LDRA
Software Technology

Technical White Paper

Understanding and Measuring Data Coupling & Control Coupling

www.ldra.com

Contents

Modular software is often seen as a feature of high quality software, but that assertion only holds true if the modules interact as expected. That is why DO-178C [1] & ED-12C [2] (henceforth referenced collectively as "DO-178C") require that both data coupling and control coupling are shown to be in accordance with requirements.

Section 6.4.4.2.c of DO-178C requires "analysis to confirm that the requirements-based testing has exercised the data and control coupling [DCCC] between code components". This directive is designed to ensure that design, integration, and test objectives are met. The document outlines how measuring control and data coupling can be achieved via a combination of control and data flow analysis, and how that process can be made more efficient using software tools.

This document explains these related coupling concepts in the general case, discusses the metrics used to measure them, and compares the available approaches to automate those analyses. It closes by considering the additional considerations implied by the specific case of multicore processor adoption.

Requirements based testing

Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements. It tests both functionality and non-functional attributes such as performance, reliability, or usability. Defects detected during the testing process go through the defect life cycle and are tracked to resolution, and defect statistics provide a measure of status of the project status.

In general, there are five discreet stages to requirements based testing:

- **Definition of test completion criteria:** Testing is completed only when all the functional and non-functional testing is complete.
- **Test case design:** Each test case has five parameters namely the initial state or precondition, data setup, the inputs, expected outcomes and actual outcomes.
- **Test execution:** Each test case is executed against the system under test and document the results.
- **Result verification:** For a test to be passed, its expected results must match actual results.
- **Test coverage verification:** Ensure that there are tests covering both functional and non-functional aspects of the requirement.

DO-178C requires bidirectional requirements traceability across the lifecycle, from system requirements to software high-level requirements, from software high-level requirements to low-level requirements, and through to test cases, test procedures and test results. Low-level requirements must then be linked to the source code in which they are implemented. Structural coverage analysis (code coverage, data coupling and control coupling) quantifies the extent to which the source code of a system has been exercised by the testing process.

Using these practices, it is possible to ensure that code has been implemented to address every system requirement, that the implemented code has been tested to completeness, and there is no code that is not attributable to a requirement. It is possible to achieve bidirectional traceability by traditional methods, but automating the process makes it is much less error prone and labour intensive. (Figure 1)

