

GUIDELINE FOR ECSS CRITICALITY CATEGORY A QUALIFICATION OF FLIGHT SOFTWARE

WHAT ARE THE CONCERNS AND WHAT TO LOOK AT
FROM A SW PA PERSPECTIVE

Andoni Arregui

2023-09-26

ESA SW PA Workshop, ESAC

GTD GmbH



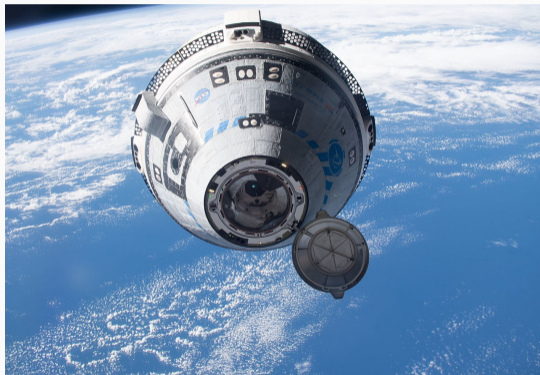
TABLE OF CONTENTS

1. Motivation
2. ECSS Category A Software
3. The Guideline We Developed
4. Takeaways

MOTIVATION

DO WE CARE?

What is this?



The Boeing CST-100 Starliner

DO WE CARE?

“We are no longer building hardware into which we install a modicum of enabling software, we are actually building software systems which we wrap up in enabling hardware. Yet we have not matured to where we are uniformly applying rigorous systems engineering principles to the design of that software. These are serious and pervasive issues that NASA will need to address in all of its programs and certainly will be critical to space exploration endeavors.”



Patricia Sanders –
NASA’s Aerospace Safety
Advisory Panel Chair

Why does Europe need Category A Qualified Software?

- To **continue being a player** in international cooperation and
- **achieving autonomy** in access to space.



Revolution Space report

Why does Europe need Category A Qualified Software?

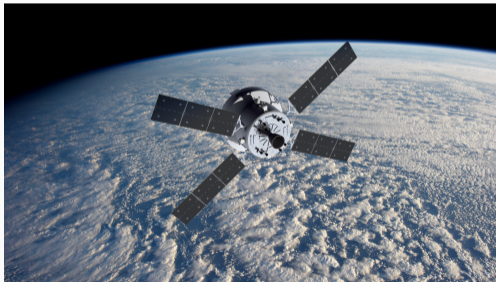
- To **continue being a player** in international cooperation and
- **achieving autonomy** in access to space.



ATV 5 units flown between 2008 and 2015

Why does Europe need Category A Qualified Software?

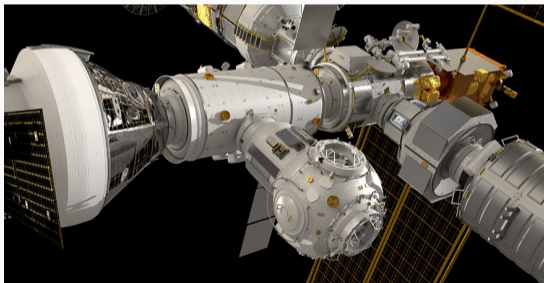
- To **continue being a player** in international cooperation and
- **achieving autonomy** in access to space.



Orion European Service Module 6 units contracted for the Artemis missions

Why does Europe need Category A Qualified Software?

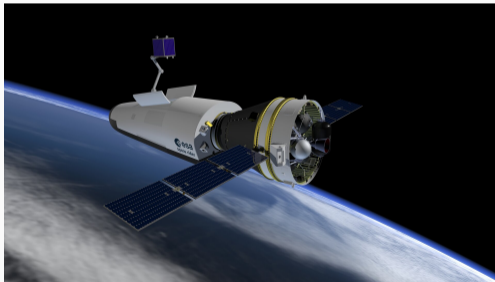
- To continue being a player in international cooperation and
- achieving autonomy in access to space.



I-Hab and ESPRIT Refueling Module Built for the lunar gateway

Why does Europe need Category A Qualified Software?

