VECTOR

TÜV SÜD Certification Safety Manual

VectorCAST

Version 20260626

Author	Jeffrey Fortin
Publisher	Vector Informatik, GmbH

Change History

Date	Changes
2021-04-30	Updated for VectorCAST 2020 SP6
2021-06-02	Updated for VectorCAST 2020 SP7 and VectorCAST 2021
2021-12-23	Updated for VectorCAST 2021 SP1 – SP4
2022-02-24	Updated for VectorCAST 2021 SP5 – SP6
2022-12-19	Updated for VectorCAST 2021 SP7 and VectorCAST 2022, 2022 SP1-SP5
2023-06-06	Updated for VectorCAST 2022 SP6 – SP9
2024-03-04	Updated for VectorCAST 2021 SP8, VectorCAST 2022 SP10 – SP11 and VectorCAST 2023, 2023 SP1 – SP3
2024-05-17	Updated for VectorCAST 2023 SP 4 – SP7
2024-10-30	Updated for VectorCAST 2024, 2024 SP1 – SP3
2025-08-06	Updated for VectorCAST 2024 SP4 – SP5
2025-12-16	Updated for VectorCAST 2024 SP6 - SP7 and 2025, 2025 SP1 -SP4
2026-02-20	Updated for VectorCAST 2024 SP6 - SP7 and 2025, 2025 SP1 -SP4 Nuclear
2026-06-26	Updated for VectorCAST 2026 SP8

1 Introduction

The document outlines the intended use of VectorCAST for embedded software development in compliance with safety standards:

IEC 61508-3:2010
IEC 62304:2015
ISO 26262-8:2018
EN 50716:2023
IEC 60880:2006
ISO 25119-3:2018/AMD1:2020

2 Contents

The PDF Bookmarks can be used to navigate to the different sections of this document. Within this PDF you will find:

- The TÜV SÜD Certificate for VectorCAST
- The TÜV SÜD Report to the Certificate
- The VectorCAST Workflow Document

ZERTIFIKAT ◆ CERTIFICATE ◆ 認證證書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

Product Service

CERTIFICATE

No. Z10 033344 0001 Rev. 03

Holder of Certificate: **Vector Informatik GmbH**
Ingersheimer Str. 24
70499 Stuttgart
GERMANY

Certification Mark:

Product: **Software Tool for Safety Related Development**

Model(s): **VectorCAST/C++ Server / Desktop Edition**
VectorCAST/Ada Server / Desktop Edition
VectorCAST/QA Server / Desktop Edition

Parameters: The certified tools fulfills the requirements for support tools according to IEC 61508-3 and EN 50716. The tools are qualified to be used in safety-related software development according to IEC 61508, EN 50716, ISO 26262, ISO 25119 and IEC 60880. The tools are suitably validated for use in safety-related development according to IEC 62304. The test report is a mandatory part of this certificate.

Tested according to:
IEC 61508-3:2010
IEC 62304:2015
ISO 26262-8:2018
EN 50716:2023
IEC 60880:2006
ISO 25119-3:2018/AMD1:2020

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. This certificate is valid until the listed date, unless it is cancelled earlier. All applicable requirements of the Testing, Certification, Validation and Verification Regulations of TÜV SÜD Group have to be complied. For details see: www.tuvsud.com/ps-cert

Test report no.: VE85148C
Valid until: 2029-10-23

Date, 2024-10-30

volker Schneider

(Dr.-Ing. Volker Schneider)

TÜV SÜD Product Service GmbH • Certification Body • Ridlerstraße 65 • 80339 Munich • Germany

TÜV®

Add value.
Inspire trust.

Report

on the

Certificate

Z10 033344 0001 Rev. 03

of the

Software Tools
VectorCAST Tools

Applicant

Vector North America Inc.

39500 Orchard Hill Place
Novi, Michigan 48375
USA

Report No.: VE85148C

Version 4.5 of 2026-06-25

Certification Body

TÜV SÜD Product Service GmbH

Ridlerstraße 65
80339 München

Table of Contents page

List of Tables

page

Modification History

Rev.	Status	Date	Author	Modification / Description
1.0	-	2010-02-24	S. Waldhausen	Initial (Report no. 717504046, 1.0)
1.1	-	2013-07-30	S. Waldhausen	Re-certification for releases 6.0 and 6.1, extension of the certification scope to EN 50128.
1.2	-	2015-05-05	S. Waldhausen, W. Schlögl	Re-certification of new releases 6.2 and 6.3 + addition of VectorCAST/Ada to the scope + consideration of IEC 62304
1.3	-	2015-08-03	S. Waldhausen	6.2: fixed TI, TCL, and TD Levels
1.4	-	2016-06-23	W. Schlögl, S. Waldhausen	Re-certification of new release 6.4f (bug fixes and new features)
1.5	-	2018-06-08	W. Schlögl	Re-certification of new release 2018 SP1 (bug fixes, new features, new product structure)
1.6	-	2019-05-06	W. Schlögl	Re-certification of new release VC 2019, Update to ISO 26262:2018 Inclusion of EN 50657:2017
1.7	-	2019-07-30	W. Schlögl	Re-certification of new release VC 2019 SP1
1.8	-	2020-02-07	W. Schlögl	Re-certification of new releases VC 2019 SP2, SP3 and SP4
2.0	-	2020-08-24	W. Schlögl	Re-certification of new releases VC 2019 SP5, SP6, VC 2020, VC 2020 SP1, Change of certificate owner to Vector Informatik GmbH
2.1	-	2020-10-27	W. Schlögl	Re-certification of new release VC 2020 SP2
2.2	-	2020-12-04	W. Schlögl	Re-certification of new release VC 2020 SP3
3.0	-	2021-04-30	W. Schlögl	Re-certification VC 2020 SP4, SP5, SP6 Company name-change to “Vector North America Inc.” New report template
3.1	-	2021-06-02	W. Schlögl	Re-certification VC 2020 SP7, VC 2021 Update to EN 50128:2011/A2:2020
3.2	-	2021-12-17	W. Schlögl	Re-certification VC 2021 SP1, SP2, SP3, SP4
3.3	-	2022-02-23	W. Schlögl	Re-certification VC 2021 SP5, SP6
3.4	-	2022-12-07	W. Schlögl	Re-certification VC 2021-SP7, 2022, 2022-SP1, SP2, SP3, SP4, SP5

3.5	-	2023-05-31	W. Schlögl	Re-certification VC 2022-SP6, SP7, SP8, SP9
3.6	-	2024-02-27	W. Schlögl	Re-certification VC 2021-SP8, VC 2022-SP10, SP11, 2023, 2023-SP1, SP2, SP3 Restructuring and name change of the product Inclusion of IEC 60880 Report template v19 considered
3.6.1	-	2024-03-04	W. Schlögl	Typo Certificate Number
3.7	-	2024-05-16	W. Schlögl	Re-certification VC 2023-SP4, SP5, SP6, SP7 Report template v20 considered
4.0	-	2024-10-30	W. Schlögl	Re-certification VC 2024, 2024-SP1, SP2, SP3 Inclusion of ISO 25119 Inclusion of ISO/SAE 21434 (Cybersecurity engineering) Replacement of EN 50128 / EN 50657 by EN 50716
4.1	-	2025-07-18	W. Schlögl	Re-certification VC 2024 SP4, SP5 Report template v24 considered
4.2	-	2025-12-15	W. Schlögl	Re-certification VC 2024SP6-SP7, 2025, 2025SP1-SP4 Report template v27 considered
4.3	-	2026-01-21	W. Schlögl	Inclusion of IEC 60880 audit results for VC 2024-SP6, SP7, 2025, 2025SP1-SP4
4.4	-	2026-04-09	W. Schlögl	Re-certification VC 2025-SP5, SP6, SP7 Report template v28 considered
4.5	Active	2026-06-25	W. Schlögl	Re-certification VC 2025-SP8

Table 1: Modification history

1 Target of Evaluation (ToE)

In December 2010 Vector North America Inc. (Vector Software Inc.) requested TÜV SÜD Rail GmbH to test and certify the VectorCAST tools according to the standard listed in clause 4 of this report.

The testing mandate comprised the tools VectorCAST/C++, VectorCAST/Cover, and VectorCAST/Manage. The aim of the tool package is the support of automated test execution for unit- and integration testing, test coverage analysis, and the management of test projects.

During the re-certification for versions 6.2 and 6.3 in 2015 the testing mandate was extended to the tool VectorCAST/Ada.

In 2016 version 6.4 (6.4f) undergoes a recertification.

In May 2018 the version VC2018 SP1 was included in the scope of the certification. The testing mandate comprised the tools VectorCAST/C++, VectorCAST/Ada, and VectorCAST/QA.

In May 2019, the certification has been extended to cover the tool version “VectorCAST 2019”. Furthermore, the second edition of ISO 26262 (ISO 26262:2018) and the railway standard EN 50657:2017 was included.

In July 2019, the certification has been extended to cover the tool version “VectorCAST 2019 SP1”.

In Jan/Feb 2020, the certification has been extended to cover the tool versions “VectorCAST 2019 SP2, SP3, SP4”.

In July/August 2020, the certification has been extended to cover the tool versions “VectorCAST 2019 SP5, SP6, VectorCAST 2020 and VectorCAST 2020 SP1”. The ownership of the certificate was transferred Vector Informatik GmbH. This has no impact on the product. The complete development lifecycle is still under control of Vector North America Inc. (Vector Software, Inc.).

In Oct. 2020, the certification has been extended to cover the tool version “VectorCAST 2020 SP2”.

In Nov./Dec. 2020, the certification has been extended to cover the tool version “VectorCAST 2020 SP3”.

In April 2021, the certification has been extended to cover the tool versions “VectorCAST 2020 SP4, SP5, SP6”. The name of the legal entity changed from Vector Software Inc. to Vector North America Inc.

In May 2021, the certification has been extended to cover the tool versions “VectorCAST 2020 SP7 and VectorCAST 2021”. Furthermore, the certification was updated with respect to EN 50128:2011/A2:2020.

In December 2021, the certification has been extended to cover the tool versions VectorCAST 2021 SP1, SP2, SP3, SP4.

In February 2022, the certification has been extended to cover the tool versions VectorCAST 2021 SP5, SP6.

In December 2022, the certification has been extended to cover the tool versions VectorCAST 2021 SP7, 2022, 2022 SP1, SP2, SP3, SP4, SP5.

In May 2023, the certification has been extended to cover the tool versions VectorCAST 2022 SP6, SP7, SP8, SP9.

In February 2024, the certification has been extended to cover the tool versions VectorCAST 2021-SP8, VC 2022-SP10, SP11, 2023, 2023-SP1, SP2, SP3. The structure and names of the

product were changed from "Professional / Enterprise Edition" to "Server / Desktop Edition". Furthermore, IEC 60880 was included in the certification scope.

In May 2024, the certification has been extended to cover the tool versions VectorCAST 2023-SP4, SP5, SP6, SP7.

In October 2024, the certification has been extended to cover the tool versions VectorCAST 2024, 2024-SP1, SP2, SP3. Furthermore ISO 25119 and the cybersecurity standard ISO 21434 were included in the certification scope and EN 50128 / EN 50657 was replaced by EN 50716.

In July 2025, the certification has been extended to cover the tool versions VectorCAST 2024-SP4, SP5.

In December 2025 / January 2026, the certification has been extended to cover the tool versions VectorCAST 2024-SP6, SP7, 2025, 2025-SP1, SP2, SP3, SP4.

In April 2026, the certification has been extended to cover the tool versions VectorCAST 2025-SP5, SP6, SP7.

In June 2026, the certification has been extended to cover the tool versions VectorCAST 2025-SP8.

The VectorCAST tool family is a set of co-operating tools for test automation and used in the development of safety related software.

