ALTEN Global Technologies

Functional safety standards comparison

Abstract

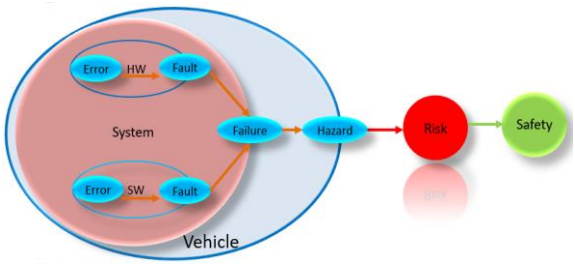
[Functional safety is very important for the safety critical systems in any domain. The main objective of the functional safety is to reduce unacceptable hazards' risk probability to acceptable levels. The standard guidelines based on past failures and fixes helps in not repeating the already identified problems. Each domain such as Aerospace, Automotive have come up with the Functional safety standards based on the domain's core functionality and associated safety context. This document aims to provide comparison between different Software Functional safety standards from Avionics, Automotive and Medical electronics domain.]

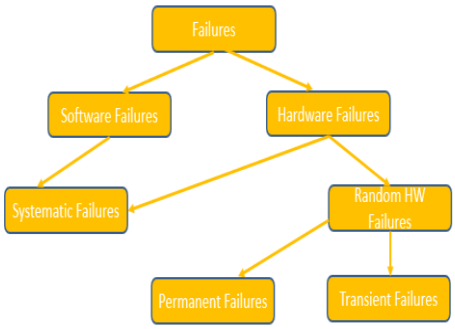
1 INTRODUCTION

This document provides overview and comparison of software functional safety standards in Avionics, Automotive and Medical electronics domains.

2 FUNCTIONAL SAFETY OVERVIEW

Error in a Hardware/Software results into a Fault. A fault or group faults in a system result into a Failure which may trigger Hazardous condition. Depends on the Hazard's severity, there will be risk to system safety. The below diagram explains the relationship between Error, Fault, Failure, Hazard, and Risk.





Failures in a system are two types a) Systematic failures b) Random hardware failures.

Systematic failures are development errors which are deterministic, risks due to systematic failures can be prevented by developing Hardware and Software to correct design standard(guidelines) and perform safety analysis and system verification. Risks due to Random hardware failures can be reduced by deploying correct active safety mechanism (e.g., Built in tests) into system.

3 FUNCTIONAL SAFETY STANDARDS OVERVIEW

The main objective of the functional safety is to reduce unacceptable hazards' risk probability to acceptable level. Functional safety standards provide methods and guidelines for all the phases and activities of system life cycle (Inception, Safety goal identification, Requirements specification, Hardware design, Software design, Safety analysis, verification, Installation, service, and Maintenance).

Avionics domain functional safety standards:

ARP 4754A: This standard provides guidelines for development of civil aircraft systems.

ARP 4761: This standard provides guidelines and methods for safety assessment for certification of civil aircraft.

DO 254: This standard provides guidelines for development of airborne electronic hardware.

DO 178C: This standard provides guidelines for production of software for airborne systems.

Automotive domain functional safety standards:

ISO 26262: This standard provides guideline for management, development, verification, production, and service of electrical and/or electronic (E/E) systems within road vehicles.

ISO 25119: This standard provides guideline for design and verification of electrical and/or electronic (E/E) systems in Tractors in agriculture and forestry.

Medical domain functional safety standards:

IEC 62304: This standard provides guideline for design, development, verification, and maintenance of medical device software. This standard is to be used together with other appropriate standards (ISO 14971, ISO 60601) when developing a medical device.

