

IPL Testing Tools and IEC880

Executive Summary

This paper describes how AdaTEST and Cantata++ can be used to assist with the development of software to the standard IEC880, Software for Computers in the Safety Systems of Nuclear Power Stations. 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.

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*



Certificate Number FM1589

*Last Update: 05/06/02 16:15
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 IEC880, Software for Computers in the Safety Systems of Nuclear Power Stations, first edition, 1986. It should be read in conjunction with the standard.

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 IEC880, 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
<ul style="list-style-type: none"> • Scripts in Ada, C or C++ • Script generation from test definition • Test definition • generation from CASE tool 	<ul style="list-style-type: none"> • Test execution • Result collection • Result verification • Timing analysis • Stub simulation • Host testing • Target testing 	<ul style="list-style-type: none"> • Metrics: <ul style="list-style-type: none"> Counts Complexity • Flowgraphs • Structure • Dataflow^{*1} • Instrumentation 	<ul style="list-style-type: none"> • Coverage: <ul style="list-style-type: none"> 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 structural 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 IEC880 and AdaTEST and Cantata++ Capabilities

Appendix E of IEC880 - Software Testing, provides guidance on software testing, describing a number of techniques and their application to software for computers in the safety systems of nuclear power stations (table E4.1 and E4.2). The implementation of such techniques using AdaTEST and Cantata++ functionality is described within the following subsections.

Unless stated otherwise, the description given is for the application of the technique during unit or integration testing.

3.1. Overview (E4.1)

When an AdaTEST or Cantata++ test script is executed, a detailed test report is output to a test results file. This report provides a case by case analysis of test case execution followed by a summary of results. The test report is annotated with version and date/time information to assist with configuration management. These reports are a

useful input to the overall *supervision of the testing procedure*, over time providing the necessary data for derivation of reliability figures using probabilistic techniques.

Such probabilistic techniques are just one aspect of *statistical testing* which is described in more detail in table E4.3 of IEC880 and in section 3.3 of this paper.

AdaTEST and Cantata++ provide static analysis in the form of metrics and program structure information. Such *program analysis* is useful in assessing the structure and complexity of software and provides a means of enforcing language subsets. However, for rigorous *program proving* specialised tools such as SPARK Examiner or MALPAS are recommended.

3.2. Systematic Tests (E4.2)

3.2.1. General (E4.2a)

The general techniques identified in table E4.2a describe general good test design practice which the test designer should consider in designing software tests. AdaTEST and Cantata++ scripts can be used to implement test cases derived using any of the six techniques listed in the table.

Both AdaTEST and Cantata++ support the placement of assertions in source code as formalised comments, which are then translated into test instrumentation by the static analysis component of the tools. By using assertions to identify traceability to requirements or design documentation, a test designer can ensure that all *individually and explicitly specified requirements and cases representative for program behaviour* have been dynamically tested.

Assertions may also be used to ensure that tests have been executed for *all input variables in extreme positions* by placing an assertion for each input at its extreme positions.

3.2.2. Path Testing (E4.2b)

AdaTEST and Cantata++ support a number of test coverage metrics which may be used to determine the completeness of path testing.

- (a) “STATEMENT_COVERAGE” measures that *every statement* has been *executed at least once*;
- (b) “DECISION_COVERAGE” measures that *every outcome of every branch* has been *executed at least once*, including the loop/exit conditions for all loops;
- (c) “BOOLEAN_OPERATOR_COVERAGE” and “BOOLEAN_OPERAND_EFFECTIVENESS_COVERAGE” are both measures that *each predicate term* has been *exercised to each branch*;

Assertions may be placed in the source file to ensure that *each loop* has been *executed with the minimum, maximum and intermediate number of repetitions*. AdaTEST and Cantata++ “DECISION_STATISTICS” may also be used to determine that loops have been *executed with the minimum, maximum and intermediate number of repetitions*.

The AdaTEST and Cantata++ PATH and VERIFY_PATH may be used in association with label comments in the source code to specify test paths and verify that a specified

test path is followed. If all test paths are specified by the test designer, then these commands will show that *every path* has been *executed at least once*.

3.2.3. **Data Movement Testing (E4.2c)**

The AdaTEST and Cantata++ PATH and VERIFY_PATH commands may also be used to verify that *every assignment to each memory place is executed at least once*, *every reference to each memory place is executed at least once*, and that *all mappings from each input to output are executed at least once*.

Assertions may also be placed in the source file to provide similar information, or to provide additional confirmation that assignments and references have been made with specified values.

3.2.4. **Timing Testing (E4.2d)**

AdaTEST and Cantata++ both provide “TIMING_ANALYSIS” commands to measure the execution time of individual tests or groups of tests. These commands may be used to *checking of all timing constraints*. However, the accuracy and resolution of timing analysis is limited to that available from the execution environment clock, so a test designer may prefer to use external timing analysis equipment.

Test designers should be careful about using AdaTEST or Cantata++ analysis instrumentation within interrupt service routines.

Placing dynamic analysis instrumentation in live interrupt service routines will slow down such critical sections of code considerably. Furthermore, processing real interrupts could lead to anomalous behaviour within the test harness.

A more fruitful approach is to build the interrupt service routines into a test harness and call them directly from the test script (with the associated hardware interrupt disabled). This will enable *all significant* or the *maximum possible combinations of interrupt sequences to be tested*.

3.2.5. **Miscellaneous (E4.2e)**

AdaTEST and Cantata++ assertions may also be used to *check for the correct position of all boundaries of the input data space* and *check for sufficient accuracy of arithmetical calculations*.

The use of AdaTEST and Cantata++ is not restricted to unit (module) testing. Tests can also be executed on integrations of software or even entire programs, to verify *module interfaces and module interaction*. The CHECK_CALLS command can be used to verify that *every module* has been *invoked at least once*. In Cantata++, CALL_SITE coverage verifies that *every invocation to a module is exercised at least once*. AdaTEST does not implement this coverage measure, but similar functionality can be provided by the use of assertions at each invocation point to a module.

Both AdaTEST and Cantata++ also provide multiple levels of trace output. Analysis of trace output can also be used to verify that *every module* has been *invoked at least once* and that *every invocation to a module is exercised at least once*.

3.3. Statistical Tests (E4.3)

AdaTEST and Cantata++ provide no explicit support for statistical tests. It is up to the test designer to implement statistical tests within an AdaTEST or Cantata++ script. AdaTEST and Cantata++ test scripts use the appropriate language (Ada, C or C++). The full capabilities of these languages are consequently available to test designers to implement statistical testing techniques.

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. Where a suitable test oracle is available, the test oracle may 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.

Both AdaTEST and Cantata++ provide facilities for reporting execution statistics on the software under test. Reports such as “STATEMENT_STATISTICS” can be output to separate text files in a format suitable for subsequent input to spreadsheets (for example, as comma separated values). A spreadsheet can then be used for more detailed statistical analysis.

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++.

Both 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

Ada (and appropriate subsets) has become the language of choice for the implementation of safety related systems. Whilst readers of this paper will primarily be interested in the application of AdaTEST, it can be seen that Cantata++ offers similar functionality for C and C++ development should such languages be used.

AdaTEST and Cantata++ are well suited to the development of software to the IEC880 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.

