

## **IPL Testing Tools And the Galileo Software Standard (GSWS)**

### **Executive Summary**

This paper describes how AdaTEST 95 and Cantata++ can be used to assist with the development of software to the Galileo Software Standard (GSWS). 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 Galileo project.

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 aerospace 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](mailto:ipl@iplbath.com)*



Certificate Number FM 01589

*Last Update: 27/07/2004 08:45:00  
File: IPL Testing Tools and Galileo.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 software, although both tools are sufficiently flexible for the testing of any Ada, C or C++ software.

The Galileo Software Standard is referred to as GSWS throughout this paper. To assist in cross-referencing, quotations and terms which are used by GSWS are shown in *italics* in this text. References from the standard will be depicted by square brackets containing the appropriate reference number, and sometimes, the sub paragraph number. For example, [Ref n.n] refers to section n.n of GSWS, and [Ref A.3, §4] refers to Appendix A.3, paragraph 4. References of the form GSWS-XX-NNNN are also provided in this paper; these are the precise requirement numbers in GSWS. For convenience, a list of all references used in a given section are provided at the beginning of that section.

*GSWS sets requirements to be followed for the management, evaluation, procurement, development, production, verification, operation and maintenance of all software products within the Galileo project* [Ref 1.1]. It describes techniques and methods appropriate to ensure the integrity and reliability of such software. The current issue is 7 draft 6, dated 07<sup>th</sup> May 2004.

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 GSWS. It should be read in conjunction with the standard.

The description consists of two parts. Sections 1 to 4 look at AdaTEST 95 and Cantata++ as tools to facilitate developing software to meet the requirements of the standard. Section 5 looks at the integrity of the tools themselves, including standards used during their development.

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

*Ref:*

A.29            *Algorithm Verification Reference Scenarios (ALG-VER)*  
6.5.2.2        *Tests Methods Requirements (GSWS-SWENG-1895, GSWS-SWENG-1910)*

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 [Ref A.29, 5.n.7].

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. Thus the requirement where *tests run on instrumented code during integration and TS Validation... shall be re-run on the non-instrumented code* [Ref GSWS-SWENG-1895] is easy to achieve.

AdaTEST 95 and Cantata++ have been ported to a wide variety of platform and compiler combinations. If the software to be developed is to be re-used on different platforms, AdaTEST 95 and Cantata++ can in most cases run the tests on the additional platform without changes to the test scripts [Ref GSWS-SWENG-1910].

AdaTEST 95 and Cantata++ are updated regularly. For the latest information on their capabilities, please see IPL's website [www.iplbath.com/tools](http://www.iplbath.com/tools).

## 2. Cantata++, AdaTEST 95 and GSWS

### 2.1. Dynamic Testing

*Ref:*

5.1.3.7	<i>SW Acceptance Review (GSWS-RV-0530)</i>
6.5.1	<i>Phase Description (GSWS-SWENG-1740)</i>
6.5.2.1	<i>Structural Coverage Requirements (GSWS-SWENG-1800)</i>
6.6.3.2	<i>Verification of GTR(IT) (GSWS-SWENG-2460)</i>
11.3	<i>ISVV Activities and Milestones (GSWS-ISVV-4750)</i>
11.3.3	<i>Implementation Activities (GSWS-ISVV-4920, GSWS-ISVV-4940)</i>
12.2.5	<i>Algorithms File</i>
A.12	<i>Software Verification and Validation Plan (SVVP)</i>
A.13	<i>Software Unit Test Plan (SUP)</i>
A.14	<i>Software Integration Test Plan (SIP)</i>
A.28	<i>Algorithm Test Cases Definition (ALG-TCD)</i>
B.2.1	<i>White Box Testing</i>
B.2.1.4	<i>Fault Injection</i>
B.2.2	<i>Black Box Testing</i>
B.2.2.2	<i>Stress Testing</i>
B.2.2.3	<i>Robustness Testing</i>
B.2.3.1	<i>Test Data Selection from Boundary Value Analysis</i>
B.2.3.2	<i>Test Data Selection from Equivalence Partitioning</i>

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 5.1.3.7, GSWS-RV-0530] at the *unit* or *integration* level. Both unit testing and integration level testing are mandated by GSWS [Ref GSWS-SWENG-1740, GSWS-SWENG-2460].