4 FUNCTIONAL SAFETY STANDARDS COMPARISON

Life cycle phase	Sub Phase	Activity	Automotive electronics functional safety standards	Avionics electronics functional safety standards	Medical electronics functional safety standards	Remarks
			ISO 26262	**DO 178C**	**IEC 62304**	
Safety classification	Concept phase	Hazard analysis	Assign Automotive safety integrity level (A, B, C, D, QM) according to the possible effects on the driver, passenger or pedestrians resulting from a HAZARD to which the SOFTWARE SYSTEM can contribute. Automotive safety integrity level is derived based on hazard's severity (i.e., effect on driver, passenger, pedestrians), probability	Assign development assurance level (A, B, C, D, E) according to the possible effects on the Flight Crew, passengers or aeroplane resulting from a HAZARD to which the SOFTWARE SYSTEM can contribute. • DAL A- Catastrophic i.e. Loss of aeroplane, • DAL B- Hazardous i.e. Large reduction in	Assign software safety class (A, B, or C) according to the possible effects on the patient, operator, or other people resulting from a HAZARD to which the SOFTWARE SYSTEM can contribute. • Class A: No injury or damage to health is possible • Class B: Non-SERIOUS INJURY is possible	In all the standards, the integrity level is derived based on how failure affects the end user's safety.

			of exposure (i.e., occurrence) and controllability (possible control by driver or other passengers at risk). Note: ASIL D: Highest integrity level, ASIL A: Lowest integrity level	• safety margin,serious or fatal injuries • DAL C- Major i.e. Signficant reduction in safety margin, increases Crew work load, possible injuries • DAL D - Minor i.e. Signficant reduction in safety margin, physical discomfort to flight crew and passengers • DAL E-No safety effect	• Class C: Death or SERIOUS INJURY is possible Note: Software may contribute to hazardous situations due to incomplete specification, COTS misbehaviour, Hardware failure forcing unpredictable Software operation and software misuse by the user.	
Software Development	Planning phase	Software Plan	Prepare the following plans: Safety plan, Software verification plan, Design and coding guidelines,	Prepare the following plans: Plan for software aspects of certification, Software development plan, Software verification plan,	Prepare the following plans: Software development plan, Software development standards, methods, and tools plan	In all the standards, Development and verification plans are mandatory.

			Tool application guidelines, Documentation plan, Software configuration management plan,	Software requirements standard, Software design standards, Software code standards Software configuration management plan,	Software integration and integration testing plan, Software verification plan, Software Risk Management plan, Documentation plan, Software configuration management plan, Software maintenance plan	In Automotive and Medical standards, additionally, a Documentation plan is required. In Medical standards, additionally, Risk management is required.
	Requirements phase	Software Requirements capture	Derive Software safety requirements from the technical safety concept and the system design specification. The following are the example categories for software safety requirements: • functions that enable the system to	Derive software requirements from System requirements. The following are the example categories software requirements • Functional requirements • Interfacing requirements	Derive software requirements from system requirements. The following are the example categories software requirements: • Functional and performance requirements esp., for SOUP (software of	In all standards, the software requirements are derived from the System requirements. In the Automotive

			achieve or maintain a safe state; • functions related to the detection, indication and handling of faults of safety-related hardware elements; • functions related to the detection, notification, and mitigation of faults in the software itself • functions related to on-board and off-board tests; • functions that allow modifications of the software during production and service; and • functions related to performance or time-critical operations • the timing constraints; • operating mode of the vehicle, the	• Performance requirements • Safety related requirements • Tolerances where applicable • Parameter related (configuration) requirements Develop software requirements to Software requirements standard.	unknown provenance i.e. COTS) items • Resource(Memory,Time..etc) requirements • Software System inputs and outputs • Interfaces between the software system and other systems • Software-driven alarms, warnings, and operator messages • Security requirements • Usability engineering requirements that are sensitive to human errors and training • Data definition and database requirements • Installation and acceptance requirements • Operation and maintenance requirements • User maintenance requirements • Regulatory requirements	standard, the certification attention is limited to Safety requirements alone, whereas the aerospace standard recommends the same treatment for safety and non-safety requirements.

			system, or the hardware, having an impact on the software.		• Risk Control measures	
	Design phase	Software Architecture	Prepare Software architecture with software external interfaces, component hierarchy levels, function call sequence, data and control flow between components, testability, and configurability. If new hazards introduced by architecture, it must be sent to Hazard analysis and risk assessment.	Prepare software architecture with interface between components (Data and control flows) and follow Software design standard.	Prepare software architecture with interfaces between the software items and the components external to the software items (both software and hardware) along with segregation between the items for risk control, also specify system hardware and software required by SOUP items.	In Aerospace and Automotive standards, the COTS is not allowed to use without qualification, whereas in medical standards, it is permitted if the COTS supports the intended use.
		Software Low level design	Prepare low level unit(s) with attributes (Consistency, simplicity, readability, comprehensibility,	Prepare low level design to Software design standard.	Prepare detailed design for each software unit and specify interface between each unit.	

