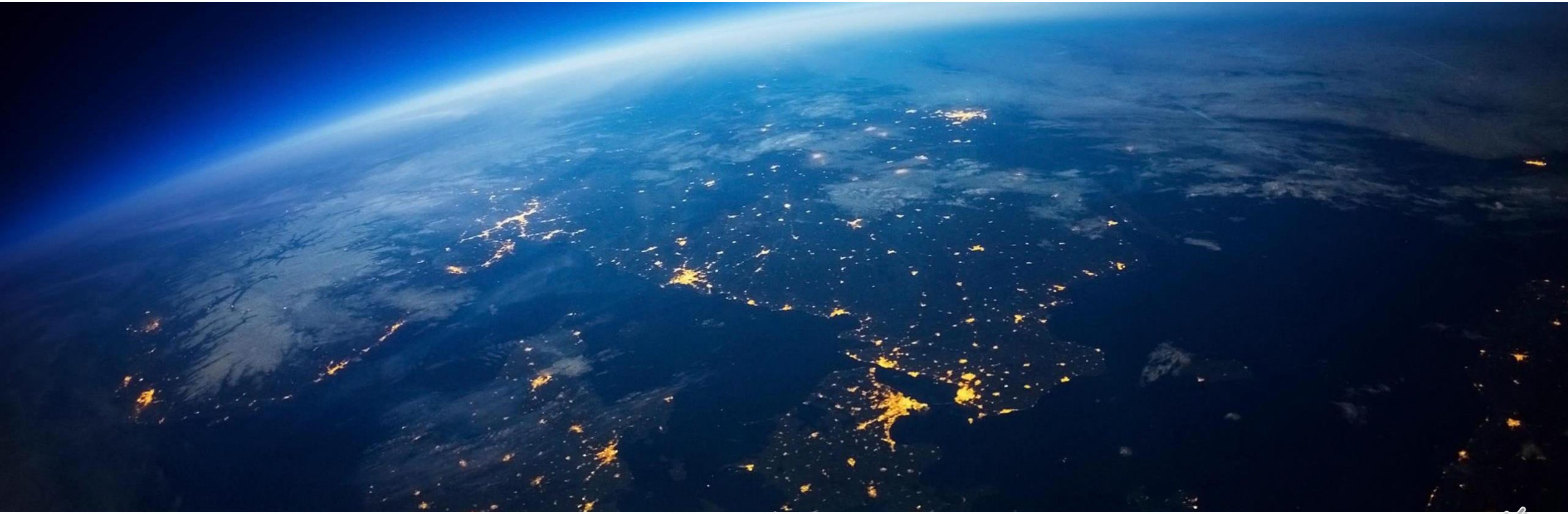
SENER Aeroespacial

CoRA-SAGE: The lessons learnt from AOCS/GNC algorithms deployment in TASTE

Antonio Figueroa González



Model Based Space System and Software Engineering – MBSE 2020

Content

- Overview of CoRA-SAGE Project
- AOCS/GNC development flow: From Simulation to HW implementation
- CoRA-SAGE as TASTE use case
- Lessons Learnt

