Framework for End-to-end Simulators

OPENSF

SYSTEM REQUIREMENTS DOCUMENT

Code	:	OPENSF-DMS-SRD-001
Issue	:	3.2
Date	:	15/01/2014

	Name	Function	Signature
Prepared by	Rui Mestre (DME)	Project Engineer	Rui Mestre
Reviewed by	Ricardo Moyano (DMS)	Review team	
Approved by	Ricardo Moyano (DMS)	Project Manager	
Signatures and approvals on original			

This page intentionally left blank

Document Information

Contract Data	
Contract Number:	22852/09/NL/FF
Contract Issuer:	ESA/ESTEC

Internal Distribution		
Name	Unit	Copies
Ricardo Moyano	DMS	1
Enrique del Pozo	DMS	1
Jose A. González	DMS	1
Rui Mestre	DME	1
Internal Confidentiality Level (DMS-COV-POL05)		
Unclassified □	Restricted ☑	Confidential □

External Distribution		
Name	Organisation	Copies
Raffaella Franco	ESTEC	1
Lavinia Fabrizi	ESTEC	1
Paolo Bensi	ESTEC	1

Archiving	
Word Processor:	MS Word 2000
File Name:	OPENSF-DMS-SRD-001-32_draft.doc

Document Status Log

Issue	Change description	Date	Approved
1.0	First version of the document	23/09/2009	
1.1	Updated including requirements for the openSF Web site access and RID resolution discussed at AR1	15/03/2010	
1.2	New document version including the openSF version 2 capabilities	30/06/2010	
1.3	New version answering the comments generated by ESA on the PDR documentation package.	15/09/2010	
1.4	Updates after AR meeting for openSF v2.0 ❑ Removed table tracing SoW requirements with the openSF version. ❑ System requirements now refer to openSF version instead of this document issue. ❑ Added requirements traceability to openSF Maintenance Proposal [MR01]. ❑ Corrected references of new requirements.	12/11/2010	
2.0	New version including extended capabilities for openSF 2.2: ❑ Parameter Perturbation plug-in (from SEPSO) ❑ Parameter Editor integration ❑ Tool management extension ❑ Check output generation ❑ MATLAB errors inclusion ❑ Import/Export capability ❑ Extended log capabilities ❑ Keyboard shortcuts ❑ GUI Isolation	10/12/2011	
3.0	New version including the specification of openSF V3 capabilities: ❑ Parallelisation ❑ Removal of logs from database ❑ Simplification of the model management chain ❑ Possibility to switch or select a model version before launching a session	18/04/2013	

	□ Possibility to remove intermediate data during the simulation executions □ Support for introducing statistical behaviour at model level □ Bypass/switch-off models □ Copy elements □ Rerun a session from a certain point □ Export capability □ Convert SQL syntax into XML for element definition		
3.1	New version answering the comments generated by ESA on the openSF V3 PDR documentation package: □ Requirements have been reviewed to check their applicability to openSF V3. □ Linux is the only OS applicable to openSF (references to Windows and MAC OS have been removed) □ Session Logs are placed in the session's output directory.	05/06/2013	
3.2	New version answering the comments generated by ESA on the openSF V3 AR documentation package: □ Implementation of RID OSF-AR3-10: Clarification on simplification of session directory name (section 3.5.6.3.1).	15/01/2014	

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1. Purpose

This is the System Requirements Document (SRD) for the openSF project as derived from the requirements identified in the Statement of Work ([SOW]), and contains:

- ❑ A general description of the framework, specifying functions and relations with former projects.
- ❑ System requirements for the application
- ❑ Forward and inverse traceability matrices between user and system requirements.

1.2. Scope

This document is aimed to describe and list the system requirements of openSF. OpenSF has been designed to cover the scientific necessity of a framework to run and analyse E2E simulation chains.

1.3. Document Structure

This document is organised as follows:

- ❑ Section 1 contains this introduction.
- ❑ Section 2 contains the list of applicable and reference documents, as well as the applicable standards to the project.
- ❑ Section 3 provides a general description of openSF.
- ❑ Section 4 presents the openSF system requirements.
- ❑ Section 5 contains the traceability matrices between the system requirements and the requirements baseline.

1.4. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are listed below:

Acronym	Description
AD	Architectural Design Applicable Document
ADD	Architectural Design Document
API	Application Programming Interface
AR	Acceptance Review
ARM	Atmospheric Radiation Measurement programme
AT	Acceptance Test

Acronym	Description
BB	Broad-Band
BBR	Broad-Band Radiometer
CASE	Computer Aided Software Engineering
CDR	Critical Design Review
CFI	Customer Furnished Item
CM	Configuration Management Configuration Manager
CMP	Configuration Management Plan
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DBMS	Database Management System
DD	Detailed Design
DDR	Detailed Design Review
DDVP	Design, Development and Validation Plan
DMS	DEIMOS Space
ECP	Engineering Change Proposal
E-R	Entity Relationship
FAT	Factory Acceptance Test
GUI	Graphical User Interface
HW	Hardware
I/F	Interface
I/O	Input/Output
ICD	Interface Control Document
IT	Integration Test
ITT	Invitation To Tender
IWC	Ice water content
KOM	Kick-Off Meeting
LW	Long-Wave
MMI	Man-Machine Interface
MoM	Minutes of Meeting
MPI	Message Passing Interface
MPL	Message-Passing Language
MR	Management Review
MSC	Meteorological Service of Canada

Acronym	Description
NCR	Non-Conformance Report
OO	Object-Oriented
OOP	Object-Oriented programming
PA	Product Assurance
PAP	Product Assurance Plan
PDR	Preliminary Design Review
PM	Progress Meeting Project Manager
PMP	Project Management Plan
QA	Quality Assurance
QAP	Quality Assurance Plan
QR	Qualification Review
RD	Reference Document
RDBMS	Relational Data Base Management System
RFD	Request for Deviation
RFQ	Request for Quotation
RFW	Request for Waiver
RID	Review Item Discrepancy
RT	Radiative Transfer
SCMP	Software Configuration Management Plan
SCR	Software Change Request
SDD	Software Design Definition (Document)
SDP	Software Development Plan
SICD	Software Interface Control Document
SIP	Software Integration Plan
SIRD	Software Interface Requirements Document
SMR	Software Modification Report
SOW	Statement Of Work
SPR	Software Problem Report
SR	Software Requirements
SRD	Software Requirements Document
SRN	Software Release Note
SRR	System Requirements Review
SS	System Specification

Acronym	Description
ST	System Test
STD	Software Transfer Document
STP	Software Test Plan
STR	Software Test Report
SUM	System User Manual
SVTR	Software Validation Test Report
SVTS	Software Validation Test Specification
SVVP	Software Verification & Validation Plan
SW	Software Short Wave
TBC	To Be Confirmed
TBD	To Be Defined / Decided
TN	Technical Note
TP	Test Plan
TR	Test Report
TRR	Test Readiness Review
TS	Technical Specification
UML	Unified Modelling Language
UR	User Requirement
URD	User Requirements Document
UT	Unit Test
V&V	Verification & Validation
WBS	Work Breakdown Structure
WPD	Work Package Description

1.5. Definitions

The definitions of the specific terms used in this document are listed below.

Definition	Meaning
Batch mode	It is the capability of the simulator to perform consecutive runs without a continuous interaction with the user. Batch mode checks the agreement or not between the output of a given module and its use as input by the next one in the sequence of the simulation.
Breakpoint	A breakpoint is an intentional stopping or pausing place in a program, put in place for debugging purposes, in case of openSF for inspecting intermediate simulation products. In openSF the breakpoint addition is only allowed between models within the sequence of them in a simulation chain. It is not possible to add breakpoints in the middle of a

Definition	Meaning
	single model execution because it is atomic from framework side.
Configuration File	A XML file that contains parameters necessary to execute a model. A configuration file instance must comply with the corresponding XML schema defined at model creation time. A special case is the global configuration file that defines the configuration parameters that are common to all models.
Framework	Software infrastructures designed to support and control the simulation definition and execution. It includes the GUI, domain and database capabilities that enable to perform all the functionality of the simulator. Models are not considered part of the framework.
Model	Executable entity that can take part in a simulation. A model can be understood, broadly speaking, also as an "algorithm". Basically, it contains the recipe to produce products function of inputs. A model contains also several rules to define the input, output and associated formats. Furthermore, its behaviour is controlled by one configuration file. Overall, the architecture of a model consists of: ❑ The source code and its binary compiled counterpart ❑ A global configuration file with the global simulation parameters. This file is the same for all the models in the simulation chain. ❑ A local configuration file with its parameters ❑ An input file that characterizes its inputs ❑ An output file that characterizes its outputs In addition, it must belong to one stage of the simulation.
Parameter	A constant whose value characterizes a given particularity of a model or models. Parameters are user-configurable, they are fixed before launching a model and, for practical reasons, not all of them shall be accessible from the HMI. A model cannot modify its own parameters (that is a task for external modules).
Parameter Perturbation	Extended sensitivity analysis capability, where multiple runs can be executed for a model, each one corresponding to a different set of configuration parameters drawn statistically according to a probability distribution set up by the user at configuration.
Session	A session is defined as an execution of a simulation, an ordered set of simulations or an iterative execution of simulation/s with different parameter values. There are no restrictions on how to concatenate these simulations, they do not have to be compatible between them but, if necessary, the final output files of a simulation can be used by the following simulation.
Simulation	A simulation is understood as a list of models (or even a model alone) that is run sequentially and produces observable results.
Stage	Entity that defines a phase in a simulation process. The stage order definition specifies the logic of the simulation sequence so a model must have associated a stage and a simulation will run the models of a stages series.
Tool	A tool is an external executable file that performs a given action to a certain group of files. Used into the openSF platform and associated to a certain file extension these tools can be called to perform off-line operations to products involved in simulations.

Definition	Meaning
Version	New instance of an existing model where either the source (and binary) code has been updated or the configuration file schema has been changed. Instead of updating its version, a model is considered as *new* if either of its input and output schema is updated. The justification for this approach considers that it is only with I/O that a model is really embedded in the openSF architecture.

2. RELATED DOCUMENTS

2.1. Applicable Documents

The following table specifies the applicable documents that shall be complied with during project development.

Table 1: Applicable documents

Reference	Code	Title	Issue
[SOW]	EOP-SFP/2009-07-1404/MAr	Statement of Work for the OpenSF end-to-end Simulation Framework Maintenance.	1.1
[ECSS]	OPENSF-ECSS-E40C	OpenSF ECSS E40C Tailoring	September 14, 2009
[DDVP]	OPENSF-DMS-DDVP-001	OpenSF Design, Development and Validation Plan	1.0
[MR01]	OPENSF-MR01	Maintenance Request for openSF v2.0	April 19, 2010
[MR02]	PFL-PSO/FF/fk/11.568	OpenSF E2E Simulation Framework Maintenance Cost Reimbursement Part, Maintenance Request No. 2	August 19 2011
[CR1]	EOP-SFP/2012-12-1686/PB/ag	Change Request for the openSF V3 activities description	-

2.2. Reference Documents

The following table specifies the reference documents to take into account during project development.

Table 2: Reference documents

Reference	Code	Title	Issue
[UML]	ISBN 0-201-57168-4	The Unified Modelling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson.	-
[ECSIM RB]	TEC-SW/ECARE-ETE/RB	End-to-End Mission/System Performance Simulator for EarthCARE – Requirements Baseline	1.2
[ECSIM SOW]	ECARE-ETE/SOW	Statement of Work End-To-End Mission/System Performance Simulator for EarthCARE	1.0
[ECSIM DDVP]	ECSIM-DMS-DDVP-001	ECSIM Design, Development and Validation Plan	2.1
[ECSIM TN]	ECSIM-DMS-TN-001	Technical Note related to the proposed ECSIM Architecture	1B
[ECSIM ICD]	ECSIM-DMS-TEC-ICD01	ECSIM Interface Control Document	1.7
[ECSIM SW]	ESA contract 15346/01/ NL/MM	EarthCare Simulator: Users Guide and Final Report	December 13, 2004
[OSFI]	OPENSF-DMS-OSFI-DM-011	Developer's manual for OpenSF Integration Libraries	1.1
[OSFEG]	OPENSF-DMS-OSFEG-DM-	OpenSF – OSFEG Developer's Manual	1.0

Reference	Code	Title	Issue
	010		
[SEPSO_SUM]	SEPSO-DMS-TEC-SUM-01	Software User Manual	1.3
[ICD]	OPENSF-DMS-ICD-001	OpenSF – Interface Control Document	2.1

2.3. Standards

The following table specifies the standards that shall be complied with during project development.

Table 3: Standards

Reference	Code	Title	Issue
[Q00A]	ECSS-Q-00A	ECSS Space Product Assurance – Policy and Principles	Issue A
[Q20A]	ECSS-Q-20	ECSS Space Product Assurance – Quality Assurance	-
[Q20-09A]	ECSS-Q-20-09A	ECSS Space Product Assurance – Non-Conformance control system	-
[Q80B]	ECSS-Q-80B	ECSS Space Product Assurance – Software Product Assurance	Draft B
[E40C]	ECSS-E-40C	Software development Standard	-

3. GENERAL DESCRIPTION

3.1. Relationship to Former Projects

The openSF project is "descendent" of a previous ESA-funded project, called ECSIM, intended to provide an End-to-end simulator for the EarthCARE mission. OpenSF project must be fully compatible with the previous ECSIM project, so OpenSF interfaces are compatible with the ones defined in the ECSIM project ([ECSIM ICD]).

The main issue in openSF was to extract the abstract and re-usable simulation functionalities that ECSIM simulator had in common with other E2E simulator projects. Beyond this point and after months of requirements study and functionalities testing, the first operational version of OpenSF was released.

Since the original OpenSF development, it has been (and continue to be) used by other ESA projects (for example GERSI, AIPC, SEPSO, S3-OGPP/OSPS). For each of those projects OpenSF is adapted in order to fulfil the project requirements and some of those adaptations and ideas are now part of the OpenSF "*core*". New features are included in OpenSF after a deep examination (consequences, drawbacks, backward compatibility etc…) performed by the management and development team.

3.2. Function and Purpose

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 is a generic simulation framework product being developed by DEIMOS Space, S.L.U. 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.

Scientific models and product exploitation tools can be plugged in the system platform with ease using a well-defined integration process. Figure 3-1 sketches this simple mechanism.

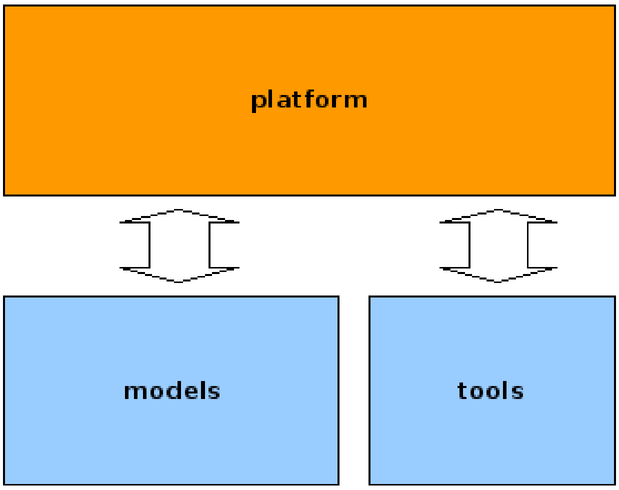


Figure 3-1: Integration of models and tools in openSF platform

Users can integrate every kind of executables as models and product tools because there is no programming language dependency. The integration is made at Operating-System level; therefore, it implies a minimal intrusion into code of scientific models.

The expected functionality and behaviour for OpenSF demands:

- ❑ **Flexibility and scalability** in order to adapt models/algorithms that are formerly running in the existing simulator but also new ones that could appear in the future.

  OpenSF must be also flexible enough to accept models written in different programming languages (IDL, Matlab, C/C++, Fortran 77 and Fortran 90).

  OpenSF must also allow the integration of third party tools for data analysis and exploitation.

