

Cantata++

Standard briefing

ISO 26262

Road Vehicles - Functional Safety

Executive Summary

This paper is intended to serve as a reference to show how Cantata++ can be used to satisfy the verification and validation requirements of the draft international standard ISO 26262.

In particular it addresses the following sections of the standard:

Part 6: Product Development – Software Level

Part 8: Supporting Processes

(Section 11 Qualification of Software Tools)



Cantata++
intelligent testing

IPL

IPL Testing Products

Eveleigh House, Grove Street,
Bath BA1 5LR,
United Kingdom

Tel: +44 (0) 1225 475000

Fax: +44 (0) 1225 444400

Email: tools@ipl.com

www.ipl.com/cantata++

1 Introduction

This paper is intended to serve as a reference to show how Cantata++ can be used to satisfy the verification and validation requirements of the draft international standard ISO 26262 “Road vehicles — Functional safety”. In particular it addresses the following sections of the standard:

- Part 6: Product Development – Software Level
- Part 8: Supporting Processes (Section 11 Qualification of software tools)

This paper should be read in conjunction with the standard which is referred to throughout this paper as ISO 26262. The rest of this paper consists of three parts:

- Section 2 provides an overview of Cantata++ intelligent testing capabilities and links to current tool information. From supporting software unit design and implementation, software unit and software integration testing to verification of software safety requirements, Cantata++ allows automotive engineers to prove their code for all ISO 26262 Automotive Safety Integrity Levels (ASIL).
- Section 3 traces the requirements of ISO 26262-Part 6, identifying the scope of those which are supported by Cantata++ and documents how the requirements are supported by Cantata++.
- Section 4 traces the requirements of ISO 26262-Part 8, section 11, identifying the Cantata++ tool qualification evidence of suitability for use in the activities defined in ISO 26262-Part 6. A Cantata++ Tool Qualification Package containing all necessary tool qualification evidence (together with the automatically produced tool deployment zip) is available free of charge from your supplier.

If you have any questions on Cantata++ or its suitability for use for ISO 26262, please contact your supplier or IPL at tools@ipl.com.

NOTES:

To assist in cross-referencing, terms which are used by the standard are shown in *italics* throughout this paper.

For this document, the following legend is used:

- R Recommended activity
- HR Highly recommended activity

ASIL is the ISO 26262 risk-based approach for determining product risk classes. Risk classes are defined as Level A through D, with ASIL D representing the highest risk.

ISO 26262 is currently a draft international standard, in the last stages of acceptance and specific recommendations referred to herein may change in the final publication expected later in 2011.

2 General Information on Cantata++

Cantata++ provides advanced high productivity techniques, allowing developers to dynamically and repeatedly prove their C/C++ code with intelligent unit and integration testing, in the most cost effective manner.

The Cantata++ Eclipse user interface provides a complete intelligent unit and integration testing environment for the creation, execution and analysis of tests. It easily integrates with the developer desk top, compilers and embedded target platforms.

Unit testing and low level integration testing are the earliest tests performed during software development. Intelligent unit and integration testing with Cantata++ allows developers to fix software bugs at the least expensive time in the development lifecycle, and can be done before bugs pollute the wider code base, or affect the final shipped devices.

2.1 Cantata++ Major Features

Unit & Integration testing

- Supports C & C++ programming languages
- Testing on development host and embedded targets

Highly automated

- Re-usable test harness
- Test script generation written in C/C++
- Unique call interface control to simulate and intercept calls
- Automatic white-box accessibility
- Large data sets and robustness testing

Automated regression testing

Automatic generation of complete baseline of unit tests for legacy C code

Most advanced integrated code coverage analysis available for C/C++

2.2 Cantata++ Further Information

For current information on Cantata++ capabilities please refer to the product website: <http://www.ipl.com/cantata++> or contact your supplier.

3 Tracing Part 6 Requirements to Cantata++

The table below traces the requirements of ISO 26262-Part 6, identifying the scope of those which are supported by Cantata++ and documents how the requirements are supported by Cantata++.

Ref	Section Name	Scope	Cantata++
1	<i>Scope</i>	NA	NA
2	<i>Normative References</i>	NA	NA
3	<i>Terms, definitions, abbreviated terms</i>	NA	NA
4	<i>Requirements for compliance</i>	Definition of requirements	Where stated herein, use of Cantata++ provides evidence that the requirement is complied with.
5	<i>Initiation of product development at the software level</i>	Identification of testing with Cantata++	Selection of Cantata++ as a <i>software tool</i> to be used in support of <i>Software unit design and implementation</i> , and for <i>Software Unit Testing</i> and <i>Software Integration and Testing phases</i> (5.4.5) including <i>guidelines for their application</i> derived from the Cantata++ user documentation. Documentation of Cantata++ planned use in <i>Software verification plan</i> (5.5.2).
6	<i>Specification of software safety requirements</i>	NA	NA
7	<i>Software architectural design</i>	NA	NA
8	<i>Software unit design and implementation</i>	Static analysis of code	Cantata++ static analysis code metrics.