2 Scope of Testing

2.1 Test Specimen

The structure of the product has changed within VC2018. The functionality of the four former products VectorCAST/C++, VectorCAST/Ada, VectorCAST/Cover and VectorCAST/Manage has been divided in the products listed below. Each of the 3 products is available in a Professional Edition and in an Enterprise Edition.

2.1.1 VectorCAST/C++ Professional Edition

VectorCAST/C++ Professional Edition is an integrated software test solution. It provides support for complete test-harness construction for unit and integration testing, test executing from GUI scripts and regression testing. VectorCAST/C++ Professional Edition also provides code coverage analysis.

2.1.2 VectorCAST/C++ Enterprise Edition

VectorCAST/C++ Enterprise Edition provides all the features of the Professional Edition and some additional functionality like “Enterprise Testing”, “Change Based Testing” and “Probe Points for Fault Injection Testing”. It also comprises the former product VectorCAST/Manage

2.1.3 VectorCAST/Ada Professional Edition

In the course of re-certification for the VectorCAST tool versions 6.2 and 6.3 (2015), VectorCAST asked TÜV SÜD to extend the scope of certification to VectorCAST/Ada. Like VectorCAST/C++, VectorCAST/Ada is a dynamic software test solution.

VectorCAST/Ada Professional Edition is an integrated software test solution. It provides support for complete test-harness construction for unit and integration testing, test executing from GUI scripts and regression testing. VectorCAST/Ada Professional Edition also provides code coverage analysis.

2.1.4 VectorCAST/Ada Enterprise Edition

VectorCAST/Ada Enterprise Edition provides all the features of the Professional Edition and some additional functionality like “Enterprise Testing” and “Change Based Testing”. It also comprises the former product VectorCAST/Manage.

2.1.5 VectorCAST/QA Professional Edition

Vectorcast/QA Professional Edition provides the features of the former products VectorCAST Manage and VerctorCAST/Cover. It provides support for continuous testing, test collaboration, test automation and parallel testing. It performs code-coverage analysis during functional or system test. VectorCAST/QA supports Statement, Branch, and MC/DC Coverage.

2.1.6 VectorCAST/QA Enterprise Edition

VectorCAST/QA Enterprise Edition provides all the features of the Professional Edition and some additional functionality like “Probe Points for Fault Injection Testing” and “Change Based Testing”.

2.2 Test Specimen - VC 2023 and later

Since VC 2023, the structure of the product has changed. The functionality of the former products VectorCAST/C++ (Professional / Enterprise Edition), VectorCAST/Ada (Professional / Enterprise Edition) and VectorCAST/QA (Professional / Enterprise Edition) has been put in the products listed below. Each of the 3 products is available in a Server Edition and in a Desktop Edition.

The VectorCAST Server Edition licenses are designed automated (parallel) test execution on a server and cannot be used with the GUI version of VectorCAST.

The VectorCAST Desktop Edition licenses are designed to be run on a developer PC. The GUI version of VectorCAST is only supported with the Desktop Edition.

2.2.1 VectorCAST/C++ Desktop / Server

VectorCAST/C++ is an integrated software test framework. It provides support for complete test-harness construction for unit and integration testing, regression testing and code coverage analysis.

2.2.2 VectorCAST/Ada Desktop / Server

VectorCAST/Ada is (like VectorCAST/C++) an integrated software test framework. It provides support for complete test-harness construction for unit and integration testing, regression testing and code coverage analysis.

2.2.3 VectorCAST/QA Desktop / Server

VectorCAST/QA provides support for continuous testing, test collaboration, test automation and parallel testing. It performs code-coverage analysis during functional or system test. VectorCAST/QA supports Statement, Branch, and MC/DC Coverage.

2.3 Workflow Document for VectorCAST

The workflow document has been created by VectorCAST in course of the certification procedure. Its purpose is to describe the use cases that were assumed for the tool qualification. It defines the scope of the certification and serves as basis for tool classification according to ISO 26262. In addition, the workflow document resumes relevant information to be considered when applying the VectorCAST products in a safety related development.

Compilers and the C Code Parser are separate tools: they are not within the scope of the tool qualification, but the respective interface modules that provide their support are.

2.4 Qualification support

Vector North America Inc. offers qualification materials that are produced for the customer specific target environment and tool chain. These “Tool Qualification” materials for VectorCAST/C++, VectorCAST/Ada, VectorCAST/QA allow application-specific qualification.

2.5 Nomenclature and Identification of VectorCAST Tools

The following table identifies the tool releases of the ToE. The software is delivered as a compressed file. (Identification information of previous releases and are documented in the corresponding previous reports).

OS	Checksum
Release 2025 SP8 (VectorCAST/C++, VectorCAST/Ada, VectorCAST/QA)	
vcast.linux64.2025sp8.tar.gz	sha512: 92cc23cd78de6fe6ee3c38a9178925913d1feb68634e294ea5f28f3ee8b2b d839836b967dd6ef184c9b5da4f3f2a1a321810527403c8dc4c4e3f8659d8585920
vcast.win64.2025sp8.zip	sha512: 63a1e5dce66a711186c9dc994ed93c7734c80185c04af54d3ec15c51929de 3996bdfe21142b3264582e7bc2dbeeb8cb6625a62790dfaea4ea4a4e34a562bb507
vcast.2025sp8.iso	sha512: 6eeef89ff646ddf4000d93d8768f3abbb5fc8a40d7ddf4f97f4cc0c35e848 76a2d89a835fcfea93bec9dccfc7ba091884f88ddfa5d65ef01082c4a76d0e13e80

Table 2: Identification

3 Certification Requirements

The certification of the VectorCAST Tools is according to the regulations and standards listed in clause 4 of this document. This certifies the successful completion of the following test segments.

I. Functional Safety including

- Functional safety management (FSM) and safety lifecycle
- Applied safety development process
- Analysis of the software
- Verification and validation procedures/activities
- Fault simulations and software tests
- Functional tests

II. Safety information in the product documentation (safety manual, user manual, installation and operating instructions)

III. Product-Related Quality Assurance in Manufacture and Product Development

3.1 Certification requirements cybersecurity

The VectorCAST tools were examined regarding the following topics, as far as applicable:

I. Cybersecurity (ISO/SAE 21434) including

- Organizational Cybersecurity Management
- Project Dependent Cybersecurity Management
- Distributed Cybersecurity Activities
- Continual Cybersecurity Activities
- Concept
- Product Development
- Cybersecurity Validation
- Production
- Operations and Maintenance
- End of Cybersecurity Support and Decommissioning
- Threat Analysis and Risk Assessment Methods

II. Cybersecurity (IEC 62443) including

- Third Party Components
- Bug Tracking / Bug reporting / Vulnerability Monitoring
- User Documentation
- Testing
- Development Environment / File Integrity / Key Management

3.2 Certification Documentation

The detailed technical evaluation is documented in the most recent version of the Technical Report:

Document No.	Description	Project No.
VE83605T	Technical Report	717534568
Safety related requirements, conditions and restrictions can be found in the following user documentation		
Workflow Document for VectorCAST	VectorCAST Safety Manual, Workflow Document	-

Table 3: Technical Report

Based on the specified purpose of use of the VectorCAST Tools in development of safety critical applications, the certification is based on the set of standards listed in clause 4 of this document. The issuance of the certificate states compliance with these references unless specifically noted otherwise.

4 Standards and Guidelines

The regulations and guidelines which form the basis of the type testing are listed below.

4.1 Functional Safety

No.	Reference	Description
/N1/	IEC 61508-3:2010 EN 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems Part 3: Software requirements
/N2/	ISO 26262-8:2018	Road vehicles - Functional safety Part 8: Supporting processes

Table 4: Basic safety standards

No.	Reference	Description
/N3/	ISO 25119-1:2018	Tractors and machinery for agriculture and forestry - Safety-related parts of control systems - Part 1: General principles for design and development
/N4/	ISO 25119-3:2018/ AMD 1:2020	Tractors and machinery for agriculture and forestry - Safety-related parts of control systems – Part 3: Series development, hardware and software
/N5/	EN 50716:2023	Railway Applications - Requirements for software development

Table 5: Associated safety standards

No.	Reference	Description
Remark:	*The following standards were approved by other testing services.*	
/N6/	IEC 62304:2006+A1:2015	Medical device software – Software life cycle processes
/N7/	IEC 60880:2006	Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions

Table 6: Other product related standards

4.2 Cybersecurity Standards

Remark: The following standards were approved by other testing services.

No.	Reference	Description
/N8/	ISO/SAE 21434:2021	Road vehicles — Cybersecurity engineering

Table 7: Cybersecurity standards

No.	Reference	Description
/N9/	IEC 62443-4-1:2018	Security for industrial automation and control systems Part 4-1: Secure product development lifecycle requirements

Table 8: Related Cybersecurity standards

4.3 Quality Management System

No.	Reference	Description
[M1]	QMS	Quality Management System TÜV SÜD Rail GmbH

Table 9: Quality Management System

5 Tool qualification and classification (Functional Safety)

The aim of the certification is to enable customers to apply the VectorCAST tools for safety-related development without further tool qualification activities when applying to the recommendations and conditions documented in the Safety Manual and in the Report to the Certificate.

IEC 61508:

IEC 61508:2010 demands the employment of offline support tools (7.4.4.2).
According to IEC 61508-4:2010, section 3.2.11, VectorCAST/C++, VectorCAST/Ada, and VectorCAST/QA are offline support tools. They support testing of embedded software, which yields a classification as T2 tools.
For T2 tools, IEC 61508-3:2010 requires that:

- The tool functionality and behaviour, as well as any instructions or constraints shall be documented.
- A tool validation according to 7.4.4.7 is not required for T2 tools, but it is suggested as a means to judge correctness of the tool results.

For T2 as well as for T3 tools, only qualified versions shall be used.

ISO 26262:

According to ISO 26262-8:2018, the classification depends on the detection of possible tool errors. The standard classifies software tools according to their tool impact (TI) and the probability of tool error detection (TD).

The tool impact for the VectorCAST tools is TI2, because a verification tool can fail to detect existing errors in the source code to be analyzed although it may not introduce errors into an application.TI2 requires an estimation of the tool error detection TD on customer side.

IEC 62304:

IEC 62304:2006 requires tools to be “suitably validated” (Table C.3). The tool validation according to IEC 61508 is a main aspect of the testing described in this report. Since IEC 62304 does not define how suitable validation is achieved, but refers to IEC 61508 with respect to tools, the validation can be considered suitable also in the sense of IEC 62304.

IEC 62304 AMD1:2015 does not contain changes with regard to tools.

ISO 25119:

According to ISO 25119-3 “tools and translators which are proven in use shall be applied, in order to avoid any difficulties due to translator failures which can arise during development”. The correctness of such tools shall be demonstrated by specific testing, by an extensive history of satisfactory use, or by independent verification of their output for the applicable safety-related system.”

ISO 25119-1 requires for any tool used for development, implementation or testing an assessment or verification of the application of the tool.

EN 50716:

EN 50716:2023 specifies the process and technical requirements for the development of software for programmable electronic systems for use in: - control, command for signalling applications, - applications on-board of rolling stock. As of December 2023, EN 50716 supersedes EN 50128.

The requirements for software tools (see clause 6.7 in EN 50716) are directly taken over from EN 50128. (The requirements for software tools in EN 50128 are explicitly derived from the requirements on software tools according to IEC 61508-3.) Due to the equivalences between the standards no separate testing has been performed with respect to EN 50716.

The part of the audit covering the development process, quality assurance measures, verification and validation, modification and bug handling can be taken over.

