

# Getting to grips with CAST-32A & A(M)C 20-193

---

[www.ldra.com](http://www.ldra.com)

© LDRA Ltd. This document is property of LDRA Ltd. Its contents cannot be reproduced, disclosed or utilized without company approval.

## Introduction

The Certification Authorities Software Team [1] (CAST) were a group of civil aviation regulatory officials from Asia, Europe, North and South America, including the FAA [2] and EASA [3]. CAST position papers were for the clarification of issues that related to the formally established standards in that sector, including CAST-32A “Multi-core Processors” [4] - the last position paper to be produced by the team. It will be deprecated when its successor documents AC 20-193 [5] and AMC 20-193 (known collectively as A(M)C 20-193 [6]) have both been published.

Neither CAST-32A nor A(M)C 20 193 is prescriptive. They are intended to guide and support the developer in their quest to meet a standard - typically DO-178C [7]. This paper outlines what made the CAST team turn its attention to MCPs and suggests an approach to applying the guidance.

## The criticality of worst-case execution times

Hard real-time aviation applications need to satisfy stringent timing constraints. In general, upper bounds on the execution times are needed to show the satisfaction of these constraints and (hence) ensure deterministic behaviour. Unfortunately, it is not possible in general to determine exact worst-case execution times for programs.

Suppose that a real-time system runs on a single-core processor and consists of several tasks which realize the required functionality. Figure 1 depicts several relevant properties of a real-time task [8].

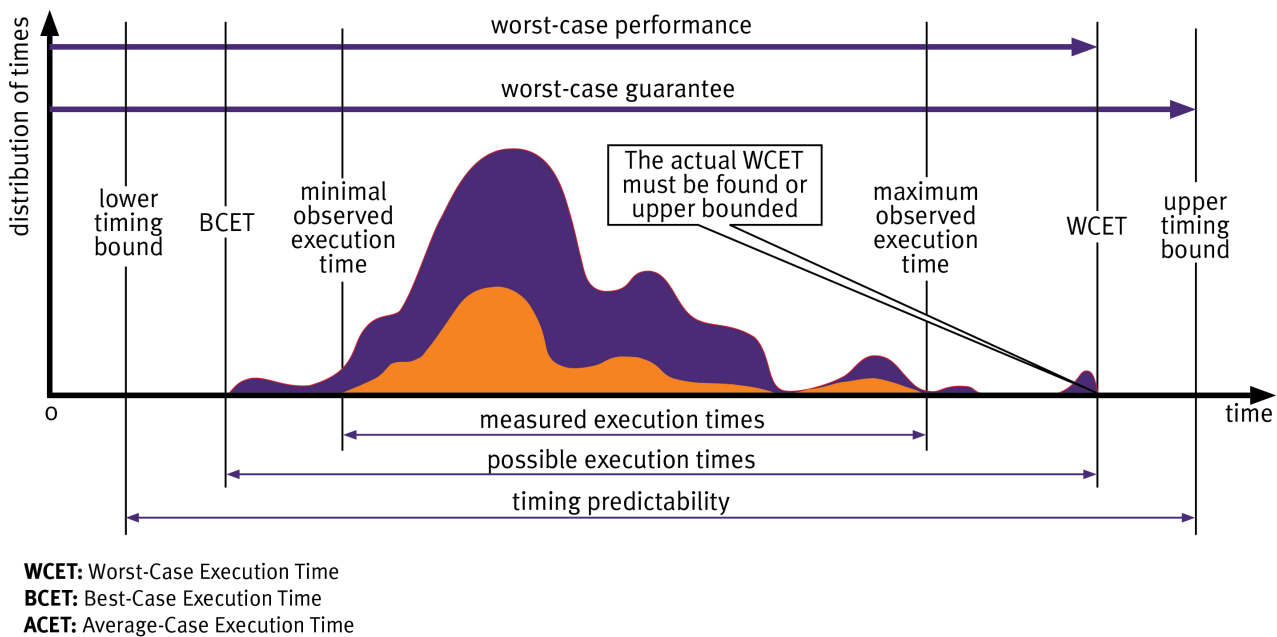


Figure 1: The WCET & BCET are the longest & shortest execution time possible for a program, respectively.