- ❑ **Manageability**, in terms of providing a unique application with a user friendly GUI for the operation of the functionality provided to the scientific community;
- ❑ **Configurability for the management between models, simulations and sessions,** i.e. input data needed for their execution. This configurability shall allow to define and select different configurations for a given simulation (e.g. one with more accurate models and one with simplified models, or one with a given failure scenario).
- ❑ **Modularity**. This requirement claims for a design where different modules perform well defined and differentiated functions while inter-module dependency is minimised. For instance, the system should be capable of inserting, deleting or modifying new elements such as the different models/algorithms.
- ❑ **Versioning control of the models**. As the model structure is well known, the framework shall provide configuration control of models, permitting to define a simulation with different versions of the constituent models.
- ❑ **Data visualisation capabilities**. The simulator shall provide the capability to visualise input (atmospheric scenarios) and output data (e.g., level 2 products or comparisons). The visualisation tools shall synthesize as much as possible information in the display. This functionality shall be fully integrated in the simulator's framework.
- ❑ **System automation improvement**. All operations shall be driven through the use of the GUI enabling the automatic consistency checking and out-of range checking avoiding thus the introduction of errors due to the manual file edition;

- ❑ **Provision of a cost-efficient operations/maintenance**. The selection of open source cost-free packages for many of the system components ranging from the operating system itself up to different models intervening in the simulations enable a cost effective solution that is also extremely relevant to minimise costs associated to the exploitation of the simulator.

3.3. Environmental Considerations

Taking into account the above-mentioned limitation, for the **OpenSF development** the following environment shall be used:

- ❑ Operating system: Linux.
- ❑ Model coding languages: IDL, Matlab, ANSI C, C++, Fortran 77 or Fortran 90
- ❑ Shell environment for batch execution (bash, tcsh, Windows command line, Cygwin, Msys etc.)
- ❑ UML for the data model specification.
- ❑ Development language: Java using JDBC library for the interface with the RDBMS.
- ❑ RDBMS: MySQL.
- ❑ Essential libraries for the current simulator will be:
  - XML libraries for ingesting the configuration files associated to each model
  - Integration libraries that form part of the OpenSF system. These libraries are described in [RD OSFI].

For **Configuration Management** the following software shall be used:

- ❑ SVN for CM tasks.

For **Project Management** the following software shall be used:

- ❑ Microsoft Project for planning and reporting of Project Management tasks.
- ❑ Microsoft Excel 2003 for reporting the software metrics to be included in the Progress Reports

For Software **Maintenance** the following platforms shall be used:

- ❑ Mantis Bug Tracking system for SPR management and framework bugs.

For **Documentation** the following software shall be used:

- ❑ Word 2003 to generate the documentation.
- ❑ Adobe Acrobat (v7) to generate the PDF format to deliver the documentation via electronic media.

3.4. OpenSF version 2 added capabilities

This section contains the list of capabilities that shall be added to openSF version 2. The descriptions are basically taken from the proposal presented by DEIMOS in response to the Maintenance Request MR01, issued by ESA. Thus, as the proposal's contents have been agreed by DEIMOS and ESA, these sections shall represent the user requirements in order to establish the traceability between the SRD and the RB, presented in section 5.

3.4.1. Multiple repository/processing chain capability

Version 1 of openSF admits only the definition of one simulation chain, and from there, the definition of models, simulations and sessions.

A processing or simulation chain is a set of processing steps where each step represents a decisive processing stage in the chain. An example of a processing chain with 4 stages is depicted hereafter.

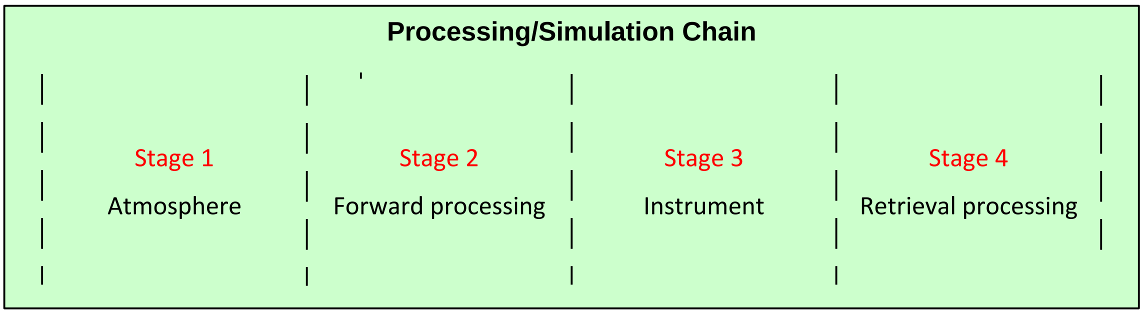


Figure 3-2: Definition of a processing chain

A simulation is a subset of stages from the simulation chain, composed by one or more of the processing stages from the simulation chain.

Upon building a simulation, models are associated to all stages defined for that simulation. Thus, the simulation shown in the example below, is constituted by three processing stages. In this case, each one is covered by one model.

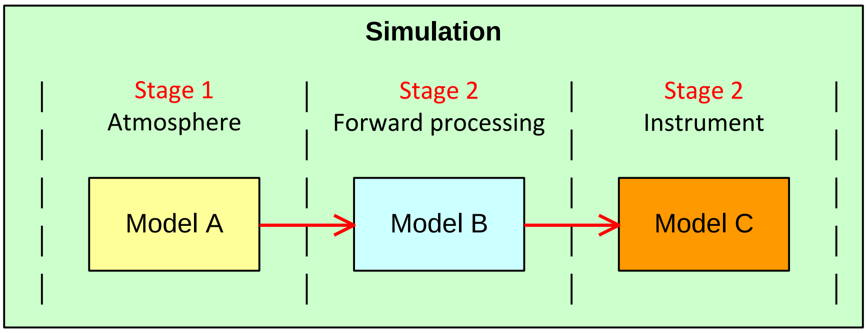


Figure 3-3: Definition of a simulation

openSF version 2 shall allow the possibility to define more than one processing chain within the same openSF instance. Thus it would be possible to hold simulations for more than one mission, or to define variations of the processing chain for one mission.

This change implies the development of a simple database manager that handles the different simulation scenarios for every mission. This database manager will create different tables for every mission allowing users to switch between missions at the openSF startup or when it is demanded.

3.4.2. Parameter Editor

A parameter management tool shall be integrated into openSF for creating and editing parameters as well as setting relationships between them with the objective to ensure the consistency of the models' configuration.

OpenSF parameter management system shall be composed of two software modules, a parameter rule editor and a parameter editor.

The **Parameter Rule Editor** shall be used offline, before the simulation definition etc, 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 session definition stage.

The **Parameter Editor** shall represent a graphical interface to visualize and edit the configuration parameters that participate in a simulation.

3.4.2.1. Parameter Rules - Grammar Definition

A simple grammar shall be designed for defining the rules that govern the parameter editor. This grammar shall be based on a XML syntax, detailed below.

A single rule is composed of:

1. A unique rule identifier

    <rule id=”ID”>

1. An operation tag, nested to the identifier. There are 3 operation types:

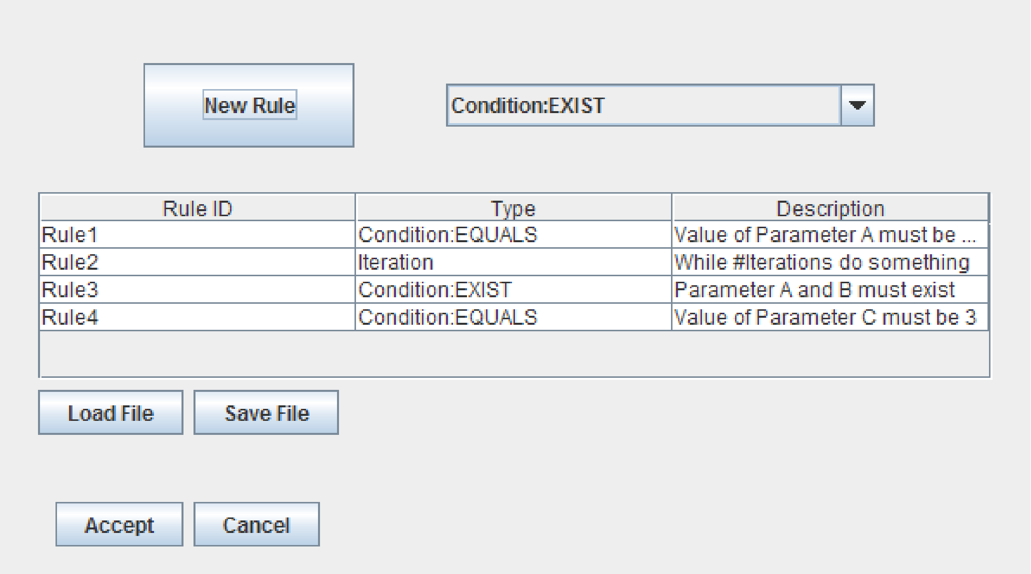
    - **Condition**: <condition type=”ConditionType”>. Condition type is a list of the most used logical operators (equals, exists, greater than, etc …)
    - **Action**: <action type=”ActionType”>
    - **IF statement**: nested to an “if” tag, a condition and an action.

This grammar allows the user to define rules (conditions, constraints etc…) when setting the models’ parameter during the session definition or edition.

XML Rules File Sample

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rulesFile>
  <rule id="ID Number 1">
    <condition type=”EQUALS”>
      <value name=”PARAM NAME”>
      <value name=”PARAM NAME”>
    </condition>
  </rule>
   <rule id="ID Number 2">
    <condition type=”EXIST”>
      <parameter name=”PARAM NAME”>
      <parameter name=”PARAM NAME”>
      <parameter name=”PARAM NAME”>
    </condition>
  </rule>
  <rule id="ID Number 3">
    <if>
      <condition type=”EQUALS”>
        <value name=”PARAM NAME”>
        <value name=”PARAM NAME”>
      </condition>
      <action type=”EQUALS”>
        <value name=”PARAM NAME”/>
          <constant value=”WHATEVER”/>
      </action>
    </if>
  </rule>
</rulesFile>
```

OpenSF also provides a graphical front-end for the creation and edition of this grammar. A draft of this form is shown below.

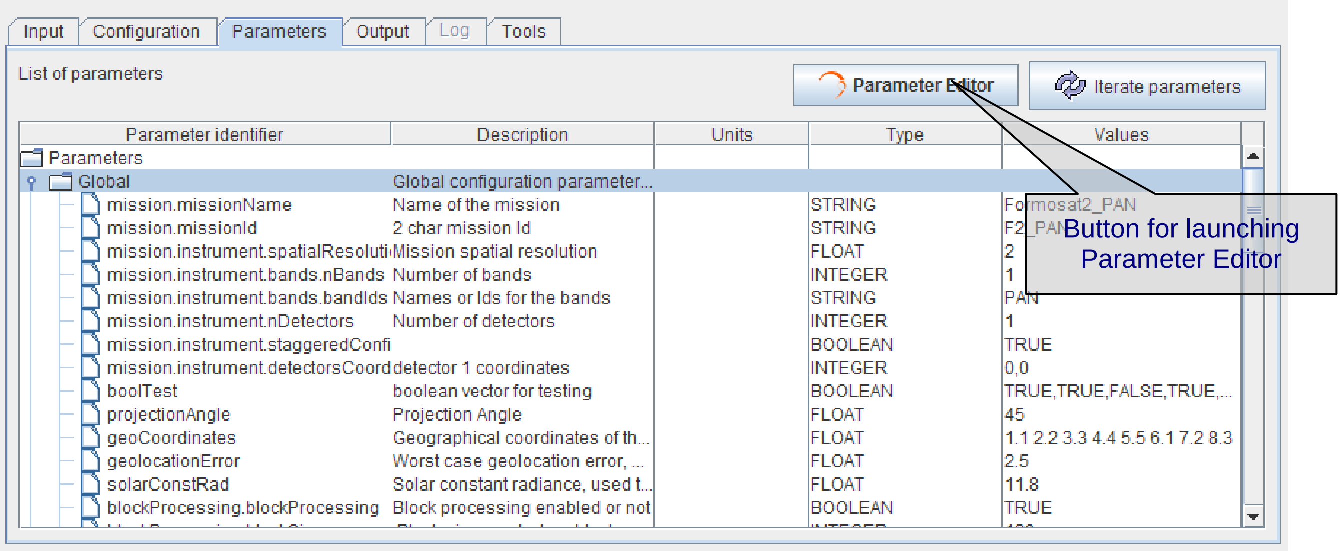


Rule ID	Type	Description
Rule1	Condition:EQUALS	Value of Parameter A must be ...
Rule2	Iteration	While #Iterations do something
Rule3	Condition:EXIST	Parameter A and B must exist
Rule4	Condition:EQUALS	Value of Parameter C must be 3

Figure 3-4: Parameter rule editor. Draft design

3.4.2.2. Parameter Editor

The Parameter Editor is a graphical interface to visualize and edit the configuration files involved in a simulation. Thus, Figure 3-5 presents the parameter viewer, which is accessible from the session interface in a dedicated tab for the session parameters edition. From this screen users can access to the Parameter Editor by clicking the corresponding button.



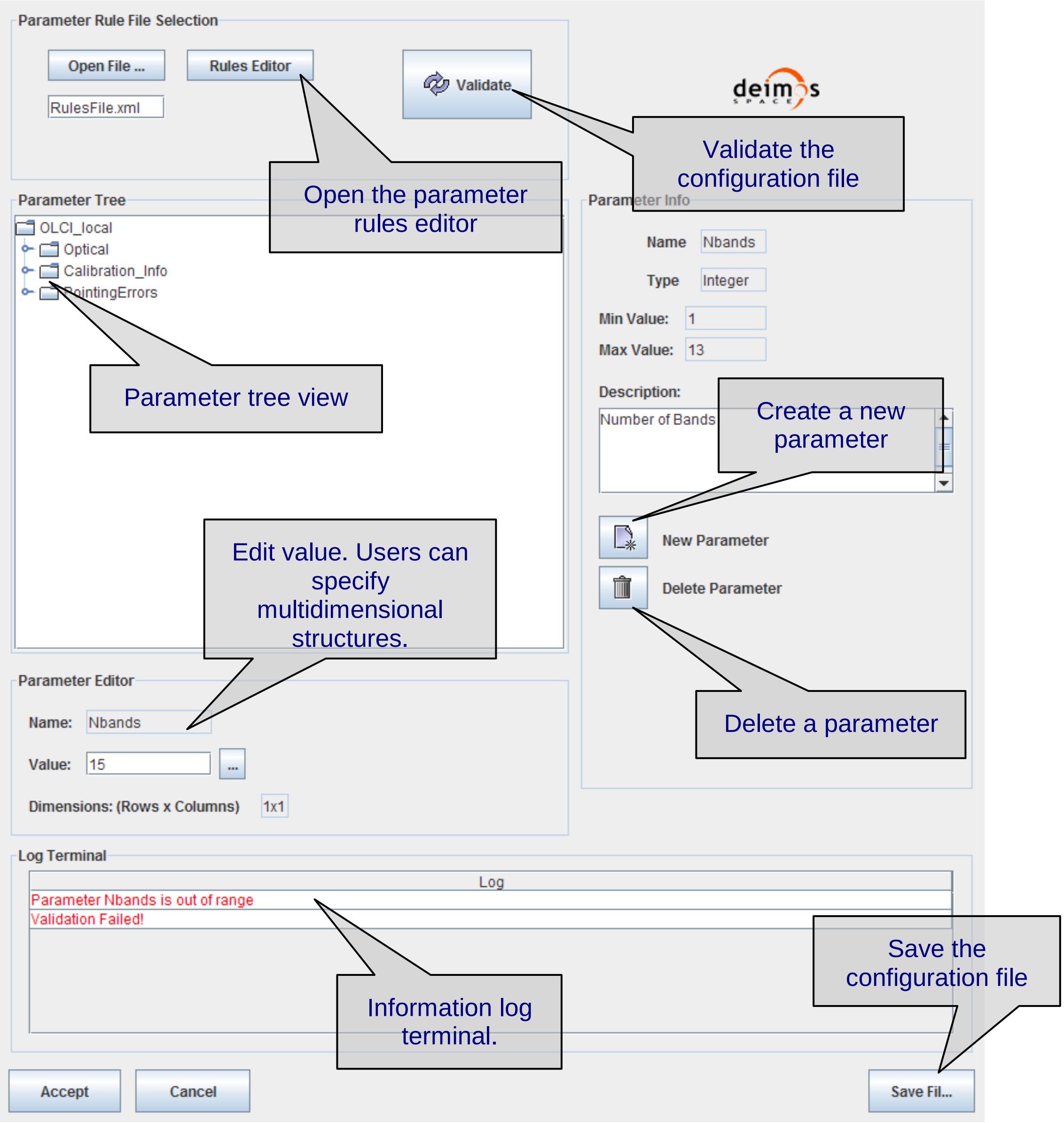
Parameter identifier	Description	Units	Type	Values
Parameters				
Global	Global configuration parameter...			
mission.missionName	Name of the mission		STRING	Formosat2_PAN
mission.missionId	2 char mission Id		STRING	F2_PAN
mission.instrument.spatialResoluti	Mission spatial resolution		FLOAT	2
mission.instrument.bands.nBands	Number of bands		INTEGER	1
mission.instrument.bands.bandIds	Names or Ids for the bands		STRING	PAN
mission.instrument.nDetectors	Number of detectors		INTEGER	1
mission.instrument.staggeredConfi			BOOLEAN	TRUE
mission.instrument.detectorsCoord	detector 1 coordinates		INTEGER	0,0
boolTest	boolean vector for testing		BOOLEAN	TRUE,TRUE,FALSE,TRUE,...
projectionAngle	Projection Angle		FLOAT	45
geoCoordinates	Geographical coordinates of th...		FLOAT	1.1 2.2 3.3 4.4 5.5 6.1 7.2 8.3
geolocationError	Worst case geolocation error, ...		FLOAT	2.5
solarConstRad	Solar constant radiance, used t...		FLOAT	11.8
blockProcessing.blockProcessing	Block processing enabled or not		BOOLEAN	TRUE

Figure 3-5: Simple parameter viewer

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.

A screenshot of the Parameter Editor can be seen hereafter.



RulesFile.xml

Name: Nbands
Type: Integer
Min Value: 1
Max Value: 13
Description: Number of Bands

Name: Nbands
Value: 15
Dimensions: (Rows x Columns) 1x1

