openSF: A GENERIC FRAMEWORK FOR END-TO-END MISSION PERFORMANCE SIMULATIONS

Ricardo Moyano [(1)] , **Enrique del Pozo** [(2)]

[(1)] *DEIMOS Space S.L.U.*
Ronda de Poniente, 19
Edificio Fiteni VI, portal 2
28760 Tres Cantos (Madrid) – SPAIN
Emails: ricardo.moyano@deimos-space.com

[(2)] *DEIMOS Space S.L.U.*
Ronda de Poniente, 19
Edificio Fiteni VI, portal 2
28760 Tres Cantos (Madrid) – SPAIN
Emails: enrique.delpozo@deimos-space.com

INTRODUCTION

In SESP 2008, DEIMOS Space presented openSF as a generic end-to-end (E2E) simulation tool able to reproduce all the significant processes and steps that impact the mission performance obtaining simulated final data products [1]. It provided end-to-end simulation capabilities that allow assessment of the science and engineering goals with respect to specific mission requirements.

The intention in 2010 is to present why openSF has been selected by ESA to become the standard software infrastructure for the development of the End-to-End Simulators (E2ES) for the future Earth Explorer missions, including a clear description of the different evolutions of the tool and the added capabilities needed to cope with the EE mission requirements (BIOMASS, COREH20 and PREMIER).

HISTORY

DEIMOS Space has been working in several contracts with ESA in the development of End-to-End simulators for the last years. As a result of these projects, DEIMOS took the initiative of extracting the simulation framework, initially linked to mission specific projects, and turn it into a generic tool, named the open Simulation Framework (openSF [2]).

The origin of openSF can be found in the EarthCARE Mission Performance end to end simulator (ECSIM [3]) where, among other activities, a new simulation framework was developed to admit models from the complete processing chain from the scene to the retrieval stages.

The framework was adapted for its usage in AIPC [5] GERSI [6], or SEPSO [7], although at that stage the framework was still a dedicated customisation of ECSIM, instead of a completely separated product. Furthermore, upon the consolidation of the preliminary version of openSF, other projects starting using it as the simulation framework. It is the case of the Sentinel- 3 O-GPP [8] and O-SPS [9] projects, under the lead of ACRI-ST and TAS-F respectively.

The following diagram shows the main projects involved in the creation of openSF:

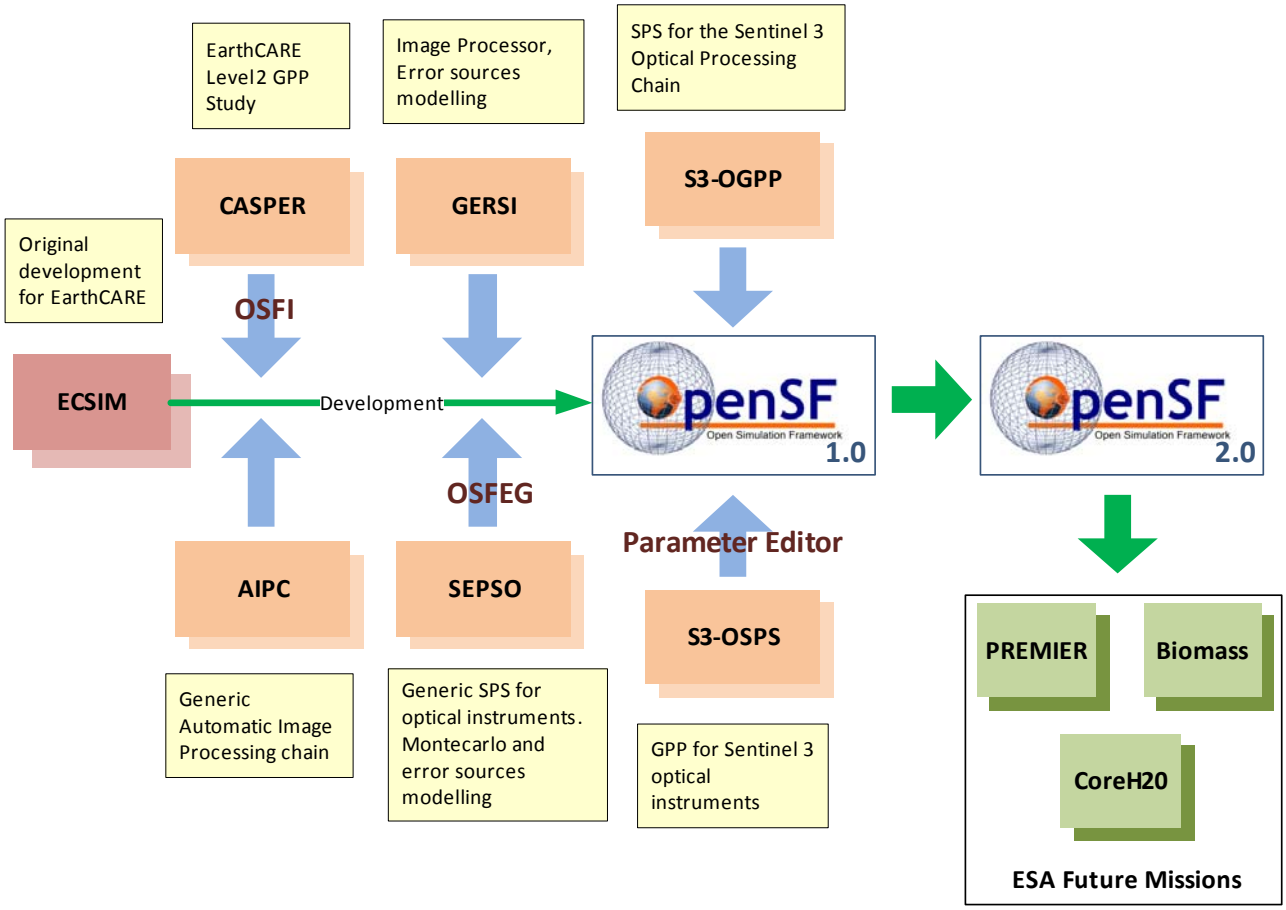

Fig. 1: OpenSF evolution

All of these projects needed specific tailorisations of the framework to meet their requirements. As a result, all new capabilities were presented to ESA, which after a thorough analysis and selection process has lead to the development of openSF version 2.0, including a new set of generic enhancements that shall be available by the end of September 2010.

The improvements extracted from the projects involved in the openSF evolution are the followings; some of them have not been integrated, by the moment, in the core development and consequently not available for version 2.0.

- OSFI libraries: developed in the CASPER project aimed to ease the algorithm integration within openSF. Originally these libraries were implemented in C++ and Fortran 90 but currently there are ports for C, Fortran 77, Matlab and IDL.
- Parameter Editor: user friendly tool aimed to ease the simulation definition stage by setting consistency relationships among input parameters and therefore avoiding potential problems that could arise during the simulation run.
- OSFEG libraries: C++ and Java libraries with the objective to create random error sources over simulation parameters. This software was former designed during GERSI project but officially released in SEPSO end-to-end simulator.
- Performance Monitoring: computer resource monitor implemented in the adapted framework fro the Sentinel 3 O-GPP project.

OPENSF OVERVIEW

All projects where openSF has been used require the definition of simulation chains, composed by a set of executable files named models run in a sequence. A model represents a component of the simulation chain, and it can implement a scientific algorithm, instrument model, data processing component or any desired part of the processing chain.

As shown in Fig. 2 a model is normally fed by one or more input files and produces as output one or more output files, which may represent input data for the subsequent models in the simulation. Additionally, a model is configured to determine its specific behaviour during the simulation, and generates log messages to inform about the progress of its execution.