IEC 60880:

The VectorCAST tools support verification activities (static analysis) in the life cycle phases 7 and 8 of figure 3 in IEC 60880, so the tool type is “verification and validation tool” (see sub-clause 14.2.3 in IEC 60880). The requirements for verification and validation tools are covered by the requirements of IEC 61508 and ISO 26262.

6 Safety Parameters

6.1 Functional Safety

The tests performed and quality assurance measures implemented by the Vector North America Inc. have shown that the VectorCAST tools comply with the testing criteria specified in clause 4 subject to the conditions defined in clause 8 and are suitable for use in development of safety related applications.

The VectorCAST tools, classified as T2 off-line tools according to IEC 61508-4:2010, are suitable to be used in safety-related development according to IEC 61508:2010, ISO 25119:2018 and EN 50716:2023 for any SIL.
The VectorCAST tools are qualified to be used in a standard-conform development process according to ISO 26262-8:2018 for any ASIL.

IEC 62304 does not define particular requirements on tool selection and tool qualification, but it refers to consult IEC 61508. So, by qualification against IEC 61508, the VectorCAST tools can be said to be suitable for use in safety related development according to IEC 62304:2006+A1:2015 for any software safety class.

The VectorCAST tools, classified as “verification and validation” tools according to IEC 60880, are suitable to be used in safety-related software development for systems performing category A safety functions.

7 Cyber Security

The documents provided by Vector North America Inc. and the audits performed have shown that the development of the VectorCAST tools complies with the applicable parts of the standards specified in clause 4.2.

8 Implementation Conditions and Restrictions

The use of the VectorCAST tools shall comply with the current version of the safety parts of the user manual, and the following implementation and installation requirements have to be followed, if the VectorCAST tools are used in safety-related developments.

- The use cases that have been in the focus of the certification activities, as well as the underlying functionalities of the VectorCAST Products, are described in the Workflow documentation “Workflow Document for VectorCAST”. In case of a deviant tool usage, additional qualification efforts may be required.
- The Workflow document bundles information that has to be considered in a safety-related development. It indicates further information sources the user can access online for up-to-date information (e.g. on supported platforms or known software bugs including available workarounds).
- The policy of the VectorCAST tools is full support of compilers, i.e. arbitrary makefile settings are allowed, build options are not suppressed. This means that the test environment is as close as possible to the application program. At the same time, compiler options should be used consciously, and their potential interference with the testing should be analyzed (e.g. code instrumentation may influence code optimization).
- When updating to a new tool version, default settings may change, and the behavior of the tool may vary due to algorithmic changes. Please check the release documentation and compare the results for your specific project.

9 Certificate Number

This report specifies technical details and implementation conditions required for the application of the VectorCAST Tools to the certificate:

Z10 033344 0001 Rev. 03

Technical Certifier

VECTOR

VectorCAST Safety Manual

Workflow Document

Version 10.0 of 2025-12-16

Author	Fortin, Jeffrey
Status	Completed
Publisher	Vector Informatik GmbH

Document Management

Review

Date Name (Role)
30-OCT-2024 Jeffrey Fortin VectorCAST Product Manager

Change History

Date	Changes (Author)
19-SEP-2022	JYU: A new format for the Workflow Document is being adopted to match other safety documents published by Vector and to align with TÜV SÜD recommendations.
06-JUN-2023	Reviewed and Updated for VectorCAST version 2022 SP6 – SP9. Only date and version has changed.
22-SEP-2023	Reviewed and updated for new VectorCAST Product Names.
07-MAR-2024	Add new use case for Google Test coverage
21-MAY-2024	Adding new use case for CUDA
18-JUL-2024	Reviewed and updated for VectorCAST 2024
01-OCT-2024	Updated EN 50716 from EN 50218
14-NOV-2024	Added Use case for Coded Tests Update Test Execution to include Test Case Variants
18-JUN-2025	Updated TCL2 and TCL3 entries. Removed TESTinsights.
14-NOV-2025	Added Use case Instrument Blocks for Statement Coverage

VECTOR

Contents

1 Introduction

1.1 Purpose of this Document

This document summarizes the workflow for the development of safety related software. For each use case, a classification of the Tool Confidence Level (TCL) is made with recommendations for the qualification methods to be used. Wherever relevant, requirements and assumptions on the context in which the tools are used that have an impact on the TCL are stated.

1.2 Product Identification

Regarding VectorCAST, all statements in the document apply to the version identified in the Report on the Certificate prepared by TÜV SÜD Rail GmbH which is included in this Safety Manual.

1.3 Normative References

The VectorCAST tools are used to satisfy the testing requirements for many regulatory standards including:

- ISO 26262 Road Vehicles – Functional Safety [1]
- IEC 61508 – Functional Safety of electrical/electronic/programmable electronic safety-related systems [2]
- EN 50716 – Railway Applications - Requirements for Software Development[3]
- IEC 62304 – Medical Device Software – Software Lifecycle Processes [4]
- IEC 60880 (Nuclear Power Plant Safety)

For each of these standards, Vector has prepared white papers detailing the specific testing requirements for each standard and the associated methods of satisfying the testing requirements with the VectorCAST tools.

1.4 Conditions of Use

VectorCAST is a software tool for testing software and is designed for testing software intended for use in a safety critical system where functional safety regulations and standards apply.

1.5 Warning and Caveats

This section addresses safety considerations that should be considered when using the various VectorCAST products. These considerations are made to ensure that users are aware of the documentation available and properly notified of the recently released bug fixes, release notes, and major functionality additions.

1.5.1 User Manuals

Please see the VectorCAST User Manuals and Vector Knowledgebase for the most up to date information. VectorCAST user manuals and installation guides can be downloaded from the VectorCAST Downloads[5] page.

1.5.2 Vector Knowledgebase

The Vector knowledgebase is used as a resource to provide information on many different topics from product installation through tool usage. These detailed articles are gathered and developed from customer comments and give access to common questions. The Vector Knowledgebase[6] can be found on the vector.com website.

[1] https://www.vector.com/int/en/products/products-a-z/software/vectorcast/application-areas/#c341256
[2] https://www.vector.com/int/en/products/products-a-z/software/vectorcast/application-areas/#c341260
[3] https://www.vector.com/int/en/products/products-a-z/software/vectorcast/application-areas/#c341267
[4] https://www.vector.com/int/en/products/products-a-z/software/vectorcast/application-areas/#c341263
[5] https://www.vector.com/vectorcast-downloads/
[6] https://support.vector.com/kb?id=kb_search&spa=1&u_kb_facet_5_lbl=9d7fca7d1b4e98508e9a535c2e4bcbb4

1.6 Release Notes and Notification

The VectorCAST product release notes are provided for each build and provide details on exactly what is included in the current release. Current and previous version release notes are available on the VectorCAST Downloads[7] page.

New release notifications are posted on the vectorcast.com website. Customers are not individually notified of a new release, in general, due to privacy laws. However, a customer may request to be notified by contacting Vector Support[8].

1.7 Responsibilities of the End-user

Some features of VectorCAST are designed for analyzing, controlling and/or otherwise influencing electronic systems in operation. Their use could cause serious operational malfunction in the surrounding environment, damage to property and/or bodily injury. Therefore, these features are exclusively intended for operation by persons who (i) have understood the possible effects of the actions which may be caused by VectorCAST; (ii) are specifically trained in the handling with VectorCAST and the electronic systems intended to be influenced; and (iii) have sufficient experience in using VectorCAST in a safe manner (hereinafter collectively "Qualified Personnel"). The end-user shall ensure that VectorCAST is only operated by Qualified Personnel.

1.8 Tool Configuration Constraints

Please see the VectorCAST user guide and install guide for the most up to date information. Those can be download from the VectorCAST Downloads[9] page.

1.9 Customer Support Information

If problems are encountered using VectorCAST, a support request can be filed using the support web form[10]. Alternatively, you can reach the Vector Support Team by email at support@vector.com. If a tool defect or bug is found, a tracking numbers is provided, and customers may request to be notified when a bug fix is delivered in a version or service pack release.

1.10 Tool Usage According to ISO 26262

In accordance with ISO 26262, any tool can be used in the development of safety-related systems. The tool user needs to ensure that, based on the use case, a tool classification was conducted and that a qualification of the tool has been executed based on this classification (and thus based on the use case).

For these steps, it is crucial in which way the tool is used (the use case). For example, if the software component is for a non-safety-relevant function, then there is a different classification than if the component is a safety-relevant component, requiring a lower demand on the tool qualification.

Each release of VectorCAST is regression tested by Vector before release, and thus prior to delivery to customers. As a tool manufacturer cannot know in advance all use cases, a classification by the tool manufacturer for all use cases is unrealistic and thus also a tool qualification (in this case: the regression tests) can never cover the use cases of all users in detail.

Therefore, the tool classification and the tool qualification must be done by the user of the tool, who has a complete understanding of the use case. The use cases, corresponding classification ratings and hints for qualification listed in Section 2, below, may serve as examples for project classifications and qualifications.

[7] https://www.vector.com/vectorcast-downloads/
[8] https://www.vector.com/int/en/support-downloads/support/
[9] https://www.vector.com/vectorcast-downloads/
[10] https://www.vector.com/us/en-us/support-downloads/issue-reporting/software-hardware/

1.11 The VectorCAST Tool Suite

VectorCAST is a suite of tools that are used for testing software and collecting metrics such as code coverage. The suite is composed of:

- VectorCAST/C++
- VectorCAST/Ada
- VectorCAST/QA

These tools are sold depending on the usage of the tool. For desktop test development, the offering is referred to as the Desktop Edition. For automated test execution on a server, the offering is referred to as the Server Edition. Therefore, you will see the product listed on sales quotes in this way. For instance, if you will be using VectorCAST for desktop unit test development for C++, you would use VectorCAST/C++ Desktop Edition. The Server Editions do not allow access to the VectorCAST GUI as they are intended only for automated test execution. The current list of offerings is:

- VectorCAST/C++ Desktop Edition
- VectorCAST/C++ Server Edition
- VectorCAST/Ada Desktop Edition
- VectorCAST/Ada Server Edition
- VectorCAST/QA Desktop Edition
- VectorCAST/QA Server Edition

1.12 Older VectorCAST Product Offerings

Previously, the VectorCAST offering was structured not by usage but by functionality. For instance, if a simple single test environment was being used then we offered a Profession Edition. For testing multiple test environments, the VectorCAST Enterprise Edition was offered. The Enterprise Edition included all the functionality of the Professional Edition along with enhanced automation, testing fixtures and scalable configuration setup. Just as with the newest offerings, these editions applied to each of the members of the tool suite. The offering previously contained:

- VectorCAST/C++ Professional Edition
- VectorCAST/C++ Enterprise Edition
- VectorCAST/Ada Professional Edition
- VectorCAST/Ada Enterprise Edition
- VectorCAST/QA Professional Edition
- VectorCAST/QA Enterprise Edition

Moving forward, only the functionality of the Enterprise Edition is offered as part of the newer VectorCAST Desktop and Server Editions. The Professional offering is no longer being offered but may still be used for older projects that have a specific configuration set for a specific product version.

In this document, it is assumed that the Desktop or Server Edition is being used. If a Profession Edition is in use, all the functionality of that edition is covered by this document but not all functions described in this document are available in the Professional Editon. Specifically functions that use the VectorCAST Project Facility for managing multiple test environments in a single project.