CoRA SAGE Overview

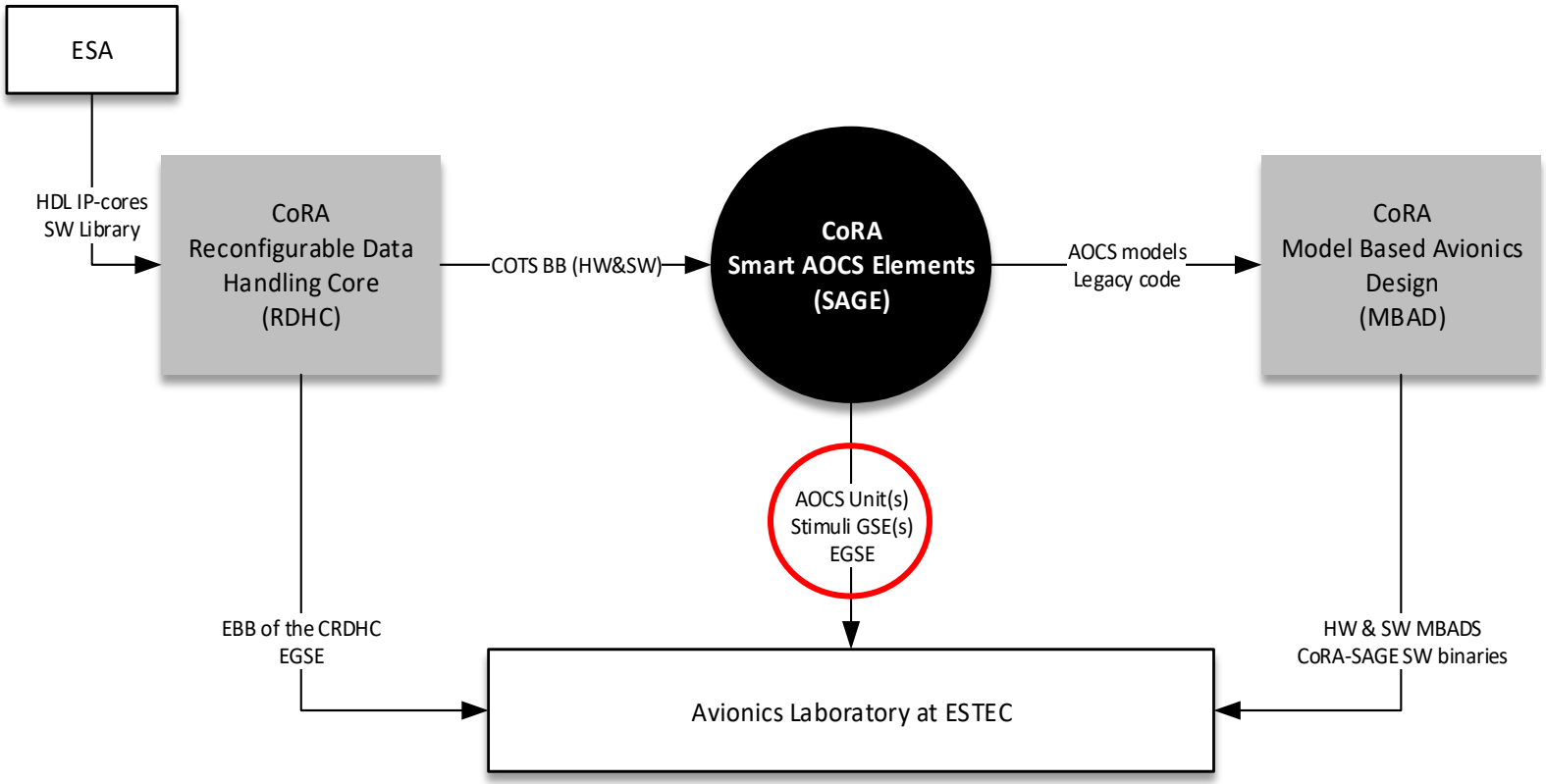
SENER Aeroespacial

CoRA-SAGE Overview

- Compact Reconfigurable Avionics (CoRA) is a co-engineering activity involving AOCS&GNC, software engineering and on-board data handling
- Objective: ***to prototype an in-flight reconfigurable avionics system***

