

WHITEPAPER

Verifying ISO 26262 Software Using CT

Document Information

Document

Distribution date June 20, 2024

Technical Support

Email help@suresofttech.com

Phone +82-31-606-2000

Legal Notice

© 2024. Suresofttech Inc. All rights reserved.

Proprietary and Confidential This material is proprietary to Suresofttech Inc. It contains trade secret and confidential information which is solely the property of Suresofttech Inc. This material is for client's internal use only. This material shall not be used, reproduced, copied, disclosed, and transmitted, in whole or in part, without the express consent of Suresofttech Inc.

Reproduction and distribution without permission from Suresofttech Inc. is prohibited.

For suggestions or inquiries regarding this document or product, please contact Suresofttech Inc.

Some images included in this document may differ slightly depending on the version. The image corresponding to the function being described is identical to the version specified in the document information.

Table of Contents

1	Introduction	4
2	Overview of ISO 26262 Standard	5
3	Introduction to CT	7
	Test Design	7
	Achieving Coverage Objectives	7
	Other Features	8
4	How CT Supports ISO 26262	9
	5.4.3 - Topics to be covered by modelling and coding guidelines	9
	7.4.3 - Principles for software architectural design	10
	7.4.12 - Safety mechanisms for error detection	10
	7.4.14 - Software architectural design verification methods	11
	9.4.2 - Software unit verification	12
	9.4.3 – Deriving test cases for software unit testing	15
	9.4.4 – Structural coverage metrics at the software unit level	15
	10.4.2 – Verification of software integration	16
	10.4.3 - Deriving test cases for software integration testing	17
	10.4.5 – Structural coverage at the software architectural level	18
	11.4.1 – Test environments for conducting the software testing	18
	11.4.3– Deriving test cases for the test of the embedded software	19
5	Conclusion	20

1 Introduction

The automotive industry has been experiencing rapid technological advancements in recent years. Automobiles are no longer simple mechanical devices but have evolved into complex systems consisting of hundreds of electronic control units (ECUs) and millions of lines of code. While these changes have greatly improved vehicle performance and convenience, they have also increased the potential risks associated with software errors. Software defects can cause vehicle malfunctions or accidents and, in severe cases, can lead to casualties.

Therefore, thorough verification of automotive software is necessary to provide safe and reliable vehicles. Software verification should be performed continuously throughout the entire development process, from the initial design stage to the final production stage, to detect and eliminate potential defects early on. Establishing an effective software verification system is crucial for ensuring vehicle safety and gaining consumer trust.

ISO 26262 is an international standard for ensuring the safety of automotive software. It is a standard that addresses the functional safety of automotive electrical/electronic systems and provides specific requirements and procedures for securing the safety of in-vehicle electronic control units and software. This standard applies throughout the entire lifecycle of the system and provides guidelines necessary for software verification.

This whitepaper discusses how CT, a software unit and integration testing automation tool developed by Suresofttech, supports the ISO 26262 standard. CT has been developed to support systematic testing of automotive software, enabling compliance with the safety criteria set forth by ISO 26262. We will explain in detail how CT can be used to satisfy the ISO 26262 standard.

2 Overview of ISO 26262 Standard

ISO 26262 is an international standard for ensuring the functional safety of automotive electrical/electronic systems. This standard defines the requirements and procedures for securing the safety of in-vehicle electronic control units (ECUs) and software. It applies throughout the entire lifecycle of the system, including system design, development, production, operation, maintenance, and decommissioning, with the goal of achieving functional safety of automotive electrical/electronic systems and preventing casualties and property loss caused by malfunctions.

ISO 26262 consists of a total of 12 parts, each addressing various activities and requirements necessary to achieve functional safety.

- Part 1: Vocabulary: Provides key terms and definitions used in the standard.
- Part 2: Management of functional safety: Defines policies, procedures, organizational structures, etc., for managing functional safety.
- Part 3: Concept phase: Defines safety goals and conducts risk analysis in the initial stage of the system.
- Part 4: Product development at the system level: Defines safety requirements in the system design and development process and designs the system architecture.
- Part 5: Product development at the hardware level: Defines safety requirements in the hardware design and development process.
- Part 6: Product development at the software level: Defines safety requirements in the software design and development process.
- Part 7: Production, operation, service and decommissioning: Defines safety activities in the production, operation, service, and decommissioning stages of the product.
- Part 8: Supporting processes: Defines processes that support safety management activities such as configuration management, change management, and documentation.
- Part 9: ASIL (Automotive Safety Integrity Level)-oriented and safety-oriented analyses: Defines ASIL determination methods and safety analysis methods. This applies safety requirements differently according to the severity and probability of risks.
- Part 10: Guidelines on ISO 26262: Provides guidelines and examples for applying the standard.
- Part 11: Guidelines on application of ISO 26262 to semiconductors: Provides guidelines for applying the standard to semiconductor devices.
- Part 12: Adaptation of ISO 26262 for motorcycles: Defines methods for adjusting and applying the standard for specific vehicle types such as motorcycles.

