Rapita Systems

Safety through quality

Guillem Bernat, Antoine Colin

AdaSpain 17th of May 2024

Motivation

- Who are we, What do we do.
- Our tools are written in Ada (parts of them).
  - Making RVS happen
  - Our view of Ada in the industry
  - Software verification of Ada
- Multicore certification

Company Overview

- We deliver verification solutions for critical embedded software **worldwide**
- Founded 2004, 80+ staff
- 3 office locations
  - Rapita Systems Ltd., **York**, UK
    - Development of embedded verification tools
    - Provider of embedded Engineering Services (including multicore)
  - Rapita Systems, Inc., **Michigan**, USA
    - Provider of embedded Engineering Services (including multicore and US-eyes only)
  - Rapita Systems S.L., **Barcelona**, Spain
    - Development of multicore verification microbenchmarks
    - Provider of multicore Engineering Services

Offices

Rapita Systems Ltd.

- Based in York, UK
- Development of embedded verification tools
- Global provider of embedded Engineering Services (including multicore services)

Rapita Systems, Inc.

- Based in Novi, Michigan
- Provider of services for **US eyes only** and defense-related verification projects, including multicore timing solutions
- Staffed by US persons, including staff trained to handle export-controlled materials

Rapita Systems S.L.

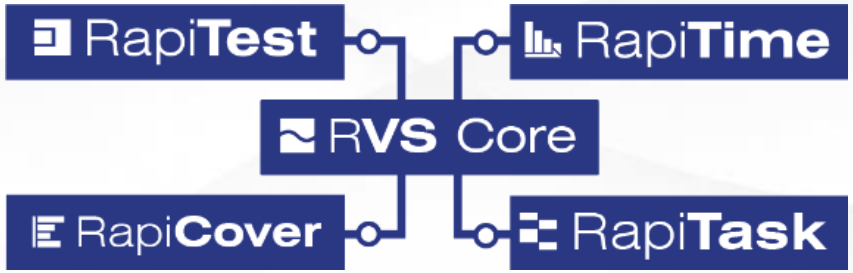
- Based in Barcelona, Spain
- Development of multicore verification microbenchmarks
- Global provider of verification services including multicore timing solutions, with an emphasis on hardware aspects

Products and Services

Rapita **Verification Suite**

On-target software verification for critical embedded systems

RVS



Multicore Avionics Certification for High-integrity DO-178C projects

MACH178

V&V Services

RapiDaemons RTBx Integration Services Customization Support Training Qualification

Addressing DO-178C with the Rapita **Verification Suite** (RVS)

DO-178C guidance

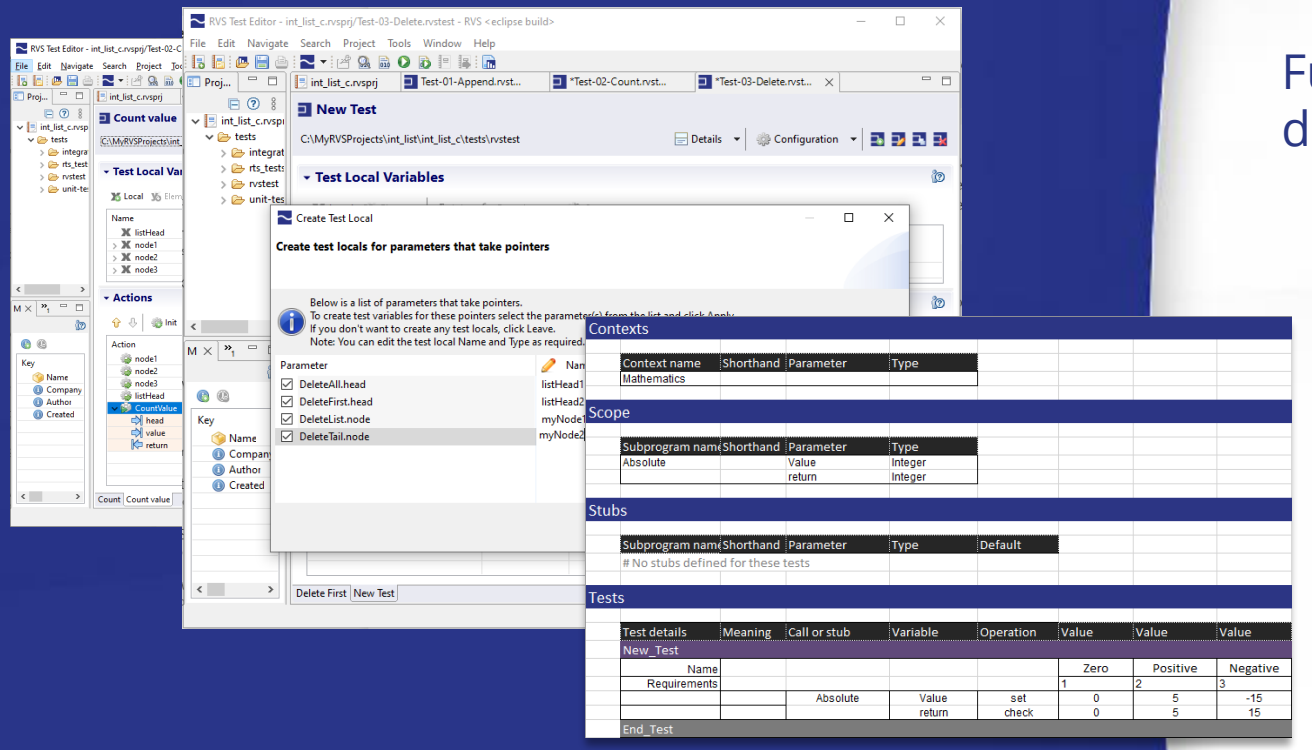
- 6.4.3 Requirements-Based Testing Methods
- 6.4.4.2 Structural Coverage Analysis
- 6.4.4.2.b Verify additional code introduced by compiler
- 6.3.4 Reviews and Analyses of the Source Code

RVS

- Rapi**Test**: Functional testing for critical software
- Rapi**Cover,** Rapi**Cover**^{Zero}: On-target structural code coverage analysis
- V&V supported by Rapi**Cover**^{Zero}
- Rapi**Time**, Rapi**Task**, Rapi**Time**^{Zero}, Rapi**Task**^{Zero}: Execution time analysis

RapiTest

RapiTest is an advanced functional testing (unit, integration, system) tool designed to work with embedded targets and complex build processes



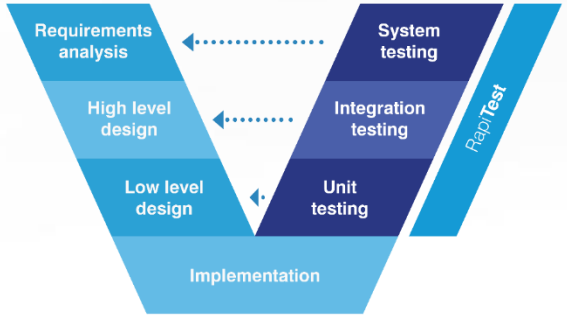
DO-178C & Functional testing

DO-178C specifies that software testing should be “requirements-based”

Functional testing checks that your software functions as expected according to its requirements

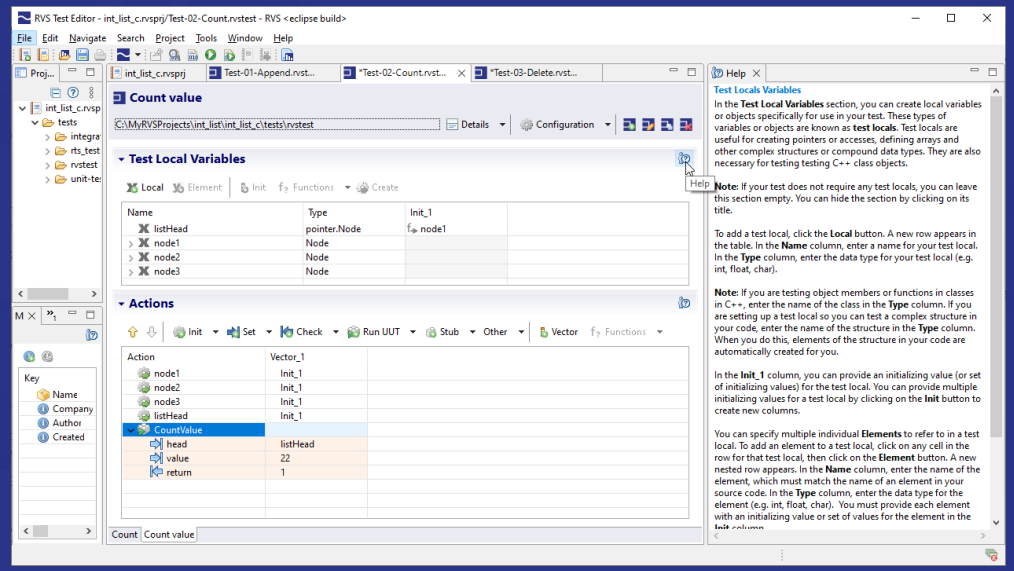
Functional testing is typically performed at different levels

- **System and integration level** testing that corresponds to high-level requirements
- **Unit testing** (for example at the function level) that corresponds to lower-level requirements



Why choose RapiTest?