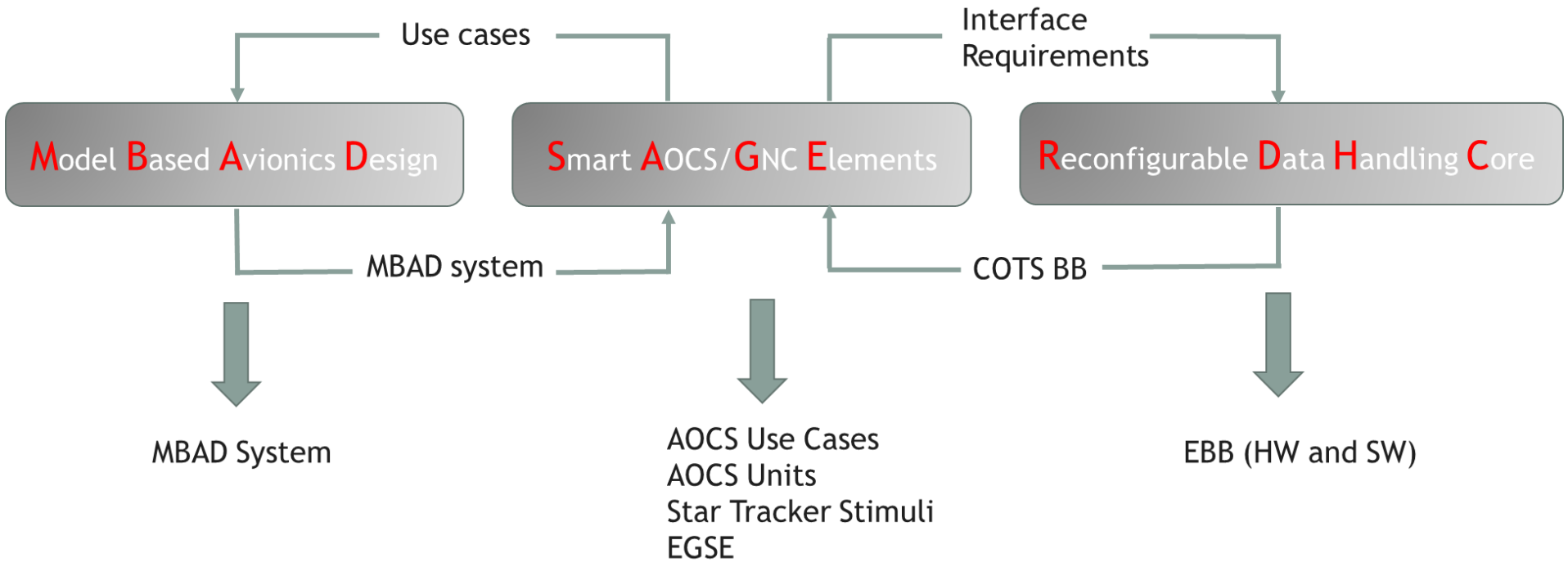
CoRA-SAGE Overview

- **CoRA** has three branches:
  1. **RDHC**: *Reconfigurable Data Handling Core*
  2. **SAGE**: *Smart AOCS/GNC Elements*
  3. **MBAD**: *Model Based Avionics Design*



CoRA-SAGE Overview

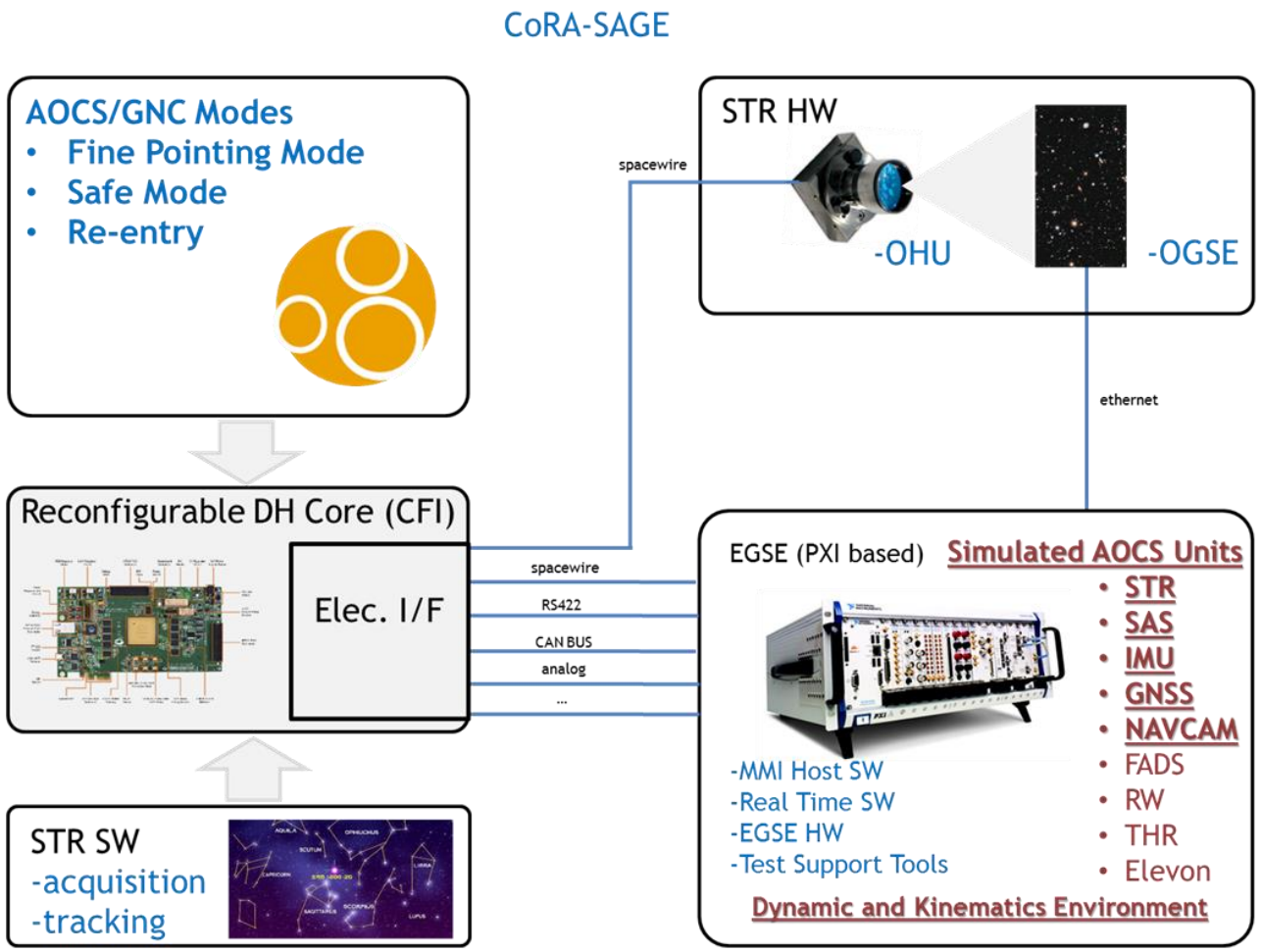
- **CoRA** has three branches… **As a result, the Exchange of information, models and code was critical:**
  - **Each branch is formed by a different consortia**
  - **The evolution of each branch is strongly interlinked to the others**
  - **There was not a company responsible of the overall system**