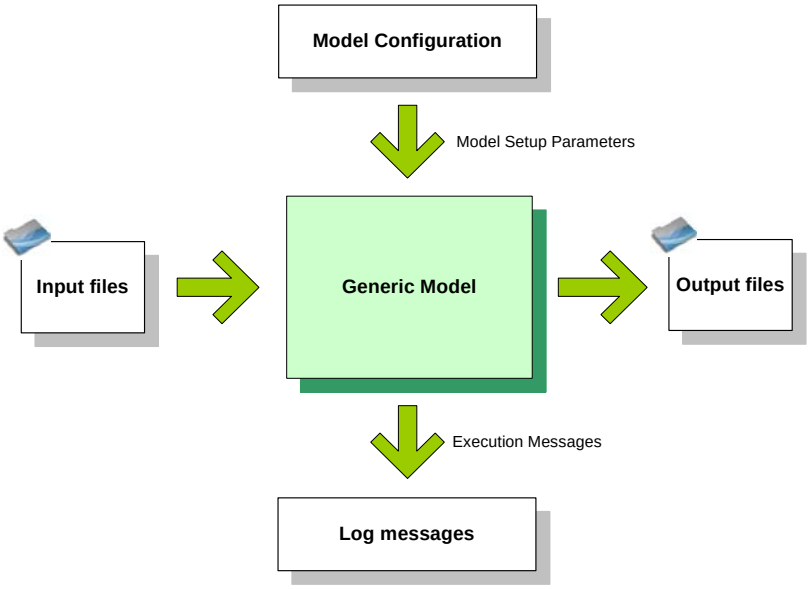

Fig. 2: Generic definition of a module

A simulation is defined by a sequence of models, corresponding to each of the processing stages defined for the simulation. Models share inputs and

An example of a simulation definition is shown below. It is constituted by three processing stages, each one covered by one model.

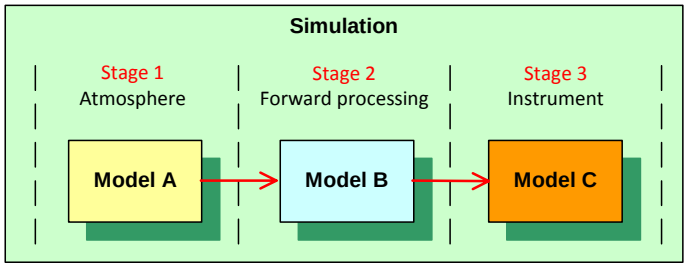

Fig. 3: Example of a simulation

For a simulation to run correctly there must be strict consistency among the outputs generated by a model and the inputs read by the next one. It is crucial that for a model to successfully run, all the input files must be available prior to the moment it is launched by openSF.

OpenSF Features

OpenSF provides the following characteristics

- **open**: openSF uses a flexible licensing scheme that allows integrating any kind of third-party developments.
- **Flexibility**: openSF is flexible enough to adapt to any processing chain, as the definition of the stages of the simulations are part of the framework's functionality.
- **Portability**: openSF framework has been written in Java, ensuring therefore portability. Furthermore, openSF has been tested on Windows, Linux and Mac OS platforms. Hardware and software requirements need to be defined for each installation indicating the precise operating systems as well as the Java versions supported for each case.
- **Expandable**: In addition to the portability, openSF includes a set of libraries to support the model integration in the framework. These libraries provide the interface between modules and the framework, and are written in C++ (with wrappers for ANSI C and Fortran 90) and Fortran 90. The inclusion of new libraries for the integration of IDL, Matlab and Fortran 77 models are part of openSF version 2.
- **User-friendly**: openSF provides a graphical Human Computer Interface and command line interface for interacting with the system. The HMI shall be upgraded with the Parameter Editor look and feel ([13]).
- **Configurable**: openSF allows users to alter its behaviour and structure by implementing well-defined XML interfaces.
- **Reliable**: openSF intercepts and stores model output logging and implements error handling procedures. These functions are included in dedicated libraries that are provided as part of the tool's deployment.
- **Powerful**: users can execute simulations in batch or iterating through different parameters. Users can interrupt the execution at different stages an resume them, or set breakpoints among the participating models in the simulation.
- **Multiple repository/processing chain capability**: openSF version 1 only admitted the definition of one simulation chain. It has been agreed to allow the possibility to define more than one processing chain within the same openSF instance so that it would be possible to hold simulations for more than one mission, or to define variations of the processing chain for one mission. For example, the Ground Processor for EarthCARE constitutes a processing chain on its won, but it can also be seen as a processing stage for the EarthCARE End-to-End simulator.

Model Integration

openSF lets users to integrate a diverse set of models that form the building blocks of a simulation process. To integrate an external model into the framework, models need to fulfil the interface requirements demanded by openSF.

