RAPID DEPLOYMENT OF DESIGN ENVIRONMENT FOR EUCLID AOCS DESIGN

Francesco Cacciatore (1), Raúl Sánchez (1), Alfredo Agenjo (1), Nicolás Puente (1), Carlos Ardura (1), Luis Olier (1), Víctor Gómez (1), Massimiliano Saponara (2), Gonzalo Saavedra (3)

(1) SENER, (2) Thales-Alenia Space Italy, (3) ESA-ESTEC

ABSTRACT

Euclid is a cosmology mission dedicated to study the geometry and the nature of the Dark Universe with unprecedented accuracy. The Euclid S/C is procured by ESA and supplied by TAS-I; SENER is the prime contractor of the AOCS sub-system, for which the work is executed in partnership with ADS-NL. In this frame, SENER has developed a reference simulation tool (the Euclid Design Simulator, EDS) which allowed a fast deployment of environment for completing AOCS SRR in 1 month and AOCS PDR in 7 additional months. The EDS allows from quick design-and-run iterations to complete Monte Carlo execution, and is the reference platform on which the AOCS algorithms are implemented and evolved up to auto-coding. The EDS is also the reference tool used to support the units' models development and specification, making use of suppliers' information. The EDS will be evolved and maintained through the whole project life cycle.

Index Terms— EUCLID, AOCS, Design Environment, Simulator

1. INTRODUCTION

Euclid is a cosmology mission dedicated to study the geometry and the nature of the Dark Universe with unprecedented accuracy. Euclid will observe a 15000 deg^2 wide area of the sky from the Lagrange point L2 of the Sun-Earth system. The scientific goals of the mission result in very demanding performances for the AOCS subsystem; as an example, the observations' Relative Pointing Error shall be kept within 75 mas over a time scale of 700 sec. Amongst other responsibilities, SENER is responsible for the design, implementation and verification of GNC/AOCS algorithms. In the frame of the activities for the Phase B2 of the study, a set of tools and methodologies were developed and employed for the design tasks undertaken. The core of the design effort resided in the preparation and set-up of the Euclid Design Simulator (EDS). Such simulator is based on the internal tool SENERIC, whose model library was used as core or starting point for the DKE, actuators and sensors models required for Euclid. Along with the units and dynamics simulation, a simulation management architecture was developed. The main user for the EDS is the AOCS engineer, focusing on design tasks. As a consequence, the EDS simulation management infrastructure was developed taking into account the following usage guidelines: fast data preparation and processing, ability of storing and retrieving simulated cases, ability of running different models with the same source data set, ability of running Monte Carlo simulations, all without requiring lengthy and complex case set-up. The product of such effort is a simulator which allows launching test cases from MATLAB Environment initialization (leveraging on native types for data definition), as well as flexibly generating and retrieving XML databases equivalent to the MATLAB native data types. The EDS currently makes use of Simulink libraries and model referencing, and further efforts are envisaged towards the auto-coding of the AOCS/GNC algorithms being designed through the EDS. It is being used for developing models and specifications aiming at an ideally seamless integration of the AOCS algorithms within the formal verification environment, in first step via Functional Engineering Simulator (FES), and later for the Real Time Simulator (RTS) production to be applied in the AOCS SCOE. EDS is applied during the whole process as the flexible reference design tool.

2. THE EDS AND THE AOCS DEVELOPMENT

In the frame of the EUCLID AOCS development, two main Simulation environments are being employed: the Euclid Design Simulator (EDS), and the Engineering Simulation Environment (ESE).

The ESE falls under responsibility of Elecnor-Deimos, and is a FES according to the standard classification of simulation tools recommended by ESA. The ESE is one of the test environments foreseen to support the overall Euclid AOCS verification and validation process. The ESE infrastructure purpose is to perform the formal verification of the AOCS requirements by analysis with respect to the Technical Specification (TS), to execute the validation of the control units models, and support the verification of the AOCS requirements by test.

The EDS, in turn, is a Design tool, meant to be employed by SENER from the beginning of the activities and maintained and used over the whole project execution. The aim of the EDS is to allow for the flexibility required in the design process, which is opposed to the more rigid environment needed and expected for the V&V phase. The

EDS is being employed for the design iterations, and is the framework in which the AOCS modes, their function, interfaces and associated databases are being defined.

AOCS sensors and actuators models are developed with an iterative process starting from the existing models in the EDS, and evolved through information provided by the units' suppliers, with previous evaluation and specification carried out using the EDS.