1.13 Overview of VectorCAST/C++ and VectorCAST/Ada

The VectorCAST/C++ and VectorCAST/Ada products help to automate the various tasks associated with unit and integration testing. The following key terms are used in this workflow document to describe the various components of VectorCAST/C++ and VectorCAST/Ada tools:

Unit Under Test (UUT)	The UUT is usually a .c or .cpp file, or an Ada package. The individual subprograms (functions, methods, or procedures) can be invoked with either parameter data or global data to execute specific source lines of code within a subprogram.
Test Driver	A test driver is a main program created specifically to invoke the various subprograms in the UUT(s).
Dependent	A dependent is a module whose subprograms would be called from the UUT.
Stub	A stub takes the place of a real dependent. A stub is generally used when you want to control the values that a dependent subprogram would return to a calling subprogram. This allows tests to be developed with specific data to drive the execution down a specific path in the code.
Test Environment	A test environment (also **test harness** or **test bed)** is an executable program built specifically to perform unit or integration testing on one or more units (files) of source code. A test environment includes a test driver, one or more UTTs, any stubs, and any real dependents. A test environment is generally able to: • Invocate all visible functions • Manipulate UUT inputs • Capture UUT outputs • Generate test reports • Capture stub input parameters • Manipulate stub output data • Test exceptions • Compare regression data

1.13.1 Unit Testing

Unit testing commonly consists of testing a single .c or .cpp file. This type of testing is done because it provides the ability to define specific input values to drive the processing down certain paths in the code. As test cases are executed, the associated code coverage is reported.

The executable environment (harness) generated by VectorCAST (Figure 1) consists of a test driver, the source files under test, any user-designated stubs for dependent units, and any real dependents. The test environment is data-driven, meaning test data is read by the environment during execution. This approach precludes any need of having to compile and link a new executable environment for each test case. VectorCAST automates the process of generating test environments.

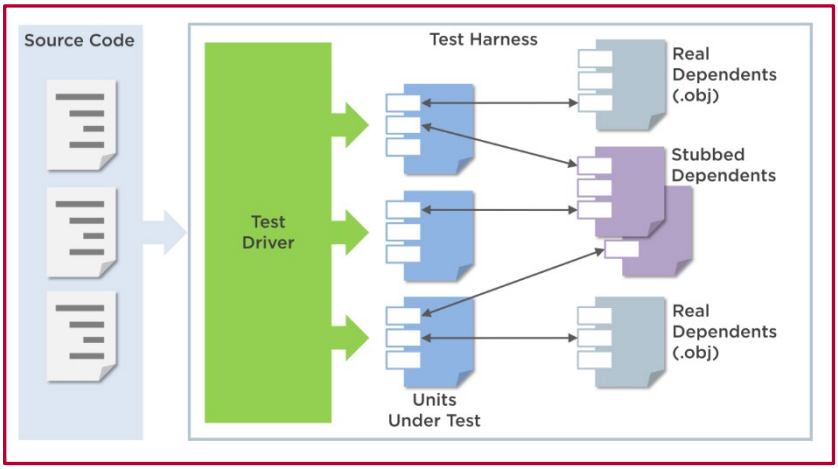

Figure 1: Test Environment

VectorCAST/C++ and VectorCAST/Ada support many different development processes and testing. Figure 2 below represents the traditional waterfall approach to testing. Using this method, once the application code is generated, unit testing is done for each individual unit, followed by integration and system test; the end-result being a fully tested application.

While this approach leads to full code coverage, it is likely that some of the coverage achieved will be redundant. It is possible to get to 100% code coverage with unit testing alone. The additional integration and system tests are performed to ensure that elevated level requirements are being met.

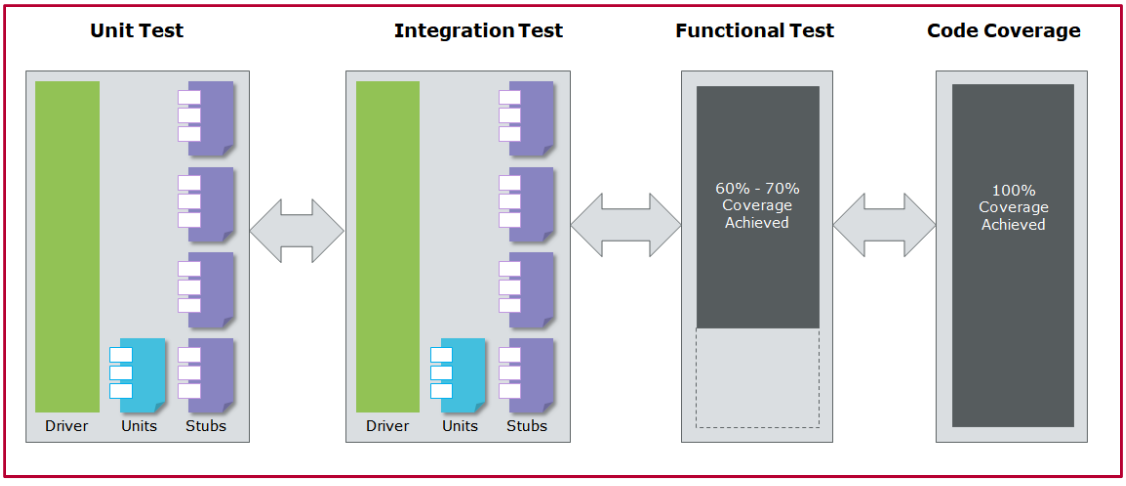

Figure 2: Traditional Waterfall Approach Test Method

In a requirements-based testing approach (as depicted in *Figure 3*) or what might also be common in a legacy application, high-level requirements are tested first. This approach will give a significant level of code coverage after executing system or functional tests. Executing all the high-level requirements-based tests can result in 60-70% of the code being covered. The remainder of the source coverage is achieved with unit test. Only those lines of code that were not executed during the system tests will be unit tested.

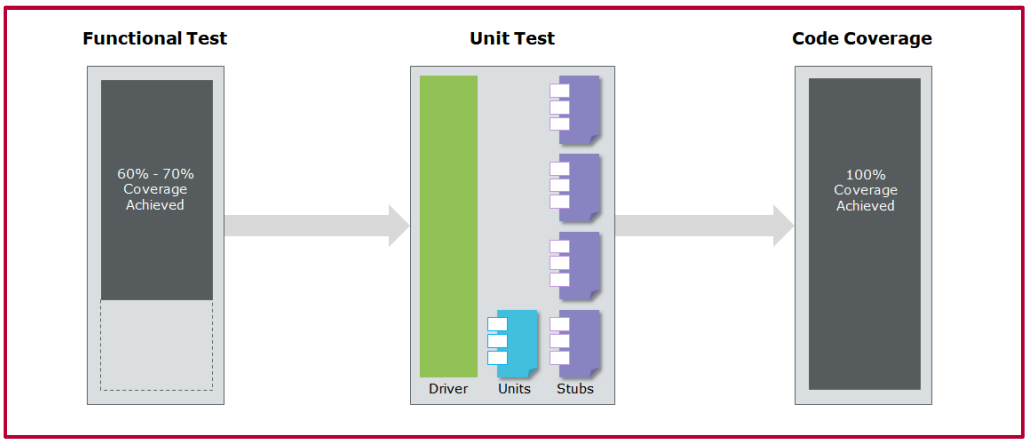

Figure 3: Requirements-Based / Legacy Application Test Method

This technique alleviates the need to do 100% unit testing on the application. The reason that this approach is used for legacy applications is due to the fact that it is sometimes difficult to go back, and unit test all source code once an application is being used in the field.

1.13.2 Integration Testing / Multiple Units Under Test

Integration testing is an extension of unit testing. It is used to check interfaces between units and requires you to combine units that make up some functional process. Many tools claim to support integration testing by linking the object code for real units with the test harness. This method builds multiple files within the test harness executable but provides no ability to stimulate the functions within these additional units.

VectorCAST/C++ and VectorCAST/Ada allows you to stimulate any function within any unit, in any order within a single test case. Testing the interfaces between units will generally uncover hidden assumptions and bugs in the application. In fact, integration testing may be a good first step for those projects that have no history of unit testing. VectorCAST/C++ and VectorCAST/Ada supports fully automated integration testing.

Key Points

- Multiple units can be included in the test environment
- Complex test scenarios for these classes where we stimulate a sequence of functions across multiple units within one test case is supported
- Code coverage metrics for multiple units can be captured

1.13.3 Anatomy of the Test Tools

The underlying functionality of the VectorCAST/C++ and VectorCAST/Ada test tools is comprised of the following components:

Parser	The parser module allows the tool to understand your code. It reads the code and creates an intermediate representation for the code (usually in a tree structure). Basically, the same as the compiler does. The output, or “parse data” is generally saved in an intermediate language (IL) file.

CodeGen	The code generator module uses the “parse data” to construct the test harness source code.
Test Harness	While the test harness is not specifically part of the tool, its generation is the result of the input provided by the user. This input (selection of the units under test and any dependent units) results in the creation the test driver and dependent stubs which when compiled and linked, results in the executable test harness.
Compiler	The compiler module allows the test tool to invoke the compiler to compile and link the test harness components.
Target	The target module allows tests to be easily run in a variety of runtime environments including support for emulators, simulators, embedded debuggers, and commercial RTOS.
Test Editor	The test editor allows the user to use either a scripting language or a sophisticated graphical user interface (GUI) to setup pre-conditions and expected values (pass/fail criteria) for test cases
Coverage	The coverage module allows the user to get reports on what parts of the code are executed by each test. It also allows for sections of code to be marked as “Covered by Analysis” or CBA. With CBA justifications can be written that will appear in the coverage reports as to why a particular piece of code was not covered by testing.
Reporting	The reporting module allows the various captured data to be compiled into project documentation.
CLI	A command line interface (CLI) allows further automation of the use of the tool, allowing the tool to be invoked from scripts, make, etc.
Regression	The regression module allows tests that are created against one version of the application to be re-run against new versions.
Requirements Gateway	Requirements can be imported from external databases and associated with test cases. The test results can then be exported to external databases.

1.13.4 Parser and Code Generation

VectorCAST/C++ uses commercial parser technology that is licensed from the industry leading parser technology company. The robustness of the parser and code generator can be verified by evaluating the tool with complex code constructs that are representative of the code to be used for your project. The VectorCAST/Ada tool uses proprietary parser technology that has been in use since 1994.

Key Points

- The parser technology is commercial for C and C++
- Tool versions are the same for C and C++
- The entire C++ language is implemented

If a parser problem is uncovered by the user, the problem may manifest itself with an incorrect test harness being generated. If this were to occur, there is a formal reporting process that includes a case number to track the issue with the parser technology company.

1.13.5 Test Driver

The Test Driver is the “main program” that controls the test. Here is a simple example of a driver that will test the ***sine*** function from the standard C library:

```
#include <math.h>
#include <stdio.h>
int main () {

      float local;
      local = sin (90.0);
      if (local == 1.0) printf ("My Test Passed!\n");
      else printf ("My Test Failed!\n");
      return 0;
}
```

Although this is a simple .c example, VectorCAST/C++ and VectorCAST/Ada provide a full-featured GUI for building test cases, integrated code coverage analysis, an integrated debugger (using the debugger from your cross-compiler environment), and an integrated target deployment.

Notice that this driver has a bug. The bug is that the sin function actually uses radians not degrees for the input angle.

An advanced test driver use case is to add User Code. This use case is available for VectorCAST/C++ and VectorCAST/Ada. For VectorCAST/C++, the test harness can also be modified using Probe Points.

VectorCAST’s User Code capability enables you to write C/C++-language expressions to set or check the value of data objects, or to do some other dynamic test case setup. With user code, you can write code to read data from a file, call initialization routines, or assign and verify data objects based on dynamic criteria.

VectorCAST Probe Points allow the user to insert user-defined blocks of code, or probe points, before or after any executable statement. Probe Points are not available for VectorCAST/Ada.

