WHITE PAPER

Meeting ISO 26262 Guidelines With the Synopsys Software Integrity Portfolio

Table of contents

The average car is expected to contain 300 million lines of code in the next decade, up from 100 million lines of code in today's cars.[1] And software is expected to account for 90% of automobile innovation.[2] Software controls everything from safety-critical systems like brakes and power steering, to basic vehicle controls like doors and windows, V2V, V2I, and sophisticated infotainment systems and telematics. However, with the exponential growth of software comes a dramatic increase in software defects. The average car is expected to contain up to 150,000 bugs,[3] many of which could damage the brand, hurt customer satisfaction and, in the most extreme case, lead to a catastrophic failure. Toyota issued two recalls of its popular Prius model in 2018, affecting 2.43 million vehicles, because of a software glitch that caused the cars to stall, increasing the risk of a crash at high speeds.[4] That's just one example out of dozens since 2000, impacting manufacturers from Alfa Romeo, Fiat, and BMW to Ford, GM, and Nissan, and affecting tens of millions of vehicles.[5]

Introduction to ISO 26262

To help address vehicle safety, the International Organization for Standardization (ISO) put forth ISO 26262 in 2011 for road vehicle functional safety. The standard was created to provide guidance to avoid the risk of systematic failures and random hardware failures through feasible requirements and processes. ISO 26262 is the adaptation of IEC 61508 to comply with needs specific to the application sector of electric and or electronic elements such as power supplies, sensors and other input devices, data highway, and other communication paths, actuators, and other output devices. The purpose of this paper is to discuss how the Synopsys software integrity portfolio can be used to help meet the guidelines set forth in ISO 26262.

The standard is comprised of 12 parts that span the breadth of the automotive safety lifecycle including management, development, production, operation service, and decommissioning.

Challenges in automotive software development

Modern software development environments and practices in the automotive domain are marked by increasing demands for agility, consolidation of functionality, and rapid change. Additionally, with the advent of autonomous vehicles and increased connectivity, the spectrum of risk and exposure has broadened.

Related challenges that Synopsys has seen across our client base include:

- Larger, more complex software
  - Complications for traceability and identification of code re-use, particularly with open source components in first-party and third-party platforms
  - Adoption and management of coding standards (e.g. MISRA) at large scale in complex codebases
- Supply chain and supplier management
  - Supply chain management of safety and security requirements
  - Use of qualified tooling during software development
- Communication reliability and robustness requirements for connected components
- Management of cybersecurity vulnerabilities during design and after release to production
- Adoption of qualified tooling for use during development

Introduction to the Synopsys Software Integrity Portfolio

Synopsys' software integrity portfolio is designed to help developers, management, and organizations easily find and fix quality and security problems early in the software development lifecycle, as the code is being written, without impacting time-to-market, cost, or customer satisfaction.

Synopsys solutions augment traditional testing, including quality assurance (QA), functional and performance testing, and security audits, providing development teams with a quick and easy way to test their code for defects and to ensure critical code has been properly tested in a non-intrusive manner. This enables development to stay focused on innovation, management to get visibility into problems early in the cycle to make better decisions, and organizations to continue to quickly deliver high-quality products to market for competitive advantage.

Synopsys provides the industry's leading development testing portfolio with tailored solutions for development and management teams that can assist organizations with achieving ISO 26262 compliance.

In addition, Coverity Static Analysis is certified by TUV SUD Product Service GmbH according to the applicable requirements of the standard IEC 61508 and ISO 26262 for developing and testing safety-critical software.

Coverity Static Analysis – Synopsys delivers the industry's most accurate and comprehensive static analysis solution. It is used by developers around the world to improve the quality of their code by enabling them to find and fix defects in C/C++, Java, and C# code (along with many other languages) faster, which results in lower overall costs. Organizations can create customized analysis rules to support their unique requirements through the Coverity Extend Software Development Kit (SDK). Coverity also includes a powerful framework for implementing custom coding policies, named Code XM. Static analysis is included in ISO 26262 as a formal verification method for adherence to the coding guidelines and can be used for reviewing pieces of code that access memory locations containing safety-related data as specified in ISO 26262 Annex D, freedom from interference by software partitioning.

Coverity Connect for on-premises deployment – This web portal solution provides a centralized defect management workflow that enables developers and managers to quickly view defects in the source code and take the appropriate action to resolve them. Developers and managers can identify defects associated with a particular Automotive Safety Integrity Level (ASIL) and find where defects occur across various code branches. This capability is a critical time-saver for development teams as code reuse is prevalent in the automotive industry.