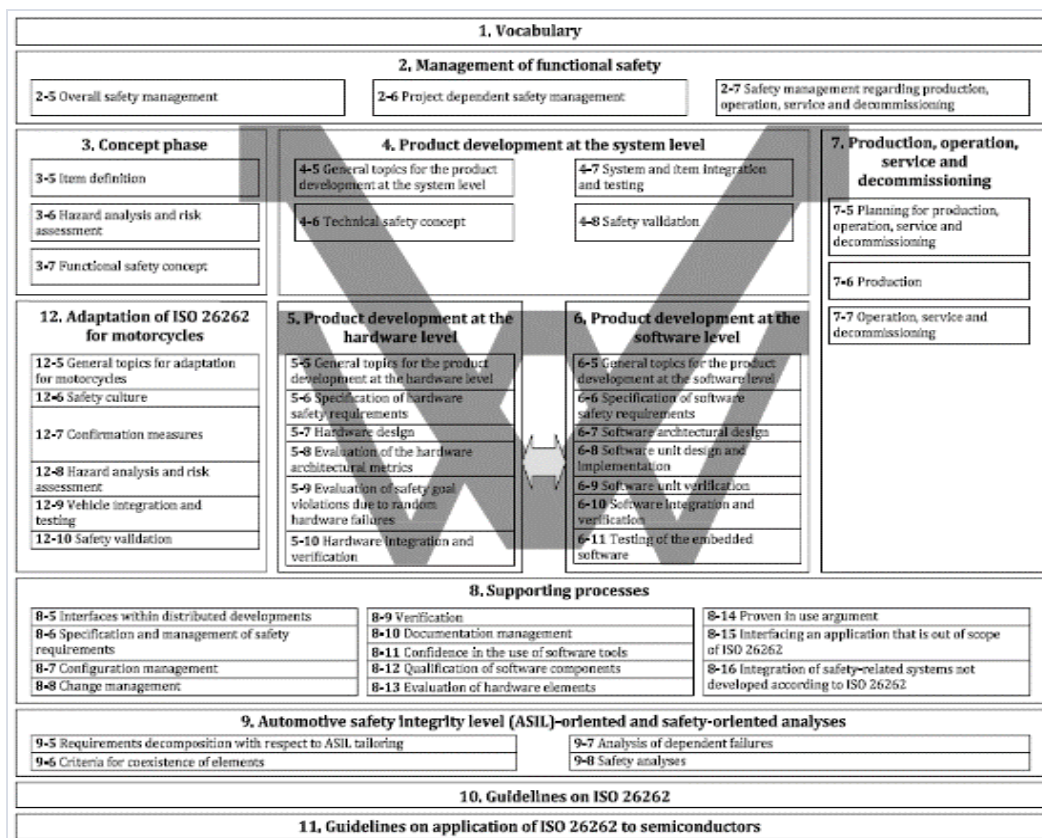


Figure 1 Overview of the ISO 26262 series of standards

One of the core concepts of ISO 26262 is ASIL (Automotive Safety Integrity Level). ASIL is a level that determines the functional safety requirements of a system, classified into four levels (A, B, C, D) based on the severity, frequency of occurrence, and controllability of potential risks. A is the lowest level, and D is the highest level. The higher the ASIL level, the more stringent safety requirements are applied.

ISO 26262 provides the following benefits:

- **Setting safety goals:** Provides guidance during the development process by clearly defining safety goals for each system.
- **Minimizing risks:** Prevents accidents by systematically analyzing potential risks and preparing appropriate countermeasures accordingly.
- **Improving reliability:** Enhances product reliability through standardized procedures and verification.
- **Regulatory compliance:** Compliance with ISO 26262 helps meet legal requirements and regulations.

ISO 26262 has become an essential standard in the automotive industry and plays a crucial role in ensuring the safety and reliability of vehicle software.

3 Introduction to CT

CT is a C/C++ code-based unit and integration test automation tool developed by Suresofttech and has received ISO 26262 certification. This certification signifies that CT meets the stringent functional safety requirements demanded in mission-critical fields such as vehicle safety. CT supports performing tests for given requirements and helps achieve structural coverage objectives. The main features of CT are as follows.

