Objetivo y Agenda

- Objetivo:
  - Presentar requisitos que exigen diferentes normas internacionales para la implementación de software crítico y analizar sus beneficios, inconvenientes y riesgos.
- Agenda:
  - ¿Qué es la dependabilidad y seguridad-safety del software?
  - ¿Qué exigen diferentes normas para su implementación y demostración?
  - Beneficios, inconvenientes y riesgos

¿Qué es la dependabilidad y seguridad-safety del software?

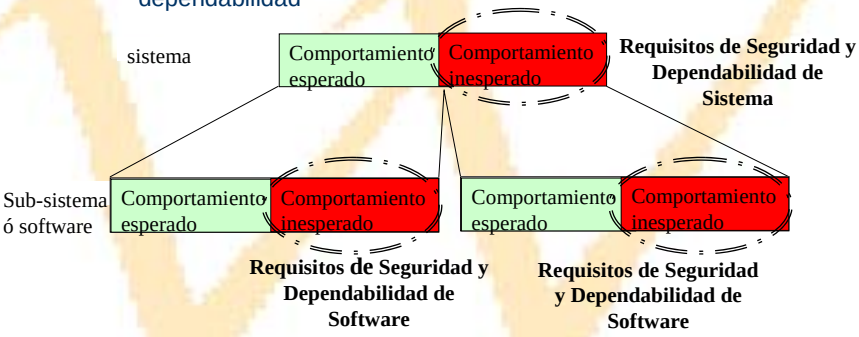
- En sistemas de consumo no considerados críticos respecto a su seguridad, la cantidad de software también se está duplicando casi cada año (ej, una televisión 'top-of-the-line' contiene software complejo cuando hace pocos años no tenía software). El usuario requiere cierto grado de disponibilidad y fiabilidad que no siempre se asegura.
- Implementar y verificar las características de seguridad 'safety' y dependabilidad en un producto software aun no se realiza sistemáticamente en muchos campos donde sí se debería considerar crítico
- Las características de dependabilidad y la seguridad se relacionan, aunque no son lo mismo.

¿Qué es la dependabilidad y seguridad-safety del software?

- La seguridad 'safety' se define como 'Esperanza de que un sistema en condiciones definidas no llega aun estado en el cual se ponga en peligro ni vidas humanas, ni la salud, ni las propiedades o el entorno'. También se define como: 'Estar libre de riesgos inaceptables'.
- La seguridad ('safety') se relaciona con el efecto final de cualquier fallo o evento. Es a nivel sistema.
- La dependabilidad se refiere a los fallos en sí mismos. En general, se contabiliza la frecuencia y cantidad de fallos en sí mismos, sin evaluar sus consecuencias peligrosas. Es también a nivel software.
- Decir software crítico se puede referir a aspectos de dependabilidad, o bien que tienen directa influencia en la seguridad del sistema.
- Es difícil imaginarse software crítico que sea seguro ('safe') pero no fiable. Si fallara mucho, pasaría a estado 'safe mode', sin hacer nada, y no sería un producto muy útil.

¿Qué es la dependabilidad y seguridad-safety del software?

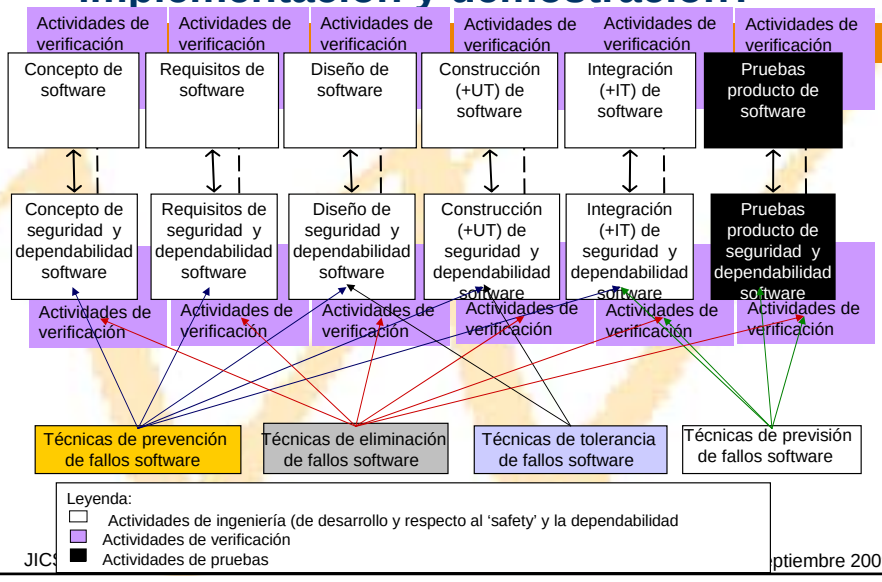
- Aumentar la dependabilidad o la seguridad del software es una cuestión de enfoque en el control/eliminación o tolerancia de fallos del software y qué partes del software son las afectadas. El término criticidad se utiliza como sinónimo de la seguridad y/o de la dependabilidad