*Unit tests* are designed to verify the *functional correctness of the code* [Ref GSWS-ISVV-4920] and *the accuracy and consistency of the source code* [Ref GSWS-ISVV-4940]. The tools place no restrictions on the techniques the user may apply to select *test cases*, which may include *fault injection* [Ref B.2.1.4], *Robustness Testing* [Ref B.2.2.3], *Stress Testing* [Ref B.2.2.2], and *test data selection from Boundary Value Analysis* [Ref B.2.3.1] and *Equivalence Partitioning* [Ref B.2.3.2].

Dynamic testing should be planned and pass/fail *validation success criteria* specified [Ref A.12, §6.3] 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 A.13, §4.2] and [Ref A.13, §4.3] as they are easily readable and auditable by quality assurance staff. The test scripts contain *test scenarios* and documentation of the *inputs, outputs* and *expected results* [Ref A.13, §4.3], [Ref A.28, §5] required by GSWS.

*The input to an integration test is the software that has been successfully tested at unit level* [Ref A.14, §4]. Integration testing focuses on the internal and external *interfaces* between the unit and external *software elements, hardware elements, or both* [Ref A.14, §4]. Cantata++ wrapping is particularly useful at integration level, providing a powerful simulation framework that tests class and function interfaces. Wrapping can also simulate difficult to test *scenarios* such as *hardware element* failure or memory failure.

AdaTEST 95 and Cantata++ support both *black box testing* [Ref B.2.1] and *white box testing* [Ref B.2.2] techniques. A *black box* test is a test of the software's externally visible functionality, whilst a *white box* test is a *structural test* of the internal workings (the algorithm) of the software under test. In the case of C++, Cantata++ provides an automated solution for the *white box* testing of private data in a class. Without such a solution, *white box* testing of C++ class implementations becomes extremely difficult.

AdaTEST 95 and Cantata++ provide rigorously developed *test harnesses* and *stub* generation tools to assist the process of unit and integration testing. These can help to meet the independence requirements of the ISVV [Ref GSWS-ISVV-4750]. AdaTEST 95 can also use TCD files (see section 1.1) to generate test scripts from. It is possible to use these files as the *Algorithm Test Cases Definition (ALG-TCD) document* [Ref 12.2.5].

Cantata++ and AdaTEST 95 can be used to test C and Ada for *flight software applications* [Ref GSWS-SWENG-1200]. Additionally, Cantata++ can test C++, and Cantata++'s ability to test C++ can be used in the case to *justify the selection of C++* [Ref GSWS-SWENG-1210].

## 2.2. Timing Analysis

*Ref:*

A.3 *Software REQ's Document (SRD), Database Req's (DBR)*  
(*Performance Requirements Section*)

AdaTEST 95 and Cantata++ provide timing analysis facilities which can be used to measure and check that *timing requirements* [Ref A.3, §4.2] have been met. *Numerical values for measurable variables, including response time* are provided by the tools. In AdaTEST 95 and Cantata++, performance requirements may also be represented as a *range of values*, with the *minimum level of performance allowed* and a *safe margin of performance above the acceptable value* being specified [Ref A.3, §4.2].

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.

## 2.3. Development Methods

*Ref:*

- 4.3.2 *Waterfall Model*
- 4.3.3 *Incremental Model*
- 6.2.2.1 *Galileo Methods List*
- 6.4 *Software Design Phase (GSWS-SWENG-1630)*
- 6.6.3 *Verification Requirements (GSWS-SWENG-2370 and GSWS-SWENG-2380)*
- 7 *Software Configuration Management (GSWS-CM-2790)*
- 11.1 *Independent SW V&V Process (General)*

Cantata++ and AdaTEST 95 can be used to assist testing of software developed using both structural and object oriented methods [Ref 6.2.2.1]. Cantata++ defines and implements object oriented test and coverage techniques which extend the traditional structural definitions, and apply them to object oriented contexts (see IPL's paper [Advanced Coverage Metrics for Object-Oriented Software](#) for more information). Cantata++ has also been integrated with market-leading UML tools (*UML is highly recommended in the Galileo project [Ref 6.2.2.1]*), providing seamless transition between development and testing, within the UML tool's IDE.

