es necesario entonces?

En primer lugar, una guía, un documento que, otorgando la flexibilidad necesaria, nos oriente hacia el cumplimiento de determinados requisitos cuya obligatoriedad está directamente relacionada con la criticidad de las aplicaciones en cuestión, estableciendo determinadas pautas, aunque sin mencionar en forma taxativa la forma y los medios para

cumplir con ellas. Y precisamente *DO-178B/DO-178C*, en su simplicidad y siendo básicamente un guía que establece lineamientos generales, ha pasado a ser *de facto* el estándar utilizado y reconocido mundialmente para el desarrollo de *software* para los sistemas de las aeronaves.

Pues bien, tenemos ya identificados los lineamientos y las herramientas necesarias ¿es eso suficiente? Aun debemos decir que no, ya que son condiciones necesarias pero no suficientes. De hecho, la serie *DO-178* por sí misma no está pensada para garantizar que los aspectos de seguridad del software se cumplen. La propia naturaleza flexible de *DO-178B/DO-178C* hace difícil su implementación por primera vez, debido a que los aspectos tratados son abstractos y no proporciona un conjunto de actividades básicas a partir de la cual comenzar.

Entonces, podemos preguntarnos ¿cuál es la clave en este tipo de desarrollos? Pues la clave, precisamente, es que sigamos un enfoque sistémico, en donde los distintos elementos que hemos identificado como necesarios sean utilizados en conjunto y permitan demostrar a la autoridad de certificación correspondiente que hemos cumplido con las pautas establecidas por las regulaciones de los aspectos del *software* incluido en los sistemas de aeronaves.

Ninguno de los elementos mencionados, por sí sólo, puede garantizar el cumplimiento de las requerimientos demandados, sino el conjunto de ellos, utilizados bajo una serie de lineamientos adecuados.

Toda organización dedicada al desarrollo de *software* crítico para la seguridad debe analizar los estándares y lineamientos establecidos por las autoridades correspondientes y capacitar a su personal en sus particularidades, tanto por motivos legales como de naturaleza comercial ya que actualmente es imposible el poder comercializar un *software* crítico para la seguridad que no haya recibido la certificación de la autoridad correspondiente.

En este sentido, es recomendable la incorporación de estos contenidos dentro de los programas de estudio correspondientes, que permitan, al menos, una introducción de los futuros profesionales de las carreras de Sistemas en la temática y peculiaridades de los sistemas críticos para la seguridad y el desarrollo de *software* para los mismos.

De esta manera, nuestro país podrá cubrir los requerimientos de personal capacitado en esta problemática, cuya demanda sin dudas se irá incrementando en el futuro cercano al incrementarse la importancia de esos sistemas para el funcionamiento de una sociedad moderna.

Apéndice A. Listado de símbolos y convenciones

A

4WD: Tracción en las cuatro ruedas (*Four-wheel drive*)

AAMI: Asociación para el Avance de la Instrumentación Médica (*Association for the Advancement of Medical Instrumentation*)

AATE: Asociación Argentina de Tecnología Espacial

ABS: Sistema de antibloqueo de frenos (*Anti-lock Braking System*)

ABWR: Reactor Nuclear Avanzado de Agua en Ebullición (*Advanced Boiling Water Reactor*)

ACAMS: Sistema de análisis y gestión de las condiciones de la aeronave (*Aircraft Condition Analysis and Management System*)

ACE: Condiciones y Eventos Anormales (*Abnormal Conditions and Events*)

ADAS: Sistemas Avanzados de Asistencia al Conductor (*Advanced Driver Assistance Systems*)

ADD: Documento del Diseño de la Arquitectura (*Architectural Design Document*)

ADIRU: Unidad de Referencia Inercial de Datos Aéreos (Air Data Inertial Reference Unit)

AEA: Asociación Electrotécnica Argentina

AEC: Comisión de Energía Atómica (*Atomic Energy Commission*, ver *USNRC*)

AECL: Compañía de Energía Atómica del Canadá (*Atomic Energy of Canada Limited*)

AFB: Base de la *USAF* (*Air Force Base*)

AFISC: Centro de Inspección y Seguridad de la Fuerza Aérea de los Estados Unidos (*Air Force Inspection and Safety Center*)

AGV: Vehículo Guiado Automático (*Automated Guided Vehicle*)

AIAA: Instituto Estadounidense para la Aeronáutica y la Astronáutica (*American Institute of Aeronautics and Astronautics*)

AMC: Comando de Materiales del Ejército de los Estados Unidos (*Army Materiel Command*)

ANS: Sociedad Nuclear de Estados Unidos (*American Nuclear Society*)

ANS: Sistemas de Navegación Aérea (*Air Navigation Systems*)

ANSI: Instituto de Estándares Nacionales de Estados Unidos (*American National Standards Institute*)

ANSP: Proveedor de Servicios de Navegación Aérea (*Air Navigation Service Provider*)

AOCS: Sistema de Control de Actitud y Órbita (*Attitude and Orbit Control System*)

API: Interfaz de Programación de Aplicaciones (*Application Programming Interface*)

ARBS: Sistema de Lanza de Reabastecimiento en Vuelo (*Airbus Military Aerial Refueling Boom System*)

ARP: Prácticas Aeroespaciales Recomendadas (*Aerospace Recommended Practices*)

ASA: Evaluación de Seguridad de la Aeronave (*Aircraft Safety Assessment*)

ASC: Control de Estabilidad Automático (*Automatic Stability Control*)

ASCM: Misiles de Crucero Antibuques (*Anti-Ship Cruise Missile*)

ASICS: Circuitos Integrados para Aplicaciones Específicas (*Application-Specific Integrated Circuits*)

ASIL: Nivel de Integridad de la Seguridad de los Automotores (*Automotive Safety Integrity Level*)

ASME: Sociedad Estadounidense de Ingenieros Mecánicos (*American Society of Mechanicals Engineers*)

ATC: Control de Tráfico Aéreo (*Air Traffic Control*)

ATM: Gestión de Tráfico Aéreo (*Air Traffic Management*)

ATM: Gestión de Tráfico Activa (*Active Traffic Management*)

ATO: Operación Automática de Tren (*Automatic Train Operation*)

ATP: Protección automática de trenes (*Auto Train Protection*)

ATS: Servicios de Tráfico Aéreo (*Air Traffic Services*)

B

BCS: Sistema de Control de Bloqueo (*Block Control System*)

BFS: Sistema de Vuelo de Respaldo (*Backup Flight System*)

BIT: Autoexamen (*Built-in Test*)

BOS: Sistema de Soporte a Decisiones en Holanda (*Beslis- en Ondersteunend Systeem*)

BPCS: Sistema de Control de Proceso Básico (*Basic Process Control System*)

BS: Estándares Británicos (*British Standards*)

C

C4ISR: Comando, Control, Comunicaciones, Computadoras, Inteligencia, Vigilancia y Reconocimiento (*Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance*)

CA: Evitación de Colisión (*Collision Avoidance*)

CAD: Diseño Asistido por Computador (*Computer-aided Design*)

CAN: Red de Área de Controlador (*Controller Area Network*)

CANDU: Deuterio Uranio del Canadá (*Canada Deuterium Uranium*)

CASE: Ingeniería de *Software* Asistida por Computadora (*Computer Aided Software Engineering*)

CBI: Enclavamiento Controlado por Computadora (*Computer Based Interlocking*)

CBTC: Control de Trenes basado en Comunicaciones (*Communications-Based Train Control*)

CCA: Análisis de Causas Comunes (*Common Cause Analysis*)

CCF: Falla de causa común (*Common Cause Failure*)

CCS: Sistema de Control Central (*Central Control System*)

C&DH: Comando y Manejo de Datos (*Command & Data Handling*)