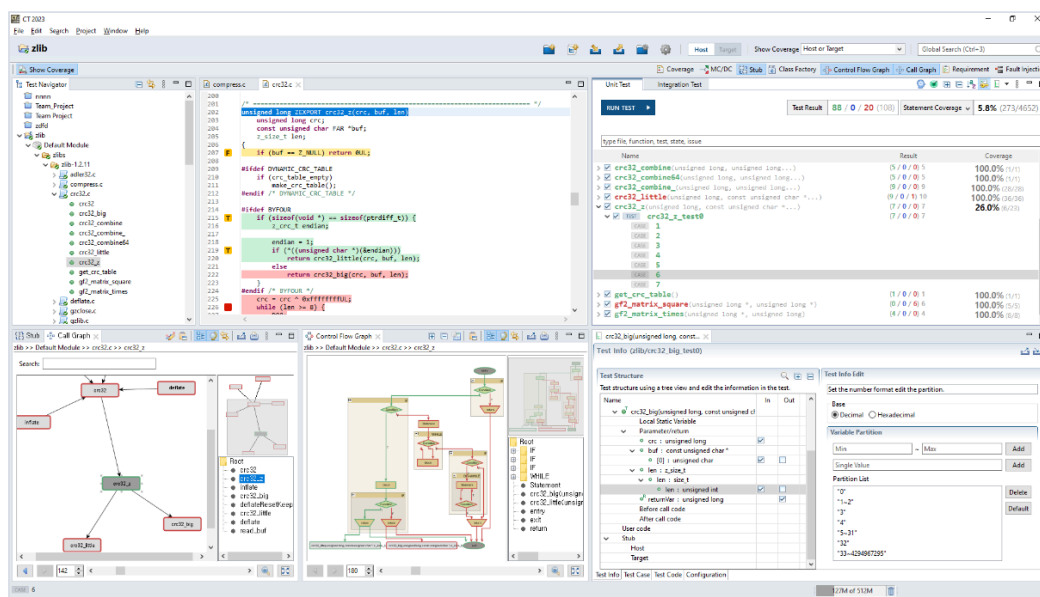


Figure 2 CT Screen

Test Design

CT automatically generates tests and test data and provides an intuitive test design interface. It can import test data in various formats and design integration tests with complex nested structures.

Achieving Coverage Objectives

CT supports statement, branch, MC/DC, function call, and function coverage and provides guidance for achieving coverage objectives. Coverage results can be viewed in the source code editor and control flow graph, and coverage results measured through system testing can be imported and merged.

Other Features

- Supports simulation and actual target environment testing
- Integration with DevOps tools
- Provides reports in various formats
- Provides requirement traceability by integrating with requirements management tools
- Team testing (improves productivity when multiple people verify the same software project)
 - Synchronizes test assets such as project settings, stubs, and mock objects to eliminate duplicate work
 - Allows real-time monitoring of executed test results on a web dashboard
- Provides the AI Assistant

CT is a tool optimized for automating various test activities required in the development process of automotive software. It has also obtained ISO 26262 certification and can be used as a test tool when developing software that satisfies this standard. In the following sections, we will examine in detail how CT satisfies the requirements of ISO 26262 for each major item, along with specific functions.

4 How CT Supports ISO 26262

We will explain in detail how CT supports ISO 26262. Specifically, this pertains to the contents of Part 6, Product development at the software level, of ISO 26262. The functions of CT corresponding to ISO 26262 are as follows.

5.4.3 - Topics to be covered by modelling and coding guidelines

This section specifies that criteria not sufficiently covered by the language itself in the selection of appropriate modelling, design, and programming languages should be addressed in the corresponding guidelines or development environment.

Table 1. Topics to be covered by modelling and coding guidelines

Topics		ASIL				CT Support
		A	B	C	D	
1a	Enforcement of low complexity	++	++	++	++	●

Source: [ISO 26262-6:2018] Table 1

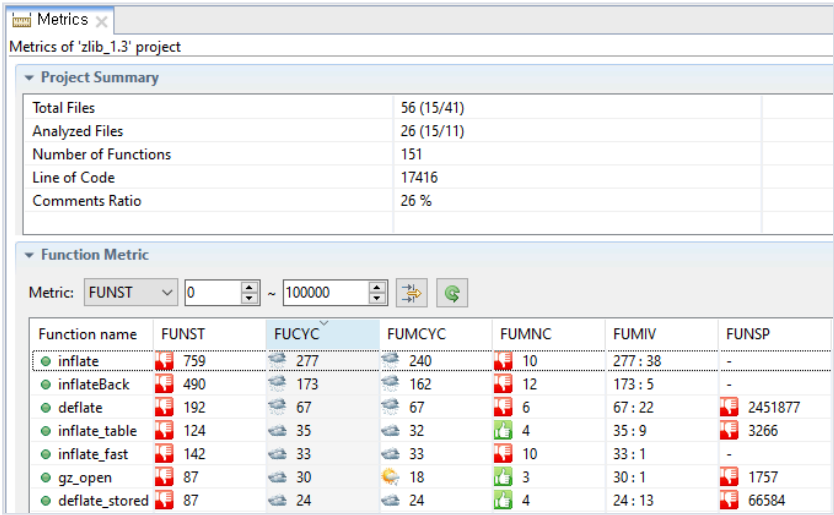


Figure 3 Metrics View

CT provides various software metrics in addition to Complexity through static analysis, helping to verify compliance with guidelines during the software development process.

7.4.3 - Principles for software architectural design

This section presents principles to be considered when designing software architecture. By adhering to these principles, the safety, reliability, and maintainability of the software can be improved.

