PARASOFT®

WHITEPAPER

Guide to ISO 26262 Software Compliance

Achieving Functional Safety in the Automotive Industry

INTRODUCTION

Some modern automobiles have more lines of code than a jet fighter. Even moderately sophisticated cars ship with larger and more complex codebases than the same line from just a few years ago. The inclusion of multi-featured infotainment systems, driver-assist technologies, and electronically controlled safety features as standard components—even in economy models—have fueled the growth of software in the automotive industry. Additionally, the emergence of driverless technology and "connected" cars that function as IoT systems on wheels will mean even larger and more complex codebases.

All of the innovation taking place in the automotive industry, though, raises concerns over the safety, security, and reliability of automotive electronic systems. The concerns are appropriate given that the automotive software supply chain is a long convoluted system of third-party providers spanning several tiers. Consider, for example, that software developed for a specific microcontroller unit (MCU) may be integrated by a third-tier provider into a component they're shipping to a second-tier provider and so on—until a composite component is delivered for final integration by the automaker.

While not all automotive software is critical to the safe operation of the vehicle, code that carries out functional safety operations must be safe, secure, and reliable. Organizations must implement strong software quality process controls around the development of safetycritical software in accordance with ISO 26262, which is a functional safety standard for automotive software. ISO 26262 provides guidance on processes associated with software development for electrical and/or electronic (E/E) systems in automobiles. The standard is aimed at reducing risks associated with software for safety functions to a tolerable level by providing feasible requirements and processes.

In this paper, we provide background information on ISO 26262 and its goals. We also discuss some of the policy-related issues associated with developing embedded software that complies with ISO 26262. Finally, we describe how Parasoft can help automotive software development organizations achieve compliance with ISO 26262.

WHAT ISO 26262 COVERS

ISO 26262 is a functional safety standard that covers the entire automotive product development process (including such activities as requirements specification, design, implementation, integration, verification, validation, and configuration). The standard provides guidance on automotive safety lifecycle activities by specifying the following requirements:

- Functional safety management for automotive applications
- The concept phase for automotive applications
- Product development at the system level for automotive applications software architectural design
- Product development at the hardware level for automotive applications software unit testing
- Product development at the software level for automotive applications
- Production, operation, service and decommissioning
- Supporting processes: interfaces within distributed developments, safety management requirements, change and configuration management, verification, documentation, use of software tools, qualification of software components, qualification of hardware components, and proven-in-use argument
- Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses

WHAT ISO 26262 DOES NOT COVER

- Unique E/E systems in special purpose vehicles such as vehicles designed for drivers with disabilities
- Safety standards for large vehicles, such as those over 3500KB (7700 pounds) gross weight
- Hazards related to electric shock, fire, smoke, heat, radiation, toxicity, flammability, reactivity, corrosion, release of energy and similar hazards, unless directly caused by malfunctioning behavior of E/E safety-related systems
- Nominal performance of E/E systems

SOFTWARE-SPECIFIC SECTIONS OF ISO 26262

Part 6 of the standard specifically addresses product development at the software level. Requirements for the following development activities are specified:

- Initialization of product development
- Specification of software safety requirements
- Software architectural design
- Unit design and implementation
- Unit testing
- Software integration and testing
- Verification of software safety requirements.

Methods defined by the ISO 26262 standard should be selected depending on the ASIL (automotive safety integrity level). The higher the ASIL, the more rigorous the methods.

Part 8, section 11, describes the software tool qualification process. Tools that automate software development activities and tasks can significantly help organizations meet ISO 26262 requirements. Software tool qualification is intended to provide evidence that tools are suitable for developing a safety-related item or element. One of the qualification methods defined in ISO 26262 relies on running the development tool on a control codebase and making sure that the product is consistent and accurate.

Qualifying Parasoft defect prevention tools and technologies involves running static analysis, flow analysis, unit tests, and any other testing practice used in your development process on a set of control code. Parasoft will consistently, accurately and objectively report errors, which ensures that the tool functions properly.