Key Points

- The driver code is automatically generated
- The following can be tested without writing any code:
  - Testing over a range of values
  - Combinatorial Testing
  - Data Partition Testing (Equivalence Sets)
  - Lists of input values
  - Lists of expected values
  - Exceptions as expected values
  - Signal handling
- Sequences of calls to different methods can be set in the same test

1.13.6 Stubbing Dependent Functions

VectorCAST/C++ and VectorCAST/Ada builds replacements (stubs or mocks) for dependent functions when you want to control the values that a dependent function returns during a test. Stubbing is an important part of integration and unit testing since it allows you to isolate the code under test from other parts of your application, and more easily stimulate the execution of the unit or sub-system of interest.

Key Points

- Stubs are automatically generated
- Complex outputs are supported automatically (structures, classes)

- Each call of the stub can return a different value
- The stubs keep track of how many times they were called
- The stubs keep track of the input parameters over multiple calls
- You can stub calls to the standard C library functions like “malloc”

1.13.7 Test Data

VectorCAST/C++ and VectorCAST/Ada uses a data-driven test execution mechanism. For a data-driven architecture, the test harness is created for all the units under test and supports all of the methods and functions defined in those units. When a test is to be run, the tool simply provides the stimulus data across a data stream such as a file handle or a physical interface like a UART.

Key Points

- The VectorCAST test harness is data driven
- Tests are executed without compiling between test runs
- Test cases can be edited outside of the IDE

1.13.8 Automated Generation of Test Data

The following paragraphs describe some of the automatic test case techniques used in VectorCAST/C++:

Min Mid Max Test Cases	Min Mid Max tests will stress a function at the bounds of the input data types. C and C++ code often will not protect itself against out-of-bound inputs. The engineer has some functional range in their mind, and they often do not protect themselves against out of range inputs.
Partitioned Test Cases	Partitioned tests create “partitions” for each data type and test the min and max value from each partition. The assumption is that values from the same partition will stimulate the application in a similar way.
MC/DC Test Cases	MC/DC tests use the MC/DC analysis to examine the unique paths that exist through a procedure. MC/DC tests can automatically create a high level of paths coverage.
Basis Path Test Cases	Basis Path tests use the basis path analysis to examine the unique paths that exist through a procedure. BP tests can automatically create a high level of branch coverage.

The key thing to keep in mind when thinking about automatic test case construction is the purpose that it serves. Automated tests are good for testing the robustness of the application code, but not the correctness (even if they provide a high level of code coverage). For correctness, you must create tests that are based on what the application is supposed to do (the requirements), not what it does do (the code).

1.13.9 Compiler Integration

VectorCAST/C++ provides integrations with many embedded compiler environments. This integration servers two main purposes: The first is to allow the test harness components to be compiled and linked automatically, without the user having to figure out the compiler options needed. The second is to honor any language extensions that are unique to the compiler being used. Especially with cross-compilers, it is very common for the compiler to provide extensions that are not part of the C/C++ language standards.

Key Points

- VectorCAST automatically compiles and links the test harness
- Honor and implement compiler-specific language extensions

- Provides a modern GUI to the compiler environment (IDE, CLI, etc.)
- Provides interface to import project settings from your development environment
- VectorCAST is integrated with the underlying debugger

1.13.10 Support for Testing on an Embedded Target

In this section, we will use the term “Tool Chain” to refer to the total cross development environment including the cross-compiler, debug interface (emulator), target board, and Real-Time Operating System (RTOS). VectorCAST/C++ has robust target integrations for your tool chain.

VectorCAST/C++ allows for “push button” test execution where all the complexity of downloading to the target and capturing the test results back to the host is abstracted into the “Test Execution” feature so that no special user actions are required.

An additional complication with embedded target testing is hardware availability. Often, the hardware is being developed in parallel with the software, or there is limited hardware availability. A key feature of VectorCAST is the ability to start testing in a native / simulator environment and later automatically transition all tests to the actual hardware.

Key Points

- VectorCAST/C++ and VectorCAST/Ada supports many embedded tool chains
- Tests can be built on a host system and later use them for target testing
- Downloading tests to the target is automatic
- Test results are automatically captured on the host
- New integrations with embedded tool chains are always being added

1.13.11 Test Case Editor

VectorCAST/C++ and VectorCAST/Ada allows you to set up test input and expected values for all constructs, including non-trivial constructs.

For example: the test case editor provides a simple way to construct a C++ class, an abstract way to set up an STL container; like a vector or a map.

Key Points

- Allowed ranges for scalar values shown
- Array sizes shown are shown
- Easy to set Min and Max values with tags rather than values. This is important to maintain the integrity of the test if a type changes.
- Special floating-point numbers are supported (e.g., NaN, +/- Infinity)?
- Automatic combinatorial tests (vary 5 parameters over a range and have the tool do all combinations of those values)
- The editor is “base aware” so that you can easily enter values in alternate bases like hex, octal, and binary?
- You can easily enter absolute tolerances for expected results, (e.g., +/- 0.05) and relative tolerances (e.g., +/- 1%) for floating point values
- Test data can be easily imported from other sources like Excel

- Test data can also be created using the Classification Tree Method through the Vector Test Data Editor (Available for C and C++ on the Windows 64-bit platform)

1.13.12 Unit / Integration Test Reporting

VectorCAST/C++ creates an easy-to-understand report showing the inputs, expected outputs, actual outputs, and a comparison of the expected and actual values.

Key Points

- Multiple output formats are supported: HTML, Text, CSV, XML
- VectorCAST provides both a high level (project-wide) report as well as a detailed report for a single function
- The report content user configurable
- The report format user configurable

1.14 Overview of VectorCAST/QA

1.14.1 Code Coverage

Both the VectorCAST/C++, VectorCAST/Ada and VectorCAST/QA tools produce code coverage. The VectorCAST/C++ and VectorCAST/Ada tools report the coverage metrics from unit and integration testing. The VectorCAST/QA tool reports coverage from system and functional test. Both tools show a listing of the original source code file with colorations for covered, partially covered, and uncovered constructs. This allows you to gauge the effectiveness of your test efforts by identifying which areas of an application were exercised during a test run. VectorCAST/QA provides a convenient way to analyze the completeness of your system tests, ensuring that applications are not released with untested code.

It is also important to understand the impact of instrumentation (the additional source code added to your application). There are two considerations: one is the increase in size of the object code, and the other is the run-time overhead. It is important to understand if your application is memory or real-time limited (or both). This will help you focus on which item is most important for your application.

Key Points

- VectorCAST/QA allows instrumentation to be integrated into your “make” or “build” system
- Provides easy to interpret coverage results. Annotated listings with a graphical coverage browser are provided as are tables of metrics.
- Sections of code can be marked as “Covered by Analysis” or CBA. With CBA justifications can be that will appear in the coverage reports as to why a particular piece of code was not covered by testing.
- Coverage capture is flexible and can be captured in RAM
- Statement, Branch, MC/DC, Function and Function Call coverage are supported
- Multiple coverage types be captured in one execution
- Coverage data can be shared across multiple test environments (e.g., coverage can be captured during system testing and be combined with the coverage from unit and integration testing)
- Aggregate coverage for all test runs in a single report

VectorCAST/QA allows you to analyze any portion of your application, or the entire application at once. For each file that is analyzed, VectorCAST/QA creates a multi-tabbed source-viewer widget containing the following information:

Coverage Summary	Provides a color-coded view of your source code that identifies code that is completely covered, partially covered, or uncovered. (– according to the selected coverage metrics)
Metrics Summary	Provides a tabular list of code complexity and currently achieved source-code coverage for each subprogram.

Basis Path Analysis	Shows all basis paths for each subprogram.
MC/DC	Modified Condition/Decision Coverage for the RTCA DO-178C standard for Level A flight software, ISO 26262 for ASIL D, and IEC 61508 and EN 50716 SIL 4.

The coverage data generated by VectorCAST/QA during system requirements-based testing can be shared with the VectorCAST/C++ unit and integration test tool. This sharing allows 100% coverage to be achieved with a combination of system, unit, and integration testing.

1.14.2 Coverage Reports

VectorCAST/QA generates report output that can be viewed with the integrated code-coverage browser or extracted to files for inclusion in project documentation. Report information includes the code covered, a basis-path analysis, and code-complexity metrics.

1.14.3 Embedded Target Support

VectorCAST/QA supports testing on a variety of embedded target architectures. Whether you are using an industry standard such as INTEGRITY® or VxWorks®, a custom kernel, or a bare board, VectorCAST/QA is the solution for embedded target-code coverage. VectorCAST/QA is normally used during system testing to capture the code coverage from system tests.

1.14.4 Function Call Coverage

Function call coverage is referenced in ISO 26262 is shown in VectorCAST with the Function Call Coverage Type Coverage code-coverage browser and in the Metrics Reports.

1.15 Overview of VectorCAST/C++ and VectorCAST/Ada Advanced Use Cases

VectorCAST/C++ and VectorCAST/Ada advanced use cases satisfy two basic goals for any project. The primary goal is to save time testing. The secondary goal is to allow the created tests to be leveraged over the life cycle of the application. This means that that the time and money invested in building tests should result in tests that are re-usable as the application changes over time and easy to configuration manage.

The VectorCAST/C++ and VectorCAST/Ada advanced use cases allow you to import previously developed VectorCAST/C++ and VectorCAST/Ada test environments into regression test suites, providing a single point-of-control for all unit and integration test activities. At-a-glance logs, summary reports, and color-coded pass/fail criteria highlight the status of each test within the regression suite.

The VectorCAST/C++ and VectorCAST/Ada Project facility allow you to take standalone VectorCAST/C++ and VectorCAST/Ada environments and import them into a VectorCAST project. These individual "test environments" can then be grouped into larger "Environment Groups" and "Test Suites". Environments can be members of multiple Environment Groups, and Environment Groups can be assigned to multiple Test Suites. (This enables users to structure their VectorCAST projects to match the architecture of their application. Because Environment Groups and Test Suites can be easily duplicated, the same tests can be run using various source baselines, on different host platforms, or with a different compiler or embedded target.

The VectorCAST/C++ and VectorCAST/Ada Project is used by the entire development team. Software managers use the high-level reports and graphs to track testing progress and trends. QA Engineers use the tool to easily design test campaigns and monitor release readiness. Developers use the tool to identify and resolve defects.

The Management Summary report (Figure 4) enables users to view the status of each test case. Data is automatically recorded for build status and time, test execution status and time, and code coverage achieved.

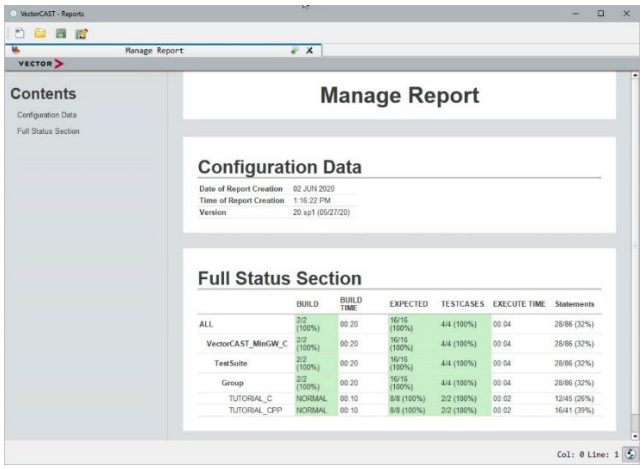

Figure 4: VectorCAST Management Summary Report

Using the integrated Python interpreter, additional "tests" can be added for each component and a column for the "test" will be added to the report. For example, a user might want to compare the test execution time to some threshold or might want to perform a file "diff" between a test artifact and a known previous result.

Key Points

- All files are saved to facilitate automatic regression testing
- The VectorCAST/C++ and VectorCAST/Ada provides a complete and documented Command Line Interface (CLI)
- The harness files are generated automatically and therefore do not need to be configuration managed
- Integration with configuration management tools is provided
- Multiple baselines of code with the same set of test cases can be run automatically
- Supports distributed testing to allow portions of the tests to be run on different physical machines to speed up testing.

