Sistemas Digitales de Seguridad

El gran desarrollo de los sistemas automáticos de operación y control ha multiplicado el número de aplicaciones de seguridad en las que un fallo puede poner en peligro vidas humanas. El primer paso en el desarrollo de esos sistemas es la realización de un análisis de amenazas y determinación del riesgo, con el fin de definir el nivel de integridad de la aplicación. Este nivel influirá en todos los aspectos del proyecto, desde la elección de la arquitectura hardware a los métodos de desarrollo usados y el grado de pruebas que hay que realizar. En este artículo se presentan cuáles deben ser los pasos a seguir en el desarrollo de estos sistemas.

Introducción

Desde que aparecen en el mercado los microprocesadores, su precio ha ido descendiendo rápidamente, lo que ha motivado un sorprendente incremento de su uso en todos los campos. La mayor parte de estos micro-pro-cesadores no se utilizan como ordenadores personales, sino que están en sistemas "empotrados" donde la presencia del procesador es invisible al mundo exterior. Los sistemas de este estilo van desde complicados sistemas de control de vuelo hasta lavadoras, desde sistemas de apagado de una central a frenos antibloqueo. La mayor diferencia entre los sistemas empotrados y las aplicaciones de máquinas de propósito general se refiere a las consecuencias de una operación incorrecta. El fallo de algunos sistemas empotrados puede representar un peligro grave para las personas, el entorno o la economía de las empresas.

Existen diversas definiciones para el término seguridad ("*safety*")[1] . Utilizando la propuesta por (Storey, 1996): "Seguridad: propiedad de un sistema de que no causará daños humanos o pérdidas". La traducción del término "*safety-critical system*" podría ser sistema crítico de seguridad, o sistema de seguridad crítico. En cualquier caso hace referencia a un sistema en el que se garantiza la seguridad.

Aunque todos los estándares relativos a los sistemas de seguridad enfatizan la importancia de un proceso de desarrollo bien definido, es posible utilizar un proceso de desarrollo completo y consistente, pero inseguro. La seguridad trata de evitar los problemas conocidos, evitar la complejidad innecesaria y usar los principios de ingeniería simples y bien establecidos que se sabe que proporcionan resultados seguros. Por tanto, hay que tener en cuenta que disponer de un proceso de desarrollo certificado y bien definido no es suficiente para producir un sistema digital seguro, aunque indudablemente contribuye a ello.

Este artículo muestra en primer lugar cuáles son los criterios deseables en un sistema de seguridad. A continuación, se van a enumerar y explicar los pasos que se deben seguir al abordar el desarrollo de un sistema de estas características. Este artículo se centra en el desarrollo de sistemas digitales de seguridad, no en sistemas analógicos.

Finalmente, se presentarán las conclusiones de este trabajo.

Criterios de Seguridad

En los sistemas críticos de seguridad (Storey, 1996), la seguridad debe formar parte de los propios requisitos del sistema. Los criterios de seguridad deseables en un sistema (que deben quedar plasmados en los requisitos) son:

- Fiabilidad: probabilidad de un sistema de funcionar correctamente en un periodo de tiempo dado bajo unas condiciones de operación.
- Disponibilidad: probabilidad de que el sistema esté funcionando correctamente en un momento dado.

[1] En este punto, es necesario distinguir otro término que en castellano se traduce también por seguridad y que es el término inglés "*security*" (Douglas, 1999). Un sistema seguro ("*secure*") es un sistema inmune a los intentos, intencionados o no, de violar las barreras de seguridad utilizadas en un determinado lugar. A lo largo de este artículo, siempre que aparezca el término "seguridad" estará referido al término ingles "*safety*".

Yolanda González Arechavala
Licenciada en Informática por la Universidad del País Vasco. Es profesora en el Departamento de Sistemas Informáticos y colaboradora del Instituto de Investigación Tecnológica de la ETS de Ingeniería de la UPCo. Este trabajo ha sido parcialmente financiado con una beca de la Asociación de Ingenieros del ICAI.

Fernando de Cuadra García
Dr. Ingeniero Industrial del ICAI. Es Profesor Propio Ordinario de la ETSI ICAI de la UPCo, y actualmente director de esta escuela.