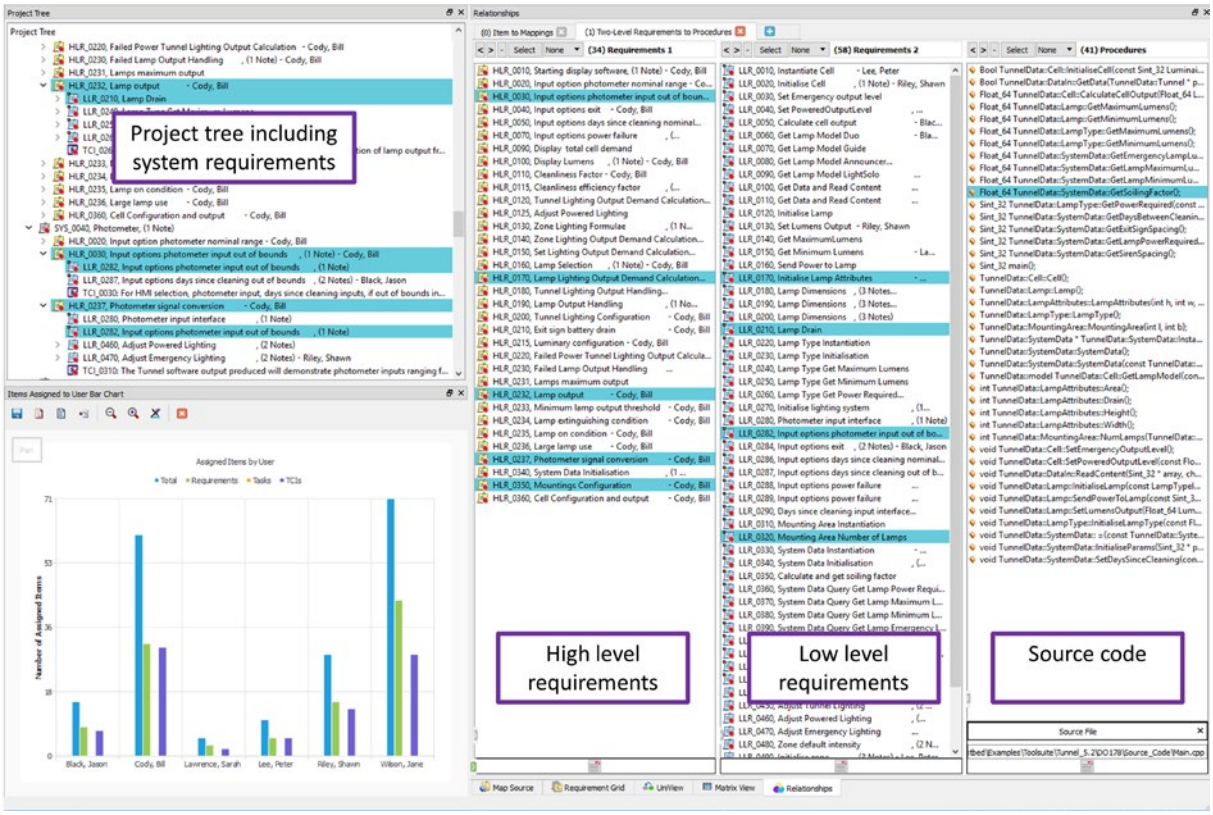


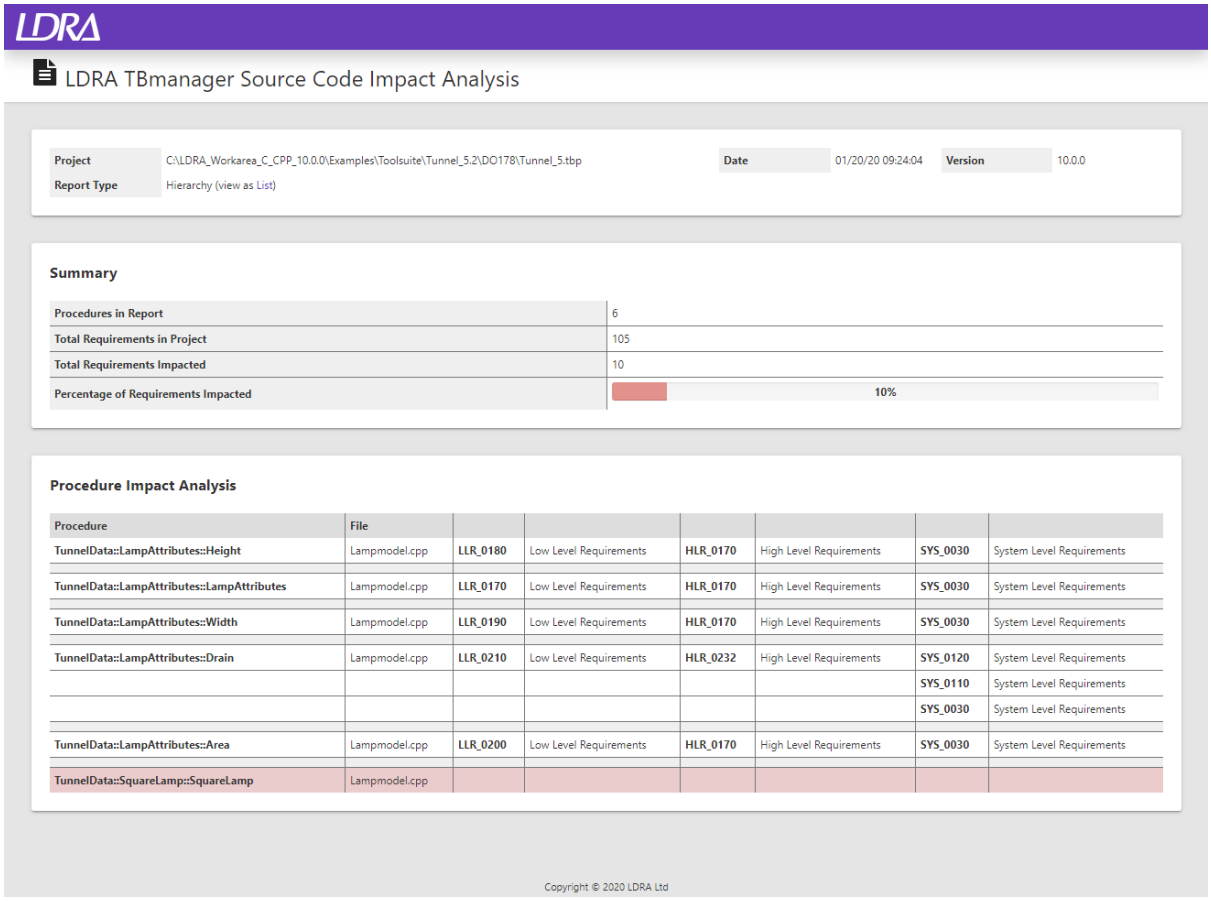
Figure 1: Automating requirements traceability with the TBmanager [3] component of the LDRA tool suite [4]

Change Impact Analysis (CIA)

In general, Change Impact Analysis (CIA) is used to assess the potential effects of changes made to a system. It can be particularly important in large-scale software development projects, where even a seemingly small change can lead to unintended consequences that disrupt the entire system.