Table 2. Principles for software architectural design

Principles		ASIL				CT Support
		A	B	C	D	
1e	Loose coupling between software components	++	++	++	++	

Source: [ISO 26262-6:2018] Table 3

CT helps identify coupling between components by analyzing data and control flow between software components through static analysis. This enables development organizations to verify compliance with the loose coupling principle of software architecture and improve the design if necessary.

7.4.12 - Safety mechanisms for error detection

This section specifies that safety mechanisms for error detection and error handling should be applied at the software architecture level based on the results of safety-oriented analysis. Some of the safety mechanisms mentioned here can be tested using CT's functions.

Among the safety mechanisms for error detection, the items supported by CT are as follows:

- Range checks of input and output data
 - After generating tests in CT, the range of input and output data can be checked in the Test Info tab. Additionally, automatically generated inputs, outputs, and expected values can be checked and directly modified in the Test Case tab.
- Plausibility check
 - CT supports requirements-based testing. By importing requirement information and linking it with test cases, it can verify whether the software's behavior conforms to the requirements.
- Detection of data errors
 - Using CT's fault injection or stub features, data errors can be intentionally injected, and the software's response to them can be tested.

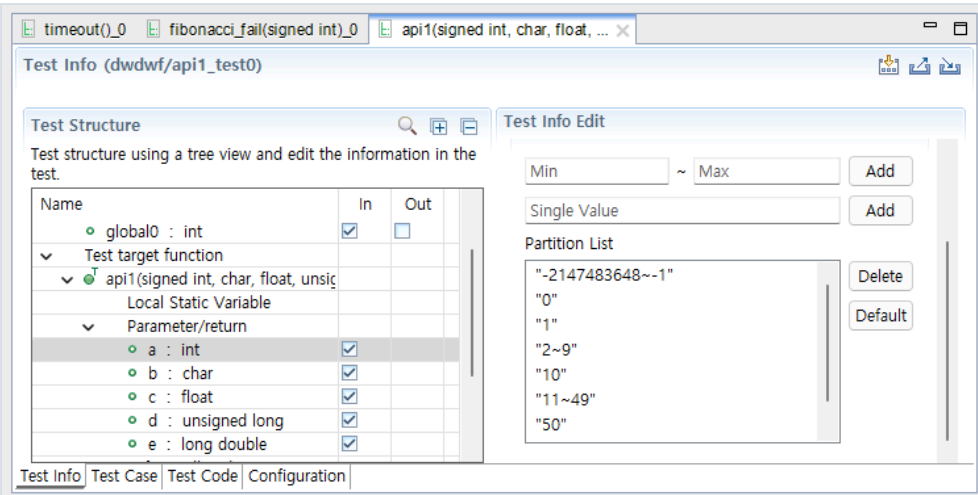


Figure 4 Data Range Verification (Test Info Tab)

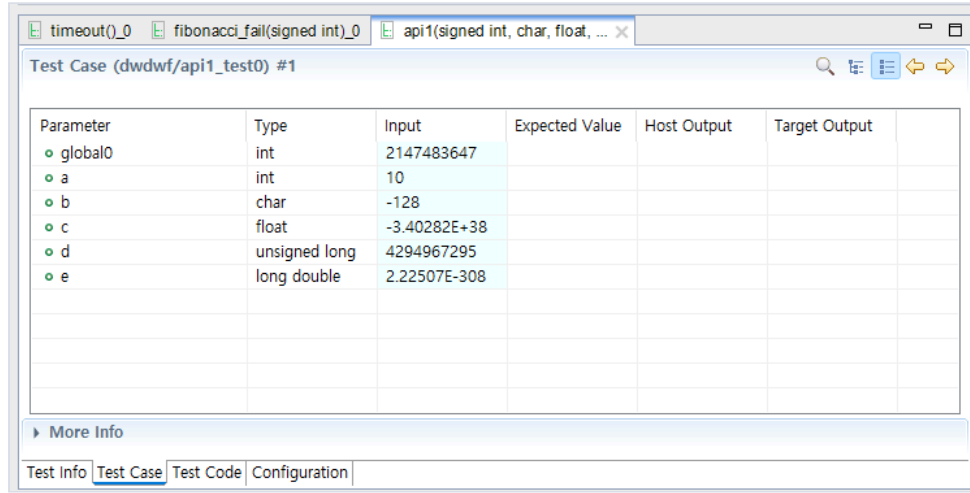


Figure 5 Editing Input and Expected Values (Test Case Tab)

7.4.14 - Software architectural design verification methods

This section specifies methods for verifying software architectural design.

Table 3. Methods for the verification of the software architectural design

Methods		ASIL				CT Support
		A	B	C	D	
1f	Control flow analysis	+	+	++	++	●
1g	Data flow analysis	+	+	++	++	●

Source: [ISO 26262-6:2018] Table 4

CT identifies control and data flow by statically analyzing the entire program. Control flow information can be viewed in the control flow graph view below.

