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

Ramos, J.J. [(1)], Acarreta, J.R. [(1)], Franco, R. [(2)], Moyano, R. [(1)].

[(1)]Deimos Space, S.L.
Ronda de Poniente, 19
Edificio Fiteni VI, portal 2, 2º
28760 Tres Cantos (Madrid), Spain
Email: jose-julio.ramos@deimos-space.com
[(2)]ESA-ESTEC
Keplerlaan 1
2201 AG Noordwijk, The Netherlands
Email: rafaella.franco@esa.int

INTRODUCTION

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.
A specific mission E2E simulator is able to reproduce all significant processes and steps that impact the mission performance and gets simulated final data products.
OpenSF started as a result of the development of the E2E simulator for EarthCARE, ECSIM. This achievement has been obtained thanks to the joint work with ESA, covering their requirements and suggestions and having the collaboration of the KNMI (Royal Netherlands Meteorological Institute).
OpenSF is a generic simulation framework product being developed by Deimos Space, S.L. aimed to cope with these major goals. It provides end-to-end simulation capabilities that allow assessment of the science and engineering goals with respect to the mission requirements.
OpenSF represents a simple, robust and extremely user-friendly tool easily adaptable to cope with any mission requirements.

CONCEPT

OpenSF is an independent framework providing added-value functionalities to scientific simulations, substituting complex and rigid scripts, Simulink projects or similar developments.
Scientific models and product exploitation tools can be plugged in the system platform with ease using a well-defined integration process.

SIMULATION ENVIRONMENT

OpenSF identifies and manages three aspects of a simulation environment:

- Static structure
- Dynamic flow
- Data creation and exploitation

Static structure

Simulations are constituted by a hierarchical structure of building blocks, the independent algorithms (models), and the relationships between them (interfaces or “file descriptors”). Diagrams in Fig. 1 and Fig. 2 show simple representation of the following concepts:

- Models – executable entities that will perform the scientific or engineer calculations;
- File descriptors – to define which models can be linked in which way;
- Stages / families – to group models conceptually or by goal;
- Simulations – sequences of models linked by the descriptors.

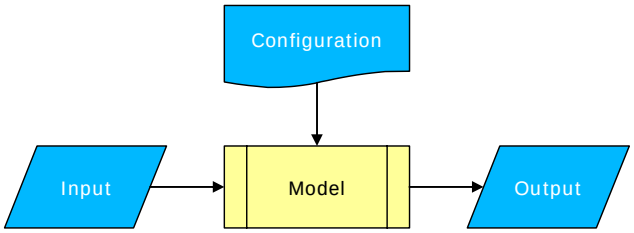

Fig. 1. Model definition including input/output and configuration files.

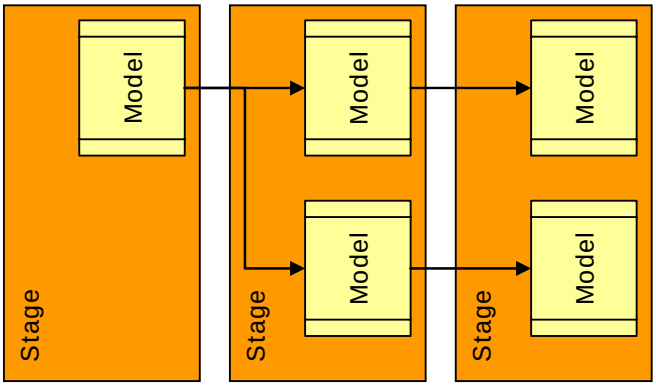

Fig. 2. Simulation definition

This modular architecture allows reusing and substituting any part of the simulation. For instance, model developers can experiment with different implementations of the same algorithm.

Simple and complex simulation chains can be defined. Chains can be made by one sole model or any concatenation of models. Currently, the only limitation for defining complex simulation chains is to avoid “closed loops”, that is, one data path cannot contain the same model twice.

Users can define different partitions of a bigger simulation structure as desired. It is possible to focus only in certain branches (e.g. to simulate instrument losses), or stages (e.g. to simulate the ground segment processors). Fig. 3 shows a valid ECSIM simulation chain where it can be seen multiple branches as it simulates four different instruments with synergetic retrievals.

Moreover, users can perform some operations to a static structure to define more complex scenarios. For example, two or more simulation structures can be scheduled for executing in batch mode, in steps, one after another, without human interaction.