CoRA-SAGE Overview

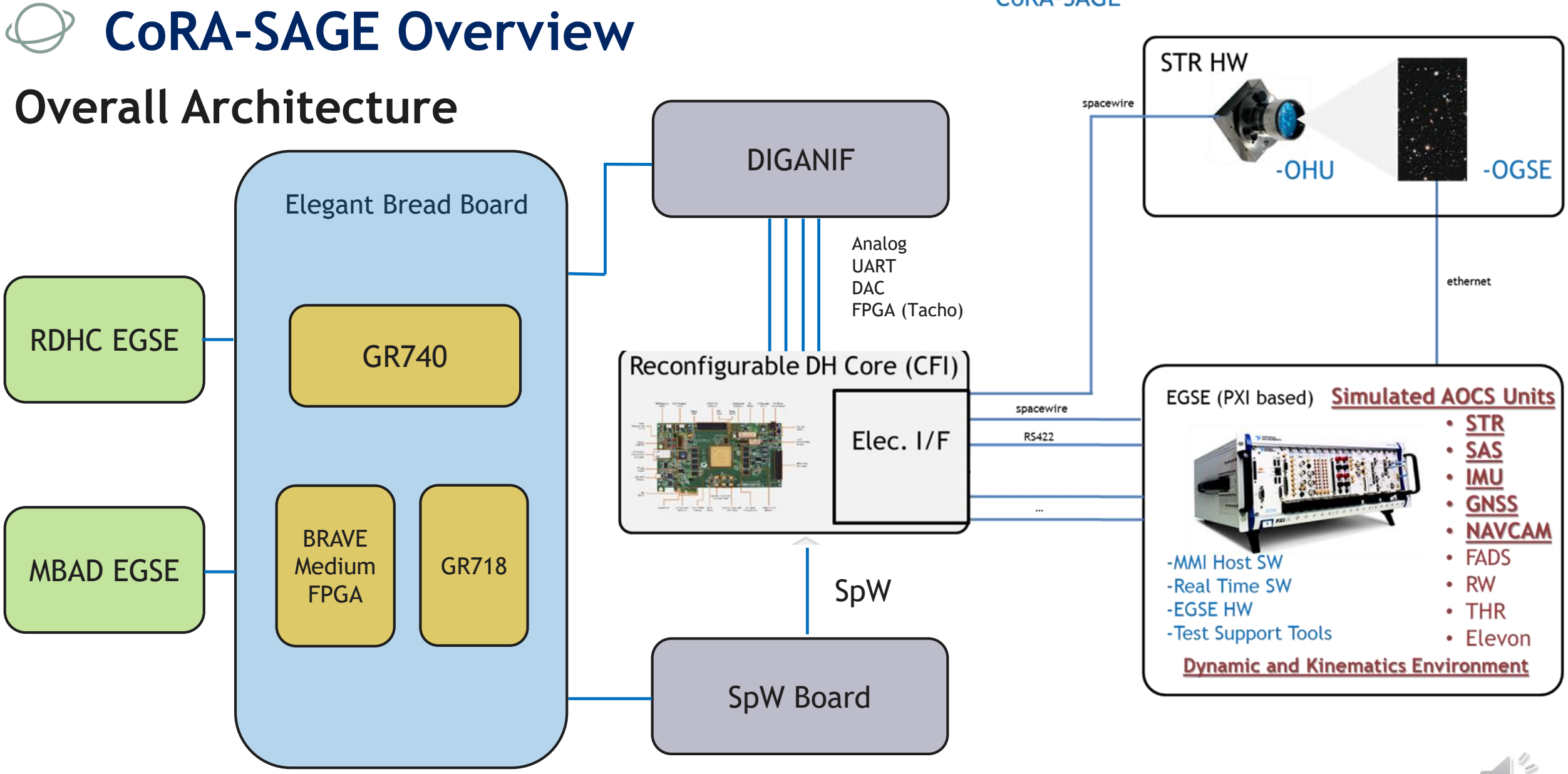
CoRA-SAGE Objectives:

