Introduction to ISO 26262

To help address veh459 cele saf, the 10 ernatdizat626) put forth26262

Topic	Synopsys Software Integrity Portfolio Support	Rule Mapping	ASIL			
			A	B	C	D
No unconditional jumps (1i)	In the context of the C and C++ languages, these issues would be addressed by coding standards checkers.	MISRA-C 2012 Section 8.15 rules would identify control flow issues such as unconditional jumps.	++	++	++	++
No recursions (1j)	Coverity is able to identify both direct and indirect recursion at considerable depth.	MISRA-C 2012 Rule 17.2 forbids recursion.	+	+	++	++

Testing of the embedded software (ISO 26262:2018, Part 6, Section 9)

Section 9 of the standard covers software unit verification and outlines a number of requirements to support the core functional requirements as well as highlighting safety relevant activities which should also be carried out at this phase of the life cycle.

Synopsys software integrity tooling supports a number of these methods directly, but in addition, Synopsys recommends the output from prior tooling operation should be leveraged as part of manual review activities and become integrated into the software sign-off process.

ISO 26262:2012, Part 6, Table 7: Methods for software unit verification

Topic	Synopsys Software Integrity Portfolio Support	ASIL			
		A	B	C	D
Walk through (1a)	For manual review processes, it's recommended that users consult the output from the relevant analysis tools as part of their review process: • Coverity static analysis findings • Defensics testing reports • Black Duck policy reports and Bill of Materials In addition to this, functionality such as secondary review may be implemented in each tool, before a finding is dismissed, as a matter of good practice in defect management.	++	+	o	o
Pair-programming (1b)		+	+	+	+
Inspection (1c)		+	++	++	++
Semi-formal verification (1d)	Coverity utilizes both semiformal and formal methods as part of its analysis—for example, when traversing conditions in code—and uses this information to augment its understanding of the program under analysis.	+	+	++	++
Formal verification (1e)		o	o	+	+
Control flow analysis (1f)	Coverity creates an internal graph of program control flow and uses this to detect control flow–related problems such as unreachable code, infinite loops, and dead code.	+	+	++	++
Data flow analysis (1g)	Coverity performs value tracking internally to identify several categories of defect including tainted data flow, buffer size miscalculations, and division by zero errors.	+	+	++	++
Static code analysis (1h)	Coverity performs static code analysis based on abstract representation to detect coding standards violations.	++	++	++	++
Static analyses based on abstract representation (1i)	Coverity performs static code analysis based on abstract representation to detect many categories of defects such as concurrency issues, security issues, memory management, and resource management, which extends beyond the level of complexity of simple coding standards checks.	+	+	+	+

Requirements based test (1j)	For network interfaces and file formats, Defensics generates test cases based on protocol specifications as requirements.	++	++	++	++
Interface test (1k)	For network interfaces and file formats, Defensics supports testing interfaces that communicate in both supported and proprietary protocols.	++	++	++	++
Fault injection test (1l)	Defensics generates test cases for use in fault injection by manipulating message payload, sequence, and meta information, according to the standard or custom protocol or format specified.	+	+	+	++
Resource usage evaluation (1m)	Coverity detects a number of categories of resource mismanagement including excessive program stack size allocations and failure to release allocated resources (resource leaks).	+	+	+	++
Back to back comparison test between model and code, if applicable (1n)	Synopsys consulting services can provide code review and threat model creation, which can be used to compare implemented code to software design.	+	+	++	++

Testing of the embedded software (ISO 26262:2018, Part 6, Section 11)

Software unit testing is an important requirement in the ISO26262 standard. Software unit tests must be planned, specified, and executed.

The ISO standard specifies that the goal of these verification activities is not just to affirm compliance with specification of hardware-software interface(s), but also to provide confidence in the absence of unintended functionality and properties.

To address this part of the standard, Synopsys proposes that organizations adopt fuzz testing (“fuzzing”) – a method by which known good data is mutated to create new test cases, intended to discover and exercise boundary conditions in the target device code.

Defensics is applicable primarily to interfaces exchanging data with external systems – such as network protocols, or file

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com