

WHITEPAPER

# Using CT for DO-178C Software Verification

---

# Document Information

---

## Document

**Distribution date**      December 20, 2023

## Technical Support

**Email**                      help@suresofttech.com  
**Phone**                      +82-31-606-2000

### Legal Notice

© 2023. 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 | Overview                                                                      | 4  |
| 2 | What is DO-178C                                                               | 5  |
| 3 | What is CT                                                                    | 7  |
|   | Test design                                                                   | 7  |
|   | Achieve coverage goals                                                        | 7  |
|   | Other Features                                                                | 7  |
| 4 | How CT Supports DO-178C                                                       | 9  |
|   | Section 6.3.4 – Reviews and Analyses of the Source Code                       | 9  |
|   | Section 6.4.1 – Test Environment                                              | 10 |
|   | Section 6.4.2 – Requirements-based Test Selection                             | 11 |
|   | Section 6.4.2.1 & 6.4.2.2 – Normal & Robustness Range Test Cases              | 11 |
|   | Section 6.4.3 – Requirements-Based Testing Methods                            | 12 |
|   | Section 6.4.4 – Test Coverage Analysis                                        | 13 |
|   | Section 6.4.4.1 & 6.4.4.2 – Requirements-Based & Structural Coverage Analysis | 13 |
|   | Section 6.4.5 – Reviews and Analyses of Test Cases, Procedures, results       | 14 |
|   | Section 12.2 – Tool Qualification                                             | 15 |
| 5 | Conclusion                                                                    | 16 |

# 1 Overview

---

The role of software in the modern aviation industry is very important, and it is directly related to the safety of aircraft. The aviation industry operates under particularly high safety standards and stringent regulatory environments, where software plays an essential role in controlling the core functions of aircraft and ensuring operational stability. In this context, the DO-178C standard is positioned as a key guideline in the aviation software development and certification process.

DO-178C is an internationally recognized guideline to ensure the safety and reliability of software systems on board aircraft. This standard regulates the development and certification process throughout the software life cycle and ensures that software related to aircraft safety undergoes rigorous quality control and verification processes. DO-178C provides standards for achieving an appropriate level of safety required at each stage of software development, making it an important guideline for aircraft manufacturers and software developers.

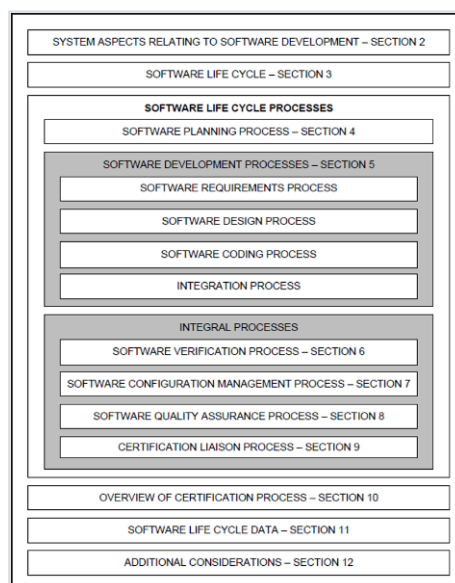
The requirements for tools used in the DO-178C certification process are different from other standards. Typically, standards such as ISO-26262, IEC-61508, and IEC-62304 allow the use of tools that have been certified by an independent body. However, under DO-178C, this practice does not apply. DO-178C does not allow certification for tool use, and each tool must undergo a process to verify its qualifications in the context of the specific project. (See DO-178C Chapter 12.2)

This white paper covers how CT, a software unit and integration test automation tool developed by Suresofttech, supports the DO-178C standard. CT is designed to meet the complex and rigorous requirements of aviation software development and certification. Here we will specifically explain how to meet the specific criteria of DO-178C.