A task typically shows a certain variation of execution times depending on the input data or different behaviour of the environment. The shortest execution time is called the Best-Case Execution Time (BCET), and the longest time is called the Worst-Case Execution Time (WCET). In most cases the state space is too large to exhaustively explore all possible executions and thereby determine the exact WCET and BCET, but there are approximations that allow these values to be estimated to a useful extent [9]

Single-core processors cannot run multiple processes in parallel, and instead use rapid scheduling to make it appear as though they do. Multicore Processors (MCPs) introduce an extra level of complication in that they genuinely do run multiple processes in parallel – and the task of finding a schedule of X tasks on Y processor cores such that all tasks meet their deadlines has no efficient algorithm.

Exacerbating that problem, hardware interference can occur anywhere hardware is shared between processes (Figure 2). For example, often an entire hierarchical memory is shared so that interference is possible in many places.

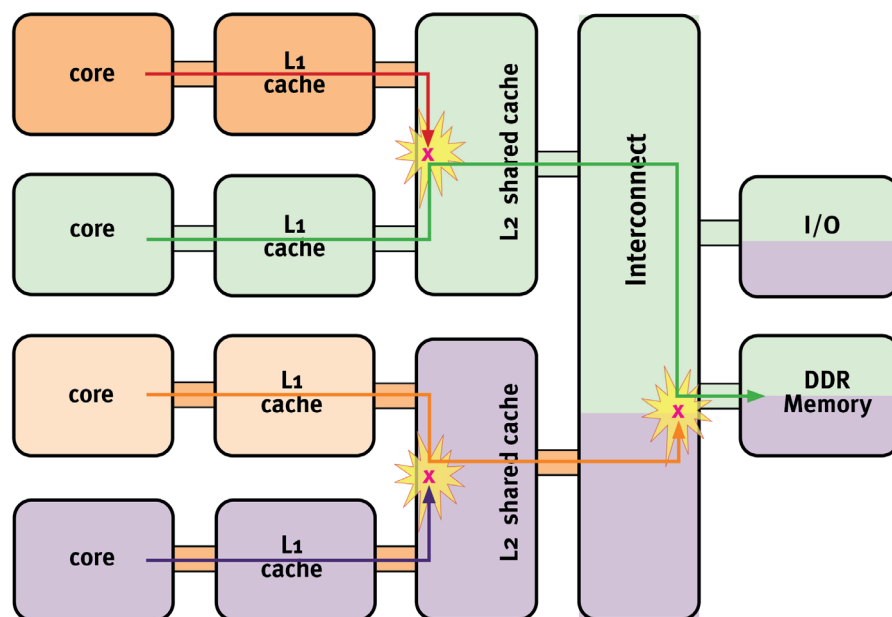


Figure 2: A representation of hardware interference in shared hierarchical memory [14]

These interference channels cause the execution time distribution to spread. Instead of a tight peak, the distribution becomes right skewed with a long tail [10].

It doesn't stop with memory. For example, Hardware Shared Resources (HSRs) on the Xilinx Zynq UltraScale+ [11] with its ARM Cortex-A53 CPU cores include the Snooper control unit, accelerator coherency port, bus interface unit, translation control unit...

## Dealing with the demands of CAST-32A & A(M)C 20-193

Figure 3 is a representation of the relationships between the key civil aviation documents relevant to this paper. In general, the following discussion references the FAA nomenclature ("DO...") but it is equally applicable to the EASA harmonized documents ("ED...") also shown in the illustration.