3. SENERIC

SENER developed a suite of high fidelity tools for the design analysis and validation of an AOCS, organized in a framework called SENERIC. This framework was initially funded by the Spanish Ministry of Technology and was later turned into an internally funded R&D project for its continuation.

Currently, the emphasis has been put on the development of an extensive library of simulation models within the MATLAB/Simulink environment. These models range from dynamics and environmental models to AOCS hardware models to complete closed loop simulators. Visualization, analysis, and post-processing tools are also included.

SENERIC is in continuous development since all tools and models that are developed in individual projects are being added to it for further reuse.

The suite contains models and libraries for:

- AOCS Equipment: sensors and actuators
- AOCS/GNC Functions: for rapid prototyping, analysis and design.
- Environment: force-torque perturbations, ephemerides, time references
- Mathematical operations & transformations
- S/C Dynamics and kinematics

SENERIC was employed in the design of the Planck [1] IXV AOCS [2], in PROBA-3 [3] and [4], and in the OPTOS spacecraft for the design and automatic code generation of the Attitude Control System, thus successfully proving its readiness level.

In the frame of EUCLID, SENERIC models have been used as core and starting point for the development of the EDS dynamics, actuators and sensors.

In addition, the SENERIC mathematical library was taken as the basis from which the EUCLID-AOCS mathematical library was created, with the final goal of being autocoded into flight software.

4. DYNAMICS, SENSORS AND ACTUATORS MODELS

The core of the EDS is composed by the simulation models for dynamics, sensors and actuators.

The dynamics is based on a joint integration of the S/C and reaction wheels, sloshing, and telescope Filter Wheel Assembly (FWA/GWA) dynamics equations.

In this section a brief description of the main models implementations is provided. The EDS Simulink core is show in Figure 2.The S/C equations of motion consist of a rigid body with a set of AOCS reaction wheels and the FWA/GWA wheel. The following equation represents the dynamics that are implemented in the numerical model.

$$\begin{bmatrix} I_{tot} & I_{RW}U_{RW} \\ I_{RW}U_{RW}^{T} & I_{RW}I_{n\times n} \end{bmatrix} \begin{pmatrix} \dot{\bar{\omega}}_{SC} \\ \dot{\bar{\omega}}_{RW} \end{pmatrix} =$$
$$\begin{pmatrix} -\bar{\omega}_{SC} \times (I_{tot}\bar{\omega}_{SC} + I_{RW}U_{RW}\bar{\omega}_{RW} + \bar{u}_{FWA}I_{FWA}\omega_{FWA}) \\ -\bar{T}_{f}(\bar{\omega}_{RW}) \end{pmatrix} +$$
$$\begin{pmatrix} \bar{T}_{ext} \\ \bar{T}_{RW} \end{pmatrix} + \begin{pmatrix} -\bar{u}_{FWA}I_{FWA}\dot{\omega}_{FWA} \\ \bar{0} \end{pmatrix}$$

Eq. 1: Dynamics equation

The FWA/GWA is modelled by explicitly imposing a kinematic motion by directly injecting the angular rate and acceleration profiles. Naturally, they are not independent and the rate is the integral of the acceleration. This approach ensures both a correct resulting torque profile and conservation of angular momentum of the closed system.

The main point concerning the implementation of the MPS model was the simulation of the prescribed error characteristics, given in terms of noise PSD. The required MPS noise PSD profile has been realized by opportune filtering of a zero-mean band-limited white noise. Specifically, a discrete time realization (based on Tustin transformation) of a 2-poles filter was employed. An example realization of the MPS noise is provided in the time-series plotted in Figure 1.