- Full automation of testing process
- Flexible test formats
- CI & RQM tool compatible
- Qualifiable for DO-178C
- Supports Ada, C/C++

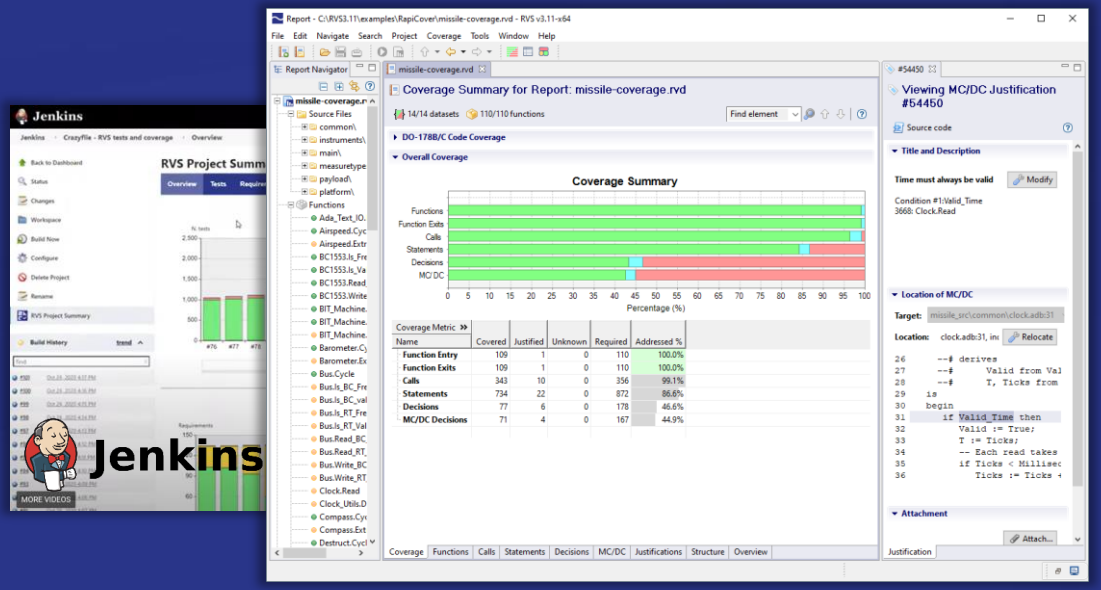


ReqIF Requirements Interchange Format
Ansys SCADE SUITE
LYNX | Luminosity
CODEO
OpenArbor
visure
GCC
Bamboo
VxWorks
Jama software
Jenkins
GNU Make

RapiCover

An advanced structural coverage analysis tool designed specifically for optimal embedded performance

- Supports Ada, C/C++
- Supports any target hardware
- Qualifiable for DO-178C



Code Coverage for DO-178C

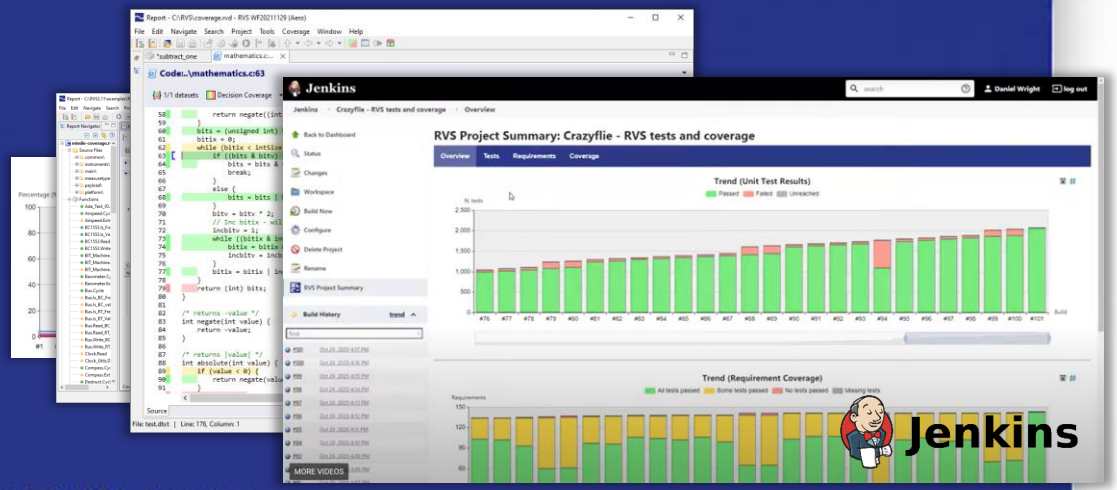
- The measurement of structural code coverage is a key activity for DO-178C compliance
- The level of code coverage required by DO-178C is dependent on the DAL level of your project

DAL A	DAL B	DAL C
MC/DC	Decision coverage	Statement coverage
Decision coverage	Statement coverage	
Statement coverage		

- DO-178C (via DO-330) requires “qualification” of the tool you use to collect code coverage metrics

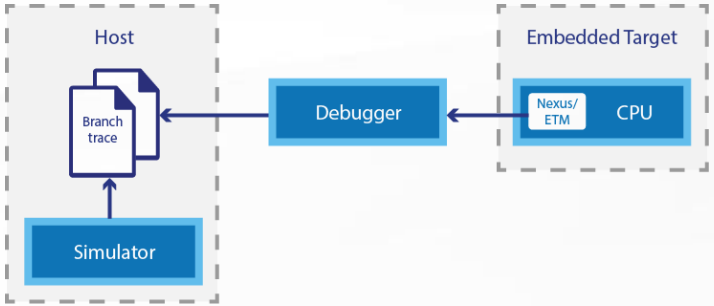
Why choose RapiCover?

- Collect coverage, including MC/DC, on-host & target
- Reduce test builds needed for analysis on constrained targets
- Justify untestable code
- Save time with efficient “merge and mark” verification workflow
- Connect with your continuous integration tool for efficiency



RapiCover Zero

- **Object code** coverage analysis
  - Supplement your source code results with object code results
- Collect coverage for libraries without source code
- “Test what you fly”
- View mapping between source code and object code



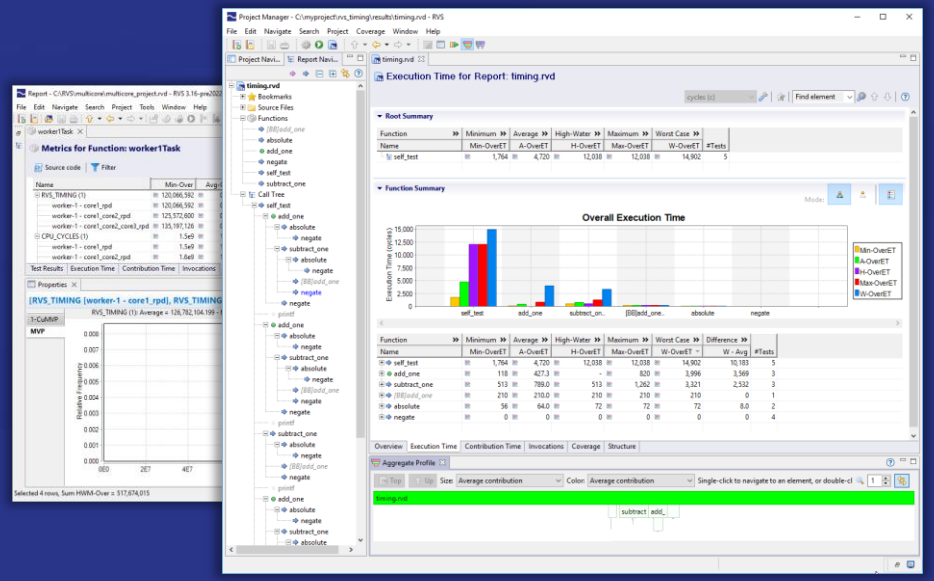
Collins Aerospace

*“Compared to previous tools we’ve used in the past, Rapi**Cover**’s performance has been much more reliable and robust”*

RapiTime

An advanced timing analysis tool designed specifically for optimal embedded performance

- Supports Ada, C/C++
- Supports any target hardware
- Qualifiable for DO-178C

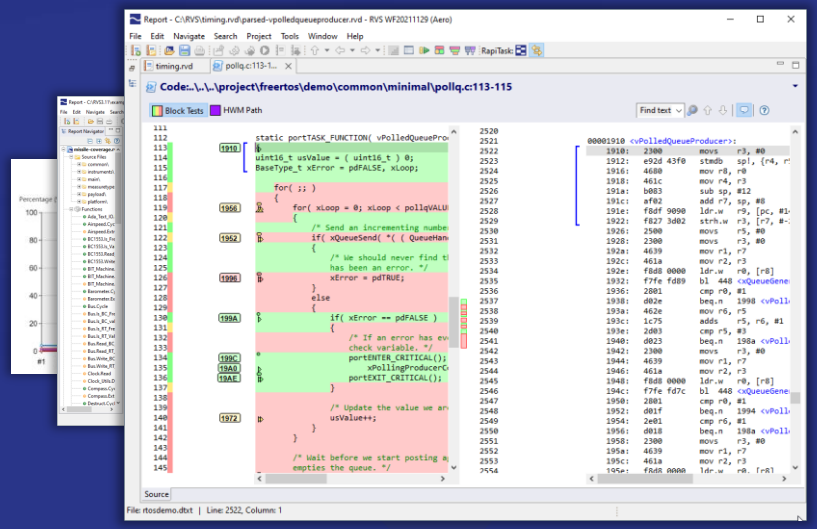


Timing analysis for DO-178C

- The measurement of software timing behavior is a key activity for DO-178C compliance
- DO-178C, 11.20i: explicitly mentions "timing margins including worst-case execution time“
- If software delivers its output "too late“ then it can fail to comply with its requirements
- Rapi**Time** uses a hybrid approach:
  - On-target collection of metrics
  - Offline analysis with information obtained during testing to build up a model of the overall code structure & paths
  - Combining these approaches to compute worst-case execution times in a way that captures execution time variation on individual paths due to hardware effects