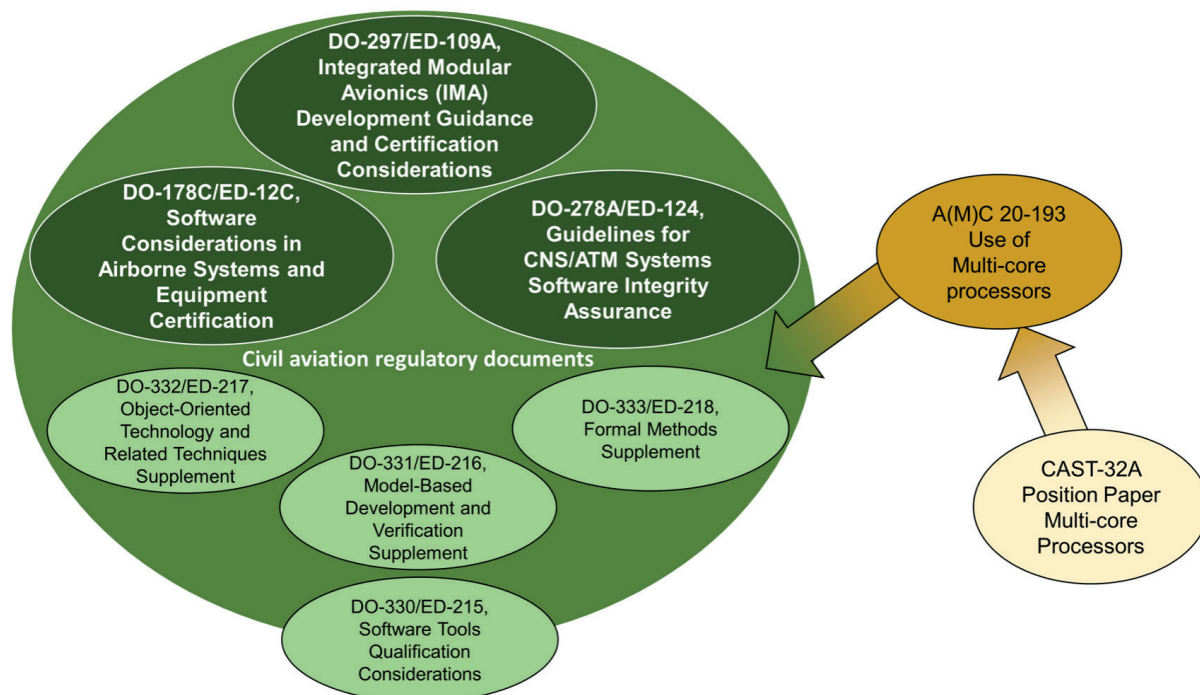


Figure 3: A representation of civil aviation regulatory and advisory documents<sup>1</sup>

<sup>1</sup> CNS/ATM is an abbreviation for Communication, Navigation, Surveillance, and Air Traffic Management. Also note that DO 330 is shown to be overlapping the civil aerospace sector because it is also applicable outside that domain.

As a result of the issues surrounding execution times and MCPs, CAST-32A & A(M)C 20 193 place significant focus on the provision of evidence that allocated resources are adequate to allow for worst case execution times.

Adapting the development process to enable such provision requires a mechanism to analyse execution times, and a controlled, iterative development process to allow for their optimization.

## The analysis of execution times

There are long standing and proven mechanisms available to measure the properties of software code which are independent of their execution on any particular platform. For example, Halstead's metrics [12] reflect the implementation or expression of algorithms in different languages to evaluate the software module size, software complexity, and the data flow information – and these can be calculated precisely from the static analysis of the source code (Figure 4). Such an approach can identify which sections of code are the most demanding of processing time but cannot provide absolute values for maximum time elapsed.

| Halsteads (Cashregister.c) |                 |                |                  |                 |            |        |        |
|----------------------------|-----------------|----------------|------------------|-----------------|------------|--------|--------|
| File                       | Total Operators | Total Operands | Unique Operators | Unique Operands | Vocabulary | Length | Volume |
| Total for Cashregister.c   | 149             | 230            | 16               | 56              | 72         | 379    | 2338   |

Figure 4: Halstead's Metrics calculated using the LDRA tool suite.

The only way to ascertain how that information translates into time elapsed for a particular function or call tree is by measuring it in the environment in which it is intended to run. That can be measured dynamically by deploying the TBrwn component of the LDRA tool suite, complete with the supplementary TBwcet module.

