

# IPL Testing Tools and IEC 61508

## Executive Summary

This paper describes how AdaTEST 95 and Cantata++ can be used to assist with the development of software to the standard IEC 61508: Functional Safety - Safety-Related Systems. In particular, it shows how AdaTEST 95 and Cantata++ can be used to meet the verification and testing requirements of this standard. It also shows that the tools have been produced to a high standard, such that their use for dynamic testing will not compromise the safety and integrity of the software being tested.

The material presented here is suitable for inclusion in a justification for the use of the products on a safety critical software development.

IPL is an independent software house founded in 1979 and based in Bath. IPL was accredited to ISO9001 in 1988, and gained TickIT accreditation in 1991. Both AdaTEST 95 and Cantata++ have been produced to these standards.

1.

### Copyright

This document is the copyright of IPL Information Processing Ltd. It may not be copied or distributed in any form, in whole or in part, without the prior written consent of IPL.

*IPL  
Eveleigh House  
Grove Street  
Bath  
BA1 5LR  
UK  
Phone: +44 (0) 1225 475000  
Fax: +44 (0) 1225 444400  
email: [ipl@iplbath.com](mailto:ipl@iplbath.com)*



Certificate Number FM 01589

## 2. Introduction

AdaTEST 95 and Cantata++ are tools which support the dynamic testing, dynamic (coverage) analysis, and static analysis of Ada, C and C++ software. The original requirements for the tools included the suitability for testing safety critical software, although both tools are sufficiently flexible for the testing of any Ada, C or C++ software.

This paper describes how AdaTEST 95 and Cantata++ can be used to enable a software development to meet many of the verification and testing requirements of IEC 61508: Functional Safety - Safety-Related Systems. It should be read in conjunction with the standard which is referred to throughout this paper as IEC 61508.

To assist in cross-referencing, terms which are used by the standard are shown in *italics* in this text. Where clause numbers of the standard are referenced, these have been checked to be valid against IEC 61508-3:2001, dated December 2001.

The description consists of two parts. Section 3 looks at AdaTEST 95 and Cantata++ as tools to facilitate developing software to meet the requirements of the standard. Section 4 looks at the integrity of the tools themselves, including standards used during their development.

## 3. AdaTEST 95 and Cantata++ - Capabilities and Use

### 3.1. Overview of Requirements

The IEC 61508 standard is structured as 7 parts which identify process issues, techniques and measures applicable to all aspects of functional safety. The parts which identify validation, verification and test requirements which are relevant to AdaTEST 95 and Cantata++ are:

(a) IEC 61508 Part 3: Software Requirements;

<i>CLAUSE 7.1</i>	- <i>Software Safety Lifecycle-General</i>
<i>CLAUSE 7.3</i>	- <i>Software Safety Validation Planning</i>
<i>CLAUSE 7.4.1</i>	- <i>Software Design and Development - Objectives</i>
<i>CLAUSE 7.4.4</i>	- <i>Requirements for Support Tools and Programming Languages</i>
<i>CLAUSE 7.4.5</i>	- <i>Requirements for Detailed Design and Development</i>
<i>CLAUSE 7.4.6</i>	- <i>Requirements for Code Implementation</i>
<i>CLAUSE 7.4.7</i>	- <i>Requirements for Software Module Testing-</i>
<i>CLAUSE 7.4.8</i>	- <i>Requirements for Software IntegrationTesting</i>
<i>CLAUSE 7.5</i>	- <i>Programmable Electronics Integration</i>
<i>CLAUSE 7.6</i>	- <i>Software Operation and Modification Procedures</i>
<i>CLAUSE 7.7</i>	- <i>Software Safety Validation</i>
<i>CLAUSE 7.8</i>	- <i>Software Modification</i>
<i>CLAUSE 7.9</i>	- <i>Software Verification</i>
<i>CLAUSE 8</i>	- <i>Functional Safety Assessment</i>
<i>ANNEX A</i>	- <i>Guide to the Selection of Techniques and Measures</i>
<i>ANNEX B</i>	- <i>Detailed Tables</i>

(b) IEC 61508 Part 7: Overview of Techniques and Measures.