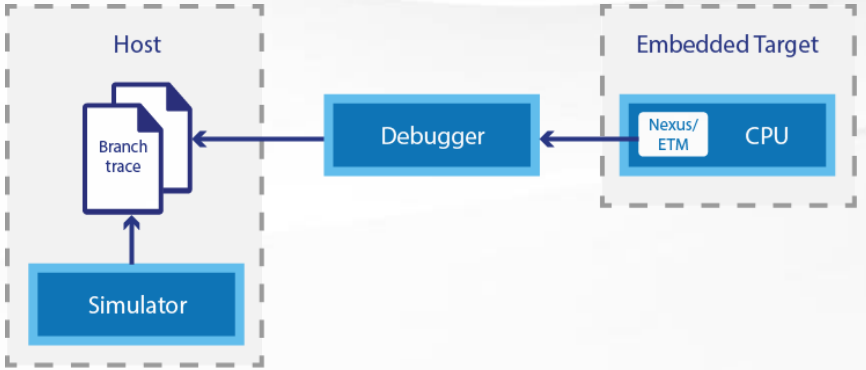
Why choose RapiTime?

- Collect and calculate timing metrics, including WCET, on-host & target
- Identify code to optimize for worst-case behavior
- Debug rare timing events
- Produce evidence for DO-178 and ISO 26262 certification
- Futureproof your timing analysis workflow for the use of multicore processors
- Recognized by the FAA as "*an example of a mature tool*" for dynamic timing analysis



RapiTime Zero

- **Object code** timing analysis including WCET
- Collect timing metrics without source code
- "Test what you fly"



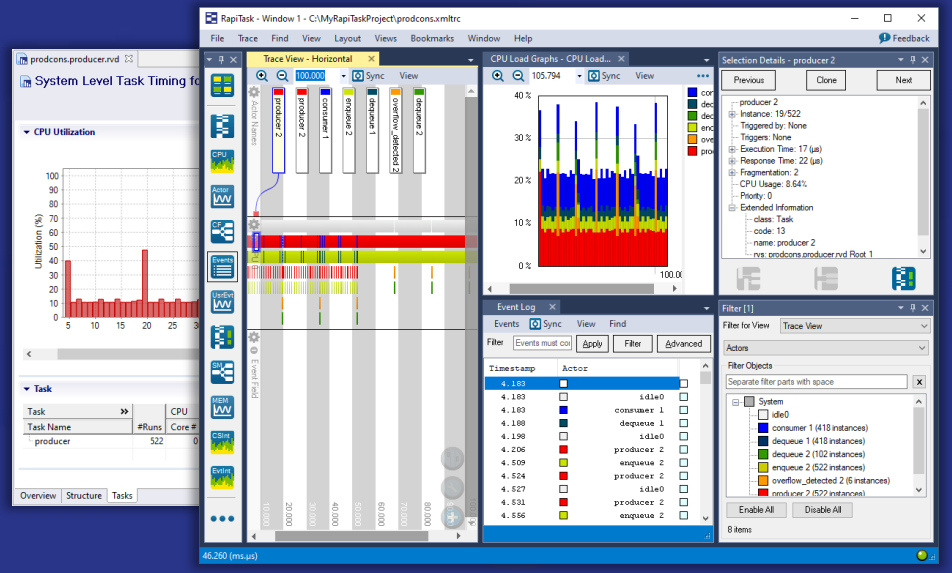
LEONARDO

*"With Rapi**Time**, we discovered the possibility to reduce by 10% the time spent by a Computer Software Configuration Item"*

RapiTask

A tool to provide an overview of scheduling behavior

- Supports Ada, C/C++
- Supports any target hardware
- Optimize platform utilization



Why choose RapiTask?

- Examine Ada, C or C++ applications via scheduling visualization
- Locate rare timing events that need attention
- Identify bottlenecks in your application by analyzing capacity issues
- Compare scheduling algorithms from different RTOSs
- Reduce development costs
- Futureproof your timing analysis workflow for the use of multicore processors

RapiTask *Zero*

- Object code scheduling analysis
- Collect scheduling behavior metrics for libraries without source code

Qualifying software tools for critical contexts

When working to standards/guidelines such as DO-178B/C or ISO 26262, verification tools must often be *qualified*

Objective			Applicability				Output	
ID	Description	Ref.	A	B	C	D	Description	Ref.
A-7, 5	Test coverage of software structure (modified condition/decision) is achieved.	6.4.4.2	•				Software Verification Results	11.14
A-7, 6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.2a 6.4.4.2b	•	•			Software Verification Results	11.14
A-7, 7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.2a 6.4.4.2b	•	•	○		Software Verification Results	11.14

Target integration service (TIS)

R**VS** plugin qualification kit (QKIT)

Qualification Kits provide DO-330 COTS Tool Developer evidence

Qualified target integration service (QTIS)

Provide evidence that the integration of RVS into your target development environment is robust

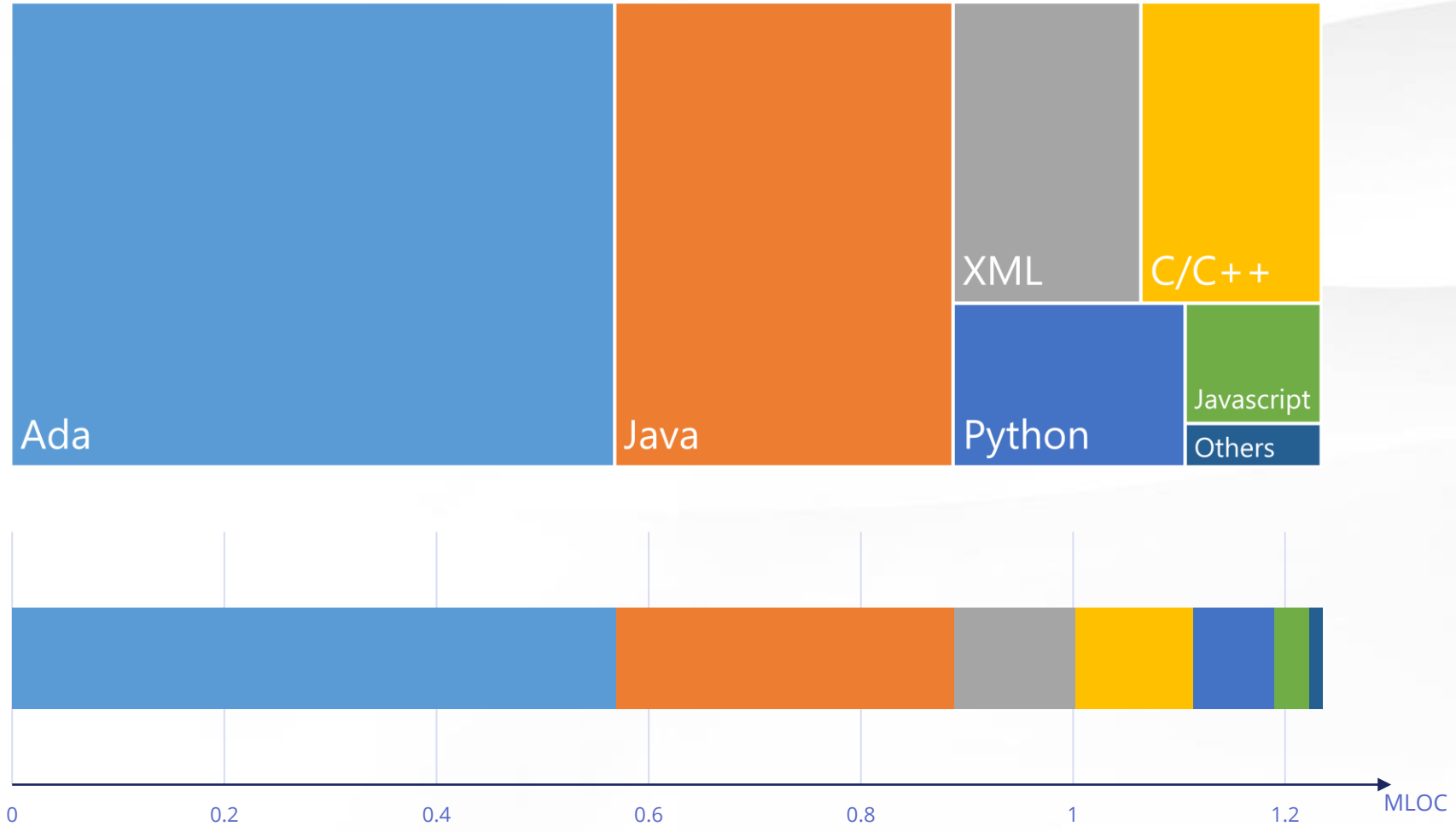
Collins Aerospace

"The quality and ease-of-use of Rapita's Qualification products and services is second to none"