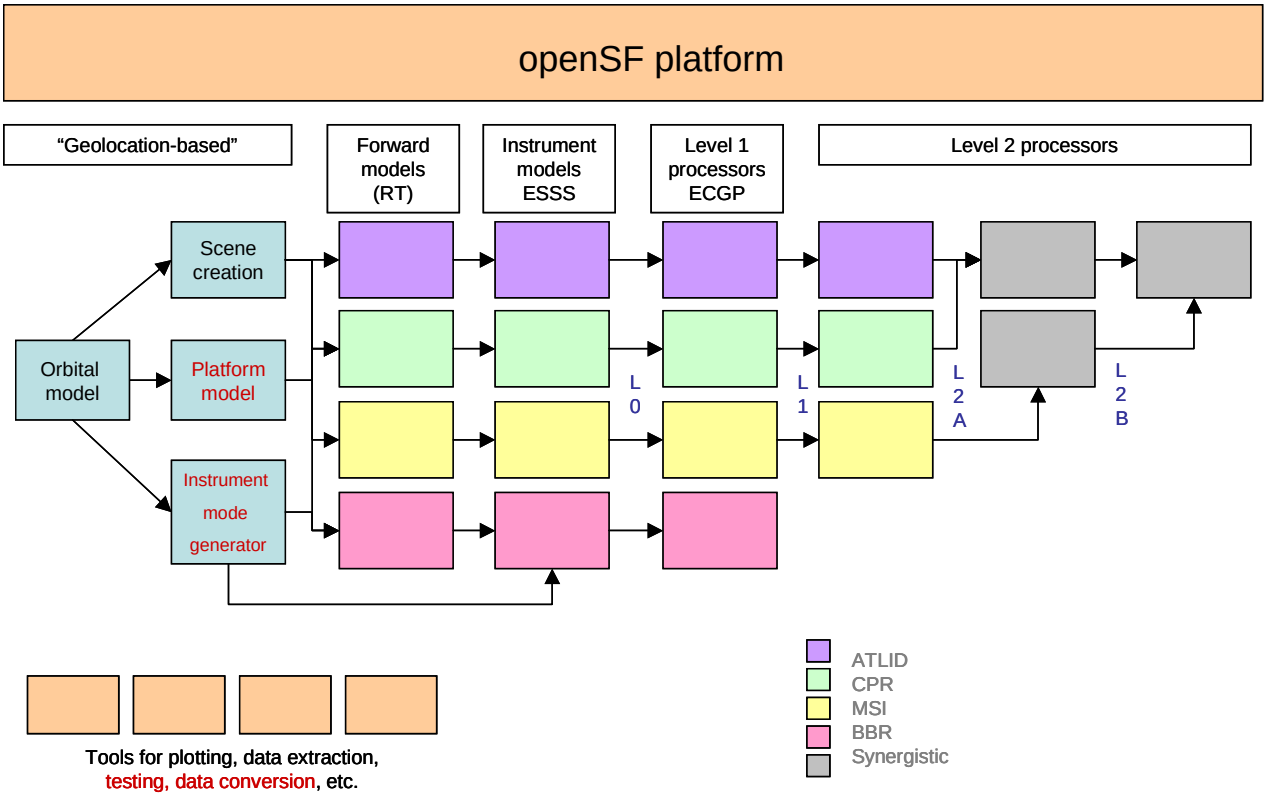

Fig. 3. Possible ECSIM simulation chain

Dynamic flow

Once the static structure of the simulation is defined, openSF users can start with the actual execution process. The openSF system controls this simulation process, activating every model executable in the precise time and also the data flow, providing these models with needed input and configuration files and storing in a dedicated folder the generated output.

Data flow is controlled by openSF but users must provide the location of some initial input and auxiliary files the simulation definitions. Intermediate products are also controlled and stored in the same dedicated placeholder.

Users can alter the value of configuration parameters provided to models via their configuration files. The system proposes a set of default parameters but users can easily use their own. A powerful functionality lets one user to try with different parameter values, for example, iterating through an arithmetic sequence of floating point numbers. OpenSF helps to construct replicated chains, executing some models with different configuration profiles.

OpenSF performs a validation check prior to the execution to ensure that every needed input file is available (meaning that the user has defined its current location) or is pending of generation by a previous model.