```
Parameter Nbands is out of range
Validation Failed!
```

Open the parameter rules editor

Validate the configuration file

Parameter tree view

Create a new parameter

Edit value. Users can specify multidimensional structures.

Delete a parameter

Save the configuration file

Information log terminal.

Figure 3-6: Parameter Editor

3.4.3. Multi-language model support

openSF version 1 offers native support for model integration in the three most common scientific programming languages: ANSI C, C++ and Fortran 90. For that, openSF is delivered with integration libraries in these programming languages so that models can be developed using a specific API for satisfying a series of interface requirements that openSF demand for working correctly. More information can be found in [OSFI].

OpenSF v2 shall extend the model integration by supporting additional programming languages. These are IDL, Matlab and Fortran 77.

The work needed to fulfil this task implies to develop the integration libraries in these new languages and to test the model's integration adequately. For that, models shall be implemented using the OSFI functions, simulations with these models shall be created and run to check that the behaviour is as expected.

The development of a Fortran 77 interface will be based on a wrapper over the common OSFI C++ libraries. This mechanism is the one used for the libraries in ANSI C and Fortran 90.

Matlab and IDL interfacing issues are specified in the reference document [OSFI]. For these two languages, specific development licenses are needed for the libraries implementation to cope with the three different platforms where openSF is supported.

3.5. OpenSF version 2.2 added capabilities

This section contains the list of capabilities to be part of openSF version 2.2 that have been requested by ESA in [MR02]. Following the same approach as in section 3.4, as no user requirements have been specified, the next sections provide a detailed description of the new capabilities, basically extracted from the proposal DEIMOS submitted in response to that Maintenance Request.

3.5.1. Parameter Perturbation

The idea is to improve the Sensitivity analysis approach provided in openSF by combining it with the perturbation functions used in SEPSO.

In openSF version 2.0 users can perform sensitivity analysis of certain parameters by creating iterative sessions. This is a powerful feature that helps to run a large number of simulations by changing values of the configuration parameters.

Furthermore, the development of SEPSO introduced some extended capabilities in openSF on this respect. Multiple executions for the same model where possible, each run corresponding to a different set of configuration parameters drawn statistically according to probability distributions. The result of the executions is averaged to come up with a mean and a sigma values, that is, there is a XML product with the mean values, another with the standard deviation ones.

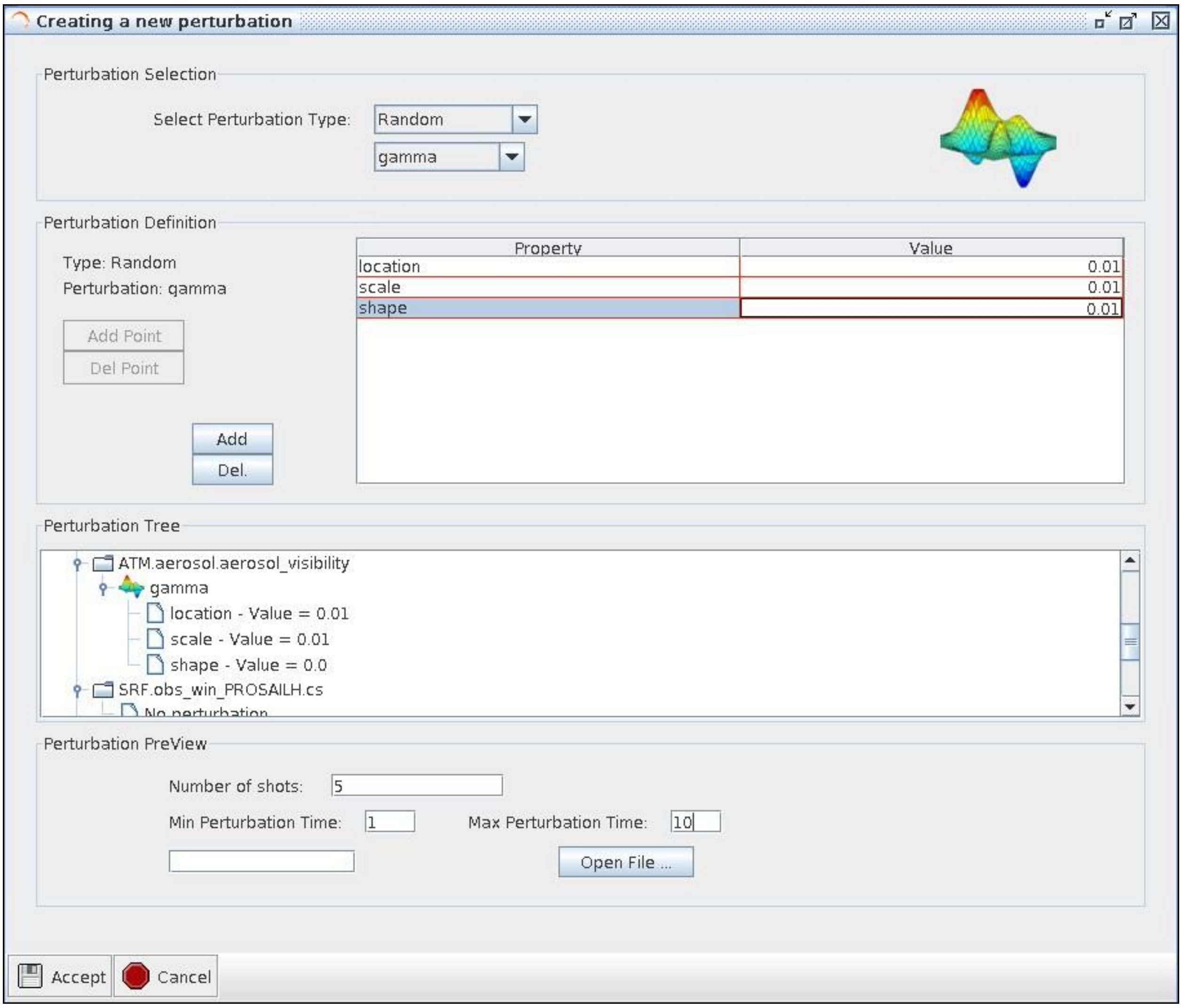
The idea for openSF v2.2 is to combine the statistical functions from SEPSO into the sensitivity analysis capabilities of the current openSF so that new numerical sequences can be created and therefore to have a simulation tree able to run with the combination of all parameter values.

Next subsections describe the HMI features that enable to attach the new perturbation functions to parameters, and the full list of these statistical functions that shall be applied.

3.5.1.1. Perturbation Parameters - Human Machine Interface

The definition of a perturbation required to implement a new software module in the framework, developed under the SEPSO project frame. This software module consists in a dedicated HMI where users can easily define any of the supported perturbation functions (analytical, random etc…) for a set of configuration parameters.

Figure 3-7 shows the main frame of the Parameter Perturbation interface.



Select Perturbation Type: Random
gamma

Type: Random
Perturbation: gamma

Property	Value
location	0.01
scale	0.01
shape	0.01

- ATM.aerosol.aerosol_visibility
  - gamma
    - location - Value = 0.01
    - scale - Value = 0.01
    - shape - Value = 0.0
- SRF.obs_win_PROSAILH.cs
  - No perturbation

Number of shots: 5

Min Perturbation Time: 1   Max Perturbation Time: 10

Figure 3-7: openSF Parameter Perturbation interface

The description of the Parameter Perturbation graphical interface is given below (extracted from SEPSO's SUM, [SEPSO_SUM]):

- The upper part of the window includes the type of error associated to a parameter. Several types of functions are available to the user. Upon selecting one (*random gamma* distribution in Figure 3-7), the "Perturbation Definition" area will change to indicate which parameters for the function are necessary (for instance, the typical Gaussian distribution needs a mean value and a sigma or standard deviation). For the random gamma the parameters are location, scale and shape. Once the distribution of error is fixed writing values in the "Value" column, it is necessary to move to the lower panel "Perturbation Tree".

- ❑ The Perturbation tree shows all the configuration parameters where a perturbation can be included. If nothing is attached there is a label that reads “no perturbation”.
- ❑ To add a perturbation to a given parameter (*aerosol visibility* in Figure 3-7), click on the parameter then select the button “add” above the perturbation tree. This will change the “no perturbation” label into a label indicating the perturbation defined in the upper part of the window (in this case a gamma function). Notice that the values for each of the parameters of the error perturbation is also shown besides the parameters perturbed.
- ❑ The last point is to include the number of shots, that is, the number of times the module will be executed. In each of the runs the perturbation function will have a different value, following the distribution selected in the “Perturbation Selection” frame in the upper part of the window. The number of shots is at the end of the window “Creating a new perturbation”.
- ❑ If the error function assigned to the parameter is an analytical one, it is assumed that the perturbation per shot actually can be represented by a time series. For example, it could be a perturbation $Y = cos(\Omega*t)$; then *t* values (in number equal to Number of shots) will be drawn between “Min Perturbation Time” and “Max. Perturbation Time”. Note that the step between points is as usual (max-min) / number of shots

3.5.1.2. Parameter Perturbation - Error Functions

The list of currently implemented error functions extracted from [OSFEG] is:

3.5.1.2.1. Deterministic Functions

Deterministic functions are those whose value it is known in all the time domain.

❑ **Affine**

Calculates the perturbation as an affine value. An affine transformation consists in a linear transformation and a translation.

- error = a1 + a0 * t

❑ **Bias**

Calculates the perturbation as a constant value.

❑ **Linear**

Calculates the perturbation as a linear value:

- p = a * t

This is a particular case of affine transformation when translation variable is equals to 0.

❑ **Parabolic**

Calculates the perturbation as a parabolic value.

- $p = a_0 + a_1 * t + a_2 * t^2$

❑ **Polynomial**

Calculates the perturbation as a generic polynomial value. This function has as many float parameters as degrees of the desired polynomial plus one.

❑ **Step**

Calculates the perturbation as step function.

- if simTime < t => p = a_0
- if simTime > t => p = a_1

❑ **Sinusoidal**

Calculates the perturbation as sinusoidal function

- p = a * sin(2 * pi * f * t + phi)
- f(Hz)
- phi(deg)
- t(secs)

❑ **Tangent**

Calculates the perturbation as tangent function

- p = a * tan(2 * pi * f * t + phi)
- f(Hz)
- phi(deg)
- t(secs)

Remember that the tangent function have singularities when the angle evaluated is (+-)*n*pi/2.

3.5.1.2.2. Sampling Functions

Error Generation libraries implement 3 interpolation methods, linear, polynomial and spline sampling.

In order to define the points of the interpolation there is a common set of variables that are listed below.

- xMin: Min value of abscise axis
- xMax: Max value of abscise axis
- step: Increment between abscise values

The number of points must be :

$$\frac{xMax - xMin}{step} = nValues$$

❑ **Linear Sampling**

This function makes an interpolation with the given points assuming it follows a linear rule. In out of range values

❑ **Polynomial Sampling**

This interpolation method builds a polynomial grade n, being n the number of specified points. This interpolation minimizes the Least Square Error. Ref: Neville Method.

❑ **Spline Sampling**

Interpolate the given “n” points with Cubic Splines Method.

3.5.1.2.3. Non-deterministic Functions

Common random function implementation with seed management for testing purposes.

- Beta Distribution
- Gamma Distribution
- Exponential Distribution
- Normal Distribution
- Uniform Distribution
- Poisson Distribution
- Truncated Gaussian Distribution
- Uniform Discrete Distribution
- Distribution with custom Probability Density Function

They return the value of a random variable generated with a custom pdf given. It is only recommended to use it by expert developers/scientists.

3.5.1.2.4. *Binary and Composite Operations*

Error Generation Libraries implements the basic mathematical operations in binary mode. The operations implemented are:

- Addition
- Subtraction
- Multiplication
- Division
- Exponentiation
- Root

Composite operations consist of a deterministic function with one or more of its parameters following another function or binary operation.

3.5.1.3. Multiple Run Handling

During the openSF training session ESA proposed to evaluate the impact in openSF to manage multiple executions of one or more models within a given simulation, as illustrated in Figure 3-8.

This is the case of models that need to handle n acquisitions, and therefore cannot continue to the next processing stage until the n instances have been completed.

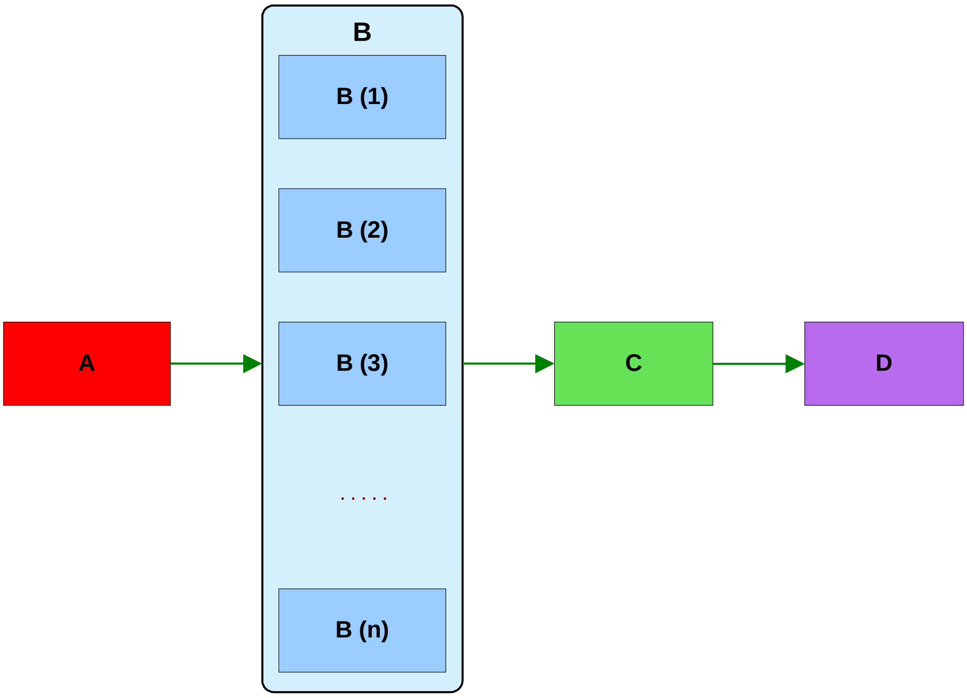


Figure 3-8: Simulation with a model executing N times

DEIMOS has experience handling this topic in past projects such as SEPSO, where a specific mode (named FULL Mode) was defined involving multiple runs for the same model, each one having a different set of configuration parameters. It has been further demanded in the BIOMASS E2E simulator project, and has been solved at the model level, this is, the model itself is in charge of setting the loop for the algorithm invocation, and the collection and preparation of the output data, to be in line with the next model's inputs.

The new approach for multiple runs in openSF version 2.2 is summarised hereafter.

OpenSF iteration of parameters has been previously based on the combination of all possible values. This approach results in an exponential increasing number of executions depending on the number of parameters being iterated and the number of parameter values (i.e. the iteration of two parameters with ten different values each one involves one hundred executions)

In many cases the solution provided by openSF is not the optimal one, statistical models (as in SEPSO case), models that need to run in different modes for each simulation (Sentinel 3 processors) or models that need to be executed more than one time depending on a parameter (BIOMASS acquisitions). For coping with the needs of such simulators, two new possibilities will be implemented for handling iteration/perturbation of parameters or multiple runs. Each model can be configured with different parameter perturbations/iterations, no extra simulations shall be setup just a loop is introduced executing the model "*N*" times. Note that the output files/folders of each model will be the same and consequently **it is model responsibility the handling of this issue**.

The other approach that will be added is the possibility of setting up a number of simulations equal to the number of shots configured for the perturbations of each model (this approach has a constraint, all model perturbations shall be set-up with the same number of shots, the no-perturbed ones will be re-run with the same parameters). This approach is presented in Figure 3-9.

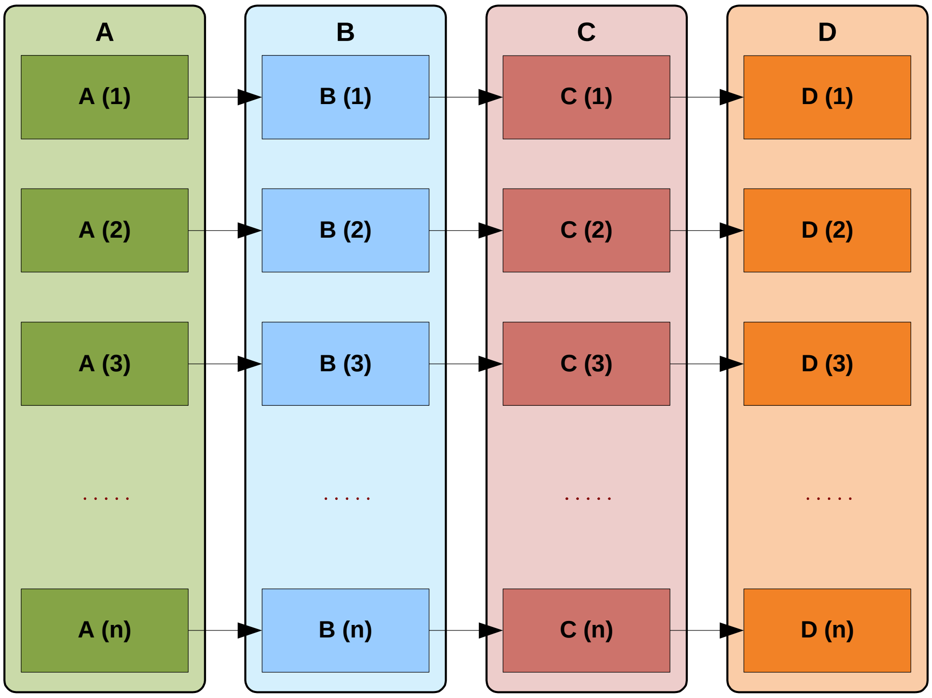