## 2 What is DO-178C

DO-178C, "Software Considerations in Airborne Systems and Equipment Certification" is an international standard for the development and maintenance of aviation industry software. This standard provides guidelines for the development and life cycle management of software that has a significant impact on aviation safety. DO-178C was released in 2012, replacing its predecessor, DO-178B. DO-178C specifies methods to manage software quality and minimize software defects through clear requirements, detailed design, thorough testing, and appropriate quality assurance processes.

DO-178C describes the development and maintenance process of aircraft software by dividing it into three life cycles. These are the planning process life cycle, development process life cycle, and integration process life cycle. The planning process life cycle contains development plan contents, the development process life cycle contains requirements, design, coding, and integration contents, and finally, the integrated process life cycle contains verification, configuration management, quality assurance, and certification process contents.



**Figure 1** DO-178C structure overview

DO-178C also provides reliability level distinctions for determining the reliability level of software. This is based on the potential impact of a software defect on aircraft operations. This division is divided into five levels, from Level A (highest reliability) to Level E (lowest reliability).

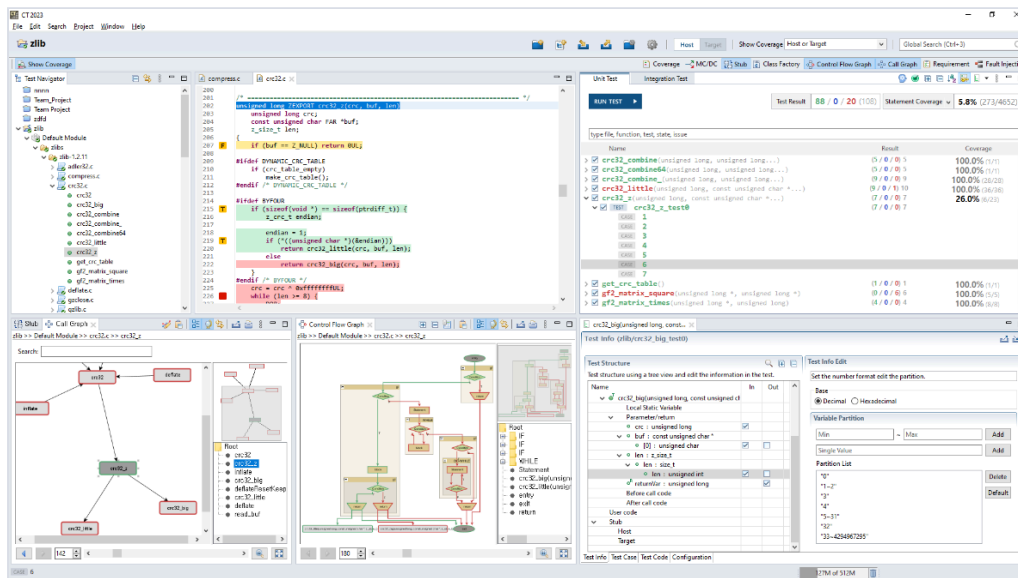
- a. Level A: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft.
- b. Level B: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a hazardous failure condition for the aircraft.
- c. Level C: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft.
- d. Level D: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft.
- e. Level E: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function with no effect on aircraft operational capability or pilot workload. If a software component is determined to be Level E and this is confirmed by the certification authority, no further guidance contained in this document applies.

Figure 2 DO-178C reliability level

Compliance with the DO-178C standard is a key element in ensuring the safety of aircraft software. This standard is essential to improve software quality, reduce risks from software defects, and comply with international aviation regulations.

### 3 What is CT

CT is a C/C++ code-based unit and integration test automation tool developed by Suresofttech. It was developed to achieve software testing and structural code coverage in mission-critical fields where safety and reliability are important, such as the aviation industry. The main features of CT are:



**Figure 3** CT run screen

## Test design

Automatically generates tests and test data, and provides an intuitive test design interface. It has the ability to import test data in various formats and allows for integrated test design with complex nested structures.