In the current context, Change Impact Analysis (CIA) enables teams to assess how a modification in one component might affect coupled components elsewhere in the system. This is crucial for maintaining system integrity over time, particularly in environments subject to incremental development and re-certification.

Performing CIA through traditional manual means is laborious and prone to error. Automating it by leveraging requirements traceability and project baseline analysis helps to address what is often a project management headache (Figure 2).



LDRA TBmanager Source Code Impact Analysis

Project	C:\LDRA_Workarea_C_CPP_10.0.0\Examples\Toolsuite\Tunnel_5.2\DO178\Tunnel_5.tbp	Date	01/20/20 09:24:04	Version	10.0.0
Report Type	Hierarchy (view as List)				

Summary

Procedures in Report	6
Total Requirements in Project	105
Total Requirements Impacted	10
Percentage of Requirements Impacted	10%

Procedure Impact Analysis

Procedure	File						
TunnelData::LampAttributes::Height	Lampmodel.cpp	LLR_0180	Low Level Requirements	HLR_0170	High Level Requirements	SYS_0030	System Level Requirements
TunnelData::LampAttributes::LampAttributes	Lampmodel.cpp	LLR_0170	Low Level Requirements	HLR_0170	High Level Requirements	SYS_0030	System Level Requirements
TunnelData::LampAttributes::Width	Lampmodel.cpp	LLR_0190	Low Level Requirements	HLR_0170	High Level Requirements	SYS_0030	System Level Requirements
TunnelData::LampAttributes::Drain	Lampmodel.cpp	LLR_0210	Low Level Requirements	HLR_0232	High Level Requirements	SYS_0120	System Level Requirements
						SYS_0110	System Level Requirements
						SYS_0030	System Level Requirements
TunnelData::LampAttributes::Area	Lampmodel.cpp	LLR_0200	Low Level Requirements	HLR_0170	High Level Requirements	SYS_0030	System Level Requirements
TunnelData::SquareLamp::SquareLamp	Lampmodel.cpp						

Figure 2: Automating Change Impact Analysis [5] with the LDRA tool suite

Data Coupling & Control Coupling (DCCC)

DO-178C includes the following definitions in Appendix B:

- "**Data Coupling:** The dependence of a software component on data not exclusively under the control of that software component",
- "**Control Coupling:** The manner or degree by which one software component influences the execution of another software component"

DO-178C § 6.4.4.2.c specifies a requirement for "analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components."

Data coupling analysis is focused on the observation and analysis of data elements as they are set and used ("set/use pairs") across component boundaries during requirements-based tests.

In the context of DO-178C, a Computer Software Configuration Item (CSCI) is a discrete, identifiable unit of software that is developed, managed, and verified as a standalone component within a system's configuration management and certification framework. Coupling analysis across these CSCI boundaries is a critical activity, and it allows developers to isolate and verify the intended interactions within well-defined software boundaries.

Also leveraging requirements-based tests, control coupling analysis is focused on identifying where one software component or CSCI influences the control flow of another — typically through the passing of flags, operation codes, or other parameters that determine internal logic decisions. The goal is to ensure that such interactions are clearly defined, constrained by the design, and consistent with the software requirements.

The manual analysis of coupling using debuggers or trace logs is fraught with overhead, scalability, and fallibility issues. Automating the analysis process significantly reduces the effort required and improves consistency across verification activities.

DCCC: The case for a hybrid static & dynamic approach

Under DO-178B [6], the predecessor to DO-178C, the treatment of software coupling was vague and open to interpretation. While the standard required verification of software integration and interface consistency, it offered no explicit guidance on analysing how components interact through shared data or control logic. As a result, coupling analysis was often overlooked or addressed informally, with no consistent expectation across projects or certification authorities.

DO-178C addressed this ambiguity by introducing a clear requirement to analyse both data coupling and control coupling at Design Assurance Level A, as formalised in section 6.3.4.d and included in Table A-4 of the standard. It requires that component integration verification includes an explicit assessment of how components depend on and influence each other through data and control mechanisms.

Crucially, this analysis cannot rely on static methods alone. While static analysis can help identify potential interactions, it cannot confirm whether those interactions occur or behave correctly during execution. As clarified in CAST-19 “Data Coupling and Control Coupling” [7], meeting the DO-178C objective demands evidence gathered through requirements-based testing, making dynamic analysis essential at DAL A. Only runtime verification can confirm that data and control flows between components operate safely and in accordance with the system’s requirements.