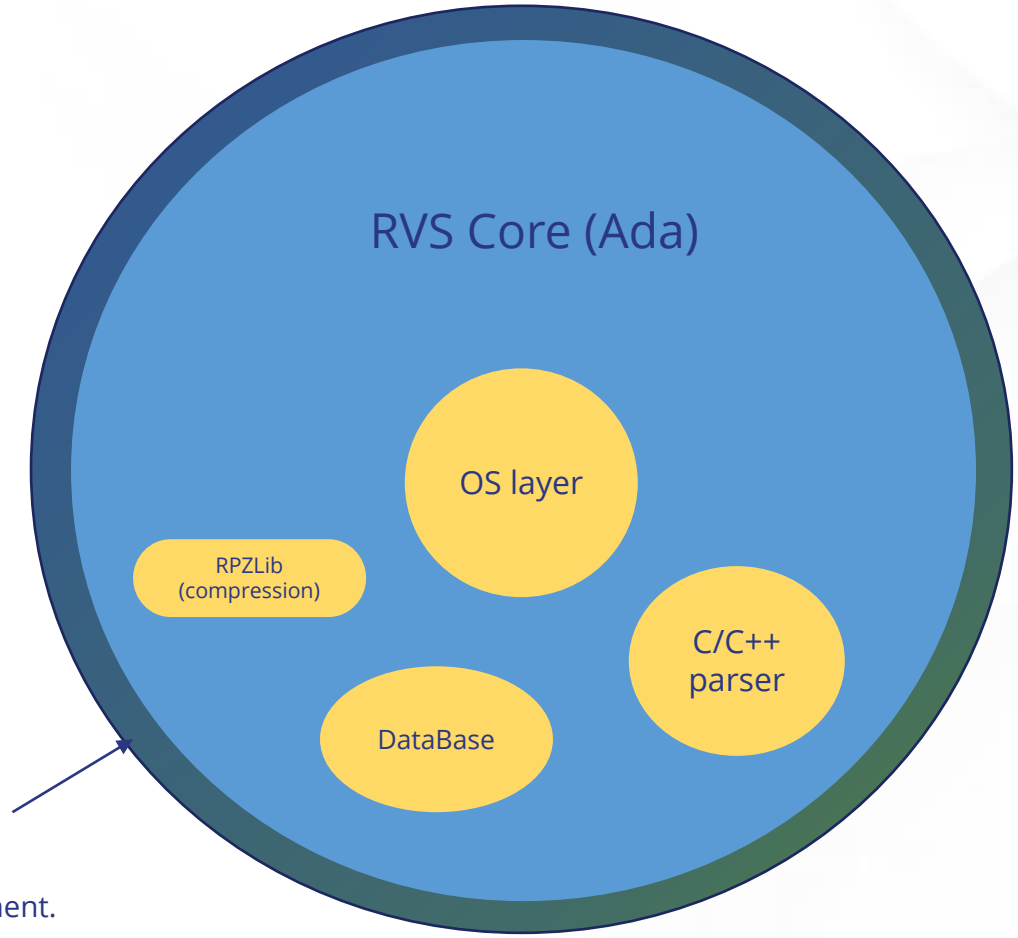
RVS is written (some of it) in Ada.

- Antoine Colin

What is RVS made of ?



Ada in the RVS software architecture



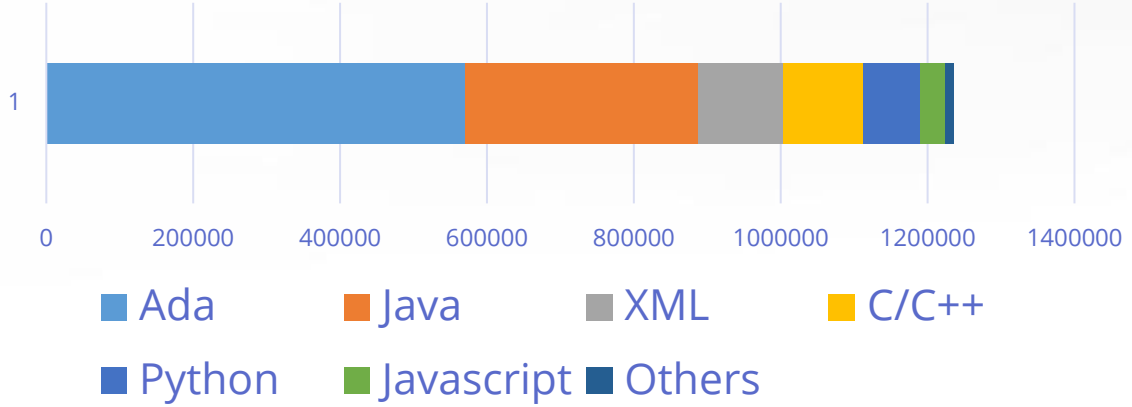
Application Layer.

Interfacing with
customer environment.

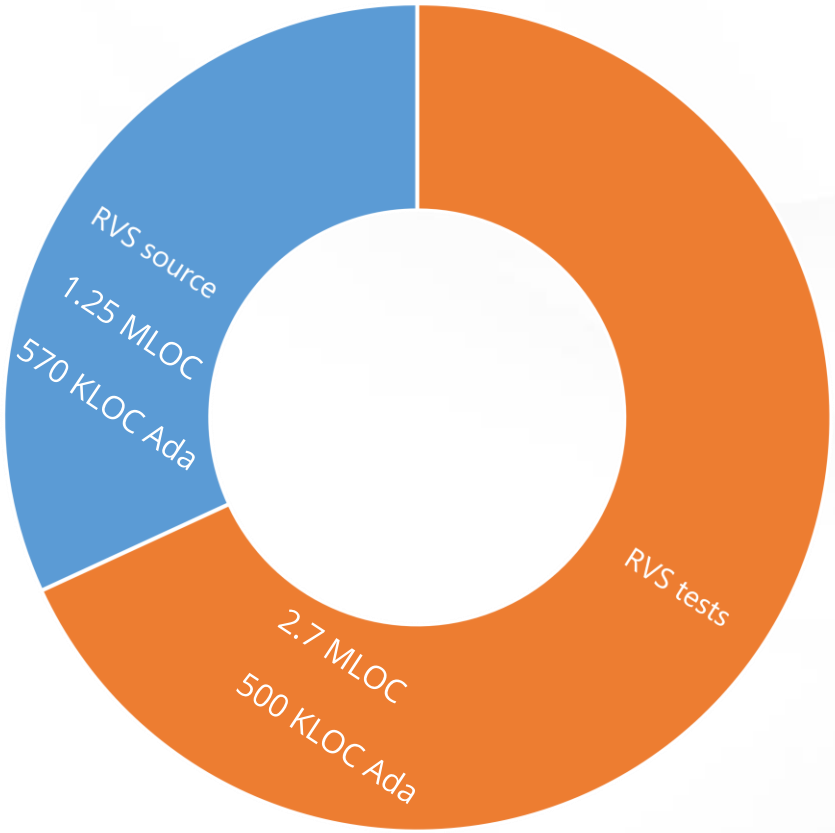
eclipse

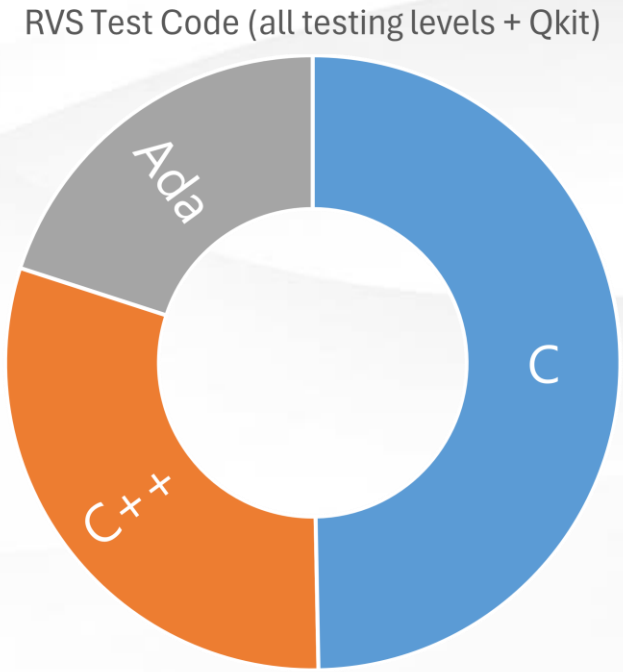
RVS GUI

Documentation



Any other Ada code in RVS?





- Total: ~4 MLOC (1.7 MLOC Ada)

What does it take to make RVS?