### 3.2. General Description of AdaTEST 95 and Cantata++

AdaTEST 95 and Cantata++ support the analysis and testing of Ada, C, and C++ software at all levels, from module test through to full integration testing. The facilities provided by AdaTEST 95 and Cantata++ are summarised in Table 1.

Testing		Analysis	
Test Preparation	Dynamic Testing	Static Analysis	Dynamic Analysis
<ul style="list-style-type: none"> <li>• Scripts in Ada, C or C++</li> <li>• Test Script wizards</li> <li>• Script generation from test definition data<sup>*2</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Test execution</li> <li>• Result collection</li> <li>• Result verification</li> <li>• Timing analysis</li> <li>• Stub simulation</li> <li>• External call wrapping<sup>*1</sup></li> <li>• Host testing</li> <li>• Target testing</li> </ul>	<ul style="list-style-type: none"> <li>• Metrics: Code Counts Code Complexity</li> <li>• Metrics in formats: CSV List file<sup>*2</sup> Dynamically checkable<sup>*2</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Coverage: Entry points Statements Decisions Conditions MC/DC Call-Pairs<sup>*1</sup> Exceptions<sup>*2</sup></li> <li>• Trace</li> <li>• Assertions<sup>*2</sup></li> <li>• Paths<sup>*2</sup></li> </ul>

**Table 1 - AdaTEST 95 and Cantata++ Facilities**

\*1: Cantata++ only \*2: AdaTEST 95 only

Testing with AdaTEST 95 and Cantata++ is centred on a dynamic test harness. The test harness can be used to support testing at all levels, from module test through to full integration testing, in both host and target environments. In the host environment tests can either be run directly or under the control of a debugger. In the target environment, tests can be run directly, under an emulator, or under a debugger.

Tests are controlled by a structured test script. Test scripts are written in the respective application language, Ada, C or C++, and can be produced independently from the code of the application. Test scripts are fully portable between host and target environments. Test scripts may be written directly by the user, or can be generated by AdaTEST 95 and Cantata++ using a test script wizard. AdaTEST 95 also has the option of generating scripts from test case definition (TCD) data files. In addition, the tools allow the production of scripts where the test data is kept in separately maintainable form such as data tables, files, spreadsheets or databases.

AdaTEST 95 and Cantata++ static analysis provides static analysis of source and also instruments code in preparation for later dynamic analysis. With AdaTEST 95, the results of static analysis are reported in a list file and can be made available to the test script through the instrumented code. Both tools support the production of static analysis results in CSV format, which allows them to be exported to spreadsheets and databases.

AdaTEST 95 and Cantata++ dynamic analysis provides a number of analysis commands which can be incorporated into test scripts. During test execution these commands are used to gather test coverage data, to trace execution, to verify data flow, and to report on and check the results of static analysis and dynamic analysis.

When a test script is executed, AdaTEST 95 and Cantata++ produce a test report giving a detailed commentary on the execution of the tests, followed by an overall pass/fail summary at the end of the report. The anticipated outcomes of static analysis, dynamic

analysis and dynamic testing can thus be specified in a single test script and automatically checked against actual outcomes.

To assist configuration management, all AdaTEST 95 and Cantata++ outputs include date and time information. The reports output by AdaTEST 95 and Cantata++ would typically be incorporated into test logs, review logs, safety logs etc.

Instrumented code is not a compulsory part of testing with AdaTEST 95 and Cantata++. The AdaTEST 95 and Cantata++ test harnesses can be used in a 'non-analysis' mode with uninstrumented code.

### 3.3. Dynamic Testing

The *dynamic testing* facilities of AdaTEST 95 and Cantata++ enable the user to exercise the software using test scripts which are structured as a series of test cases. The tools place no restrictions on the techniques the user may apply to select test cases. Both *functional testing* and *structure-based testing* techniques can be used within test scripts. The following test case selection criteria and techniques identified by IEC 61508 are all compatible with AdaTEST 95 and Cantata++:

*Boundary value analysis*

*Error guessing*

*Equivalence Classes and Input Partition Testing*

*Structure-Based Testing*

*Cause Consequence diagrams*

*Interface Testing*

*Event Tree Analysis*

IEC 61508 correctly demands planning of *dynamic testing* as part of the design process for each software object. The products' structured test scripts can be used as *Software Design Test Specifications* and *Software Module Test Specifications*, as they are easily readable and auditable by quality assurance staff.