The Open Simulation Framework Integration Libraries (OSFI) have been generated on order to ease the integration of models into the open SF. They contain a set of routines with a well-defined public interface (and therefore hiding the implementation details) that permits developers to:

- Read configuration files
- Read the list of input files to be ingested into the model
- Read the list of output files the model must generate
- Write log messages (with different categorisation) so that openSF can print them in the log message area of the simulation execution window
- Report the progress of the model execution

openSF offers native support for model integration in the three most common scientific programming languages: ANSI C, C++, Fortran 90 and has been recently extended to support Matlab, IDL and Fortran 77. This means that an OSFI library exists for each language, and therefore, models using the API are ready to be integrated into a simulation chain within the framework.

OPENSF ADDITIONS

OpenSF has suffered as series of enhancements during the last two years that makes the framework suitable to support the execution of a large range of diverse processing chains. The most relevant ones are presented below.

Open Simulation Framework Error Generation Libraries

In the frame of concept and feasibility studies for the Earth Observation (EO) activities, mission performance in terms

of final data products needs to be predicted by means of so-called end-to-end (E2E) simulators. For sake of performing sensitivity analysis of the satellite arises the necessity of creating random error sources that modify the algorithms within an E2E simulation chain.

Under this scenario appears the goal of performing a statistical analysis of the E2E simulator driven by the errors and perturbations present in the parameters involved in a simulation chain. The Open Simulation Framework Error Generation Libraries (OSFEG) will be used as a tool to ease the mathematical modeling of a perturbation within statistical analysis scenarios. OSFEG offers to developers a well-documented interface to ease the modeling and generation of a perturbation over desired parameters. The libraries provide an error-modeling interface based on a XML file definition and its correspondent implementation in C++.

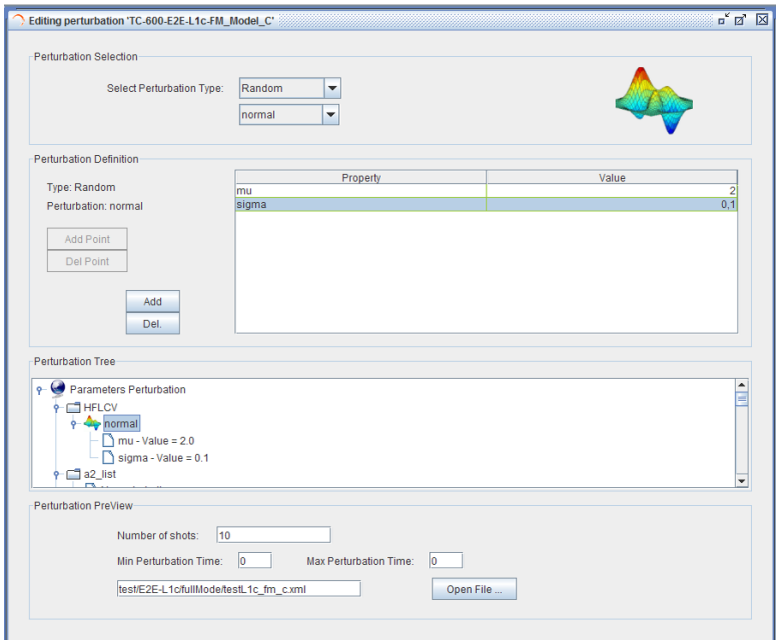

Fig. 4: OSFEG management interface

Parameter Editor

A parameter in openSF is a constant whose value characterizes a given particularity of a simulation algorithm. Parameters are user-configurable, they are fixed before launching a model and can change the behavior of a model/algorithm within and E2E simulation during the run.

DEIMOS is currently developing, in the frame of the Sentinel-3 Optical System Performance Simulator (S3-OSPS [9]), a parameter management tool used for creating and editing parameters as well as setting relationships between them with the objective to ensure the consistency of the models' configuration.

This tool is born from the necessity of performing a simulation consistency checking before running the simulation chain. Analyzing the parameter consistency before executing end-to-end simulator minimizes the potential problems that could arise during the run. This aspect becomes relevant when the simulator is related to on-board and operational software where resources and time consumption are huge.

OpenSF parameter management system is composed of two software modules, a *Parameter Rule Editor* and a *Parameter Editor*. The first tool is used offline, before the simulation definition, and consists in a simple grammar and a graphical editor allowing the user to define a set of rules. These rules will be used to validate the parameters entered by the user in the simulation definition stage. The grammar used for defining constraints and relationships between parameters has been designed using a XML intuitive syntax.