CEA: Comité Electrotécnico Argentino

CECOM: Comando de Comunicaciones Electrónicas del Ejército de los Estados Unidos (*U.S. Army Communications-Electronics Command*)

CENELEC: Comité Europeo de Normalización Electrotécnica (*Comité Européen de Normalisation Electrotechnique)*

CGI: Elemento de Calidad Comercial (*Commercial Grade Item*)

CI: Inspección de la Configuración (*Configuration Inspection*)

CIA: Agencia Central de Inteligencia de los Estados Unidos (*Central Intelligence Agency*)

CIC: Centro de Información de Combate (*Combat Information Center*)

CITeA: Centro de Investigación y Desarrollo de Tecnologías Aeronáuticas

CMM: Modelo de Madurez de Capacidades (*Capability Maturity Model*)

CMMI: Integración del Modelo de Madurez de Capacidades (*Capability Maturity Model Integration*)

CMMS: Sistema de Gestión de Configuración y Mantenimiento (*Configuration and Maintenance Management System*)

CNS: Red de Comunicaciones en los Ferrocarriles (*Communication Network System*)

CNS: Comunicación, Navegación y Vigilancia (*Communication, Navigation and Surveillance*)

COMAU: Consorcio de Máquinas y Herramientas (*COnsorzio MAcchine Utensili)*

CONAE: Comisión Nacional de Actividades Espaciales

COPANT: Comisión Panamericana de Normas Técnicas

COTS: Programas Informáticos Comerciales (*Commercial Off-The-Shelf*)

CPU: Unidad Central de Procesamiento (*Central Processing Unit*)

CR: Requerimiento de Cambio (*Change Request*)

CRU: Uso de los Recursos de Computación (*Computer Resource Utilization*)

CSA: Agencia Espacial Canadiense (*Canadian Space Agency*)

CSDN: Red de Desarrolladores de *Software* China (*Chinese Software Developers Network*)

D

DAL: Nivel de Garantía del Diseño (*Design Assurance Level*)

DAL: Nivel de Garantía del Desarrollo (*Development Assurance Level*)

DD: Diagrama de Dependencia (*Dependence Diagram*)

DER: Representante de Ingeniería Designado (*Designated Engineering Representative*)

DID: Documentación de entrada para el Diseño (*Design Input Documentation*)

DIGAMC: Dirección General de Aeronavegabilidad Militar Conjunta

DIGID: Dirección General de Investigación y Desarrollo de la FAA

DLA: Análisis Lógico del Diseño (*Design Logic Analysis*)

DO: Estándares de la Comisión Radiotécnica para la Aeronáutica (*Radio Technical Commision for Aeronautics*)

DOE: Departamento de Energía de los Estados Unidos (*Department of Energy*)

DPCS: Sistema de Procesamiento de Datos y Control (*Data Processing and Control System*)

DR: Reporte de Discrepancias (*Discrepancy Report*)

DSTG: Grupo para la Ciencia y Tecnología de la Defensa de Australia (*Defence Science and Technology Group*)

DSTL: Laboratorio de Ciencia y Tecnología para la Defensa (*Defence Science and Technology Laboratory*)

E

EASA: Agencia Europea de Seguridad Aérea (*European Aviation Safety Agency*)

EBU: Unidad de Gestión Eléctrica (*Electric Brain Unit*)

ECC: Verificación y Corrección de Errores (*Error Checking and Correcting*)

ECHA: Análisis de Peligros de las Condiciones Ambientales (*Environmental Condition Hazard Analysis*)

ECU: Unidad de Control Motriz (*Engine Control Unit*)

ECU: Unidad de Control Electrónica (*Electronic Control Unit*)

EDL: Entrada, Descenso y Aterrizaje (*Entry, Descent and Landing*)

EFIS: Sistema Electrónico de Instrumentos de Vuelo (*Electronic Flight Instruments System*)

EGNOS: Sistema Europeo de Navegación por Superposición Geoestacionaria (*European Geostationary Navigation Overlay System*)

EICAS: Sistema de Indicación de Motor y Aviso a la Tripulación (*Engine Indicating and Crew Alerting System*)

EMI: Interferencia Electromagnética (*Electromagnetic Interference*)

EN: Estándares Europeos (*European Standards*)

ENG-SE-SPE: Especificaciones del Grupo de Servicio de Ingeniería para el Departamento de Ingeniería de Señales en los ferrocarriles de Australia (*Engineering Service Group - Signal Engineering Department Specification*)

EPRI: Instituto de Investigación en Energía Eléctrica (*Electric Power Research Institute*)

ERA: Modelo de Entidad-Relación-Atributo (*Entity-Relationship-Attribute Model*)

ESA: Agencia Espacial Europea (*European Space Agency*)

ESAS: Sistema de Accionamiento de la Seguridad de Emergencia (*Emergency Safety Actuation System*)

ESF: Característica de Seguridad de Ingeniería (*Engineered Safety Feature*)

ESP: Control de estabilidad (*Electronic Stability Program*)

ESS: Detección de Stress del Ambiente (*Environmental Stress Screening*)

ETCS: Sistema de Control de Trenes Europeo (*European Train Control System*)

EURECA: Portador Recuperable Europeo (*European Retrievable Carrier*)

EUROCONTROL-SAM: Metodología de Evaluación de la Seguridad de la Organización Europea para la Seguridad de la Aeronavegación (*European Organization for the Safety of Air Navigation - Safety Assessment Methodology*)

F

FAA: Fuerza Aérea Argentina

FAA: Administración Federal de Aviación de los Estados Unidos (*Federal Aviation Administration*)

FAB: Fuerza Aérea Brasileña

FAC: Comité Federal Consultivo de los Estados Unidos (*Federal Advisory Committee*)

FACI: Inspección de Configuración del Artículo Original (*First Article Configuration Inspection*)

FADEC: Control Electrónico Digital de Autoridad Total (*Full Authority Digital Electronic Control*)

FAM: Modelo de Arquitectura Funcional (*Functional Architecture Model*)

FAT: Pruebas de Aceptación de Fábrica (*Factory Acceptance Test*)

FBD: Diagrama de Bloques de Función (*Function Block Diagram*)

FCS: Sistema de Combate Futuro del Ejército de los Estados Unidos (*Future Combat System*)

FDA: Agencia Federal de los Estados Unidos para la Administración de Alimentos y Medicamentos (*Food and Drug Administration*)

FDIR: Detección, Aislamiento y Recuperación de Averías (*Failure Detection, Isolation, and Recovery*)

FIR: Región de Información de Vuelo (*Flight Information Region*)

FIS: Servicios de Información de Vuelo (*Flight Information Services*)

FIT: Pruebas de Inserción de Fallas (*Fault Insertion Testing*)

FIT: Pruebas de Inmunidad de Averías (*Failure Immunity Testing*)

FMEA: Análisis de Modos y Efectos de las Averías (*Failure Modes and Effects Analysis*)

FMS: Sistema de Gestión de Vuelo (*Flight Management System*)

FPA: Análisis de Puntos de Función (*Function Point Analysis*)

FPA: Averías por año (*Failures per annum*)

FPGA: Matrices de Puertas Programables (*Field Programmable Gate Arrays*)

FRE: Unidad Reemplazable de Campo (*Field Replaceable Unit*)

FTA: Análisis de Árbol de Fallas (*Fault Tree Analysis*)

FVL: Lenguaje de variabilidad completa (*Full Variability Language*)

FY: Año Fiscal (*Fiscal Year*)

G

GAO: Oficina de Responsabilidad Gubernamental de los Estados Unidos (*Government Accountability Office*)

GESTALT: Estimación Basada en Grillas de la facilidad de Tránsito de una Superficie Aplicada al Terreno Local (*Grid-based Estimation of Surface Traversability Applied to Local Terrain*)