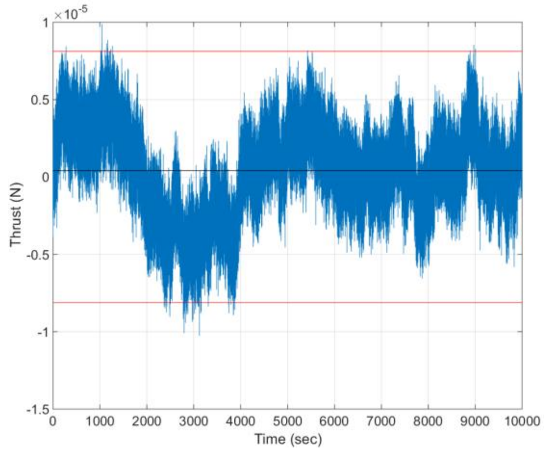

Figure 1: MPS noise time-series realization

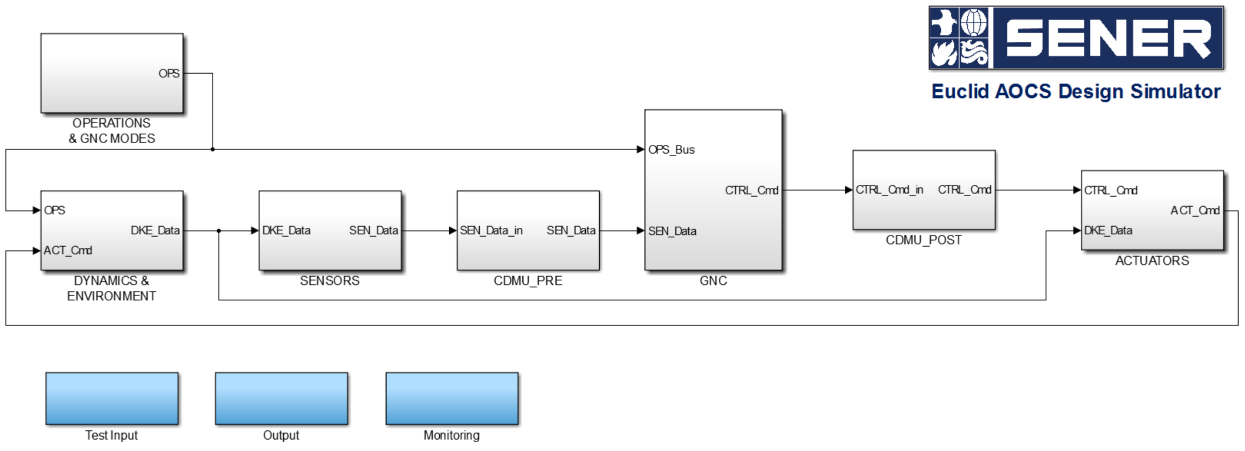

Figure 2: EDS Simulink Core

As for the MPS, a detailed modelling of the Gyroscope noise characteristics was carried out by opportune filtering of white noise sources. The implementation takes into account the following noise contributions: Angle Random Walk, Bias Instability (Flicker Noise), Rate Random Walk. Each of these noise elements has been tuned to match the Allan variance shape provided for the gyro unit.

Accelerometers were based on a similar approach to the one employed for gyroscopes; linear accelerations derived by the placement of the sensor and the S/C rotational dynamics were included in the model.

The RCS model includes the following key elements: integration & sampling, thrust time profile, noise and plume impingement. To correctly simulate the effect of a RCS pulse independently of the sample time set into the EDS, the thruster force is scaled over an AOCS cycle in order to obtain an analogue linear impulse to the one obtained with the nominal thrust and the corresponding pulse duration (lower than the AOCS sample time). Thrust rise and tail profiles are modelled, and a uniform random distribution on the thrust magnitude is added. The thrust alignment w.r.t. the thruster's axis is modelled with a uniform random distribution; the thrust angle misalignment remains constant within a simulation while the circumferential angle around the thrust axis is uniformly distributed. This is generated between different RCS pulses.

Sun sensors were simulated taking into account deviations from the theoretical cosine curve for the output current with a polynomial representation. Along with typical sensor noise, the Earth albedo effect is included relying on an offline computation taking into account Earth illumination and the actual sensor field of view.

The Reaction Wheels Assembly (RWA) dynamics model complements the attitude dynamics model. This RWA dynamics computes the friction per wheel using the static friction as well as the viscous and Coulomb frictions. The RWA torque model includes a dead-time during torque sign reversal. During this time, the RWA does not command torque. The dynamical response represented by a first order transfer function is included in the simulations. The RWA is thus represented as a sort of distributed model, with separate integration of the equations of motion (along with the S/C) and dynamic characteristics simulation.

The Fine Guidance Sensor (FGS) simulation was carried out making use of an in-house custom built model, implementing the customer-specified sensor characteristics. The sensor tracking logic (measurement lock, acquisition delays, data outages, measurement delays) were modelled, along with the absolute/relative tracking performances as function of the S/C angular velocity. A custom wrapping layer was developed to allow the EDS to be able to easily swap such in-house model for a customer-provided FGS model. The customer-provided data and model was incrusted in the EDS architecture with ideally none (or at least minimum) modifications.