Ref	Section Name	Scope	Cantata++
8.1	<i>Objectives</i>	Third objective is to <i>verify the design of the software units and their implementation</i>	Cantata++ static analysis code metrics used to support verification of the design. NOTE: Cantata++ dynamic testing is recommended to verify the implementation (see sections 9 and 10 below)
8.2	<i>General</i>	Implementation <i>verified before proceeding to software unit testing phase.</i>	Cantata++ static analysis code metrics used on the implementation source code prior to dynamic unit testing.
8.3	<i>Inputs to this clause</i>		
8.3.1	<i>Prerequisites</i>	<i>Software verification plan</i> And <i>Software verification report (8.3.1.)</i>	Definition of Cantata++ static analysis code metrics to be used. Code metrics results reported.
8.3.2	<i>Further supporting information</i>	<i>Software tool application guidelines (8.3.2)</i>	Tool application guidelines derived from Cantata+ user documentation.
8.4	<i>Requirements and recommendations</i>		
8.4.1	<i>Requirements and recommendations</i>	NA	
8.4.2	<i>Requirements and recommendations</i>	NA	

Ref	Section Name	Scope	Cantata++
8.4.3	<i>Requirements and recommendations</i>	<p>Demonstration that <i>the implementation of the software unit shall achieve the following properties:</i></p> <ul style="list-style-type: none"> a) <i>Avoidance of unnecessary complexity</i> b) <i>testability; and</i> c) <i>maintainability</i> 	<p>Cantata++ code metrics used to demonstrate levels of: <i>complexity, testability and maintainability.</i></p> <p>Examples of such code metrics are:</p> <ul style="list-style-type: none"> • Halstead's Software Science metrics, • McCabe's, Myers' and Hansen's cyclomatic complexity metrics • Average and maximum nesting level. • Counts of language constructs (comments, lines of code, statements, parameters, etc) • Object oriented metrics sets
8.4.4.	<i>Requirements and recommendations</i>	<p>Demonstration that the design principles listed in <i>Table 9 shall be applied to achieve the following properties:</i></p> <ul style="list-style-type: none"> d) <i>Simplicity</i> f) <i>Robustness</i> h) <i>Testability during software unit testing</i> i) <i>No unconditional jumps</i> 	<p>Cantata++ code metrics used to demonstrate principles of: <i>simplicity, robustness and testability.</i></p> <p>See 8.4.3 for examples of Cantata++ code metric</p> <p>Cantata++ code metrics report functions containing number of GOTO statements, used and unused GOTO Labels</p>

Ref	Section Name	Scope	Cantata++
8.4.5	<i>Requirements and recommendations</i>	c) <i>the compliance of the source code with the coding standards</i>	Cantata++ code metrics assist in demonstration of compliance in with coding standards. Examples of such code metrics are: <ul style="list-style-type: none"> • Unreachable code • Switch statements with no defaults or fallthroughs • Number of GOTO statements, used and unused GOTO Labels • Lack of cohesion methods
Table 10		<i>1a - Informal Verification</i>	See Table 11 below
Table 11		<i>1e - Inspection of the source code 1f - Walkthrough of the source code</i>	Cantata++ static analysis code metrics can be used to assist both these methods for the informal verification of the software unit implementation
8.4.6	<i>Requirements and recommendations</i>	Ensuring that other included functions <i>cannot impair the compliance with the software safety requirements.</i>	Cantata++ static analysis does not add or include any software in the production release.
8.5	<i>Work products</i>	<i>Software verification report (refined) 8.5.3</i>	Cantata++ static analysis code metrics are provided at the function, class, translation unit, or system level, as appropriate and reported as .csv files which can be included in the <i>Software verification report (refined)</i>
9	<i>Software Unit Testing</i>	Unit testing	The main purpose of Cantata++ is software unit (and integration) testing.

Ref	Section Name	Scope	Cantata++
9.1	<i>Objectives</i>	<i>Demonstrate that the software units fulfil the software unit specifications and do not contain undesired functionality.</i>	<p>Cantata++ unit tests can demonstrate fulfilment of the requirement specifications.</p> <p>Cantata++ also automatically checks for unexpected exceptions, call orders and global data modifications to help detect undesired functionality.</p>
9.2	<i>General</i>	<i>Procedure for testing against unit specification and the tests are carried out in accordance with this procedure</i>	<p>Cantata++ test scripts can be used to document the detailed procedures for tests against specific requirements in each test case comment or for the whole test script.</p> <p>Cantata++ test scripts are human readable C/C++ source code which can be reviewed against an external procedure to verify that the tests contained therein are in accordance with the procedure.</p>
9.3	<i>Inputs to this clause</i>		
9.3.1	<i>Prerequisites</i>	<p><i>Software verification plan</i></p> <p>AND</p> <p><i>Software verification report</i></p>	<p>Definition of Cantata++ unit testing techniques to be used.</p> <p>Unit test results reported</p>
9.3.2	<i>Further supporting information</i>	<i>Software tool application guidelines</i>	Tool application guidelines derived from Cantata+ user documentation.
9.4	<i>Requirements and recommendations</i>		