## Achieve coverage goals

CT supports statement, branch, MC/DC, function calls, and function coverage, and provides guidance for achieving coverage goals. You can check the coverage results by linking the source code and control flow graph, and after measuring system coverage, you can quickly achieve the coverage goal by testing only the parts that are not covered.

## Other Features

- Supports simulation and actual target environment testing

- Can be linked with DevOps tools
- Provides reports in various formats
- Provides requirements tracking feature in conjunction with requirements management tool
- Provides team testing feature that allows testing to be shared and conducted on multiple PCs
  - Eliminate duplicate work by synchronizing test assets such as project settings, stubs, and mock objects.
  - Performed test results can be checked in real time on the web dashboard.

## 4 How CT Supports DO-178C

We will explain in detail how CT supports DO-178C. CT features corresponding to the main sections of DO-178C are summarized as follows:

### Section 6.3.4 – Reviews and Analyses of the Source Code

This section states that the source code must match the data flow and control flow defined in the software architecture.

CT provides the ability to verify that the data interface and source code defined in the software design are implemented identically, and to confirm that the call relationship is the same as in the design. You can understand the flow of code with the control flow graph view, and view call relationships with the function call graph view.

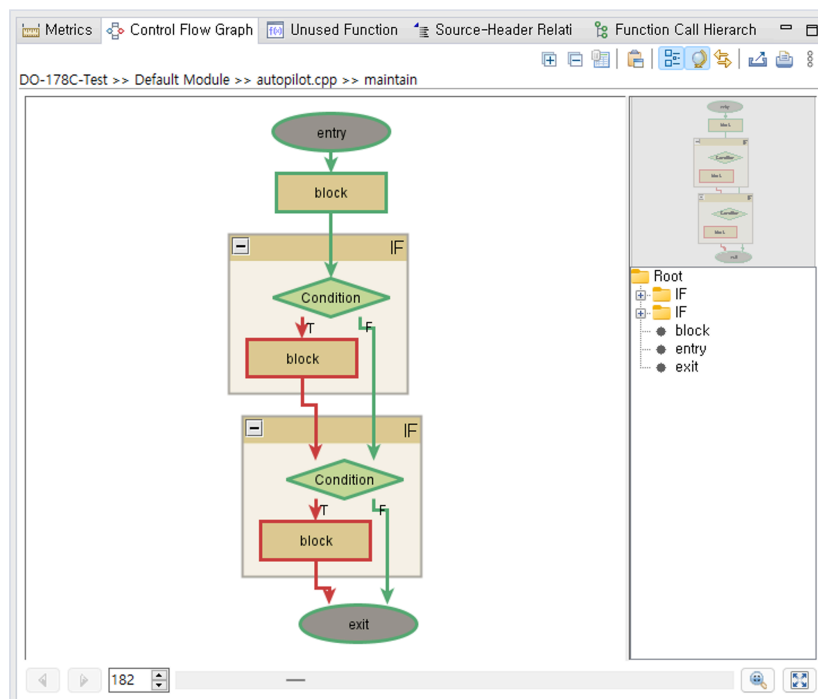


