

IPL Testing Tools and RTCA/DO-178B

Executive Summary

This paper describes how AdaTEST 95 and Cantata++ can be used to assist with the development of software to the standard RTCA/DO-178B, 'Software Considerations in Airborne Systems and Equipment Certification'. 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 an airborne software development.

IPL is an independent software house founded in 1979 and based in Bath. The company has carried out work in the area of safety-related avionics systems since 1987 and its Software Code of Practice has been approved by several major aerospace suppliers. IPL achieved accreditation to the ISO9001 quality standard in 1988 and TickIT in 1992. Both AdaTEST 95 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 FM 01589

Last Update: 08/07/2003 15:02
File: DO178B.DOC

1. 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 airborne software, although both tools are sufficiently flexible for the testing of any Ada, C or C++ software.

RTCA/DO-178B ‘Software Considerations in Airborne Systems and Equipment Certification’ (referred to as DO-178B throughout this paper) is an international standard relating to the safety and airworthiness of software for avionics systems. It describes techniques and methods appropriate to ensure the integrity and reliability of such software. The standard was first published as RTCA/DO-178 in 1980, and reissued as DO-178A in 1985. The current issue is DO-178B.

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 DO-178B. It should be read in conjunction with the standard. To assist in cross-referencing, terms which are used by DO-178B are shown in *italics* in this text.

The description consists of two parts. Section 2 looks at AdaTEST 95 and Cantata++ as tools to facilitate developing software to meet the requirements of the standard. Section 3 looks at the integrity of the tools themselves, including standards used during their development.

2. Cantata++, AdaTEST 95 and DO-178B

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

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.

2.2. Dynamic Testing

Ref:

- 2.6 - *System Requirements Considerations for Software Verification*
- 4.4.2 - *Language and Compiler Considerations*
- 6.0 - *Software Verification Process*
- 6.1 - *Software Verification Process Objectives*
- 6.4 - *Software Testing Process*
- 6.4.2 - *Requirements Based Test Case Selection*
- 6.4.2.1 - *Normal Range Test Cases*
- 6.4.2.2 - *Robustness Test Cases*
- 7 - *Software Configuration Management Process*
- 11.3 - *Software Verification Plan*
- 11.13 - *Software Verification Cases and Procedures*
- 11.14 - *Software Verification Results*

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* [Ref 11.13]. The tools place no restrictions on the techniques the user may apply to select *test cases*, including *Normal Range Test Cases* [Ref 6.4.2.1] and *Robustness Test Cases* [Ref 6.4.2.2]. Both *requirements-based* testing [Ref 6.4.2] and structural testing techniques can be used within test scripts. All test case selection criteria and techniques identified by DO-178B are compatible with AdaTEST 95 and Cantata++.

Dynamic testing should be planned and *pass/fail* criteria specified [Ref 11.13, 11.14] as part of the design process for each software object. The products' structured test scripts can be used as software *test cases* and *test procedures* [Ref 11.13] as they are easily readable and auditable by quality assurance staff. The test scripts contain documentation of the *inputs*, *conditions* and *expected results* [Ref 11.13] required by DO-178B.

AdaTEST 95 and Cantata++ tests allow the developer to demonstrate to a high degree of confidence that *the executable object code satisfies software requirements* [Ref 6.1], and through negative testing facilities, that it does not *exhibit specific anomalous behaviour* [Ref 2.6]. 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.

The products allow the developer to test software at all levels required by DO-178B: *Low Level Testing, Software Integration Testing and Hardware/Software Integration Testing*. [Ref 6.4]

Test scripts can be placed under *software configuration management* [Ref 7] 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, it is possible to have *independence of software verification process activities* [Ref 6.0, 11.3].

The tools place no special constraints [Ref 4.4.2] on the code being tested with respect to special compiler or linker options. They are thus, less intrusive than many other facilities such as debuggers, simulators and emulators. However, AdaTEST 95 and Cantata++ are fully compatible with such tools.

2.2.1. *Host and Target Testing*

Ref:

4.4.3 - *Software Test Environment*

6.4.1 - *Test Environment*

6.4.3a - *Requirements-Based Hardware/Software Integration Testing*

AdaTEST 95 and Cantata++ are specifically designed to allow a seamless transition between testing in the *host* and *target* [Ref 4.4.3, 6.4.1] environments. Tests can be developed and debugged in the convenience of the *host* environment before submitting them to the rigours of the *target* hardware, where they can be run without change. This allows *hardware/software integration* [Ref 6.4.3a] tests to be performed with the same test tools as for all other types of test. The tools' coverage and static analysis facilities may also be used in the target environment, if required.