Afterwards, openSF creates necessary configuration files, places them in the dedicated folder and start the execution of the models, one after another, providing them with needed files.

Model executions are started in a separate thread for ensuring independency of the GUI operations and the simulation process and using different memory spaces. This thread is monitored by openSF to intercepting its output and creating a log file. This logging can store information, debug, warning, errors and progress messages to show current simulation status. OpenSF also performs a secure error handling since intercepts unexpected crashes, stopping the system execution.

Once a simulation execution reaches its end, whether it was successful or had any kind of fatal error, it is stored in a dedicated folder in the file system and as an element of the execution archive. Then, users can access to every archived execution to see a final report, to access to its log or every product file involved in it. At the end, users can repeat successful executions or re-start/resume failed ones.

Data creation and exploitation

Another aspect of the simulation environment is the management of needed and generated files, whether there are considered as final or auxiliary products. OpenSF accepts every kind of file thanks to a plug-in mechanism. This mechanism lets to include specific external tools or GUIs for data handling:

- Managing data and define the input contents.
- Examining the output with tools for plotting, browsing, importing and exporting products, etc.

Users can associate a certain tool to a certain file extension (as many operating systems already do). The system also provides a view of the local file-system for quickly access to product files and tools.

DOCUMENTATION

OpenSF comes with a complete set of documents to give third-party developers full information of its characteristics:

- Complete testing – system and unit tests ensuring system's quality;
- Interface Control Document – for model developers, declaring integration requirements;
- Software User Manual – for end-users, giving an operations and reference manual for the HMI and command-line operations.

CHARACTERISTICS

Open

OpenSF uses a flexible licensing scheme that allows integrating it in any kind of third-party developments. Core library is distributed under the GNU Lesser General Public License or LGPL that is a free software license published by the Free Software Foundation. LGPL places copy-left restrictions on the program itself but does not apply these restrictions to other software that merely links with the program.

Portable

OpenSF is a multi-platform system with minimal hardware and software requirements. Technologies used are:

- Java(TM) 2 Runtime Environment, Standard Edition 1.5 or superior
- MySQL client and server 5 or superior and MySQL connector/J 5.0.4
- XML technologies

It works in PC and Mac machines, both 32 and 64 bits versions. It is available in a wide variety of Operating Systems:

- Windows™ O.S. family.

- Linux distributions: RedHat, openSUSE, Debian, Ubuntu, etc.
- Mac™ O.S. family.

OpenSF infrastructure does not penalize in excess the memory and speed specifications of a scientific model. The only functionality that could slow down the execution of a model within the platform is the number of log messages shown. OpenSF intercepts, parses and stores them so these operations could be a major source of slowness in an execution. An analysis of the temporal penalty (shown in Fig. 4) demonstrates that it not penalizes the execution in excess.

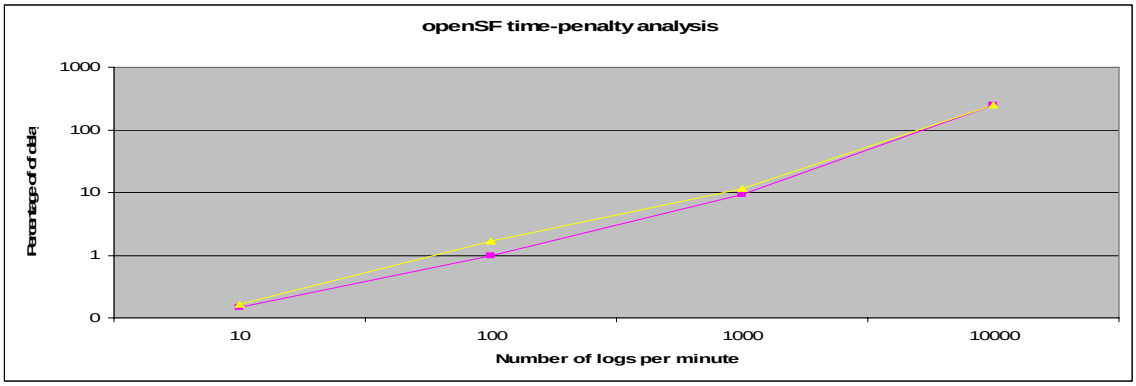

Fig. 4. OpenSF time-penalty analysis

This graph shows two series, the one in fuchsia shows the time penalty observed with a simple model running in an Operating System shell. It can be seen that a model outputting one hundred logs per minute will slow its performance by one percent. Series in yellow shows the time-penalty for models running with-in the openSF system. It can be seen that it follows the same tendency as in the other scenario, with an added offset of around one percent.