Figure 4 Control Flow Graph View

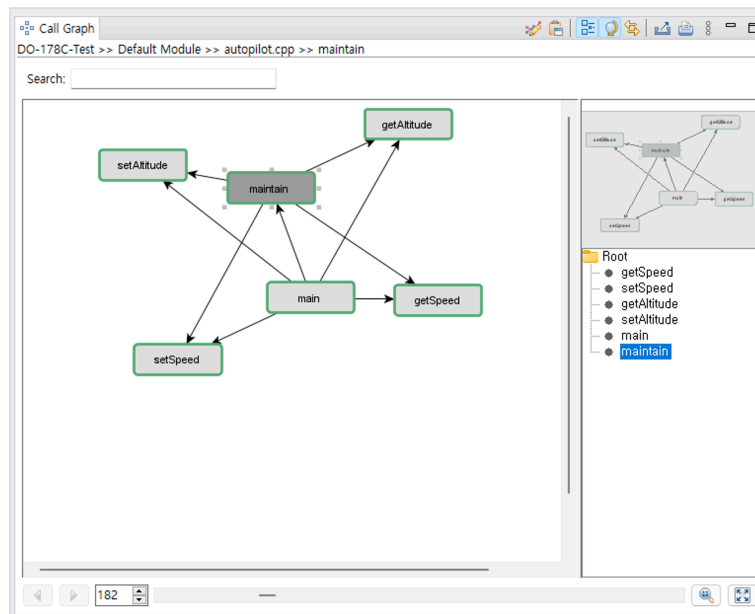


Figure 5 Call Graph View

## Section 6.4.1 – Test Environment

This section states that more than one test environment may be required to satisfy software testing objectives. Although testing the entire application on the target system may be considered an “ideal” environment, running and collecting requirements-based coverage and structural coverage in a fully integrated environment may not be feasible. Testing needs to be performed on small, independent components in a simulated environment.

In relation to this, CT provides the ability to perform tests not only in Windows and Linux-based simulation environments, but also in actual target environments. The host and target switches at the top of the tool allow you to perform tests in your preferred environment and view the results.

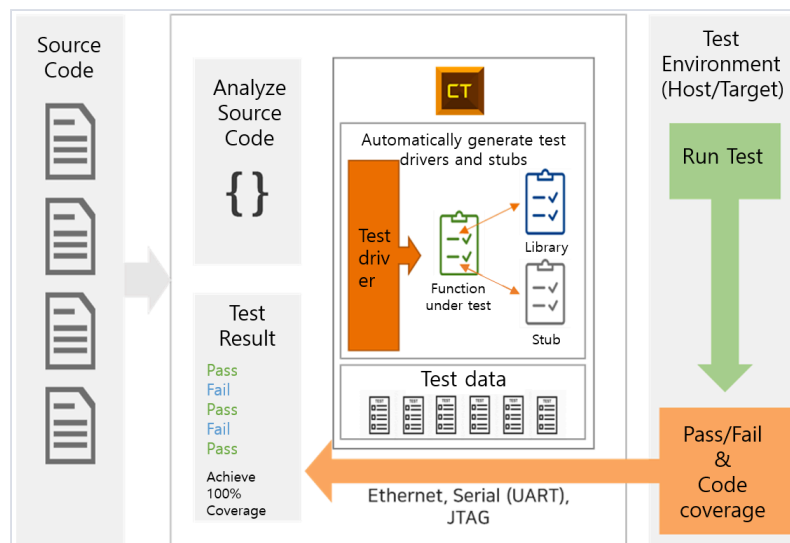


Figure 6 CT Test Flow

## Section 6.4.2 – Requirements-based Test Selection

This section states that software verification should be "requirements-based," as opposed to source-code-based.

CT can take requirements, associate them with tests, and revisit those tests when requirements change. This supports requirements-based test selection. In the Requirements view, you can view imported requirements and the results of tests associated with those requirements.

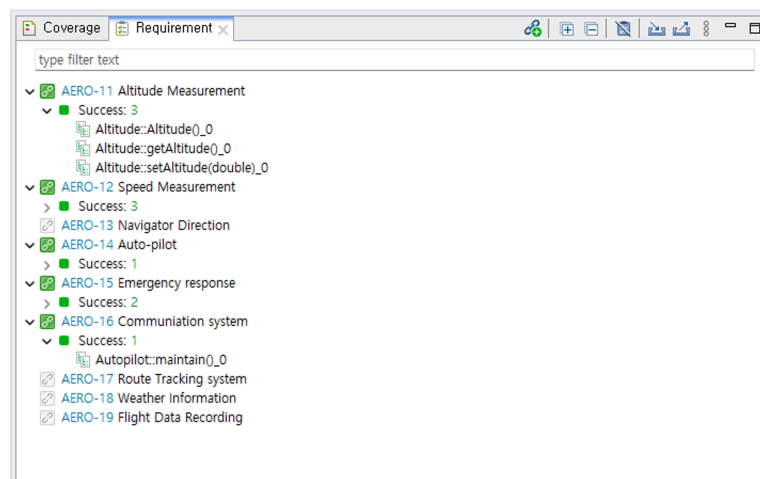


Figure 7 Requirement View

## Section 6.4.2.1 & 6.4.2.2 – Normal & Robustness Range Test Cases

This section states that test cases should be developed, including test cases for normal and abnormal ranges.