Ref	Section Name	Scope	Cantata++
9.4.1	<i>Requirements and recommendations</i>	Planning, specification and executing unit tests	<p>Cantata++ test scripts can be <i>planned, specified and executed in accordance with ISO 26262-8 Clause 9</i></p> <p>Where test executables are built using any of the Cantata++ instrumentation technologies (test accessibility or coverage), the same tests can also be re-executed with this instrumentation disabled meaning that <i>the test objects in the software testing are the software units.</i>(NOTE 1)</p>

Ref	Section Name	Scope	Cantata++
9.4.2	<i>Requirements and recommendations</i>	<p>That methods in Table 12 demonstrate that the software units achieve:</p> <p>a) <i>compliance with the software unit design specification</i></p> <p>b) <i>compliance with the specification of hardware-software interface</i></p> <p>c) <i>correct implementation of the functionality</i></p> <p>d) <i>absence of unintended functionality</i></p> <p>e) <i>robustness</i></p> <p>f) <i>Sufficiency of the resources to support the functionality</i></p>	<p>Cantata++ can be used to demonstrate the following:</p> <p>Correct expected behaviour of the software unit (requirements based dynamic checks)</p> <p>Correct expected behaviour of the hardware software interface (dynamic checks using stubs isolates and wrappers)</p> <p>Correct implementation (dynamic white-box testing using correct call sequences and parameters/returns passed, correct read/write of global/static/private data)</p> <p>Dynamic checks white-box testing as above plus automatic checks for unexpected exceptions, and global data modifications to help detect undesired functionality.</p> <p>Cantata++ user defined table driven tests for large data sets plus automatic pre-defined robustness rule set tests generate input data for each data type. Cantata++ wrapping can also inject hard to trigger error conditions to verify error handling. Code coverage can also prove <i>the absence of inaccessible software</i>.</p> <p>Cantata++ Call Interface Control points can measure, check, and set resource sufficiency for unit tests. Cantata++ can also use static memory pools for resource validation.</p>

Ref	Section Name	Scope				Cantata++
Table 12	<i>Methods for Software Unit Testing</i>	ASIL				Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class
		A	B	C	D	
	<i>1a) Requirement-based test</i>	HR	HR	HR	HR	<p>Cantata++ intelligent testing capabilities make requirements based testing as efficient as possible. A test case template is provided for each function, with the GUI identifying all requirements derived input and expected output parameter variables, accessibility of global data, and calls made by the function. Required data values may be used singularly or in combination through the GUI or external CSV data.</p> <p>Requirements references may be tagged into test scripts or individual test cases and results in multiple formats to monitor which requirements were met, and which were not. Requirements tags in Cantata++ have been integrated with requirements management tools such as DOORS, REQTIFY etc.</p>
	<i>1b) Interface test</i>	HR	HR	HR	HR	<p>Cantata++ provides powerful Call Interface Control for function call interfaces when one function calls another. Provided as stubs and isolates to simulate or wrapping to intercept these calls. Each has programmable instances to perform different test scenarios on each occasion a call is made. Functions (including static) can be called directly from the test script with precise, boundary or ranges of valid and invalid inputs as either user defined values or predefined according to data type.</p>

Ref	Section Name	Scope				Cantata++				
	<i>1c) Fault injection test</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>HR</i>	<p>Faults may be injected by modification of corrupt values of function parameters, global or static data variables, at any function point in the test script including Call Interface Control. Getters and Setters for Registers may also be used at these points for fault injection.</p> <p>Cantata++ tests can also run under the control of debuggers for yet finer fault injection.</p>				
	<i>1d) Resource usage test</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>HR</i>	<p>Partial control is offered through static memory pools in tests, and additional tools can be used to measure resource consumption on targets.</p>				
	<i>1e) Back-toBack Test between Model and Code (if applicable)</i>	<i>R</i>	<i>R</i>	<i>HR</i>	<i>HR</i>	<p>Cantata++ is integrated with modelling tools such Rhapsody®, so that auto-generated code can be verified from the tool, or can be alongside modelling tools such as Simulink® to verify auto-generated code outside the model.</p>				
9.4.3	<i>Requirements and recommendations</i>	<i>That test case shall be derived using the methods listed in Table 13</i>				<p>Cantata++ supports each of the methods for deriving test cases in Table 13.</p>				
Table 13	<i>Methods for Deriving Test Cases for Software Unit Testing</i>	<p style="text-align: center;">ASIL</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; text-align: center; width: 25%;">A</td> <td style="border: 1px solid black; text-align: center; width: 25%;">B</td> <td style="border: 1px solid black; text-align: center; width: 25%;">C</td> <td style="border: 1px solid black; text-align: center; width: 25%;">D</td> </tr> </table>				A	B	C	D	<p>Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class</p>
A	B	C	D							
	<i>1a. Analysis of Requirements</i>	<i>HR</i>	<i>HR</i>	<i>HR</i>	<i>HR</i>	<p>Requirements based test cases can be identified using a requirements tag for each test case.</p>				