- Develop AOCS/GNC to be exercised in the reconfigurable avionics
- Develop the Electronic Ground Support Equipment, which includes:
  - Electrical interfaces of simulated AOCS components (GNSS, SAS, IMU, FADS, RCS, RW) with the Avionics
  - Simulation of sensors, actuators and dynamics-kinematics and environment
  - Optical Ground support equipment for the STR
  - Stimulation of a Star Tracker Unit (HIL)



CoRA-SAGE Overview

Overall Architecture



AOCS/GNC development flow: From Simulation to HW

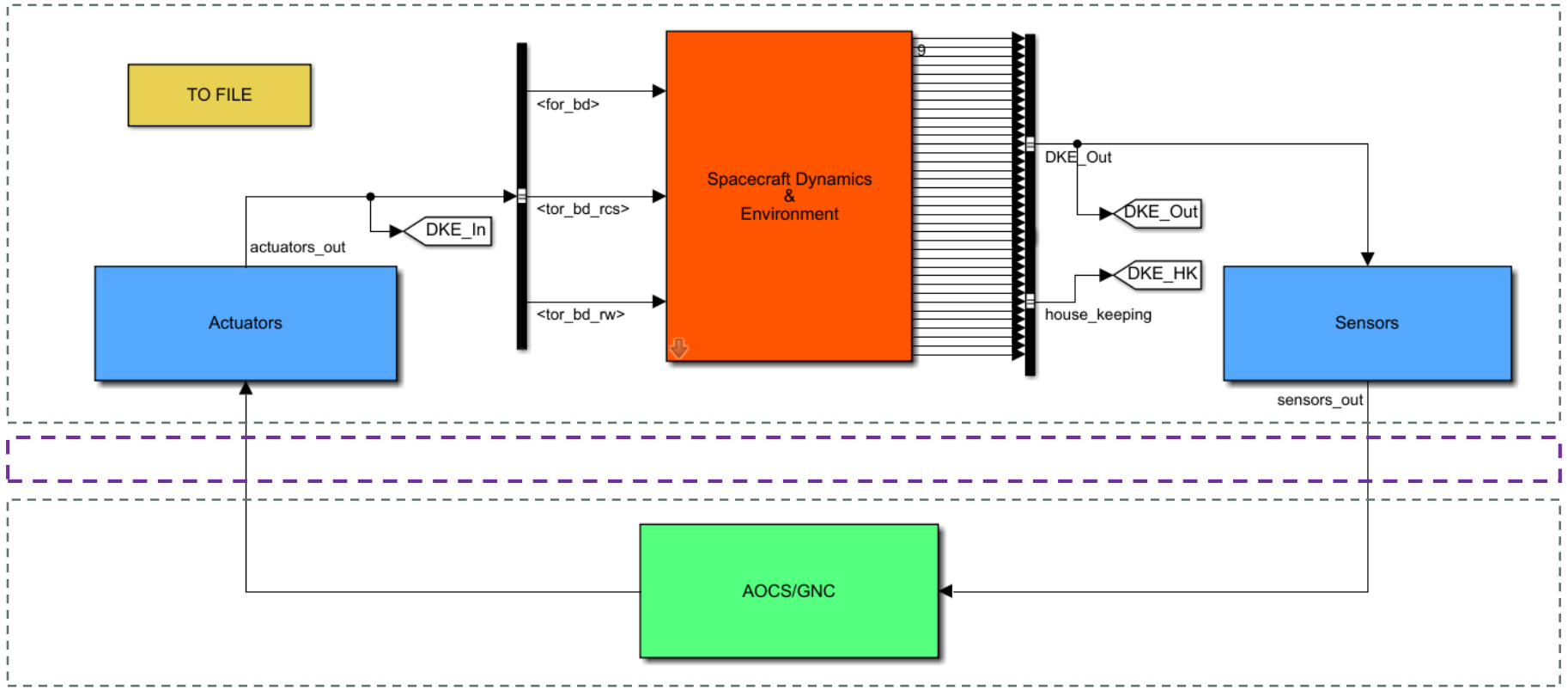
SENER Aeroespacial

CoRA-SAGE AOCS/GNC development flow

- CoRA-SAGE selected Space Rider reusable servicing vehicle as strawman test configuration.
- From the full set of the Space Rider mission, SENER included in CoRA-SAGE the Fine Pointing Mode (FPM), Safe Mode (SM) and Re-Entry Mode (REM), which are representative of the AOCS/GNC modes present in any space mission
- The Functional Engineering Simulator as well as the AOCS/GNC modes were developed in **Matlab/Simulink**

CoRA-SAGE AOCS/GNC development flow

Dynamics-Kinematic and Environment; and Sensors and Actuators to be executed by EGSE



System interfaces

AOCS/GNC modes to be deployed in the Bread-Board

CoRA-SAGE AOCS/GNC development flow

- The design of the AOCS/GNC modes should be modular and allow to reconfigure the AOCS/GNC functions partitioning between hardware and software implementations, hence allowing the implementation of the same algorithm both in FPGA or in the processor

- Thus, AOCS/GNC shall comply with the pointing and performances requirement using a fixed-point implementation

CoRA-SAGE AOCS/GNC development flow

- AOCS/GNC shall comply with the pointing and performances requirement using a fixed-point implementation… Thus, many development steps were required:

- AOCS/GNC algorithms design using floating point representation
  - Demonstration of performances
    - AOCS/GNC conversion into fixed-point
      - Demonstration of performances
        - Automatic Code Generation
          - Verification of the generated code in an emulated processor
            - Embedding of AOCS/GNC into the CoRA platform
              - Verify the HIL closed loop features
                - Closed Loop tests at CoRA System Level

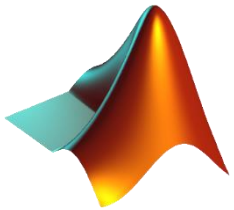
CoRA SAGE as TASTE use case

CoRA-SAGE as TASTE use case

TASTE tool served to centralize the interfaces definition, which affects to:

- FES in Matlab/Simulink environment ⟷ AOCS/GNC design
- EGSE interfaces provision ⟷ EGSE design
- Data Handling Core interfaces ⟷ CoRA-RDHC and CoRA-MBAD projects
- Digital/Analogue converter

CoRA-SAGE as TASTE use case



- AOCS/GNC algorithms design using floating point representation
  - Demonstration of performances
    - AOCS/GNC conversion into fixed-point
      - Demonstration of performances
        - Automatic Code Generation

taste
The Assert Set of Tools for Engineering

- Verification of the generated code in an emulated processor
  - Embedding of AOCS/GNC into the platform
    - Verify the HIL closed loop features
      - Closed Loop tests at CoRA System Level