Static analysis matters!

Although dynamic analysis is therefore essential for verifying that data and control couplings behave correctly during execution in accordance with DO-178C, in isolation it is limited to the interactions exercised by test cases. Any unexecuted paths or conditions remain unanalysed, leaving gaps in coverage that are unacceptable in high-integrity systems.

Static analysis addresses this by providing a complete view of all potential data and control dependencies, regardless of runtime behaviour. It can uncover hidden or indirect couplings that tests may not trigger, ensuring that no critical interactions are overlooked.

Used together in an integrated hybrid approach, static and dynamic analysis offer both completeness and precision: static identifies all possible couplings, while dynamic confirms which ones occur and behave correctly. This combined approach is essential for robust, standards-compliant verification.

Automated hybrid coupling analysis from LDRA

The patented [8] hybrid approach adopted in the LDRA tool suite begins by performing static analysis, using software comprehension and control/data flow engines to identify instances of control coupling.

Once these control couples are identified, targeted control coupling test runs are defined by the tool and executed using dynamic analysis, providing evidence of how these interactions behave at runtime. The tool derives control coupling coverage based on the portions of code exercised during these tests.

In parallel, the mechanism identifies data couples within the code - specifically, relationships between variables and parameters that span component boundaries. Using insights from the static analysis, corresponding data coupling tests are automatically generated and executed on instrumented code, ensuring that relevant execution paths are exercised. The outcomes of these tests form the basis of the data coupling coverage report.

Finally, the results of both control and data coupling analyses are consolidated into a comprehensive report, providing a clear, test-driven assessment of coupling coverage across the source code.

Control coupling analysis reporting

The procedural/functional call coverage reporting illustrated in Figure 3 supports verification of "The manner or degree by which one software component influences the execution of another software component." (DO-178C § 6.4.4.2.) It summarises the completeness of testing in accordance with the control coupling objectives specified in DO-178C.

The uppermost image in Figure 3 shows a report where 17 control calls have been identified, with 2 relating to intra-file calls (within this file) and the remainder relating to external calls. The lower image shows a report that identifies the procedures within a particular file, the control couples associated with procedures, and the extent to which they have been exercised.

These reports help to identify any gaps and guide targeted verification activities and ultimately provide evidence that all couples have been validated.