Ref	Section Name	Scope				Cantata++
	1b. Generation and Analysis of Equivalence Classes	R	HR	HR	HR	Equivalence classes may be generated through user defined inputs or robustness Rule Sets of multiple values based on data type for each parameter on an interface. Checks can be set for specific values or ranges.
	1c. Analysis of Boundary Values	R	HR	HR	HR	Boundary analysis can be done using Cantata++ checks, and also measured using operator code coverage.
	1d. Error Guessing	R	R	R	R	<p>These test cases <i>can be based on data collected through a “lessons learned” process and expert judgment.</i></p> <p>Cantata++ provides a range of capabilities for ‘what if’ error guessing scenarios, such as raised exceptions, uninitialized pointers. Such ‘crash’ inducing test cases can be enabled / disabled to prevent pollution of regression suites.</p>
9.4.4	<i>Requirements and recommendations</i>	<p><i>structural coverage shall be measured in accordance with the metrics listed in Table 14</i></p> <p><i>Note 3 (instrumented code)</i></p>				<p>Cantata++ supports all the structural coverage metrics from Table 14 (plus others).</p> <p>Cantata++ code coverage data is produced in various graphical diagnostic formats, and Rule Sets embed coverage targets within dynamic test checks.</p> <p>Where test executables are built using any of the Cantata++ instrumentation technologies (test accessibility or code coverage), the same tests can also be re-executed with this instrumentation disabled</p>

Ref	Section Name	Scope	Cantata++				
		<p><i>Note 4 (coverage rationale)</i></p> <p><i>A rationale is to be given for the level of coverage achieved (e.g. for accepted dead code or code segments depending on different software configurations), or else code not covered can be verified using complementary methods (e.g. inspections).</i></p>	<p>Cantata++ identifies uncovered code in various graphical views for analysis and plan text reports for certification evidence.</p> <p>Where code is only executed according to different configurations, Cantata++ coverage by test case reports can assist documenting such a rationale.</p> <p>Cantata++ user defined contexts for code coverage may also be used for such purposes.</p>				
<p><i>Table 14</i></p>	<p><i>Structural Coverage Metrics at the Software Unit Testing</i></p>	<p>ASIL</p> <table border="1" style="margin: auto;"> <tr> <td style="width: 20px;">A</td> <td style="width: 20px;">B</td> <td style="width: 20px;">C</td> <td style="width: 20px;">D</td> </tr> </table>	A	B	C	D	<p><i>Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class</i></p>
	A	B	C	D			
	<p><i>1a. Statement Coverage</i></p>	HR	HR	HR	HR	<p>Cantata++ provides Statement coverage</p>	
	<p><i>1b. Branch Coverage</i></p>	R	HR	HR	HR	<p>Cantata++ provides Decision Coverage</p>	
<p><i>1c. MC/DC (Modified Condition/ Decision Coverage)</i></p>	R	R	R	HR	<p>Cantata++ provides Boolean Operand Effectiveness coverage (both masking and unique cause variants), which with Decision coverage is MC/DC coverage.</p> <p>A truth table is provided for each Boolean Operand within the Cantata++ Coverage Viewer.</p>		

Ref	Section Name	Scope	Cantata++
9.4.5	<i>Requirements and recommendations</i>	<p>Test Environment</p> <p>Note 3 (different execution environments)</p>	<p>The same Cantata++ tests can be executed on host-native, simulators, emulators and on the processor of the target system.</p> <p>Cantata++ can be used for driving <i>software-in-the loop, processor-in-the-loop, or hardware-in-the-loop</i> tests.</p>
9.5	<i>Work products</i>	<p><i>Software verification plan (refined) 9.5.1</i> AND <i>Software verification specification 9.5.2</i></p> <p><i>Software verification report (refined) 9.5.1</i></p>	<p>The Software verification plan and Software verification specification will include the definition of test cases (using Cantata++ capabilities) and expected results for the software unit.</p> <p>Cantata++ test results in various formats are available for inclusion in or acting as the <i>Software verification report</i> for each unit tested. The plain text CTR results file direct from the target is most suitable for qualifiable tool results.</p>
10	<i>Software integration and testing</i>	Integration Testing	The main purpose of Cantata++ is software integration (and unit) testing.
10.1	<i>Objectives</i>	<i>Demonstrate that the software architectural design is correctly realised by the embedded software</i>	Cantata++ integration tests can demonstrate the correct realisation of the architectural design in the integrated embedded software.

