IPL Testing Tools and EN50128

Executive Summary

This paper describes how AdaTEST and Cantata++ can be used to assist with the development of software to the standard EN50128, Software for Railway Control and Protection Systems. In particular, it shows how AdaTEST and Cantata++ can be used to meet the verification and testing requirements of the standard. It also shows that AdaTEST and Cantata++ have been produced to a standard of sufficiently high integrity that their use for dynamic testing will not compromise the safety 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 and Cantata++ have been produced to these standards.

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

Certificate Number FM1589

Last Update:00/00/0000 00:00
File: Document1

1. Introduction

AdaTEST 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 and Cantata++ can be used to enable a software development to meet the verification and testing requirements of EN50128, Software for Railway Control and Protection Systems. It should be read in conjunction with the standard.

EN50128 standard currently has provisional status, signified by the prefix "pr". Throughout this paper, reference to EN50128 should be interpreted as a reference to the provisional standard, dated November 1995.

Section 2 of this paper provides a general description of AdaTEST and Cantata++ as tools to facilitate developing software to meet the requirements of the standard. Section 3 looks at the requirements of the standard and describes how AdaTEST and Cantata++ can be used to help meet the requirements. Section 4 looks at the integrity of the tools themselves, including standards used during their development. To assist in cross referencing to EN50128, techniques which are identified by the standard are shown in *italics* in the text of this paper.

2. A General Description of AdaTEST and Cantata++

AdaTEST 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 and Cantata++ are summarised in Table 1.

Testing		**Analysis**	
Test Preparation	**Dynamic Testing**	**Static Analysis**	**Dynamic Analysis**
• Scripts in Ada, C or C++ • Script generation from test definition • Test definition • generation from CASE tool	• Test execution • Result collection • Result verification • Timing analysis • Stub simulation • Host testing • Target testing	• Metrics: Counts Complexity • Flowgraphs • Structure • Dataflow*1 • Instrumentation	• Coverage: Statements Decisions Conditions MCDC*2 Calls Call-Pairs*1 Exceptions*2 • Trace • Assertions • Paths

Table 1 - AdaTEST and Cantata++ Facilities

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

Testing with AdaTEST 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. The AdaTEST and Cantata++ simulation facilities enable input/output devices to be simulated in the host environment.

Tests for both *functional testing* and *structure based testing* 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 automatically by AdaTEST or Cantata++ from a set of test case definitions. Test case definitions can in turn either be written directly by the user, or can be imported from a CASE tool.

AdaTEST and Cantata++ *static analysis* analyses source code and also instruments the code in preparation for later *dynamic analysis*. The results of *static analysis* are reported in a list file and can be made available to the test script through the instrumented code. *Static analysis* results can also be exported to spreadsheets and databases.

AdaTEST 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 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 and Cantata++ outputs include date and time information. The reports output by AdaTEST 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 and Cantata++. The AdaTEST and Cantata++ test harnesses can be used in a 'non-analysis' mode with un-instrumented code.

3. The Requirements of EN50128 and AdaTEST and Cantata++ Capabilities

EN50128 defines a number of process requirements for software development (sections 6 to 18). Within each process activity, the standard identifies a number of *techniques* and *measures* and their application to the development of software at each level of integrity (see annexe A and annexe B of EN50128). The implementation of such techniques and measures using AdaTEST and Cantata++ functionality is described within the following subsections.

3.1. Software Integrity Levels

AdaTEST and Cantata++ can be used to meet the verification and testing requirements for software at all software integrity levels. Section 4 of this paper shows how AdaTEST and Cantata++ have been produced to a standard of sufficiently high integrity that their

use for dynamic testing will not compromise the safety integrity of the software being tested.

The way in which a project plans to use AdaTEST or Cantata++ to test system safety functions should be documented in the system safety plan.

3.2. Personnel and Responsibilities

This section of EN50128 has two main themes in this area:

- Staff should have *appropriate training, experience and qualifications* for the roles they are assigned;
- Minimum levels of *independence* between the safety *assessor*, *designer/implementer*, *verifier* and *validator* are defined for each integrity level.

AdaTEST and Cantata++ are particularly easy tools to use. Staff competent in programming in the respective languages (Ada, C and C++) can quickly become proficient users of the tools. IPL provides written *training* material with the products and also provides more formal hands-on *training* courses.

IPL also provides a general *training* course on Unit Test Planning and Implementation, which includes topics such as overall module testing strategies, planning module testing and designing module test cases.