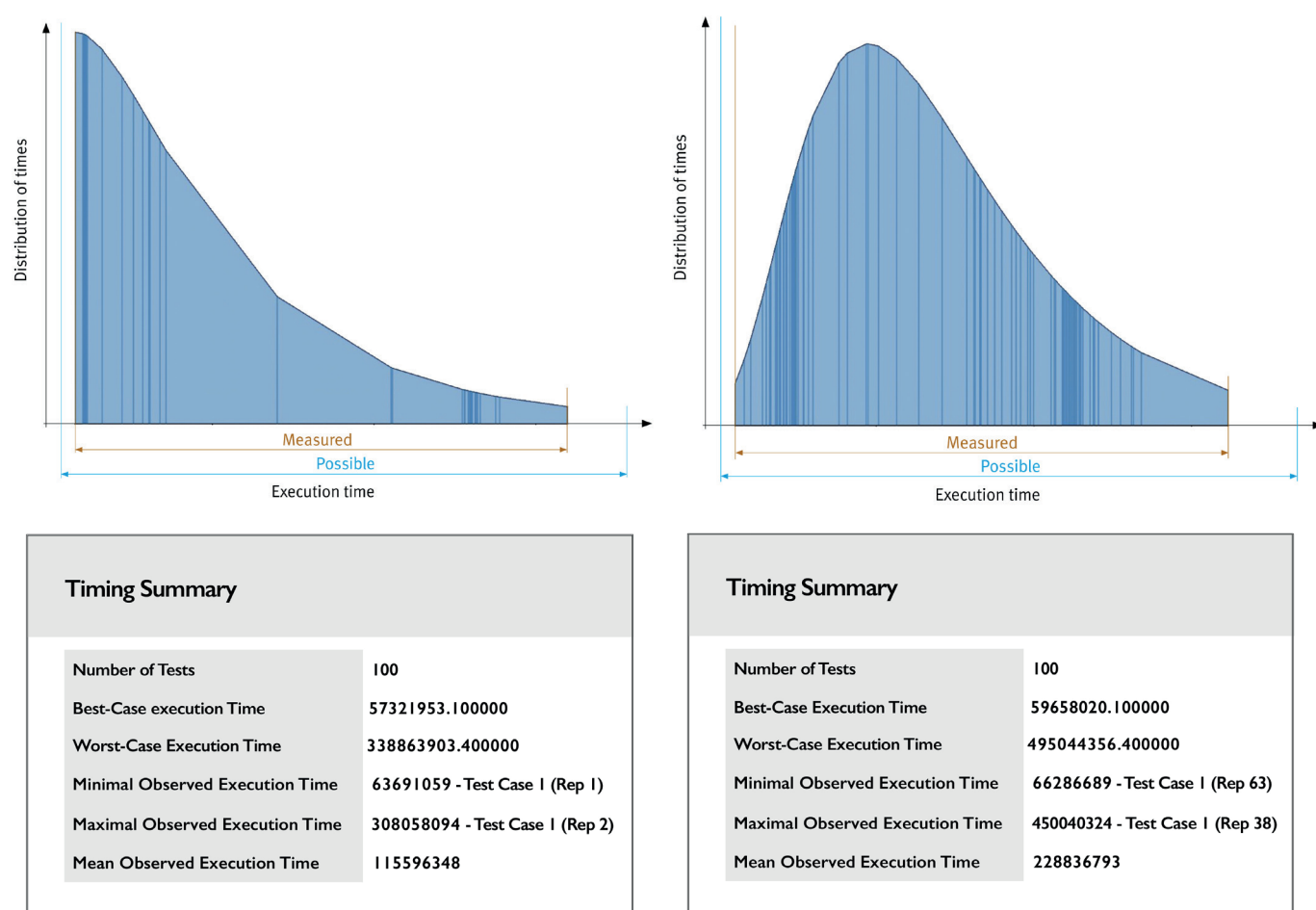


Figure 5: Histograms showing the spread of execution times as displayed by the LDRA tool suite.

This unit test tool automates the measurements, running the specified tests repeatedly and providing a graphical representation of the variation in execution times (Figure 5). It allows the user to select their preferred hardware stress utility (stress-ng [13], for example) to load the different shared resources on the target processor to varying degrees. The example shows that interference has a detrimental effect on execution time.

It is also important to analyse the performance of the application of a whole. This is achieved by leveraging the LDRA tool suite's source substitution mechanism to integrate those unit test driver(s) into the application.

TBwcet provides the flexibility to perform system tests or to drill down into performance at a unit test level. It therefore presents the ideal toolkit for performing interference research, and ultimately for demonstrating that WCET verification requirements are met.

## Embracing interference research

The empirical nature of verification and validation in the MCP makes it imperative that project managers are equipped to adapt readily to changing requirements and configurations.

A slow and steady approach to measuring the effects of interference and mitigating for them is recommended [14] (Figure 6).

Under these circumstances, an automated mechanism is helpful to keep track of what needs revisiting for renewed verification and validation, and hence to keep the project on schedule and within budget (Figure 7).