2.2.2. *Timing Analysis*

Ref:

6.4.2.1 - *Normal Range Test Cases*

6.4.3 - *Requirements-Based Testing Methods*

AdaTEST 95 and Cantata++ provide timing analysis facilities which can be used to measure and check that *execution time requirements* [Ref 6.4.2.1, 6.4.3] are met.

Timing analysis uses the clock provided by the run-time system, 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.

2.2.3. *Simulation*

Ref:

6.4.3a - *Requirements-Based Hardware/Software Integration Testing*

6.4.3b - *Requirements-Based Software Integration Testing*

6.4.3c - *Requirements-Based Low Level Testing*

Software referenced by the software under test may either be built into a test executable, or simulated using the tools' stub simulation facilities. This allows either an integration or isolation approach to *low level testing* [Ref 6.4.3c] to be taken.

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 Integration Testing* [Ref 6.4.3b] and *Hardware/Software Integration Testing* [Ref 6.4.3a].

Cantata++ has an additional feature called 'wrapping' which allows interfaces to external software objects to be monitored and modified while including that external software in the test. The two principal uses of this are to check values output to the external software and to change the values returned.

2.2.4. *Test Reports*

Ref:

- 6.3 - *Software Reviews and Analyses*
- 6.3.4 - *Reviews and Analyses of the Source Code*
- 6.3.6 - *Reviews and Analyses of the Test Cases, Procedures and Results*
- 7 - *Software Configuration Management Process*
- 11.14 - *Software Verification Results*
- A-7 - *Verification of Verification Process Results*

The *verification results* [Ref 11.14] 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* [Ref 6.3.6, A-7.2]. 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 *software configuration management* [Ref 7], all AdaTEST 95 and Cantata++ outputs include details of the *configuration item* and date/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 reports output by AdaTEST 95 and Cantata++ would typically be incorporated into test logs, review logs, safety logs etc., forming a valuable input to *Software Reviews and Analyses* [Ref 6.3, 6.3.4, 6.3.6].

2.2.5. *Regression Testing and Testing of Multiple-Version Dissimilar Software*

Ref:

- 2.3.2 - *Multiple-Version Dissimilar Software*
- 12.1.1 - *Modifications to Previously Developed Software*
- 12.1.3 - *Change of Application or Development Environment*
- 12.3.3 - *Considerations for Multiple-Version Dissimilar Software Verification*

The tools allow easy regression testing when previously developed or modified software has to be *reverified* [Ref 12.1.1], 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.

When creating a library of verified modules, the tools allow easy reverification when the modules are used on new projects (or when a change of development or target environment is necessary [Ref 12.1.2]), provided the software and corresponding test scripts are configuration managed in a repository. Test scripts are fully portable between different compilers and operating systems.

Where *Multiple-Version Dissimilar* software systems are developed [Ref 2.3.2, 12.3.3] AdaTEST 95 and Cantata++ can be used to re-run *test cases* on each version of the software. This allows either a ‘shared’ verification approach to be taken, or alternatively, *test cases* can be exchanged between the software streams. The portability of the tools and their test scripts further facilitates this technique.

2.3. Test Coverage Analysis

Ref:

4.4.2 - *Language and Compiler Considerations*

6.4.4.2 - *Structural Coverage Analysis*

A-7 - *Verification of Verification Process Results*

The dynamic analysis parts of AdaTEST 95 and Cantata++ monitor *structural coverage* [Ref 6.4.4.2] during testing through a number of coverage metrics. Coverage requirements are most conveniently incorporated into test scripts in the form of (user-defined) rulesets, using the coverage analysis wizards.

Call coverage can be used to check that all modules in the software under test are invoked at least once, thus ensuring that *control coupling between the code components* is exercised [Ref 6.4.4.2, A-7.8]. Cantata++ has call-pair coverage which can be used to check that all invocation routes to a module are exercised. Related analysis provided by AdaTEST 95 includes execution path verification. Both tools offer execution tracing to assist with debugging.

Modified Condition/Decision Coverage (MC/DC) [Ref A-7.5] is provided for all decisions and Boolean expressions in the source code. Automated checking for 100% *Modified Condition/Decision Coverage* will ensure that each Boolean operand can independently affect the outcome of the overall expression or condition.