• Operación libre de fallo ("*Failsafe operation*"): cuando un sistema posee un conjunto de estados de escape que se pueden identificar como seguros –como la parada de un tren– el sistema puede designarse como "libre de fallo" asegurando que se adoptan dichos estados seguros en el caso de que se produzca un fallo.
• Integridad del sistema: habilidad de detectar fallos en su propia operación e informar al operador.
• Integridad de los datos: capacidad de un sistema de preservar los datos en sus bases de datos y detectar, y si es posible corregir, errores que hayan ocurrido.
• Recuperación del sistema: sobre todo en sistemas sin estados seguros, es de vital importancia recuperar el funcionamiento normal una vez que se haya detectado un fallo.
• "Mantenibilidad": es la facilidad del sistema para ser mantenido o devolver un sistema a sus condiciones de operación designadas.

La confianza ("*dependability*") como propiedad deseable de un sistema de seguridad va a cuantificarse con varios de estos factores como la fiabilidad y la disponibilidad. En muchos casos, los requisitos de seguridad entran en conflicto con otros requisitos del sistema, como puede ser el bajo coste o el tiempo de desarrollo.

Una máxima que se debe seguir en el desarrollo de un sistema, sea o no de seguridad, es que cuanto más sencillo sea, mucho mejor. Hay que tener en cuenta que la seguridad no se consigue simplemente con usar un conjunto de métodos de desarrollo adecuados. Debe existir una gestión de la seguridad que planifique, organice, monitorice y evalúe los aspectos de seguridad de un proyecto. Dada la importancia de la gestión de la seguridad es esencial que desde la cima de una empresa se instaure una cultura de seguridad y se defina una política de seguridad con el fin de establecer las prácticas adecuadas de trabajo en la organización.

Ocho pasos para construir un sistema de seguridad

Para conseguir un sistema seguro hay que utilizar un proceso de desarrollo centrado en la seguridad, además de las tareas de gestión de la seguridad asociadas. Centrándose únicamente en el proceso de desarrollo, (Douglas, 1999) propone ocho pasos para construir un sistema de seguridad:

1. Identificación de las amenazas

El primer paso del análisis de amenazas identifica las amenazas potenciales del sistema. Por supuesto, un funcionamiento normal del sistema no debe producir amenazas. Una vez que se han identificado las amenazas, es necesario identificar los defectos que provocan esas amenazas. En el análisis de amenazas se utiliza un conjunto de técnicas, cada una de las cuales pone de relieve distintas características del sistema que se está investigando. Algunas de las técnicas son:
• FTA ("*Fault-tree analysis*"), es el método comúnmente utilizado para analizar los defectos. Se parte del sistema en un estado no-seguro (con una amenaza) y se va buscando "hacia atrás" las causas que puedan haberlo causado. El FTA combina gráficamente las condiciones del defecto usando operadores booleanos. En la parte inferior de la figura aparecerán los defectos raíz y en la parte superior aparece la condición resultante no-segura. Por tanto, el FTA es una herramienta para ayudar a la identificación de los defectos raíz que conducen a la amenaza. Los dispositivos seguros no deberían producir una amenaza ante la presencia de un único fallo.
• ETA ("*Event tree analysis*"), toma como punto de partida los eventos que pueden afectar al sistema y hace un seguimiento de ellos para analizar sus posibles consecuencias. Estos eventos incluyen tanto aquéllos asociados con la operación esperada del sistema como los asociados con condiciones de fallo, de manera que gran parte de este análisis se realiza con operaciones que no tienen implicaciones de seguridad. El problema que tiene es que en cuanto el sistema aumenta un poco su complejidad los árboles que se obtienen son muy grandes.
• FMEA ("*Failure modes and effect analysis*"), es otra técnica utilizada en estos análisis, que al contrario que el FTA (que comienza con la amenaza e intenta identificar los defectos precursores de la misma), comienza con todos los componentes y sus modos de fallo y persigue los efectos de esos fallos hasta determinar sus últimas consecuencias en el sistema completo. El análisis puede realizarse tanto a nivel de componentes hardware como a nivel funcional usando un enfoque modular. Este método, al considerar todos los fallos de los componentes, es particularmente bueno en la detección de condiciones donde un fallo (no se suelen considerar múltiples fallos) puede producir una situación

peligrosa. Debido al alto coste que conlleva la realización del FMEA y a otros problemas, se suele utilizar al final del proceso de desarrollo, aplicándolo a áreas críticas en vez de a todo el sistema.