			robustness, and testability).			
	Safety analysis	Software Design failure mode effect analysis	Perform software design failure mode effect analysis to check the efficiency of safety mechanisms.	Derived software high level and low-level requirements shall be sent to system safety assessment. Note: Derived means requirements that are not directly tracing to its parent requirements.	Analysis of software contributing to hazardous situations	In Automotive standards, explicit software design FMEA needs to be performed to improve the controllability of failures at the early design stage. In Medical standards, Risk analyses must be performed to reduce the software failures that lead to

						Hazardous situations.
		Software Dependent failure mode effect analysis	Analyse freedom from interference and interdependencies between the software components.	None	None	Software-dependent failure mode analysis is not a requirement in the Aerospace and Medical standard.
	Development phase	Source code development	Allows both model-based code generation and normal source code development.	Develop the source code to Software code standards. Allows both model-based code generation and normal source code development. Note, Refer DO-331 for model-based development guidelines and DO-332 for object-	Develop the source code to Software code standards. Implement the Risk control measures as per the requirements.	Both Aerospace and Automotive standards have explicitly stated acceptance of Model-based development. The medical standard did not provide any guidelines on Model-

				oriented technology guidelines.		based development.
	Verification phase	Reviews	Perform Walk-throughs, Inspection and Reviews to check the phase outputs are implemented from phase inputs and standards met.	Perform Walk-throughs, Inspection and Reviews to check the phase outputs are implemented from phase inputs and standards met	Perform Walk-throughs, Inspection and Reviews to check the phase outputs are implemented from phase inputs and standards met	Verification objectives are same in all the standards.
		Analysis	Perform the following analysis: Source code Static analysis Control flow analysis Data flow analysis Stack analysis	Perform the following analysis: Source code Static analysis Control flow analysis Data flow analysis Stack analysis	Perform the following analysis: Event sequence analysis Data and control flow analysis Resource usage analysis	

			Timing analysis Traceability analysis Requirement coverage analysis Structural coverage analysis	Timing analysis Traceability analysis Source code to object code traceability analysis Requirement coverage analysis Structural coverage analysis	Fault handling (error definition, isolation, and recovery) analysis Initialisation of variables analysis Self-diagnostics analysis Memory management and memory overflows analysis Boundary conditions analysis	
		Software requirements testing	Software safety requirements testing verifies the following: a) Software compliance with software requirements	Software requirements testing verifies the following: a) Software compliance with software requirements	Software system requirements testing verifies the following: a) Software compliance with software requirements	In all the standards, requirements-based testing is recommended, and a high

			b) Software robust with the requirements Expected coverage metrics: • software requirements coverage Software requirements testing can use the following environment(s): • Target Hardware-in-the-loop • Target representative Simulators/Emulators • Vehicle	b) Software robust with the requirements c) compatible with target computer Expected coverage metrics: • software requirements coverage • Data and control coupling coverage • Structural coverage (if possible) Software requirements testing can use the following environment(s): • Target environment • Target representative environement (e.g.	Expected coverage metrics: • software requirements coverage • RISK CONTROL measures • Usability • test types (e.g., fault, installation, stress) Software requirements testing can use the following environment(s): • Simulated environment • Actual target hardware • Medical Device	degree of credit is given when the test environment is close to the final target environment.