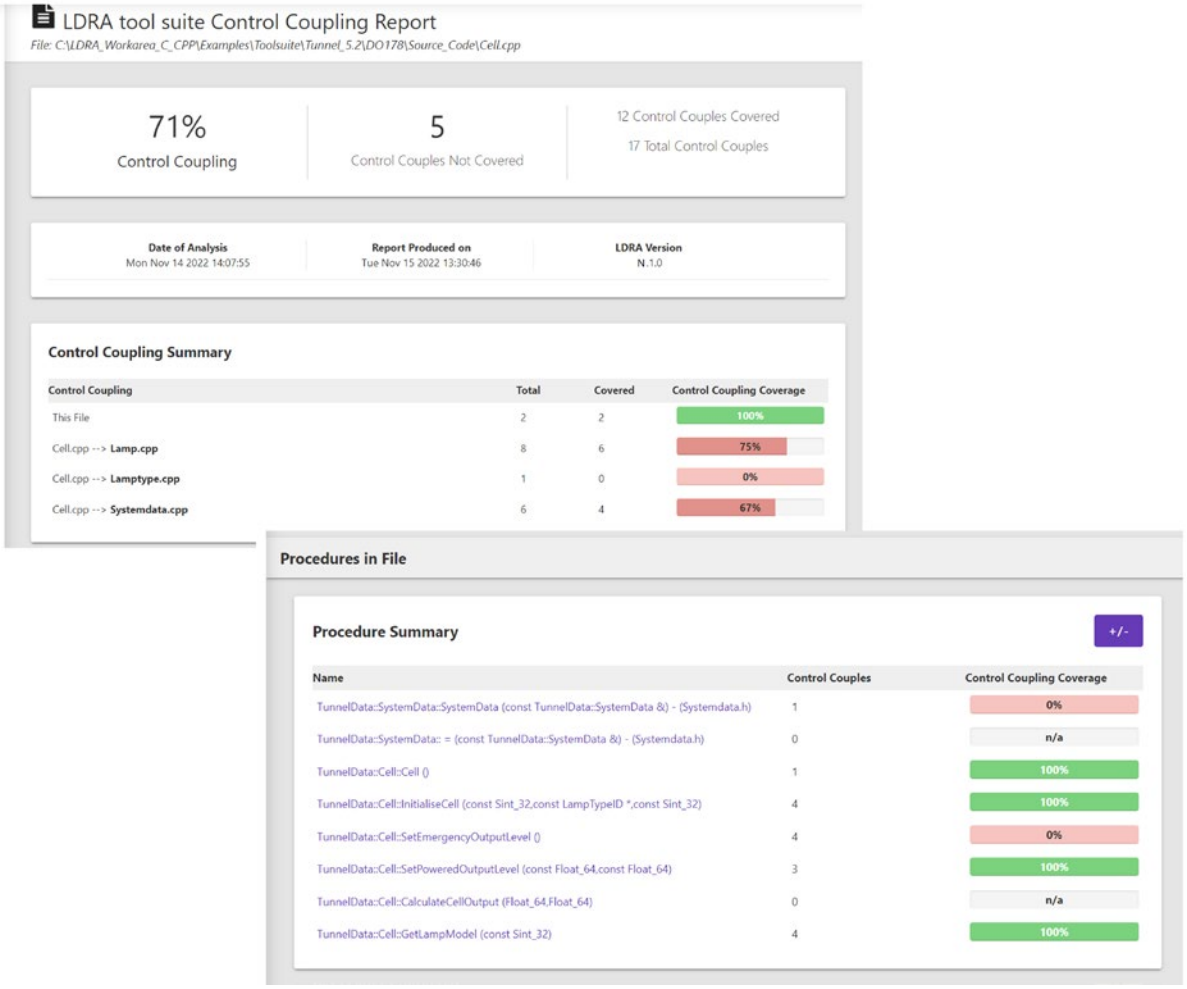


Figure 3: Control coupling analysis with the LDRA tool suite

Data coupling analysis reporting

The data coupling reporting illustrated in Figure 4 supports the verification of “the dependence of a software component on data not exclusively under the control of that software component.” (DO-178C § 6.4.4.2.) It summarizes the completeness of testing in accordance with the data coupling objectives specified in the DO-178 standard.