AdaTEST 95 and Cantata++ tests allow the developer to demonstrate that the software performs its intended function, and through negative testing facilities, that it does not perform unintended functions. The expected outcome of *dynamic testing*, *static analysis* and *dynamic analysis* can be built into test scripts. These expected outcomes can then be checked automatically by AdaTEST 95 and Cantata++ against actual outcomes when the tests are executed. This enables comprehensive *check lists* to be fully automated.

Test scripts can be placed under *software configuration management* together with the code to which they relate. However, visibility of the code to the tester is not necessary in order to carry out the tests. Hence, testing can be carried out by *independent* authorities, if necessary.

If *error seeding* is used, the test script can be executed with the seeded software and the effectiveness of the tests evaluated by comparison of the test reports for the seeded and un-seeded software.

*Information hiding and encapsulation* can potentially obscure data from the test environment. To overcome this problem. IPL recommends the use of software test

points to provide visibility of data for testing purposes. See the IPL paper “Achieving Testability when Using Ada Packaging and Data Hiding Methods” for more information. Cantata++ supports automated white-box testing by default.

The tools allow easy regression testing when conducting *re-verification* and *re-validation*, as tests can be “batched-up” and re-run automatically. The test pass/fail status is returned to the operating system to allow special action to be taken in the case of tests which fail. Similarly, when creating a *library of trusted/verified modules and components*, the tools allow easy re-verification when the modules are used on new projects, provided the software and corresponding test scripts are configuration managed in a repository.

### **3.3.1. *Timing Analysis***

AdaTEST 95 and Cantata++ provide timing analysis facilities which can be used to measure and check that *performance requirements* are met. The timing analysis facilities are particularly useful during *stress testing*.

Timing analysis uses the clock provided by the environment, so accuracy and resolution are dependent on the environment. In circumstances where greater accuracy or resolution is required, code can be included within a test script to interface to suitable timer devices. The results of such timings can then be verified from the test script.

### **3.3.2. *Simulation***

Software referenced by the software under test may either be built into a test executable, or simulated using the AdaTEST 95 and Cantata++ stub simulation facilities. If stub simulation is used to simulate device interfaces, the behaviour of the software under various outside world device conditions can be tested before the actual devices are available.

Cantata++ has another facility called Wrapping, which allows for the monitoring of all data across calls in an integrated software system. It can thus be used to check values going ‘out’ of the software being tested, and also change values being passed back to the software under test.

Another use of AdaTEST 95 and Cantata++ is for a test script to directly provide instructions to operating system facilities and device drivers. In this way entire subsystems can be simulated using AdaTEST 95 and Cantata++; an essential part of software and system integration. Simulation is often the only way to provide realistic loads during *stress testing*.

### **3.3.3. *Test Reports***

The *test reports* produced by AdaTEST 95 and Cantata++ provide a clear, auditable account of the tests performed. These results give both a detailed report on the execution of the tests and a summary of the overall test status. Each test has an unequivocal statement of test pass/fail to aid easy checking of summary results. The location of any errors detected is highlighted in the test report by the inclusion of comprehensive diagnostic information whenever a check fails. A summary of the type and number of errors detected is given at the end of the *test report*.

To assist *configuration management*, all AdaTEST 95 and Cantata++ outputs include date and time information. When *dynamic analysis* facilities are used, the filenames of original uninstrumented code and the date and time of instrumentation are also included in the test report.

The *test reports* output by AdaTEST 95 and Cantata++ would typically be incorporated into test logs, review logs, safety logs etc., forming a valuable input to activities such as *Fagan inspections, formal design reviews* and *walk throughs*.

### 3.4. Test Coverage Analysis

The *dynamic analysis and testing* parts of AdaTEST 95 and Cantata++ monitor the test coverage of *structure-based testing* through a number of coverage metrics.

*Statement Coverage* is provided for statements within normal processing (and statements within exception handlers for Ada). Statement Statistics reports can be used to show that *it is possible to execute all code statements*. Automated checking for 100% Statement Coverage will ensure that every code statement is executed at least once.

Decision Coverage includes the coverage of *branches*. Decision Statistics reports can be used to provide evidence that *both sides of every branch* have been executed. Automated checking for 100% Decision Coverage will ensure that every *branch* outcome is executed at least once.