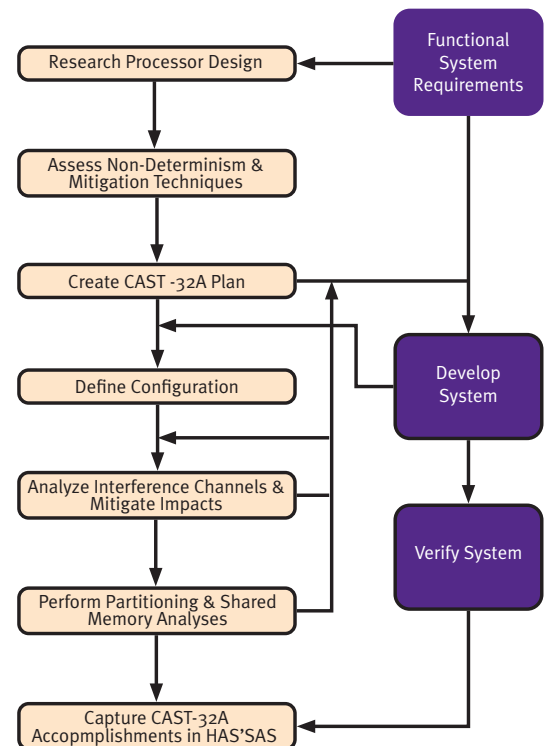


Figure 6: Recommended multicore certification approach

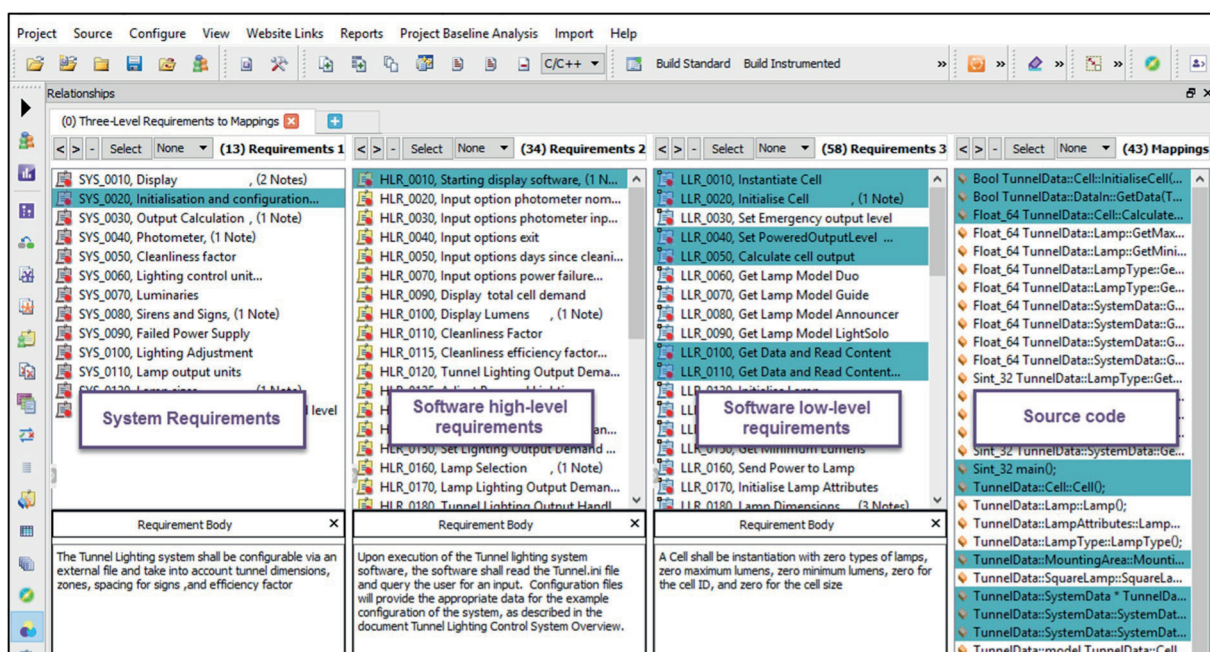


Figure 7: Tracing requirements and objectives with the TBmanager component of the LDRA tool suite



## CAST-32A & A(M)C 20-193 Objectives

There are ten objectives in CAST-32A & A(M)C 20-193 which supplement the DO-178C processes for multicore processors throughout its V-model lifecycle. They span from planning through to verification, while focusing on shared resources and interference channels. Neither CAST 32A nor A(M)C 20-193 prescribe methods for achieving those objectives, leaving that to the system integrator and their suppliers.

The objectives laid out by the CAST-32A & A(M)C 20-193 objectives detail the issues surrounding the adoption of these techniques. Key considerations include the following.

- **MCP\_Planning\_1** discusses the specific items to be considered in planning a project around multicore execution with a particular focus on with respect to resource management. Automated tracking of the concurrent fulfilment of CAST-32A & A(M)C 20 193 objectives, those of a standard (typically DO-178C), and the project specific requirements can help to address a project management headache.