CoRA-SAGE as TASTE use case

TASTE tool served to centralize the interfaces, and made compatible the Simulink algorithms with the TASTE environment:

1. Interface Definition in TASTE
2. TASTE generates the AOCS/GNC interfaces in Matlab/Simulink
3. The AOCS/GNC algorithms are easily plugged into the TASTE skeletons in Matlab/Simulink, being compatible with the rest of the FES

This allowed:

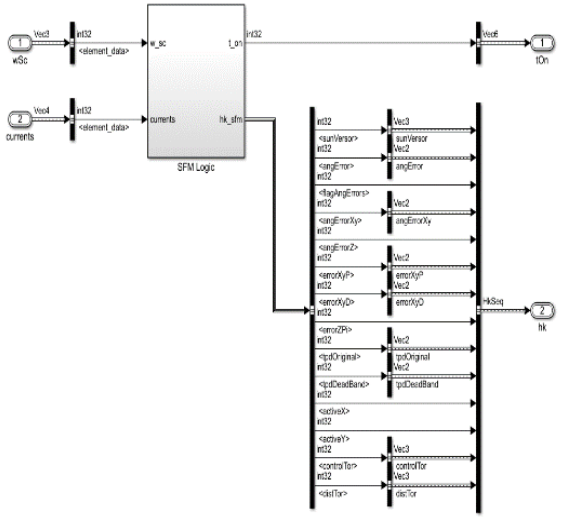
- ***Sharing the AOCS/GNC models and code with the SAGE-SW team and the other CoRA projects, assuring compatibility with the MBAD system***
- ***Incremental code verification from the early project phases***

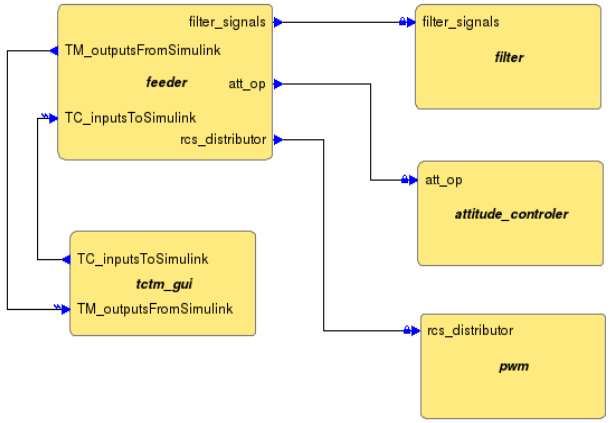
CoRA-SAGE as TASTE use case

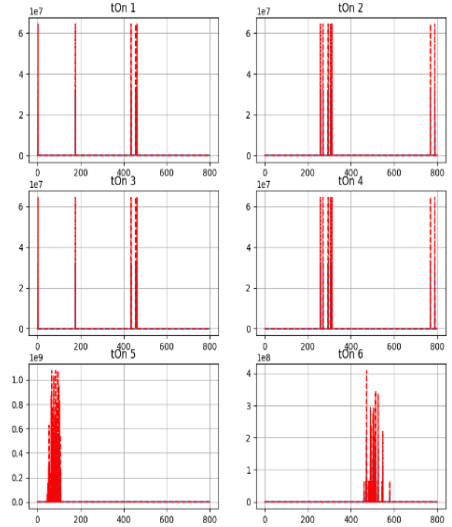
The TASTE features for early verification and testing of the generated software (GUIs and Python scripts) were employed to verify the deployment