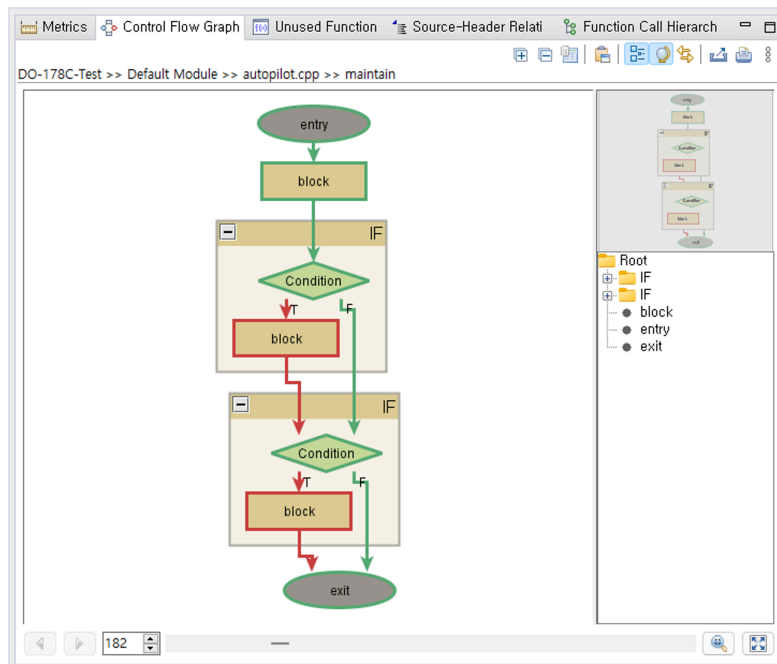


Figure 6 Control Flow Graph View

9.4.2 - Software unit verification

According to Clause 9 of ISO 26262-8:2018, the software unit design and implemented software units shall be verified using an appropriate combination of the methods presented in Table 7.

Table 4. Methods for software unit verification

Methods		ASIL				CT Support
		A	B	C	D	
1f	Control flow analysis	+	+	++	++	●
1g	Data flow analysis	+	+	++	++	●
1j	Requirements-based test	++	++	++	++	●
1k	Interface test	++	++	++	++	●
1l	Fault injection test	+	+	+	++	●
1n	Back-to-back comparison test between model and code, if applicable	+	+	++	++	●

Source: [ISO 26262-6:2018] Table 7

- 1f: Control flow analysis
 - CT performs static analysis to analyze the control flow of the entire program.
- 1g: Data flow analysis
 - CT performs static analysis to analyze the data flow of the entire program.
- 1j: Requirements-based test
 - CT supports requirements-based testing. It can import requirements from requirements management tools and link them to tests to ensure traceability.

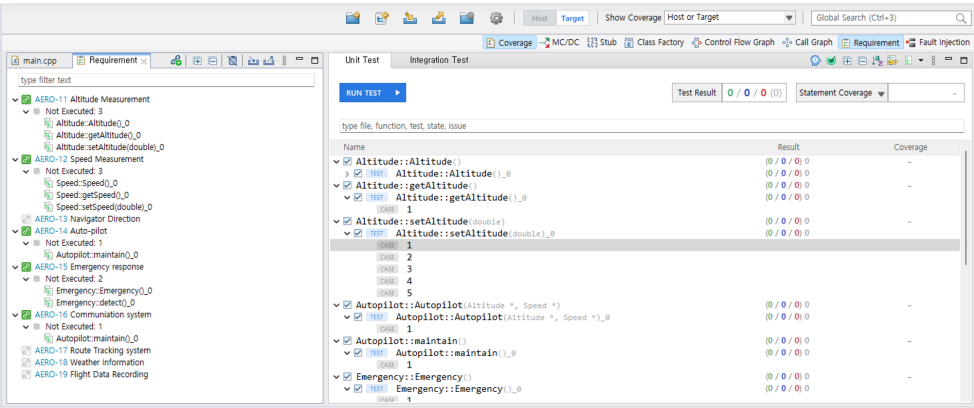


Figure 7 Linking Requirements and Tests

- 1k: Interface test
 - CT provides integration testing, stubbing, and fault injection features to verify that the interfaces between software units function correctly.

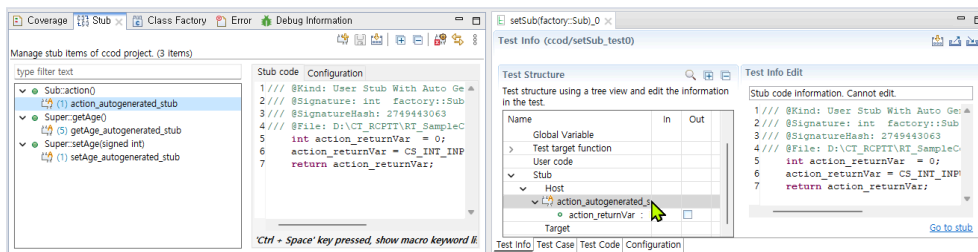


Figure 8 Stub View and Test View

- 1l: Fault injection test
 - CT supports fault injection tests. You can input the code to be inserted before or after a selected line in the fault injection view.