Coverity Policy Manager – This solution enables organizations to establish and enforce consistent policies tied to safety requirements defined in ISO 26262, by ASIL level. It enables users to define clear and comprehensible policies to meet the key requirements for this standard. Once the policies have been established, organizations can test against them with Coverity Static Analysis and quickly visualize areas of risk in the project by component and ASIL level. Managers and executives get a hierarchical view of risk, can understand the relative effort required to address the defect, and can drill down to details to pinpoint the specific issues or verify that specific safety requirements have been satisfied.

Black Duck – The Black Duck solution from Synopsys performs software composition analysis (SCA), which enables software developers to identify third-party and/or open source components within their existing codebase. A typical HMI system may encompass hundreds of software packages, many of which originate from open source platforms such as Automotive Grade Linux (AGL) or Android. By utilizing Black Duck's software composition analysis, a developer may easily discern which packages are re-use and apply "proven in use" arguments to exclude these from safety scope if appropriate. The additional benefits of adopting Black Duck in this way also include alerting to software license obligations, as well as security vulnerabilities that originate from open source software packages.

Defensics – Fuzz testing, or "fuzzing," is a method of test case generation and delivery that manipulates normal or expected inputs in an attempt to trigger failure modes within the target system. Through fuzzing, it is possible to identify crash-causing software defects that may be the root cause of cybersecurity vulnerabilities or safety problems. Defensics is the industry's only professional grade fuzz testing tool and is well known for identifying the anomaly that gave rise to the Heartbleed SSL vulnerability in 2014.

Consulting services – Through a network of more than 200 consultants worldwide, Synopsys provides expert security services to clients across all industries. Within the automotive industry, Synopsys provides Threat And Risk Assessment (TARA) as well as security validation (penetration testing) engagements. In addition, Synopsys can provide expert strategy consulting to help organizations develop competencies within their teams.

Applying the Synopsys software integrity portfolio to ISO 26262 requirements

General topics for the product development at the software level (ISO 26262:2018, Part 6, Section 5)

Use of continuous integration, and integration of automated tooling

The ISO 26262 standard calls out examples of methods and development approaches that support consistency of development activities and work products. In particular highlighted in Note 1, Example 2 is the role of automated tooling to achieve this goal.

All of the Synopsys software integrity tooling detailed in this document are highly amenable to automated invocation, and include plugins for the most common continuous integration and continuous delivery tools such as Jenkins and Azure DevOps.

Organizations seeking to adopt such automation are encouraged to consider the following qualities for Static Analysis within continuous integration scenarios:

- Time taken to process large volumes of code – changes need to be analyzed quickly
- Ability to run in a fast incremental mode against small changes only
- Ability to run against partial code changes, rather than the entire codebase
- Volume of output generated with each small (incremental) change
- Advanced workflow features for managing the issues identified
- Alerts when new problems occur in past analysis runs, and automatically generating items on the development team backlog when this occurs

Cybersecurity

ISO 26262:2018 specifically notes that cybersecurity may be considered during the development of embedded software. Synopsys strongly recommends that the topics of cybersecurity, quality, risk, and safety be considered in a unified approach. Many of the tools and methods highlighted in industry cybersecurity standards overlap with the tools and activities that organizations typically adopt for safety, quality, and reliability.

Although the scope of ISO 26262:2018 relates primarily to functional safety, Synopsys helps organizations carry out a range of security activities that may generate new safety hazards as defined in Clause 6.4.5 and Annex E of ISO 26262:2018 Part 2:

- Risk assessment / threat modeling (TARA) carried out during design activities (Appendix E.3.2)
- Static code analysis to identify coding standards and security vulnerabilities during development (Appendix E.3.3)
- Identification of robustness failures that may cause security vulnerabilities, via automated fuzzing (Appendix E.3.3)
- Automated and manual penetration testing at the unit, component, and system level during verification and validation phases (Appendix E.3.3)
- Identification of open source packages, and ongoing monitoring for new vulnerabilities in third-party open source software components (Appendix E.3.4)

Synopsys is also involved as a member in formulating the SAE J3061 and ISO 21434 standards, which define comprehensive strategies for automotive cybersecurity.

Distributed development – Augmenting software integrity requirements within the development interface agreement (DIA)

ISO 26262:2018 provides, through Part 8, a number of clauses to specify a development interface agreement (DIA) to facilitate end-to-end collaboration in a distributed development scenario.

Typically, Synopsys customers include in the DIA the requirements for tool usage, scope, and reporting. Some of these requirements include:

- Metrics reports (e.g. HIS metrics), generated from static code analysis
- Scope of coding standards (e.g. MISRA) application, enforcement, and reporting requirements
- Requirements to perform particular types of security testing (e.g. penetration testing, risk assessment/TARA)
- Number of test cases and time taken to run fuzz testing
- Declaration of open source components in a software bill of materials (SBOM)

These requirements, in turn, are passed down the supply chain to software suppliers, ensuring assurance activities are performed and execution data is reported when producing software deliverables.