Ref	Section Name	Scope	Cantata++
10.2	<i>General</i>	Tests against architectural design and the interfaces between software units and components	Cantata++ tests scripts can be used to verify the correctness of the architectural design and through wrapping interception verify the correctness of the actual interfaces between any unit or component within or external to the integrated software.
10.3	<i>Inputs to this clause</i>		
10.3.1	<i>Prerequisites</i>	<i>Software verification plan, Software verification specification AND Software verification report</i>	Definition of Cantata++ integration testing techniques to be used. Definition of test cases (using Cantata++ capabilities) and expected results for the integrated software. Integration test results reported.
10.3.2	<i>Further supporting information</i>	<i>Software tool application guidelines</i>	Tool application guidelines derived from Cantata+ user documentation.
10.4	<i>Requirements and recommendations</i>		
10.4.1	<i>Requirements and recommendations</i>	Planning integration hierarchically	Cantata++ can support any level or sequence of planned integrations. Each integration test would consist of a Cantata++ test executable or a series of executables under the control of a Cantata++ test script.

Ref	Section Name	Scope	Cantata++
10.4.2	<i>Requirements and recommendations</i>	Planning, specification and executing integration testing	<p>Cantata++ test scripts can be <i>planned, specified and executed in accordance with ISO 26262-8 Clause 9</i></p> <p>Where test executables are built using any of the Cantata++ instrumentation technologies (test accessibility or coverage), the same tests can also be re-executed with this instrumentation disabled meaning that <i>the test objects in the software testing are the software units.(NOTE 1)</i></p>
10.4.3	<i>Requirements and recommendations</i>	<p>That methods in Table 15 shall be applied to demonstrate that both the software components and the embedded software achieve:</p> <p>a) compliance with the software architectural design</p> <p>b) compliance with the specification of the hardware-software interface</p> <p>c) correct implementation of the functionality</p>	<p>Cantata++ can be used to demonstrate:</p> <p>Correct expected behaviour of the architectural design (requirements based dynamic checks)</p> <p>Correct expected behaviour of the hardware software interface (dynamic checks using stubs and wrappers)</p> <p>Correct implementation (dynamic white-box testing using correct call sequences and parameters/returns passed, correct read/write of global/static/private data)</p>

Ref	Section Name	Scope					Cantata++
		<p><i>d) robustness</i></p> <p><i>e) sufficiency of the resources to support the functionality</i></p>					<p>Cantata++ call sequences and Code coverage will prove the <i>absence of inaccessible software</i>. Cantata++ wrapping can also inject hard to trigger error conditions to verify error handling.</p> <p>Cantata++ Call Interface Control points can measure, check, and set resource sufficiency for unit tests. Cantata++ can also use static memory pools for resource validation.</p>
Table 15	Methods for Software Integration Testing	ASIL					<i>Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class</i>
		A	B	C	D		
	<i>1a. Requirement-Based Test</i>	<i>HR</i>	<i>HR</i>	<i>HR</i>	<i>HR</i>		<p>As per Table 12.</p> <p>Except that multiple units can be verified together. Here the sequence of calls and data manipulations is important between units, and can be very easily verified using Expected Calls, and checks on data at all the test / call control points.</p>
	<i>1b. Interface Test</i>	<i>HR</i>	<i>HR</i>	<i>HR</i>	<i>HR</i>		<p>As per Table 12</p> <p>Note that Cantata++ Wrapping call interface interceptions, provides interception control for function calls within the compilation boundary, which is not possible with stubs. This means that the same powerful degree of control can be exercised at both unit and integration levels of testing.</p>
	<i>1c. Fault Injection Test</i>	<i>R</i>	<i>R</i>	<i>HR</i>	<i>HR</i>		As per Table 12

Ref	Section Name	Scope				Cantata++
	<i>1d. Resource Usage Test</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>HR</i>	As per Table 12
	<i>1e. Back-to-Back Test between Model and Code (if applicable)</i>	<i>R</i>	<i>R</i>	<i>HR</i>	<i>HR</i>	As per Table 12

Ref	Section Name	Scope	Cantata++			
10.4.4	<i>Requirements and recommendations</i>	<i>test methods shall be derived using the methods listed in Table 16</i>	Cantata++ supports each of the methods for deriving test cases in Table 16.			
Table 16	<i>Methods for Deriving Test Cases for Software Integration Testing</i>	ASIL				<i>Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class</i>
		A	B	C	D	
	<i>1a. Analysis of Requirements</i>	HR	HR	HR	HR	Requirements based test cases can be identified using a requirements tag for each test case.
	<i>1b. Generation and Analysis of Equivalence Classes</i>	R	HR	HR	HR	Equivalence classes may be generated through user defined inputs or robustness Rule Sets of multiple values based on data type for each parameter on an interface. Checks can be set for specific values or ranges.
	<i>1c. Analysis of Boundary Values</i>	R	HR	HR	HR	Boundary analysis can be done using Cantata++ checks, and also measured using operator code coverage.
<i>1d. Error Guessing</i>	R	R	R	R	<p>These test cases can be based on data collected through a “lessons learned” process and expert judgment.</p> <p>Cantata++ provides a range of capabilities for ‘what if’ error guessing scenarios, such as raised exceptions, uninitialized pointers. Such ‘crash’ inducing test cases can be enabled / disabled to prevent pollution of regression suites.</p>	