In relation to this, CT can perform tests within the normal input range, boundary value tests, and tests where failure is inevitable. This supports the creation of normal and abnormal range test cases.

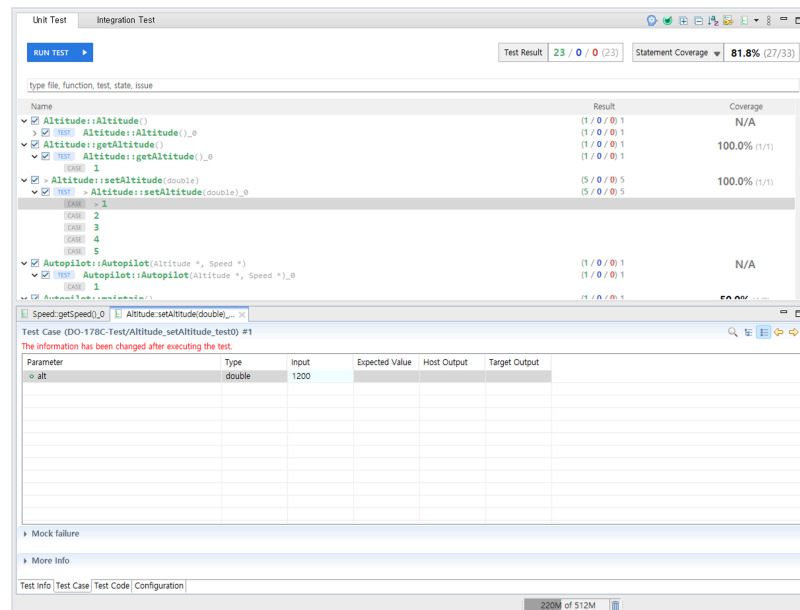


Figure 8 Test case Edit View

### Section 6.4.3 – Requirements-Based Testing Methods

This section specifies that requirements-based hardware/software integration testing, software integration testing, and low-level testing should be possible.

CT can perform testing on the actual target, at the host level, and at the source level. It supports various requirements-based testing methods using the test environment mentioned in Section 6.4.1 and the requirements view mentioned in Section 6.4.2.