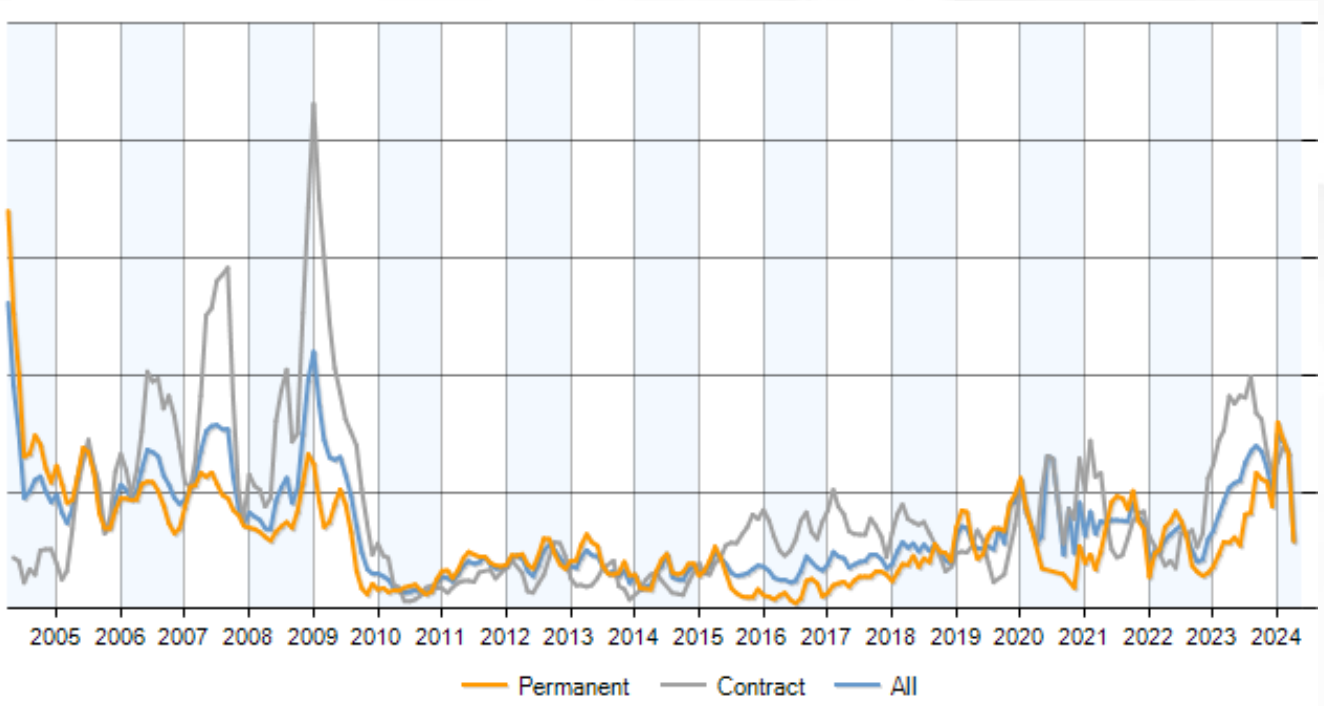
- We based our SW engineering processes on GitLab
  - Planning and project management
    - Issue Tracking, Project Planning, Project management
  - Version Control
    - Branching, merging, back-porting, …
  - Implementation and verification
  - Reviews
    - Code peer review, review checklist, review logs
  - Continuous integration
  - Packaging & Release
  - Collaboration and Communication
    - Architecture and Code documentation, Issue discussions

Ada goes here

Use of Ada in the Industry

Use of Ada in the Industry

- Trend: Job Adverts mentioning Ada (UK)



Perceived recruitment challenge

"We can’t use Ada on our project because recruiting Software Engineers who know how to use it or can learn it is very difficult”

Recruiting Software Engineers

- Our experience, some examples:
  - Will
    - “I had experience of C and Java but had never heard of Ada before I joined Rapita”
    - “I just picked up a copy of the Ada 95 book and got stuck in”
  - Dave
    - “I’d heard good things about it but some people said it was horrible”
    - “Ada offers a safety net in the way other languages don’t”
  - Chris
    - “I had no knowledge of Ada before joining Rapita, and as a result was able to start learning the language without any prejudices”
    - “the Ada learning curve is be comparable to other languages, and the intuitive syntax makes it easy to decipher example code”

Ada SW verification, easier than for C/C++ ?

Timing Analysis of Ada

- Does Ada make timing analysis easier?
  - Easier (than C++) to understand/review the code for potential timing issues
  - But…

RapiTime

(demo)

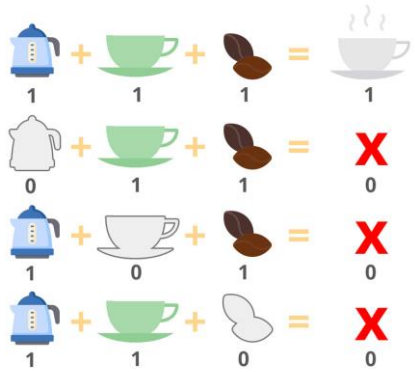
Structural Coverage Analysis of Ada

RapiCover

- Quick introduction to MC/DC coverage

The MC/DC Criterion
"Each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by varying just that condition while holding fixed all other possible conditions".

```
if  kettle and then cup and then coffee then
   return cup_of_coffee;
else
   return none;
end if;
```



	Inputs			Outputs
Test	Kettle	Mug	Coffee	Result
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

Structural Coverage Analysis of Ada

RapiCover

- Does Ada make SCA easier?
  - Yes!
  - Example: decision coverage
    - Case statements: ‘default’ case only when required
      - No dead code added for guideline compliance
  - Example: MC/DC
    - Proper Boolean type
    - Bitwise operators clearly separated from Boolean ones
    - Not mixing Boolean and non-Boolean data in expressions

Structural Coverage Analysis of Ada

RapiCover

- How many conditions is too many conditions?

RapiCover

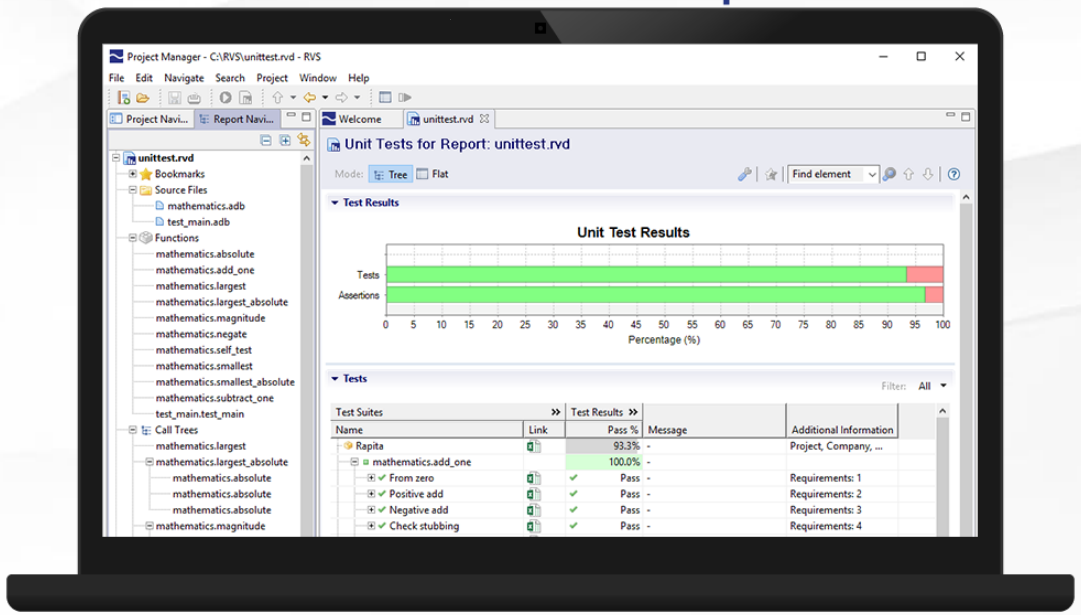
(demo)

Unit Testing of Ada

- Does Ada make unit testing easier?

- Black-box vs White-box testing
- Example pros:
  - Ada Specification
  - Value range for types
- Example cons:
  - Ada ‘private’

RapiTest



RapiTest

- RapiTest lets you test and stub private objects in your code without needing to write specific test code to expose them or modify your code.

```
package Conversions is

   type Speed_Data is private;

   function Get_Speed(Velocity : Float)
            return Speed_Data;

private

   type Motion_Type is ( Stationary,
                         Forward,
                         Backward
                       );

   type Speed_Data is record
      Direction : Motion_Type;
      Speed     : Float;
   end record;

end Conversions;
```

Scope

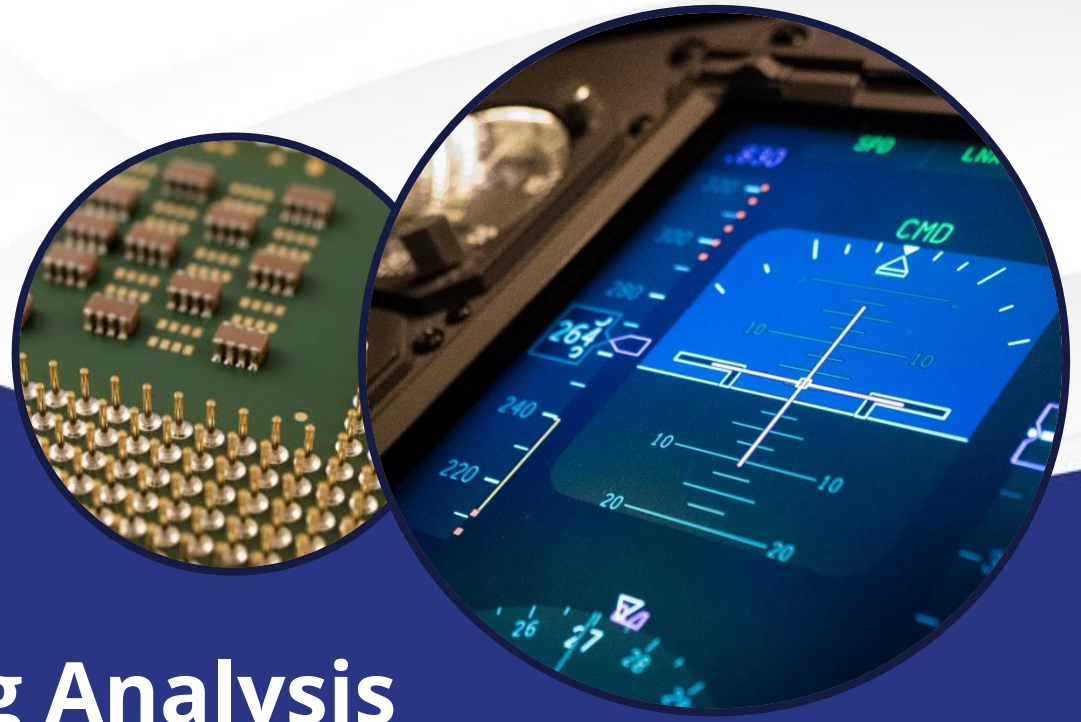
Subprogram name	Shorthand	Parameter	Type
Get_speed		Velocity	Float
		return	Conversions.Speed_Data