A range of Boolean Expression Coverage metrics provide coverage analysis and reports for *compound conditions*. Automated checking for 100% Boolean Expression Coverage will ensure that *every condition in a compound conditional branch* is exercised.

For Ada software, the Exception Statistics report available through AdaTEST 95 gives a detailed analysis of the execution of exception handlers. Automated checking for 100% Exception Coverage will ensure that all exception handlers have been executed by the tests.

Call coverage can be used to check that all modules in the *Call Graph* are invoked at least once. Cantata++ call-pair coverage can be used to check that all invocation routes to a module are exercised. Related analysis provided by AdaTEST 95 and Cantata++ includes execution path verification (AdaTEST 95 only) and execution tracing, which can be used to analyse tests for *entire path* coverage.

Execution tracing facilities within the tools allow the monitoring of *control flow* and *data flow* through the code. Tracing may be selected for subprogram entry points, statements, data, decisions, or Boolean expressions. Trace output can be used to establish that every feasible *potential path* in the code is executed at least once by the testing process, and to indicate possible errors by identifying *variables that are read before being written, variables that are written more than once without being read, and variables that are written but never read*.

AdaTEST 95 can be used to demonstrate *data flow* coverage and verify that specified conditions (such as variables holding particular values) should be true at least once during test execution. Another form of assertion provided by AdaTEST 95 is the dynamic assertion. Dynamic assertions are specified in a similar form to Data Assertions, however, Dynamic assertions are used to verify assertion conditions which must always be true. Thus data assertions and dynamic assertions can be used to verify that test cases fulfil *boundary value analysis* and *equivalence class partitioning*.

Where software implements a *finite state machine*, AdaTEST 95 and Cantata++ assertions can be used to verify state-transition coverage. More details are given in the IPL paper “Testing State Machines with AdaTEST 95 and Cantata++”. Cantata++ has a specific facility for monitoring coverage within a (user-defined) state context.

The results of AdaTEST 95 and Cantata++ *dynamic analysis*, including coverage of assertion conditions, can be checked against specified levels, reported in detail, or exported and imported between test scripts. These advanced coverage and assertion facilities provided by AdaTEST 95 and Cantata++ provide a dynamic alternative to many functions which would often be associated with *static analysis*.

### 3.5. Static Analysis

AdaTEST 95 and Cantata++ *static analysis* calculates metrics mainly relating to code construct use and software complexity. With both tools the metrics are available in CSV format. This can be output to a spreadsheet or database for further analysis or storage. With AdaTEST 95 there are also options for outputting the metrics to a list file (text) format or dynamically checking metrics during a test run.

From a test script, *metrics* can be checked against user defined limits. A limit of 0 will result in a test failure if the corresponding construct is used at all. This includes counts of comments and identifiers, but obviously the meaningfulness of comments and identifier names cannot be checked automatically.

The ability to check language construct usage and code complexity provides a means of enforcing a *modular approach* and facilitates the detection of poor programming structure, excessive complexity and deviations from the required *language subset* and *coding standards*. For example, a programming standard might permit a maximum of two loops in a module of code. With AdaTEST 95 and Cantata++, a check can be built into a test script to report modules containing more than two loops.

Static Analysis provided by AdaTEST 95 and Cantata++ does not support: automated checking of the compliance of the code to the design specification; symbolic execution; or formal proof. However, the advanced dynamic analysis facilities, as described in the preceding sections of this paper, provide some compensation for this. If dynamic analysis is an unacceptable alternative, a specialised tool such as SPARK Examiner or MALPAS should be used in addition to AdaTEST 95 or Cantata++.

### 3.6. Language

AdaTEST 95 and Cantata++ are specialised tools for the testing of software written in the Ada, C and C++ languages. They are capable of analysing and testing code written in the full Ada, C and C++ languages or *language subsets* and enforcing *language subsets* and *coding standards*. Some application specific languages such as SPARK are subsets of Ada and can therefore be analysed and tested using AdaTEST 95.

Test scripts are normally written in a minimal subset of the respective language, but the full capabilities of the language are available for use in scripts when required. This enables the functionality of test scripts to be extended to facilitate techniques such as *Monte-Carlo simulation* and *probabilistic testing*. The tools ensure that AdaTEST 95 and Cantata++ commands are used in a structured and disciplined manner.