In the *Parameter Editor* users are able to:

- Open a rules file in order to validate a set of parameters with it.
- Edit the parameter rules file.
- Check the errors through an information log panel.
- Visualize all the parameters of a configuration file with an intuitive tree view.
- Create and delete parameters.
- Edit the values of a parameter.
- Save the configuration file.

The figure below shows the *Parameter Editor* interface making use of the future Java Swing look and feel that is planned to be the openSF default interface.

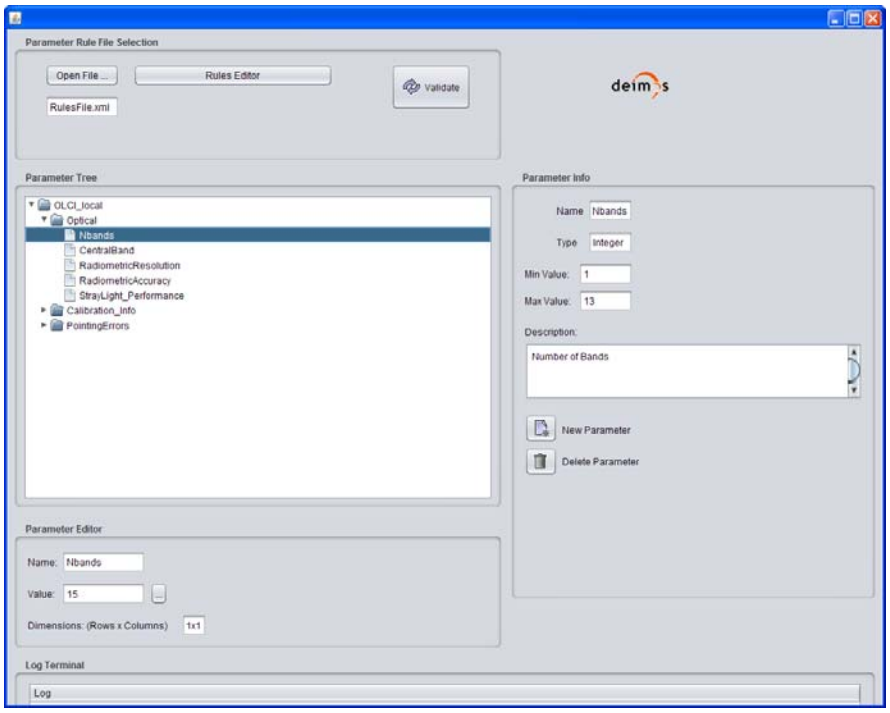

Fig. 5: Parameter Editor interface

Performance Monitoring

Nominally openSF monitors the simulation time of the running models. However, this kind of computer resource control is good enough for small studies but it becomes insufficient in near-operational environments. This is the case of Sentinel 3 Optical Generic Processor Prototype (S3-OGPP [8]) where memory usage and processor load is a critical issue.
In the frame of this project a simple interface for visualizing the more critical computer resources has been implemented but it is under study the monitoring of other indicators like free disk space, system processes etc. Current resources visualized are:

- Memory Consumption (RAM usage)
- Processing resources (CPU load)

Due to the hard OS dependency of these performance indicators, the information showed in the framework could be wrong or just approximated. Fig. 6 shows the graphical interface for RAM memory monitoring implemented in S3-OGPP openSF tailorisation.

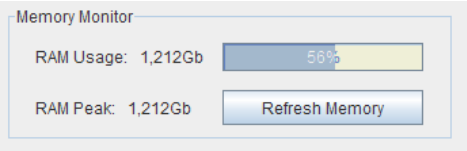

Fig. 6: openSF RAM Monitor

OPENSF USE CASES

Statistical End-to-End Performance Simulator for Optical Imaging Sensors (SEPSO)

SEPSO is a study whose purpose is to define and prototype a generic Statistical End-to-end Performance Simulator for Optical passive imaging sensors in the spectral range between 0.3 and 15 µm from Earth Observation and Space Science missions. SEPSO aims at simulating a single-pixel for all product levels (0, 1a, 1b, 1c, 2a and 2b) and with its associated accuracy and confidence level.

