

IPL Testing Tools and the MISRA Development Guidelines

Executive Summary

This paper describes how AdaTEST and Cantata++ can be used within a software development adhering to the MISRA Development Guidelines for Vehicle Based Software. In particular, it shows how AdaTEST and Cantata++ can be used to meet the verification and testing requirements of the guidelines. 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 in the development of vehicle based software.

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 guidelines.

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: 30/05/2002 11:33
File: MISRA.DOC*

1. Introduction

The MISRA Development Guidelines for Vehicle Based Software describes software engineering practices for the development of all vehicle based software, from software with no safety implications through to safety critical software. The guidelines have been produced by the Motor Industry Software Reliability Association (MISRA), an organisation of major motor manufacturers, component suppliers and consultants to the automotive industry.

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 AdaTEST and Cantata++ are sufficiently flexible for the testing of any Ada, C and C++ software.

This paper describes how AdaTEST and Cantata++ can be used within a project applying the MISRA Development Guidelines for Vehicle Based Software. It should be read in conjunction with the guidelines. To assist in cross referencing to the guidelines, key items identified by the guidelines are shown in *italics* in the text of this paper.

The description consists of two parts. Section 2 gives a general description of AdaTEST and Cantata++. Sections 3 and 4 review verification and testing activities required by the MISRA Development Guidelines for Vehicle Based Software, and how AdaTEST and Cantata++ can be used in relation to the guidelines.

The MISRA Development Guidelines for Vehicle Based Software make reference to *fundamental concepts* drawn from a range of present and emerging standards. In support of these standards, IPL provides the following AdaTEST and Cantata++ support papers:

- IPL Testing Tools in the ISO9001/BS5750/TickIT Software Quality Process;
- IPL Testing Tools and the IEC 61508 standard.

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 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.

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 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 CASE tools.

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.

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

AdaTEST and Cantata++ static analysis provides static analysis of source and also instruments 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.

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 non-instrumented code.

Software development in the automotive industry is primarily concerned with C and assembly languages. For readability, the following two sections of this paper consequently discuss the use of Cantata++ within a framework of the MISRA Development Guidelines for Vehicle Based Software. However, readers should bear in mind that AdaTEST has similar capabilities should the Ada language be used.

3. The MISRA Development Guidelines for Vehicle Based Software - Software Lifecycle

3.1. Project Planning

As a CASE tool, or more specifically a CAST (Computer Aided Software Testing) tool, the use of Cantata++ should be identified during *project definition*. *Verification and validation* in the *lifecycle plans* should identify unit (module) testing and software integration testing as phases of the lifecycle where Cantata++ is used.

Cantata++'s static analysis reports, test case definitions, test scripts and test results should be identified amongst the *specific software deliverables*. The *verification plan* should identify how Cantata++'s *static analysis* and *dynamic testing* will be used. In particular, Cantata++ supports *black box testing* for *functional* tests, *performance* tests and *stress* tests, and supports *white box testing* for *structural* tests, *path* tests and *branch* tests, with automated analysis of *decision coverage* and *condition coverage*.

To achieve independence from the design and implementation team, Cantata++ scripts may be prepared and executed by a person or team exhibiting a *degree of independence* appropriate to the integrity requirements of the software.

Cantata++ test case definition files, test scripts and test result files may also be passed to an *independent assessor and/or auditor* as part of the planned product assessment.

Where software is being considered for *reuse*, evidence of appropriate thorough testing with Cantata++ will provide confidence that the software proposed for reuse meets integrity requirements.

3.2. Integrity

Cantata++ has been developed according to the IPL Quality Management System (QMS), which has been accredited to ISO 9001 and TickIT. It has been used extensively for the verification and testing of software at all *integrity levels*. For example, in air traffic control systems, in aircraft flight recorders, in road traffic systems and in automotive systems.

The IPL QMS does not map directly onto MISRA Development Guidelines for Vehicle Based Software, having originated from requirements of the defence industry, but the development practices used do fulfil all of the objectives of the MISRA Development Guidelines for Vehicle Based Software.

All Cantata++ software has been fully *module tested* to achieve *100% white box coverage* of all Cantata++ *coverage analysis* metrics, including coverage of statements, decisions, and complex conditions. The IPL support paper "Structural Coverage Metrics" provides detailed discussion of the merits of a wide selection of structural coverage metrics, showing that those supported by Cantata++ are both practical and effective.

Revisions to Cantata++ are formally *configuration managed*, through a formalised *change and build control* process.

Of particular note is the degree of defensive programming employed in the development of Cantata++ to ensure that there are no failure modes which could result in a test being passed incorrectly. When Cantata++ is used, all internal data is systematically checked

after each test case to verify that errors in the software under test cannot lead to undetected corruption of the test results.

Users of Cantata++ are therefore assured that it has both been produced to a high standard, and proven through extensive use, such that its use for dynamic testing cannot compromise the integrity of the software being tested.

Cantata++ and AdaTEST can be used to test software written in the full C, C++ and Ada languages, or *restricted subsets* of these languages. Dynamic testing and coverage analysis with Cantata++ can be used to show that the compiled *object* code accurately reflects the *source* and the design of the software. When *assembly languages* are used, Cantata++ can be used to test assembly language software from scripts written in C (provided that the assembly language can be called from C). However, Cantata++ cannot measure test coverage of software written in *assembly languages*.

All Cantata++ *static analysis* outputs and dynamic test results are time and date stamped as an aid to *configuration management*. IPL recommend that Cantata++ outputs should be associated with the item of software tested for *configuration management* purposes.