2 Example Use Cases

All following examples are covered under the TÜV SÜD Rail GmbH certification scope. Please read the section, 1.10 Tool Usage According to ISO 26262, for more information on how to interpret this section.

2.1 VectorCAST/Ada and VectorCAST C/C++

2.1.1 Creating New Environment

Intended Purpose:	Environments are the functioning frame of VectorCAST testing. The place to setup all the vital parts of the test system. Getting a new environment configured allows the users to set the foundation of future testing.
Environmental, Functional and Process Constraints:	• VectorCAST installed • Valid license • VectorCAST does not display any warning when started • Writable permission for working directory • Valid compiler is available and accessible
Description:	1. Start VectorCAST. 2. Set working directory: the default location for building environments and loading environment script that has been saved with relative path. The working directory also contains the local configuration file. 3. Start the environment wizard and choose the desired environment setup. 4. Set the correct configuration in environment wizard, including but not limited to compilation system, compiler options, source file location, desired UUT and stubs, user code, build options and name of the environment. 5. Build the environment when satisfied with settings.
Output:	Environment overview report
Tool Impact:	TI1 (No possibility of impact on safety requirement) Rationale: • No interaction of VectorCAST with system components
Tool Error Detection:	TD3 (Low degree of confidence of prevention or detection) Rationale: • Error displayed during setup does not indicate problem with existing system
Tool Confidence Level:	TCL1 (Determined from TI1 and TD3)
Recommended method of qualification	Not required

2.1.2 Building and Executing Test Cases

Intended Purpose:	Setup for unit testing of source code and execute test in VectorCAST to test functions of a system component. View test results via passed/failed output.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system
Description:	1. Open environment. 2. Select a subprogram of Unit Under Test (UUT) for simple test cases, a simple test case corresponds to a single invocation of UUT. 3. Assign values to parameters of the subprogram under test, any parameters of the stubbed dependent subprogram and any global data object. 4. Enter values for individual scalar data items. 5. Select a collection of simple test cases for compound test cases. A compound test cases may invoke a UUT multiple times with a variety of data within a single execution. 6. Add, edit, duplicate, rename, delete test cases until satisfactory. 7. Execute all or a selected number of test cases. (Note that tests may be selected or de-selected using the Test Case Variants.)
Output:	• Available test cases are shown in the Test Case Tree. • Execution Status viewer automatically opens and provides detailed real time test case execution information. • Test Case Tree is updated with a PASS/FAIL designation after execution.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g. by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST

	can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.3 Incremental Rebuild

Intended Purpose:	Using Incremental Rebuild feature in VectorCAST to perform Change-Based-Testing. The VectorCAST Project Manager automatically identifies the minimum tests that must be run for each code change. Automatically re-runs only those tests that have been identified as affected by code change.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Environment is whitebox • Coverage instrumented • Environment has been built and executed previous to the changes in the unit under test (UUT) or dependent body without any errors. • A change is made to the UUT or a non-stubbed dependent.
Description:	1. Save and recompile the source code after the code change. 2. Select Environment->Incremental Rebuild from the menu for the corresponding environment. 3. VectorCAST determines what has changed, updates the associated portion of the test harness, and recompiles and then re-instruments those portions. 4. View the displayed result of the incremental rebuild via the Incremental Rebuild Report.
Output:	• Incremental Rebuild Report • Execution and coverage status are selectively removed following an incremental rebuild.

	• Unaffected test cases retain their execution and coverage data.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g. by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.4 Using Code Coverage

Intended Purpose:	The Code Coverage tool determines which lines of source code (statement), which branches of source code (branch), or which equivalence pairs (MC/DC) have been executed by one or more sets of test case data. Using code coverage to see the completeness of the test suite and analyze the untested code to design test cases.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system

	• Have or will incorporate test cases
Description:	1. Open environment 2. Initialize coverage: Select **Coverage => Initialize =>** a. Statement b. Branch c. MC/DC d. Statement + MC/DC e. Statement + Branch 3. (Optional) Initialize coverage for units other than the unit under test (UUT): Select **Coverage => Initialize Custom…** 4. (Optional) Avoid instrumenting sections of source code by delimiting the section with source code comment markers. 5. (Optional) Enable coverage using the **Coverage => Enable** command for the current open environment if previously disabled. 6. View coverage results in Coverage Viewer 7. (Optional) Customize the Coverage Viewer 8. (Optional) Open test case that covers a line in Coverage Viewer 9. (Optional) Map line selection to original source view in Coverage Viewer 10. (Optional) Discard all existing coverage data by choosing **Coverage => Remove All Coverage Data** 11. (Optional) View Aggregate Coverage Report for an annotated source listing and metrics table giving the complexity and level of coverage for each subprogram 12. (Optional) View Metrics Report for the metrics table and pair status if MC/DC or Statement + MC/DC is initialized. 13. (Optional) View Code Coverage Summary to see coverage information for all the units in an environment
Output:	• Coverage data in multiple forms
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test.

	• The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.5 Using User Code

Intended Purpose:	Write expressions to set or check the value of data objects or do dynamic test case setup. Write code to read data from a file, call initialization routines, or assign and verify data objects based on dynamic criteria.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system
Description:	1. Open environment 2. (Optional) Environment user code: **Environment => User Code => Edit** 3. (Optional) Test case user code: Testcase User Code tab of the Test Case Editor 4. (Optional) Individual parameter user code: right-clicking on a parameter in the Parameter Tree and electing “Properties…” or double-clicking the parameter and selecting the User Code tab **5.** (Optional) Stub user code: **Environment => Configure Stubs => Edit** 6. (Optional) Insert user code tag via the User Code Tags icon or **View => Dock Control => User Code Tags** 7. (Optional) Using external editor for user code a. Choose an external editor in **Tools => Options**

	b. Invoke editor by right-clicking user code text areas and then select **Invoke external editor…** 8. (Optional) Import contents of file into user code by right-clicking user code text area and then select **Import file contents…** 9. Test compile the user code via the **Test Compile** button 10. Save the user code 11. (Optional) Recompile environment 12. (Optional) Remove user code
Output:	• Error messages if unsuccessful
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.6 Using VectorCAST Covered by Analysis (CBA)

Intended Purpose:	Users augment test coverage metrics with coverage analysis data sets. Using the Coverage Analysis Editor and combines the test and analysis coverage metrics in a single report. VectorCAST CBA can also import analysis files, including those generated by third party tools.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Instrumented for coverage
Description:	1. Open the environment 2. Add Coverage Analysis for the selected unit, select **CBA** button or right-click the unit under test and select **Add Coverage Analysis** 3. Use the Coverage Analysis Editor a. Statement Coverage – mark statement or condition "considered covered" b. Branch Coverage – mark condition as having the True **(T)** or False **(F)** branch covered c. MC/DC Coverage – annotate one or more rows of the Equivalence Pairs table as covered 4. (Optional) Double click on existing Analysis result in Test Case Tree to Rename, Delete, select Coverage and Deselect Coverage 5. (Optional) Change Covered-by-Analysis display color via **Tools => Options Coverage** tab **Report** sub-tab 6. (Optional) Import Analysis files via **Coverage => Import Coverage Analysis** 7. (Optional) Export Analysis files via **Coverage => Export Script => Analysis Results...** 8. (Optional) Remove Analysis files by right-clicking on the result and select **Delete** 9. Re-Instrument with CBA data.
Output:	• Covered by Analysis Report, accessible via **Test => View => Covered by Analysis Report** • Aggregate Coverage Report • Metrics Report
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.

Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.7 Using Requirements Gateway

Intended Purpose:	VectorCAST Requirement Gateway (RGW 3.0) provides traceability between software requirements and test cases and allows the import and mapping of requirements to test cases. Import testing requirements from Application Lifestyle Management (ALM) or Requirements tool to a VectorCAST RGW repository, assign specific requirements to testcases in a unit test environment and then export the resulting pass or fail status back to the ALM tool.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing test cases • ALM or Requirements tool available and in use • ALM or Requirements tool supported by VectorCAST RGW • No existing error in ALM or Requirements tool

	• Connectivity between VectorCAST host/server with ALM or Requirements tool's host/server
Description:	1. Set the Requirements Gateway (RGW)'s repository location by selecting **Tools => Requirements Gateway => Options...** from the Menu Bar. 2. Open the Requirements Gateway dialog using one of the three options: a. The RGW button on the Toolbar b. Drop down menu next to the Toolbar and select the name of the gateway c. Select Tools => Requirements Gateway => <gateway name> 3. Select the correct Gateway Profile (ALM or Requirements tool or CSV) 4. Authenticate with the gateway 5. Import requirements from the Requirements subsystem into the Repository, fill in all the desired associated attributes 6. View imported requirements 7. Link requirements to test cases 8. Run test cases 9. Export test results to the gateway, specify the export settings
Output:	• Import result (success or failure) • Export result (success or failure)
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities.

	• The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert. • Errors can be detected on multiple interfaces.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.8 Customizing Reports

Intended Purpose:	VectorCAST Reports can be customized to contain any information with any layout. Reports can be produced as either HTML or text and a PDF from the HTML version.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing test cases • Initialized coverages • Knowledge of CSS, HTML, Python, JSON
Description:	1. (Optional) Customize existing reports a. (Optional) Add and remove existing sections b. (Optional) Specify report sections to replace c. (Optional) Specify report section order 2. (Optional) Customizing Templates a. Create configuration directory b. Configure VectorCAST to use custom templates from the command line c. Add custom footer by editing the footer.html.template file d. Add custom logo to reports by editing the main.html.template file e. Customizing report style by create a custom style sheet in .css format and add it via **Tools => Options** and click the **Report** and then **Format** sub-tab or from the command line. 3. (Optional) Create new reports

	a. Create a main report file; the main entry point for the report with the initial data, the report name and describes the sections of the report and order in which they appear. b. Create a section file that contains the data that goes into the report c. Create a template which describes the layout of the data d. Run the new report from the command line or create a script to run the report and add it to Custom Tools
Output:	• Custom report generated if successful
Tool Impact:	TI1 (No possibility of impact on a safety requirement) Rationale: • Since the modification of VectorCAST report has no bearing on system components, VectorCAST does not have an impact in safety requirements • Modification of report does not alter test case values, only the display of those values.
Tool Error Detection:	TD1 (High degree of confidence of prevention or detection) • An error in format of report is highly visible and easy detected
Tool Confidence Level:	TCL1 (Determined from TI1 and TD1)
Recommended method of qualification	Not Required

2.1.9 Using Jobs Windows and Monitor

Intended Purpose:	View output in real time when running a target test as the commands are running.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing test cases
Description:	1. Open the Jobs window, a tab (auto-hidden window) located at the bottom of the main window. 2. If the Jobs window is hidden, the status bar displays the current or last command run. Hover over the window to open it and use the pin icon to keep the window open. 3. Single-clicking on a command line highlights the associated line in the Messages window. Hovering over a command line shows the exit code and stdout and stderror for that line.

	4. Open the Job Monitor by double-clicking a command line in the Jobs window or right-click a command line and select **Job Status**. 5. For compile or link errors that occurs in user code during test execution, click on the **Test Command** button to re-run the command displayed in the Job Status window. 6. View the automatically opened Exestuation Status viewer when a test case is executed. Click the **Show Details** button to display the full details of the execution in real time.
Output:	• Commands and their results
Tool Impact:	TI1 (No possibility of impact on a safety requirement) Rationale: • The viewing of execution status or commands has no bearing on system components, VectorCAST does not have an impact in safety requirements
Tool Error Detection:	TD1 (High degree of confidence of prevention or detection) • Helps display errors
Tool Confidence Level:	TCL1 (Determined from TI1 and TD1)
Recommended method of qualification	Not Required

2.1.10 Changing Application Preferences

Intended Purpose:	Change VectorCAST's look and feel for better productivity.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system
Description:	1. Set the style of the main window to look and feel like one of four GUI standards in **Tools => Options => GUI** 2. Setup external text editor in the **GUI** tab, **Text Editor Option** sub-tab 3. Setup external browser in the **GUI** tab, **External Browser Option** sub-tab 4. Toggle gridlines in test cases in the **Test Case Options** sub-tab 5. Alphabetize test cases by clicking on the title bar of the Test Case column on the Test Case Tree to toggle the display from an alphabetical listing of Unit, Subprogram, and Test Case names to a listing based on the definition on the order of those items. 6. Change VectorCAST's font by going to **Tools => Options => GUI** and click the **Change Application Font** button in the Other Options panel.