ISO 26262 COMPLIANCE AND POLICY-DRIVEN DEVELOPMENT

A particular feature that makes developing compliant embedded software so challenging is the gap between software development and business expectations. Software engineers make business-critical decisions every day in the form of their coding practices, quality activities, and engineering processes. As software permeates critical functions associated with functional safety, these engineering decisions can lead to significant business risks. E/E systems in automobiles that must conform to ISO 26262 are particularly vulnerable to risks because the standard specifies very detailed lifecycle processes throughout the approximately 400 pages intended to answer a simple, yet ambiguous, question: Is this safe?

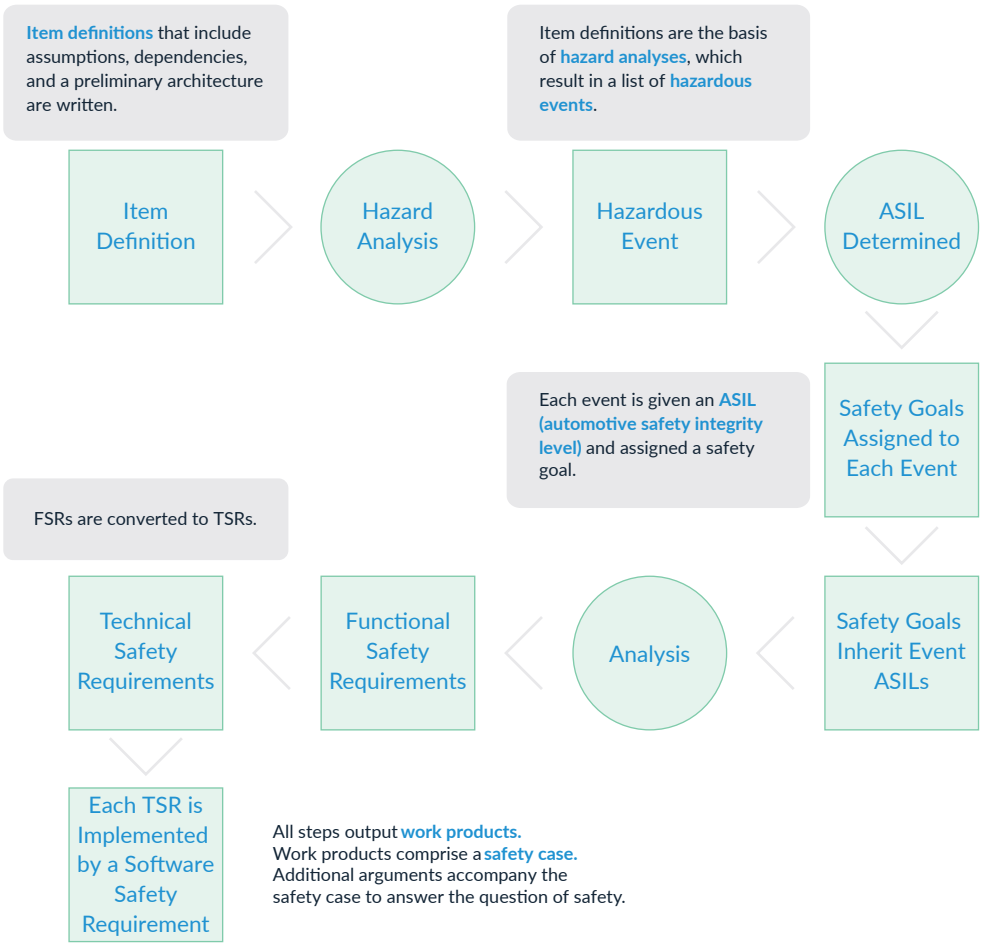

Figure 1: Software development lifecycle defined by ISO 26262

The purpose of ISO 26262 is to outline the policy surrounding the processes in Figure 1, but policies specific to the organization can be integrated at any step.

The key to reining in these risks is to align software development activities with your organization's business goals. This can be achieved through policy-driven development, which ensures that engineers deliver software according to your expectations. Policy-driven development involves:

- Clearly defining expectations and documenting them in understandable polices.
- Training the engineers on the business objectives driving those policies.
- Enforcing the policies in an automated, unobtrusive way.