Figure 3-9: Number of simulations equal to the number of defined shots

3.5.2. Parameter Editor and Perturbation integration

The parameter editor, currently an external tool separated from openSF, shall be embedded in openSF.

- ❑ A new button will be added in the Session Edition window in order to launch the Parameter Editor with the session correspondent configuration files already loaded.
- ❑ A menu will be added in order to launch Parameter Editor from openSF MMI.
- ❑ For the perturbation integration a new button will be added in the Session Edition frame allowing users to create and modify the perturbations explained in section 3.5.1.

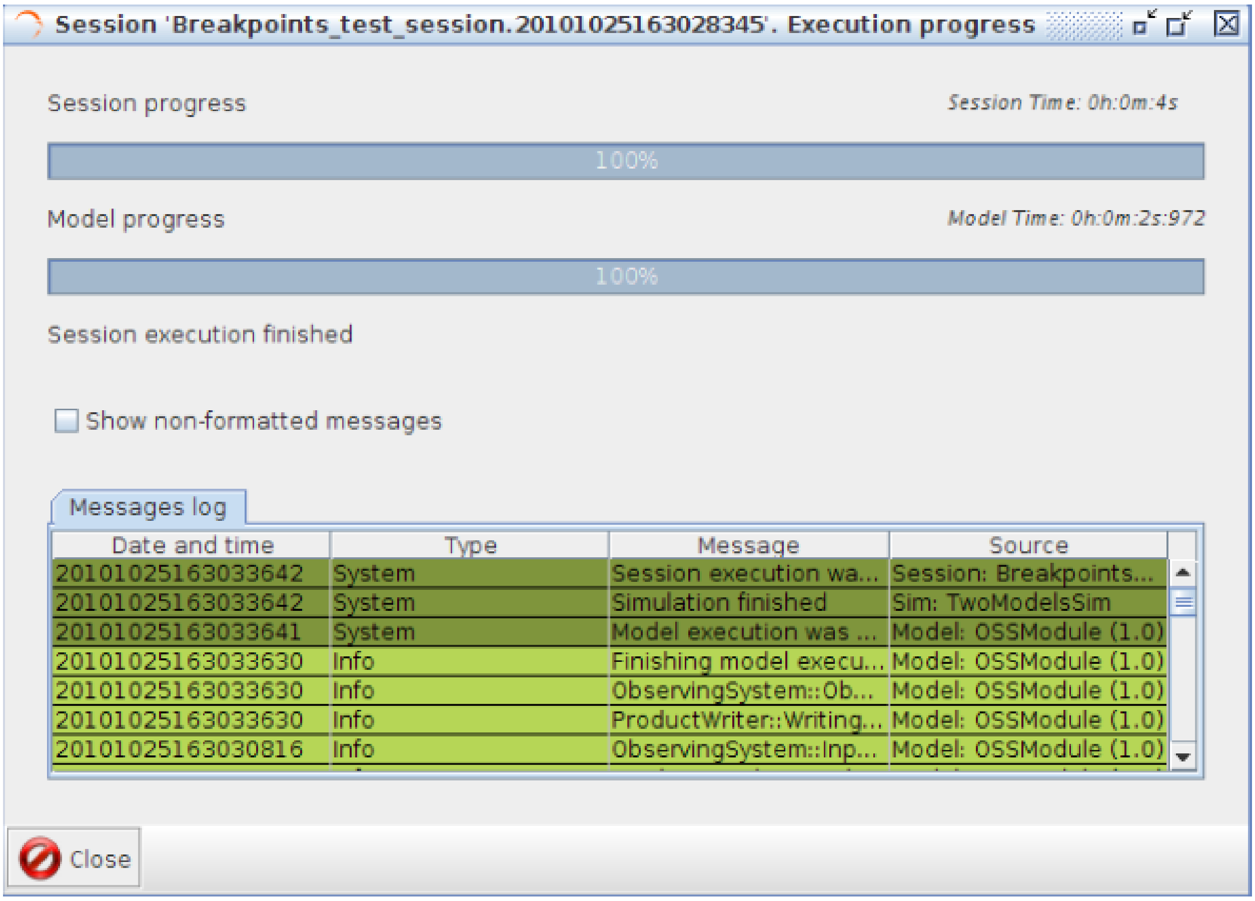
3.5.3. Tool Management

Currently openSF supports the association of one tool to one file extension (E.g, the Gimp tool is associated to jpg files). The proposed activity is the implementation of a mechanism allowing to associate more than one file extension to a single tool. In the example provided before, the Gimp tool could be also associated to gif, and png file types.

3.5.4. Check Output Generation

OpenSF currently uses the command line calling convention (see [ICD]) to communicate each model where the output files must be written in order to properly connect the rest of models within a simulation chain. Nevertheless openSF relies on the model and does not check if output files have been successfully created or not.

It is proposed the addition of a basic mechanism to check if an executed model has written the output files specified by openSF. The result will be a new log message warning the user about the result of this checking operation. The log message will be presented as a *Warning* in the Session Execution window.



Session 'Breakpoints_test_session.20101025163028345'. Execution progress

Session progress Session Time: 0h:0m:4s

100%

Model progress Model Time: 0h:0m:2s:972

100%

Session execution finished

□ Show non-formatted messages

Messages log

Date and time	Type	Message	Source
20101025163033642	System	Session execution wa...	Session: Breakpoints...
20101025163033642	System	Simulation finished	Sim: TwoModelsSim
20101025163033641	System	Model execution was ...	Model: OSSModule (1.0)
20101025163033630	Info	Finishing model execu...	Model: OSSModule (1.0)
20101025163033630	Info	ObservingSystem::Ob...	Model: OSSModule (1.0)
20101025163033630	Info	ProductWriter::Writing...	Model: OSSModule (1.0)
20101025163030816	Info	ObservingSystem::Inp...	Model: OSSModule (1.0)

Figure 3-10: openSF Session Execution window

3.5.5. MATLAB errors inclusion

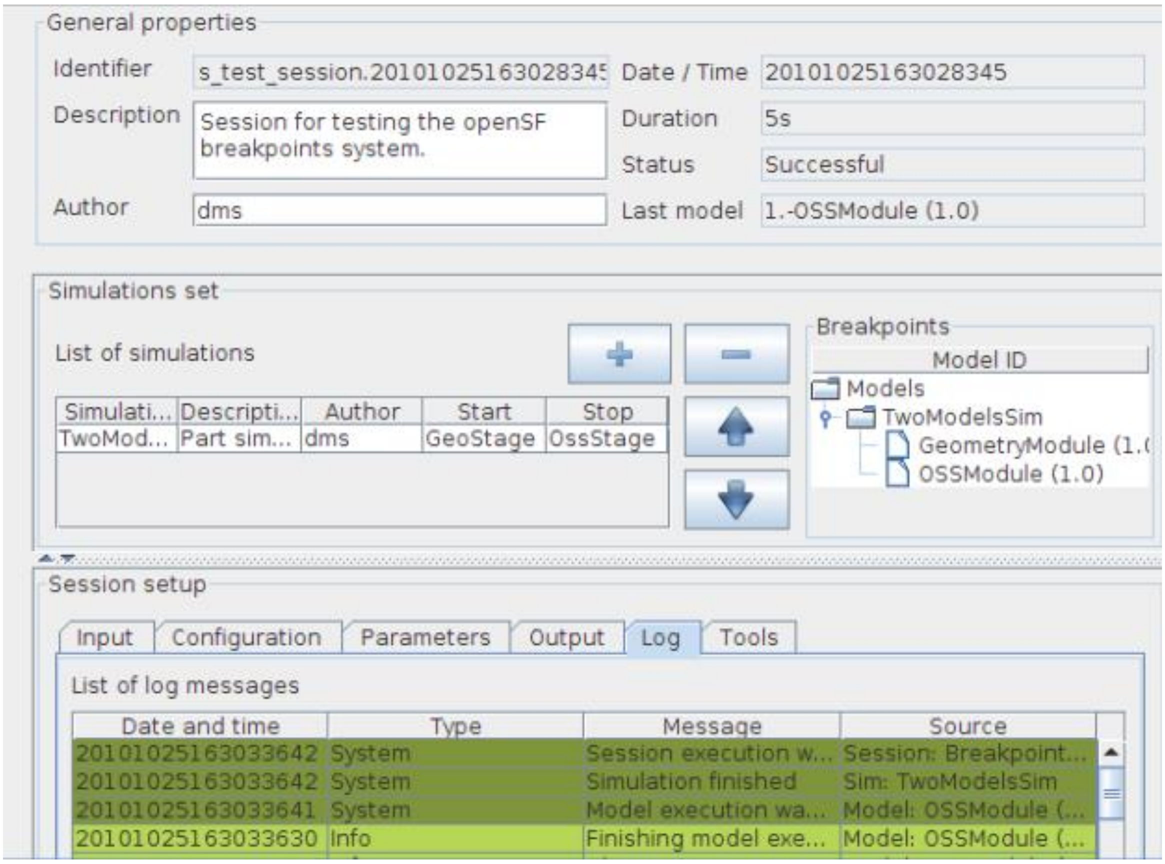
This new feature is the inclusion in the OSFI libraries of some error messages that are silently reported by MATLAB and consequently not captured by openSF. An example is the out of memory error when MATLAB is not able to allocate more memory for one model.

Additionally this activity will study the performance of openSF regarding the management of MATLAB and IDL processes as it has been reported that openSF introduces an overhead when running models written in these languages.

3.5.6. Extended log capabilities

3.5.6.1. Record the model execution time

As shown in Figure 3-10, upon completion of a session execution, openSF presents the total time of the session. This time is stored in the openSF database for later access from the Session Results view.



General properties

Identifier: s_test_session.20101025163028345
Date / Time: 20101025163028345
Description: Session for testing the openSF breakpoints system.
Duration: 5s
Status: Successful
Author: dms
Last model: 1.-OSSModule (1.0)

Simulations set

List of simulations

Simulati...	Descripti...	Author	Start	Stop
TwoMod...	Part sim...	dms	GeoStage	OssStage

List of log messages

Date and time	Type	Message	Source
20101025163033642	System	Session execution w...	Session: Breakpoint...
20101025163033642	System	Simulation finished	Sim: TwoModelsSim
20101025163033641	System	Model execution wa...	Model: OSSModule (...
20101025163033630	Info	Finishing model exe...	Model: OSSModule (...

Figure 3-11 openSF Session Results View

On the other hand the time spent by each of the models involved in a simulation is not stored anywhere and consequently this information is lost (although this is not completely true, as the session log contains the date and time of log messages so the information can be retrieved).

It is proposed to store in the database the time spent by all models within a session. The session results will be also slightly changed in order to present to the user the *"Model duration"*.

The model duration interface will graphically show the time consumed by each model within a session.

3.5.6.2. Change date and time format in log messages

As shown in Figure 3-10 and Figure 3-11 the current log message contains a *Date and time* field that is presented using the format "*YYYYMMDDHHmmSSsss*", with *sss* denoting miliseconds. This time format is not easy understandable by users so it is proposed to change it to a more comprehensive one.

The new format proposed is the UTC ASCII one "*YYYY-MM-DD HH:mm:SS:sss*". Example: 2016-04-16 18:00:01:000. In this improvement it is not needed to change the way openSF internally handles dates, only the way it is presented to the user.

3.5.6.3. Change session results name

This update refers to the same issue presented above but applied to the name of the session executions. The proposed session naming is presented below.

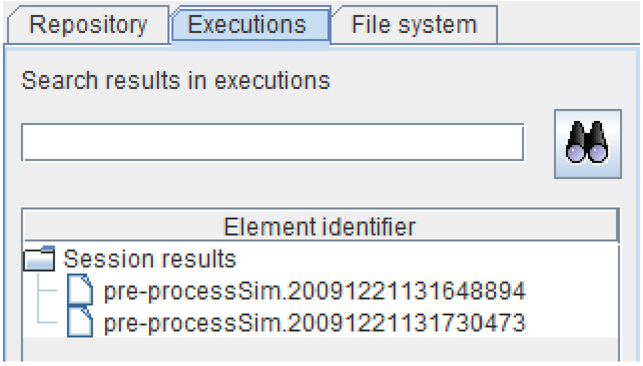

Repository
Executions
File system
Search results in executions
Element identifier
Session results
pre-processSim.20091221131648894
pre-processSim.20091221131730473

Figure 3-12 openSF Executions Tab

3.5.6.3.1. Simplify session results directory names

openSF approach for naming session execution directories as well as session execution supporting files involves the use of names with a timestamp. The use of a timestamp is meant to ensure a unique identification of the session folder and files. Nevertheless, upon openSF user's request, it was identified that handling such names with timestamp is not "user-friendly". In order to overcome this inconvenience openSF shall support a more user-friendly naming.

In order to simplify session results directory names symbolic links shall be used. Each time a session is executed a Linux symbolic link shall be generated in the file system with the name of the session being executed and pointing to the corresponding session execution directory. Each time a session is re-run the symbolic link is re-generated pointing to the latest session execution (the one with the latest timestamp).

3.5.6.4. Detached log window

It is proposed that execution log window can be detached from main frame and therefore maximized as much as computer screen allows.

3.5.6.5. Show all terminal messages in execution log window

It has been additionally requested that all the messages shown on the xterm from which openSF is launched shall appear in the log.

For example on the log it can be seen only "model has crashed" but on the xterm there is a "error while loading shared libraries"

Thus, the load area of the openSF would have two simultaneous log windows, one with well formatted messages and one with unformatted / system messages.

3.5.7. Keyboard shortcuts

Currently openSF provides a basic mechanism that allows the use of keyboard shortcuts in the openSF main frame. Users have reported that it is not strong enough and it is proposed to extend it for the most representative openSF functionalities such as run a session or launch the Parameter Editor.

Additionally is also proposed the association of a key for some specific buttons. For instance, "A" accept, "C" cancel, "D" delete etc…

3.5.8. OSFI

There has been also a request to change the C OSFI library in order to include a log function that accepts messages with variable fields, similar to the printf function in C.

3.5.9. Backup and Import/export capabilities

Currently openSF does not provide any mechanism to perform the following operations:

- ❑ Backup of the openSF entities structure (stages, models, simulations etc…)
- ❑ Import/export of sessions

An enhancement for openSF is the implementation of these operations. An overview of the implementation details is given in the next two sub-sections.

3.5.9.1. Backup of the openSF entities

The implementation of this new capability will consist on the addition of a “Backup” action to the Database/Repository existing capabilities (new database, delete, import etc…).

This backup operation will perform a dump of a selected repository, creating a SQL script ready to be executed and imported in other openSF instance installed in any other machine. This capability is related with the administration mechanisms of MySQL database (*mysqldump*).

3.5.9.2. Import/export mechanism for sessions

The steps to be performed by the import/export capability will be:

Export session

- ❑ Create an SQL file listing all elements (stages, descriptors, models, simulations, etc.) comprising the desired session to export.
- ❑ Create a compressed file containing all files needed for the execution of the session, as well the archives obtained from the executions of that session.

Import session

- ❑ Execute the SQL file produced as the outcome from the export operation, and check that all entities related with the imported session are available in the openSF repository.
- ❑ Create a folder with the new session containing all data included in the compressed file created by the export capability.

3.5.10. GUI isolation

It has occurred many times that openSF is running a time-consuming simulation and the user has accidentally closed the GUI. This action causes to kill the running simulation, losing all generated data (log and intermediate products) of the running model.

What is therefore proposed is to display a dialogue box informing the user about this circumstance when closing the GUI.

3.6. OpenSF version 3 added capabilities

[CR1] lists the activities to be implemented as part of the openSF release 3. They are:

- ❑ *REQ-1*: Parallelisation. Implement a basic scheduler at simulation level
- ❑ *REQ-2*: Removal of logs from database
- ❑ *REQ-3*: Simplify the model management chain, including the capability to run sub- chains, and the use of mandatory & optional inputs files.
- ❑ *REQ-4*: Possibility to switch or select a model version before launching a session, instead of defining a new simulation
- ❑ *REQ-5*: Possibility to remove intermediate data during the simulation executions
- ❑ *REQ-6*: Support for introducing statistical behaviour at model level. This implies the availability of OSFEG libraries, which were part of previous contracts using openSF.
- ❑ *REQ-7*. Capability to bypass/switch-off models
- ❑ *REQ-8*. Capability to copy elements (models, simulations, descriptors)
- ❑ *REQ-9*. Rerun a session from a certain point, with previous data
- ❑ *REQ-10*. Export capability, i.e. save model configuration file in database, check integrity
- ❑ *REQ-11*. Convert SQL syntax into XML for defining elements of OpenSF externally

They are analysed and described in the following sections.

3.6.1. Parallel processing

3.6.1.1. Parallel Processing Needs

The necessity of high throughput systems in simulation environments, especially in space software systems, where the complexity of some algorithms/processes is high and the volume of data to process can be huge. This fact together with some of the most popular simulator applications such as sensitivity or uncertainty analysis, which typically require a lot of executions, make high performance implementations a design goal when developing simulation systems.

Particularizing for E2E simulators, some of the tasks that have classically required more processing resources have been:

- ❑ Forward models simulating physical or weather analysis (radiative transfer models etc…)
- ❑ Scene generators for complex acquisition instruments
- ❑ Line-of-sight models, atmosphere ray tracing.
- ❑ Montecarlo analysis
- ❑ Processing of huge satellite images in several bands and with high resolutions.

There are a lot of parallel technologies within the software community. Among them, and considering the man cost and implementation complexity, DEIMOS chooses the **multicore programming** for the parallel computing in openSF.