AdaTEST and Cantata++ test scripts and result files provide a means of communication for test designs and results between *independent* staff and organisations. Furthermore, the automated repeatability of tests built using AdaTEST and Cantata++ enables *independent* staff to repeat tests at will. More detailed descriptions of the use of AdaTEST and Cantata++ in *Software Design and Development*, *Software Verification and Testing*, *Software Validation* and *Software Assessment* are given in following sections of this paper.

3.3. Lifecycle Issues and Documentation

AdaTEST and Cantata++ impose no restrictions on selection of a *lifecycle model*. Activities such as testing individual modules and integration testing of collections of modules should be considered *elementary tasks*, where:

- The *defined input* is a test script;
- The *defined activity* is execution of the test script and correction of identified errors;
- The *defined output* is the tested and corrected item with a test results file to show that all tests have been passed and coverage objectives have been met.

IPL recommends that a formalised interpretation of the above statements should be made in the *software quality assurance plan.*

A project's decision to use AdaTEST or Cantata++, together with appropriate explanation, will also need to be documented in the *system safety plan*, *software development plan*, *software verification plan*, *software integration plan*, *software/hardware integration plan*, *software validation plan* and *software maintenance plan*. More details of the application of AdaTEST and Cantata++ to each of the associated activities is given in following sections of this paper.

Test cases defined in the *software module test specification* and *software design test specification* will be translated into AdaTEST or Cantata++ scripts during the appropriate lifecycle activity. There may also be situations where test cases in the *software requirement test specification* can best be implemented as AdaTEST or Cantata++ test scripts.

The *static analysis* capabilities of AdaTEST and Cantata++ should be used to analyse source code as part of source code verification, forming a valuable input to the *source code verification report*.

The test result files output by AdaTEST and Cantata++ test execution can be used as a basis for test reports (in particular, the *software module test report* and the *software integration report*), and as an input to the *software assessment report*. Where AdaTEST or Cantata++ are used for higher levels of testing, test result files can also contribute towards the *software/hardware integration report*.

3.4. Software Requirements Specification

AdaTEST and Cantata++ have no direct impact on the *software requirements specification.* However, decisions made when developing the *software requirements specification* can have a significant impact on testability and testing later in the software life cycle.

In particular, the way in which requirements are expressed can have a significant impact on the degree of difficulty involved in specifying and executing tests.

3.5. Software Architecture

Decisions taken when developing the *software architecture* can greatly affect the testability of the software. Consideration of how the software will be *verified*, *validated* and *assessed* at this stage of a development can have large dividends during later *lifecycle phases*. For example, it is frequently beneficial to plan to build test points into the software to facilitate later *white box* testing.

EN50128 requires that *standard software* (commercial off-the-shelf software) which is proposed for use within the overall *software architecture* must be included in the *software validation process*, with additional constraints on the way in which standard software can be used at *software integrity levels* 3 or 4.

AdaTEST and Cantata++ test scripts can be used to test *standard software* as part of the overall *software validation process.* However, coverage analysis will only be possible if the source code for the standard software is available. *Standard software* is usually procured as object code modules or executables, in which case AdaTEST or Cantata++ coverage analysis will not be available.

Previously developed software has to be *verified*, *validated* and *assessed* to the appropriate *software integrity level*. This may already have been done on a previous project, ideally to the EN50128 standard. Nevertheless, should *changes* be necessary to *previously developed software*, tests will have to be modified and repeated. If *previously developed software* is to be used at a higher *software integrity level* than that to which it was originally developed, tests will have to be enhanced and repeated. Even if *previously developed software* will be used without change at an appropriate *software integrity level*, tests should be repeated in the new environment.

In all cases, the maintainability, portability and automated repeatability of test scripts developed using AdaTEST or Cantata++ will serve to facilitate such re-use, providing an excellent mechanism for the continued *verification, validation* and *assessment* of *standard software* and *previously developed software* in the new development.

3.6. Software Design and Development

Design consideration of how the software will be *verified*, *validated*, *assessed* and maintained should continue through this *lifecycle phase*. *Module test specifications* should be developed concurrently with the *software design specifications*.

It is notable that EN50128 requires the use of automated testing tools (such as AdaTEST or Cantata++).

During coding, the *static analysis* facilities of AdaTEST or Cantata++ can be used to monitor the size and complexity of the software produced. The *static analysis* facilities may also be used to monitor the use of language subsets and coding standards.

Dynamic testing can use AdaTEST or Cantata++ scripts to automate execution of the *software module test specifications*, with the corresponding test result files being used as an input to the *software module test reports.* AdaTEST and Cantata++ dynamic analysis results are also output to the test result files, facilitating *verification* that each module has *correctly satisfied its test specification.*

AdaTEST and Cantata++ scripts can also be used to automate integration testing of *previously tested modules of software*.