Ref	Section Name	Scope	Cantata++																
10.4.5	<i>Requirements and recommendations</i>	<i>structural coverage shall be measured in accordance with the metrics listed in Table 17</i>	Cantata++ supports all the structural coverage metrics from Table 17 (plus others). Cantata++ code coverage data is produced in various graphical diagnostic formats, and RuleSets embed coverage targets within dynamic test checks.																
Table 17	<i>Structural Coverage Metrics at the Software Architectural Level</i>	<table border="1" style="margin: auto;"> <thead> <tr> <th colspan="4">ASIL</th> </tr> <tr> <th>A</th> <th>B</th> <th>C</th> <th>D</th> </tr> </thead> <tbody> <tr> <td>R</td> <td>R</td> <td>HR</td> <td>HR</td> </tr> <tr> <td>R</td> <td>R</td> <td>HR</td> <td>HR</td> </tr> </tbody> </table>	ASIL				A	B	C	D	R	R	HR	HR	R	R	HR	HR	<i>Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class</i>
	ASIL																		
	A	B	C	D															
R	R	HR	HR																
R	R	HR	HR																
<i>1a. Function Coverage</i>	R	R	HR	HR	Function coverage is reported as Cantata++ Entry-point coverage.														
<i>1b. Call Coverage</i>	R	R	HR	HR	Call coverage is reported as Cantata++ Call-pair coverage														
10.4.6	<i>Requirements and recommendations</i>	<i>Unspecified software components shall be identified.. removed or deactivated.</i>	Cantata++ coverage identifies untested code as a result of specification (requirements based) test cases which can then be removed or deactivated.																
10.4.7	<i>Requirements and recommendations</i>	Test Environment Note 3 (different execution environments)	The same Cantata++ tests can be executed on host-native, simulators, emulators and on the processor of the target system. Cantata++ can be used for driving <i>software-in-the loop, processor-in-the-loop, or hardware-in-the-loop</i> tests.																

Ref	Section Name	Scope	Cantata++
10.5	<i>Work products</i>	<p><i>Software verification plan (refined) 10.5.1</i></p> <p><i>Software verification specification (refined) 10.5.2</i></p> <p><i>Software verification report (refined) 10.5.4</i></p>	<p>The <i>Software verification plan and Software verification specification</i> will include the definition of test cases (using Cantata++ capabilities) and expected results for the integrated embedded software.</p> <p>Cantata++ test results in various formats are available for inclusion in or acting as the <i>Software verification report</i> for each unit tested. The plain text CTR results file direct from the target is most suitable for qualifiable tool results</p>
11	<i>Verification of software safety requirements</i>	Safety Requirements	While the main purpose of Cantata++ is dynamic software testing for functional and non-functional behaviour, tests can also demonstrate the correctness of software safety requirements.
11.1	<i>Objectives</i>	<i>Demonstrate that the embedded software fulfils the software safety requirements</i>	Cantata++ unit and integration tests can be used to demonstrate fulfilments of safety requirements at any stage of testing.
11.2	<i>General</i>	Tests against the safety requirements	Cantata++ tests executed on the target environment can be used to demonstrate through dynamic checks that the embedded software satisfies its safety requirements.
11.3	<i>Inputs to this clause</i>		

Ref	Section Name	Scope	Cantata++
11.3.1	<i>Prerequisites</i>	<i>Safety plan, Software verification plan (refined), Software verification specification, Software verification report, Integration testing report</i>	<p>Definition of safety testing approaches with Cantata++ and testing techniques to be used.</p> <p>Definition of test cases (using Cantata++ capabilities) and expected results for the embedded software.</p> <p>Integration test results reported.</p>
11.3.2	<i>Further supporting information</i>	<p><i>Validation plan (refined),</i></p> <p><i>Guidelines for the application of methods</i></p> <p><i>Software tool application guidelines</i></p>	<p>Definition of validation approaches with Cantata++ and testing techniques to be used.</p> <p>Tool application guidelines derived from Cantata+ user documentation.</p>
11.4	<i>Requirements and recommendations</i>		
11.4.1	<i>Requirements and recommendations</i>	Planning, specification and executing safety verification	Cantata++ test scripts can be <i>planned, specified and executed in accordance with ISO 26262-8 Clause 9</i>
11.4.2	<p><i>Requirements and recommendations</i></p> <p>Table 18</p>	<p><i>Tests shall be conducted in the test environments listed in Table 18</i></p> <p><i>1a - Hardware-in-the-loop</i></p>	Cantata++ tests executed on the target environment can be used to demonstrate fulfilment of software safety requirements at unit and integration levels with hardware in the loop.