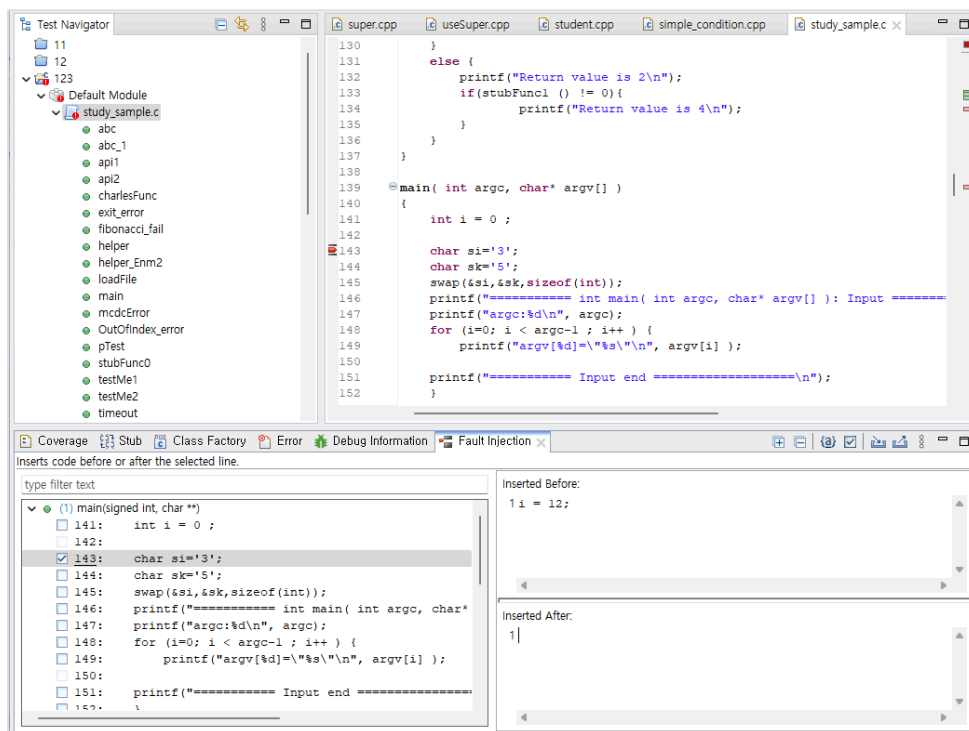


Figure 9 Fault Injection View

- 1n: Back-to-back comparison test between model and code, if applicable
 - CT can import test case information exported from modeling tools.

9.4.3 – Deriving test cases for software unit testing

To meet the specifications for appropriate test cases for software unit testing according to the preceding section 9.4.2, the methods listed in Table 8 below should be used.

Table 5. Methods for deriving test cases for software unit testing

Methods		ASIL				CT Support
		A	B	C	D	
1a	Analysis of requirements	++	++	++	++	●
1b	Generation and analysis of equivalence classes	+	++	++	++	●
1c	Analysis of boundary values	+	++	++	++	●
1d	Error guessing based on knowledge or experience	+	+	+	+	●

Source: [ISO 26262-6:2018] Table 8

- 1a: Analysis of requirements
 - CT allows you to view requirements in the requirements view and review the linked tests.
- 1b: Generation and analysis of equivalence classes
 - Equivalence classes are groups of test input data categorized according to specific criteria. CT partitions input data and automatically generates test data.
- 1c: Analysis of boundary values
 - CT analyzes the boundary values of input variables and generates test data.
- 1d: Error guessing based on knowledge or experience
 - CT provides various test design methods to anticipate and test potential software errors using experience and knowledge.

9.4.4 – Structural coverage metrics at the software unit level

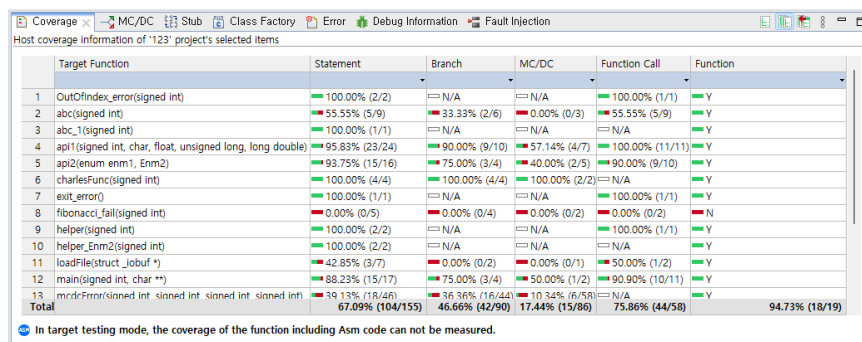
For sufficient verification of software unit tests, the completeness of test cases must be evaluated, and it must be demonstrated that unit test objectives have been appropriately achieved. The content of Table 9 below is the structural coverage metrics required to demonstrate the achievement of objectives.