Tests

Test details	Call or stub	Variable	Operation	Value
New_Test				
Name				**Zero velocity**
	Get_Speed	Velocity	set	5
	:uut	return.Direction	check	Forward
	:uut	return.Speed	check	5
End_Test				

- This applies to
  - private ‘anything’ in packages in Ada
  - static variables and functions in C
  - private members in C++

RAPITA Systems
A DANLAW Company



DO-178C Multicore Timing Analysis

The challenges, best practices and a commercial solution

DO-178C MCP Certification

- DO-178C / ED-12C defines guidance for safety-critical airborne software
- **A(M)C 20-193 extends DO-178C guidance for MCP platforms**

History of CAST-32A & A(M)C 20-193

- **CAST-32A** introduced considerations for MCPs - 10 new objectives
- CAST-32A has been formalized as part of the DO-178C/ED-12C standard for MCPs
- The new formal guidance is defined in a new document called “AMC 20-193” in Europe (EASA) and “AC 20-193” in the US (FAA)

- **A(M)C 20-193**
  - EASA’s AMC 20-193 was released in Jan 2022
  - FAA’s AC 20-193 was released in Jan 2024
  - Rapita provided input into formulation process
  - Number of differences between CAST-32A and A(M)C 20-193…

The multicore challenge

Why not before?

- Complex & non-deterministic
- Lack of certification guidelines
- Expense of verification
- Lack of commercial solution

Why now?

- SWaP improvements
- Lack of long-term availability SCPs
- New guidelines – CAST-32A & A(M)C 20-193
- Clear roadmap to certification

Challenges of multicore verification

Identifying types of interference

- Direct, Indirect & Unknown sources

Check your assumptions, including

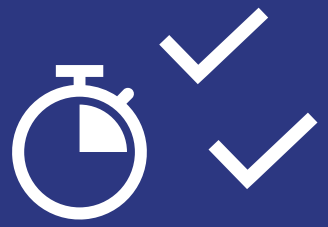
- Can you trust your measuring device,
  e.g., Performance Monitoring Counter (PMC)
- Can you trust RTOS-provided temporal isolation mechanisms?
- How does “partitioning” work? Is it effective?
- Confirm HW mitigation effectiveness

Quantification

- High water mark and worst-case tests
- Hybrid measurement-based approach
- Static WCET analysis insufficient for MCP

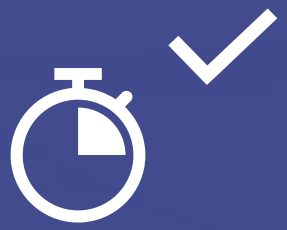
Challenges of multicore analysis

Category I systems
Deterministic



- Simple architecture
- Repeatable
- Small and Simple code structure
- → Simple measurements may be ok, or cycle counting

Category II systems
Predictable



- More complex but still predictable (e.g. limited cache use)
- Variability in execution time on same test runs
- Complex code and larger systems
- → measurements and analysis

Category III systems
Known Unpredictability

- Advanced processor features
- More unpredictability but quantifiable and controllable
- Large variability in execution times (e.g. L1 to L3 cache)
- Complex code and large
- → Lots of measurements and analysis

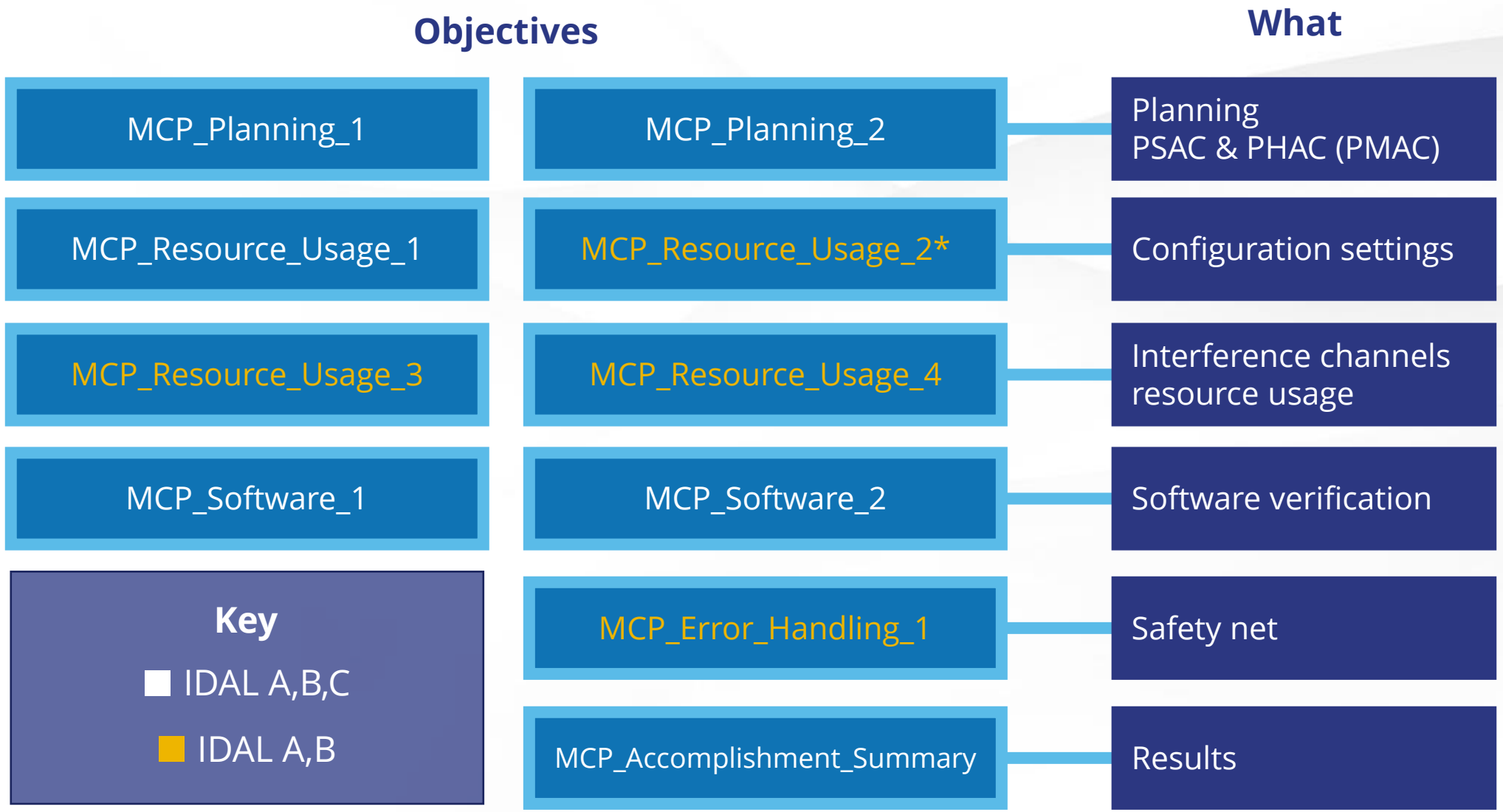
Category IV systems
Unknown Unpredictability

- Multicore, GPU
- Interference channels (some hidden)
- IP not disclosed
- Complexity
- → Deeper measurements and analysis

Check your assumptions

- How do you trust your "Measuring device" (e.g. event counters)
  - Do event counters count things right?
  - Do event counters count the right things?
- Do you trust RTOS-provided temporal isolation mechanisms?
  - Who provides the evidence?
  - RTOS: provides everything related to COTS OS implementation
  - Applicant: everything related to the specific usage of the COTS OS
- How does "partitioning" work? Is it effective?
  - Is it possible to provide true time partitioning in a multicore processor?
  - Issues e.g. MSHR and other register buffer overflows
- Are disabled devices "really" disabled?
- What is the impact of a change in the SW?
  - How sensitive is execution time to changes in SW
  - Small impact, no impact, or huge impact?