With specific reference to Annexe A Clause 11 of EN50128, AdaTEST and Cantata++ can assist with the following techniques and measures:

- *Modular Approach.* AdaTEST and Cantata++ encourage a *modular approach* to software development, facilitating the testing of modules in isolation, or in top-down or bottom-up strategies.

Metrics provided by AdaTEST and Cantata++ can be checked against specified limits to enforce the *modular approach* measures identified by Annexe A Table D.10: *module size limited*, *information hiding/encapsulation*, *parameter number limit* and *one entry/one exit point in subroutines and functions*. A *fully defined interface* is enforced by the Ada language (in C and C++, full definition of an interface is a manual discipline).

- *Design and Coding Standards*. The *static analysis* and *metrics* facilities of the tools can be used to enforce some aspects of coding standards.
- *Analysable Programs*. The *static analysis* facilities of the tools can be used to analyze source code.
- *Programming Language*. AdaTEST and Cantata++ support the analysis and testing of software written in Ada, C and C++, or *language subsets* of these languages. *Programming 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 and Cantata++ will be unavailable or severely restricted when testing software written in other *programming languages*.

- *Language Subset*. The *static analysis* and *metrics* facilities of the tools can be used to help enforce language subsets.
- *Library of Trusted/Verified Modules and Components*. Establishing and using such a library can be facilitated by maintaining an AdaTEST or Cantata++ test script alongside each component of the library.
- *Functional and Black Box Testing*; The tools support the specification and execution of both *functional*/*black box* test cases and *structure-based*/white box test cases.
- *Performance Testing*; The *dynamic analysis* statistics functions and timing analysis functions provided by the tools can be used to assist with the implementation of *performance tests*.
- *Interface Testing*; Test scripts built using AdaTEST and Cantata++ exercise a module of software by calling its interfaces. The stub simulation facility can be used to simulate called interfaces. Thus both interfaces in to a module and a modules use of other interfaces can be tested.
- *Data Recording and Analysis*. The test results files output by AdaTEST and Cantata++ when a test script is executed provides a complete record of the execution of all test cases. *Dynamic analysis* commands can be placed in a test script to provide multiple levels of execution trace, multiple levels of execution statistics, and test coverage analysis.
- *Object Oriented Programming*. Cantata++ supports analysis and testing of software written in C and C++. (However, IPL would not recommend C++ as a language suitable for the implementation of safety related software). AdaTEST is fully compatible with Ada 83, supporting the analysis and testing of object oriented features of Ada83. AdaTEST is currently being enhanced to support the more sophisticated object oriented features of Ada95.

3.7. Software Verification and Testing

The *software verification plan* has to document all the *criteria*, *techniques* and *tools* to be used for *verification* during each *lifecycle phase*, including *the selection and utilisation of software test equipment* and *the test environment, tools, configuration and programs*, such as AdaTEST or Cantata++.

The *software verification plan* must also specify the *degree of test coverage required to be achieved*. AdaTEST and Cantata++ provide automated measurement of a comprehensive range of test coverage metrics, including statement coverage, decision coverage, and a selection of Boolean condition coverage measures. Furthermore, users may attach assertions to source code to facilitate further coverage measures.

When using AdaTEST or Cantata++, all testing is *fully documented* in the form of test scripts and test result files. The test results file should be retained as evidence that *the software has passed the verification or the reasons for the failures*.

All test execution is performed by fully *automatic means* and all tests are fully *repeatable.* Thus all testing can be *regarded as part of the verification.* The automated repeatability of tests also serves to enhance *maintainability* of the software and the tests.

AdaTEST and Cantata++ static analysis provides an input to *source code verification* through a wide range of static analysis metrics, including size metrics, complexity metrics and language use metrics.

The use of AdaTEST and Cantata++ is not limited to module testing. The use of AdaTEST and Cantata++ described in the above paragraphs is also applicable to *software integration testing*.

With specific reference to Annexe A Clause 12 of EN50128, AdaTEST and Cantata++ can assist with the following *techniques* and *measures*:

- *Probabilistic Testing*. Loops can be used in AdaTEST and Cantata++ scripts to repeat individual tests or sequences of tests with variations in test data. Such variations could be randomly generated or created as a function of the number of loops. Provided that a suitable test oracle is available, the test oracle can be interrogated from the test script to give anticipated outcomes which can be checked against actual outcomes.

AdaTEST and Cantata++ scripts can also be interfaced to data files generated by a simulator or an existing system, with large volumes of test inputs and corresponding anticipated outcomes being read from a file and applied to the software under test by a loop in the test script.