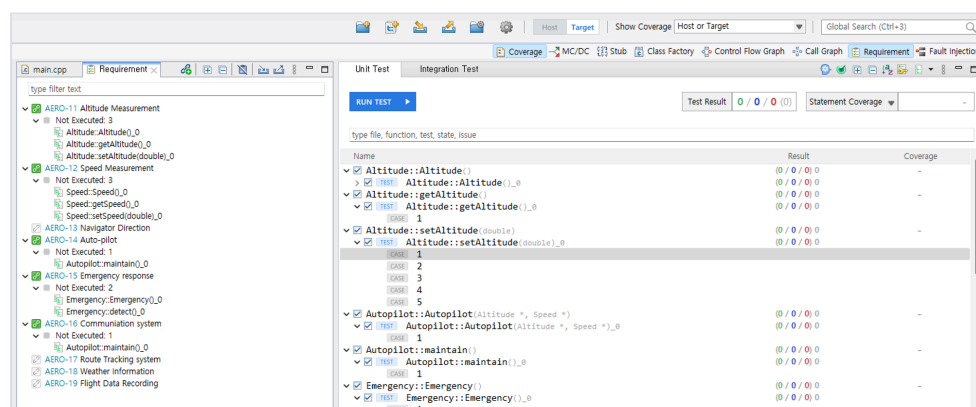
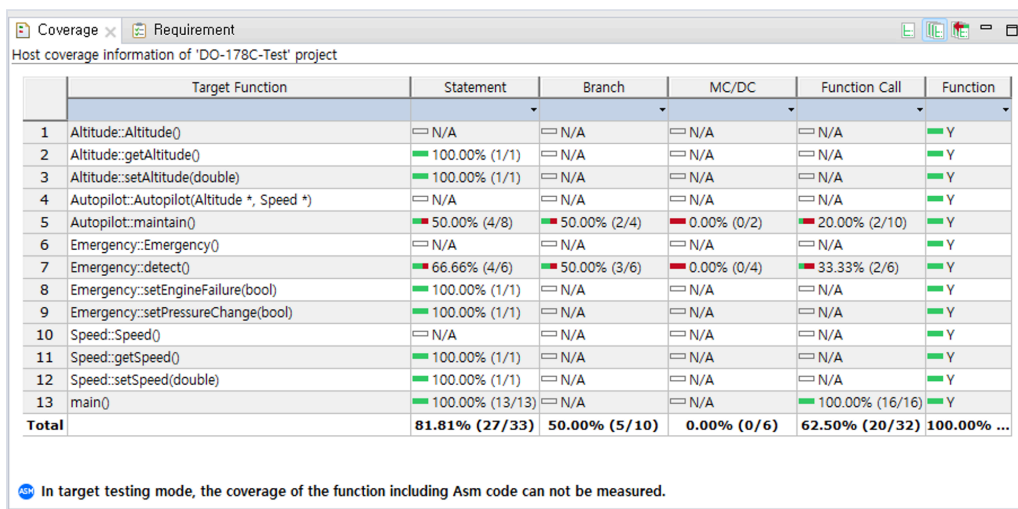


Figure 9 Requirement View and Test View

## Section 6.4.4 – Test Coverage Analysis

This section states that test coverage analysis should include requirements-based coverage and structural coverage analysis. The first step is to analyze the test cases to ensure that all requirements are associated with a specific test case. The second step is to verify that requirements-based testing executes the code to an appropriate level.

CT supports requirements-based coverage and structural coverage. Additionally, it supports achieving test coverage for data coupling and control coupling.



|              | Target Function                           | Statement             | Branch               | MC/DC              | Function Call         | Function           |
|--------------|-------------------------------------------|-----------------------|----------------------|--------------------|-----------------------|--------------------|
| 1            | Altitude::Altitude()                      | N/A                   | N/A                  | N/A                | N/A                   | Y                  |
| 2            | Altitude::getAltitude()                   | 100.00% (1/1)         | N/A                  | N/A                | N/A                   | Y                  |
| 3            | Altitude::setAltitude(double)             | 100.00% (1/1)         | N/A                  | N/A                | N/A                   | Y                  |
| 4            | Autopilot::Autopilot(Altitude *, Speed *) | N/A                   | N/A                  | N/A                | N/A                   | Y                  |
| 5            | Autopilot::maintain()                     | 50.00% (4/8)          | 50.00% (2/4)         | 0.00% (0/2)        | 20.00% (2/10)         | Y                  |
| 6            | Emergency::Emergency()                    | N/A                   | N/A                  | N/A                | N/A                   | Y                  |
| 7            | Emergency::detect()                       | 66.66% (4/6)          | 50.00% (3/6)         | 0.00% (0/4)        | 33.33% (2/6)          | Y                  |
| 8            | Emergency::setEngineFailure(bool)         | 100.00% (1/1)         | N/A                  | N/A                | N/A                   | Y                  |
| 9            | Emergency::setPressureChange(bool)        | 100.00% (1/1)         | N/A                  | N/A                | N/A                   | Y                  |
| 10           | Speed::Speed()                            | N/A                   | N/A                  | N/A                | N/A                   | Y                  |
| 11           | Speed::getSpeed()                         | 100.00% (1/1)         | N/A                  | N/A                | N/A                   | Y                  |
| 12           | Speed::setSpeed(double)                   | 100.00% (1/1)         | N/A                  | N/A                | N/A                   | Y                  |
| 13           | main()                                    | 100.00% (13/13)       | N/A                  | N/A                | 100.00% (16/16)       | Y                  |
| <b>Total</b> |                                           | <b>81.81% (27/33)</b> | <b>50.00% (5/10)</b> | <b>0.00% (0/6)</b> | <b>62.50% (20/32)</b> | <b>100.00% ...</b> |

