ECSS Category A Software Evidence: Why and How?

How to Upgrade your Flight Software to ECSS Category A

Andoni Arregui*, Fabian Schriever*, Thomas Wucher*, Christoph Weiß*, Andreas Jung**

2023-06-07

DASIA 2023

* GTD GmbH, ** European Space Agency (ESA)

Table of Contents

Motivation

Initial Questions

Do you know the software you fly?

- Do you think you thoroughly tested the logic you implemented in software?

Initial Questions

Do you know the software you fly?

- Do you think you thoroughly tested the logic you implemented in software?

You may think so but you didn't

You don't have complete evidence that every condition you have in your decisions contributes to the decisions as specified.

Initial Questions

Do you know the software you fly?

- Do you think you thoroughly tested the logic you implemented in software?

You may think so but you didn't

You don't have complete evidence that every condition you have in your decisions contributes to the decisions as specified.

Do you know the software you fly?

- Do you think you are flying functions you never tested nor verified?

Initial Questions

Do you know the software you fly?

- Do you think you thoroughly tested the logic you implemented in software?

You may think so but you didn't

You don't have complete evidence that every condition you have in your decisions contributes to the decisions as specified.

Do you know the software you fly?

- Do you think you are flying functions you never tested nor verified?

You may think you don't but you do

You are flying functions introduced by the cross-compiler that no one validated.

European Need for Category A Software

Why does Europe need Category A Qualified Software?

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

European Need for Category A Software

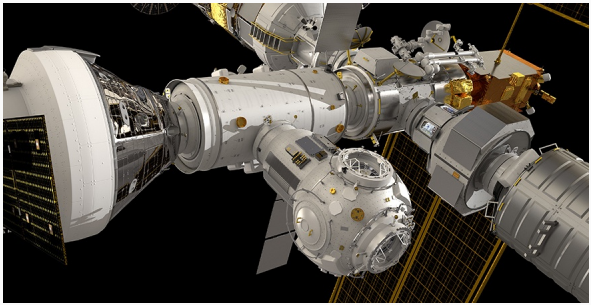
Why does Europe need Category A Qualified Software?

- 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.

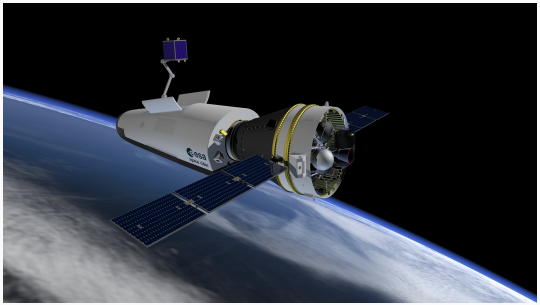


I-Hab and ESPRIT Refueling Module Built for the lunar gateway

European Need for Category A Software

Why does Europe need Category A Qualified Software?

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

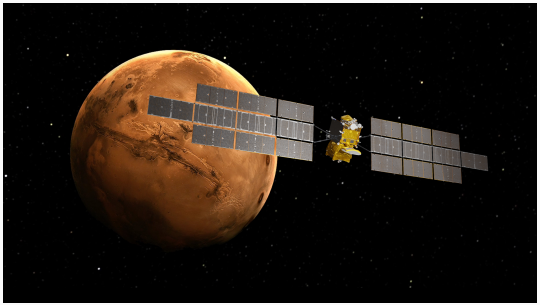


Space-Rider First one to be launched end of 2024

European Need for Category A Software

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

European Need for Category A Software

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

European Challenge with Category A Software

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**

European Challenge with Category A Software

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 experiences are flawed

- ATV MSU: Category A approach based on 1 false assumption
- ESM PDE: Category A approach based on 2 false assumptions

ECSS Category A Software

Main ECSS Requirements for Category A Software

What is needed on top of the Category B requirements

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

What are the main concerns left for Category A software?

1. Are the requirements detailed enough for the criticality level?
2. Has the implemented software logic been sufficiently tested?
3. Has the executable production introduced code that has not been verified nor tested?
4. Have the requirements been validated on a sufficiently representative platform and environment?
5. Has the ISVV activity been adequately carried out in accordance with the required criticality level?

Importance of MC/DC

MC/DC is a syntactical attribute of the source code

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

Importance of MC/DC

MC/DC is a syntactical attribute of the source code

Rewriting 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.

Importance of MC/DC

MC/DC is a syntactical attribute of the source code

Rewriting 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.