SEPSO simulator is being used for validating Sentinel 2 test data. Fig. 7 shows the correspondent end-to-end simulation chain as well as details about the statistical engine implementation.

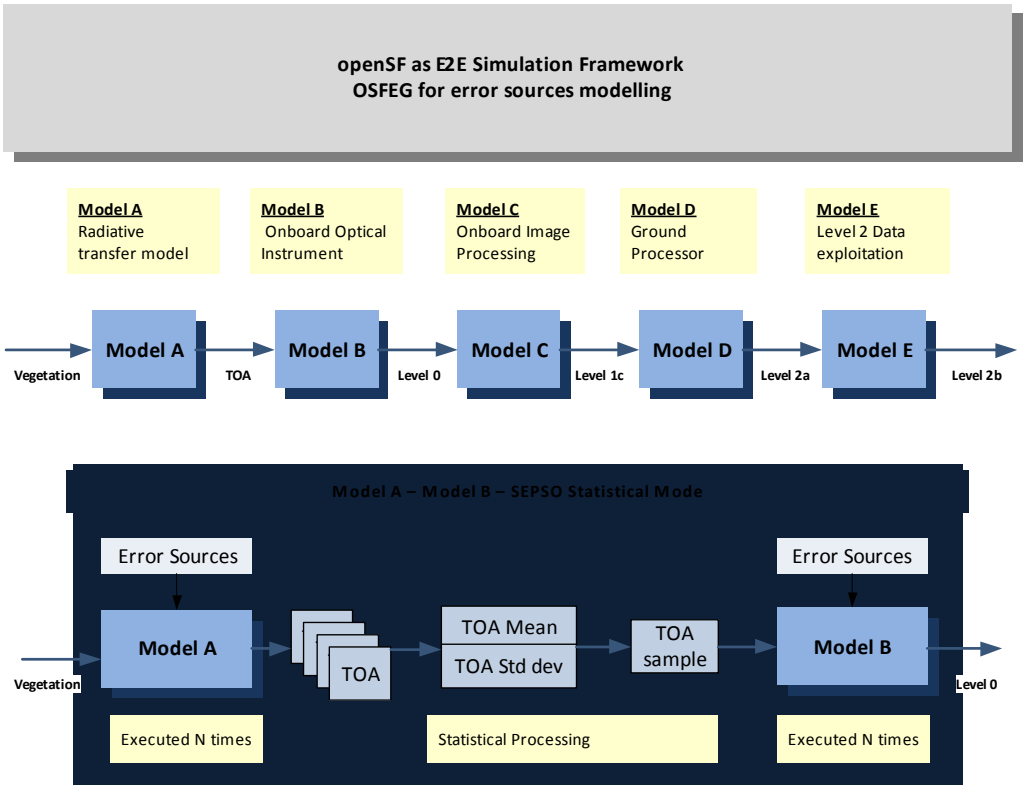

Fig. 7: Diagram of SEPSO processing chain

Sentinel 3 Optical Generic Processor Prototype (S3-OGPP)

The objective of the GMES Sentinel-3 Level 0 / Level 1 Optical Prototype (S3 O-GPP) Development is to design, implement, verify, and maintain the L0 and L1 algorithms related to ESA Sentinel-3 optical missions, within the context of Global Monitoring for Environment and Security (GMES).

The O-GPP shall be able to generate single instruments Level 0 and Level 1b products for OLCI and SLSTR, as well as the synergetic Level 1c Vegetation product, gathering data from both instruments. It shall be able to ingest raw data

from the two Sentinel-3 instruments as well as the Navigation and Attitude raw data (NAVATT packets) provided by the platform. It shall provide intermediate data at several steps of the involved processing in support to its own verification process and to mission performance assessment.

The S3 O-GPP will be a tool providing:

- Support the development of the OLCI and SLSTR instruments
- Support the validation of the Level 0 and Level 1 processors of the instruments OLCI and SLST
- Support the validation of the Level 1c, understood as the combination of L1b streams from instruments.
- Support the PDGS (Payload data ground segment) development and testing

All these facts make the S3-OGPP a really challenging project where openSF capabilities will be tested in a near operational environment. The simulation chain correspondent to S3-OGPP is illustrated below.