Tool qualification

Coverity Static Analysis is certified by TÜV SÜD Product Service GmbH as meeting the requirements for support tools according to IEC 61508-3. The tool is qualified for use in safety-related software development according to ISO 26262, IEC 61508, EN 50128, and EN 50657. The tool is classified as T2, for use up to ASIL D in accordance with ISO 26262:2011-8.

The documentation pack for the Coverity distribution includes the necessary functional safety manual, which describes tool operation and failure modes – including the risk of misconfiguration, and of false positives and false negatives.

Validation of the software tool

For ASIL D development, teams that must perform tool validation according to ISO 26262 Part 8-11 (“Confidence in the use of software tools”) need to complete tool validation within their build environment. This helps ensure that safety-critical defects are not missed due to installation or configuration errors.

The Coverity Qualification Kit ensures that Coverity is configured and operating properly within the customer’s build environment. This self-test feature produces a tool qualification report describing the tests that were run and the results of those tests to validate that Coverity is properly configured. The qualification process is consistent with the recommendations of ISO 26262 Part 8-11.4.9.

Software modeling and coding guidelines (ISO 26262:2018, Part 6, Section 1)

As part of the initiation of the product development phase at the software level, ISO 26262 created a set of coding and modeling guidelines that are published in Table 1 of the Software Development Module. The Synopsys software integrity portfolio supports these guidelines in the following manner:

Table legend:

++ indicates highly recommended

+ indicates recommended

o indicates that the method has no recommendation for or against its usage for the identified ASIL

ISO 26262:2018, Part 6, Table 1: Topics to be covered by modeling and coding guidelines

Topic	Synopsys Software Integrity Portfolio Support	ASIL			
		A	B	C	D
Enforcement of low complexity (1a)	Coverity will analyze code and compute cyclomatic complexity and Halstead metrics. Policies can be defined according to HIS metrics to identify functions which exceed a defined threshold of complexity.	++	++	++	++
Use of language subsets (1b)	Coverity allows the enforcement of commonly used language subsets and coding standards – e.g. MISRA C/C++, AUTOSAR C++, CERT C/C++, and others. Custom coding rules can be authored for specific API or organizational coding standards as required using the Code XM extension framework.	++	++	++	++
Enforcement of strong typing (1c)	C and C++ are considered less strongly typed than other languages such as Java because of their support for implicit and explicit casting. Coverity will automatically find unsafe casting and flag the occurrence as a defect. Additional checks can be created through the Coverity Code XM framework. For example, if casting is disallowed, a custom checker could be created to create a defect for every cast operation.	++	++	++	++
Use of defensive implementation techniques (1d)	Coverity enforces defensive programming by highlighting as an error failure to check return value of any function; not just checking for null but verifying or testing returned value for possible error conditions. This is the 'CHECKED_RETURN' rule in the Coverity engine.	+	+	++	++
Use of established design principles (1e)	Black Duck can be utilized to verify that already known good components are utilized within the development, and alert in the case that disallowed or unknown components are found. Through the Coverity SDK, custom analysis rules can be created to test for specific violations of select design principles such as the use of global variables.	+	+	++	++
Use of unambiguous graphical representation (1f)	Not applicable	+	++	++	++
Use of style guides (1g)	Coverity includes a powerful framework for implementing custom coding policies, named Code XM. Users of Coverity can customize this tool to match their own style guides or engage Synopsys services to assist in developing custom rules where required.	+	++	++	++
Use of naming conventions (1h)	The Coverity Code XM extension framework can be used to create a custom check for naming convention violations.	++	++	++	++
Concurrency aspects (1i)	While coding standards such as MISRA will restrict the available concurrency functions available for use, Coverity includes a number of built-in checks specifically targeted at finding concurrency related errors including deadlocks, resource exhaustion, and inconsistent usage of locking and thread management routines.	+	+	+	+

Software unit design and implementation (ISO 26262:2018, Part 6, Section 8)

Once the architectural design is complete, the next stage in the ISO 26262 standard is software unit design and implementation.

The standard supplies numerous guidelines for software design and implementation to ensure the correct order of execution, consistency of interfaces, correctness of data flow and control flow, simplicity, readability and comprehensibility, and robustness.

During development, Synopsys enables developers to ensure that their code conforms to ISO 26262 design principles. This is accomplished primarily by implementing industry-specific coding standards rules such as MISRA C/C++.