¿Qué es la dependabilidad y seguridad-safety del software?

- En general, el software crítico en un sistema crítico existe cuando:
  - el software no falla de modo que cause o contribuya a un estado peligroso del sistema.
  - el software no falla en la detección o la corrección estado peligroso del sistema.
  - el software no falla en la mitigación o reducción del impacto del posible efecto si ocurre el accidente.
- Los fallos de software son fallos sistemáticos: se relacionan con una causa que sólo se puede eliminar modificando el diseño o código (o manuales de usuario/operaciones, etc.)

¿Qué exigen diferentes normas para su implementación y demostración?



¿Qué exigen diferentes normas para su implementación y demostración?

- SIL o Nivel de Criticidad: indicador del grado de confianza requerida para un producto o sistema, por lo que determinan qué medidas hay que poner para que, o bien la función de mitigación sea fiable o que el fallo ('failure') que pueda ocasionar un peligro no ocurra o se reduzca la severidad del efecto.

DO178B — **Nivel de SW**

Actividad	Clases de software			
	A	B	C	D
El código fuente cumple los requisitos de bajo nivel.	●	●	○	
El código fuente implementa la arquitectura del software.	●	○	○	
El código fuente es verificable.	○	○		
El código fuente es acorde a los estándares.	○	○	○	
La cobertura de las pruebas de todas las estructuras internas del software es completa	●	●		

LEYENDA:
● → La actividad debería realizarse por personas u organizaciones independientes.
○ → La actividad debería satisfacerse.
(en blanco) → El cumplimiento de la actividad queda a elección del usuario.

¿Qué exigen diferentes normas para su implementación y demostración?

- Según qué estándar, requerirá:
  - El uso de diferentes técnicas (prevención, tolerancia y/o análisis y eliminación) en las diferentes etapas del ciclo de vida del software para cada nivel de criticidad.
  - Que las mismas actividades, técnicas y métodos se utilicen para el software configurable y sus datos, pues hay sistemas críticos que reutilizan software crítico genérico, pero configurable a través de sus datos, y éstos son críticos también.
  - Diferentes niveles de independencia del personal para realizar algunas actividades.
  - Exigencias respecto a la confianza necesaria respecto al reuso como parte del software crítico.
  - Exigencias respecto a las herramientas a ser utilizadas para el desarrollo, la verificación y la validación del software crítico, que pueden afectar a la calidad del producto final

¿Qué exigen diferentes normas para su implementación y demostración?

Técnica o Análisis \ Nivel de esfuerzo de seguridad de software	MIN	MOD	FULL
8.4.1 Chequeos de código y estándares	*	*	*
8.4.2 Pruebas de unidad (obligatorio para software crítico) 8.5.8 Planes de pruebas de unidades críticas	√√	√√	*
8.4.4 Ejecución del programa con datos críticos	√	√	√√
8.5.1 Análisis del flujos de control	∅	∅	√
8.5.2 Análisis de los flujos de datos del código	√	√√	*
8.5.3 Análisis de Interfaces del código	√	√√	*
8.5.4 Análisis de código que no se usa	√	√√	*
8.5.5 Análisis de interrupciones	√	√√	*
8.5.6 Análisis de cobertura de las pruebas	√	*	*
8.5.7 Inspecciones formales del código fuente	√√	*	*
8.5.9 Análisis de los tiempos, Throughput, y ocupación	√√	*	*