	7. Automatically save test cases before execution by choosing the option in the **GUI** tab. 8. Turn on a reminder before closing multiple tabs by choosing the options in the **GUI** tab.
Output:	
Tool Impact:	TI1 (No possibility of impact on a safety requirement) Rationale: • The options of application preference have no bearing on system components, VectorCAST does not have an impact in safety requirements
Tool Error Detection:	TD1 (High degree of confidence of prevention or detection)
Tool Confidence Level:	TCL1 (Determined from TI1 and TD1)
Recommended method of qualification	Not Required

2.1.11 Using VectorCAST Probe Points

Intended Purpose:	Insert user-defined blocks of code, or probe points, before or after any executable statement. Probe points can be inserted during unit, API, or system testing and are created and maintained on a per-unit basis.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Instrumented for coverage
Description:	1. Open environment. 2. Open the Coverage and Probe Viewer via either right-click on the unit under test (UUT) or any function under the UUT and then **Edit Probe Points** or the Probe Point icon in the toolbar. 3. Small black dots in the Coverage and Probe Viewer shows where probe points can be inserted. Click the black dot next to the executable line will change the dot to a green circle, indicating an active probe point and the Probe Point editor is opened for entering the source code for the probe point. 4. (Optional) Insert function probe points at either the entry point of a function or at the end of a function. 5. (Optional) Insert File Scope probe points. 6. Insert, edit, or reactivate probe points.

	7. (Optional) Import/Export probe points. 8. Test compile probe points by clicking on the button located in the upper right of the Probe Point Editor and the File Scope Probe Point Editor. 9. Save and apply the probe points
Output:	• Error message if fail to compile source code entered for probe points • Probe Points Report
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.12 Using Static Analysis Interface

Intended Purpose:	VectorCAST has the ability to integrate with static analysis tools. The static analysis tools help with detecting bugs, finding defects, and identifying unreachable code.

Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Static analysis tool available
Description:	1. Open environment. 2. Configure static analysis tool through the **Static Analysis** menu bar selection and then **Edit Analysis Tools**. 3. Use the VectorCAST provided template or select the custom Icon and Target paths. 4. Fill in unique name and associated arguments. 5. Select whether to add the tool to the Menu and or the Tool Bar and press **Apply**. 6. Run the static analysis tool through either the Menu Bar, the Tool Bar, or right clicking on a unit node or subprogram node in the Project Tree and select **Analyze Source => <Tool Name>**. 7. View the result of the analysis using either the Menu Bar, the Tool Bar, or right clicking on a unit node or subprogram node in the Project Tree and select **View Analysis => <Tool Name>**.
Output:	• Static Analysis Report • Error message if static analysis tool was configured incorrectly.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities.

	• The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.13 Building and Executing Test Case using Coded Tests

Intended Purpose:	Setup for unit testing of source code and execute coded tests in VectorCAST to test functions of a system component. View test results via passed/failed output.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Compiler must support C++11 or higher
Description:	1. Open environment. 2. Update the environment to include Enable Support for Coded Tests 3. A new node labeled Coded Tests is added to the environment 4. From that node, right click and select Create Coded Tests File and save 5. An example Coded Tests file source file will be added where coded tests can be created 6. Save the file after tests are added and execute the environment
Output:	• Available test cases are shown in the Test Case Tree. • Execution Status viewer automatically opens and provides detailed real time test case execution information. • Test Case Tree is updated with a PASS/FAIL designation after execution.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.

Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g. by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.1.14 Building and Executing Test Cases based on Test Variant Logic

Intended Purpose:	Build and execute selected test cases based on Test Variant Logic. View test results via passed/failed output.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system
Description:	1. Open environment with tests for multiple code variants. 2. A logic file can be added to a test environment by selecting the Variant Logic button in the tool bar. 3. Edit the logic file in the Logic Editor to define your Logics.Enter values for individual scalar data items. 4. Save the file and close the tab and the Logic Assignment tab will appear. 5. Select a test case and then select the drop-down menu next to the Assigned Logic: field to assign a Logic to a test case 6. Save the Assignments and run the test cases.

Output:	• Available test cases are shown in the Test Case Tree. • Execution Status viewer automatically opens and provides detailed real time test case execution information. • Test Case Tree is updated with a PASS/FAIL designation after execution.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g. by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2 VectorCAST/QA

2.2.1 Single User Setting Up an Initial VectorCAST Project

Intended Purpose:	A typical testing use case scenario is a single user setting up an initial VectorCAST project. Getting the initial project setup is the building block on which all other

Environmental, Functional and Process Constraints:	• VectorCAST installed • Valid license • VectorCAST does not display any warning when started • Writable permission for working directory • Existing source code
Description:	1. Start VectorCAST. 2. Create an empty project from **File => New => VectorCAST Project => Empty Project.** 3. Fill in the project name, select the Compiler from the drop-down menu and enter the path to the base directory. 4. **Create** the project and the Project Tree for the newly created project will open. 5. Customize the build configuration for the project by opening the Configuration Options Editor. 6. Create a new environment by right-click on the Group node in the Project Tree and **Create Unit Test Environment.** 7. Enter the environment name and other desired options. 8. Click Build when finished and the newly built environment will automatically open in the Environment View panel for test creation. 9. Add test cases to the subprograms. 10. Execute and verify that the test cases all pass. 11. View the Manage Report.
Output:	Manage Report Error messages
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale:

	• Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.2 Sharing VectorCAST Project with Entire Team

Intended Purpose:	This use case scenario puts a project under source code management and then having the team build on the project's initial configuration work. Regression tests are set up to automatically run each evening and generate result reports to be reviewed by the team each morning.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing source code management
Description:	1. VectorCAST configured to use the existing SCM (source code management) system. 2. Commit new or existing project to SCM. 3. Clone the repository of the project locally. 4. Create a new environment by right-click on the Group node in the Project Tree and **Create Unit Test Environment**. 5. Adds test cases for the subprograms. 6. Commit and push the changes following successful local build and execute for the new environment. 7. Create a script file to automatically run regression tests and generate a report that is saved in an accessible location by the team. 8. An execution scheduler can run the script each day or another desired period of time.
Output:	Execution report

Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.3 Building a VectorCAST Project from Existing Environments

Intended Purpose:	Many projects have a number of existing test environments. This scenario demonstrates building a new VectorCAST Project by leveraging a set of existing environments.
Environmental, Functional and Process Constraints:	• VectorCAST installed • Valid license • VectorCAST does not display any warning when started • Writable permission for working directory • Existing VectorCAST environments with no build errors

Description:	1. Create a project using **File => New => VectorCAST Project => From Existing Environments.** 2. Fill in the project name and select Existing Environments as the environment import source. 3. Use the **Add Search Directory Recursively** button to navigate to the parent directory of the existing environments to be imported. 4. Select the environments to be added from a list of test environments VectorCAST finds in specified search directories. 5. Before selecting the **Finish** button, use the right-click context menu on the nodes to add additional items, rename existing items or delete items. 6. The Project Tree automatically builds after selecting the **Finish** button. Execute tests by right-clicking on the node and selecting **Build/Execute => Full** from the context menu.
Output:	Execution report Error messages
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.

Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.4 Managing Configuration Options for Multiple Environments

Intended Purpose:	The ability to control test configurations across multiple test nodes is a powerful time-saving feature of the VectorCAST Project facility. In this use case the project architect applies common configuration options to all of the project's environments.
Environmental, Functional and Process Constraints:	• VectorCAST installed • Valid license • VectorCAST does not display any warning when started • Writable permission for working directory • Existing VectorCAST environments with no build errors
Description:	1. Setup common configuration options by selecting the desired environment and right-click on the Configuration node icon and select **Copy**. 2. Right-click in the Configuration Options column next to the node and select **Paste** from the context menu. A Configuration node icon will appear in the column. 3. Clear the Search Directories option from the common configuration. Right-click on the test suite Configuration node icon and select **Clear Single Option => Source Directories**. 4. Clear the elevated options from the individual environment nodes. Right-click on the Configuration icon next to any selected environment and select **Clear Single Option => Coverage Type** from the context menu. 5. Add a new test suite node and reconfigure options by duplicating the first test suite through right-click and selecting **Duplicate.** 6. Right-click on the Configuration node icon for the new test suite and select the desired **Coverage Type**. 7. Build and execute the new test suite.
Output:	Execution report Error messages
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements

	in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.5 Testing with Multiple Configurations

Intended Purpose:	The VectorCAST Project use case reuse existing project tests in a new configuration. The ability to easily share tests and test configurations improves workflow and increases team productivity.
Environmental, Functional and Process Constraints:	• VectorCAST installed • Valid license • VectorCAST does not display any warning when started • Writable permission for working directory • Existing VectorCAST project with no errors
Description:	1. Opening local project and right-click on the project root node and select **Add Compiler => (desired compiler options).** A new compiler node is displayed in the Project Tree.

	2. Drag and drop the original node to the newly created node. This clones all of the children of the original node to the new node. These are not copies of the original test suite but are the exact same set of tests as in the original node, being used in a new configuration. 3. Build and execute the nodes. The same tests have been run in tow different configurations. Any future changes made to a test in one configuration will update that test in all configurations. 4. Add a new environment through **Create Unit Test Environment** to the existing test suite and it is cloned to each configuration node where that test suite is used. 5. Execute test cases and verify expected results.
Output:	Execution report Error messages
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)

Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.6 Change-Based Testing

Intended Purpose:	The VectorCAST Project testing use case demonstrates making a change to a single line of source code and then using Change-Based Testing to automatically run only the sub-set of tests affected by the code change.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Environment is whitebox • Coverage instrumented • Environment has been built and executed previous to the changes in the unit under test (UUT) or dependent body without any errors. • A change is made to the UUT or a non-stubbed dependent.
Description:	1. Make a change in the source code. 2. Save and recompile the source code after the code change. 3. Select Environment->Incremental Rebuild from the menu for the corresponding environment. 4. VectorCAST determines what has changed, updates the associated portion of the test harness, and recompiles and then re-instruments those portions. 5. View the displayed result of the incremental rebuild via the Incremental Rebuild Report.
Output:	• Incremental Rebuild Report • Execution and coverage status are selectively removed following an incremental rebuild. • Unaffected test cases retain their execution and coverage data.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints:

	• Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.7 Sharing Tests and Results Between Multiple Users

Intended Purpose:	One of the key benefits of the VectorCAST Project is the ability to share tests and test results across a software project. In this use case, import results from the master project and uses those results to identify any regression in his suite of tests due to the previous day's changes. Any broken tests are fixed locally and pushed to the master project.
Environmental, Functional and Process Constraints:	• VectorCAST installed • Valid license • VectorCAST does not display any warning when started • Writable permission for working directory • Existing VectorCAST environments with no build errors • Existing source control management
Description:	1. Checks out a copy of the project from configuration control and makes a connection to the master project. 2. Open the local project that was just cloned. 3. Right-click on the **Import Results** node and select **Add.** 4. Enter the path to the master project and select **Import**.