A **multi-core processor** is a single computing component with two or more independent actual processors (called "cores"), which are the units that read and execute program instructions. The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can be parallelized to run on multiple cores simultaneously (Amdahl's law).

Multicore is typical implementation of the Divide and Conquer programming paradigm making it suitable for any problem where this theory can be applied.

3.6.1.2. openSF Multicore Adaptation

openSF target system is typically a computer or server. The popularization (cost reduction) of multicore processors implies that currently almost every target computer has two or more cores. Furthermore Java provides a set of tools for thread scheduling and synchronization allowing openSF to launch different simulation models in different cores.

Below these lines we describe an example of multicore technology application using openSF. For sake of clarity, it is assumed that each model involved consumes one processing time unit

Figure 3-13 presents a simple simulation chain involving four models and their dependencies. Figure 3-14 depicts the old concept of a single thread simulation execution within openSF. We see that the total execution time of the full simulations is 4 units.

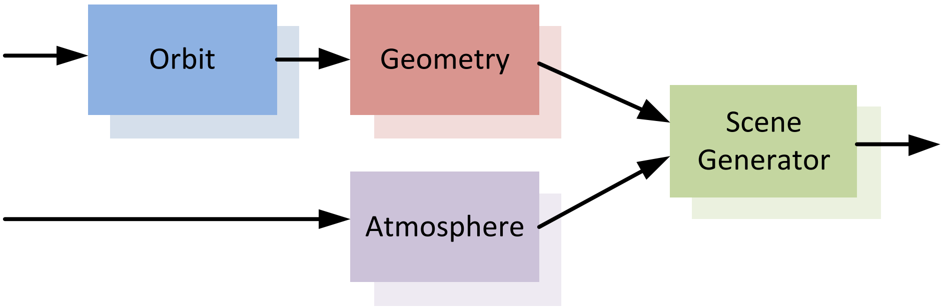


Figure 3-13: Example of a simulation

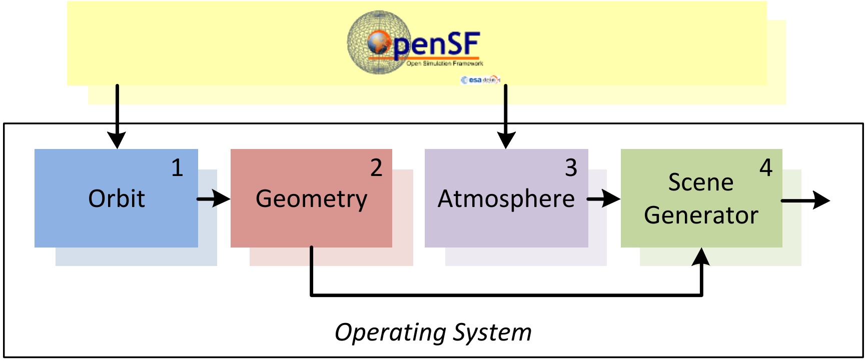


Figure 3-14: Single thread execution

Figure 3-15 below presents the same simulation execution but run in a bi-core machine with the openSF multicore support. The execution time is 3 units.

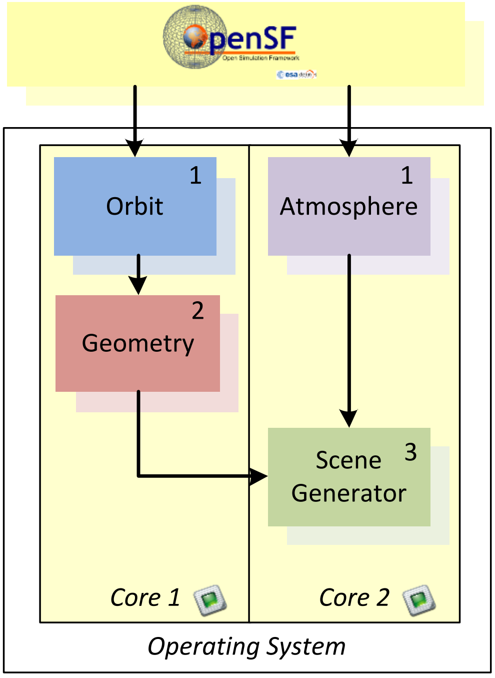


Figure 3-15: Multi-thread, parallel execution

3.6.2. Removal of logs from the database

It is proposed in [CR1] to remove the storage of logs from the database aiming to prevent the database from being overloaded due to the big size of log messages that a session/simulation execution may involve. Log sessions shall be kept as files that are stored as part of the session execution data. They can therefore be consulted at any time by users from the openSF HMI.

3.6.3. Simplification of the management of the model chains

The purpose of simplifying the management of simulations comes from users that have demanded more flexibility in the execution of a given chain. If there is a chain A+B+C, and later on, the aim is to run A+B, it becomes unfriendly the need to define a new session / simulation for A+B only.

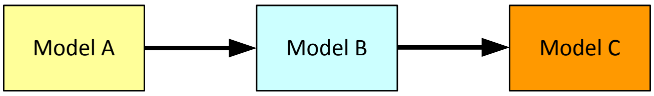


Figure 3-16 Simple model chain

Imagine the first executable (Model A) is a very time-consuming model that can be executed once to feed subsequent models. The idea is therefore to permit users to create a sub-chain with models B and C using the same simulation definition. The execution will take as input (for Model B) the outputs of model A, previously stored from another execution.

Please notice the problem can be solved with the current version of openSF, by creating two different simulations (one with only Model A and another one with Models B+C), and that is precisely something that we want to avoid with this solution.

3.6.4. Select model versions for a simulation execution

Similarly to the above, and in order to make more flexible the definition of a given simulation, it is proposed to permit to select a specific version of a model for a simulation execution.

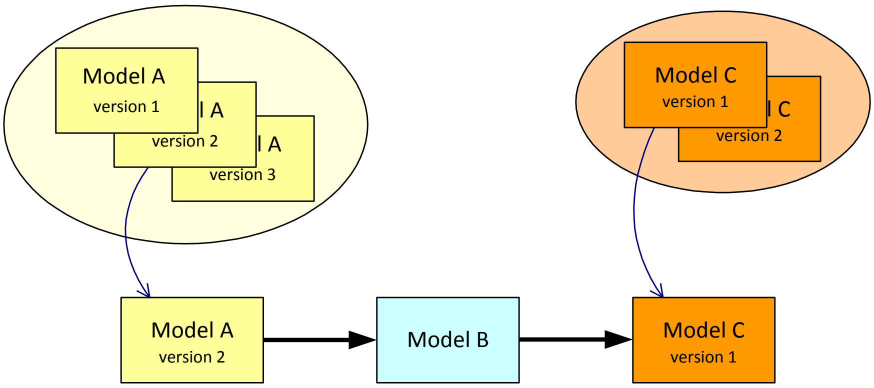


Figure 3-17: Model chain with different model versions

3.6.5. Removal of intermediate data

openSF stores all data produced as output of the running models of simulations. The purpose of this capability is to permit the possibility to user to decide whether they want to remove all the intermediate data generated as part of a simulation execution.

Thus, in the example provided in the figure below, if the user selects the removal of intermediate data, the files coloured in red would be removed once the simulation has been completed.

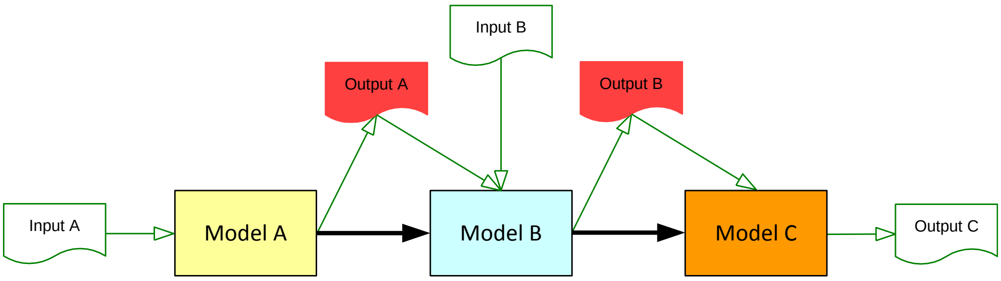


Figure 3-18: Removing intermediate data of a simulation

3.6.6. Integration of OSFEG

This capability deals with the inclusion of OSFEG into openSF. OSFEG is a set of libraries that provide primitives for the generation of analytical and stochastic perturbations, or a combination of them, that can be invoked from models participating in openSF simulations.

3.6.7. Bypass/switch-off models

This functionality enables users to switch off certain models when running simulations. As a result, openSF will inform the user of the data files needed to be provided due to the omission of models.

3.6.8. Copy elements

Functions to copy elements currently do not exist in openSF. It is certainly a useful operation that can ease the definition of simulation elements to user.

Thus, it is proposed to include new functions to copy descriptors, models, and simulations. Thus users shall only need to define a new name of the copied element.

3.6.9. Rerun a session from a previous point

This capability is very much related to the "Bypass/switch-off models" function, but in this case, the objective is to rerun a simulation avoiding a number of models from the beginning of a simulation.

It happens many times that a simulation is constituted by a very time consuming model at the beginning leading to long delays for the simulation to complete.

The idea is to avoid these situations by permitting users to skip models at the beginning of the simulations, and therefore start them from a certain point. However, the data from non-executed models is needed for the re-run. Before executing the simulations the user needs to define the data files needed for the run.

The figure below shows a simple example. Models A, B and C constitute the simulation. If we want to rerun it but starting from B, we need to provide the output of Model A, from a previous run.

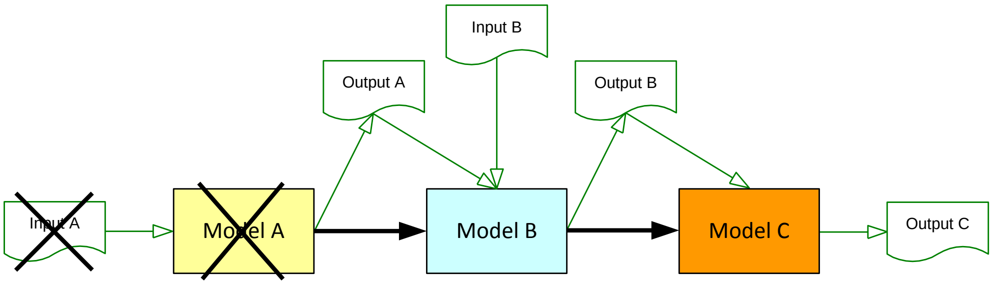


Figure 3-19: Run simulation from Model B

3.6.10. Export capability

The capability deals with the possibility of exporting the data associated to a model that has already taken place in a simulation. Thus, the data exported of the model is comprised by the configuration and the input files.

We assume that the inverse operation is also requested, meaning that a model can be imported into an openSF instance from the data obtained from the export operation. As the contents of the export relate to data files, it is required that the model exists in the target openSF instance.

3.6.11. Convert SQL syntax into XML for element definition

This capability is related to define openSF elements externally, not using the HMI. Thus, IO descriptors, models, simulations and session can be managed via XML files.

4. SYSTEM REQUIREMENTS SPECIFICATION

4.1. Requirement Naming Conventions

The following conventions will be used for the derived system requirement naming:
All system requirements will be named as

SR–XXX–NNNN/VER

where:

XXX represents the type of software requirement, which for OpenSF, are the following:

CON	–	Software Configuration and Delivery Requirements
FUN	–	Functional
IMP	–	Design and Implementation Constraints
INS	–	Adaptation and Installation
INT	–	Interfaces
MAI	–	Maintainability
OPE	–	Operational
PER	–	Performance
RES	–	Resources (Hardware and Software)
SAF	–	Safety
SEC	–	Security and Privacy
VVA	–	Verification, Validation and Acceptance

NNNN is a number providing an ordering within each requirement type. It starts at 0010 and two consecutive requirements are increased in 10 to allow the introduction of additional requirements in later versions of the document.

VER is the openSF version where the requirement was introduced or last changed.

Each requirement is presented in a tabular form constituted by four fields:

1. System requirement identifier
2. Validation method for the requirement:
   - **Test** (T) – Execution of the element under certain conditions to check the outputs corresponding to particular inputs;
   - **Inspection** (I) – Exhaustive evaluation of the code by manual reading;
   - **Analysis** (A) – Deduction method applied to documentation, code, test results, etc; it is partially or totally automated;
   - **Review** (R) – Review of project documentation.

3. Traceability to the [SOW] requirements. Additionally new requirements, not present in [SOW], have been traced to [MR01].
4. Requirement text

4.2. Functional Requirements

The OpenSF software functional requirements are basically described below according to [SOW].

4.2.1. General

SR-FUN-0010/1.0 T FUN-010/FUN-030/FUN-040/FUN-110/FUN-210

OpenSF shall provide a framework including functions to perform:

❑ Model management

❑ Simulation management

❑ Session management

❑ Session execution

❑ Session results post-processing and visualisation.

SR-FUN-0020/1.0 T FUN-070

OpenSF shall manage the addition of breakpoints during the session definition (just before the session execution).

SR-FUN-0030/1.0 FUN-040/FUN-130/FUN-150/FUN-160

OpenSF shall include a mechanism to integrate third party tools.

SR-FUN-0040/1.0 T FUN-250

OpenSF shall allow the access to the help system through a user-friendly interface.

SR-FUN-0050/1.0 R FUN-300

OpenSF shall include a splash screen system allowing the users to change the displayed screen during the start up of the application.

4.2.2. Model management

SR-FUN-0060/1.0 T FUN-170

OpenSF shall provide access to the models contained in the system.

SR-FUN-0070/1.0 T FUN-170

OpenSF shall provide a function to add a new model into the system.

SR-FUN-0080/1.0 T FUN-170

OpenSF shall accept as input for the model creation:

- ❑ Model identifier
- ❑ Model version
- ❑ Type (Stage)
- ❑ Description
- ❑ Author
- ❑ Source code
- ❑ Binary code (compiled source)
- ❑ XML file schema, including configuration parameters that defines the model's behaviour
- ❑ IO specification.

SR-FUN-0090/1.0 T FUN-170/INT-040/CON-100

OpenSF shall accept models that comply with the corresponding openSF ICD.

SR-FUN-0100/1.0 T FUN-170

DELETED

SR-FUN-0110/1.0 T FUN-170

DELETED

SR-FUN-0120/1.0 T FUN-170

DELETED

SR-FUN-0130/1.0 T FUN-170/INT-030/INT-050

OpenSF shall accept as IO specification descriptors that identify the input data files read by the model and the output data files the model produces as result.

SR-FUN-0140/1.0 T FUN-170

OpenSF shall present the list of available IO descriptors to the user when specifying the IO files of a model permitting the user to select one associated to its model's input or output data.

SR-FUN-0150/1.0 T FUN-170

OpenSF shall permit the creation of new IO descriptors in case the ones present in the system are not suitable to the model being created.

SR-FUN-0160/1.0 T FUN-170

OpenSF shall provide a function to delete the IO descriptors.

SR-FUN-0170/1.0 T FUN-170

For consistency reasons, upon the deletion of an IO descriptor, OpenSF shall remove all the references to this descriptor from the database. It includes models and consequently simulations and sessions related to these models.

SR-FUN-0180/1.0 T FUN-170

OpenSF shall accept as input for the IO descriptor creation:

- ❑ Tag (identifier) that uniquely identifies the IO descriptor within the system
- ❑ General description of the IO descriptor
- ❑ Number of files
- ❑ For each file:
  - Name of the file as read by the model
  - Description

SR-FUN-0190/1.0 T FUN-170

Upon the model addition, OpenSF shall store in the database the information associated to the model.

SR-FUN-0200/1.0 T FUN-170

OpenSF shall fail to create a model if any of the mandatory fields for the model have not been specified or the consistency checks for the model addition fail.

SR-FUN-0210/1.0 T FUN-170

OpenSF shall fail to create a model if the identifier is not unique in the system.

SR-FUN-0220/1.0 T FUN-170

OpenSF shall provide a function to modify models.

NOTE: This is understood as the modification of parameters, but not its source code and / or input and output characteristics.

SR-FUN-0230/1.0 T FUN-170

In the model modification, OpenSF shall provide the list of models available in the system.

SR-FUN-0240/1.0 T FUN-170

In the model modification, OpenSF shall accept as input only the description and/or author fields of the model.

SR-FUN-0250/1.0 T FUN-170

Upon the model modification, OpenSF shall update in the database the information associated to the model.