When conducting such tests, it is advisable to disable full AdaTEST or Cantata++ output during the loop, so as to avoid creating an enormous results file. AdaTEST or Cantata++ import/export commands can be used to place such loops of tests within a sequence of scripts, so that a summary of test outcomes from the loop is reported at the end of the script containing the loop.

- *Static Analysis*. AdaTEST and Cantata++ *static analysis* calculates *metrics*, analyses program structure and data, and instruments source in preparation for dynamic analysis.

- *Dynamic Analysis and Testing*. The primary purpose of AdaTEST and Cantata++ is to support *dynamic testing* of software and to provide *dynamic analysis* while software is being tested.

- *Metrics* (including counts of the use of language constructs and complexity analysis metrics) are reported in *static analysis* listings and can be made available to the test script through the instrumented code. *Static analysis* results including *metrics* can also be exported to spreadsheets and databases.

Traceability Matrix. One means of maintaining a *traceability matrix* is to embed formalised traceability statements as comments in documentation and code. The comments can then be parsed to build a *traceability matrix*. If this approach is adopted, traceability statements can be included in test scripts as comments, or even echoed to test results files using the comment command. Test scripts and test results files can then be parsed to provide an input to the *traceability matrix*.

Within Annexe A, table D.2 refers in more detail to specific *dynamic analysis and testing techniques* and *measures*. AdaTEST and Cantata++ are capable of supporting all of the referenced *techniques* and *measures*. In particular:

- *Boundary Value Analysis* is a test case design technique. Once designed, test cases can be implemented within AdaTEST or Cantata++ scripts. AdaTEST and Cantata++ assertions can be embedded in source code as formal comments to measure that specified *boundary values* have been tested.

- *Error Guessing* is a test case design technique. Once designed, test cases can be implemented within AdaTEST or Cantata++ scripts.

- *Error Seeding*. All AdaTEST and Cantata++ test scripts are automatically repeatable. Consequently, once a script has been developed, test execution can be repeated against software which has been subjected to *error seeding* and the test results compared to give a measure of test effectiveness.
- *Performance Modelling*. AdaTEST and Cantata++ provide facilities for analyzing and measuring performance of the software under test. Real measures taken from AdaTEST or Cantata++ test result files can be fed back to performance models
- *Equivalence Classes and Input Partition Testing* is a test case design technique. Once designed, test cases can be implemented within AdaTEST or Cantata++ scripts. AdaTEST and Cantata++ assertions can be embedded in source code as formal comments to measure that specified *equivalence classes and input partitions* have been tested.
- *Structure Based Testing* is a test case design technique often associated with structural coverage measures. Once designed, test cases can be implemented within AdaTEST or Cantata++ scripts. AdaTEST and Cantata++ coverage analysis can be used to measure the structural coverage of tests using a selection of metrics including statement coverage, decision coverage and a range of condition coverage measures.

3.8. Software/Hardware Integration

The *software/hardware integration plan* has to document the *test environment including tools, support software and configuration description*. The planned use of AdaTEST or Cantata++ consequently has to be stated in the *software/hardware integration plan*.

A key feature of AdaTEST and Cantata++ is the portability of test scripts between the host and the target environment. A test can be developed and executed in a host environment, then moved to the target environment when the time comes for *merging of the software system on to the target hardware*.

During system integration, there is frequently a need to simulate interfaces, providing test data injection, and recording and analysis of outputs. AdaTEST and Cantata++ can assist with system integration by:

- Simulation of an interface with software internal to the system being integrated;
- Control of external hardware to provide a simulation of the actual interfacing hardware.

As in the host environment, the test result files from *hardware/software integration* tests conducted using AdaTEST and Cantata++ should be saved to provide *auditable* proof of testing. Test result files also provide *machine readable* recording suitable for *subsequent analysis*.

An IPL paper, “Host - Target Testing” provides further detail on the use of AdaTEST and Cantata++ for testing host-target software developments.

Annexe A Clause 13 of EN50128 references *techniques* and *measures* applicable to *software/hardware integration*. Both *functional/black box testing* and *performance testing* have been addressed previously in response to Annexe A Clause 11.

3.9. Software Validation

AdaTEST and Cantata++ can be used to provide *simulation* of interfacing systems during *software validation*, using similar techniques to those listed previously under *software/hardware integration*:

- *Simulation* of an interface with software internal to the system being integrated;
- Control of external hardware to provide a *simulation* of the *required input signals* and analysis of output signals against anticipated *output signals with their sequences and values.*

Refer to the IPL paper, "Host - Target Testing" for further detail on the use of AdaTEST and Cantata++ to construct simulations.