Cantata++ and AdaTEST 95 work well in both a *Waterfall Model* [Ref 4.3.2], used mainly for software with stable and definite requirements, and an *Incremental Model* [Ref 4.3.3]. In either case, *verification* of the behaviour model is required by the GSWS standard [Ref GSWS-SWENG-1630].

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

AdaTEST 95 and Cantata++ both provide provision for the dynamic checking of *exceptions* [GSWS-SWENG-2370 and 11.1].

Test software (including test scripts and test results files) are subject to *software configuration management* [Ref 7.0, GSWS-CM-2790 and A.40]. Both AdaTEST 95 and Cantata++ provide all test scripts and results in human readable, ASCII text format, making them extremely easy to check into and out of configuration management systems [Ref GSWS-CM-2790].

AdaTEST 95 and Cantata++ can both be integrated easily into Makefiles, batch files, scripts, and other automated environments, either on the command line, or within integrated development environments (IDE's). The tools are also sensitive to the build options (e.g. defines) that are used at build time. Thus, if different configurations of the software are being built (*e.g. table size, compilation options, run time code*) it is

easy to test the *correctness* of this *configurable code* with AdaTEST 95 and Cantata++ [Ref GSWS-SWENG-2380].

## 2.4. Host and Target Testing

*Ref:*

<i>Glossary</i>	<i>Glossary of Terms (Page 18)</i>
4.3.7	<i>Reference Life Cycle for Simulators (GSWS-LC-0370)</i>
6.6	<i>Software Integration and TS-Validation Phase</i>

AdaTEST 95 and Cantata++ are specifically designed to allow a seamless transition between testing in the host and *hardware target* [Ref Glossary] 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 6.6] 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.

Cantata++ and AdaTEST 95 can either be configured to run on real target hardware, or in a host based software simulator (or both if necessary). Thus the tools can help with the *validation Testing Reports for simulation validation at model and unit level* [Ref GSWS-LC-0370]

## 2.5. Test Reports

*Ref:*

5.1.3.4	<i>SW Test Readiness Review (SW-TRR) (GSWS-RV-0500)</i>
5.1.3.5	<i>SW Critical Design Review (SW-CDR) (GSWS-RV-0510)</i>
5.1.3.7	<i>SW Acceptance Review (SW-AR) (GSWS-RV-0530)</i>
9.3.3	<i>SW Product Assurance Verification Tasks (GSWS-PA-3605)</i>
12.3	<i>Contractual Applicability of SW Deliverables</i>
Table 12-D	<i>Contractual Applicability of DJF</i>
B.2.4.1	<i>Test Results Analysis</i>

The verification results 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.

These reports can be used directly in the *SW Test Readiness Review (SW-TRR)* [Ref GSWS-RV-0500], to assist with the *review of the Unit Test Results* [Ref GSWS-RV-0500, point 4] and *review of the integration test results* [Ref GSWS-RV-0510]. Cantata++ and AdaTEST 95 can also assist with the *verification that all the Technical Specification and interfaces requirements have been successfully validated and verified* during the *SW Critical Design Review* [Ref GSWS-RV-0510]. The reports can also be used by the *Software Product Assurance Representative* to verify that *statement coverage, tests coverage are achieved according to Section 6.5 and Section 6.6 of the GSWS* [Ref GSWS-PA-3605].

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. Thus the tools can be used to help *verify that the complete set of test cases is run on the same software version* [GSWS-RV-0530].

AdaTEST 95 and Cantata++ test reports can also be used to meet the *Contractual Applicability of SW Deliverables* [Ref 12.3] requirement, a key component of which is the *SW Verification Report (Acronym SVR)* [Ref Table 12-D Contractual Applicability of DDF].

Finally, AdaTEST 95 and Cantata++ test results can form an essential part of the *Test Results Analysis* [Ref B.2.4.1] phase, used to *assure the completeness of test activities*. As part of this review, *the reviewer should ask questions about whether the test objectives were met, whether test results matched the expected outputs and whether the results were traceable to the test cases* [Ref B.2.4.1]. This information is easily referenced in the AdaTEST 95 and Cantata++ test results and log files.