By adopting a policy-driven strategy, businesses are able to accurate and objectively measure productivity and application quality, which lowers development costs and reduces risk.

With public safety, potential litigation, market position and other consequences on the line, it behooves software development teams and people in the traditional business management positions to come together on policy and implement the strategy into their software development lifecycle.

TRY PARASOFT DTP

Learn more about policy-driven development.

PARASOFT SUPPORT FOR ISO 26262

Parasoft DTP facilitates the software quality tasks specified in ISO 26262, including static analysis, data flow static analysis, metrics analysis, peer code review, unit testing and runtime error detection. This provides teams a practical way to prevent, expose, and correct errors in automotive functional safety systems. DTP collects data generated by software engineering processes, such as static code analysis violations, test results, code metrics, coverage analysis, source control check-ins, defect tracking systems, etc., and generates meaningful views of the correlated and prioritized data.

The real power of DTP is the Parasoft Process Intelligence Engine (PIE), which performs an additional post analysis on the development artifacts collected in order to pinpoint risk in the code while highlighting opportunities for improving the your development processes. DTP reports the problematic code and a description of how to fix it to the engineer's IDE based on the organization's programming policy.

The specific sections of the ISO 26262, part 6: Product development: software level that can be addressed or partially addressed with Parasoft are described below. The information presented here is intended to serve as an introduction to ISO 26262 software verification and validation processes with Parasoft. Please refer to the standard and consult functional safety experts for clarification of any requirements defined by the ISO 26262 standard.

INITIALIZATION OF PRODUCT DEVELOPMENT AT THE SOFTWARE LEVEL

This section of the ISO 26262 Part 6 standard defines general information about the process of software development and validation.

5.4.6 Requirements for achieving correctness of software design and implementation. Methods described here apply to both modeling and programming languages.

REQUIREMENT	PARASOFT CAPABILITY
Enforcement of low complexity	» Reports cyclomatic complexity, essential complexity, Halsted complexity, and other code metrics
Use of language subsets	» Coding standards enforcement, e.g., detection of unsafe language constructions
Enforcement of strong typing	» Implicit conversions detection
Use of defensive implementation techniques	» Enforces defensive programming against appropriate coding standards rules, e.g., checking the return value of malloc, checking the error code value returned by called functions, etc.
Use of established design principles	» Enforcement of industry coding standards rule sets, e.g. MISRA C/C++, JSF, HIS source code metrics, etc.
Use of unambiguous graphical representation	» Enforcement of specific formatting conventions
Use of style guides	» Enforcement of specific coding conventions
Use of naming conventions	» Enforcement of specific naming conventions

SOFTWARE UNIT DESIGN AND IMPLEMENTATION

This section defines the process of specifying and implementing software units, as well as the verification of the design and implementation.

8.4.4 Specifies the design principles for software unit design and implementation.

REQUIREMENT	PARASOFT CAPABILITY
Design principles for software unit implementation, e.g. initialization of variables, No implicit type conversions, etc.	Static analysis: » MISRA C rules » MISRA C++ rules » MISRA C 2023 » MISRA C++ 2023 » Additional standards

Please refer to the Satisfying ASIL Requirements with Parasoft C/C++test paper for additional information about C/C++test support for specific software unit implementation design principles.

8.4.5 Specifies the verification methods for checking software unit design and implementation.

REQUIREMENT	PARASOFT CAPABILITY
Control flow analysis	» Control Flow Analysis
Data flow analysis	» Data Flow Analysis
Static code analysis	» Coding standards enforcement
Inspection of the source code	» DTP Change Explorer
Walkthrough of the source code	» DTP Change Explorer

SOFTWARE UNIT TESTING

This section defines the process of planning, defining, and executing software unit testing.