Estándar de NASA para ‘Software safety’

	Técnica/Medida*	Ref.	SIL1	SIL2	SIL3	SIL4
1	Detección de fallo y diagnóstico	C.3.1	---	R	HR	HR
2	Códigos de detección y corrección de los errores	C.3.2	R	R	R	HR
3a	Programación por reafirmación de averías	C.3.3	R	R	R	HR
3b	Técnicas basándose en dispositivos externos de seguridad	C.3.4	---	R	R	R
3c	Programación diversa	C.3.5	R	R	R	HR
3d	Bloque de recuperación	C.3.6	R	R	R	R
3e	Recuperación hacia atrás	C.3.7	R	R	R	R
3f	Recuperación hacia delante	C.3.8	R	R	R	R
3g	Mecanismos de recuperación del fallo por relanzamiento de ejecución	C.3.9	R	R	R	HR
3h	Casos de memorización ejecutada	C.3.10	---	R	R	HR
4	Degradación "elegante"	C.3.11	R	R	HR	HR
5	Inteligencia artificial – corrección de fallo	C.3.12	---	NR	NR	NR
6	Reconfiguración dinámica	C.3.13		NR	NR	NR
7a	Métodos est MASCOT, S					
7b	Métodos ser					
7c	Métodos est CSP, HOL,					
8	Herramienta					
NOTA –	Conviene requisito CEI 615					
*	Las técnicas y me alternativas, se in alternativas.					

ara su

IEC 61508

	Técnica/Medida*	Ref.	SIL1	SIL2	SIL3	SIL4
1	Diagramas de bloques lógicos/funcionales	Véase la nota	R	R	HR	HR
2	Diagramas de secuencia	Véase la nota	R	R	HR	HR
3	Diagramas de flujo de datos	C.2.2	R	R	R	R
4	Autómatas de estados finitos/diagramas de cambios de estado	B.2.3.2	R	R	HR	HR
5	Redes de Petri temporales	B.2.3.3	R	R	HR	HR
6	Tablas de decisión/de verdad	C.6.1	R	R	HR	HR
NOTA – Los diagramas de bloques del softwares/funcionales y los diagramas de secuencia se describen en la Norma CEI 61131-3.						
* Las técnicas y medidas apropiadas se deben seleccionar en función del nivel de integridad de seguridad.						

JICS.XI –

Métodos y medidas (Borrador ISO/IEC 26262)		ASIL			
1	Un punto de entrada y uno de salida en subprogramas y funciones[a]	A	B	C	D
2a	No objetos ó variables dinámicos[a, b]	+	+	++	++
2b	Pruebas on-line durante la creaciónde variables dinámicas[b]	+	++	++	++
3	Inicialización de variables[a]	+	++	++	++
4	No uso múltiple de variables[a]	++	++	++	++
5	Evitar uso de variables globales o justificar su uso[a]	+	+	++	++
6	Uso limitado de punteros[a]	o	+	+	++
7	No conversiones de tipos implícitas[a, b]	+	++	++	++
8	No flujo de control o de datos implícito	+	++	++	++
9	No jumps no-condicionales[a, c, d]	++	++	++	++
10	No recursións[d]	+	++	++	++

[a,] Las medidas 1, 2a, 3, 4, 5, 6, 7 y 9 puede no sea aplicable a notaciones de modelado gráfico utilizado en desarrollos con modelado.
[b] Las medidas 2a y 2b no son necesarias si se usa un compilados que asegure que hay suficiente espacio asignado a las variables y objetos dinámicos o que inserta pruebas para la asignación dinámica de espacio (esto es, 'stack bound checking' – chequeo de los límites de las pilas)
[c] Las medidas 7 y 9 no son aplicables a lenguaje ensamblador
[d] Las medidas 9 y 10 previenen los jumps y variables globales al flujo de control y de datos modelados
Nota 1 Para lenguaje C, [MISRA C] cubre la mayor parte de las medidas enumeradas en la tabla.
Nota 2 Dependiendo del lenguaje de modelado o programación utilizado, será necesario adaptar esta tabla

¿Qué exigen diferentes normas para su implementación y demostración?