ISO 26262:2012, Part 6, Table 6: Design principles for software unit design and implementation

Topic	Synopsys Software Integrity Portfolio Support	Rule Mapping	ASIL			
			A	B	C	D
One entry and one exit point in subprograms and functions (1a)	Coverity automatically analyzes return statements to determine if more than one entry or exit point exists in a component or function.	MISRA C 2004 Rules 14.4 and 14.7, MISRA C 2012 Rules 15.1 and 15.5	++	++	++	++
No dynamic objects or variables, or else online testing during their creation (1b)	Coverity automatically analyzes the code to identify if the use of dynamic objects are properly tested during their creation. For example, users can analyze the code to ensure that if malloc() is used, the return must be checked.	MISRA-C 2012 Directive 4.12	+	++	++	++
Initialization of variables (1c)	Coverity automatically tests the code for uninitialized variables.	MISRA C 2004 Rule 9.1, MISRA C 2012 Rules 9.1 and 9.4	++	++	++	++
No multiple use of variable names (1d)	Coverity automatically creates parse warnings that appear as actionable defects to the developer for such issues as local hiding local, local hiding parameter, and linkage conflict issues.	MISRA C 2004 Rule 5.5, MISRA C 2012 Rules 5.8 and 5.9	++	++	++	++
Avoid global variables or else justify their usage (1e)	In the context of the C and C++ languages, these issues would be addressed by coding standards checkers.	MISRA-C 2012 Rule 5.1 and 5.2, together with Rule 5.8 would effectively constrain this behavior.	+	+	++	++
Limited use of pointers (1f)	In the context of the C and C++ languages, these issues would be addressed by coding standards checkers.	MISRA-C 2012 Section 8.18 would limit the scope of risks originating from pointer usage.	+	++	++	++
No implicit type conversions (1g)	In the context of the C and C++ languages, these issues would be addressed by coding standards checkers.	MISRA-C 2012 Section 8.10 rules would identify violations of this requirement.	+	++	++	++
No hidden data flow or control flow (1h)	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 discrepancies.	+	++	++	++

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 formats. Where a communication protocol is proprietary, Defensics can be utilized as a configurable engine via the Defensics SDK to create custom protocol implementations.

ISO 26262:2018 Table 13: Test environments for conducting the software testing

Topic	Synopsys Software Integrity Portfolio Support	ASIL			
		A	B	C	D
Hardware-in-loop (1a)	Defensics is able to integrate with any third-party hardware-in-loop system via customized instrumentation scripting. The extensibility is extremely flexible and well documented. Synopsys is also able to provide services to integrate with common HIL test stands.	++	++	++	++
Electronic control unit network environments (1b)	Defensics is intended primarily for testing interfaces between software components and supports over 200 protocols for this purpose, including common bus protocols, such as CAN and CAN-FD.	++	++	++	++
Vehicles (1c)	Defensics is suitable for direct connection to vehicles in test environments and includes advanced features for bus diagnostics for this purpose.	+	+	++	++

ISO 26262:2018 Table 14: Methods for tests of the embedded software

Topic	Synopsys Software Integrity Portfolio Support	ASIL			
		A	B	C	D
Requirements-based test (1a)	Defensics data is based upon protocol standards and RFCs. For proprietary protocols, Defensics is extensible via an SDK.	++	++	++	++
Fault-injection test (1b)	Defensics is intended to manipulate extreme corner cases of protocol implementations.	+	+	+	++

ISO 26262:2018 Table 15: Methods for deriving test cases for software unit testing

Topic	Synopsys Software Integrity Portfolio Support	ASIL			
		A	B	C	D
Analysis of requirements (1a)	Defensics data is based upon protocol standards and RFCs. For proprietary protocols, Defensics is extensible via an SDK.	++	++	++	++
Generation and analysis of equivalence classes (1b)	Defensics generates test cases that are modeled on real protocol behavior and dialogs.	+	++	++	++
Analysis of boundary values (1c)	Defensics is intended to manipulate extreme corner cases of protocol implementations.	+	++	++	++
Error guessing based on knowledge or experience (1d)	Defensics' premise is that fuzz testing is known to trigger unexpected conditions within applications, including some well-known software security vulnerabilities.	+	+	++	++
Analysis of functional dependencies (1e)	Black Duck Binary Analysis from Synopsys can be used to determine what functionality is present in a particular software module by identifying dependent libraries (e.g. network servers, protocol parsers, and file handlers). This can then be utilized to formulate plans for fuzz testing.	+	+	++	++
Analysis of operational use cases (1f)	As part of a TARA/HARA activity, risks associated with external interfaces should be identified. This in turn would be used to formulate a plan for fuzz testing scope and requirements.	+	++	++	++

Resources

1. "The race for cybersecurity: Protecting the connected car in the era of new regulation," McKinsey & Company, Deichmann, Johannes; Klein, Benjamin; Scherf, Gundbert; Stuetzle, Rupert; October 10, 2019.
2. "Volkswagen CEO expects software to make up 90 percent of auto industry innovation," Reuters, March 12, 2019.
3. "Tech.View: Cars and software bugs," The Economist, May 16, 2010.
4. "Potentially deadly automotive software defects," Better Embedded System SW, September 25, 2018.
5. Ibid.

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