GEU: Unidad de Guiado Electrónico (*Guidance Electronics Unit*)

GLONASS: Sistema Satelital de Navegación en Órbita Global (*Global Orbiting Navigation Satellite System*)

GMES: Monitoreo Global para el Medio Ambiente y la Seguridad (*Global Monitoring for Environment and Security*)

GN&C: Guiado, Navegación y Control (*Guidance, Navigation and Control*)

GNSS: Sistema Satelital de Navegación Global (*Global Navigation Satellite System*)

GPC: Computadora de Propósito General (*General Purpose Computer*)

GPL: Licencia Pública General (*General Public License*)

GPS: Sistema de Posicionamiento Global (*Global Positioning System*)

GPWS: Sistema de Alerta de Proximidad al Suelo (*Ground Proximity Warning System*)

GPU: Unidad de Procesamiento de Gráficos (*Graphics Processing Unit*)

GUI: Interfaz Gráfica de Usuario (*Graphical User Interface*)

H

HA: Dirección de Caminos en Inglaterra (*Highways Agency*)

HAL/S: Lenguaje Ensamblador de Alto Nivel (*High-Order Assembly Language*)

HAR: Reporte de Análisis de Peligros (*Hazards Analysis Report*)

HARA: Análisis de Peligros y Evaluación de Riesgos (*Hazard Analysis and Risk Assessment*)

HCSS CG: Grupo de Coordinación para el *Software* y Sistemas de Alta Confiabilidad (*High Confidence Software and Systems Coordinating Group*)

HEA: Análisis de Errores Humanos (*Human Error Analysis*)

HEV: Vehículo de Motor Híbrido (*Hybrid-Engine Vehicle*)

HLR: Requerimiento de Alto Nivel (*High Level Requirement*)

HOL: Lógica de Orden Superior (*Higher Order Logic*)

HOL: Lenguaje de Alto Nivel (*High Order Language*)

HMI: Interacción Hombre-Máquina (*Human-Machine Interaction*)

HRT: Estrictamente en Tiempo Real (*Hard Real-Time*)

HRTM: Modelo del Tracto Respiratorio Humano (*Human Respiratory Tract Model*)

HSI: Indicador de Situación Horizontal (*Horizontal Situation Indicator*)

HTTP: Protocolo de Transferencia de Hipertexto (*Hypertext Transfer Protocol*)

HUD: Sistema de visualización frontal de datos (*Head-up display*)

Hz: Hertz, unidad de frecuencia

I

IAEA: Agencia Internacional para la Energía Atómica (*International Atomic Energy Agency*)

I&C: Instrumentación y Control (*Instrumentation and Control*)

ICE: Emulación en circuito (*In-circuit Emulation*)

ICS: Estado de Colisión Inevitable (*Inevitable Collision State*)

IDA: Área de Intercomunicación de Datos (*Intercommunication Data Area*)

IDE: Entorno de Desarrollo Integrado (*Integrated Development Environment*)

IEC: Comisión Electrotécnica Internacional (*International Electrotechnical Commission*)

IECCA: Comité Inter-Establecimiento de Aplicaciones Informáticas (*Inter-Establishment Committee on Computer Applications*)

IEE: Instituto de Ingenieros Eléctricos (*Institution of Electrical Engineers*)

IEEE: Instituto de Ingeniería Eléctrica y Electrónica (*Institute of Electrical and Electronics Engineers*)

IET: Instituto de Ingeniería y Tecnología (*Institution of Engineering and Technology*)

IFAC: Federación Internacional para la Automatización de Controles (*International Federation of Automatic Control*)

iFACT: (*interim Future Area Controls Tools Support*)

IFIP: Federación Internacional para el Procesamiento de la Información (*International Federation for Information Processing*)

IFSTTAR: Instituto Francés para las Ciencias y Tecnologías del Transporte y su planificación, desarrollo y redes (*Institut français des sciences et technologies des transports, de l'aménagement et des réseaux*)

IHA: Análisis de Peligros Intrínsecos (*Intrinsic Hazard Analysis*)

IIE: Instituto de Ingenieros Incorporados (*Institution of Incorporated Engineers*)

IMA: Aviónica Modular Integrada (*Integrated Modular Avionic*)

IMI: Encendido del motor por error (*Inadvertent Motor Ignition*)

IMS: Sistemas de Fabricación Integrados (*Integrated Manufacturing Systems*)

IMU: Unidad de Medición Inercial (*Inertial Measurement Unit*)

INMAE: Instituto Nacional de Medicina Aeronáutica y Espacial

IoT: Internet de las Cosas (*Internet of Things*)

IPF: Facilidad de Procesamiento de Integridad (*Integrity Processing Facility*)

IPO: Modelo de Entrada-Proceso-Salida (*Input-Process-Output Model*)

IPR: Revisión a lo largo del Proceso (*In-Process Review*)

IRAM: Instituto de Racionalización Argentino de Materiales, actualmente denominado Instituto Argentino de Normalización y Certificación

ISA: Sociedad Internacional de Automatización (*International Society of Automation*)

ISO: Organización Internacional de Normalización (*International Organization for Standardization*)

ISS: Estación Espacial Internacional (*International Space Station*)

ITEA: Tecnología de la Información para el progreso europeo (*Information Technology for European Advancement)*

ITP: Procedimientos de Pruebas de Integración (*Integration Test Procedures*)

ITR: Reporte de Pruebas de Integración (*Integration Test Report*)

IUA: Instituto Universitario Aeronáutico

IVA: Validación Independiente (*Independent Validation*)

J

JNES: Organización para la Seguridad en Energía Nuclear del Japón (*Japan Nuclear Energy Safety*)

JPL: Laboratorio de Propulsión a Chorro (*Jet Propulsion Laboratory*)

JSF: Avión de Ataque Conjunto (*Joint Strike Fighter*)

JSOW: Armamento Táctico de alcance medio (*Joint Stand-Off Weapon*)

K

KB: Base de Conocimiento (*Knowledge Base*)

KLOC: Miles de Líneas de Código (*Kilo Lines of Code*)

KTA: Comisión de Estándares en Seguridad Nuclear de Alemania (*Kerntechnischer Ausschuss*)

L

LAN: Red de Área Local (*Local Area Network*)

LCU: Unidad de Control de Lanzamiento (*Launch Control Unit*)

LD: Dominio Limitado (*Limited Domain*)

LD: Diagramas *Ladder* (*Ladder Diagrams*)

LEU: Unidad Electrónica Lateral (*Line-side Electronic Unit*)

LLR: Requerimiento de Bajo Nivel (*Low Level Requirement*)

LOCS: Líneas de Código Fuente en Servicio (*Lines of Code in Service*)

LOR: Nivel de rigor (*Level of Rigor*)

LOT: Nivel de Confianza (*Level of Trust*)

LRU: Unidades de Línea Reemplazables (*Line-Replaceable Units*)

LSA: Análisis de Soporte Logístico (*Logistic Support Analysis*)

LTD: Diseño de Pruebas de Nivel (*Level Test Design*)

LTR: Reporte de Pruebas de Nivel (*Level Test Report*)

LUCOL: Lenguaje de Control de Lucas Aerospace (*LUcas Control Language*)

LVL: Lenguaje de variabilidad limitada (*Limited Variability Language*)

M

MA: Análisis de Markov (*Markov Analysis*)

MC/DC: Condición Modificada/Cobertura de Decisión (*Modified Condition/Decision Coverage)*

MCS: Sistema de Control de la Misión (*Mission Control System*)

MER: Vehículo de Exploración de Marte (*Mars Exploration Rover*)

MET: Tiempo Transcurrido de la Misión (*Mission Elapsed Time*)