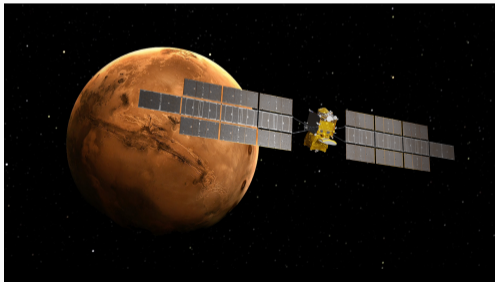
- To **continue being a player** in international cooperation and
- **achieving autonomy** in access to space.



Space-Rider First one to be launched end of 2024

Why does Europe need Category A Qualified Software?

- To **continue being a player** in international cooperation and
- **achieving autonomy** in access to space.



Mars Sample Return - Earth Return Orbiter

Why does Europe need Category A Qualified Software?

- To **continue being a player** in international cooperation and
- **achieving autonomy** in access to space.



ADRIOS Launch in 2026

ECSS CATEGORY A SOFTWARE

What is needed on top of the Category B requirements

- MC/DC Structural coverage
- **Verification** of additional **Object Code**

MC/DC structural coverage

5.8.3.5 Verification of code

- b. The supplier shall verify that the following code coverage is achieved

Code coverage versus criticality category	A	B	C	D
Source code statement coverage	100%	100%	AM	AM
Source code decision coverage	100%	100%	AM	AM
Source code modified condition and decision coverage	100%	AM	AM	AM

NOTE: "AM" means that the value is agreed with the customer and measured as per ECSS-Q-ST-80 clause 6.3.5.2.

EXPECTED OUTPUT: Code coverage verification report [DJF, SVR; CDR, QR, AR].

NOTE This requirement is met by running unit, integration and validation tests, measuring the code coverage, and achieving the code coverage by additional (requirement based) tests, inspection or analysis.

- c. Code coverage shall be measured by analysis of the results of the execution of tests.

3.2.18 modified condition and decision coverage

measure of the part of the program within which every point of entry and exit has been invoked at least once, every decision in the program has taken “true” and “false” values at least once, and each condition in a decision has been shown to independently affect that decision’s outcome

NOTE A condition is shown to independently affect a decision’s outcome by varying that condition while holding fixed all other possible conditions.

Unique Cause MC/DC

This understanding of MC/DC is deprecated since 2001 (CAST-6 DO-178B)

- e. In case the traceability between source code and object code cannot be verified (e.g. use of compiler optimization), the supplier shall perform additional code coverage analysis on object code level as follows:

Code coverage VS. criticality category	A	B	C	D
Object code coverage	100%	N/A	N/A	N/A
NOTE: N/A means not applicable.				

What is needed on top of the Category B requirements

- **Verification** of additional **Object Code**

Table A-5: Test coverage requirements

COVERAGE TYPE	CRITICALITY CATEGORY				
	A	B	C	D	
Statement Coverage (Source Code)	100 %	100 %	100 %	AM	
Statement Coverage (Object Code)	100%	AM	AM	AM.	
Decision Coverage (Source Code)	100%	100%	AM	AM.	
Modified Condition & Decision Coverage (Source Code)	100%	AM.	AM	AM.	

NOTE: "AM" means that the value is agreed with the customer and measured as per ECSS-Q-ST-80 clause 6.3.5.2.

Only statement, no decision coverage on object code?

Branches in object code will not be properly exercised. Example

We are not used to it

- *SW Engineering:*
 - Neither to comply with its **requirements**
 - nor to produce the required **evidence**
- *SW Product Assurance:*
 - Neither to **interpret** the evidences
 - nor to ask the right **questions**

Past Cat-A approaches sometimes flawed

- ATV MSU: Only does condition coverage on object code, no MC/DC¹.
- ESM PDE: Requires *atomic decisions*, losing MC/DC error detection capacity and seems to incorrectly assess object code traceability².

¹§5.2 in *Category A Software Development for the ATV*, Boudillet, Berthelie, Zekri, 2005

²§3.5 and §3.7 in *Critical Software for Human Spaceflight*, Preden, Kaschner, Rettig, Rodriggs, 2019

What are the main concerns left for Category A software?