Star trackers are modelled taking into account their integration and processing delay, the measurement bias and a velocity-dependent noise. Spatial effects due to FOV and detector characteristics (see [5] and [6]) are modelled respectively as a first and second transfer functions; such transfer functions are dynamically integrated, as they have non-constant gains that are function of the S/C angular velocity and rotation axis direction. Such implementation allows transforming the spatial errors into temporal noises with the motion of the S/C, and to consider them as biases when inertial pointing is achieved.

The AOCS is executed as part of the onboard software, running on the CDMU (Central Data Management Unit). In order to represent correctly the timely data flow due to the finite AOCS execution time, specific CDMU pre-GNC and post-GNC functional subsystems were included in the model.

5. EDS WORKFLOW AND INFRASTRUCTURE

One of the main issues observed in the EUCLID AOCS design process, common to all similar projects, is the contrast between two of the main needs of the design team: on one hand the design of AOCS algorithms and functions, and on the other hand the testing and performance assessment of such algorithms.

The design activity requires an environment whose main characteristic is the flexibility. The design engineer must be able to execute a high number of subsequent loops involving design change, simulation execution, and output post-processing and results inspection. This workflow often requires interacting with the simulator sensors, actuators and dynamic models, and it's not limited to the on-board algorithms domain.

The testing and performance assessment, requires a more constraining workflow, managing large data sets, and with the ability of executing and storing in an efficient and orderly way different set of simulation cases, including Monte Carlo runs.

Both the rapid design loop and the organized case runs execution involve interacting with the Simulator datasets and models, with the AOCS algorithms and their associated database, and with the AOCS interfaces and datatypes definition. All these elements are part of the project development, and are included in the design evolution.

The above considerations have been the main driver for the shaping of the EDS infrastructure and use.

The EDS is composed of the following high level components: a Simulink architecture model, a MATLAB infrastructure to manage the simulations execution and associated data, and a set of utilities which aim at improving the efficiency of the workflow, comprising of all the ancillary operations needed for AOCS design and test. The EDS architecture, even though tailored in this case for its use in the EUCLID AOCS design, is built in such a way to allow reuse for other purposes minimizing the refurbishment effort.

5.1. Simulations execution

Three execution modes are allowed in the simulator: Design run, Systematic run, and Monte Carlo.

The Design mode allows the design engineer to rapidly run simulation cases and iterate on the algorithms design and simulator parameters, minimizing the set-up effort. The aim here was to configure the simulation with a single MATLAB command. Source data can be loaded either from default data sets or from specific input cases (see Section 5.2 for data management details). No complex post-processing is foreseen for this execution mode.

The Systematic run mode exploits the same data management system as the Design mode, but adds a wrapper for automatic execution of predefined cases in terms of Simulator and AOCS data sets.

The Monte Carlo mode exploits the systematic run mode, adding an additional engine for data perturbation and multiple shots management. The Monte Carlo engine allows a flexible definition of the variables to be perturbed, and of what distribution shall be employed; customized perturbation functions can be associated to any given variable with a plug-and-play philosophy that requires no modification by the user of the Monte Carlo engine, thanks to a standard function header definition.

Simulation post-processing is based on custom user-defined MATLAB functions, tailored to fit the needs of the simulations executed. Such custom functions are anyway expected to locate, manipulate and store data according to the common standard architecture employed in the EDS.

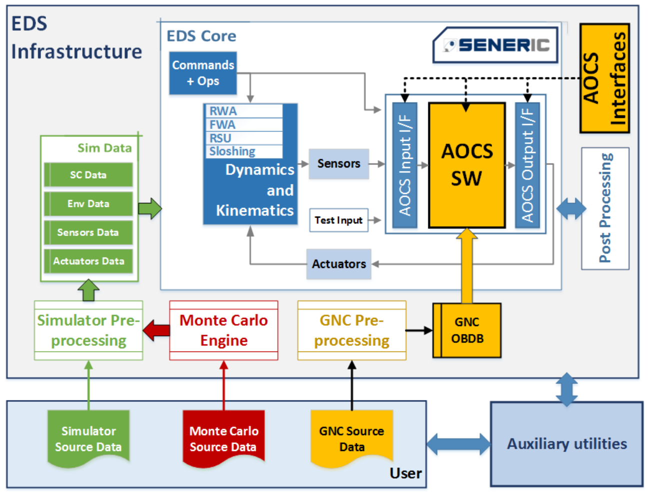

Figure 3: EDS functional architecture

The elements composing the EDS functional architecture are presented in Figure 3. It is important to highlight that all of the AOCS components (software algorithms, interfaces, on-board database) will be autocoded into the actual flight software. As a consequence, all AOCS parts are well separated from the Simulator architecture, even though functionally embedded into it.