MFD: Pantallas Multifunción (*Multi-function Display*)

MIL-STD: Estándares militares de los Estados Unidos (*United States Military Standards*)

MISRA: Asociación para la Confiabilidad del *Software* en la Industria del Motor (*Motor Industry Software Reliability Association*)

MIT: Instituto Tecnológico de Massachusetts (*Massachusetts Institute of Technology*)

MMA: Aeronave Marítima de Misión Múltiple (*Multi-mission Maritime Aircraft*)

MMU: Unidad de Gestión de Memoria (*Memory Management Unit*)

MoD: Ministerio de Defensa del Reino Unido (*Ministery of Defense*)

MPCV: Vehículo de Traslado Multipropósito (*Multi-purpose Crew Vehicle*)

MRTT: Avión de Transporte y Reabastecimiento Aéreo Multipropósito (*Multi Role Tanker Transport*)

MSAW: Advertencia de Altitud Mínima de Seguridad (*Minimum Safe Altitude Warning*)

MSS: Sistema de Servicio Móvil (*Mobile Servicing System*)

MUF: Foro de Usuarios de *MASCOT* (*MASCOT User's Forum*)

MWF: Funciones que deben funcionar (*Must Work Functions*)

N

NAS: Sistema Nacional del Espacio Aéreo (*National Airspace System*)

NASA: Administración Nacional de la Aeronáutica y el Espacio (*National Aeronautics and Space Administration*)

NASS: Subsistema de Supervisión y Gestión activa de la red de Tráfico en el Reino Unido (*Network Active Traffic Management Supervisory Subsystem*)

NATO: Organización del Tratado del Atlántico Norte (*North Atlantic Treaty Organization*)

NATS: Servicios Nacionales de Tráfico Aéreo del Reino Unido (*National Air Traffic Services*)

NAVSEA: Comando Naval de Sistemas Marítimos de los Estados Unidos (*Naval Sea Systems Command*)

NIRS: Instituto Nacional de Ciencias Radiológicas (*National Institute of Radiological Sciences*)

NIST SP: Procedimientos Estándares del Instituto Nacional de Estándares y Tecnología de los Estados Unidos (*National Institute of Standards and Technology - Standard Procedures*)

NITRD: Oficina de Coordinación Nacional de los Estados Unidos para la Investigación y Desarrollo de las Redes y las Tecnologías de la Información (*U.S. National Coordination Office for Networking and Information Technology Research and Development*)

NMR: Redundancia Modular Múltiple (*N-Folder Modular Redundancy*)

NOAA: Administración Nacional Oceánica y Atmosférica *(National Oceanic and Amospheric Administration)*

NQA: Aseguramiento de la Calidad Nuclear (*Nuclear Quality Assurance*)

NSAM: Estándares de la Armada de los Estados Unidos

NDS: Diagramas de Nassi-Shneiderman (*Nassi-Shneiderman Diagram*)

NSI: Sin impacto para la seguridad (*No Safety Impact*)

NSWC: Centro de Guerra Naval de Superficie (*Naval Surface Warfare Center*)

NUREG: Regulaciones Nucleares en Estados Unidos (*Nuclear Regulatory*)

O

OBD: Diagnóstico de a bordo (*Onboard diagnostic*)

OBS: Sistema de a bordo (*On-board system*)

OCS: Sistema de Control de a Bordo (*Onboard Control System*)

OEE: Eficiencia Total del Equipamiento (*Overall Equipment Efficiency*)

OHA: Análisis de Peligros de Operaciones (*Operating Hazard Analysis*)

OI: Incremento Operacional (*Operational Increment*)

OMT: Técnica de Modelado de Objetos (*Object Modeling Technique*)

OOA: Análisis Orientado a Objetos (*Object Oriented Analysis*)

OOADA: Análisis y Diseño orientados a objetos con Aplicaciones (*Object-Oriented Analysis and Design with Applications*)

OOD: Diseño Orientado a Objetos (*Object Oriented Design*)

OOSE: Ingeniería de *Software* orientada a objetos (*Object-Oriented Software Engineering*)

OSS: *Software* de Código Abierto (*Open Source Software*)

P

PADT: Herramienta de Programación y Depuración (*Programming And Debugging Tool*)

PASA: Evaluación Preliminar de la Seguridad de la Aeronave (*Preliminary Aircraft Safety Assessment*)

PASS: *Software* Primario de los Sistemas de Aviónica (*Primary Avionics System Software*)

PEO: Asociación de Ingenieros de Ontario, Canadá (*Professional Engineers of Ontario*)

PES: Sistema Electrónico Programable (*Programmable Electronic System*)

PFD: Pantalla Principal de Vuelo (*Primary Flight Display*)

PGA: Matriz de Puertas Programable (*Programmable Gate Array*)

PHA: Análisis Preliminar de Peligros (*Preliminary Hazard Analysis*)

PL: Niveles de Rendimiento (*Perfomance Level*)

PLC: Controlador Lógico Programable (*Programmable Logic Controller*)

PL/M: Lenguaje de Programación para Microcomputadoras (*Programming Language for Microcomputers*)

PoC: Demostración del Concepto (*Proof of Concept*)

POL: Lenguaje Orientado a Problemas (*Problem Oriented Language*)

PRA: Análisis de Riesgos Particulares (*Particular Risks Analysis*)

PSA: Analizador de Enunciado de Problemas (*Problem Statemente Analyzer*)

PSL: Lenguaje de Enunciado de Problemas (*Problem Statement Language*)

PSS: Especificaciones de Procedimientos y Estándares de la Agencia Espacial Europea (*Procedures Specifications and Standards - European Space Agency*)

PVS: Sistema de Verificación de Prototipo (*Prototype Verification System*)

Q

QM: Gestión de Calidad (*Quality Management*)

R

RAD: Desarrollo Rápido de Aplicaciones (*Rapid Application Development*)

RAMS: Confianza, Disponibilidad, Facilidad de Mantenimiento y Seguridad (*Reliability, Availability, Maintainability and Safety)*

RDT&E: Investigación, Desarrollo, Pruebas y Evaluación (*Research, Development, Testing and Evaluation*)

RFT: Tolerancia a Fallos Redundante (*Redundant Fault Tolerant*)

RID: Discrepancias en la Revisión de Ítems (*Review Item Discrepancy*)

RIMS: Estaciones de Referencia y Supervisión de Integridad (*Ranging and Integrity Monitoring Stations*)

RISC: Computador con Conjunto de Instrucciones Reducidas (*Reduced Instruction Set Computer*)

RMA: Análisis de Tasa Monotónica (*Rate Monotonic Analysis*)

ROI: Retorno de la Inversión (*Return On Investment*)

ROM: Memoria de sólo lectura (*Read Only Memory*)

RPS: Sistema de Protección del Reactor (*Reactor Protection System*)

RTCA: Comisión Radiotécnica para la Aeronáutica (*Radio Technical Commission for Aeronautics*)

RTE: Gestionador del Tiempo de Ejecución (*Run-Time Executive*)

RTOS: Sistema Operativo en Tiempo Real (*Real-Time Operating System*)

RTTI: Información de Tipos en Tiempo de Ejecución (*Run-Time Type Information*)

S

SAD: Dispositivo de seguro de armas (*Safe-Arm Device*)

SADT: Técnicas de Análisis y Diseño de Sistemas (*System Analysis and Design Techniques*)

SAE: Sociedad de Ingenieros de Automoción (*Society of Automotive Engineers*)

SAFECOMP: Conferencia Internacional sobre la Seguridad, Confiabilidad y Protección de las Computadoras (*International Conference on Computer Safety, Reliability and Security)*

SAIC: Corporación Internacional para Aplicaciones de la Ciencia (*Science Applications International Corporation*)