1. Are the **requirements detailed enough** for the criticality level?
⇒ Quite subjective but Cat-A needs more concrete requirements Example
2. Has the implemented software **logic been sufficiently tested**?
⇒ MC/DC is required for Cat-A

What are the main concerns left for Category A software?

1. Has the executable production **introduced object code** that has **not been verified nor tested**?
⇒ Verification of additional object code required for Cat-A
2. Have the requirements been **validated** on a **sufficiently representative platform** and environment?
⇒ Validating on and closing on non fully representative platforms may hide errors
3. Has the **ISVV** activity been **adequately** carried out in accordance with the required criticality level?
⇒ Tasks like IVE.CA.T3 do not even require unit tests to be cross-compiled for target

MC/DC is a syntactical attribute of the source code

Restructuring source code on purpose will lower its error detection potential.

This complex decision:

```
bool complex_decision(bool a, bool b, bool c, bool d) {  
    return ((a && b) || (c && d));  
}
```

Will require 4 tests to achieve MC/DC.

NOTE: The 4 test cases refer to the ones needed to achieve the so called *masking* MC/DC with a number of tests $2 \cdot \lceil \sqrt{n} \rceil$, where n is the number of conditions in the decision.

Rewriting it as

```
bool complex_decision(bool a, bool b, bool c, bool d) {  
    bool result = false;  
    if (a && b)  
        result = true;  
    if (c && d)  
        result = true;  
    return result; }
```

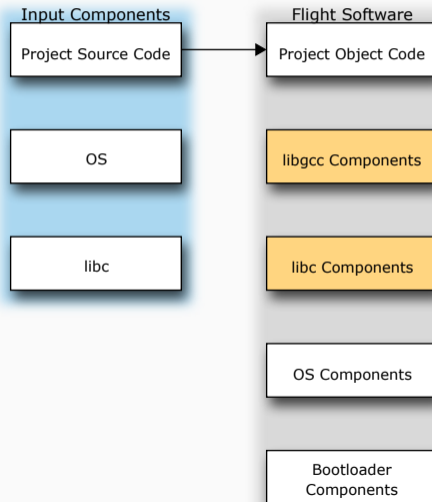
Will *only* require 3 tests and will fail to detect a regression if the `if (c && d)` decision is removed.

SW PA shall ask:

Did you enforce simple or atomic decisions in source code to cheat on MC/DC?

IMPORTANCE OF ADDITIONAL OBJECT CODE VERIFICATION

A **flight software** is usually composed of **more than only the project source code**:



Compilers & linkers introduce additional object code to flight executable

The flight software is not only composed of your project source code!

Additional object code in flight software

1. Elements from the compiler library such as `.udiv` from `libgcc`
2. Elements from the standard C library not explicitly called by the source code such as `memset()`
3. &c.

IMPORTANCE OF ADDITIONAL OBJECT CODE VERIFICATION

You won't notice at first these functions being added

Adding a modulo operator on 64 bit integers will do this on SPARC V8 architectures:

```
unsigned long long int compute (unsigned long long int a, unsigned long long int b) {  
    return a % b; }
```

compute:

```
save    %sp, -96, %sp
```

```
...
```

```
mov     %i0, %o0
```

```
! This is such a hidden call  
to libgcc function __umoddi3
```

```
call    __umoddi3, 0
```

```
...
```

```
restore %g0, %o1, %o1
```

SW PA shall ask:

Are all function calls in your object code also listed in your design?

IMPORTANCE OF ADDITIONAL OBJECT CODE VERIFICATION

The structure of your object code is not the same as your source code

1. Compilers generate **extra branches** where there were non in source code
2. Compilers **rearrange execution paths** within functions for **optimization**

Structural coverage on source code not sufficient

The project has **no evidence** that these **new branches and path** structures have ever been exercised or **verified**.

SW PA shall ask:

Can you prove that your object code has no untested branches?

THE GUIDELINE WE DEVELOPED

Contractual Context

The work has been carried out under ESA Contract No. 4000138220/22/NL/AS/adu since 2022.