				Emulator, Simulator)	Testing methods: • Black box testing • White box testing	
		Software integration testing	Software integration testing verifies the embedded software a) compliance with Software architecture b) compliance with Hardware and software interface specification c) have sufficient resources (Memory and CPU cycles) for execution Software Integration testing methods: a) Requirements based testing b) Interface testing	Software integration testing verifies: a) Inter relationship (input and output flows) between the software components as per software requirements and software architecture Expected coverage metrics: • Data and control coupling coverage	Software integration and integration testing verifies: a) Inter relationship (input and output flows) between the software components as per software requirements and software architecture. b) Program responses to invalid, unexpected, and special inputs. c) Timing requirements are met Software integration testing can be performed:	In all the standards, the main objective of Software integration testing is to verify the data and control coupling between the software components.

			c) Resource testing d) Fault injection testing e) Back-to-back comparison testing (in case of Model based development) Expected coverage metrics: • Function coverage • Call coverage Software integration testing can use the environment(s): • model-in-the-loop tests; • software-in-the-loop tests; • processor-in-the-loop tests; • hardware-in-the-loop tests	Software integration testing can use the environment(s): • Target environment • Target representative environement (e.g. Emulator, Simulator)	• In a simulated environment • On actual target hardware • On the full Medical Device Testing methods: • White box testing • Black box testing	

		Unit testing	Unit testing verifies the software unit a) Complies with unit design b) Robust with unit design c) does not have dead code d) have sufficient resources (Memory and CPU cycles) for execution Unit testing methods: a) Requirements based testing b) Interface testing c) Resource testing d) Fault injection testing e) Back-to-back comparison testing (in	Unit testing verifies the software unit a) compliance with low level requirements b) robust with low level requirements Expected structural coverage metrics: • Statement coverage • Decision coverage • MCDC Software unit testing can use the environment(s): • Target environment • Target representative environement (e.g. Emulator, Simulator)	Unit testing verifies the Software Unit met the acceptance criteria below: a) proper event sequence. b) data and control flow. c) planned resource allocation. d) fault handling (error definition, isolation, and recovery). e) initialisation of variables. f) self-diagnostics. g) memory management and memory overflows; and h) boundary conditions.	In all the standards, the main objective of unit testing is verifying software compliance with the low-level requirement and structural coverage.

			case of Model based development) Expected structural coverage metrics: • Statement coverage • Decision coverage • MCDC Unit testing can use the environment(s): • model-in-the-loop tests; • software-in-the-loop tests; • processor-in-the-loop tests; • hardware-in-the-loop tests.			
	Traceability		Traceability needs to be produced among System Requirements, Software Requirements, Design, Code and Test reports	Traceability needs to be produced among System Requirements, Software Requirements, Design, Code and Test reports	Traceability needs to be produced among System Requirements, Software Requirements, Design, Code and Test reports	All the standards recommend Traceability among System Requirements,

						Software Requirements, Design, Code, and Test reports to ensure all the requirements are implemented, and no additional implementation exists without requirement.
Risk Management			**ISO 26262**	**DO 178C**	**IEC 62304**	
			Perform Software design failure mode effect analysis and testing to confirm all the risks are handled correctly in the software design.	Perform testing to confirm all the risks(faults) are handled i.e., unacceptable faults are handled correctly in the software.	Perform risk analysis to identify the software contributing to Hazardous situations. Specify the Risk control measures and implement the software accordingly.	In all the standards, high importance is given to manage the failures and its impact.

					Prepare traceability from software HAZARDS and Software Item to verification of Risk control measure.	
Configuration			**ISO 26262 Part 8**	**DO 178C**	**IEC 62304**	
	Planning phase	Configuration plan	Prepare the following plans: Configuration plan Change management plan	Prepare the following plans: System configuration plan, HW configuration plan, Software configuration plan	Prepare the Software configuration management plan	In all the standards, configuration management objectives are the same.
	Configuration phase	Configuration	Unique identification of configuration item and version, establish baseline, maintain archival and retrieval.	Unique identification of configuration item and version, establish baseline, maintain archival and retrieval.	Unique identification of configuration item and version, establish baseline, maintain archival and retrieval.	
		Change control	Prepare problem reports, Investigate the problems,	Prepare problem reports, Investigate the	Prepare problem reports, Investigate the problems,	