SAS: Resumen de Cumplimientos del *Software* (*Software Accomplishment Summary*)

SBC: Ordenador de placa reducida (*Single Board Computer*)

SCA: Área de Seguridad y Control (*Safety and Control Area*)

SCA: Análisis de Circuitos (*Sneak Circuit Analysis*)

SCADE: Ambiente de Desarrollo para Aplicaciones Críticas para la Seguridad (*Safety Critical Application Development Environment*)

SCI: Índice de la Configuración del *Software* (*Software Configuration Index*)

SCM: Gestión de Configuración del *Software* (*Software Configuration Management*)

SCMP: Plan de la Gestión de Configuración del *Software* (*Software Configuration Management Plan*)

SCR: Reducción de Costos del *Software* (*Software Cost Reduction*)

SCR: Revisión de Conformidad del *Software* (*Software Conformity Review*)

SCR: Revisión del Concepto del *Software* (*Software Concept Review*)

SCR: Requerimiento de Cambios en el *Software* (*Software Changes Request*)

SCS: Sistema de Control de Estación (*Station Control System*)

SDI: Dirección Segura (*Safe Direction*)

SDLC: Ciclo de Vida del Desarrollo del *Software* (*Software Development Life Cycle*)

SDP: Plan de Desarrollo del Software (*Software Development Plan*)

SDS: Especificación del Diseño del *Software* (*Software Design Specification*)

SDV: Verificación Sistemática del Diseño (*Systematic Design Verification*)

SECI: Índice con la Configuración del ambiente del ciclo de vida del *software* (*Software Life Cycle Environment Configuración Index*)

SEE: Entorno de Ingeniería del *Software* (*Software Engineering Environment*)

SEM: Metodología de Ingeniería de Sistemas (*Systems Engineering Methodology*)

SFTA: Análisis de Árbol de Fallos del *Software* (*Software Fault Tree Analysis*)

SIDFA: Sistema de Investigación y Desarrollo de la FAA

SIF: Función instrumentada de seguridad (Safety Instrumented Function)

SIL: Nivel de Integridad de la Seguridad (*Safety Integrity Level*)

SIS: Sistema instrumentado de seguridad (*Safety Instrumented System*)

SLP: Posición Limitada de Seguridad (*Safely Limited Position*)

SLS: Velocidad limitada de seguridad (*Safely Limited Speed*)

SM: Movilidad en la superficie (*Surface Mobility*)

SNMP: Protocolo Simple de Administración de Red (*Simple Network Management Protocol*)

SO: Operaciones en la superficie (*Surface Operations*)

SOS: Parada de Operación Segura (*Safe Operating Stop*)

SOUP: Software de procedencia desconocida (*Software of unknown provenance*)

SOW: Declaración de Trabajo (*Statement of Work*)

SP: Posición Segura (*Safe Position*)

SPARC: Arquitectura de procesador escalable (Scalable Processor Architecture)

SPR: Reporte de Problemas en el Software (*Software Problem Report*)

SPH: Manual de Estándares y Procedimientos (*Standards and Procedures Handbook*)

SPMP: Plan de Gestión del Proyecto de *Software* (*Software Project Management Plan*)

SQA: Aseguramiento de la Calidad del *Software* (*Software Quality Assurance*)

SQAP: Plan de Aseguramiento de la Calidad del *Software* (*Software Quality Assurance Plan*)

SQAR: Registros de Aseguramiento de la Calidad del *Software* (*Software Quality Assurance Records*)

SRASW: *Software* de Aplicación Relacionado con la Seguridad (*Safety-Related Application Software*)

SREM: Metodología de Ingeniería de Requerimientos del Sistema (*System Requirements Engineering Methodology*)

SS: Sensor Solar (*Sun Sensor*)

SS1: Parada de seguridad Tipo 1 (*Safe Stop 1*)

SS2: Parada de seguridad Tipo 2 (*Safe Stop 2*)

SSC: Estructuras, Sistemas y Componentes (*Structures, Systems and Components*)

SSDS: Sistema de Auto Defensa de Buques (*Ship Self-Defense System*)

SSH: Manual de Seguridad del Sistema (*System Safety Handbook*)

SSH: Intérprete de Órdenes Seguro (*Secure SHell*)

SSHA: Análisis de Peligros de los Subsistemas (*Subsystem Hazard Analysis*)

SSL: Capa de Conexión Segura (*Secure Socket Layer*)

SSM: Control de velocidad segura (*Safe Speed Monitor*)

SSP: Plan de Seguridad del *Software* (*Software Safety Plan*)

SSR: Grabadora de estado sólido (*Solid-State Recorder*)

SSR: Requerimiento de Seguridad del Sistema (*System Safety Requirement*)

ST: Rastreador Estelar (*Star Tracker*)

ST: Lenguaje de Texto Estructurado (*Structured Text*)

STANAG: Acuerdos de Estandarización de la Organización del Tratado del Atlántico Norte (*Standardization Agreement - North Atlantic Treaty Organization*)

STD: Documento de Transferencia del *Software* (*Software Transfer Document*)

STE: Ambiente de Pruebas del *Software* (*Software Test Environment*)

STO: Desconexión segura del par de fuerza motriz (*Safe Torque Off*)

SUM: Manual del Usuario del Software (*Software User's Manual*)

SwSIL: Nivel de Integridad de Seguridad del *Software* (*Software Safety Integrity Level*)

SWSPP: Plan para el Programa de Seguridad del *Software* (*Software Safety Program Plan*)

T

TARDIS: Pantalla del Sistema Avanzado de Información de Radar del avión de combate Panavia Tornado (*Tornado Advanced Radar Display Information System*)

TCAS: Sistema de Alerta de Tráfico y Evasión de Colisión (*Traffic Collision Avoidance System*)

TCP/IP: Protocolo de Control de Transmisión / Protocolo de Internet (*Transmission Control Protocol / Internet Protocol*)

TMR: Redundancia Modular Triple (*Triple-Modular Redundant*)

TPA: Analizador Térmico de la Central Nuclear (*Thermal Plant Analyzer*)

TÜV: Asociación de Inspección Técnica de Alemania (*Technischer Überwachungs-Verein*)

U

UAV: Vehículo Aéreo no Tripulado (*Unmanned Aerial Vehicle*)

UCAV: Vehiculo Aéreo de Combate no Tripulado (*Unmanned Combat Air Vehicle*)

UML: Lenguaje de Modelado Unificado (*Unified Modeling Language*)

URET: Herramienta para la Evaluación de Requerimientos del Usuario (*User Request Evaluation Tool*)

URL: Localizador de Recursos Uniformes (*Uniform Resource Locator*)

USAF: Fuerza Aérea de los Estados Unidos (*United States Air Force*)

USN: Marina de los Estados Unidos (*United States Navy*)

USNRC: Comisión Reguladora Nuclear de los Estados Unidos (*United States Nuclear Regulatory Commission*)

UTC: Tiempo Universal Coordinado (*Universal Time Coordinated - Temps Universel Coordonné*)

V

V&V: Verificación y Validación (*Verification and Validation*)

VAL: Vehículo Automático Ligero (*Véhicule Automatique Léger*)

VLS : Sistema de Lanzamiento Vertical (*Vertical Launching System*)

W

WCET : Peor Caso del Tiempo de Ejecución (*Worst-Case Execution Time*)

X

XML : Lenguaje de Marcas Extensible (*eXtensible Markup Language*)

Apéndice B. Palabras clave

Software, seguridad, criticidad, Infraestructura, Medicina, Energía Nuclear, Transporte, Ferrocarriles, Automóviles, Aviación, Espacio, Defensa, Industria, Estándares, Metodologías, *DO-178B, DO-178C*