### • MCP\_Resource\_Usage\_1

MCPs are extremely complex and have many settings that can affect execution and potentially impact system safety. The empirical nature of MCP resource configuration and scheduling optimization makes it highly likely that the configuration will change during development. Keeping track of changing statuses when it does so is important (Figure 8).

| Objectives                   | Artifacts | Assets | Last Verifi... | Objective ...  | Objective St... | Objective S... | Placeholders |
|------------------------------|-----------|--------|----------------|----------------|-----------------|----------------|--------------|
| MCP_Accomplishment_Summary_1 | 0         | 1      | Thu Sep 30 ... | Accomplish...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Error_Handling_1         | 0         | 1      | Thu Sep 30 ... | Failures       | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1               | 0         | 0      | Thu Sep 30 ... | Software Pl... | CAST-32A        | Partial        | 0.00%        |
| MCP_Planning_1:1             | 0         | 1      | Thu Sep 30 ... | MCP Proces...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1:2             | 0         | 1      | Thu Sep 30 ... | MCP Active...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1:3             | 0         | 1      | Thu Sep 30 ... | MCP softwa...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1:4             | 0         | 1      | Thu Sep 30 ... | MCP Dyna...    | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1:5             | 0         | 1      | Thu Sep 30 ... | IMA platform   | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1:6             | 0         | 1      | Thu Sep 30 ... | Robust Res...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_1:7             | 0         | 1      | Thu Sep 30 ... | Methods an...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_2               | 0         | 0      | Thu Sep 30 ... | Software / ... | CAST-32A        | Partial        | 0.00%        |
| MCP_Planning_2:1             | 0         | 1      | Thu Sep 30 ... | MCP shared...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Planning_2:2             | 0         | 1      | Thu Sep 30 ... | Hardware ...   | CAST-32A        | Partial        | 100.00%      |
| MCP_Resource_Usage_1         | 0         | 1      | Thu Sep 30 ... | MCP config...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Resource_Usage_2         | 0         | 1      | Thu Sep 30 ... | Means of M...  | CAST-32A        | Partial        | 100.00%      |
| MCP_Resource_Usage_3         | 0         | 1      | Thu Sep 30 ... | Interferenc... | CAST-32A        | Partial        | 100.00%      |
| MCP_Resource_Usage_4         | 0         | 0      | None           | Available R... | CAST-32A        | Unfulfilled    | 0.00%        |
| MCP_Software_1               | 0         | 1      | Thu Sep 30 ... | Applicable ... | CAST-32A        | Partial        | 100.00%      |
| MCP_Software_2               | 0         | 1      | None           | Data and C...  | CAST-32A        | Partial        | 16.67%       |

Figure 8: A graphical representation of progress towards meeting CAST 32A & A(M)C 20-193 objectives in the TBmanager component of the LDRA tool suite

- **MCP\_Resource\_Usage\_3** is concerned with the identification of interferences channels likely to cause an issue for the application under development, and to identify a means of mitigation. The identified channels provide focus for subsequent interference research.
- **MCP\_Resource\_Usage\_4** and **MCP\_Software\_1** are both concerned with ensuring that that allocated tasks can be achieved in the time allotted to them, and that adequate resources are allocated for that to be achieved.

CAST-32A & A(M)C 20-193 go on to emphasize the significance of Robust Resource and Time Partitioning and suggest that their provision makes it acceptable “*may verify applications separately on the MCP and determine their WCETs separately.*”

As CAST-32A & A(M)C 20-193 imply, the use of robust partitioning is likely to make interference mitigation easier. However, DO-178C places the onus on the developer to demonstrate that interference mitigation is effective – bearing in mind that DO 178C is prescriptive, whereas CAST-32A & A(M)C 20-193 are not.

- **MCP\_Software\_2** references control and data coupling analysis, which look to show that the software modules affect one another only in the ways intended by the software design. They are already addressed in DO-178C (for example) but carry additional significance here because flawed coupling is likely to have an impact on timing, and because data that is handled at different times by different threads on different cores are worthy of special attention.

LDRA's static analysis tools support control coupling and data coupling analysis relating to data and control flow between software components and applications (Figure 9).