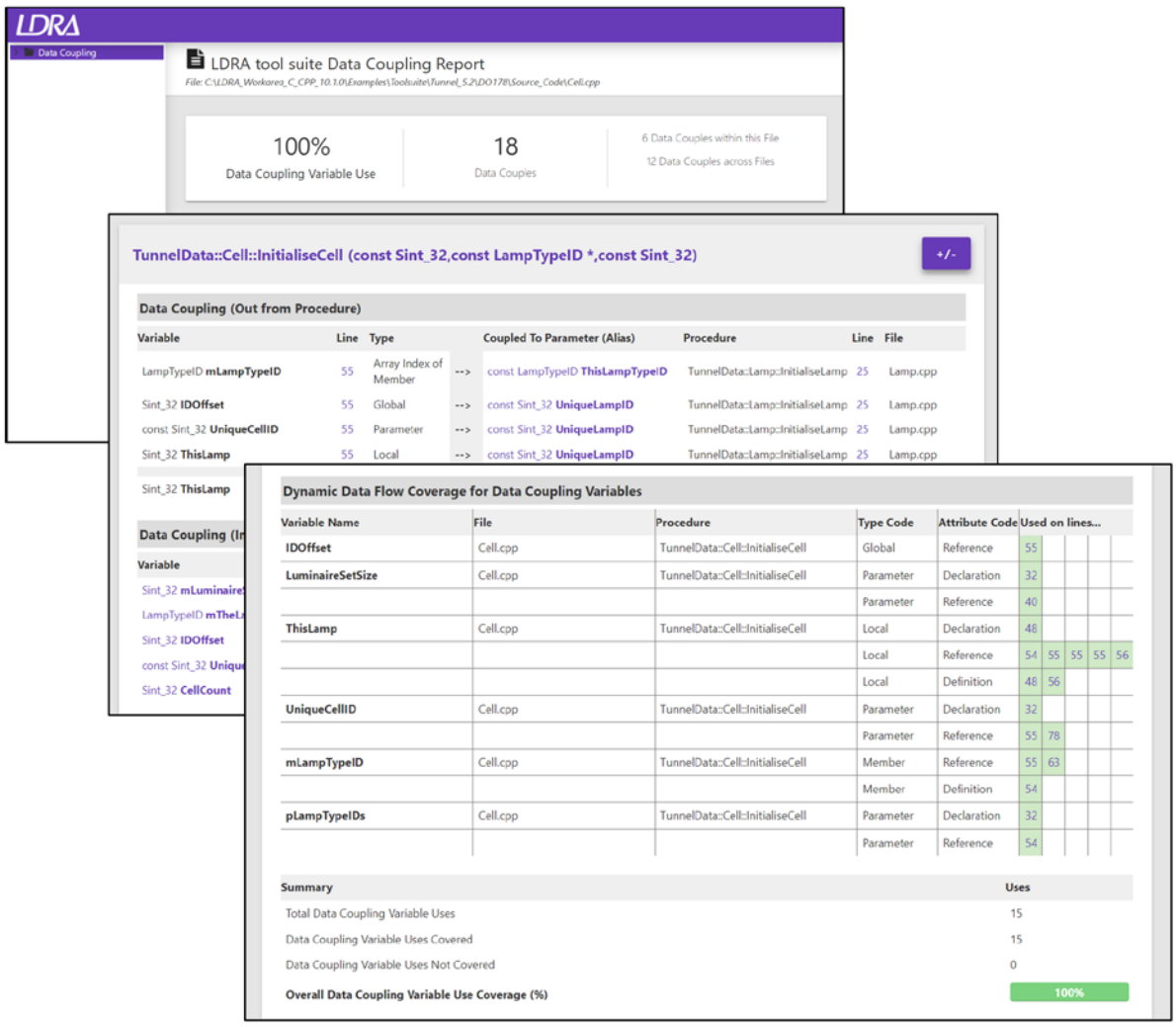


Figure 4: Data coupling analysis with the LDRA tool suite

Multicore processor considerations

The increasing adoption of multicore processors (MCPs) in airborne systems promises clear benefits - enhanced performance, lower power consumption, and reduced size and weight. However, these advantages come with increased complexity in the development process - including DCCC.

In multicore systems, shared resources - such as interconnects, memory, and cache subsystems - introduce interference channels that can obscure or disrupt observable data and control flows between components operating on different cores (Figure 5).

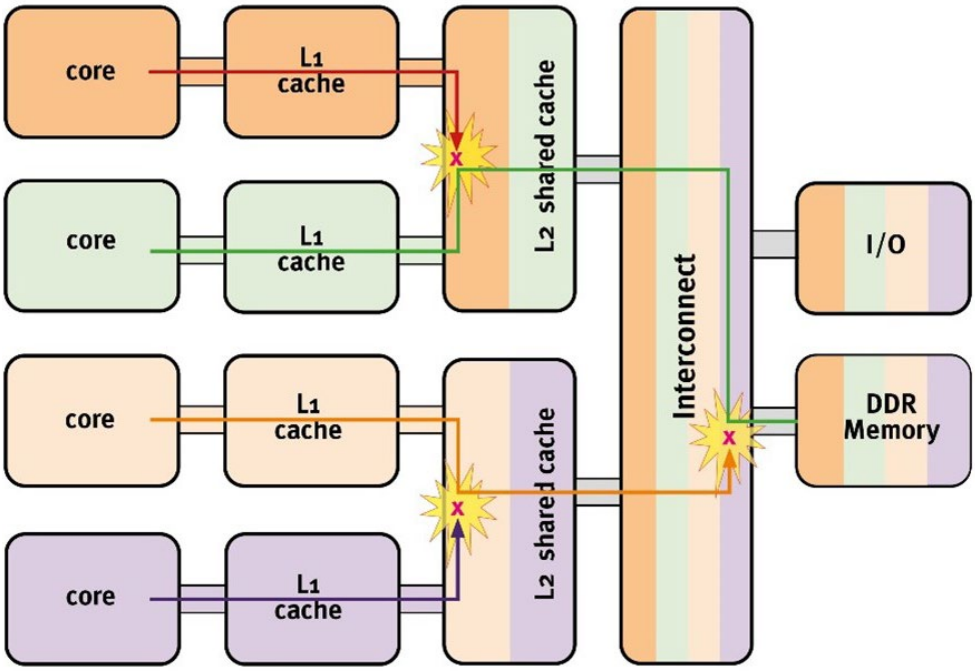


Figure 5: A representation of hardware interference in shared hierarchical memory

Interference is a critical concern in hard real-time systems, as it affects both the duration and predictability of task execution times. This variability can undermine the system's ability to meet strict timing constraints, potentially leading to deadline violations and compromising overall system reliability and safety (Figure 6).