SR-FUN-0260/1.0 T FUN-170/FUN-100/FUN-190/FUN-290 /CON-020

OpenSF shall provide a function to add a new version of an existing model.

SR-FUN-0270/1.0 T FUN-170/FUN-100/FUN-190/FUN-290 /CON-020

In the model version addition, OpenSF shall provide the list of models available in the system.

SR-FUN-0280/1.0 T FUN-170/FUN-100/FUN-190/FUN-290 /CON-020

In the model version addition, OpenSF shall accept as input:

- ❑ Model version
- ❑ Description
- ❑ Author
- ❑ Source code
- ❑ Binary code

SR-FUN-0290/1.0 T FUN-170/FUN-100/FUN-190/FUN-290 /CON-020

Upon the model version addition, OpenSF shall update in the database the information associated to the model.

SR-FUN-0300/1.0 T FUN-170/FUN-100/FUN-190/FUN-290 /CON-020

OpenSF shall fail to add a new version of a model if the identifier/version couple is not unique in the system.

SR-FUN-0310/1.0 T FUN-170

OpenSF shall provide a function to delete models.

SR-FUN-0320/1.0 T FUN-170

OpenSF shall accept as input for the model deletion the model identifier and version.

SR-FUN-0330/1.0 T FUN-170

For consistency reasons, upon a model deletion, OpenSF shall remove all the data belonging to the model from the database. This includes the simulations and sessions where the deleted model is participating, and consequently all the session results produced by previous runs, as well as the IO descriptor.

4.2.2.1. Stage Management

SR-FUN-0340/1.0 T FUN-170/CON-140

OpenSF shall provide access to the stages contained in the system.

SR-FUN-0350/1.0 T FUN-170/CON-140

OpenSF shall provide a function to create a new stage.

SR-FUN-0360/1.0 T FUN-170/CON-140

OpenSF shall accept as input for the stage creation:

- ❑ Stage identifier
- ❑ Description
- ❑ Stage position within the simulation execution.

SR-FUN-0370/1.0 T FUN-170/CON-140

OpenSF shall fail to add a new version of a model if the identifier/version couple is not unique in the system.

SR-FUN-0380/1.0 T FUN-170/CON-140

Upon the stage creation, OpenSF shall create a new entry in the database with the information associated to the new stage.

SR-FUN-0390/1.0 T FUN-170/CON-140

OpenSF shall provide a mechanism to edit a stage. The edition capabilities are:

- ❑ Change the description of the stage.
- ❑ Change the position of the stage within the simulation execution.

SR-FUN-0400/1.0 T FUN-170/CON-140

Upon the stage edition, OpenSF shall update in the database the information associated to the model.

SR-FUN-0410/1.0 T FUN-170/CON-140

OpenSF shall provide a function to delete a stage from the system.

SR-FUN-0420/1.0 T FUN-170/CON-140

For consistency reasons, upon a stage deletion or a change of position within the simulation execution OpenSF shall remove all the data belonging to the stage from the database. This includes the models where the stage is participating, and consequently the simulations, sessions and results produced by previous runs.

4.2.3. Simulation management

SR-FUN-0430/1.0 T FUN-110

OpenSF shall provide access to the simulations contained in the system.

SR-FUN-0440/1.0 T FUN-110

OpenSF shall provide a function to create a new simulation.

SR-FUN-0450/1.0 T FUN-110

OpenSF shall accept as input for the simulation creation:

- Simulation identifier
- Description
- Author
- List of models participating in the simulation

SR-FUN-0460/1.0 T FUN-110

For the model selection in the simulation creation, OpenSF shall present the list of available models that can be connected to the most immediate predecessor model in the simulation. This shall be done according to the defined Input descriptor of the model. In case of being the first model in the simulation, all models shall be presented.

SR-FUN-0470/1.0 T FUN-110

OpenSF shall be able to define the starting and ending stage of the simulation. Starting and ending stage can be equal (e.g., testing of only one model)

SR-FUN-0480/1.0 T FUN-110

OpenSF shall present the target models to take part in the simulation in the order they were selected.

SR-FUN-0490/1.0 T FUN-110

Upon the simulation creation, openSF shall store in the database the information associated to the simulation.

SR-FUN-0500/1.0 T FUN-110

OpenSF shall fail to create a simulation if the identifier is not unique in the system.

SR-FUN-0510/1.0 T FUN-110

OpenSF shall fail to create a simulation if any of the mandatory fields for the simulation have not been specified or the consistency checks [1]for the simulation creation fail.

SR-FUN-0520/1.0 T FUN-110

OpenSF shall provide a function to modify simulations.

SR-FUN-0530/1.0 T FUN-110

In the simulation modification, openSF shall accept as input:

- ❑ Description
- ❑ Author

SR-FUN-0540/1.0 T FUN-110

In the simulation modification, OpenSF shall present the information associated to a simulation so that it can be modified by the user, with the exception of the identifier.

SR-FUN-0550/1.0 T FUN-110

Upon the simulation modification, OpenSF shall update in the database the information associated to the simulation.

SR-FUN-0560/1.0 T FUN-110

[1] Consistency checks referring to the expected inputs by a model are the only responsibility of OpenSF. It is up to the error handling within the model to evaluate consistency among the functions that belong to a model.

OpenSF shall provide a function to delete simulations.

SR-FUN-0570/1.0 T FUN-110

OpenSF shall accept as input for the simulation deletion the simulation identifier.

SR-FUN-0580/1.0 T FUN-110

For consistency reasons, upon a simulation deletion, OpenSF shall remove all the data belonging to the simulation from the database. This refers to all sessions that contain the simulation and the results produced by previous runs of the simulation.

4.2.4. Session Management

SR-FUN-0590/1.0 T FUN-110/FUN-210

OpenSF shall provide access to the sessions contained in the system.

SR-FUN-0600/1.0 T FUN-110/FUN-210

OpenSF shall provide a function to create a new session.

SR-FUN-0610/1.0 T FUN-030/FUN-090/FUN-110/FUN-210

OpenSF shall accept as input for the simulation creation:

- ❑ Session identifier
- ❑ Description
- ❑ Author
- ❑ List of simulations.
- ❑ List of corresponding model configuration files used by each model
- ❑ Global parameter file, common to all models in a simulation chain.
- ❑ List of input files (in case of models with IO descriptors different from DEFAULT).

SR-FUN-0620/1.0 T FUN-110/FUN-180/FUN-210

For the simulation selection in the session creation, OpenSF shall present the list of available simulations within the system. OpenSF shall provide a function to add and remove simulations from a session.

SR-FUN-0630/1.0 T FUN-110/FUN-210

Upon the session creation, OpenSF shall store in the database the information associated to the session.

SR-FUN-0640/1.0 T FUN-110/FUN-210

Upon the session creation, OpenSF shall store in a temporal directory the shell script associated to the simulation for the batch execution.

SR-FUN-0650/1.0 T FUN-110/FUN-210

OpenSF shall fail to create a session if the identifier is not unique in the system.

SR-FUN-0660/1.0 T FUN-110/FUN-210

OpenSF shall fail to create a session if any of the mandatory fields for the session have not been specified.

SR-FUN-0670/1.0 T FUN-110/FUN-210

OpenSF shall provide a function to modify sessions.

SR-FUN-0680/1.0 T FUN-030/FUN-110/FUN-210/CON-010

In the session modification, OpenSF shall accept as input:

- Description
- Author
- List of simulations.
- List of corresponding model configuration files used by each model
- Global parameter file, common to all models in a simulation chain.
- List of input files.
- List of output files.

SR-FUN-0690/1.0 T FUN-110/FUN-210

In the session modification, openSF shall present the information associated to a session so that it can be modified by the user, with the exception of the identifier.

SR-FUN-0700/1.0 T FUN-110/FUN-210

In the session modification, openSF shall provide a functionality to save the modified session as a new one with a different identifier entered by the user.

SR-FUN-0710/1.0 T FUN-110/FUN-170/FUN-210

OpenSF shall present values of the model parameters (global and local) involved in the session that is being modified.

SR-FUN-0720/1.0 T FUN-110/FUN-210

Upon the session modification, OpenSF shall update in the database the information associated to the simulation.

SR-FUN-0730/1.0 T FUN-110/FUN-210

OpenSF shall provide a function to delete sessions.

SR-FUN-0740/1.0 T FUN-110/FUN-210

OpenSF shall accept as input for the session deletion the session identifier.

SR-FUN-0750/1.0 T FUN-110/FUN-210

For consistency reasons, upon a session deletion, OpenSF shall remove all the data belonging to the session from the database. This refers to all the results produced by previous runs of the session.

SR-FUN-0760/1.0 T FUN-130/FUN-210

In the session creation and modification openSF shall provide a function to schedule the execution of third party viewers.

SR-FUN-0770/1.0 T FUN-070/FUN-210

OpenSF shall provide a mechanism to schedule breakpoints during the session execution.

4.2.4.1. Model Parameters Management

SR-FUN-0780/1.0 T FUN-170/CON-060

OpenSF shall provide a function to edit the model parameters.

SR-FUN-0790/1.0 T FUN-170/CON-050

Upon model parameter edition, OpenSF shall perform a consistency and out-of range checking.

SR-FUN-0800/1.0 T FUN-170/CON-050

OpenSF shall present a warning message if the out-of range checking fails.

SR-FUN-0810/1.0 T FUN-170/CON-050

OpenSF shall not allow the user to enter a parameter if the consistency checking fails.

SR-FUN-0820/1.0	T	FUN-050/CON-060

OpenSF shall provide a function to iterate model parameters in order to support Monte Carlo analysis.

SR-FUN-0830/1.0	T	FUN-050/CON-060

OpenSF shall execute as many simulations as iterations of the model parameters in order to perform a Monte Carlo analysis of the results.

4.2.5. Session execution

SR-FUN-0840/1.0	T	FUN-210

OpenSF shall provide a function to run a session.

SR-FUN-0850/1.0	T	FUN-210

OpenSF shall present the list of sessions, enabling the user to select the desired ones for execution.

SR-FUN-0860/1.0	T	FUN-210

OpenSF shall accept as input for the session execution:

- ❑ Session identifier (mandatory)
- ❑ Author (optional)
- ❑ Textual description, detailing the purpose of the simulation (optional)

SR-FUN-0870/1.4	R	FUN-210

Upon the simulation creation, OpenSF shall store in a temporal directory the shell script associated to the simulation for the batch execution.

SR-FUN-0880/1.0	T	FUN-210

OpenSF shall display log messages informing the user about the simulation execution.

SR-FUN-0890/1.0	T	FUN-210

OpenSF shall provide a function to abort a running simulation.

SR-FUN-0900/1.0	T	FUN-210

OpenSF shall check the presence in the file system of session input files, model configuration files and global configuration file.

SR-FUN-0910/1.0 T FUN-210

OpenSF shall warn the user about session missing files.

SR-FUN-0920/1.0 T FUN-210/FUN-070

OpenSF shall stop the execution in a controlled manner due to a scheduled breakpoint.

SR-FUN-0930/1.0 T FUN-210/FUN-070

OpenSF shall provide a function to resume a previously stopped simulation.

4.2.6. Session results visualisation

SR-FUN-0940/1.0 T FUN-160/FUN-210

OpenSF shall provide a function to visualise the results of a simulation.

SR-FUN-0950/1.0 T FUN-160/FUN-210

OpenSF shall present the list of sessions that have been executed at least once, enabling the user to select the desired one for results visualisation.

SR-FUN-0960/1.0 T FUN-160/FUN-210

OpenSF shall accept as input for the visualisation of results:

- ❑ Session identifier
- ❑ Date and time when the simulation was run

SR-FUN-0970/1.0 T FUN-160/FUN-210

Upon selection of a session OpenSF shall present the following information to the user:

- ❑ Global simulation results
- ❑ Model results
- ❑ Log

SR-FUN-0980/1.0 T FUN-160/FUN-210

The “General session results” option shall present generic information to all sessions:

- ❑ Session identifier
- ❑ Date and time it was executed
- ❑ Duration (in minutes and integer seconds) of the simulation

- ❑ Result of the session: Successful, Failed or Undone.
- ❑ The different comparison procedures that were applied at simulation execution time and those that can be applied upon finalisation (product tools).

SR-FUN-0990/1.0 T FUN-040/FUN-130/FUN-150/FUN-210