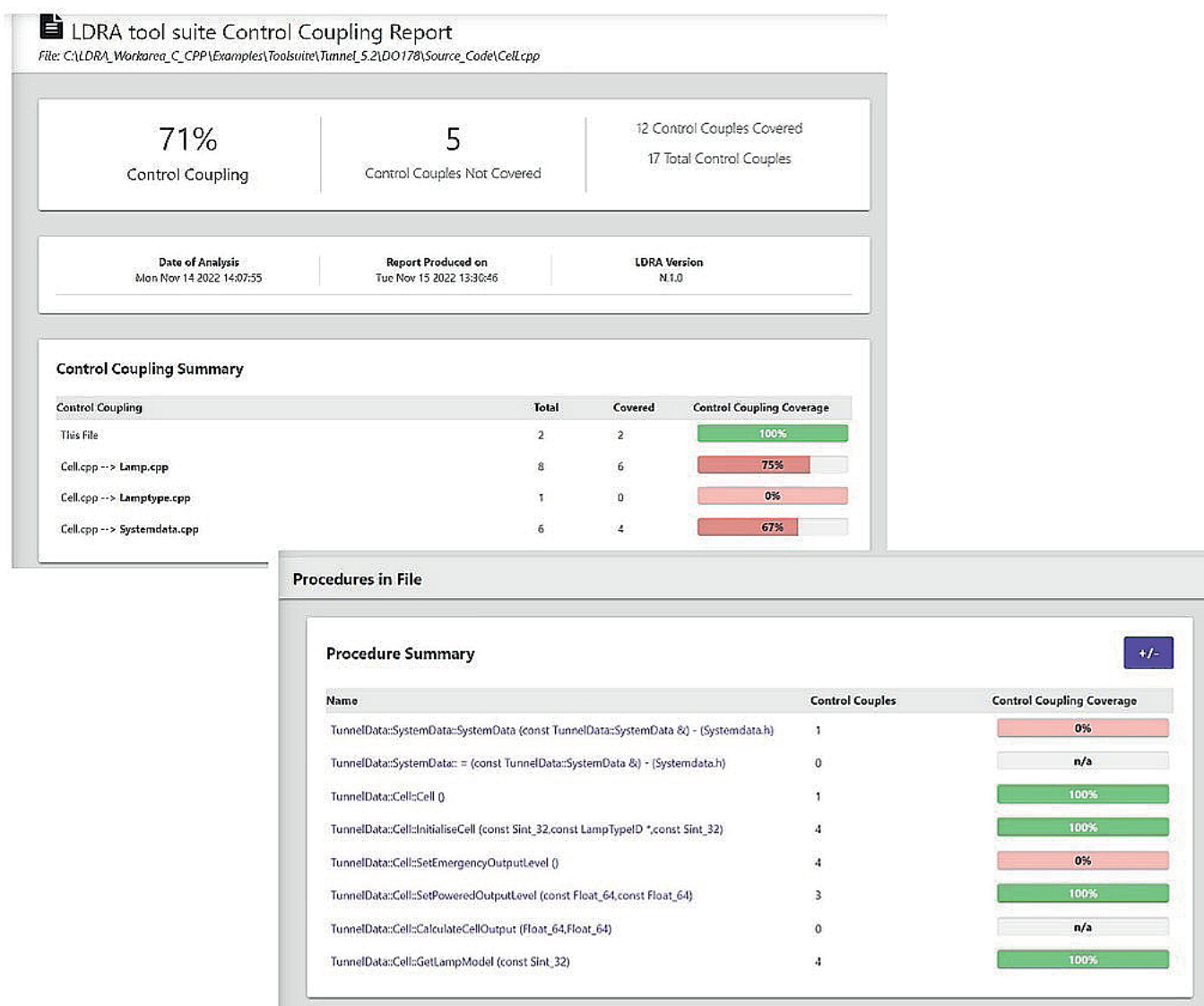


Figure 9: Data coupling analysis and control coupling analysis [28] with the LDRA tool suite

## Conclusions

Concerns over the suitability of Multicore Processors (MCPs) for safety critical applications are nothing new. There are challenges with MCPs in these environments, especially with respect to interference. The provision of robust partitioning between domains is likely to be helpful - although DO-178C dictates that the responsibility for demonstrating adequate resourcing remains firmly with the application developer.

The use of an iterative development model helps to overcome this challenge. Repeatedly testing and adapting the system as it is developed ensures that the potential for such interference is minimized, and hence that WCET constraints are met. An integrated, automated requirements traceability system such as the TBmanager component of the LDRA tool suite is invaluable for keeping track of that process.