			change control, Change review, Protection against unauthorized access.	problems, change control, Change review, Protection against unauthorized access.	change control, Change review, Protection against unauthorized access.	
Quality Assurance			**ISO 26262 Part 8**	**DO 178C**	**IEC 62304**	
	Quality assurance phase	Confirms lifecycle process and activities are performed according to the plans.	Recommends Quality management system complying with Quality management standards ISO/TS 16949, or ISO 9001 or equivalent.	Compliance to quality assurance objectives in the Avionics standards (DO 178C) need to be met.	Recommends Quality management system complying with: • Quality management standard ISO 13485 or • a national quality management system standard or • a quality management system required by national regulation Also recommends ISO/IEC 90003 for guidance on applying quality management system requirements to software.	In all the standards, quality assurance is mandatory to confirm products are developed according to the plans.
Independence			**ISO 26262 Part 2**	**DO 178C**	**IEC 62304**	

			Only Confirmation review, audit and assessment activities need to be performed independently based on Automotive safety Integrity level.	Both engineering and certification activities need to be done independently based on Development assurance level.	Independence of personnel is not included in this standard. It is considered covered in ISO 13485.	For higher safety integrity level systems, independence is recommended in all standards.
Deviations			**ISO 26262 Part 2**	**ARP 4754A , DO 254 and DO 178C**	**IEC 62304**	
			Deviation is not acceptable, but the tailoring is acceptable. The subphase, activity or task can be combined or split or added to another phase or subphase.	Deviations from the standards need to be recorded in the Plans and get approved from the certification authority.	Where any requirements in the standard contain “as appropriate” and were not performed, documentation for the justification is necessary for later assessment.	In all the standards, the deviations/tailoring are acceptable with approval.
Certification/Assessment			**ISO 26262 Part 2**	**DO 178C**	**IEC 62304**	
	Certification/Assessment phase	Certification	Evaluates achieved functional safety by checking safety case (work products) compliance to	Evaluates achieved functional safety by checking accomplish summary compliance to avionics safety standards (ARP 4754A,	Evaluates Compliance by inspection of all documentation required by this standard	In aerospace standards, the external audit is mandatory, whereas, in automotive

			automotive safety standard ISO 26262	ARP 4761, DO-254 and DO-178C)	including the Risk Management File, and assessment of the processes, activities and tasks required for the software safety class.	and medical standards, the external audit is required for high Safety integrity systems.
Maintenance			**ISO 26262**	**DO 178C**	**IEC 62304**	
	Maintenance phase	Change of Application or Development Environment	Perform change impact analysis. As part of this analysis, identify effected phases, Activities, and artefacts. Perform the artefacts update according to change request approval process. Follow problem reporting process including change review and change control to address the problem on already released certified software.	Perform change impact analysis. As part of this analysis, identify effected phases, Activities, and artefacts. Perform the artefacts update according to change request approval process. Follow problem reporting process including change review and change control to address the problem on	Perform change impact analysis. As part of this analysis, identify effected phases, Activities, and artefacts. Perform the artefacts update according to change request approval process. Follow problem reporting process including change review and change control to address the problem on already released certified software.	In all the standards, the maintenance objectives recommend impact analysis and updating the system according to the change review and control process.

			Conduct regression tests to demonstrate that defects have not been introduced into previously integrated software.	already released certified software. Conduct regression tests to demonstrate that defects have not been introduced into previously integrated software.	Conduct regression tests to demonstrate that defects have not been introduced into previously integrated software.	

5 CONCLUSION

Manufacturers shall derive safety goals specific to safety context of the System is very important. Once the safety goals are derived, right Safety class needs to be assigned, so the right rigor will be applied during the development and verification of the System. Any deviations from Mandatory guidelines in the standards need to be agreed at the start project. The accomplishment summary shall provide the deviations from the plans and open problem reports of the project, the safety team shall go through all the deviations and open problem reports to identify the safety impact.

All these functional safety standards are providing state of the art guidelines specific to that domain. Manufacturers shall adhere to the standards and develop safety critical system for safer world.

6 REFERENCES

1. DO 178C: Software considerations in Airborne systems and Equipment certification

2. ISO 26262: Road vehicles Functional safety

3. IEC 62304: Medical device software life cycle processes

About Author

Appala Naidu B is a senior software architect working for safety-critical embedded systems development for 18years in the Aerospace and Automotive domains. He has very good experience with Functional safety products development and certification.