User-friendly

OpenSF provides a graphical and a command-line Human-Computer Interface for interacting with the system.

Fig. 5 shows the graphical, Multiple-Document Interface. This presentation pattern allows to contain some internal frames and at the left-side hand, a quick-access bar for accessing to every simulation component.

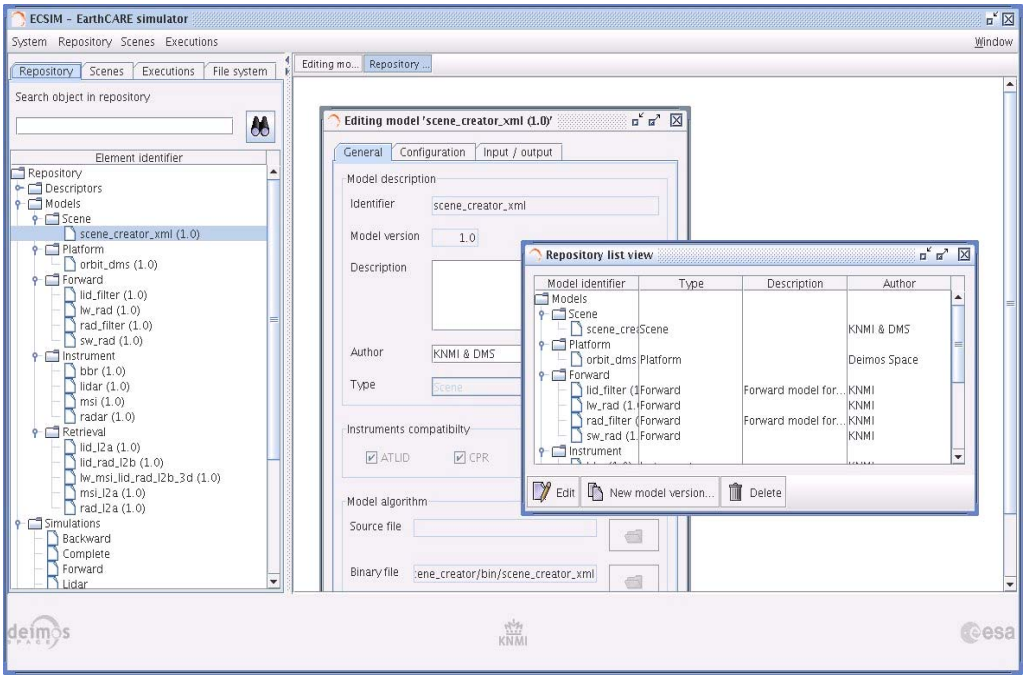

Fig. 5. ECSIM-HMI example, showing multiple internal frames.

Configurable

OpenSF lets users to alter its behavior and structure by using an internal database and implementing well-defined XML interfaces. For instance, XML configuration files specify model parameters and the database supports the simulation structure definition.

Expandable

Users can integrate every kind of executables (no programming language dependency) as models and product tools. Models must comply with a minimal list of integration requirements but product tools do not have any special restriction

The Interface Control Document (ICD) describes the model integration requirements and some development guidelines. These requirements imply a minimal intrusion into code of scientific models covering mainly these four topics:

- Command line arguments – executables must accept a defined list of arguments.
- Error handling – successful executables must return a zero code to the operating system.
- Logging – output messages to be intercepted by openSF must comply with a given format.
- Configuration files – models can accept a specific XML format file if they want users to access and control their parameter values.

Even if those requirements are rather simple and easy to comply, openSF could provide some Integration libraries to ease the model integration process. These libraries come with a complete documentation for developers.

A compatible model can be integrated into the openSF system just by following these steps in the system repository:

1. Create new file input and output file descriptors to declare needed and generated files. Users can also reuse some already-defined interfaces.

2. Create a new model in the system repository declaring its identifier, version, location of the executable, location of the default configuration file and the input and output file descriptors.
3. Now that the model is integrated in the system repository, it can be connected with other models with “compatible” interfaces and form a simulation chain.