In the next issue of DO178, the current term *Modified Condition/Decision Coverage* may be redefined as “Unique Cause Modified Condition/Decision Coverage”. A new coverage measure which allows the rules of MC/DC to be relaxed slightly will be introduced under the term “Masking Modified Condition/Decision Coverage”. Both of these measures are fully supported in both AdaTEST 95 and Cantata++.

Decision Coverage [Ref A-7.6] is provided for all decisions in the source code and provides a measure of the coverage of all conditional branches. Automated checking for 100% *Decision Coverage* will ensure that every branch outcome is executed at least once.

Statement Coverage [Ref A-7.7] 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.

Exception coverage (AdaTEST 95) can be used to measure test coverage of *exception handling* and *built in error detection* [Ref 4.4.2b].

Execution tracing facilities within the tools allow the monitoring of control flow through the code. Tracing may be selected for subprogram entry points, statements, decisions, or boolean expressions. Trace output can be used to establish that every feasible path in the code is executed at least once by the testing process.

AdaTEST 95 Data Assertions can be used to verify that specified conditions (such as variables holding particular values) should be true at least once during test execution. AdaTEST 95

The results of AdaTEST 95 and Cantata++ test coverage analyses, including coverage of assertion conditions, can be checked against specified levels, reported in detail,

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.

Cantata++ also offers a supplement to structural coverage, namely context coverage. This allows users to check whether the required structural coverage levels have been reached in a number of different contexts – namely inherited classes, states (user-defined) and threads (user-defined).

2.4. Static Analysis

Ref:

6.3.4 - *Reviews and Analyses of the Source Code*

6.4.4.3 - *Structural Coverage Analysis Resolution*

11.8 - *Software Code Standards*

The *Static Analyzer* facilities provided by AdaTEST 95 and Cantata++ calculates metrics, analyses program structure and data, and instruments source in preparation for dynamic analysis.

Metrics collection provides counts of the use of language constructs and *complexity* metrics. AdaTEST 95 metrics are reported in static analysis listings, and can be made available to the test script through the instrumented code. This allows the use of most language constructs to be monitored, and complexity analysis metrics to be reported and checked against user-defined limits.

The ability to check language construct usage and code complexity facilitates the detection of poor programming structure, excessive complexity and deviations from the required *Software Code Standards* [Ref 6.3.4, 11.8]. 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 made to report modules containing more than two loops. Similarly, the tools can be used to ensure that a given language subset is adhered to. Static analysis results can also be exported to spreadsheets and databases.

The tools provide many of the well-known *complexity* metrics, including those due to McCabe, Myers, Halstead, Hansen and Harrison, and also the wealth of raw data upon which these metrics are based. AdaTEST 95 and Cantata++ allow the user to define, calculate and verify their own *complexity* metrics. In addition, the tools allow the *complexity of logical expressions* to be measured and verified [Ref 6.3.4].

Static analysis performed by AdaTEST 95 and Cantata++ can also detect *dead (unreachable) code* [Ref 6.4.4.3]. Metrics are also provided for the *nesting level* of the control structure [Ref 6.3.4].

3. Tool Integrity and Development Standards

Ref:

4.4.1 - *Software Development Environment*

12.2 - *Tool Qualification*

12.2.1 - *Qualification Criteria for Software Development Tools*

12.3.5 - *Product Service History*

The integrity of the toolsets used in the development of safety critical airborne software is a significant consideration. This is especially true of dynamic testing tools [Ref 4.4.1].

AdaTEST 95 and Cantata++ have been developed to the highest practical standards and comply with the requirements of DO-178B on *Tool Qualification* [Ref 12.2] for *Software Development Tools* [Ref 12.2.1].

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

AdaTEST 95 and Cantata++ are required to 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 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 (AdaTEST 95).
- 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 testing, from module level upwards.

AdaTEST 95 and Cantata++ are in widespread use, providing additional confidence in the tools integrity through the *Product Service History* [Ref 12.3.5]. Clients and *Certification Authorities* wishing to audit the development and continuing maintenance of the products may do so by arrangement with IPL.

For an up to date statement on the qualification status of our tools see <http://www.iplbath.com/pdf/p0838.pdf>.

4. Conclusion

AdaTEST 95 and Cantata++ are well suited to the development of software to the DO-178B 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.