Test result files from *software validation* tests conducted using AdaTEST and Cantata++ should be saved as an input to the *software validation report* and to provide *auditable* proof of validation.

With specific reference to Annexe A Clause 14 of EN50128, AdaTEST and Cantata++ can assist with the following *techniques* and *measures*:

- *Probabilistic Testing* is addressed above under the response to Annexe A Clause 12.
- *Performance Testing* is addressed above under the response to Annexe A Clause 11.
- *Functional/Black Box Testing* is addressed above under the response to Annexe A Clause 11.
- *Modelling*. AdaTEST and Cantata++ scripts can be used to exercise *models* in the same way as which they are used to exercise the software under test. Thus a test script can be used to test or evaluate a model. A test script can also reference a proven model to act as an *oracle* for *probabilistic testing*.

3.10. Software Assessment

AdaTEST and Cantata++ results files provide the *software assessor* with documented and auditable evidence of static analysis, dynamic analysis and testing activities. Furthermore, should the *software assessor* not be present when a particular test is executed, the test can easily be repeated in the *presence of the assessor*. Should any *additional verification or validation* work be required, test cases can be added to AdaTEST or Cantata++ scripts and the tests repeated.

Two particular *techniques* and *measures* referenced by Annexe A Clause 15 of EN50128 are the primary function of AdaTEST and Cantata++: *Static software analysis* and *dynamic software analysis*. Another primary function of AdaTEST and Cantata++ is dynamic testing. Dynamic testing forms part of the *techniques/measures* identified as *verification* and *validation*. *Checklists* can be built into test scripts and may reference the outcome of *static analysis*, *metrics*, *dynamic analysis* and dynamic testing.

3.11. Software Quality Assurance

AdaTEST and Cantata++ were both developed to ISO9001 (*EN29001*) TickIT accredited quality management system. When used as part of an *EN29001* accredited quality management system, AdaTEST and Cantata++ promote and facilitate good testing practices and provide auditable records of all test outcomes. An IPL paper "Cantata++ and AdaTEST in the ISO9001/BS5750/TickIT Software Quality Process"

provides further information on use of the tools in an ISO9001 (*EN29001*) accredited quality management system.

As described previously under *software architecture*, AdaTEST and Cantata++ can be used as part of a strategy to assure the quality and integrity of both *standard software* and *previously developed software*.

3.12. Software Maintenance

When changes are made, corresponding changes can be made to AdaTEST and Cantata++ test scripts and the modified item re-tested.

The automated repeatability of tests developed using AdaTEST and Cantata++ enables full *regression testing* to be conducted at minimum cost. For example, within the AdaTEST and Cantata++ development teams at IPL, batch files were used to fully repeat all module tests at regular intervals.

3.13. Systems Configured by Application Data

Generic systems configured by application data require additional testing which falls into two general categories:

- Testing to ensure that the system is truly generic and will work with any valid set of configuration data;
- Testing the system once it has been configured, for each configuration.

A testing strategy based on AdaTEST or Cantata++ can provide assistance in both of these areas. Once a test script has been created and basic test cases developed, it is a simple task to cut, paste and edit test cases to extend the test script with variations of existing tests.

It is also possible to create test scripts which are configured by application data. These Configurable test scripts can then be used to test specific configurations of a generic software item or system.

4. Tool Integrity and Development Standards

AdaTEST and Cantata++ have been developed according to the IPL Quality Management System (QMS), which has been accredited to ISO 9001 and TickIT.

The integrity of the tools used in the development of safety critical software is a significant consideration. Tool integrity is particularly important where high level languages such as Ada, C and C++ are used, as even safe subsets of high level languages are often unsuitable for reverse translation.

AdaTEST and Cantata++ support the development of all standards of Ada, C and C++ software, including safety critical software. The development of AdaTEST and Cantata++ followed IPL's normal ISO9001/TickIT development standards with some additional high integrity measure 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 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).

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). Tests can be built using just the core functionality.

f) Configuration Management.

g) Rigorous dynamic testing, from module level upwards.

The safety critical development standards used in the development of AdaTEST included consideration of those used for the EuroJet project, and Defence Standards 00-55 and 00-56. Formal methods were not used in the specification of AdaTEST or Cantata++.

AdaTEST and Cantata++ are in widespread use, providing additional confidence in the tool's integrity. 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 and Cantata++ are well suited to the development of software to the EN50128 standard, facilitating a high degree of automation of the verification and test techniques required for effective use of the standard.

AdaTEST and Cantata++ have been developed to the highest practical standard for software verification tools. They have both proven integrity and reliability through extensive use.

It is believed that AdaTEST 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 and Cantata++ for testing software which is not for safety critical use.