Referencias bibliográficas

Capítulo 3

1. John C. Knight (2002) *"Safety Critical Systems: Challenges and Directions"*, *Department of Computer Science, University of Virginia*

2. *President's Information Technology Advisory Committee* (1999) *"Report to the President - Information Technology Research: Investing in Our Future", USA*

3. J. C. Laprie (1992) *"Dependability: Basic Concepts and Terminology", Springer-Verlag, Vienna*

4. *International Electrotechnical Commision* (1990) *"IEC 60550-191 International Electrotechnical Vocabulary - Dependability and quality of service"*

5. J. C. Laprie, op. cit.

6. Dr. Andrew J. Kornecki (2014) *"Safety Critical Software Development for Complex Systems Methods and Tools", Department of Electrical, Computer, Software and System Engineering, Embry Riddle Aeronautical University*

7. Dr. Andrew J. Kornecki, op. cit.

8. Avizienis, A., Laprie, J. C., Randell, B. y Landwehr, C. (2004) *"Basic concepts and taxonomy of dependable and secure computing", Vytautas Magnus University*, Kaunas, Lituania

9. Dr. Andrew J. Kornecki, op. cit.

10. Dr. Andrew J. Kornecki, op. cit.

11. Avizienis, A. et al., op. cit.

12. Dr. Andrew J. Kornecki, op. cit.

13. Avizienis, A. et al., op. cit

14. Dr. Andrew J. Kornecki, op. cit.

15. Avizienis, A. et al., op. cit

16. Avizienis, A. et al., op. cit

17. Donald G. Firesmith (2004), "*Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements*", *Journal of Object Technology*, vol. 3, nro. 3

18. Philip Koopman (2014) "*Critical Systems and Software Safety - Distributed Embedded Systems*", *Carnegie Mellon University*

19. Dr. Michael F. Siok (2013) "*Software Safety - Process Overview and Application*", *Lockheed Martin Aeronautics Company*

20. *U. S. Public Law 107-296* (2002) *U.S. Government Printing Office, 107th Congress, USA*

21. Jacob Olcott (2012) "*Confronting Cyber Risk in Critical Infrastructure - The National and Economic Benefits of Security Development Processes*", *Good Harbor Consulting LLC*

22. Daniel Jackson, Martyn Thomas y Lynette I. Millett (2007) "*Software for Dependable Systems: Sufficient Evidence?*", *Committee on Certifiably Dependable Software Systems, National Research Council*

23. *Committee on Technology National Science and Technology Council* (2007) "*Suplement to the President's Budget for Fiscal Year 2008", The Networking and Information Technology Research and Development Program*

24. Daniel Jackson et al., op. cit.

25. Daniel Jackson et al., op. cit.

26. *Board on Population Health and Public Health Practice* (2011) "*Public Health Effectiveness of the FDA 510(k) Clearance Process - Measuring Post-market Performance and Other Select Topics", Institute of Medicine of the National Academies, The National Academies Press, Washington, USA*

27. John Fox (2001) "*Designing Safety into Medical Decisions and Clinical Processes*", *Computer Safety, Reliability and Security* - Vigésima Conferencia Internacional - Budapest, Hungría, *SAFECOMP*

28. Richard C. Fries (1997) "*Reliable Design of Medical Devices*", *Marcel Dekker Inc., New York, USA*

29. Richard C. Fries, op. cit.

30. *Parasoft Corporation* (2011) "*The challenge: reducing the burden of complying with Pharmaceutical Industry Standards*", *Monrovia, California*

31. *Parasoft Corporation* (2014), "*FDA Software validation beyond Static Analysis*", *Monrovia, California, USA*

32. *Critical Software* (2014) "*Dependable Technologies for Critical Systems: better data management, better patient outcomes*", *Coimbra, California*

33. *Food and Drug Administration* (2002) "*General Principles of Software Validation; Final Guidance for Industry and FDA Staff*", *U.S. Department Of Health and Human Services*

34. *Unit B2 - Health Technology and Cosmetics* (2012) "*Medical Devices: Guidance document - Qualification and Classification of stand- alone software*", *European Commission DG Health and Consumer Directorate B*

35. *Unit B2 - Health Technology and Cosmetics*, op. cit.

36. Dr. Paul Anderson (2015) "*Embedded Medical Device Software—How and Why To Use Static Analysis To Update Legacy Code*", *Embedded Systems Engineering, GrammaTech, USA*

37. *Food and Drug Administration* (2013) "*Medical Device Recall Report FY2003 to FY2012*", *Center for Devices and Radiological Health - Office of Compliance - Division of Analysis and Program Operations*

38. *Food and Drug Administration*, op. cit.

39. *Board on Health Care Services* (2012) "*Health IT and Patient Safety - Building Safer Systems for Better Care*", *Committee on Patient Safety and Health Information Technology, The National Academies Press, Washington*

40. Barry Barber y Ray Rogers (2003) "*Safety in health information systems*", *Safety-Critical Systems Club - Newsletter Volume 12 Number 2*

41. Consejo de Seguridad Nuclear (2014) "*Licensing of safety critical software for nuclear reactors - Common position of international nuclear regulators and authorized technical support organizations*"

42. *Atomic Energy of Canada Ltd. and Ontario Power Generation, Inc.* (1999) "*STANDARD CE-1001-STD Revision 2 Standard for Software Engineering of Safety Critical Software*", *CANDU® Computer Systems Engineering Centre of Excellence*

43. Warren L. Persons y J. Dennis Lawrence (1993) IEEE Estándar 379 en "*Class 1E Software Verification and Validation: Past, Present, and Future*", *Lawrence Livermore National Laboratory, California, USA*

44. Warren L. Persons y J. Dennis Lawrence, op. cit.

45. Nicolas Sannier y Benoit Baudry (2012) "*Defining and Retrieving Themes in Nuclear Regulations*", *Fifth International Workshop on Requirements Engineering and Law (RELAW)*

46. *National Research Council* (1997) "*Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues*", *Committee on Application of Digital Instrumentation and Control Systems to Nuclear Power Plant Operations and Safety, USA*

47. *National Research Council,* op. cit.

48. *National Research Council,* op. cit.

49. *National Research Council,* op. cit.

50. *Japan Nuclear Energy Safety Organization* (2007) "*Digital Instrumentation and Control Systems for Safety System and Main Control Room Design in Japan Nuclear Power Station*"

51. Mark Lawford y Alan Wassyng (2012) "*Formal Verification of Nuclear Systems*", *Information & Security - An International Journal - Vol. 28 - No. 2*

52. *International Atomic Energy Agency* (2009) "*Implementing Digital Instrumentation and Control Systems in the Modernization of Nuclear Power Plants*", *IAEA Nuclear Energy Series No. NP-T-1.4, Vienna*

53. Consejo de Seguridad Nuclear, op. cit.

54. Jonathan F. Luedeke (1995) "*Safety of High Speed Ground Transportation Systems - Analytical Methodology for Safety Validation of Computer Controlled Subsystems - Volume II: Development of a Safety Validation Methodology", U.S. Department of Transportation, Federal Railroad Administration*

55. Alexei Iliasov y Alexander Romanovsky (2015), "*Formal Modelling of Railway Safety and Capacity*", *Safety-Critical Systems Club*

56. B. Ning (2010) "*Advanced Train Control Systems- CTCS Chinese Train Control System*", *Beijing Jiaotong University, China*

57. *IEEE Rail Transit Vehicle Interface Standards Committee of the IEEE Vehicular Technology Society* (1999) "*IEEE Standard for CBTC Performance and Functional Requirements"*

58. N. Bin, T. Tao, Q. K. Min y G. C. Hai (2010), "*Advanced Train Control Systems - CBTC (Communication Based Train Control): system and development*", *Department of Control Engineering, School of Electronics and Information Engineering, Beijing Jiaotong University, China*