## 2.6. Regression Testing

Ref:

- 6.5.3.1      *Verification of Source and Executable Object Code (GSWS-SWENG-2130)*
- 6.6.3.2      *Verification of GTR (IT) (GSWS-SWENG-2450, GSWS-SWENG-2460)*
- 6.10         *Software Maintenance Phase (GSWS-SWENG-2730)*
- 8.2          *SW DAL Reduction Procedure (GSWS-RAMS-3420)*
- A.12         *Software Verification and Validation Plan (SVVP) (Section 5.5)*

The tools allow easy regression testing when previously developed or modified software has to be re-verified. Regression testing lends itself to *periodic*, automated re running of the software under test [Ref GSWS-SWENG-2130].

Regression tests are useful for products undergoing *corrective maintenance* [Ref GSWS-SWENG-2730]. The regression tests can be re-run automatically after any change to the maintained product – this will help verify that any fixes, *enhancements* or *modifications* to the software do not cause other parts of the system to fail.

Where *Multiple-Version Dissimilar* software systems are developed [Ref GSWS-RAMS-3420] 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.

The regression testing framework offered by the tools help with the *activity of re-verification*. They also help to ensure that *modifications have not caused unintended effects*, and that the *system still complies with its specified requirements* [Ref A.12, 5.5].

## 3. Test Coverage Analysis

Ref:

- 6.5.2.1 *Structural Coverage Requirements (GSWS-SWENG-1800, GSWS-SWEG-1810)*
- 6.5.2.2 *Tests Methods Requirements (GSWS-SWENG-1830, GSWS-SWENG-1870, GSWS-SWENG-1880, GSWS-SWENG-1895, GSWS-SWENG-1900)*
- 6.5.3.1 *Verification of Source and Executable Object Code (GSWS-SWENG-1980, GSWS-SWENG-1990)*
- A.13 *Software Unit Test Plan (SUP)*
- B.1.2.2 *Finite State Machines*
- B.2.1.1 *Statement Coverage*
- B.2.1.2 *Decision Coverage*
- B.2.1.3 *Modified Condition/ Decision Coverage*

The dynamic analysis parts of AdaTEST 95 and Cantata++ monitor *structural coverage* [Ref 6.5.2.1]. *Statement Coverage*, *Decision Coverage* and *Modified Condition & decision coverage* of source code are all provided by both Cantata++ (for C,C++) and AdaTEST 95 (for Ada83 and Ada95) [Ref GSWS-SWENG-1800].

The coverage values that have been achieved during testing are reported in full by Cantata++ and AdaTEST 95, and their values can be checked within a test script to ensure that a particular coverage target has been achieved. This helps with GSWS Requirement GSWS-SWEG-1810, which requires that additional test cases be added to reach the required coverage level. Cantata++ Studio additionally offers coverage by test case, showing clearly which statements, decisions and Boolean expressions have been executed by which test case. This information is useful in the *Test Coverage Matrix*, which requires a *table with the trace between unit test cases and Software Code* [Ref A.13, 5]. Cantata++ also offers a powerful simulation framework called **wrapping** to help achieve coverage in `difficult to test` situations, such as out of memory, or target hardware failure.

Call coverage can be used to check that all modules in the software under test are invoked at least once, thus ensuring *interface testing (in terms of data coupling coverage)* is achieved [Ref GSWS-SWENG-1830]. *Boundary Value Analysis* [Ref GSWS-SWENG-1870] and *Equivalence Partitioning* [Ref GSWS-SWENG-1880] can be achieved with the help of relational coverage offered by Cantata++, and decision coverage offered by Cantata++ and AdaTEST 95.

*Statement Coverage* [Ref B.2.1.1] is a *technique to produce test cases that exercise at least once each statement of the program* [Ref B.2.1.1]. *Statement Coverage* statistics and reports produced by AdaTEST 95 and Cantata++ 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* [Ref B.2.1.2] is a *technique to produce test cases that exercise at least once each branch of the program* [Ref B.2.1.2]. *Decision Coverage* is provided by AdaTEST 95 and Cantata++ 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.

*Modified Condition/Decision Coverage (MC/DC)* [Ref B.2.1.3] is provided by AdaTEST 95 and Cantata++ 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.