A(M)C 20-193 and CAST-32A Objectives

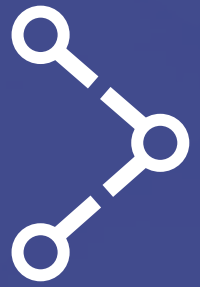


Types of interference

Direct

- **Competition for resources between SW components**
- Contention on the bus
- Competition for cache
- Access to memory

Indirect

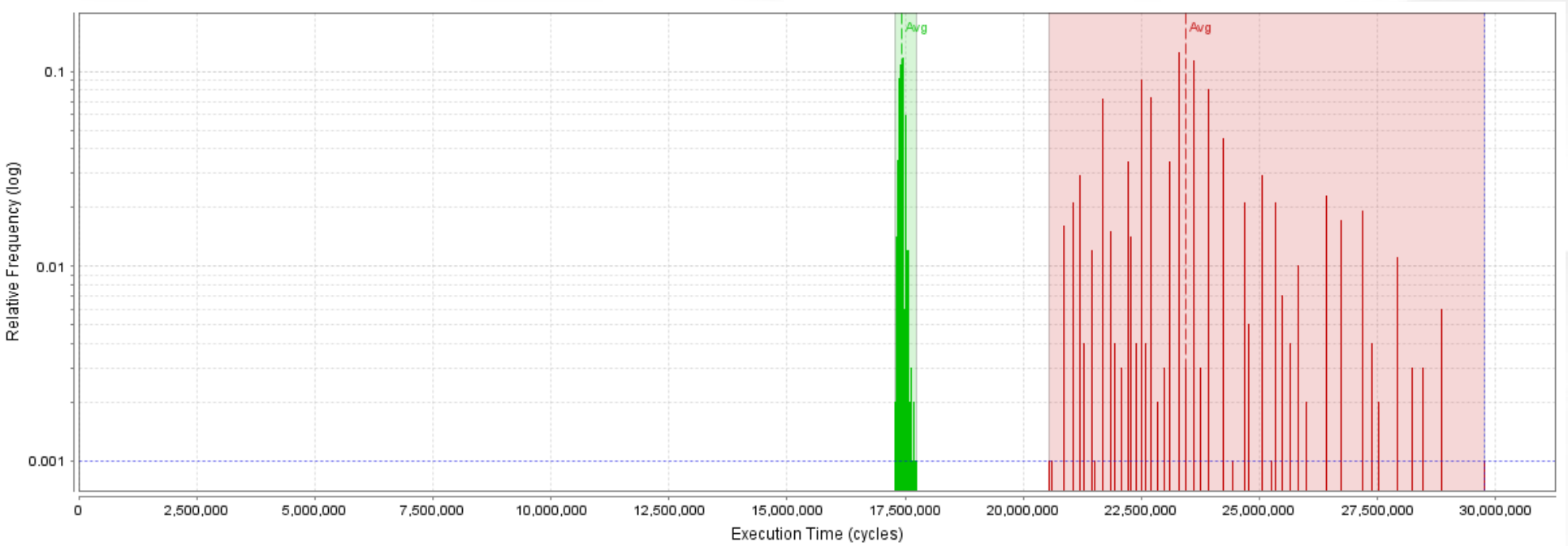


- **Execution time affected by hardware unpredictability**
- Different routing through interconnect
- Variability in memory access latencies
- Cache coherency protocols: snooping requests may block cache

Unknown

- **“That wasn’t supposed to happen”**
- P4080 bus arbitration is not fair
- Clock frequency slows down if CPU overheats due to too many FPU operations
- CPU behaviour/latency undocumented

How does interference affect predictability?



No contending RapiDaemons

3 contending RapiDaemons

Data source: open source canny edge detection software running on T2080

Quantification of the Effect of Interference Channels

- Main questions to be answered:
  - How can interference channels be exercised?
  - How can their potential impact be measured?
  - How can mitigation mechanisms be verified?
  - What is the effect of interference on software WCET with mitigations in place?

MCP_Resource_Usage_3:

The applicant has identified the interference channels that could permit interference to affect the software applications hosted on the MCP cores, and has verified the applicant's chosen means of mitigation of the interference.

MCP_Software_1:

The applicant has verified that all the software components hosted by the MCP meet the objectives of the applicable software guidance. In particular, the applicant has verified that all the hosted software components function correctly and have sufficient time to complete their execution when all the hosted software and hardware of the MCP is executing in the intended final configuration.

The way in which the applicant should satisfy this objective depends on the type of the MCP platform:

— ***MCP platforms with robust partitioning:***

Applicants who have verified that their MCP platform provides both robust resource partitioning and robust time partitioning (as defined in this AMC) may verify software applications separately on the MCP and determine their WCETs separately.

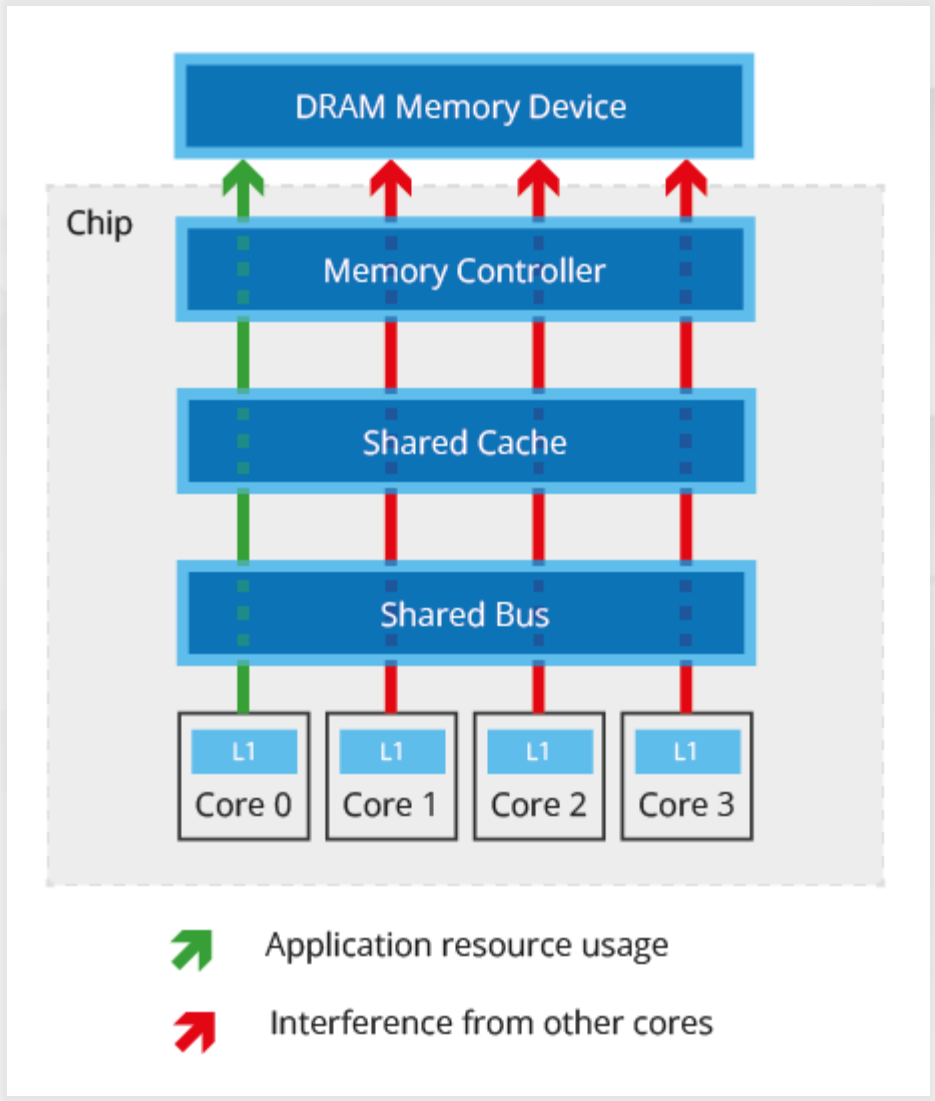
— ***All other MCP platforms:***

Applicants may verify separately on the MCP any software component or set of requirements for which the interference identified in the interference analysis is mitigated or is precluded by design. Software components or sets of software requirements for which interference is not avoided or mitigated should be tested on the target MCP with all software components executing in the intended final configuration, including robustness testing of the interfaces of the MCP.

The WCET of a software component may be determined separately on the MCP if the applicant shows that time interference is mitigated for that software component; otherwise, the WCET should be determined by analysis and confirmed by test on the target MCP with all the software components executing in the intended final configuration.