OpenSF shall provide a mechanism to run third party tools to post-process/visualize the session results.

SR-FUN-1000/1.0 T FUN-130/FUN-140/ FUN-210

OpenSF shall provide data comparison procedures through the use of third party tools.

SR-FUN-1010/1.0 T FUN-120

Upon selection of the “Log” option OpenSF shall present the list of log messages generated by the monitoring and control function of the simulation run plus any exception reported by the models.

SR-FUN-1020/1.0 T FUN-120

The information contained in each log message shall be

- ❑ Date and time of the log message generation
- ❑ Severity level of the message (I=Info, W=Warning, E=Error)
- ❑ Source: model and function within the model that triggers the exception
- ❑ Message text

4.2.7. Web site access

SR-FUN-1030/1.0 T [SOW] §3.2

OpenSF shall provide a web site for registered users access.

SR-FUN-1040/1.0 T [SOW] §3.2

OpenSF web site shall allow adding “issues” upon the latest delivery of the openSF framework. These issues can be formal SPR encountered by users or to make suggestions for possible improvements.

SR-FUN-1050/1.0 T [SOW] §3.2

Users shall add/view issues using the Mantis system, accessed through the openSF web site.

SR-FUN-1060/1.0 T [SOW] §3.2

OpenSF web site shall allow users to download the latest delivery of the openSF tool.

SR-FUN-1070/1.0 T [SOW] §3.2

New (non-registered) users shall be able to request to be registered into the openSF web site.

SR-FUN-1080/1.0 T [SOW] §3.2

Requests from non-registered users shall be sent to the openSF team at DEIMOS and to a dedicated ESA user (configurable).

NOTE: Upon acceptance of the request, ESA shall inform DEIMOS to proceed with the user registration.

SR-FUN-1090/1.0 T [SOW] §3.2

New registered users shall be informed via e-mail that they have been authorised to access the openSF A dedicated ESA user (configurable) shall receive requests from non-registered users.

NOTE: Upon acceptance of the request, ESA shall communicate with DEIMOS to proceed with the user registration.

4.2.8. Multiple repository capability

SR-FUN-1100/2.0 T [MR01]

openSF shall allow the possibility to define one or more processing chains within the same openSF instance.

This change implies the development of a simple database manager that handles the different simulation scenarios for every mission. The database shall therefore handle more than one processing chain allowing users to switch between missions at the openSF startup or when it is demanded

SR-FUN-1110/2.0 T [MR01]

openSF shall distinguish each processing chain by a unique identifier.

4.2.9. Parameter editor

SR-FUN-1120/2.0 T [MR01]/CON-030/CON-040

The parameter editor shall allow managing parameter files whose structure is defined using models. These models shall allow to define:

- ❑ Conditional expressions linking a parameter attribute with another parameter attribute.
- ❑ Links towards other parameters files with their associated models. These models shall include for each parameter:
  - Identification (understandable name),
  - Type (chain of characters, Boolean, list of values, short integer, real, character, etc),
  - Dimension,
  - Default value (optional),

- Minimum and maximum range (optional),
- Commentary or help text (optional)

SR-FUN-1130/2.0 T [MR01]/CON-030/CON-040

The parameter editor shall allow guiding and controlling the user's input during the creation and the modification of parameters files to avoid the production of inconsistent files.

SR-FUN-1140/2.0 T [MR01]/CON-030/CON-040

For each parameter, the editor shall display its characteristics (identification, type, dimension, default value, validity range, … see requirement above). For each input value, the editor shall perform validity checks wrt the parameter characteristics (e.g. compliance of the input value with the parameter type; compliance of the input value with the validity range)

SR-FUN-1150/2.0 T [MR01]

The parameter editor shall allow editing parameter files contained within a simulation configuration file (Ex: A model configuration file containing a FILE parameter with other parameters inside it).

SR-FUN-1160/2.0 T [MR01]

The parameter editor shall follow dynamically the structure defined in the model, for example:

❑ If the user changes the number of iterations for a loop, the editor shall generate the appropriate number of sets of parameters

❑ If the user changes the condition for a switchable set of parameters, the editor shall generate the corresponding set of parameters.

SR-FUN-1170/2.0 T [MR01]

The parameter editor shall provide edition facilities such as sequential and guided edition for creation and modification, search, replace, cut/copy/paste.

SR-FUN-1180/2.2 T [MR02].2

The parameter editor shall be integrated within openSF.

SR-FUN-1190/2.2 T [MR02].2

The parameter editor shall be invoked by clicking a dedicated button, part of openSF GUI.

4.2.10. Parameter perturbation

SR-FUN-1200/2.2 T [MR02].1

OpenSF shall allow to associate perturbation functions to model configuration parameters enabling multiple runs for that model during a simulation execution.

SR-FUN-1210/2.2 T [MR02] .1

Upon the selection of the parameter to be perturbed OpenSF shall present a window containing the following information to fully define the error to be introduced to the parameter:

- ❑ Perturbation function/type
- ❑ Perturbation parameters, depending on the selected perturbation type
- ❑ Number of shots, this is, the number of times the model is to be run.

SR-FUN-1220/2.2 T [MR02] .1

The perturbation functions applicable to parameters shall be:

- ❑ Deterministic functions
- ❑ Sampling functions: linear, polynomial and spline sampling
- ❑ Non-deterministic functions:
  - Beta Distribution
  - Gamma Distribution
  - Exponential Distribution
  - Normal Distribution
  - Uniform Distribution
  - Poisson Distribution
  - Truncated Gaussian Distribution
  - Uniform Discrete Distribution
  - Distribution with custom Probability Density Function
- ❑ Basic mathematical operations

SR-FUN-1230/2.2 T [MR02] .1

OpenSF shall present as output one of the following options (previously selected by the user) upon the addition of a perturbation to a model or set of models:

- ❑ One simulation including as many executions of the perturbed models as indicated by the number of shorts for each one.
- ❑ The number of simulation executions equal to the number of shots defined within the model perturbation.

4.2.11. Tool extension management

SR-FUN-1240/2.2 T [MR02].3

OpenSF shall allow to associate more than one file extension to a single tool.

4.2.12. Output files check

SR-FUN-1250/2.2 T [MR02].4

OpenSF shall check at simulation time that models write the output files at the specified locations.

SR-FUN-1260/2.2 T [MR02].4

In case files are not generated at the specified location, openSF shall raise a Warning log message in the Session Execution window indicating such circumstance.

4.2.13. Extended log capabilities

SR-FUN-1270/2.2 T [MR02].5

OpenSF shall capture errors generated by MATLAB (i.e. out of memory) in terms of log messages.

SR-FUN-1280/2.2 T [MR02].11

OpenSF shall store the log messages per model.

SR-FUN-1290/2.2 T [MR02].6

OpenSF shall log the execution time of all models participating in a running session.

SR-FUN-1300/2.2 T [MR02].7

The date and time format of log messages shall be YYYY-MM-DD HH:mm:SS:sss.

SR-FUN-1310/2.2 T [MR02].8

The date and time format of log session name shall be YYYY-MM-DD HH:mm:SS:sss.

SR-FUN-1320/2.2 T [MR02].10

OpenSF shall include a log function in the C OSFI library that displays log messages including variable fields.

SR-FUN-1330/2.2 T [MR02].12

OpenSF shall allow users to maximise the log window for a better comprehension of the messages.

SR-FUN-1340/2.2 T [MR02].14

OpenSF shall capture the exception messages that lead to the termination of the model executable and shall present them in the log.

SR-FUN-1345/3.0 T [CR1].2

OpenSF shall output the log into a file in the session’s output directory.

4.2.14. Shortcuts activation

SR-FUN-1350/2.2 T [MR02].15

OpenSF shall provide the means to carry out the most representative operations through the activation of keyboard shortcuts:

- ❑ System Menu
  - Ctrl+T: Tools
  - Ctrl+O: Configuration
  - Ctrl+P: Parameter Editor
  - Ctrl+H: Help
  - Ctrl+A: About openSF
  - Ctrl+X: Exit openSF
- ❑ Repository menu
  - Ctrl+D: Descriptors
  - Ctrl+G: Stages
  - Ctrl+M: Models
  - Ctrl+I: Simulations
  - Ctrl+S: Sessions
- ❑ Executions menu
  - Ctrl+R: Executions Results
  - Ctrl+L: Logs

4.2.15. Backup capability

SR-FUN-1360/2.2 T [MR02].9

OpenSF shall provide the means to perform a backup of the whole openSF data repository.

SR-FUN-1370/2.2 T [MR02].9

The outcome of the backup operation shall be a SQL script ready to be executed in any openSF instance.

4.2.16. Import/Export capability

SR-FUN-1380/2.2 T [MR02].9

OpenSF shall include import and export capabilities related to sessions.

SR-FUN-1390/2.2 T [MR02].9

OpenSF shall provide a function to export a session.

SR-FUN-1400/2.2 T [MR02].9

The input data needed to perform an export is the identifier of the session.

SR-FUN-1410/2.2 T [MR02].9

The output of an export operation is an SQL script including all elements of the session and a zip files with the data files needed for its execution.

SR-FUN-1420/2.2 T [MR02].9

OpenSF shall provide a function to import a session.

SR-FUN-1430/2.2 T [MR02].9

The input data needed to perform an import are the SQL file and the compresses file produced from the export capability.

SR-FUN-1440/2.2 T [MR02].9

The output of an import operation is the creation of a new session with the data included in the SQL script, and, in case of an executed session, the creation of a zip file including all IO files.

4.2.17. Parallelization

SR-FUN-1450/3.0 T [CR1].1

OpenSF shall parallelize the execution of simulations by parallelizing the execution of the models that comprise a simulation session.

SR-FUN-1460/3.0 T [CR1].1

When associating perturbation functions to model configuration parameters (enabling multiple runs for a given model during a simulation execution) openSF shall allow the parallelization of model execution shots.

SR-FUN-1470/3.0 T [CR1].1

When associating perturbation functions to model configuration parameters (enabling multiple runs for a given model during a simulation execution) openSF shall ask the user if parallelization of model execution shots should be performed.

4.2.18. Flexible Session Management

SR-FUN-1480/3.0 T [CR1].3

OpenSF shall allow the user to execute a sub-chain of a given simulation chain based on a previously executed session.

SR-FUN-1490/3.0 T [CR1].4

OpenSF shall allow the user to select a model version before launching a session execution.

SR-FUN-1500/3.0 T [CR1].5

OpenSF shall allow the user to select if the intermediate data generated during a session execution shall be deleted.

SR-FUN-1510/3.0 T [CR1].5

In case the removal of intermediate data has been selected, the output of the session shall be constituted by the last models output files.

SR-FUN-1520/3.0 T [CR1].7

OpenSF shall allow the user to bypass or switch-off a model during a session execution.

SR-FUN-1530/3.0 T [CR1].9

OpenSF shall allow the user to select a given point (identified by a model) in a simulation chain from where to re-execute a previously executed session.

4.2.19. Model Export/Import capability

SR-FUN-1540/3.0 T [CR1].10

OpenSF shall include import and export capabilities related to models relative to a particular session.

SR-FUN-1550/3.0 T [CR1].10

OpenSF shall provide a function to export a model from a session.

SR-FUN-1560/3.0 T [CR1].10

The input data needed to perform an export of a model is the identifier of the model and the session identifier where the model has been run.

SR-FUN-1570/3.0 T [CR1].10

The output of a model export operation is a compressed file with the data files needed for the model execution.

SR-FUN-1580/3.0 T [CR1].10

OpenSF shall provide a function to import a model as part of the definition of a given session.

SR-FUN-1590/3.0 T [CR1].10

The input data needed to perform a model import are compressed file produced from the model export capability.

SR-FUN-1600/3.0 T [CR1].10

openSF shall check that the import of the model and data ensures the integrity of the session.

SR-FUN-1610/3.0 T [CR1].10