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

Figure 10 Coverage View

## Section 6.4.4.1 & 6.4.4.2 – Requirements-Based & Structural Coverage Analysis

This section states that requirements-based testing should determine how well the software implementation has been verified (6.4.4.1) and which code structures were not covered during the requirements testing process (6.4.4.2).

In this regard, CT supports requirements-based coverage analysis and structural coverage analysis. Through this, you can determine the traceability of requirements and source code, and check data coupling and control coupling.

Requirement Test Coverage Report

Project : DO-178C-Test

Date Created : 2023-12-01 15:16:05

Version : 2023

| Result Summary                     |                      | Tests and Test Cases Summary |         |       |              |       |
|------------------------------------|----------------------|------------------------------|---------|-------|--------------|-------|
| Number of requirements             | 9                    | Number of tests              | 10      |       |              |       |
| Requirements traceability coverage | 55.56%               | Number of test cases         | 0       |       |              |       |
| Requirements consistency coverage  | 55.56%               |                              |         |       |              |       |
| Requirement Result Details         |                      |                              |         |       |              |       |
| All tests succeeded                | 5                    |                              |         |       |              |       |
| All tests executed, some failed    | 0                    |                              |         |       |              |       |
| All tests executed, some error     | 0                    |                              |         |       |              |       |
| Connected tests are not executed   | 0                    |                              |         |       |              |       |
| No tests connected                 | 4                    |                              |         |       |              |       |
| Requirements Test Result           |                      |                              |         |       |              |       |
| Key                                | Title                | Success                      | Failure | Error | Not Executed | Total |
| AERO-11                            | Altitude Measurement | 3                            | 0       | 0     | 0            | 3     |
| AERO-12                            | Speed Measurement    | 3                            | 0       | 0     | 0            | 3     |
| AERO-14                            | Auto-pilot           | 1                            | 0       | 0     | 0            | 1     |
| AERO-15                            | Emergency response   | 2                            | 0       | 0     | 0            | 2     |
| AERO-16                            | Communication system | 1                            | 0       | 0     | 0            | 1     |
|                                    | Total                | 10                           | 0       | 0     | 0            | 10    |

Figure 11 Requirement Test Coverage Report

## Section 6.4.5 – Reviews and Analyses of Test Cases, Procedures, results

This section states that you must ensure that the test results are satisfactory, the procedures were correct, and the results were expected.

CT helps you conduct test reviews by outputting test cases, test procedures, and results as reports. This helps you determine whether the test was performed properly and produced the results you expected.

Test Report(Host)

Project: DO-178C-Test

Create Date : 2023-12-01 15:18:09

Version : CT 2023

Total Coverage

| Statement      | Branch        | Call           | MC/DC       | Function        |
|----------------|---------------|----------------|-------------|-----------------|
| 81.81 %(27/33) | 50.00 %(5/10) | 62.50 %(20/32) | 0.00 %(0/6) | 100.00 %(13/13) |

Unit Test

| Total test | Succeed test | Total test case | Succeed test case |
|------------|--------------|-----------------|-------------------|
| 13         | 13           | 23              | 23                |