59. N. Bin et al., op. cit.

60. Andrew Ryan (2010) "*Formal Specification of Moving Block Railway Interlocking using CASL*", *Department of Computer Science, University of Wales Swansea*

61. M. Antoni y N. Ammad (2008) "*Formal validation method and tools for French computerized railway interlocking systems*", *SNFC*, Francia

62. L. Lindqvist y R. Jadhav (2010) "*Advanced Train Control Systems - Application of communication based Moving Block systems on existing metro lines*", *Centre of Excellence, Bombardier Transportation, Spain and Bombardier Transportation, USA*

63. T. Matsuki, T. Okamiya, S. Hondo y H. Tsuruga (2010) "*Advanced Train Control Systems - Fully digitalized ATC (Automatic Train Control) system of integrated functions of train-protection and interlocking*", *East Japan Railway Company and Hitachi Ltd., Japan*

64. N. Bin et al., op. cit.

65. N. Bin et al., op. cit.

66. C. Braban y P. Charon (2010) "*Advanced Train Control Systems - Re-signaling the Paris Line 1: from driver-based to driverless operation*", *Siemens Transportation Systems and Paris Urban Transport Operator, France*

67. W. A. M. Barter (2010) "*Advanced Train Control Systems - ERTMS Level 2: effect on capacity compared with "best practice" conventional signalling*", *First Class Partnerships Ltd, UK*

68. W. A. M. Barter, op. cit.

69. Enclavamientos y Señalización Ferroviaria S.A. (2010) "Sistemas de Seguridad - Enclavamiento Electrónico", ENYSE - Madrid

70. *Railtrack PLC* (1998) "*Signalling and Operational Telecommunications Design: Safety Requirements*", *Safety & Standards Directorate - Railtrack House - Euston Square, London*

71. Barbara J. Czerny, Ph.D.; Joseph G. D'Ambrosio, Ph.D., Paravila O. Jacob, Ph.D., Brian T. Murray, Ph.D.; Padma Sundaram (2003) "*A software safety process for safety-critical advanced automotive systems", Delphi, Corp., Proceedings of the 21st International System Safety Conference - Ottawa - Canada*

72. Barbara J. Czerny et al., op. cit.

73. Nicolas Navet y Francoise Simonot-Lion (2009) "*Automotive Embedded Systems Handbook*", *CRC Press*

74. Dave Higham (2013) "*An Introduction to ISO 26262 Functional Safety of Road Vehicles*", *Safety-Critical Systems Club - Newsletter Volume 22 Number 2*

75. Nicolas Navet y Francoise Simonot-Lion, op. cit.

76. Nicolas Navet y Francoise Simonot-Lion, op. cit.

77. Nicolas Navet y Francoise Simonot-Lion, op. cit.

78. Nicolas Navet y Francoise Simonot-Lion, op. cit.

79. Nicolas Navet y Francoise Simonot-Lion, op. cit.

80. Nicolas Navet y Francoise Simonot-Lion, op. cit.

81. Nicolas Navet y Francoise Simonot-Lion, op. cit.

82. Nicolas Navet y Francoise Simonot-Lion, op. cit.

83. Masaru Kurihara (2010) "*Automotive ECU software development with SCADE Suite*", *Electronics Engineering Department - Fuji Heavy Industries, Embedded News*

84. *GrammaTech* (2011) "*Simplifying ISO 26262 Compliance with GrammaTech Static Analysis Tools*", *Technical White Paper*

85. William S. Greenwell (2003) "*Learning Lessons from Accidents and Incidents Involving Safety-Critical Software Systems*", *Faculty of the School of Engineering and Applied Science - University of Virginia*

86. *Federal Aviation Administration* (2007) "*Assesment of Software Development Tools for Safety-Critical, Real-Time Systems*", *U.S. Department of Transportation*

87. *SESM Finmeccanica Company* (2011) "*Toward a New ATM Software Safety & Security Assessment Methodology*"

88. Ian Moir y Allan Seabridge (2001) *"Aircraft Systems - Mechanical, Electrical and Avionics Subsystems Integration", Professional Engineering Publishing Limited, London and Bury St Edmunds, UK*

89. William S. Greenwell, op. cit.

90. *Federal Aviation Administration* (2000) *"FAA System Safety Handbook, Chapter 1: Introduction"*

91. *Committee on Science - House of Representatives* (1999) *"FAA needs to improve controls over use of foreign nationals to remediate and review software", Computer Security - Report of the Chairman, General Accounting Office*

92. William S. Greenwell, op. cit.

93. Rogelio Palomo Pinto (2007), "Aviónica y Sistemas de Navegación", Universidad de Sevilla

94. http://en.wikipedia.org/wiki/Electronic_flight_instrument_system

95. http://en.wikipedia.org/wiki/Electronic_flight_instrument_system

96. http://en.wikipedia.org/wiki/Electronic_flight_instrument_system

97. División de Ingeniería Aeronáutica del Instituto Tecnológico de Aeronáutica (ITA) de Brasil (s.f.), http://www.aer.ita.br/~bmattos/mundo/forward/eicas.html

98. Ian Moir y Allan Seabridge, op. cit.

99. *National Research Council* (1993) *"An Assessment of Space Shuttle Flight Software Development Processes", Committee for Review of Oversight Mechanisms for Space Shuttle Flight Software Processes, Aeronautics and Space Engineering Board, Washington D.C.*

100. Robert Dewar (2002) "*Ada Core - Case Study - MDA - Canadian Space Arm*", *Ada Core Technologies, COTS Journal*

101. *National Research Council,* op. cit.

102. R. H. Pierce, Sandra Ayache, R. Ward, J. Stevens, Helen Clifton, J. Galle (1997) *"Capturing and Verifying Performance Requirements for Hard Real Time Systems", 1997 Ada-Europe International Conference on Reliable Software Technologies, Springer, London*

103. Jens Eickhoff (2012), *"Onboard Computers, Onboard Software and Satellite Operations: An Introduction", Springer-Verlag,* Berlín

104. Ljerka Beus-Dukic (2001) "*Safety-Critical Systems Club - Newsletter - COTS real-time operating systems*", *Volume 10, Number 3, UK*

105. Malcolm Macdonald y Viorel Badescu (2014), *"The International Handbook of Space Technology", Springer-Verlag,* Berlín

106. Malcolm Macdonald y Viorel Badescu, op. cit.

107. Malcolm Macdonald y Viorel Badescu, op. cit.

108. Daniel L. Dvorak (2009) "*NASA Study on Flight Software Complexity*", *Systems and Software Division - Jet Propulsion Laboratory - California Institute of Technology, USA*

109. *NASA/JPL/Cornell University (s.f.), Maas Digital LLC,* Dominio Público

110. *National Research Council,* op. cit.