- FMECA ("*Failure modes, effects and criticality analysis*"), es una extensión del FMEA que tiene en cuenta la importancia de cada fallo de un componente según las consecuencias de los fallos particulares y la probabilidad de que ocurran, de manera que se identifican aquellas secciones del sistema donde los fallos son más importantes. De esta forma, se consigue dirigir los esfuerzos a las áreas con más necesidades.
- HAZOP ("*Hazard and Operability studies*"), usa una serie de palabras guía para investigar los efectos de las desviaciones del comportamiento normal durante cada fase de la operación del sistema. Se desarrolló en principio en la industria química, y aunque se ha comprobado que es muy eficiente para responder a preguntas de estilo "*what-if*", su aplicación en muy tediosa y necesita mucho tiempo.

2. Determinación del riesgo

El análisis de riesgo investiga las consecuencias de las amenazas y la probabilidad de que ocurran. Para ello, se define el nivel de severidad de la amenaza (consecuencias que puede tener) y la frecuencia relativa de la misma (probabilidad de que ocurra). Combinando la severidad y la frecuencia se obtiene la clasificación del riesgo asociado con una amenaza particular. A esta clasificación se la suele llamar clase de riesgo, o nivel de riesgo o factor de riesgo.

La mayor parte de los estándares define un número de clases de riesgo de manera que se establecen una serie de técnicas de desarrollo y diseño asociados a cada categoría. En el caso del estándar IEC 61508 (IEC, 2000), se definen cuatro clases de riesgo. La clase I se corresponde con el accidente más serio y la IV con el menos serio. En la tabla 1 se muestra la relación que existe entre las clases de riesgo y la severidad y frecuencia de la amenaza, pero el propio estándar reconoce que esta relación podría variar dependiendo del sector industrial donde se aplique.

La interpretación de las clases de riesgo se muestra en la tabla 2.

Conociendo el riesgo de cada amenaza, se podrá determinar cuáles pueden ser aceptables y cuáles no.

3. Definición de las medidas de seguridad.

Una vez que se ha determinado el riesgo para las distintas amenazas, es necesario determinar cuáles deben ser las medidas de seguridad encargadas de tratar aquellas amenazas de mayor riesgo.

La obtención de un sistema totalmente seguro es imposible: la meta es producir un sistema que sea suficientemente seguro, presentando los beneficios que tiene. Se trata de que sea tan seguro como sea razonable.

No es aceptable una amenaza que puede tener consecuencias catastróficas y que pueda ocurrir frecuentemente. Sin embargo, puede ser aceptable una situación en la que a menudo ocurra una amenaza insignificante o una situación en que la ocurrencia de un accidente catastrófico sea improbable o remota. La aceptabilidad de un nivel de riesgo viene determinada por los beneficios asociados a tolerar ese riesgo y por la cantidad de esfuerzo que requiera reducirlo.

El estándar IEC 61508 (IEC, 2000) divide los niveles de riesgo en tres bandas: región de riesgo inaceptable (clase I de riesgo), región de riesgo ALARP ("*as low as is reasonably practicable*", clases II y III de riesgo) y la región de riesgo aceptable donde el riesgo es tan pequeño que es insignificante (clase IV de riesgo). Por lo tanto, deben determinarse las medidas de seguridad necesarias para

Tabla 1: Clasificación del riesgo en IEC 61508

	Severidad			
Frecuencia	**Catastrófica**	**Crítica**	**Marginal**	**Insignificante**
Frecuente	I	I	I	II
Probable	I	I	II	III
Ocasional	I	II	III	III
Remoto	II	III	III	IV
Improbable	III	III	IV	IV
Increíble	IV	IV	IV	IV

Tabla 2: Interpretación de las clases de riesgo en el IEC 61508

Clases de Riesgo	Interpretación
I	Riesgo intolerable
II	Riesgo indeseable, y tolerable sólo si la reducción del riesgo es impracticable o si el coste es enormemente desproporcionado con la mejora obtenida
III	Riesgo tolerable si el coste de la reducción del riesgo podría exceder la mejora obtenida
IV	Riesgo insignificante

reducir los riesgos inaceptables y los riesgos ALARP que se consideren oportunos (estos riesgos son tolerables únicamente si su reducción es impracticable o si el coste de su reducción es tan grande que es desproporcionado respecto a la ganancia obtenida).