Unit Test Summary

| Test                                        | Total test case | Success | Failure | Error | Statement      | Branch        | Call           | MC/DC       |
|---------------------------------------------|-----------------|---------|---------|-------|----------------|---------------|----------------|-------------|
| Altitude:Altitude()_0                       | 1               | 1       | 0       | 0     | 0.00 %(0/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Altitude:getAltitude()_0                    | 1               | 1       | 0       | 0     | 3.03 %(1/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Altitude:setAltitude(double)_0 [r]          | 5               | 5       | 0       | 0     | 3.03 %(1/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Autopilot: Autopilot(Altitude *, Speed *)_0 | 1               | 1       | 0       | 0     | 0.00 %(0/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Autopilot:maintain()_0                      | 1               | 1       | 0       | 0     | 12.12 %(4/33)  | 20.00 %(2/10) | 6.25 %(2/32)   | 0.00 %(0/6) |
| Emergency:Emergency()_0                     | 1               | 1       | 0       | 0     | 0.00 %(0/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Emergency:detect()_0                        | 1               | 1       | 0       | 0     | 12.12 %(4/33)  | 30.00 %(3/10) | 6.25 %(2/32)   | 0.00 %(0/6) |
| Emergency:setEngineFailure(bool)_0          | 2               | 2       | 0       | 0     | 3.03 %(1/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Emergency:setPressureChange(bool)_0         | 2               | 2       | 0       | 0     | 3.03 %(1/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Speed:Speed()_0                             | 1               | 1       | 0       | 0     | 0.00 %(0/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Speed:getSpeed()_0                          | 1               | 1       | 0       | 0     | 3.03 %(1/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| Speed:setSpeed(double)_0                    | 5               | 5       | 0       | 0     | 3.03 %(1/33)   | 0.00 %(0/10)  | 0.00 %(0/32)   | 0.00 %(0/6) |
| main()_0                                    | 1               | 1       | 0       | 0     | 39.39 %(13/33) | 0.00 %(0/10)  | 50.00 %(16/32) | 0.00 %(0/6) |

Dec 1, 2023 3:18 PM

Figure 12 Test Report

## Section 12.2 – Tool Qualification

That section states that if a software tool is used to automate the verification process, it must be qualified. Tool Qualification Levels are classified by the following criteria:

- Criteria 1: A tool whose output is part of the airborne software and thus could insert an error.
- Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:
  - 1) Verification process(es) other than that automated by the tool, or
  - 2) Development process(es) that could have an impact on the airborne software.
- Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error.

Additionally, the TQL can be determined using the decision table together with the software reliability classification described earlier in DO-178C.

| Software Level | Criteria |       |       |
|----------------|----------|-------|-------|
|                | 1        | 2     | 3     |
| A              | TQL-1    | TQL-4 | TQL-5 |
| B              | TQL-2    | TQL-4 | TQL-5 |
| C              | TQL-3    | TQL-5 | TQL-5 |
| D              | TQL-4    | TQL-5 | TQL-5 |

**Figure 13** Tool Qualification Level Determination

CT has no direct impact on airborne software and is a tool that is not intended to be used to eliminate or reduce verification processes other than automated processes. So CT does not fall under criteria 1 or 2, but falls under criterion 3 because it could fail to detect an error within its intended use. And combined with the software reliability classification of DO-178C, the TQL of CT corresponding to criterion 3 is determined as TQL-5. This means that CT is a suitable tool for verification at all software reliability levels.

## 5 Conclusion

---

In this white paper, we specifically analyzed how Suresofttech's CT supports the aviation industry's DO-178C standard. At its core, CT was developed to meet the complex requirements of aviation software development and certification. This tool provides features such as automating test design, achieving structural coverage, and supporting testing in real target environments.

These features of CT directly correspond to several requirements of the DO-178C standard. Test environment setup, requirements-based test selection, and test coverage analysis meet the core elements of the DO-178C standard, thereby contributing to improving the safety and reliability of aviation software.

In conclusion, CT is evaluated as an effective tool for software development and certification compliant with the DO-178C standard. The support of these tools is likely to play an important role in safety and quality management of software in the aviation industry.