Cantata++ supports the concept of user-defined contexts in coverage gathering. Using this innovative technique, it is possible to define state based coverage, where the execution of code is measured on a per-state basis. This technique is particularly useful in discovering faults when the system is defined as a *finite state machine* [Ref B.1.2.2].

Accidental execution of de-activated code can be detected by Cantata++ through the advanced use of infeasible pragmas [Ref GSWS-SWENG-1900].

AdaTEST 95 and Cantata++ coverage reports can help identify where dead or deactivated code is present [Ref GSWS-SWENG-1980 and GSWS-SWENG-1990].

## 4. Static Analysis and Test Metrics

Ref:

Table 9-A      *The Quality Model Framework*

9.4.1          *The Galileo Software Quality Model Framework*

B.1.3.1        *Static Code Analysis*

B.4.3          *Language Subset*

D.6            *MAIN\_AN\_1: Cyclomatic Number*

D.7            *MAIN\_MO\_1: Nesting Levels*

The Galileo Software Quality Model Framework is a *mechanism to define a consistent and useful metrics program* [Ref 9.4.1]. AdaTEST 95 and Cantata++ help to prove the *functionality, reliability and maintainability* of the system as defined by this Quality Model Framework, through the use of *Static Code Analysis* [Ref B.1.3.1].

The Static Analysis facilities provided by AdaTEST 95 and Cantata++ calculate 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* [Ref D.6] 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 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 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 [Ref B.4.3]. Static analysis results can also be exported to spreadsheets and databases.

The tools provide many of the well-known *complexity* [Ref D.6] 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.

Static analysis performed by AdaTEST 95 and Cantata++ can also detect *dead (unreachable) code* [Ref B.1.3.1]. Metrics are also provided for the *nesting level* of the control structure [Ref D.7].

## 5. Tool Integrity and Development Standards

*Ref:*

B.4.2	<i>Qualified Tools</i>
B.4.3	<i>Language Subset</i>
3.3.2	<i>Contribution of Software to upper-level RAMS Analyses</i>
9.5	<i>Sub-Contractor Audits</i>
10.4.3	<i>Transfer Phase</i>
10.4.4.1	<i>Procured Software Tools</i>

The integrity of the toolsets used in the development of safety critical aerospace software is a significant consideration. This is especially true of dynamic testing tools.

AdaTEST 95 and Cantata++ have been developed to the highest practical standards and comply with the requirements of GSWS on *Tool Qualification* [Ref B.4.2] for *procured Software Tools* [Ref GSWS-PR-4510]. 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) [Ref 3.3.2].
- d) Independent Audit (AdaTEST 95) [Ref 9.5].
- e) The use of a safe language subset [Ref B.4.3] 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. Clients and Certification Authorities wishing to audit the development and continuing maintenance of the products may do so by arrangement with IPL [Ref GSWS-PR-4490].

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

## 6. Conclusion

In summary, the table below shows where AdaTEST 95 and Cantata++ static analysis and dynamic coverage analysis can help developers to develop software to the GSWS standard. Requirements specified in the two left hand columns can be found in [Ref Table 9-A].

Goal Properties	Associated metrics	How IPL Can Help
Functionality	Requirements Allocation Tests and Valid. Coverage Completeness.	Full coverage reports provided by Cantata++ and AdaTEST 95
	Testing/Validation Progress	Regression testing framework offered by Cantata++ and AdaTEST 95
Reliability	Structural Coverage	Structural Coverage reports provided by Cantata++ and AdaTEST 95
Maintainability	Cyclomatic Number	Provided by Cantata++ and AdaTEST 95 Static Analysis
	Nesting Levels Modularity Size Profile Number of Exit Number of Entry	Provided by Cantata++ and AdaTEST 95 Static Analysis
	Code Comment Frequency	Provided by Cantata++ and AdaTEST 95 Static Analysis
Suitability for Safety	Safety Activities Adequacy	Cantata++ and AdaTEST 95 in use on many safety critical projects
System Engineering Effectiveness	Code Size Stability	Provided by Cantata++ and AdaTEST 95 Static Analysis Metrics

AdaTEST 95 and Cantata++ are well suited to the development of software to the GSWS 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.