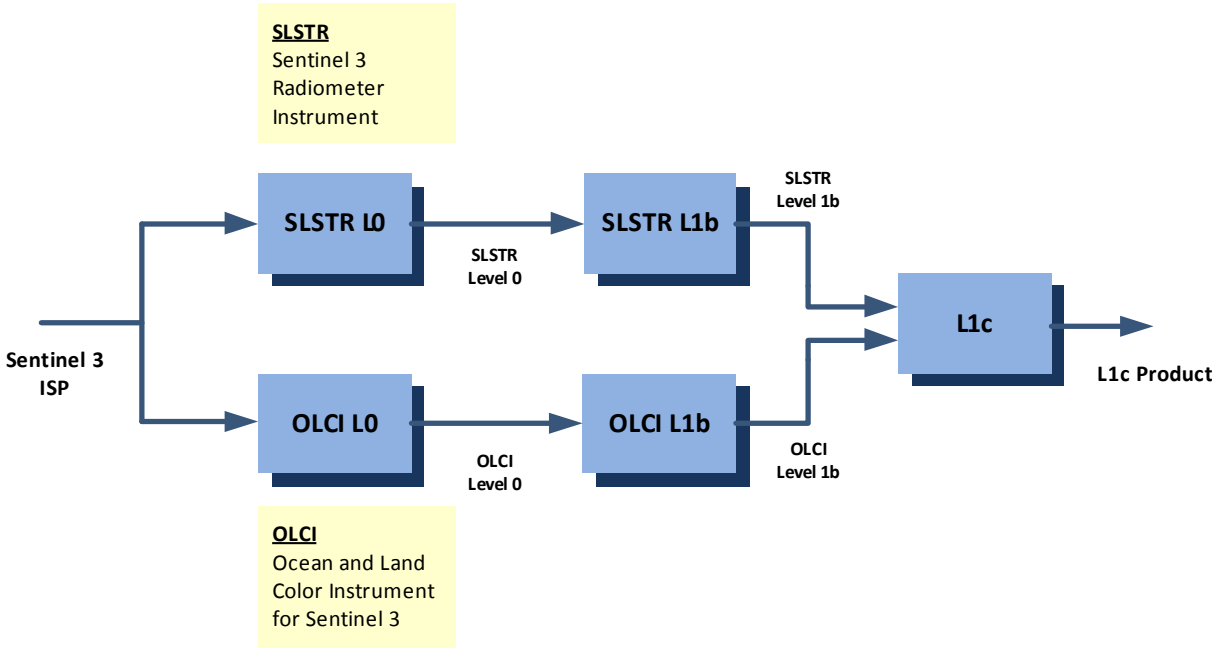

Fig. 8: S3-OGPP Processing Chain

REFERENCES

[1] Ramos, J.J, Acarreta, J.R., Moyano, R., Franco, R.: "openSF: A Generic Framework For End-to-end Mission Performance Simulations", SESP 2008
[2] OpenSF - ESTEC Contract No. 22852/09INUFF
[3] ECSIM - ESTEC Contract No. 20003/065/NL
[4] CASPER - ESTEC Contract No. 20880/07/NL7EL
[5] AIPC - ESTEC Contract No. 21251/07/NL/HE
[6] GERSI - ESTEC Contract No. 20226/06/NL/HE
[7] SEPSO - ESTEC Contract No. 21445/08/NL/NE
[8] S3-OGPP - ACRI_ST Contract No. S3-CO-ACR-SY-00645
[9] S3-OSPS - Thales Alenia Space Contract No. 1550001670
[10] CoreH2O EE candidate mission
[11] BIOMASS EE candidate mission
[12] PREMIER EE candidate mission.
[13] Jasper Potts and Peter Zhelezniakov, Nimbus: Making Swing Look Sexy, Java One technical conferences, Sun Microsystems. http://developers.sun.com/learning/javaoneonline/sessions/2009/pdf/TS-5579.pdf
[14] Jarle G. Hulaas and Dimitri Kalas, "Monitoring of resource consumption in Java-based application servers"
[15] M. Tim Jones, "GNU/Linux Application Programming" Charles River Media; 1 edition (February 2, 2005)