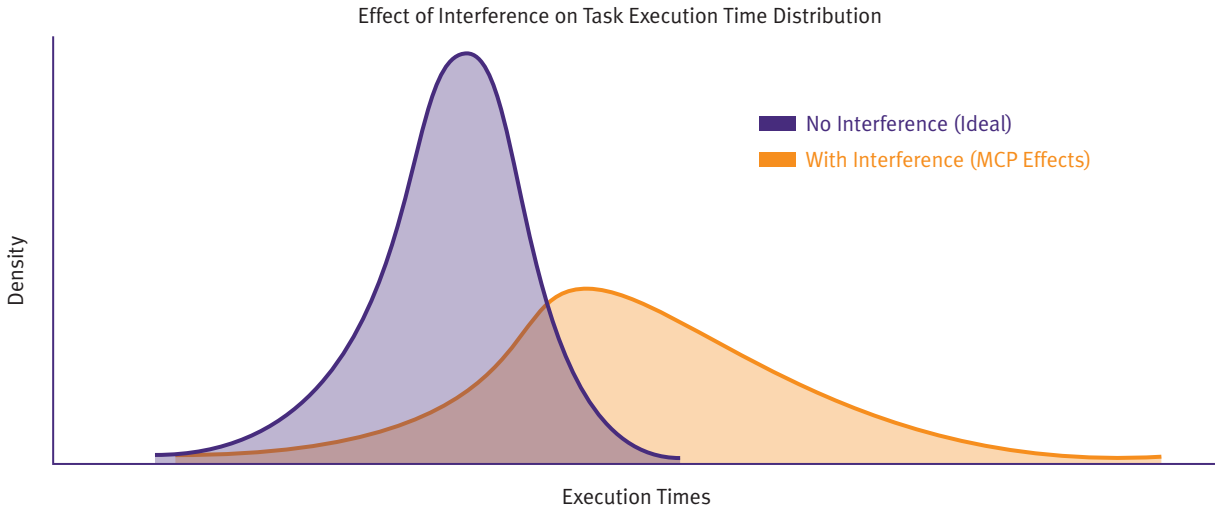


Figure 6: Interference affects both the duration and predictability of task execution times.

For example, a producer task on one core may write data to a shared memory buffer, which a consumer task on another core subsequently reads. Although the logical data coupling exists, the actual communication may be affected by cache coherence policies, memory access timing, or core-level scheduling, making it difficult to observe and validate these interactions through static analysis alone. Similarly, control coupling - perhaps a task on one core altering execution logic on another via control flags, for example - may be indirect and timing-sensitive, necessitating runtime confirmation.

As in the general case, complete coupling analysis for each CSCI is essential, and in multicore environments it must demonstrate that inter-core interactions - such as a producer task on one core and a consumer task on another - are correctly exercised and verified.

An Asymmetric Multicore (AMP) setup with shared L2 cache provides the basis for a more nuanced example of those challenges. In this instance, two entirely independent OS are running on separate cores, and no shared memory or application interaction has been designed into the system. Interference between interconnects, the L2 cache, or elsewhere will result in indirect, time sensitive coupling causing runtime variability.

These factors combine to make a hybrid approach especially vital in multicore systems considering A(M)C 20-193 [9] (formerly CAST-32A [10]), which provides guidance for demonstrating multicore airworthiness. A(M)C 20-193 stresses the importance of understanding how shared resources can influence timing and behaviour, reinforcing the need to verify that inter-core data and control interactions do not undermine determinism or safety.

The LDRA approach to multicore coupling analysis

The LDRA hybrid approach to DCCC addresses the challenges posed by multicore systems. The static analysis phase recognises shared variables, control parameters, global state, and inter-task interactions - whether they are intra- or inter-core.

As for the general case, the process continues with dynamic analysis, where requirements-based test cases are applied to instrumented software. LDRA's instrumentation technology isolates coverage metrics and coupling observations on a per-core basis. This allows for precise evaluation of inter-core interactions without conflating behaviours from unrelated tasks.

Graphically presented results enable engineers to visualise data and control coupling paths, trace them back to the original requirements, and identify gaps or anomalies in execution. When interactions span multiple cores, the tools determine whether the coupling is both observable and verifiable—whether explicitly through application-level interaction or implicitly due to shared memory or cache behaviour.

This detailed insight into inter-core interaction and shared resource interference enables engineers to identify root causes and shape targeted test scenarios for interference and execution time analysis (Figure 7).