3.3. Requirements Specification

When preparing the requirements for control systems, Cantata++ can be used to exercise and evaluate *models, animations* and *prototypes* of *complex control calculations* which are used to *validate* requirements.

3.4. Design

Cantata++ is compatible with both traditional structured design methods and with *object oriented* design. The design of software can have a significant impact on its testability, irrespective of the design method used. As *verification and validation* effort represents a significant proportion of the software lifecycle, *designing for verification and validation* is a worthwhile investment.

Automotive applications involve embedded systems and host-target development. *Verification and validation* plans may therefore benefit from host-target testing. See the IPL product support paper "Host-Target Testing", which describes techniques for host-target testing and for designing software for ease of testing in both the host and target environments.

Cantata++ places no restrictions upon the use of *floating point arithmetic*. Cantata++ can be used to test any numeric type, including both hardware and software *floating point arithmetic*.

If modelling is used as an aid to the design of control systems, Cantata++ can be used to control the model. During the integration testing phase of the lifecycle, timing measurements made using Cantata++ may be fed back to improve the accuracy of design models.

Where *redundancy and diversity* are used to improve integrity, 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 diverse implementations. The portability of Cantata++ and its test scripts between development platforms and target platforms further facilitates this technique.

3.5. Programming

Cantata++'s *static analysis* facilities can be used to help enforce *programming codes of practice*. In particular, *language usage* can be monitored and the use of language features can be checked for conformance to a *restricted subset*. Furthermore, Cantata++ provides a range of *complexity* and *structure* metrics which can also be used to enforce design and programming complexity restrictions.

Cantata++'s *static analysis* outputs may optionally be exported to a data analysis tool such as a spreadsheet. IPL can provide macros for Microsoft Excel to assist with analysis of Cantata++ metrics. Static analysis reports and subsequent spreadsheet analysis can provide useful insight during *code reviews and walkthroughs*.

Cantata++ static analysis results may also be checked automatically against *programming standards* during execution of a Cantata++ test script.

3.6. Testing

Dynamic testing with Cantata++ is controlled from a test script. The test script may be either written manually or generated automatically from a set of test case definitions. Negative testing is testing to make sure that software does not do anything that it shouldn't. When test scripts are generated automatically, Cantata++ automatically generates substantial negative testing to assist with the *discovery of errors*.

Each test case is specified in terms of the *test inputs* and the expected *test outcome*, as a pass or *acceptance criteria*. Where *likely failure modes* have been identified by the designer, specific test cases can be incorporated to *discover errors* leading to such failure modes.

Dynamic tests are then executed by building the test script into a test executable with the *module* or *integration* of modules being tested. During test execution, Cantata++ performs *coverage analysis* for *statements*, *branches* (decisions) and *conditions*.

Traceability from test scripts to the appropriate *software specification* can be shown by placing appropriate comments in test scripts, which are echoed in test results during test execution. Traceability management tools can then be used to verify that all requirements and design statements have been tested.

Cantata++ supports both *black box* and *white box* testing. IPL recommends an isolation approach to module testing, as described in the IPL support paper "Organisational Approaches to Unit Testing". Techniques for selecting test cases, including *branch* (decision) testing, *condition* testing, *equivalence partitioning* and *boundary value analysis* are described in the IPL support paper "Designing Unit Test Cases".

3.7. Product Support

Cantata++ is a development tool and is not suited to on-board or *off-board diagnostics* of operational products. However, thorough module testing and integration testing with Cantata++ will give major benefits during *software maintenance*. Automated and repeatable tests facilitate continued application of the *procedures used for product development* through reducing the cost of full regression testing when *post-production modifications* are made.

When *changing or upgrading a development environment*, the high portability of Cantata++ means that all test scripts can be easily moved to the new environment. Furthermore, IPL is committed to maintaining backward compatibility from all new releases of Cantata++, so that upgrades to Cantata++ have minimal impact on existing test scripts.

4. The MISRA Development Guidelines for Vehicle Based Software - Software Quality Planning

Cantata++ test scripts are written in a constrained subset of the C language. Any engineer with *practical experience* in C programming can readily understand Cantata++ test scripts. If it is planned to use the Cantata++ test case definition facility to generate test scripts automatically, then engineers may benefit from IPL's one day training course.

Where engineers are familiar with programming, but lack experience in testing, IPL offers a further one day training course entitled "Unit Tests Planning and Implementation". This course teaches basic testing practices and is independent of the tools used.

Cantata++ may be used to support a comprehensive *software process metrics* activity. Cantata++ static analysis results may be output to a spreadsheet and analyzed against *software process metrics*. An example of such a metrics program is described in the IPL paper "Metrics Collection in Code and Unit Test as Part of continuous Quality Improvement", which was published in the proceedings of EuroSTAR93 and in the Journal of Software Testing Verification & Reliability.

Where *subcontracting* is involved, the maintenance of software by the prime contractor is often a consideration. To ensure subcontracted software is both reliable and maintainable, a prime contractor could require a subcontractor to test all software using Cantata++. Even if a subcontractor is not using Cantata++, the prime contractor can use Cantata++ to help *audit* the quality of delivered software. For subcontract management, IPL offers a three day training course entitled "The Procurement of Complex Systems".

5. Conclusion

AdaTEST and Cantata++ are well suited to the development of software to the MISRA Development Guidelines for Vehicle Based Software, facilitating a high degree of automation of the verification and test techniques required for effective use of the guidelines.

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

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 the cost effective testing of software at all levels of safety criticality, from a normal ISO9001 development process through to level 4.