Powerful

As presented before, users can execute simulations in batch or iterating through different parameters.
Users can interrupt the execution at different stages and resume or re/start them afterwards.
These functionalities allow performing complete parametrical analysis of a complete system, facilitating the user operations.

Statistical simulations

OpenSF could provide a way to apply statistical methods (Monte Carlo simulations) for system performance analysis.
Some libraries for error definition and generation are being produced. Error sources include:

- Polynomial functions (bias, affine, linear, parabolic, etc.);
- Random distributions (beta, gamma, exponential, normal, etc.);
- Harmonic functions;
- Step functions;
- Sampled with linear, polynomial or sp-line interpolation;
- Plus simple arithmetical operations to combine them.

These libraries will provide drivers for performing Monte Carlo simulations (sketched in Fig. 6) with given scientific models. Once the multiple shots are finished, user would like to characterize the system by analyzing the output files, so some analysis tools are also being produced.
These functionalities imply that there should exist a kind of product harmonization so these analysis tools can ingest them. Importing/exporting operations are also valid options.

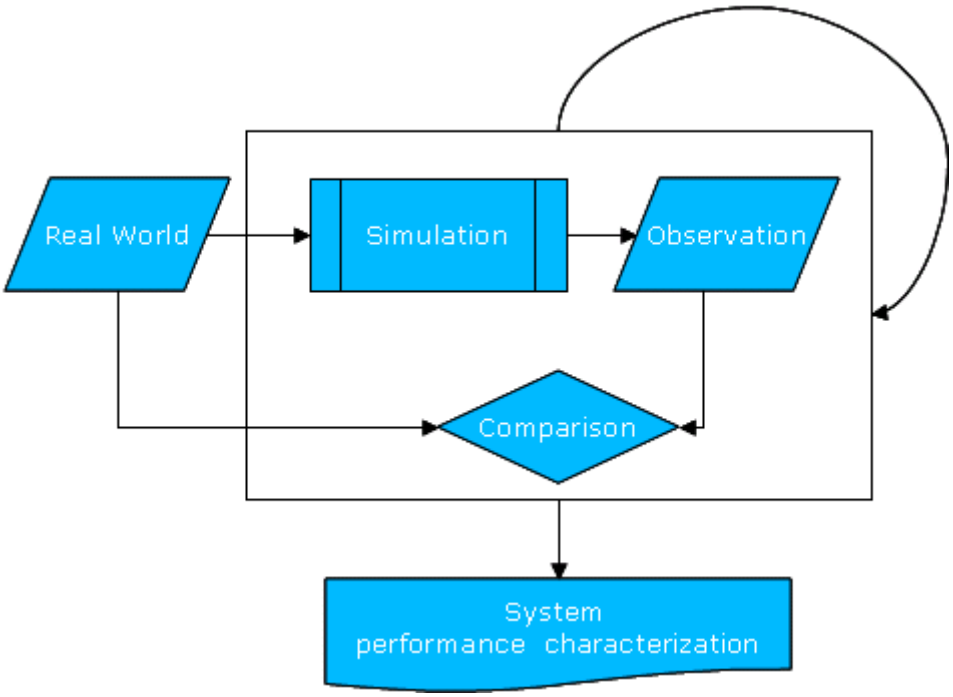

Fig. 6. System performance characterization using Monte Carlo techniques.

APPLICATIONS

OpenSF has been successfully used in some ESA-funded projects, notably:

- ECSIM, an E2E system performance simulator for the EarthCARE mission;
- Processing chain prototypes for Earth-Observation Optical missions (works in progress).

This system can be used in the following domains:

- E2E System Performance Simulators (SPS) - for both parametrical and statistical analysis;
- Ground Processor Prototypes (GPP) – focusing in the retrieval stages;
- Scene generators – for creating specific input data used in other simulation stages.

CONCLUSIONS

OpenSF represents an affordable, simple and generic software simulation infrastructure that can be easily configured and adapted to any space mission with minimal impact on scientific models participating in the simulation.
Core components are already produced but some functionality is still a work in progress.

REFERENCES

ECSIM project - ESA’s contract number 20003/065/NL.

ACKNOWNLEDGMENTS

We would like to thank the contributions of all the ECSIM team members in Deimos Space, S.L., in ESA and in the KNMI. Their suggestions and contributions inspired this work.