Al finalizar estos tres pasos, se crea un documento que contiene todas las amenazas identificadas y especifica el riesgo de las mismas. Utilizando el análisis de amenazas, se van a determinar los requisitos de seguridad del sistema, y periódicamente será necesario volver sobre el análisis de amenazas y riesgos para comprobar y en su caso incluir, nuevas amenazas que hayan podido añadir al sistema las sucesivas decisiones de diseño tomadas.

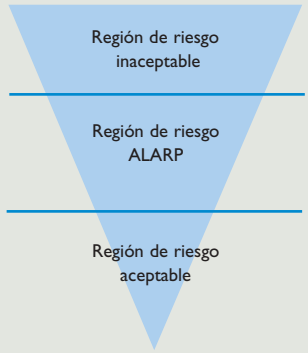
4. Creación de los requisitos de seguridad

Según el nivel de riesgo del sistema, van a ser necesarias más o menos medidas de seguridad, que van a implicar distintos requisitos de seguridad. Una vez conocidos los riesgos es necesario determinar el nivel de integridad del sistema o de algún componente. La asignación del nivel de integridad a un sistema, o parte de un sistema, se basa en la clasificación de los riesgos asociados a él. Pero riesgo y nivel de integridad son términos distintos. El riesgo es una medida de la probabilidad y de las consecuencias de una amenaza mientras que el nivel de integridad es una medida de la probabilidad de que un sistema de seguridad realice satisfactoriamente sus tareas de seguridad bajo cualquier condición en un periodo de tiempo.

En la IEC 61508 se definen cuatro niveles de integridad según el número máximo de veces que un sistema desarrollado para un cierto nivel puede esperarse que falle en un periodo de tiempo determinado (a mayor nivel de integridad menor probabilidad de fallo pueden tener).

El nivel de integridad de un sistema o subsistema impone un requisito relativo a la confianza en el sistema. Los métodos de desarrollo y el nivel de prueba que se deben usar para el desarrollo de ese sistema buscan aumentar la confianza del sistema hasta el nivel requerido. En los desarrollos de seguridad no sólo es necesario conseguir un alto nivel de integridad sino poder demostrar que se ha conseguido. Por lo tanto, el nivel de integridad determina cuál va a ser el proceso a seguir para obtener un desarrollo de calidad (empleando las técnicas impuestas por la ingeniería del software).

Figura 1: Regiones de riesgo del IEC 61508



La decisión de cuándo es práctica una reducción de un riesgo asociado a una aplicación particular es un tema de juicio profesional. En aplicaciones muy críticas la determinación de la aceptabilidad de un sistema dado lo determina la autoridad certificadora. En la figura 1 el triángulo invertido indica el costo de la reducción del riesgo. A mayor riesgo, mayor será el esfuerzo necesario para reducirlo ya que deben realizarse más tareas que aseguren que se llega a un nivel de seguridad aceptable.

Tabla 3: Tasas de fallo para los niveles de integridad del IEC 61508

Nivel de Integridad	Modo continuo de operación (probabilidad de fallo peligroso por año)	Modo de operación por petición (probabilidad de fallo para realizar la función designada por petición)
4	$\geq 10^{-5}$ to $< 10^{-4}$	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$	$\geq 10^{-2}$ to $< 10^{-1}$

5. Creación de diseños seguros

El diseño debe tener en cuenta la complejidad del problema fundamental y añadir las medidas de seguridad adicionales que identifiquen y traten posibles modos de fallo. Un argumento aplicable a todo diseño es que siempre hay que utilizar la tecnología más simple para conseguir las características de seguridad.