AdaTEST 95 and Cantata++ provide no facilities to directly help with *formal proof* of a program. However, it is possible to formally annotate test scripts and to include test scripts in formal proofs. It is conceivable that a development of safety critical software could enforce a formally proven language subset for use in test scripts.

Test scripts may either be written manually in the native language, or created automatically by AdaTEST 95 and Cantata++ using a test script wizard. However, where tool integrity is an issue, automatic generation of test scripts should not be used without a manual verification of the correctness of the resulting test script.

Languages other than Ada, C and C++ can be tested provided that they are accessible through language interfacing (including machine code insertions and assembler). The static and dynamic analysis capabilities of AdaTEST 95 and Cantata++ will be unavailable or severely restricted when testing software written in other languages.

### 3.7. Intrusive and Non-Intrusive Testing

Use of AdaTEST 95 and Cantata++ for test coverage requires the instrumentation of the software under test, using the instrumenter and static analyser which forms part of the toolset. Obviously, performing tests on code which is instrumented (and therefore changed) would not be compliant with the requirements for safety critical software. AdaTEST 95 and Cantata++ address this problem by operating in one of two modes.

In Analysis mode, the toolsets expect the software under test to be instrumented for coverage analysis. All analysis commands are fully executed. In Non-Analysis mode, the toolsets expect the software under test to be uninstrumented. All analysis commands are ignored, but all other (dynamic testing) commands are fully executed. Naturally, checks are made to ensure the compatibility of the software under test with the mode of the toolsets.

The two modes of AdaTEST 95 and Cantata++ should be used to execute each test script twice. Initially, Analysis mode can be used to ensure a thorough coverage of the software under test by the test script. When full coverage has been achieved, the same test script can be run again in Non-Analysis mode on uninstrumented software, to ensure that the dynamic test results are unchanged.

## 4. Tool Integrity and Development Standards

The integrity of the toolsets used in the development of safety critical software is a significant consideration. Ideally, all development should be conducted using *certified tools*. This is especially true of dynamic testing tools.

The AdaTEST 95 and Cantata++ toolsets have been developed according to the IPL Quality Management System (QMS), which is certificated to ISO 9001 and TickIT.

AdaTEST 95 and Cantata++ support the development of all standards of Ada, C and C++ software, including safety critical software. The development of AdaTEST 95 and Cantata++ followed IPL's normal ISO9001/TickIT development standards with some additional high-integrity measures for certain parts of the products.

Key points of the development and ongoing maintenance are:

- (a) The IPL Quality Management System, accredited to ISO 9001:2000 and TickIT;

- (b) The IPL Software Code of Practice, forming part of the Quality Management System;
- (c) *Hazard analysis* and the maintenance of a hazard log (AdaTEST 95);
- (d) *Independent audit*;
- (e) The use of a safe *language subset* for the core functionality. Minimal exceptions to this subset for other functionality only where absolutely necessary and justified in the hazard reporting process (AdaTEST 95). Tests can be built using just the core functionality;
- (f) *Configuration management*;
- (g) *Rigorous dynamic analysis and testing*, from the *software module testing* level upwards.

The safety critical development standards used included consideration of those used for the European Fighter Aircraft, and Defence Standards 00-55 and 00-56. Formal methods were not used in the specification of Cantata++ or AdaTEST 95. AdaTEST 95 has been audited and approved as a *certified tool* for use in airborne software to the DO178B standard

AdaTEST 95 and Cantata++ are in widespread use, providing additional confidence in the tools integrity as they have been *proven in use*. Clients and certification authorities wishing to audit the development and continuing maintenance of the products may do so by arrangement with IPL.

## 5. Conclusion

AdaTEST 95 and Cantata++ are well suited to the development of software to the IEC 61508 standard and facilitate a high degree of automation of the verification and test techniques required for effective use of the standard.

AdaTEST 95 and Cantata++ have been developed to the highest practical standard for software verification tools.

It is believed that AdaTEST 95 and Cantata++ are the only tools to offer this comprehensive functionality and the only testing tools developed to such high standards. However, this does not preclude the use of AdaTEST 95 and Cantata++ for the cost-effective testing of software which is not for safety critical use.