111. Daniel L. Dvorak, op. cit.

112. Daniel L. Dvorak, op. cit.

113. Daniel L. Dvorak, op. cit.

114. Malcolm Macdonald y Viorel Badescu, op. cit.

115. Malcolm Macdonald y Viorel Badescu, op. cit.

116. Ljerka Beus-Dukic, op. cit.

117. Ljerka Beus-Dukic, op. cit.

118. William Scherlis, Enita Williams y Jon Eisenberg (2011), "*Critical Code: Software Producibility for Defense*", *National Research Council*, *Computer Science and Telecommunications Board (CSTB*)

119. *PR Newswire* (2006) "*Boeing Selects Wind River Workbench as the Device Software Development Tool Suite for the U.S. Army's Future Combat System (FCS) Program*", *USA*

120. William Scherlis et al., op. cit.

121. Rob Shenk (2008) "*Joint Services Open House airshow", Andrews AFB, USA*

122. Capers Jones (2002) "*Defense Software Development in Evolution", Crosstalk-The Journal of Defense Software Engineering, USA*

123. *Department of Defense* (1990) "*System Safety Engineering - Design Guide for Army materiel*", *USA*

124. William Farr (2003) "*Current and Future Challenges of Software Reliability Assessment*", *U.S. Army Conference on Applied Statistics*

125. *Department of Defense* (1998) "*DEF(AUST) 5679 The Procurement of Computer-Based Safety Critical Systems", Australian Defence Standard*

126. *NATO* (1997) "*NATO STANAG 4404 NATO Standardization Agreement - Safety Design Requirements and Guidelines*"

127. Siddhartha R. Dalal, Jesse H. Poore y Michael L. Cohen (2003) "*Innovations in Software Engineering for Defense Systems*", *Oversight Committee for the Workshop on Statistical Methods in Software Engineering for Defense Systems, Committee on Applied and Theoretical Statistics, National Research Council, USA*

128. *U.S. Navy* (s.f.) *USS John S. McCain (DDG-56)*

129. www.defenseimagery.mil (2013)

130. Michael J. Sullivan (2012) "*Joint Strike Fighter - Restructuring Added Resources and Reduced Risk, but Concurrency Is Still a Major Concern*", *Testimony Before the Subcommittee on Tactical Air and Land Forces, Committee on Armed Services, House of Representatives, United States Government Accountability Office*

131. Joan D. Winston y Lynette I. Millett (2007) "*Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale*", *Committee on Advancing Software-Intensive Systems Producibility, National Research Council, USA*

132. Joan D. Winston y Lynette I. Millett, op. cit.

133. Joan D. Winston y Lynette I. Millett, op. cit.

134. *Siemens AG* (2011) "*Functional safety: From necessity to competitive advantage*", *Safety Integrated for Factory Automation*

135. Josef Börcsök (2004) "*Safety Systems*", *HIMA Paul Hildebrandt GmbH + Co KG, Germany*

136. Robin Carver (2010) "*EN ISO 11161 - Safe design of Integrated Manufacturing Systems*", *UK*

137. Johan Hedberg, Andreas Söderberg y Jan Tegehall (2011) "*How to design safe machine control systems – a guideline to EN ISO 13849-1*", *SP Technical Research Institute of Sweden*

138. Timo Malm y Marita Hietikko (2011) "*Safety-critical software in machinery applications - The role of software in safety-critical systems*", *VTT Technical Research Centre of Finland*

139. *Siemens AG - Digital Factory* (2015) "*The Fast and Easy Way to Safe Machines – at Highest Productivity Exploiting the advantages of integrated safety technology*"

140. *Siemens AG - Digital Factory*, op. cit.

141. *Siemens AG - Digital Factory*, op. cit.

Capítulo 4

1. *Sixth Annual Conference on Software Assurance* (1991) *"Computer Software in civil aircraft", USA*

2. *AdaCore* (2014) "*What is DO-178B?*"

3. *Esterel Technologies SA* (2012) "*Efficient Development of Safe Avionics Display Software with DO-178B Objectives Using Esterel SCADE® - Third Edition (Revision 1)", Elancourt, France*

4. *Esterel Technologies SA*, op. cit.

5. Ben Brosgol y Cyrille Comar (2010) "*DO-178C: A New Standard for Software Safety Certification*", *SSTC 2010, Salt Lake City, Utah, USA*

6. Bill Stclair y Nat Hillary (2010) "*DO-178C: Improved certification for cost-effective avionics systems*", *Vita Technologies*

7. *Wind River Systems, Inc.* (2015) "*Wind River RTCA DO-178 Software Certification Services"*

8. Mehmet Kerim Çakmak (2013) "*Managing DO-178 compliance with IBM Rational Platform", Journal of KONBiN*, Ankara, Turquía

9. Vance Hilderman y Tony Baghi (2007) "*Avionics Certification: A Complete Guide to DO-178 (software), DO-254 (hardware)*", *Avionics Communications Inc., Leesburg, Virginia, USA*

10. Bernard Dion (2007) "*How to meet EUROCAE ED-12B / RTCA DO-178B International Software Safety Regulation for Airborne Systems while reducing development cost*", *International Conference New Challenges in Aeronautics, ASTEC '07*, Moscú

11. David Beberman y Joe Wlad (2010) "*C++ software development for DO-178 safety-critical applications*", *Military & Aerospace Electronics*

12. Mehmet Kerim Çakmak, op. cit.

13. Vance Hilderman y Tony Baghi, op. cit.

14. *Esterel Technologies SA* (2012) "*Methodology Handbook Efficient Development of Safe Avionics Software with DO-178B Objectives Using SCADE Suite® Fifth Edition (Revision 2)", Elancourt*, Francia

15. *GrammaTech Inc.* (2010) "*Simplifying DO-178B Certification with GrammaTech Static Analysis Tools*", *Ithaca, Nueva York, USA*

16. *Esterel Technologies SA*, op. cit.

17. Adam Trujillo, Cynthia Dunlop y Bisher Ahdab (2012) "*Developing DO-178B/C Compliant Software for Airborne Systems*", *Parasoft Corporation, Monrovia, California, USA*

18. *AdaCore* (2014) "*GNAT Pro Safety-Critical*"

Capítulo 5

1. *AdaCore* (2010) *"Barco selects GNAT Pro for Advanced Avionics Applications"*

2. *AdaCore* (2011) *"Rockwell Collins Selects GNAT Pro for Advanced Avionics Display"*

3. *ANSYS Advantage* (2013) *"Safe Landing Esterel solutions help Crane Aerospace & Electronics to design braking systems that are certified for safety", Volume VII, Issue 1*

4. *AdaCore* (2009) *"Thales Aerospace Division Selects GNAT Pro for Airbus A350 XWB Xtra Wide-Body*"

5. Grupo de Investigación de Defensa (s.f.), Portal Dintel GID

6. Revista Aeroespacio (2013) Nro. Noviembre

7. *AdaCore* (2011) *"Eurocopter Selects GNAT Pro for Military Helicopter ARINC 653 Project*"

8. *AdaCore* (2012) "*Embraer Selects Ada and AdaCore's GNAT Pro for AMX Upgrade*"

9. *AdaCore* (2011) "*Airbus Military Certifies to DO-178B level A Using GNATcheck*"

10. *AdaCore* (2006) "*Case Study BAE Systems Eurofighter Typhoon*"

11. *Lockheed Martin Corporation* (2005) "*Joint Strike Fighter Air Vehicle C++ Coding Standards for the System Development and Demonstration Program*"

12. *AdaCore* (2009) "*Lockheed Martin Selects GNAT Pro for C-130J Software - AdaCore's Ada development environment chosen for Block 7.0 Upgrade*"

13. K. J. Harrison (2003) "*Static Code Analysis on the C-130J Hercules Safety-Critical Software*", *Aerosystems International, UK*

14. Paul Parkinson (2004) "*Case Study: Development of the Tornado Advanced Radar/Map Display Information System (TARDIS)*"

15. *Wind River* (2005) "*BAE SYSTEMS Selects Wind River for Use in the Tornado Advanced Radar Display Information System-TARDIS*"

16. *Wind River* (2005) "*Wind River Device Software Optimization Technology Flying at the Heart of the Boeing KC-767A Tanker Aircraft's Integrated Modular Avionics Design*"

Capítulo 7

1. Sue Payton, J.C. Herz, Mark Lucas y John Scott (2006), "*Open Technology Development Roadmap Plan*", *Advanced Systems & Concept, Washington DC, USA*

2. http://mil-oss.org/ (s.f.) "*Mil-OSS Military Open Source Software - Utilizing and Developing Open Technologies for National Defense*"