	5. Results imported by the clone project's .vcm file are automatically refreshed every time the project is opened, or they can be manually refreshed by right-clicking on the Imported Results node and choosing **Refresh**. 6. Debug any test failure by building a local copy through **Build/Execute => Full**. 7. Examine the failure and fix any issues and rebuild and execute the tests. 8. VectorCAST automatically detect changes to the test script and displays the changes for review and acceptance. 9. Commit the changes to source code management.
Output:	Execution report Error messages Difference checker
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)

Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.8 Creating a Compiler Node Using an Existing .CFG File

Intended Purpose:	Projects often have existing CCAST_.CFG files that contain complex compiler and project settings. VectorCAST provides a feature allowing the import of .CFG files to create new Compiler nodes. In this use case, loading an existing .CFG file and creates a new compiler node where the team can begin creating new environments.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system • No existing error in previous .CFG file
Description:	1. Right-click on the Project root node and select **Add Compiler => From Configuration File**. Navigate to the location of the .cfg file and select the Open button. 2. Right-click on the new compiler node and select **Open Configuration** to open the Configuration Editor. 3. Double check the configuration options that VectorCAST has loaded from the existing .cfg file.
Output:	NA
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities.

	• The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.9 Using the Jobs Window and Job Monitor

Intended Purpose:	View output in real time when running a target test as the commands are running.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing test cases
Description:	1. Open the Jobs window, a tab (auto-hidden window) located at the bottom of the main window. 2. If the Jobs window is hidden, the status bar displays the current or last command run. Hover over the window to open it and use the pin icon to keep the window open. 3. Single-clicking on a command line highlights the associated line in the Messages window. Hovering over a command line shows the exit code and stdout and stderror for that line. 4. Open the Job Monitor by double-clicking a command line in the Jobs window or right-click a command line and select **Job Status**. 5. For compile or link errors that occurs in user code during test execution, click on the **Test Command** button to re-run the command displayed in the Job Status window. 6. View the automatically opened Exestuation Status viewer when a test case is executed. Click the **Show Details** button to display the full details of the execution in real time.
Output:	• Commands and their results
Tool Impact:	TI1 (No possibility of impact on a safety requirement) Rationale:

	• The viewing of execution status or commands has no bearing on system components, VectorCAST does not have an impact in safety requirements
Tool Error Detection:	TD1 (High degree of confidence of prevention or detection) • Helps display errors
Tool Confidence Level:	TCL1 (Determined from TI1 and TD1)
Recommended method of qualification	Not Required

2.2.10 Customizing Reports

Intended Purpose:	VectorCAST Reports can be customized to contain any information with any layout. Reports can be produced as either HTML or text and a PDF from the HTML version.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing test cases • Initialized coverages • Knowledge of CSS, HTML, Python, JSON
Description:	1. (Optional) Customize existing reports a. (Optional) Add and remove existing sections b. (Optional) Specify report sections to replace c. (Optional) Specify report section order 2. (Optional) Customizing Templates a. Create configuration directory b. Configure VectorCAST to use custom templates from the command line c. Add custom footer by editing the footer.html.template file d. Add custom logo to reports by editing the main.html.template file e. Customizing report style by create a custom style sheet in .css format and add it via **Tools => Options** and click the **Report** and then **Format** sub-tab or from the command line. 3. (Optional) Create new reports a. Create a main report file; the main entry point for the report with the initial data, the report name and describes the sections of the report and order in which they appear.

	b. Create a section file that contains the data that goes into the report c. Create a template which describes the layout of the data d. Run the new report from the command line or create a script to run the report and add it to Custom Tools
Output:	• Custom report generated if successful
Tool Impact:	TI1 (No possibility of impact on a safety requirement) Rationale: • Since the modification of VectorCAST report has no bearing on system components, VectorCAST does not have an impact in safety requirements • Modification of report does not alter test case values, only the display of those values.
Tool Error Detection:	TD1 (High degree of confidence of prevention or detection) • An error in format of report is highly visible and easy detected
Tool Confidence Level:	TCL1 (Determined from TI1 and TD1)
Recommended method of qualification	Not Required

2.2.11 System Test Automation

Intended Purpose:	VectorCAST provides a single point of control and reporting for all test activities. By combining Code Coverage, Change-Based Testing (CBT) and System Test functionalities into a single tool, VectorCAST/QA automates system testing and enables faster release cycles.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system • Understanding of Python
Description:	1. Edit the Configuration script, right-click on a Cover Environment node and select **System Testing => Edit Script**. 2. (Optional) Add a test case that is already defined in the source code to the configuration script. 3. (Optional) Add ManualTestCase to the Python configuration script. a. Add manual test entry to the Python list b. **Save** and **Run** manual test case c. In the Test Runner dialog, click on **Start Test** d. Record results by clicking on the appropriate **Pass** or **Fail** button

	4. (Optional) Run incremental build/execute automatically by right-click on the Environment node and select **Build/Execute => Incremental** 5. (Optional) Enabling component coverage a. Modify the python configuration script, **system_tests.py** b. Add in the list of components c. Build/Execute the components by performing a full **Build/Execute => Interactive** on the environment d. View the Incremental Rebuild Report 6. (Optional) Generate a Change Impact Report via **Project => Change Impact Report...**
Output:	• Error messages • Incremental Rebuild Report • Change Impact Report
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)

Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.12 Python API

Intended Purpose:	VectorCAST provides a Python API to allow programmatic access to test metrics data contained in a VectorCAST project. The Python API is commonly used to export test data to third party tools and to create custom test status reports.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system • Understanding of Python
Description:	1. Create or edit existing example Python script provided in **$VECTORCAST_DIR/python/examples**. 2. Generate a custom report listing the execution results for a VectorCAST Project. 3. The print_result.py file iterates a project and prints each field within the API. 4. Open a DOS command window and enter the command: *manage --project=[my project name] --python-script=%VECTORCAST_DIR%/python/examples/print_results.py* 5. When the script is located, the report is generated and output to standard I/O.
Output:	• Error messages • Custom script result
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale:

	• Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.13 Jenkins Integration

Intended Purpose:	VectorCAST and Jenkins integration enables users to implement a Continuous Test process, where each code change is completely tested against all existing test cases prior to integration. This results in the shortest possible feedback loop for the developer, allowing bugs to be identified within minutes of code changes, rather than weeks. Two plugins are available from the Jenkins website for use with VectorCAST projects: VectorCAST Execution and VectorCAST Coverage. The VectorCAST Execution plugin allows the user to create, delete and update jobs to build and run projects. The VectorCAST Coverage plugin allows the user to capture code coverage reports from projects.
Environmental, Functional and Process Constraints:	• Unit test environment is setup correctly • Valid license for feature usage • No existing error in VectorCAST or in operating system • Existing test cases • Jenkins is running and no existing errors
Description:	1. Download and install the VectorCAST plugin(s) from Jenkins. 2. Setup the VectorCAST Execution plugin for single checkout. 3. Create a new Jenkins job and schedule it within Jenkins. 4. Fill in the project paths, environment setup and SCM information. 5. Setup SCM checkout snippet by using the Jenkins snippet generator. 6. Build the pipeline job. 7. Review results and job artifacts.

Output:	• Results of job execution • Artifacts generated if successful • Trend graphs
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process such as system tests. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert. • Errors can be detected on multiple interfaces.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.14 Google Test Coverage

Intended Purpose:	The VectorCAST QA testing use case demonstrates connecting an existing Google Test framework to system test environment and applying coverage.

Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system • Environment is whitebox • Existing Google Test framework
Description:	1. Create a System Testing environment, select **File => New => System Testing Environment** from the Menu Bar 2. Choose the Compiler and enter options such as #defines, source file extensions, directories, preprocessor commands, etc. 3. Enter the required fields such as project and environment name. **4.** Select **Google Test** from the **System Test Template** drop down menu. 5. Locate the source file and select the coverage options. 6. Click **Build** to finish building the environment. 7. Right click on the newly created environment and select **System Testing => Edit Script** 8. Edit the opened python script to fill in the testing environment values such as *self.locationWhereWeRunMake, self.topLevelMakeCommand, self.locationWhereWeRunTests, self.nameOfTestExecutable.* Save the changes. 9. Build and execute the environment by right click on the environment name and then **Build/Execute => Full** 10. Open the environment to view coverage or coverage reports.
Output:	• Coverage displays or coverage reports if successful • Error messages if unsuccessful
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing of the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process. • Google Test framework has done initial testing. Rationale:

	• Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.15 VectorCAST with CUDA Applications

Intended Purpose:	VectorCAST/QA integrates with the NVIDIA® CUDA® development environment. VectorCAST is customized to handle both the CUDA C++ language extensions and the multi-GPU compile and link workflow.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing error in VectorCAST or in operating system • CUDA environment is available and free of errors
Description:	1. Preparing CUDA source for VectorCAST 1.1. Copy the original source code once for each architecture variant. 1.2. There will be one variant for each GPU device architecture and one for the CPU host. 1.3. Replace the original source file with an aggregator that uses #ifdef and #include logic to selectively include each variant copy during compilation. 1.4. The original source is backed up prior to replacement. 1.5. VectorCAST project will contain multiple variant copies of the original source, and each variant is processed for just one host/device architecture. 2. If not electing to append the c_cover_io.c source to one of the host-side source files (either a host variant of a CUDA source file, or a C or C++ file) then update the build system to compile and link.
Output:	• Testing and coverage are generated if successful

	• Processing will be stopped and error messages if unsuccessful
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process. • Google Test framework has done initial testing. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

2.2.16 Instrument Blocks for Statement Coverage

Intended Purpose:	Reduces the amount of program memory used. It is intended for users who have limited program memory for their application.
Environmental, Functional and Process Constraints:	• Valid license for feature usage • No existing errors in VectorCAST or in operating system • The option applies when instrumenting for Statement coverage and any aggregate coverage type that includes statement.

	• When importing coverage from one VectorCAST to another, both environments must have the same setting for the "Instrument blocks for statement coverage" option when the source files were instrumented.
Description:	1. Open chosen environment for program memory reduction. 2. Choose Tools => Options and click the Coverage tab. Then click the Options tab and the Instrumentation Options sub-tab if it is not selected. 3. Save the change.
Output:	• Processing will be stopped and error messages if unsuccessful • TESTINSS.DAT files and the instrumented files are smaller.
Tool Impact:	TI2 (Possibility of impact on a safety requirement) Rationale: • Since a malfunction of the system component can lead to a violation of a safety goal, VectorCAST may have an impact on the safety requirements in case of incorrect stimulation of values, incorrect checking of values and incorrect verdict displayed.
Tool Error Detection:	TD2 (Medium degree of confidence of prevention or detection) Necessary Constraints: • Securing the test setup by execution of an initial stress test and comparison of the observed with the expected results for this test. • The performed analysis is supported by additional tests later in the development process. Rationale: • Configuration changes of the used PC due to automatic updates of the operating system (e.g., by security updates) or installation of other applications leading to wrong logging and display behavior of VectorCAST can be detected with high confidence by the performed stress test done in advance of the intended test activities. • The initial stress test supports the tests already done by the tool manufacturer during the tool development. • An error in build/execution can be detected with medium confidence since errors might occur during the intended test activities which do not occur during the initial stress test of the test setup. • Performance analysis is viewed by more than one expert.
Tool Confidence Level:	TCL2 (Determined from TI2 and TD2)
Recommended method of qualification	For the qualifications of software tools classified at TCL2, the methods listed in *Table 5 – Qualifications of software tools classified TCL2* of the *ISO 26262-8:2018(E)* shall be applied.

3 Appendix

3.1 Glossary

Unit Testing	Testing a single unit under test. Normally a function, class or method of a class.
System Testing	Testing an entire application, program or firmware load.