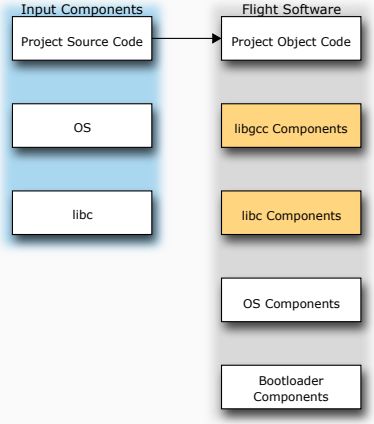
Rewriting it as

```
bool complex_decision(bool a, bool b, bool c, bool d) {
    return 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.

Importance of Additional Object Code Verification

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



Importance of Additional Object Code Verification

Compilers and linkers will introduce additional object code to your executable

Your flight software is not only composed of your project source code.

Additional object code in your flight software

1. Elements from the compiler library such as `.udiv` from `libgcc`
2. Elements from the standard C library you are not explicitly using 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;
}
```

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     %i2, %o2
    mov     %i3, %o3
    mov     %i0, %o0
    call    __umoddi3, 0
    mov     %i1, %o1
    mov     %o0, %i0
    jmp     %i7+8
    restore %g0, %o1, %o1
```

Importance of Additional Object Code Verification

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

1. The compiler will generate branches where there where non in source code
2. The compiler will rearrange execution paths within functions for optimization purposes

Structural coverage on source code not sufficient

The project has no evidence that these new branches and path structures have been ever exercised nor verified.

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 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 and Tools

Method

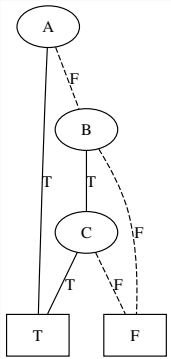
- Step-wise systematic method to cover the two main gaps:
  - MC/DC Coverage (Has already been referenced by NASA-HDBK-2203)
  - Verification of additional object-code (Often called object to code traceability)
- Best Practices
- FAQs

Tools

- All open-source based
- Alternative proprietary tools can be used
- To help with the following tasks:
  - Assess MC/DC coverage
  - Establish semantic equivalence of object code with and without debug information
  - 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

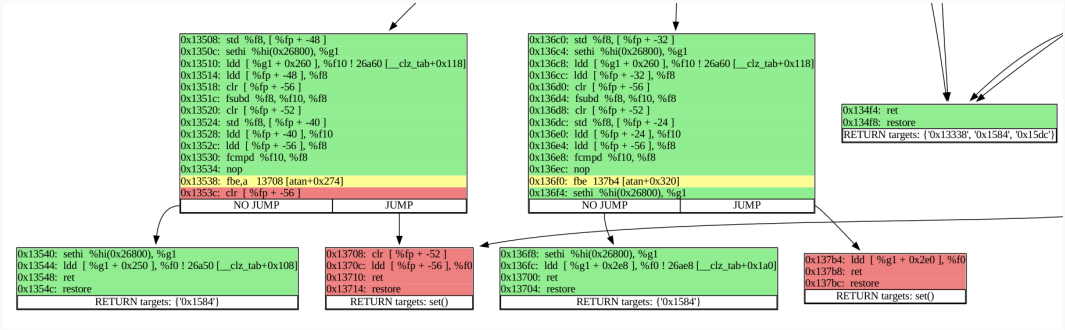
The source code structural coverage of a function including MC/DC and its corresponding Binary Decision Diagram as assessed by the proposed tools:

```
4                :            : /* Correct implementation where gcov requires MC/DC */
5                :          3 : bool version_1(bool a, bool b, bool c)
6                :            : {
7                :          3 :     bool result = false;
8                :            :
9  [ +  +  +  -  :          3 :     if (a || (b && c))
          +  + ]
10               :          2 :         result = true;
11               :            :     else
12               :          1 :         result = false;
13               :            :
14               :          3 :     return result;
15               :            : }
```



Assessment of Object Code

The object code of a function, its control flow graph structure and object code coverage can be assessed in this form:



Self Criticism

We shall not trust the compiler for Category A software

1. Object code analysis information is 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?

The use of complementary tools avoids this problem

1. The assessment tools used do not need to be part of the cross-compiler.
   - Other versions can be used.
   - Analogous tools from the `llvm` project can be used
2. The linker information is used to cross-check the compiler generated information.
3. The completeness of the object code to source code and the source code to object code traceability can be verified.

Takeaways

Some Best Practices

DOs

1. Gather structural coverage data only with unit tests.
2. Gather source code structural coverage with non optimized compilation
3. Execute unit tests on target.
4. Check for function symbols in object code that come from outside of the project source code

DON'Ts

1. Rewrite source code to have only simple decisions (e.g., `if (A)`)
2. Gather structural coverage data only on optimized object code (and not on source code)
3. Assume you have object to source traceability because you have source traces for all your object code
4. Assume you completed object code coverage by checking all project source code functions

Further Steps

Further Steps

Enhancement of the method

- The open source tools proposed for the method and the ones specifically developed for it need to be properly qualified to the corresponding ECSS level.
- Further target architectures need to be added (RISC-V, ARM, Power-PC)

What about the data?

- Is all the data we fly justified? Do we read and write it?
- Apart from being range checked, how are the configurable values related to the tests defined and executed?