Table 6. Structural coverage metrics at the software unit level

Methods		ASIL				CT Support
		A	B	C	D	
1a	Statement coverage	++	++	+	+	●
1b	Branch coverage	+	++	++	++	●
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++	●

Source: [ISO 26262-6:2018] Table 9

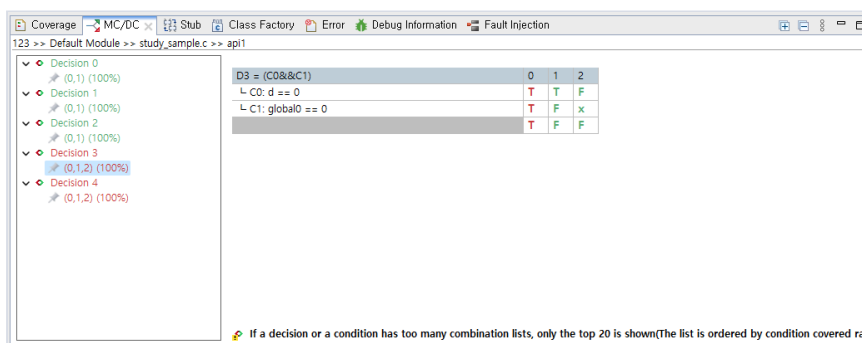
CT supports statement coverage, branch coverage, and MC/DC. Coverage results can be viewed in the Test View and Coverage View and exported as reports. The MC/DC View provides detailed information necessary for achieving MC/DC objectives.



Target Function	Statement	Branch	MC/DC	Function Call	Function
1 OutOfIndex_error(signed int)	100.00% (2/2)	N/A	N/A	100.00% (1/1)	Y
2 abc(signed int)	55.55% (5/9)	33.33% (2/6)	0.00% (0/3)	55.55% (5/9)	Y
3 abc_1(signed int)	100.00% (1/1)	N/A	N/A	N/A	Y
4 api1(signed int, char, float, unsigned long, long double)	95.83% (23/24)	90.00% (9/10)	57.14% (4/7)	100.00% (11/11)	Y
5 api2(enum enm1, Enm2)	93.75% (15/16)	75.00% (3/4)	40.00% (2/5)	90.00% (9/10)	Y
6 charlesFunc(signed int)	100.00% (4/4)	100.00% (4/4)	100.00% (2/2)	N/A	Y
7 exit_error()	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
8 fibonacci_fail(signed int)	0.00% (0/5)	0.00% (0/4)	0.00% (0/2)	0.00% (0/2)	N
9 helper(signed int)	100.00% (2/2)	N/A	N/A	100.00% (1/1)	Y
10 helper_Enm2(signed int)	100.00% (2/2)	N/A	N/A	N/A	Y
11 loadFile(struct_jobuf *)	42.85% (3/7)	0.00% (0/2)	0.00% (0/1)	50.00% (1/2)	Y
12 main(signed int, char **)	88.23% (15/17)	75.00% (3/4)	50.00% (1/2)	90.90% (10/11)	Y
13 mcdrError(signed int, signed int, signed int, signed int)	30.13% (18/46)	36.36% (16/44)	10.34% (6/58)	N/A	Y
Total	67.09% (104/155)	46.66% (42/90)	17.44% (15/86)	75.86% (44/58)	94.73% (18/19)

In target testing mode, the coverage of the function including Asm code can not be measured.

Figure 10 Coverage View



Decision	Condition	0	1	2
Decision 0	D3 = (C0&&C1)	T	T	F
Decision 1	L C0: d == 0	T	F	x
Decision 2	L C1: global0 == 0	T	F	x

If a decision or a condition has too many combination lists, only the top 20 is shown(The list is ordered by condition covered ra

Figure 11 MC/DC View

10.4.2 – Verification of software integration

According to Clause 9 of ISO 26262-8:2018, software integration shall be verified using an appropriate combination of the methods in Table 10.

Table 7. Methods for verification of software integration

Methods	ASIL				CT Support
	A	B	C	D	
1a Requirements-based test	++	++	++	++	●
1b Interface test	++	++	++	++	●
1c Fault injection test	+	+	++	++	●
1e Back-to-back comparison test between model and code, if applicable	+	+	++	++	●
1f Verification of the control flow and data flow	+	+	++	++	●

Source: [ISO 26262-6:2018] Table 10

- 1a: Requirements-based test
 - CT supports requirements-based testing. It can import requirements from requirements management tools and link them to tests to ensure traceability.
- 1b: Interface test
 - CT provides integration testing, stubbing, and fault injection features to verify that the interfaces between software units function correctly.
- 1c: Fault injection test
 - CT supports fault injection tests. You can input the code to be inserted before or after a selected line in the fault injection view.
- 1e: Back-to-back comparison test between model and code, if applicable
 - CT can import test case information exported from modeling tools.
- 1f: Verification of the control flow and data flow
 - CT performs static analysis of the entire program to identify control and data flows.