Una aproximación general a los diseños seguros es:

- Trabajar a partir de los requisitos de seguridad.
- Adoptar una arquitectura fundamental robusta y sencilla.
- Periódicamente, durante el diseño, volver al análisis de amenazas para añadir aquéllas producidas por fallos asociados al diseño específico adoptado.
- Aplicar medidas de programación en detalle ("*programming-in-the-small*") para proporcionar niveles adecuados de detección y corrección.
- Adoptar un conjunto consistente y apropiado de estrategias para detectar los defectos una vez que se han identificado.
- Desarrollar pruebas para el arranque, y pruebas de autocomprobación que se dispararán periódicamente en tiempo de ejecución, con el fin de identificar defectos latentes.
- Utilizar técnicas de tolerancia a fallos con el fin de realizar un diseño que consiga que los defectos no produzcan un fallo del sistema.

6. Implementación de la seguridad

Las reglas fundamentales de una programación segura son:

- Hazlo correcto, fácil de entender y legible (mejor que rápido).
- Verifica que tu programa continúa siendo correcto durante la ejecución del mismo. Utiliza la programación "a la defensiva" (comprobar explícitamente que todo lo que se está suponiendo es cierto).
- Los programadores que trabajan en un lenguaje conocido son más productivos y producen menos errores.

Hay una serie de requisitos genéricos para que un lenguaje de programación pueda ser utilizado en la implementación de un sistema de seguridad, como son:

- Es crucial que los programas sean predecibles antes de la ejecución: su funcionalidad, comportamiento temporal, uso de recursos y respuesta en caso de fallo.
- El diseño del programa y las construcciones lingüísticas empleadas deben estar orientadas a conseguir un programa correcto, validable y predecible.
- Debe ser fácil de entender ("*well-understood*"), de aprender, de usar además de ser fácil de implementar y de razonar.
- Debe proporcionar características apropiadas al dominio de aplicación.
- Debe ser posible verificar que un programa escrito con dicho lenguaje es correcto con respecto a la especificación expresada en notación formal.
- Disponer de compiladores y herramientas asociadas garantizados.

Los estándares de seguridad (CENELEC, 1998; IEC, 2000) definen ciertos lenguajes como prohibidos (como el C) y recomiendan otros (como Ada y en particular, un subconjunto seguro de Ada, el Spark Ada).

Según Rowe (Rowe, 1994) el lenguaje de programación más utilizado actualmente en seguridad es el Ada. En la industria ferroviaria se ha tomado dicho lenguaje como estándar de hecho. Otros lenguajes que también se utilizan son el ensamblador (el siguiente más utilizado), C++ y Modula-2. El estándar IEC 61508 recomienda el uso de Ada para los sistemas de nivel de integridad más alto.

7. Proceso para garantizar la seguridad

El proceso de desarrollo que se debe utilizar para los sistemas de seguridad debe seguir los principios básicos de la ingeniería del software, pero haciendo hincapié en volver continuamente al análisis de amenazas, realizar revisiones muy detalladas, verificar la consistencia entre diseño e implantación y la adherencia a los estándares de código, además de identificar cómo se trata cada amenaza.

El IEC 61508 ("*Functional safety of electrical/electronic/programmable electronic safety-related systems*") (IEC, 2000) es el estándar básico para los desarrollos de sistemas de seguridad y proporciona una ruta para la implantación de dichos sistemas con tecnología eléctrica/electrónica/programable (E/E/PE). La estrategia del estándar es en primer lugar obtener los requisitos de seguridad del sistema a partir del análisis de amenazas y riesgos, y después diseñar el sistema de seguridad cumpliendo esos requisitos (como se ha visto a lo largo de este estudio). Tiene en cuenta todas las posibles causas de fallo incluyendo los defectos aleatorios de hardware, los defectos sistemáticos tanto en hardware como en software y los factores humanos.

8. Pruebas exhaustivas

Mediante la validación y la verificación es posible comprobar que efectivamente todas las amenazas se tratan como se había especificado, además de comprobar también el resto de los requisitos impuestos al sistema.

La verificación comprueba que la salida de una fase cumple lo requisitos impuestos por la fase previa, de manera que se comprueba que no existe variación de los requisitos del sistema entre fases. Pero si la especificación de entrada es errónea, el proceso de verificación no tiene por qué detectarlo. Por lo tanto, la verificación debe ser completada con un proceso de validación. La validación es el proceso que confirma que el sistema completo es apropiado y consistente con los requisitos impuestos por el cliente.