Quantification

- WCET analysis methods:
  - High water mark and worst-case tests
  - Hybrid measurement-based approach
  - Static WCET (does not work on multicores)
    - FAA: “Abstractions used for the WCET evaluation, for instance processor models, may not be correct or be so inaccurate that the computed WCETs are too pessimistic”
- 'Worst-case' tests are different with / without interference
- Rapi**Daemons**
  - DO-330 qualified interference generators
- Run applications together with Rapi**Daemons**
  - Difference in execution time with and without Rapi**Daemons** is the impact of interference
- WCET of SW component with final configuration is upper-bounded by WCET with Rapi**Daemons**

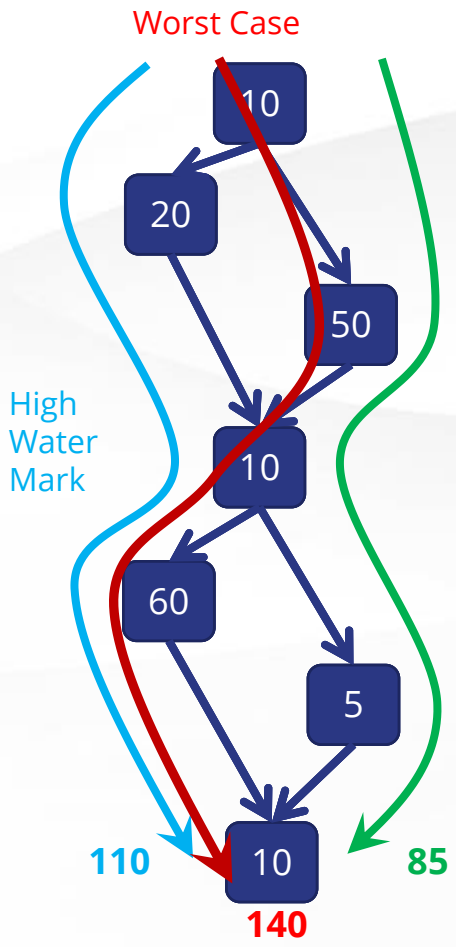


What multicore WCET is not

- An analytical hard upper-bound on the possible execution time
  - This is similar to the single-core case, but there are more ways things can go wrong
- A probabilistic upper-bound, with a well-defined probability of exceedance

What multicore WCET is

- A safe upper-bound on the execution time of the software under analysis, determined using worst-case interference on all identified interference channels.
- Expected to be used in conjunction with an engineering safety margin and a multicore safety net (see MCP_Error_Handling_1).



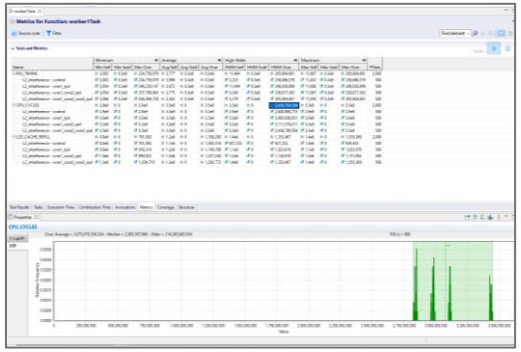

```
...
if( a )
{
    f1();
}
else
{
    f2();
}

...
if( b )
{
    f3();
}
else
{
    f4();
}

f5();
```

A Commercial solution: MACH178

RVS



The R**VS** toolsuite enables you to automate key parts of the multicore timing analysis workflow.

Services

Software/platform analysis & characterization, Target Integration Services, Training & Consulting.

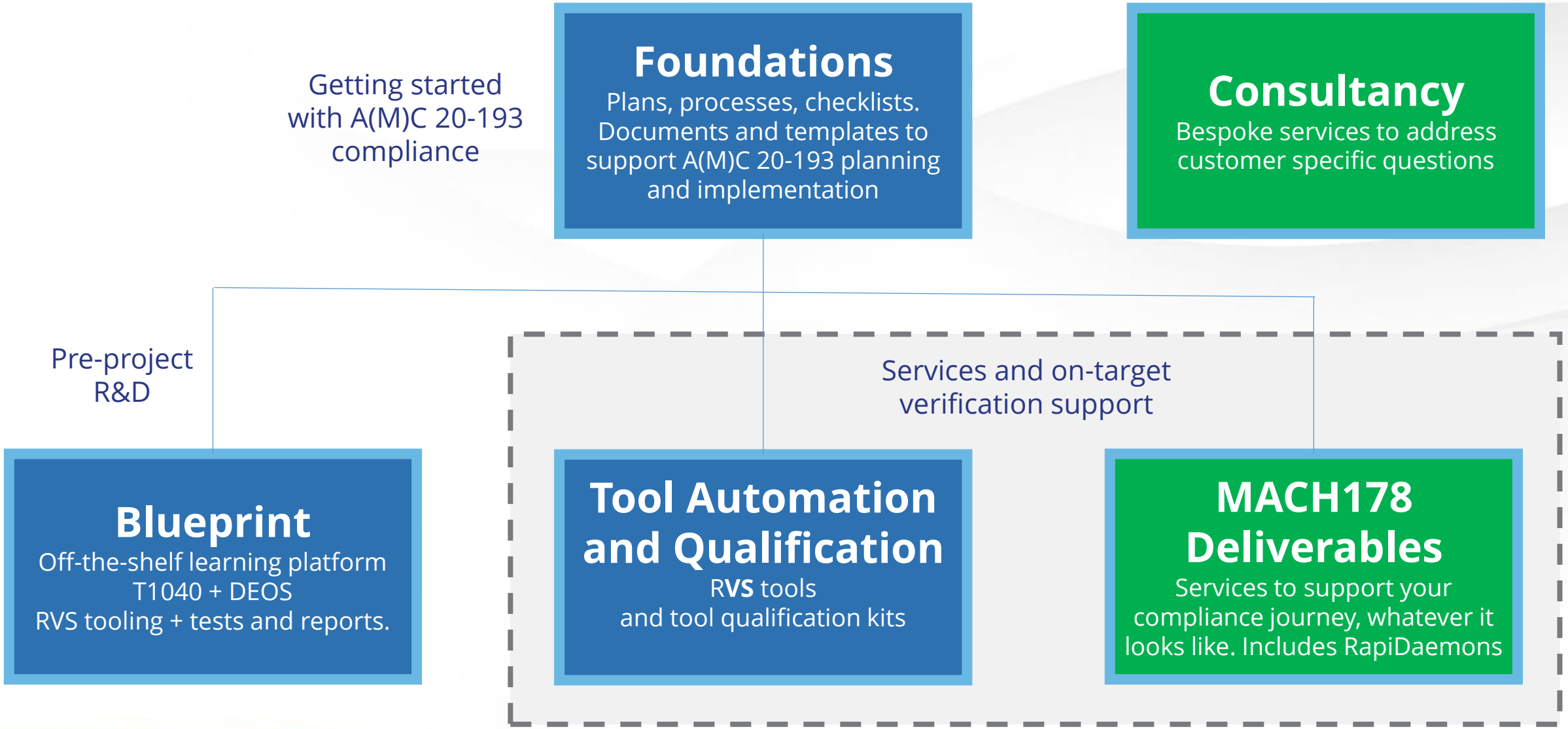
Rapi**Daemons**

Applications that deliver configurable contention for measuring the impact of interference.

Certification artifacts

A range of reports, tests, process documents and templates to meet certification objectives.

MACH^{178} Components – Overview



What's in... MACH178 Services?

Services to support your compliance journey, whatever it looks like

- **Platform Analysis and Characterization** to produce A(M)C 20-193 compliance evidence for your platform using the **MACH178** Foundations plans, processes and checklists.
- **Training** to help you understand A(M)C 20-193 compliance and how to use the **MACH178** Foundations workflow to produce compliance evidence
- **Integration** to set up a verification environment to support use of the **MACH178** workflow on your specific project
- **Consultancy** to answer any questions you may have about A(M)C 20-193 compliance
  - How to evaluate multicore processors
  - Whether it's right to use single core activation
  - How to achieve MCP_Software_1 and _2 objectives
  - And more...

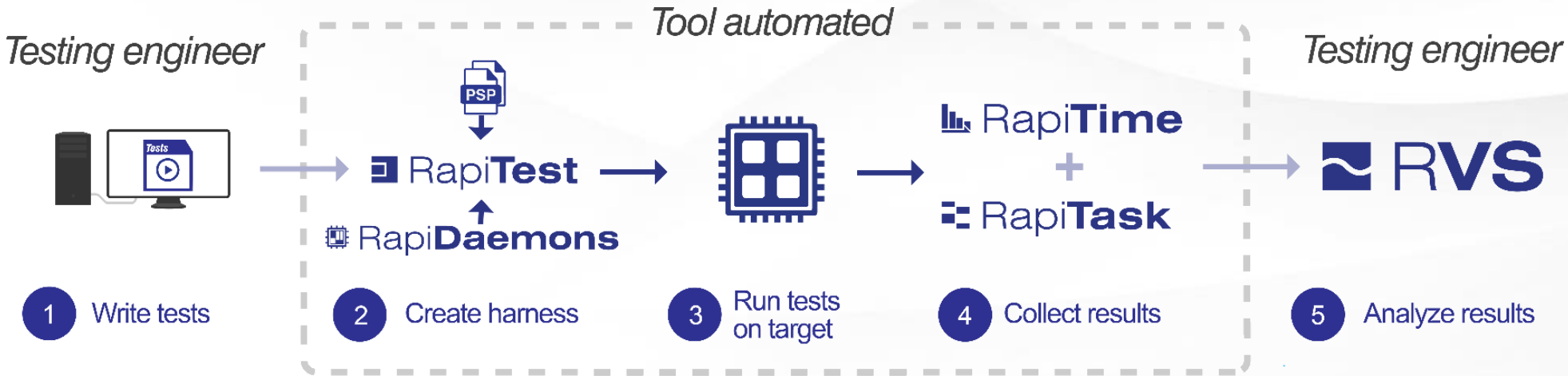
Software Tools

RapiTest

Produce and run tests that exercise MC software for execution time.

RapiTime

Automatically collects execution time metrics while tasks run on-target.



RapiDaemons

RapiDaemons create interference while multicore tasks are analyzed.

RapiTask

Automatically measures and reports scheduling metrics.