- Los diferentes requisitos para la implementación de la seguridad y dependabilidad del software en sistemas críticos requieren:
  - Que el equipo humano tenga conocimientos de los mecanismos específicos de seguridad y dependabilidad, tanto para la ingeniería (tolerancia a fallos y prevención de fallos), como para la verificación y análisis del software (eliminación de fallos). Que sepan seleccionar cuáles son las técnicas a utilizarse y combinarse, valorando otros requisitos de rendimiento, tiempo real, ocupación de memoria, operacionales, etc. y, en la mayor parte de los casos, a ser propuestas y negociadas con el cliente. Por tanto, en las organizaciones:
    - Posible formación necesaria
    - Limitación del personal que podrá desarrollar estos productos

¿Qué exigen diferentes normas para su implementación y demostración?

  - Roles independientes dentro de cada proyecto: a veces los que desarrollan deberán ser independientes de los que verifican, de los que validan y de los que evalúan la seguridad, por tanto:
    - Más compleja organización aumentando la gestión de los proyectos
    - Más compleja organización implica más formalidad en el proyecto, que puede conllevar a más duración y más esfuerzo
  - Cambios en las herramientas que se utilicen para los desarrollos: no cualquier herramienta puede ser utilizada, por tanto:
    - Compra y formación de nuevas herramientas de desarrollo, o
    - Cualificación y pruebas de las herramientas existentes
  - Que la reutilización de software ya existente se realice cumpliendo los mismos requisitos que el software crítico nuevo, requiere realizar actividades extra.

Beneficios, inconvenientes y riesgos

- El desarrollo de software crítico es más caro y largo de implementar.
- Pero:
  - ¿Qué riesgos se asumen y quién los asume si no se implementan los sistemas críticos con alto contenido en software con estos requisitos?
  - ¿Cómo se asegura mayor fiabilidad y seguridad del software empotrado en los más de 100 ECUs de un automóvil cuando hoy día, en general, no se exige que demuestren el cumplimiento de este tipo de requisitos? (Nota: La norma ISO/IEC 26262 para seguridad funcional en vehículos de carretera aun no se ha publicado, y aun no es de obligado cumplimiento).
  - ¿Cómo asegurar que los dispositivos médicos, cada vez más electrónicos (controlados por software), cumplan los más estrictos requisitos de dependabilidad y de seguridad?
  - ¿y el software del cajero automático? ¿y el software de control semafórico?...
  - ¿Quién debe decidir qué requisitos son los que se implementarán y después se verificarán y validarán en estos sistemas críticos? ¿Quién será el responsable de su aseguramiento?

Beneficios, inconvenientes y riesgos

- Frecuentemente, son los fabricantes los que deciden el riesgo a ser sumido por los usuarios.
- Frecuentemente, estos riesgos se miden por datos de mercado y ventas, más que por datos de riesgos de seguridad para los usuarios.
- Pero, por otro lado, ¿Se sabe el riesgo que asume al utilizar o depender de sistemas críticos controlados por software?
- Además, las reparaciones y mantenimiento de estos sistemas y de su software son cada vez más sofisticados.
- ¿Quién deberá asegurar la seguridad y satisfacción de los usuarios de estos sistemas?
- ¿Es necesaria más regulación y legislación para exigir a los fabricantes cumplir unos mínimos de calidad (ej que en las homologaciones de coches se revise el software y sus características, etc.)?
- ¿Serán los seguros más baratos cuanto más fiable y seguro sea el software que va a bordo de los automóviles pues menos se tendrá que ir a repararlo? ¿Habrá un mayor período de garantía para los sistemas de consumo más sofisticados?

Conclusiones

- Sistemas más sofisticados son más caros
- Queremos mejor seguridad y mayores garantías
- Alguien tiene que velar por asegurar la seguridad y disponibilidad de los sistemas controlados por software
- Algunas medidas a tomar:
  - Más formación
  - Más información
  - Más regulación
  - Más métodos de abaratamiento:
    - Fomento del reuso y arquitecturas estándares fiables
    - Menos sofisticación
    - etc

Gracias

- Por SU atención
- Para cualquier otra información

rodriguezdapena@softwcare.com