5.2. Simulation data management

Data management is based on the standard concepts of source data base, pre-processing, simulation database.

The source data base, as expected, contains all the data that the user can directly manipulate to set-up a simulation case. The pre-processing step transforms such user-friendly parameters to the often less intuitive data needed by the simulator core. As a simple example, misalignments error can be defined as conic errors around a given direction, and are then pre-processed in reference frame transformations for their implementation in the simulation.

In line with the duality previously discussed between the needs for design and performance assessment, data management relies on two source data-base initialization modes: m-files, and xml databases.

The use of m-files is retained to allow quick set-up and debug of the simulator, while xml databases allow setting up complex simulation cases in a more structured and controlled fashion. Xml databases and m-files retain full compatibility: xml datasets can be generated directly from m-files, making use of specific utility functions included in the EDS pack, which aim at avoiding lengthy set-up of complex xml files from scratch. This functionality was thought specifically for the design phase, where both the AOCS/GNC and the EDS are under continuous development, and the data structures are not consolidated enough to result in frozen xml files. In such conditions the development typically advances making use of MATLAB native files, and the changes can be quickly reverted to new xml templates.

5.3. AOCS-AUTO Interfaces

Interfaces are a key element of the AOCS, and their definition plays a relevant role in the design and development process.

The AOCS SW modules will have logic interfaces with the sensor and actuators through the AASW-MAN (manually coded AOCS software), and also with the AASW-MAN directly. From a software point of view all the interfaces will be with the AASW-MAN, so the AASW-MAN will call a function that it is part of AASW-AUTO (auto coded AOCS software) through this function will be transferred the IN/OUT parameters. The transferred information will consists of AOCS Sensors (input parameters), AOCS Actuators (output parameters), input data failure detection reporting, unit configurations, on-board database parameters, GNC states, internal parameters to be provided as house-keeping telemetry, failures detected during AOCS-AUTO (Autocoded AOCS code) software execution.

A strong data typing approach is employed in the definition of the AOCS modes Simulink models. Such approach relies on the following elements:

- Input/Output AOCS-AUTO data are organized into Simulink Buses, analogous to C data structures
- The AOCS interfaces are defined in terms of such data buses and their elements, with their data types and dimensions
- The use of Simulink require existing Bus Object for the definition of the models input and output ports.
- The Bus Objects are autocoded into C-code type definitions, which constitute part of the actual flight software
- The interfaces must be iterated and documented, and will be needed by the AOCS-MAN SW and by the ESE to embed the AOCS-AUTO code.

Taking into account all the above, the EDS was provided with a utility that allows generating the interfaces making use of a dedicated Simulink model. The workflow employed is the following:

- The design engineer defines interface signals, data types and sizes in the mentioned dedicated Simulink model
- A set of custom utilities generates and stores the required Simulink Bus Objects deriving from the interfaces definition
- The blocks building the interfaces signals are stored into libraries, available for the EDS.
- At EDS level, the sensors and actuators data, unit configurations, simulated commands are employed by the interfaces library blocks to build the signals sent to the AOCS modes models during simulations
- The defined interfaces are reported in automatically generated Excel document

The aim of such infrastructure and workflow is to rigidly link the actual interfaces specified for the AOCS modes, the simulated signals sent to the AOCS at simulation level, the C-code objects ultimately generated, and the associated documentation. Having all elements of the chain is a feature thought useful for the minimization of errors, and for limiting the effort associated with the interfaces definition iterations.

6. AOCS SOFTWARE DEVELOPMENT

The AOCS is being developed according to the Model-Based Design (MBD) philosophy. The use of this methodology introduces certain differences from a traditional (manual code) software development approach.

The MBD is a model centric approach where except for the initial set of requirements the artefacts that represent the software, interfaces, architecture and tests will be mainly models. This design approach relies for the EUCLID AOCS on the MathWorks framework, MATLAB/Simulink environment and the code generation toolboxes.

In this frame, the system model is at the centre of the development process, and the model evolves from requirements development, through design, implementation, and testing. The model is an executable specification that is continually refined throughout the development process. During model development, simulation show whether the model works correctly.

The AOCS algorithms will be part of the model to be included in the AASW (AOCS Application Software); such algorithm models will eventually be developed as generated code on the embedded processor running together with the AASW -MAN and CASW (CDMU Application Software). The code produced by the models will be verified for confirmation of the results obtained at the model level.