9.4.1 Describes general information about unit test execution.

REQUIREMENT	PARASOFT CAPABILITY
Unit test execution	» Unit test execution module » Reports module for presenting results
Unit test specification	» Configurable unit test generation module creates tests according to the defined specification. » Test Case Explorer module presents a list of all defined test cases with pass/fail status.

9.4.2 Describes methods used to specify and execute unit tests.

REQUIREMENT	PARASOFT CAPABILITY
Requirement-based tests	» Bidirectional traceability of test and requirements » Requirements testing coverage reports
Unit test specification	» Maps test cases with requirements and/or defects in conjunction with the DTP. » Supports user defined test cases created manually and tests created with the Test Case Editor.
Interface tests	» Uses function stubs and data sources to emulate behavior of external components for automatic unit test execution.
Fault injection tests	» Enforcing fault conditions using function stubs. » Automatic unit test generation using different set of preconditions (min, max, heuristic values).

Please note that Parasoft allows for packaging test cases into groups to allow easier management of the tests (such as execution of the tests from a single group only).

9.4.3 Defines methods that should be used to create test cases.

REQUIREMENT	PARASOFT CAPABILITY
Analysis of requirements	» Parasoft DTP provides requirements to code and requirements to test traceability
Generation and analysis of equivalence classes	» Uses factory functions to prepare sets of input parameter values for automated unit test generation » Uses data sources to efficiently use a wide range of input values in tests
Analysis of boundary values	» Automatically generates test cases (such as heuristic values, boundary values). » Employs data sources to use a wide range of input values in tests.
Error guessing	» Uses the function stubs mechanism to inject fault conditions into tested code. » Flow Analysis results can be used to write additional tests.

9.4.4 Defines the methods for demonstrating the completeness of the test cases.

REQUIREMENT	PARASOFT CAPABILITY
Statement coverage	» Code Coverage module
Branch coverage	» Code Coverage module
MC/DC (modified condition/decision coverage)	» Code Coverage module

Note that ISO 26262 Part 6, Point 9.4.4 states that if instrumented code is used to determine the degree of coverage, it may be necessary to show that the instrumentation has no effect on the test results. This is achieved by running the tests on non-instrumented code.

9.4.5 Defines the requirements for the test environment.

REQUIREMENT

Test environment for unit testing shall correspond as far as possible to the target environment.

PARASOFT CAPABILITY

- Unit test execution on both target device and simulator to perform tests in different environments (like software in the loop, processor in the loop, hardware in the loop).

SOFTWARE INTEGRATION AND TESTING

10.4.2 Describes general information about executing software integration tests.

REQUIREMENT

Integration tests

PARASOFT CAPABILITY

- Flexible configuration of tested software scope (from single function to entire application)
- Multi-metric test coverage analysis

10.4.5 Defines methods for demonstrating completeness of integration testing.

REQUIREMENT

Function Coverage

PARASOFT CAPABILITY

- Code Coverage module

10.4.7 Defines requirements for the integration test environment.

REQUIREMENT

Test environment for software integration testing shall correspond as far as possible to the target environment.

PARASOFT CAPABILITY

- Flexible stub framework.
- Service virtualization module is available to thoroughly mimic complete system.
- Coverage analysis execution on both target device and simulator to perform tests in different environments (like software in the loop, processor in the loop, hardware in the loop).

SUMMARY

Developing ISO 26262 compliant software for E/E systems in automobiles is no easy feat. But Parasoft eases the burden by offering a broad range of analysis tools and, more importantly, enabling you to automatically monitor compliance with your development policy—bridging the gap between development activities and business processes. Development teams can also generate configurable test reports that contain a high level of detail, which helps facilitate the work required for the software verification process.

TAKE THE NEXT STEP

Talk to an expert about accelerating the delivery of high-quality and compliant software with our ISO 26262 compliance tools.

ABOUT PARASOFT

Parasoft helps organizations continuously deliver quality software with its market-proven, integrated suite of automated software testing tools. Supporting the embedded, enterprise, and IoT markets, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award winning reporting and analytics dashboard delivers a centralized view of quality enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives — security, safety-critical, Agile, DevOps, and continuous testing.