La validación y verificación se llevan a cabo realizando pruebas. Las fases de prueba que existen a lo largo de todo el proceso de desarrollo incrementan la confianza en el sistema, al ir descubriendo defectos.

Las pruebas se realizan en varias etapas durante el proceso de desarrollo, y son distintas dependiendo de qué se quiera comprobar. La adecuada planificación de las pruebas es una parte esencial del proceso de desarrollo. Cada fase de prueba requiere un plan que pondrá de relieve cada prueba, su propósito, uso e interpretación.

En la literatura técnica se encuentran muchos estudios exhaustivos sobre el proceso de prueba, como (Friedman & Voas, 1995; Wallace, Ippolito, & Cuthill, 1996; Watson & McCabe, 1996). En 1995 finalizó el proyecto CONTESSE (CONTESSE, 1995), un proyecto conjunto entre universidades y empresas en Gran Bretaña, que estudiaba la verificación, validación y prueba en sistemas críticos de seguridad, haciendo especial hincapié en la simulación. En él aparecen descritos la mayor parte de los métodos de prueba que se utilizan en la actualidad en el desarrollo de este tipo de sistemas.

Conclusiones

La seguridad no está restringida únicamente a unos pocos sistemas de alta tecnología (como puede ser un sistema de control de vuelo) sino que puede afectar a una gran cantidad de desarrollos en sistemas cotidianos (por ejemplo, el sistema que controla que el microondas deje de funcionar cuando se abre la puerta). Por lo tanto, el desarrollo de aplicaciones con implicaciones de seguridad debería ser una disciplina más estudiada. Es necesario también reseñar la importancia que tiene la ingeniería del software para este tipo de desarrollos y en particular, la aplicación de las técnicas que aseguren una calidad del software (González Arechavala & Cuadra García, 2001).

Aunque únicamente se ha nombrado un estándar en este artículo, existen multitud de normas relacionadas con este tipo de desarrollos. Precisamente, en ocasiones la rigidez que imponen las normas para el desarrollo de estos sistemas provoca que el sistema no se realice de la manera óptima y que el desarrollo de estos sistemas sea muy costoso. La utilización de buenas herramientas puede ayudar a reducir en gran medida estos costes. ▮

Bibliografía

BCS. (1997). *Guidance for the adoption of tools for developing safety related software (Draft)*: British Computer Society.

CENELEC. (1998). ENV 50129: *Railway applications - Safety related electronic systems for signalling:* CENELEC: European Committee for Electrotechnical Standarization.

CONTESSE. (1995). *CONTESSE Test Handbook.*

Douglas, B. P. (1999). *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns.*

Falla, M. (1997). *Advances in Safety Critical Systems: Results and Achievements from the DTI/EPSRC R&D Programme in Safety Critical Systems:* DTI/EPSRC.

Friedman, M. A., & Voas, J. M. (1995). *Software Assessment: Reliability, Safety, Testability:* John Wiley & Sons, Inc.

González Arechavala, Y., & Cuadra García, F. d. (2001). Calidad del Software. *Anales de Mecánica y Electricidad* (Sep-Oct, Nov-Dic).

IEC. (2000). IEC 61508: "*Functional safety of electrical / electronics / programmable electronic safety-related systems*". Geneva: International Electrotechnical Commission, Geneva.

Rowe, R. (1994). *Safety-critical systems computer language survey results* (November). Available: comp.software-eng newsgroup.

Storey, N. (1996). *Safety-Critical Computer Systems*: Addison-Wesley.

Wallace, D. R., Ippolito, L. M., & Cuthill, B. (1996). *Reference Information for the Software Verification and Validation Process* (Special publication 500-234). Gaithersburg: NIST (National Institute of Standards and Technology- U.S. Department of Commerce), Computer System Laboratory.

Watson, A. H., & McCabe, T. J. (1996). *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric* (Special publication 500-235). Gaithersburg: NIST (National Institute of Stan-dards and Technology- U.S. Department of Commerce), Computer System Laboratory.