MATLAB/Simulink is being used during most of the software lifecycle in order to have a unique and integrated environment in all the major aspects of the software development. Simulink is able to link models with requirements (with the SL Verification & Validation Tool), and the Coder tools generate C code with comments and tags that trace with the model and the requirements.

The development activities and in particular the procedure, rules and guidelines developed in the frame of the EUCLID AOCS project are focused on the proper use of MATLAB/Simulink and their software development tools.

6.1. Life Cycle Model Overview

As the rest of the AASW, the Model-Based design development and validation plan follows the evolutionary model. This is an iterative approach where progressive versions of the model/software are foreseen. For each version an iteration occurs and a model validation is foreseen.

As for the AASW-MAN, the autocoded software will follow the same versioning approach. Consequently, within this evolutionary development process the model evolves as a whole from initial versions where only the interfaces (blocks "frames") are set to detailed models where initial architectural blocks and their functionality are further elaborated and divided into small parts. For each development phase the complete chain can be exercised, so each model version will contribute to the RB and corresponded code version could be generated and tested.

6.2. Model Design and Implementation

The model will be developed using Simulink® block diagrams, and Embedded MATLAB™ code.

During the design process the model will evolve based on simulation loops performed in the EDS environment. During these simulations the model will be compiled and checked to verify it is well specified and complete. The simulation results obtained will be used to refine the model. Simulations performed during this phase on the EDS are equivalent to test cases, which will be the start point to develop the validation tests.

For each mode, a Simulink model will be created so its design will start by creating a root class representing the overall mode, and then the model is decomposed into smaller pieces according to the defined architecture and following the autocoding methodology.

This decomposition should be interpreted as a high-level identification of functionalities and assignment to C functions during the autocoding process. One of the steps that is considered in this part is the identification and analysis of the Simulink subsystems that constitute the model, and the corresponding mapping to the C-code function.

During this phase the model has to be pre-validated and the test cases are generated. Simulink tools will be used to assist the test cases generation based on the coverage criteria. The Model Coverage Tool will be used to determine the level of coverage achieved from the test cases used during simulation.

The unit and integration tests will be detailed in the respective test plans. They will follow a bottom-up approach starting from the definition of "unit" blocks, designing the unit test and then performing the unit integration. According to the defined architecture the control designers and the software team will determine how the model is partitioned into lower level units as a starting point of the unit and integration test definition.

The unit and integration test will be automatically executed in the MATLAB/Simulink environment. A subset of these tests will be used in the next stages to perform regression tests with next model versions.

7. 10. REFERENCES

[1] Llorente-Martínez, J. S., Zorita, D., Agenjo, A., Cazorla, C., O'Dwyer, A., Del Cura, J. M., "Planck Acms: Autonomous and Precise Control for Slowly Spinning Satellite", *6th International ESA Conference on Guidance, Navigation and Control Systems* 2006, CDROM. id.40.1., Loutraki, Greece, October 2005.

[2] D. Gherardi, V. Marco, R. Sanchez, A. Caramagno, L. F. Peñin, M. Kerr, J. A. Béjar, E. Zaccagnino, J-P. Preaud, "IXV GNC Subsystem Design And Performances", *8th International ESA Conference on GNC Systems*, Karlovy Vary, Czech Republic, June 2011

[3] J.S. Llorente, A. Agenjo, C. Carrascosa, C. de Negueruela, A. Mestreau-Garreau, A. Cropp, A. Santovincenzo, "PROBA-3: Precise formation flying demonstration mission", *Acta Astronautica*, Volume 82, Issue 1, Pages 38–46, 6th International Workshop on Satellite Constellation and Formation Flying, January 2013

[4] L. Tarabini Castellani, A. Paoletti, A. Agenjo, J. Peyrard, A. Cropp, R. Atori, "PROBA-3 Precise Formation Flying System: Design & Development Tools and Methods for FFS Implementation in a Technology Demonstration Mission.*", ICATT4*, ESAC (Spain). 05/2010

[5] ECSS Secretariat, "Stars sensors terminology and performance specification", *ECSS*, ESA-ESTEC Requirements & Standards Division ECSS-E-ST-60-20C Rev. 1, 15/11/2008

[6] J. Eggert et al, "Pointing Error Engineering Tool PEET Software User Manual", *Astos Solutions GmbH*, ASTOS-PEET-SUM-001, 26/07/2013