Data View | Interface View | Deployment View | Concurrency View | AADL

```
TASTE-Dataview DEFINITIONS ::=
BEGIN
IMPORTS T-Int32, T-UInt32, T-Int8, T-UInt8, T-Boolean FROM TASTE-BasicTypes;

TASTE-Boolean ::= BOOLEAN

MyInteger  ::= INTEGER (-4294967296.. 4294967296)

Vec2  ::= SEQUENCE(SIZE(2)) OF MyInteger
Vec3  ::= SEQUENCE(SIZE(3)) OF MyInteger
Vec4  ::= SEQUENCE(SIZE(4)) OF MyInteger
Vec6  ::= SEQUENCE(SIZE(6)) OF MyInteger

HkSeq ::= SEQUENCE{
        sunVersor Vec3,
        angError Vec2,
        flagAngErrors MyInteger,
        angErrorXy Vec2,
        angErrorZ MyInteger,
        errorXyP Vec2,
        errorXyD Vec2,
        errorZPi MyInteger,
        tpdOriginal Vec2,
        tpdDeadBand Vec2,
        activeX MyInteger,
        activeY MyInteger,
        controlTor Vec3,
        distTor Vec3
}

END
```







Variables definition in TASTE

Makes the generated code compatible

Open loop tests Infrastructure

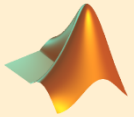
OL Verification

1 → 2 → 3 → 4

CoRA-SAGE as TASTE use case

- The open loop tests verification in TASTE paved the way to deploy successfully the AOCS/GNC modes both on the COTS-BB and on the Elegant-BB (definitive BB designed for CoRA)
- It served to
  - Verify that auto-generated code
  - Implement changes in the AOCS/GNC algorithm design quickly

- AOCS/GNC algorithms design using floating point representation
  - Demonstration of performances
    - AOCS/GNC conversion into fixed-point
      - Demonstration of performances
        - Automatic Code Generation



- Verification of the generated code in an emulated processor
  - Embedding of AOCS/GNC into the platform
    - Verify the HIL closed loop features
      - Closed Loop tests at CoRA System Level

CoRA-SAGE as TASTE use case

It served to implement changes in the AOCS/GNC algorithm design quickly while respecting the defined interfaces: *a real example*

- After the Critical Design Review of CoRA-SAGE, it was reported by CoRA-RDHC that the AOCS/GNC modes did not fit into the FPGA
- It was decided to split the AOCS/GNC modes in smaller functions
- Thus, CoRA-SAGE team was to modify the design, include extra interfaces and verify again the deployment in TASTE…
- The verification approach fasten this process!

CoRA-SAGE as TASTE use case

Overall Classical Verification Approach

- Algorithm Design
- Code specification / autocoding reqs.
- Coding / automatic code generation
- Coding of additional SW layers to deploy the AOCS/GNC mode into the target platform
- SW validation
- Algorithm V&V in the avionics test bench

Verification using TASTE

- Algorithm Design
- Interface definition in TASTE and automatic code generation compatible with TASTE
- Verification in TASTE with an emulated processor
- SW/HW partitioning from TASTE

CoRA-SAGE as TASTE use case

Classical Verification Approach

- *Algorithm design changes after CDR imply repeating a great effort*
- *SW/HW implementation requires manual partitioning*

Verification using TASTE

- *Algorithm design changes after CDR do not imply a high effort*
- *The same tool:*
  - *Defines the IF,*
  - *Integrates the automatic generated code (and is per se compatible with it)*
  - *Serves to deploy the AOCS/GNC modes in to the target*
  - *Allows SW/HW partitioning*

Lessons Learnt

Lessons Learnt

- Centralized Interfaces:
  - Successful approach: CoRA-SAGE SW and AOCS teams and other parties could share easily the TASTE models
  - Changes in the AOCS algorithms (implemented in Matlab/Simulink) did not affect the TASTE infrastructure, hence it allowed quick verification tests

- Verification using TASTE:
  - It fastened the integration of AOCS modes - BreadBoard - EGSE
  - Thanks to this approach, **we did not find any-flaws during the integration of AOCS/GNC modes with the other elements due to the AOCS/GNC implementation**

Lessons Learnt

- About TASTE:
  - ✓ Easy to plug and test different SW pieces (pure C-code, Matlab/Simulink, interface drivers…)
  - ✓ Emulation of Linux and GR740 processors served to check from the beginning that the code implementation was correct
  - ✓ What about increasing the list of emulated processors?
  - Some cons…
    - Sometimes, it was hard to debug very little mistakes
    - There is no guide for creating the test infrastructure and execution… We asked for some help at the beginning! The tests infrastructure implied:
      - Manipulating the ‘MSC’ interface
      - Extract an auto-generated skeleton
      - Modify it and run it via python