In the case of civil aviation projects, adherence to CAST-32A & A(M)C 20-193 will be in tandem with compliance to other standards too, typically DO-178C. In general, the tools required to show adherence to that standard are no different, but some characteristics are critical. For example, care needs to be taken over the efficiency of instrumentation to avoid it being detrimental to the operation of the system as a whole.

In summary, there are real challenges in the application of MCPs in hard real time safety critical civil aviation applications, but none that are insurmountable given the right tools, used diligently.

## Works cited

- [1] Federal Aviation Administration (FAA), “Certification Authorities Software Team (CAST),” 18 August 2020. [Online]. Available: [faa.gov/aircraft/air\\_cert/design\\_approvals/air\\_software/cast/](https://faa.gov/aircraft/air_cert/design_approvals/air_software/cast/). [Accessed 11 November 2021].
- [2] Federal Aviation Administration (FAA), “Federal Aviation Administration (FAA),” [Online]. Available: <https://www.faa.gov/>. [Accessed 3 November 2021].
- [3] European Union Aviation Safety Agency (EASA), “European Union Aviation Safety Agency (EASA),” 2021. [Online]. Available: <https://www.easa.europa.eu/>. [Accessed 3 November 2021].
- [4] Certification Authorities Software Team (CAST), Position Paper CAST-32A - Multicore processors, CAST, 2016.
- [5] EASA, “AMC 20-193 Use of multi-core processors,” 2022. [Online]. Available: [https://www.easa.europa.eu/sites/default/files/dfu/annex\\_i\\_to\\_ed\\_decision\\_2022-001-r\\_amc\\_20-193\\_use\\_of\\_multi-core\\_processors\\_mcps.pdf](https://www.easa.europa.eu/sites/default/files/dfu/annex_i_to_ed_decision_2022-001-r_amc_20-193_use_of_multi-core_processors_mcps.pdf). [Accessed 18th January 2023].
- [6] F. Wolfe, “EASA and FAA to Issue Further Guidance on Multicore Certification This Year,” 28 February 2020. [Online]. Available: <https://www.aviationtoday.com/2020/02/28/easa-and-faa-to-issue-further-guidance-on-multicore-certification-this-year/>. [Accessed 4 November 2021].
- [7] RTCC, DO-178C “Software Considerations in Airborne Systems and Equipment Certification”, RTCA, 2011.
- [8] R. W. e. al., “The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools,” April 2008. [Online]. Available: [http://www.es.mdh.se/pdf\\_publications/1258.pdf](http://www.es.mdh.se/pdf_publications/1258.pdf). [Accessed 1 November 2021].
- [9] C. L. LIU and JAMES W. LAYLAND, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” Journal of the Association for Computing Machinery, vol. 20, no. 1, pp. 46-61, January 1973.
- [10] T. Lovelace, “Challenges building safe multicore systems,” 15 June 2020. [Online]. Available: <https://www.lynx.com/embedded-systems-learning-center/challenges-building-safe-multicore-mcp-software-systems>. [Accessed 2 November 2021].
- [11] AMD Xilinx, “Zynq™ UltraScale+™ MPSoC,” 2023. [Online]. Available: <https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html>. [Accessed 23rd February 2023].
- [12] Halstead, Maurice H., Elements of Software Science., Amsterdam: Elsevier North-Holland, 1977.
- [13] C. King, “Ubuntu wiki: stress-ng,” 7th October 2020. [Online]. Available: <https://wiki.ubuntu.com/Kernel/Reference/stress-ng>. [Accessed 3rd February 2023].
- [14] Paul J. Parkinson (Wind River) and Harold G. Tiedeman (Rockwell Collins), “Plan With Confidence: Route to a Successful DO-178C Multi-Core Certification,” 28 September 2018. [Online]. Available: <https://www.slideshare.net/ICTperspectives/plan-with-confidence-route-to-a-successful-do178c-multicore-certification>. [Accessed 5 November 2021].
- [15] LDRA, “Control coupling analysis and data coupling analysis for embedded software,” [Online]. Available: <https://ldra.com/capabilities/data-couplingcontrol-coupling/>. [Accessed 3rd February 2023].
- [16] LDRA, “TBrun®,” LDRA, [Online]. Available: <https://ldra.com/products/tbrun/>. [Accessed 23 March 2022].
- [17] LDRA, “TBwcet®,” [Online]. Available: <https://ldra.com/products/tbwcet/>. [Accessed 3rd February 2023].



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

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