10.4.3 - Deriving test cases for software integration testing

While Clause 9.4.3 introduced test case derivation methods for unit testing, this section explains test case derivation methods for integration testing. CT supports test case derivation methods for integration testing in the same way as described in Clause 9.4.3.

Table 8. Methods for deriving test cases for software integration testing

Methods		ASIL				CT Support
		A	B	C	D	
1a	Analysis of requirements	++	++	++	++	●
1b	Generation and analysis of equivalence classes	+	++	++	++	●
1c	Analysis of boundary values	+	++	++	++	●
1d	Error guessing based on knowledge or experience	+	+	+	+	●

Source: [ISO 26262-6:2018] Table 11

10.4.5 – Structural coverage at the software architectural level

Structural coverage evaluation at the software architecture level is performed to assess the completeness of test cases for integration testing and to demonstrate that integration testing objectives have been appropriately achieved. To this end, structural coverage should be evaluated according to the methods listed in Table 12.

Table 9. Structural coverage at the software architectural level

Methods		ASIL				CT Support
		A	B	C	D	
1a	Function coverage	+	+	++	++	●
1b	Call coverage	+	+	++	++	●

Source: [ISO 26262-6:2018] Table 12

CT supports both function coverage and call coverage. Coverage results can be viewed in the Test View and Coverage View and exported as reports.

11.4.1 – Test environments for conducting the software testing

To verify that the embedded software meets the software safety requirements in the target environment, testing shall be performed in the appropriate test environments listed in Table 13, according to Clause 9 of ISO 26262-8:2018.

Table 10. Test environments for conducting the software testing

Methods		ASIL				CT Support
		A	B	C	D	
1a	Hardware-in-the-loop	++	++	++	++	●
1b	Electronic control unit network environments	++	++	++	++	●

Source: [ISO 26262-6:2018] Table 13

CT supports the test environments specified in Table 13 as follows:

- Hardware-in-the-loop (HIL): CT supports HIL test environments where embedded software can be tested using actual ECU hardware.
- Electronic control unit network environments: CT supports testing in environments that integrate some or all of the electrical systems in a vehicle, such as test benches, lab cars, or mule vehicles.

11.4.3– Deriving test cases for the test of the embedded software

While Clause 9.4.3 introduced test case methods for unit testing and Clause 10.4.3 for integration testing, Clause 11.4.3 introduces test case derivation methods for embedded software in Table 15. The test case derivation methods for embedded software are the same as the functions of CT introduced in the previous two sections.

Table 11. Methods for deriving test cases for the test of the embedded software

Methods		ASIL				CT Support
		A	B	C	D	
1a	Analysis of requirements	++	++	++	++	●
1b	Generation and analysis of equivalence classes	+	++	++	++	●
1c	Analysis of boundary values	+	+	++	++	●
1d	Error guessing based on knowledge or experience	+	+	++	++	●

Source: [ISO 26262-6:2018] Table 15

5 Conclusion

In this whitepaper, we have examined in detail how Suresofttech's CT tool supports the ISO 26262 standard. CT provides the following key functions necessary for compliance with the ISO 26262 standard:

1. Support for Software Unit Verification

CT supports the software unit verification methods required in Clause 9.4.2 of ISO 26262-6:2018. It can effectively verify compliance with functional and safety requirements of software units through various techniques such as static analysis, dynamic analysis, and requirements-based testing.

2. Support for Software Integration Verification

In accordance with Clause 10.4.2 of ISO 26262-6:2018, CT supports various test methods necessary for software integration verification. It can effectively detect errors at the integration stage through interface testing between integrated software components, fault injection testing, and verification of control and data flow.

3. Support for Structural Coverage Analysis

CT supports the structural coverage analysis required in Clauses 9.4.4 and 10.4.5 of ISO 26262-6:2018. It provides various coverage metrics such as statement coverage, branch coverage, and MC/DC at the software unit and architecture levels, enabling the evaluation and verification of test completeness.

4. Support for Embedded Software Test Environments

Among the embedded software test environments required in Clause 11.4.1 of ISO 26262-6:2018, CT supports testing in HIL (Hardware-in-the-loop) environments and electronic control unit network environments. This allows verification of the functionality and safety of embedded software under conditions close to the actual execution environment.

5. Support for Traceability and Documentation

CT efficiently supports traceability and documentation work, which is one of the overall requirements of ISO 26262. It automatically manages traceability between requirements, test cases, and test results, and relevant documents can be easily created through the report generation function.

In conclusion, CT is a tool that comprehensively supports various software verification activities and the generation of related artifacts required by ISO 26262. It directly supports core requirements of the standard such as testing at the software unit and integration stages, structural coverage analysis, and execution environment testing. In addition, it greatly reduces the workload required for standard compliance through traceability and documentation functions.

Considering these advantages, the utilization of CT can be considered essential in the software development and verification process for ensuring automotive functional safety. In the future, CT is expected to firmly support the safe and efficient development of software in the automotive industry through continuous technology advancement and standard compliance.