Ref	Section Name	Scope	Cantata++
		<i>1b – Electronic control unit network environments</i>	Cantata++ test scripts can be used to dynamically check fulfilment of software safety requirements in ECU network simulators or test benches, by either driving input data, verifying expected outputs or both.
11.4.3	<i>Requirements and recommendations</i>	<i>Tests shall be executed on the target hardware</i>	All Cantata++ tests may be executed on the target hardware
11.4.4	<i>Requirements and recommendations</i>	<p><i>Safety verification shall be evaluated in accordance with:</i></p> <p><i>a) compliance with the expected results</i></p> <p><i>b) coverage of the safety related requirements</i></p> <p><i>c) a pass or fail criteria</i></p>	<p>Cantata++ tests used for safety verification can be used to demonstrate:</p> <p>Detailed test results comparing the expected with the actual.</p> <p>Safety related requirements coverage verified in a test script through a comment tag for each test case or for the whole test script.</p> <p>All Cantata++ checks are report clear pass or fail criteria for actual results against expected results.</p>
11.5	<i>Work products</i>	<p><i>Software verification plan (refined) 11.5.1</i></p> <p><i>Software verification specification (refined) 11.5.2</i></p>	The <i>Software verification plan and Software verification specification</i> will include the definition of test cases (using Cantata++ capabilities) and expected safety test results for the embedded software.

Ref	Section Name	Scope	Cantata++
		<i>Software verification report (refined)</i> 10.5.3	Cantata++ test results in various formats are available for inclusion in or acting as the <i>Software verification report</i> for each unit tested. The plain text CTR results file direct from the target is most suitable for qualifiable tool results.

4 Tracing Part 8 Requirements to Cantata++

The table below traces the requirements of ISO 26262-Part 8 (Section 11 Qualification of software tools) identifying the Cantata++ tool qualification evidence of suitability for use in the activities defined in ISO 26262-Part 6. To assist in cross-referencing, terms which are used by the standard are shown in *italics* throughout this paper.

Ref	Section Name	Scope	Cantata++
11	<i>Qualification of software tools</i>		
11.1	<i>Objectives</i>		Cantata++ tool qualification comprising the Tool Qualification Package and a Deployment .zip file provides the necessary evidence for the activities identified in Part 6.
11.2	<i>General</i>		This document, the tool user documentation and the Tool Qualification Package provides the materials necessary for determining the tool confidence level.
11.3	Inputs to this clause		
11.3.1	<i>Prerequisites</i>	<i>Pre-determined maximum ASIL</i> <i>Safety Plan (Part 4 - 5.5.2)</i> <i>Validation Plan (Part 4 - 5.5.2)</i>	NA The specific version of Cantata++ should be identified within these documents, together with the activities identified in Part 6 for which Cantata++ will be used.
11.3.2	<i>Further supporting information</i>	<i>User Manual</i> <i>Environment and constraints of the software tool</i>	The Cantata++ complete User Manual PDF is contained in the Tool Qualification Package. The Cantata++ Deployment zip documents these for tool qualification


Ref	Section Name	Scope	Cantata++
11.4	Requirements and recommendations		
11.4.1	<i>General</i>		<p>The classification of Cantata++ may be performed independently of the development of a safety-related item.</p> <p>Validity of the classification or qualification may be confirmed prior to its use on a host development environment from the Tool Qualification Package, and for an embedded target environment once the automatic Deployment Tests for that specific environment have passed and the deployment is registered with the supplier.</p>
11.4.2.1	<i>Planning of qualification of a software tool</i>	<ul style="list-style-type: none"> a) Unique id and version b) Configuration c) Use-cases d) Environment in which the tool is executed e) Pre-defined maximum ASIL f) Methods to qualify the tool 	<p>Cantata++ build number is the unique identifier of the tool version</p> <p>Cantata++ deployment zip contains an HTML report on the exact configuration of the tool. The specific options used in any test are recorded in the (ipg.cop) options file for each test.</p> <p>The tool use cases for automating the activities identified in Part 6, should be documented in the appropriate plans.</p> <p>Cantata++ uses a checksum to confirm correct use of the tested specific target deployment.</p> <p>The appropriate plans will identify the maximum ASIL for which the tool shall be used.</p> <p>It is recommended that Cantata++ be qualified using 1b from Table 4</p>

Ref	Section Name	Scope	Cantata++
11.4.2.2	<i>Planning of qualification of a software tool</i>	a) Description of the features, functions and technical properties	The Cantata++ Product Overview provides a high level tool description. Detailed description is documented in the Manual. Both are included in the Tool Qualification Package
		b) Description of the installation process	The installation process is described in the Cantata++ Installation Guide supplied with the product.
		c) The user Manual	The Cantata++ complete User Manual PDF is contained in the Tool Qualification Package.
		d) Description of the environment required for operation	The environment required for operation of Cantata++ is automatically documented in the HTML Deployment Report contained in the Deployment Zip
		e) Description of expected behaviour under anomalous conditions	Behaviour under anomalous conditions is identified by Script Errors described in the User Manual.
		f) Description of known malfunctions and safeguards / workarounds	Known malfunctions affecting the validity of Cantata++ results are recorded in the Problem Tracker together with workarounds, and safeguards are documented in the Overview Usage Guidelines, both part of the Tool Qualification Package.
		g) Measures for detection of erroneous outputs	The Tool Qualification Package identifies measures which can be taken to detect malfunctions or erroneous outputs.