- ESA aimed at the development of a method and its corresponding tools to systematically promote ECSS Category B software to Category A.
- All the work has been carried out with great **support of the ESA** Technical Officer **Andreas Jung**.

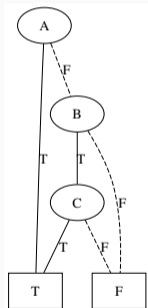
Method

- Step-wise systematic method to cover the two main gaps:
 - MC/DC Coverage (Referenced by NASA-HDBK-2203; to be integrated in the NASA CAP)
 - Verification of additional object-code (Often called object to code traceability)
- Best Practices and FAQs

Tools

- All open-source based (Alternatively proprietary tools can be used)
- Assist in the following tasks:
 - Assess MC/DC coverage
 - Gather structural coverage on object code (On a function basis)
 - Construct function Control Flow Graphs in object code and assist in the object code structural coverage assessment

The method proposes a tool (mcdc-checker) to **assess** the source code **decision structure** so that afterwards the standard tool **gcov** can be used to **assess MC/DC**.



BDD for $a \vee b \wedge c$

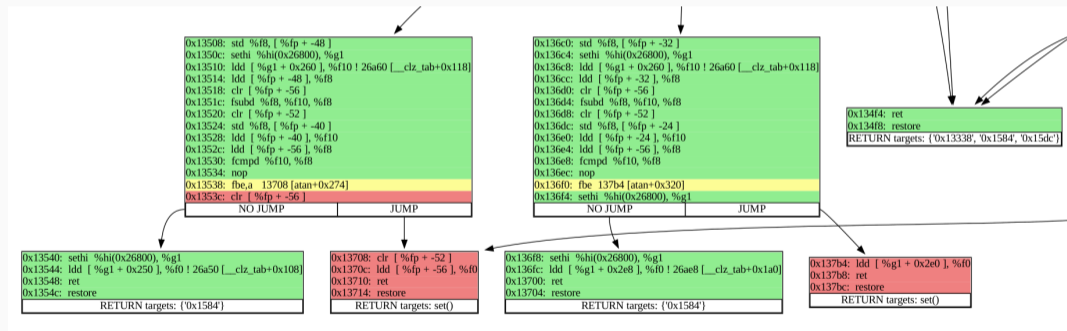
```
4 : /* Correct implementation where gcov requires MC/DC */
5 : bool version_1(bool a, bool b, bool c)
6 : {
7 :     bool result = false;
8 :
9 :     if (a || (b && c))
10 :         result = true;
11 :     else
12 :         result = false;
13 :
14 :     return result;
15 : }
```

[+ + + -
+ +]

Structural coverage equivalent to MC/DC

ASSESSMENT OF OBJECT CODE

The method helps assessing the existence of object code added during the executable production and proposes tools to gather **object code condition coverage** (occtre) and present it on the **CFG** of each function (asm2cfg).



We shall distrust the compiler for Category A software

1. Object code analysis information based on DWARF debug information generated by the cross-compiler.
2. Does the cross-compiler and/or linker add additional object code we are not detecting?

Use complementary tools for more independence

1. Assessment tools can come from different cross-compiler.
 - Newer versions can be used.
 - LLVM analogous tools can be used substituting GCC/binutils.
2. Linker information to cross-check compiler generated information.
3. Completeness of bidirectional object to source code traceability can be verified.

TAKEAWAYS

Software PA shall ask if

1. **Structural coverage** data has been gathered **only with unit tests** or also with validation tests.
2. Source code **structural coverage** has been gathered with **non optimized compilation**.
3. **Unit tests** have been **cross compiled** and executed on target.
4. The project **verified function symbols** in object code that **come** from **outside** of the project source code.

Software PA shall not

1. Allow **source code** to be rewritten with **only simple decisions** (e.g., if (A)).
2. Allow gathering **structural coverage** data only on optimized **object code** (and not on source code).
3. Accept **completeness of object to source traceability** only because **source traces** for all object code addresses have been **produced**.
4. Accept **object code coverage** is complete by checking **all project source code functions**.