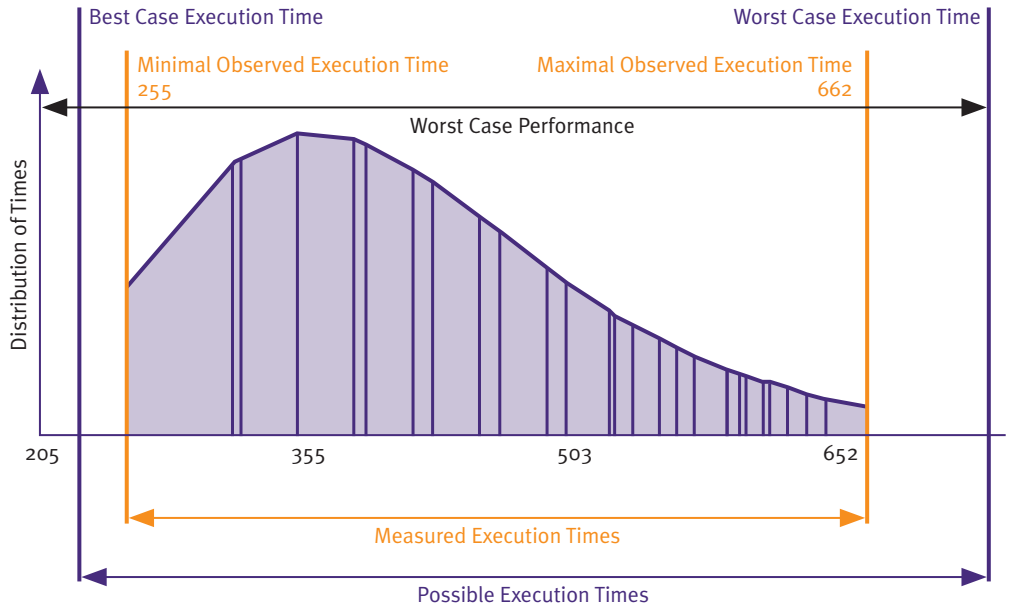


Figure 7: Execution time analysis [11] using the LDRA tool suite

In real-world projects, requirement changes are inevitable. When combined with the fine-tuning necessary for interference analysis, such changes can easily disrupt an already delicate balance. LDRA's support for change impact analysis helps assess how a modification in one component may affect coupled components elsewhere in the system - even across different cores.

In conclusion

Coupling analysis is a critical element of structural coverage verification for high-assurance software, particularly at DO-178C Design Assurance Level A. While DO-178B left the topic loosely defined, DO-178C formalised the requirement to analyse both data coupling and control coupling, demanding evidence that interactions between software components behave as intended during execution.

This paper has shown that an effective verification strategy must adopt a hybrid approach, combining static analysis to identify potential couplings and dynamic, requirements-based testing to confirm them.

The transition to multicore processors adds further complexity, introducing interference effects and non-deterministic behaviour due to shared resources. In this context, the hybrid approach becomes even more vital. LDRA's tool suite supports this need through automated coupling analysis, instrumentation, per-core coverage isolation, and traceability - enabling verification teams to satisfy both DO-178C and A(M)C 20-193 expectations with rigour and confidence.

Works cited

[1] European Organization for Civil Aviation Equipment (EUROCAE), ED-12C - Software Considerations in Airborne Systems and Equipment Certification, Paris, France: EUROCAE, 2011.

[2] RTCA, Inc., RTCA DO-178C - Software Considerations in Airborne Systems and Equipment Certification, RTCA, Inc., 2011.

[3] LDRA, "TBmanager®," [Online]. Available: https://ldra.com/products/tbmanager/. [Accessed 10th July 2025].

[4] LDRA, "LDRA tool suite," LDRA, [Online]. Available: https://ldra.com/products/ldra-tool-suite/. [Accessed 10th July 2025].

[5] LDRA, "Change Impact Analysis (CIA)," LDRA, [Online]. Available: https://ldra.com/capabilities/change-impact-analysis-cia-capability/. [Accessed 10th July 2025].

[6] RTCA, Inc., RTCA DO-178B - Software Considerations in Airborne Systems and Equipment Certification, RTCA, Inc., 1992.

[7] Certification Authorities Software Team (CAST), "CAST-19 Clarification of Structural Coverage Analyses of Data Coupling and Control Coupling," CAST Position Paper, 2004.

[8] I.J.Hennell, J.A.Hanson, M.P.Cieslar, "Verification of control coupling and data coupling analysis in software code". USA Patent US 11,474,927 B1, 18 October 2022.

[9] Federal Aviation Administration (FAA), Advisory Circular 20-193: Use of Multi-Core Processors., Washington, DC.: FAA, 2024.

[10] Certification Authorities Software Team (CAST), Position Paper CAST-32A - Multicore processors, CAST, 2016.

[11] LDRA, "TBwcet®," [Online]. Available: https://ldra.com/products/tbwcet/. [Accessed 10th July 2025].

LDRA UK & Worldwide
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

LDRA Technology Inc.
2540 King Arthur Blvd, Suite 228, Lewisville, Texas 75056
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd.
Unit No B-3, 3rd floor Tower B, Golden Enclave
HAL Airport Road Bengaluru 560017 India
Tel: +91 80 4080 8707
e-mail: india@ldra.com

DCCC white paper v2.1 07/25