Ref	Section Name	Scope	Cantata++
11.4.3	<i>Classification of a software tool</i>	<p>a) Intended purpose</p> <p>b) output</p> <p>c) environmental and functional constraints</p> <p>Tool Impact Classification</p> <p>Tool error Detection Classification</p> <p>Tool Confidence Level</p>	<p>Cantata++'s intended purpose will be Static Code Metrics analysis, unit and integration testing and code coverage activities which are identified in the relevant planning documents.</p> <p>The Tool Qualification Package identifies all the necessary Cantata++ outputs to control and submit as verification evidence</p> <p>The Tool Qualification Package Overview Usage Guidelines identify these constraints</p> <p>As there is a possibility that erroneous output would result in a safety-related requirement not being met, Cantata++ should be classified TI 1.</p> <p>TD2 should be chosen as there is a medium degree of confidence that an erroneous output will be prevented or detected in Cantata++. The rationale for not selecting a high degree of confidence is simple that unit and integration testing are likely to be the very last verification stage on much safety-related software, so there is no further back-up stage. This is therefore the conservative estimate on the level of TD classification as advised by <i>Notes 4 and 5</i></p> <p>Cantata++ should be classified as TCL 2</p>

Ref	Section Name	Scope	Cantata++
11.4.4	<i>Qualification of a software tool</i>	Qualification shall be documented a) to f)	<p>The methods in Table 4 should be used for Cantata++ as a TCL 2 tool.</p> <p>The Cantata++ Tool Qualification Package supports the information requirements for documenting the tool qualification</p>

Ref	Section Name	Scope				Cantata++
Table 4	<i>Qualification of software tools classified TCL2</i>	ASIL				<i>Requirements vary by Automotive Safety Integrity Levels (ASIL) risk class</i>
		A	B	C	D	
	1a) <i>Increased confidence from use</i>	HR	HR	HR	HR	Cantata++ has been used in many automotive safety-related projects, as well as widely in many other safety related industries for SIL3/4 IEC61508 and Level A DO-178B project. However, as very few users of Cantata++ employ precisely the same version and configuration of Cantata++ in identical development environments, this method is not recommended.
	1b) Evaluation of the development process	HR	HR	HR	HR	The Tool Qualification Package together with the Deployment zip (produced when deploying the tool to a target environment) provides all the necessary evidence to enable detailed evaluation of the development process.
	1c) Validation of the software tool	R	R	R	R	The Tool Qualification Package provides guidance on how validation of each capability of the tool may be undertaken by the end user.
	1d) Development in compliance with a safety standard	R	R	R	R	Cantata++ is developed under a QMS which supports safety standards such as IEC 61508 and DO0178B, but does not comply with them in all respects. Should this method be used in addition to 1 b), Cantata++ can be site audited by an end user for a fee.

Ref	Section Name	Scope	Cantata++
11.4.5	<i>Increased confidence from use</i>		Cantata++ has been used in many automotive safety-related projects, as well as widely in many other safety related industries for SIL3/4 IEC61508 and Level A DO-178B project. However, as very few users of Cantata++ employ precisely the same version and configuration of Cantata++ in identical development environments, this method is not recommended.
11.4.6	<i>Evaluation of the development process</i>		<p>Cantata++ is developed and maintained under Quality Management System certificated to ISO 9001:2008 by LRQA:</p>  <p>The Tool Qualification Package contains QMS process documentation, evidence and audit trails for Cantata++ to facilitate evaluation and assessment.</p>
11.4.7	<i>Validation of the software tool</i>		The Tool Qualification Package contains evidence of all tests carried out on the Cantata++ tool, and is supplemented by the Deployment Tests used to verify the correct deployment on the target environment. The Tool Qualification Package also provides guidance on how validation of each capability of the tool may be undertaken by the end user.
11.4.8	<i>Verification of qualification of software tools</i>		The Tool Qualification Package provides sufficient information to support the verification of the tool qualification.

Ref	Section Name	Scope	Cantata++
11.5	<i>Work products</i>	<p><i>Software tool classification analysis 11.5.1</i></p> <p><i>Software tool qualification plan 11.5.2</i></p> <p><i>Software tool documentation 11.5.3</i></p> <p><i>Software tool qualification report 11.5.4</i></p>	<p>The Tool Qualification Package and Deployment Zip provide all the necessary information to produce the listed Work Products.</p>