The result of a model import operation is the update of the data files supporting a model execution into an existing session containing the model.

4.2.20. Copying openSF elements

SR-FUN-1620/3.0 T [CR1].8

OpenSF shall provide an operation to copy openSF element definitions through the HMI.

SR-FUN-1630/3.0 T [CR1].8

openSF shall allow to copy the following elements: IO descriptors, tools, models, simulations and sessions.

4.2.21. Defining openSF elements externally

SR-FUN-1640/3.0 T [CR1].11

OpenSF shall allow the definition of elements externally to the application by means of a file.

SR-FUN-1650/3.0 T [CR1].11

The file containing openSF elements definitions shall be an XML file.

SR-FUN-1660/3.0 T [CR1].11

openSF shall provide an operation for importing openSF element definitions from file.

SR-FUN-1670/3.0 T [CR1].11

The openSF elements that can be defined externally to the application are: IO descriptors, tools, stages, models, simulations, sessions and parameter perturbations.

4.2.22. Error Generation capabilities

SR-FUN-1680/3.0 T [CR1].6

OpenSF shall include error generation functions in the C++ OSFEG library.

SR-FUN-1690/3.0 T [CR1].6

OpenSF shall include the following deterministic error generation functions in the OSFEG library: affine, bias, linear, parabolic, polynomial, step, sinusoidal and tangent.

SR-FUN-1700/3.0 T [CR1].6

OpenSF shall include the following sampling functions in the OSFEG library: linear sampling, polynomial sampling and spline sampling.

SR-FUN-1710/3.0 T [CR1].6

OpenSF shall include the following non-deterministic error generation functions in the OSFEG library: beta distribution, gamma distribution, exponential distribution, normal distribution, uniform distribution, Poisson distribution, truncated Gaussian distribution, uniform discrete distribution and Distribution with custom Probability Density Function.

SR-FUN-1720/3.0 T [CR1].6

OpenSF shall include the following binary and composite operations in the OSFEG library: Addition, Subtraction, Multiplication, Division, Exponentiation and Root.

SR-FUN-1730/3.0 T [CR1].6

OpenSF error generation functions in the OSFEG library shall be configured through an XML file.

4.3. Interface Requirements

SR-INT-0010/1.0 T INT-010

The interfaces between the different OpenSF models shall be developed according to openSF ICD

SR-INT-0020/1.0 T FUN-080/INT-030/INT-040

A model shall interface with OpenSF through input, output and configuration XML and XSD files.

SR-INT-0030/1.0 T FUN-080

During simulation execution, the communication between OpenSF and a model shall be done through log messages in the standard output stream (normal *output* destination of a program).

4.4. Operational Requirements

SR-OPE-0010/1.0 T None

OpenSF shall provide a Human-Machine Interface (HMI) including all functionality.

SR-OPE-0020/1.0 T FUN-060

OpenSF shall allow the execution of a simulation in two modes:

❑ Interactively, through the use of the HMI

❑ In batch mode, by means of executing a shell script command.

SR-OPE-0030/1.0 T FUN-060

For running a simulation in batch mode, the user shall execute the corresponding script.

SR-OPE-0040/1.0 T FUN-060

In batch mode, the execution of the script shall have the following usage format:

opensf -x “sessionId”, where sessionId = sessionName (date)

the session identifier is stored in the OpenSF database.

SR-OPE-0050/1.0 T FUN-050

It shall be possible to indicate the number of times the simulation shall be run according to either:

- □ A given input parameter that varies from one execution to another
- □ A list of input files.

SR-OPE-0060/1.0 T [MR02].13

It shall be possible to close the openSF GUI without killing running simulations.

SR-OPE-0070/1.0 T [MR02].13

When closing the GUI when a simulation is running, openSF shall warn the user presenting a dialogue window to cancel or continue with the action.

4.5. Resources Requirements

SR-RES-0020/1.0 T None

OpenSF shall provide a Human-Machine Interface (HMI) including all functionality based on the Java Swing library.

SR-RES-0030/1.0 T None

OpenSF data repository shall be based on MySQL.

SR-RES-0040/1.0 T CON-100

OpenSF shall accept models written in F90, C and C++.

SR-RES-0050/2.0 T CON-110

OpenSF shall accept models written in F77, Matlab and IDL.

4.6. Design and Implementation Constraints

SR-IMP-0010/1.0 R None

OpenSF development process shall be based on [DDVP].

SR-IMP-0020/1.0 R None

The OpenSF requirement analysis shall be performed using UML.

SR-IMP-0030/1.0 R None

The OpenSF design shall be performed using UML

SR-IMP-0040/1.0 R None

OpenSF shall employ a CASE tool for the OpenSF requirement analysis and design.

SR-IMP-0040/1.0 T CON-090

OpenSF shall support the following operating systems:

- Linux on SUSE / OpenSUSE 11.1

SR-IMP-0040/1.0 R CON-160

OpenSF additional libraries shall be compiled with the latest version of gcc 4.3.4.

SR-IMP-0050/1.0 I None

The HMI shall be built using the using the Java Swing technology.

SR-IMP-0060/1.0 I None

OpenSF shall make use of standard (JDBC) Java programming language for the interaction with RDBMS.

SR-IMP-0070/1.0 R CON-080

OpenSF shall run on both single and multiprocessor machines

SR-IMP-0080/1.0 R CON-150

OpenSF shall support Intel hardware architecture.

SR-IMP-0090/1.0 I CON-130

The OpenSF infrastructure source code shall have a common header template describing terms of use, license and versioning.

SR-IMP-0100/1.0 I CON-070

OpenSF shall not hard-code any simulation parameter

4.7. Security and Privacy Requirements

SR-SEC-0010/1.0 R, T None

OpenSF shall be designed so that operations to modify or delete data require confirmation of the user.

SR-SEC-0020/1.0 R None

OpenSF shall support only one type of user, with full access to the system.

4.8. Maintainability Requirements

SR-MAI-0010/3.0 R None

The textual description of the following manual procedures for the OpenSF maintenance shall be provided:

- ❑ Backup/restore
- ❑ Data management
- ❑ Software maintenance
- ❑ Migration to v3 procedure.

4.9. Safety Requirements

SR-SAF-0010/1.0 R None

OpenSF shall be designed in such a way that any application crash shall not cause to have corrupted data in the database.

SR-SAF-0080/1.0 R CON-120

OpenSF shall be thread safe.

4.10. Software Configuration and Delivery Requirements

SR-CON-0010/1.0 R None

The configuration aspects applicable for the project shall be defined in the Software Configuration Management Plan.

4.11. Adaptation and Installation Requirements

SR-INS-0010/1.0 T FUN-200/CON-090

OpenSF shall provide an automatic procedure to install the simulator for each supported platforms. The supported platforms are:

- ❑ Linux on SUSE / OpenSUSE 11.1

4.12. Verification, Validation and Acceptance Requirements

SR-VVA-0010/1.0 T None

OpenSF shall use an efficient range of representative scenes with appropriate data files for testing purposes.

SR-VVA-0040/1.0 T None

Verification at the model-level shall be intended to test one single model.

SR-VVA-0080/1.0 R None

OpenSF shall be formally accepted on the target platform at ESTEC.

5. TRACEABILITY MATRICES

5.1. Direct Traceability

This matrix maps the traceability from the User Requirements described in [SOW] to the System Requirements (SR). The table also collects the SR that are not a direct result from user requirements including an explanation in the Comment column.

RB identifier	SR identifier	Comment
FUN-010	SR-FUN-0010/1.0	
FUN-020	--	
FUN-030	SR-FUN-0010/1.0	
FUN-030	SR-FUN-0610/1.0	
FUN-030	SR-FUN-0680/1.0	
FUN-040	SR-FUN-0010/1.0	
FUN-040	SR-FUN-0030/1.0	
FUN-040	SR-FUN-0990/1.0	
FUN-050	SR-FUN-0820/1.0	
FUN-050	SR-FUN-0830/1.0	
FUN-060	SR-OPE-0020/1.0	
FUN-060	SR-OPE-0030/1.0	
FUN-060	SR-OPE-0040/1.0	
FUN-070	SR-FUN-0020/1.0	
FUN-070	SR-FUN-0770/1.0	
FUN-070	SR-FUN-0920/1.0	
FUN-070	SR-FUN-0930/1.0	
FUN-080	SR-INT-0020/1.0	
FUN-080	SR-INT-0030/1.0	
FUN-090	SR-FUN-0610/1.0	
FUN-100	SR-FUN-0260/1.0	
FUN-100	SR-FUN-0270/1.0	
FUN-100	SR-FUN-0280/1.0	
FUN-100	SR-FUN-0290/1.0	
FUN-100	SR-FUN-0300/1.0	
FUN-110	SR-FUN-0010/1.0	

RB identifier	SR identifier	Comment
FUN-110	SR-FUN-0430/1.0	
FUN-110	SR-FUN-0440/1.0	
FUN-110	SR-FUN-0450/1.0	
FUN-110	SR-FUN-0460/1.0	
FUN-110	SR-FUN-0470/1.0	
FUN-110	SR-FUN-0480/1.0	
FUN-110	SR-FUN-0490/1.0	
FUN-110	SR-FUN-0500/1.0	
FUN-110	SR-FUN-0510/1.0	
FUN-110	SR-FUN-0520/1.0	
FUN-110	SR-FUN-0530/1.0	
FUN-110	SR-FUN-0540/1.0	
FUN-110	SR-FUN-0550/1.0	
FUN-110	SR-FUN-0560/1.0	
FUN-110	SR-FUN-0570/1.0	
FUN-110	SR-FUN-0580/1.0	
FUN-110	SR-FUN-0590/1.0	
FUN-110	SR-FUN-0600/1.0	
FUN-110	SR-FUN-0610/1.0	
FUN-110	SR-FUN-0620/1.0	
FUN-110	SR-FUN-0630/1.0	
FUN-110	SR-FUN-0640/1.0	
FUN-110	SR-FUN-0650/1.0	
FUN-110	SR-FUN-0660/1.0	
FUN-110	SR-FUN-0670/1.0	
FUN-110	SR-FUN-0680/1.0	
FUN-110	SR-FUN-0690/1.0	
FUN-110	SR-FUN-0700/1.0	
FUN-110	SR-FUN-0710/1.0	
FUN-110	SR-FUN-0720/1.0	
FUN-110	SR-FUN-0730/1.0	
FUN-110	SR-FUN-0740/1.0	
FUN-110	SR-FUN-0750/1.0	

RB identifier	SR identifier	Comment
FUN-120	SR-FUN-1010/1.0	
FUN-120	SR-FUN-1020/1.0	
FUN-130	SR-FUN-0030/1.0	
FUN-130	SR-FUN-0760/1.0	
FUN-130	SR-FUN-0990/1.0	
FUN-130	SR-FUN-1000/1.0	
FUN-140	SR-FUN-1000/1.0	
FUN-150	SR-FUN-0030/1.0	
FUN-150	SR-FUN-0990/1.0	
FUN-160	SR-FUN-0030/1.0	
FUN-160	SR-FUN-0940/1.0	
FUN-160	SR-FUN-0950/1.0	
FUN-160	SR-FUN-0960/1.0	
FUN-160	SR-FUN-0970/1.0	
FUN-160	SR-FUN-0980/1.0	
FUN-170	SR-FUN-0060/1.0	
	SR-FUN-0070/1.0	
	SR-FUN-0080/1.0	
	SR-FUN-0090/1.0	
	SR-FUN-0130/1.0	
	SR-FUN-0140/1.0	
	SR-FUN-0150/1.0	
	SR-FUN-0160/1.0	
	SR-FUN-0170/1.0	
	SR-FUN-0180/1.0	
	SR-FUN-0190/1.0	
	SR-FUN-0200/1.0	
	SR-FUN-0210/1.0	
	SR-FUN-0220/1.0	
	SR-FUN-0230/1.0	
	SR-FUN-0240/1.0	
	SR-FUN-0250/1.0	
	SR-FUN-0260/1.0	

RB identifier	SR identifier	Comment
	SR-FUN-0270/1.0	
	SR-FUN-0280/1.0	
	SR-FUN-0290/1.0	
	SR-FUN-0300/1.0	
	SR-FUN-0310/1.0	
	SR-FUN-0320/1.0	
	SR-FUN-0330/1.0	
	SR-FUN-0340/1.0	
	SR-FUN-0350/1.0	
	SR-FUN-0360/1.0	
	SR-FUN-0370/1.0	
	SR-FUN-0380/1.0	
	SR-FUN-0390/1.0	
	SR-FUN-0400/1.0	
	SR-FUN-0410/1.0	
	SR-FUN-0420/1.0	
	SR-FUN-0710/1.0	
	SR-FUN-0780/1.0	
	SR-FUN-0790/1.0	
	SR-FUN-0800/1.0	
	SR-FUN-0810/1.0	
FUN-180	SR-FUN-0620/1.0	
FUN-190	SR-FUN-0260/1.0	
	SR-FUN-0270/1.0	
	SR-FUN-0280/1.0	
	SR-FUN-0290/1.0	
	SR-FUN-0300/1.0	
FUN-200	SR-INS-0010/1.0	
FUN-210	SR-FUN-0010/1.0	
	SR-FUN-0590/1.0	
	SR-FUN-0600/1.0	
	SR-FUN-0610/1.0	
	SR-FUN-0620/1.0	

RB identifier	SR identifier	Comment
	SR-FUN-0630/1.0	
	SR-FUN-0640/1.0	
	SR-FUN-0650/1.0	
	SR-FUN-0660/1.0	
	SR-FUN-0670/1.0	
	SR-FUN-0680/1.0	
	SR-FUN-0690/1.0	
	SR-FUN-0700/1.0	
	SR-FUN-0710/1.0	
	SR-FUN-0720/1.0	
	SR-FUN-0730/1.0	
	SR-FUN-0740/1.0	
	SR-FUN-0750/1.0	
	SR-FUN-0760/1.0	
	SR-FUN-0770/1.0	
	SR-FUN-0840/1.0	
	SR-FUN-0850/1.0	
	SR-FUN-0860/1.0	
	SR-FUN-0870/1.0	
	SR-FUN-0880/1.0	
	SR-FUN-0890/1.0	
	SR-FUN-0900/1.0	
	SR-FUN-0910/1.0	
	SR-FUN-0920/1.0	
	SR-FUN-0930/1.0	
	SR-FUN-0940/1.0	
	SR-FUN-0950/1.0	
	SR-FUN-0960/1.0	
	SR-FUN-0970/1.0	
	SR-FUN-0980/1.0	
	SR-FUN-0990/1.0	
	SR-FUN-1000/1.0	
FUN-220	--	

RB identifier	SR identifier	Comment
FUN-230	--	
FUN-240	--	
FUN-250	SR-FUN-0040/1.0	
FUN-260	--	
FUN-270	--	
FUN-280	--	
FUN-290	SR-FUN-0260/1.0	
FUN-290	SR-FUN-0270/1.0	
FUN-290	SR-FUN-0280/1.0	
FUN-290	SR-FUN-0290/1.0	
FUN-290	SR-FUN-0300/1.0	
FUN-300	SR-FUN-0050/1.0	
INT-010	SR-INT-0010/1.0	
INT-020	--	
INT-030	SR-FUN-0130/1.0	
INT-030	SR-INT-0020/1.0	
INT-040	SR-FUN-0090/1.0	
INT-040	SR-INT-0020/1.0	
INT-050	SR-FUN-0130/1.0	
INT-060	--	
CON-010	SR-FUN-0680/1.0	
CON-020	SR-FUN-0260/1.0	
CON-020	SR-FUN-0270/1.0	
CON-020	SR-FUN-0280/1.0	
CON-020	SR-FUN-0290/1.0	
CON-020	SR-FUN-0300/1.0	
CON-030	SR-FUN-1120/2.0	
	SR-FUN-1130/2.0	
	SR-FUN-1140/2.0	
CON-040	SR-FUN-1120/2.0	
	SR-FUN-1130/2.0	
	SR-FUN-1140/2.0	
CON-050	SR-FUN-0790/1.0	

RB identifier	SR identifier	Comment
CON-050	SR-FUN-0800/1.0	
CON-050	SR-FUN-0810/1.0	
CON-060	SR-FUN-0780/1.0	
CON-060	SR-FUN-0820/1.0	
CON-060	SR-FUN-0810/1.0	
CON-070	SR-IMP-0100/1.0	
CON-080	SR-IMP-0070/1.0	
CON-090	SR-IMP-0040/1.0	
CON-090	SR-INS-0010/1.0	
CON-100	SR-FUN-0090/1.0	
CON-100	SR-RES-0040/1.0	
CON-110	SR-RES-0050/2.0	
CON-120	SR-SAF-0080/1.0	
CON-130	SR-IMP-0090/1.0	
CON-140	SR-FUN-0340/1.0	
CON-140	SR-FUN-0350/1.0	
CON-140	SR-FUN-0360/1.0	
CON-140	SR-FUN-0370/1.0	
CON-140	SR-FUN-0380/1.0	
CON-140	SR-FUN-0390/1.0	
CON-140	SR-FUN-0400/1.0	
CON-140	SR-FUN-0410/1.0	
CON-140	SR-FUN-0420/1.0	
CON-150	SR-IMP-0080/1.0	
CON-160	SR-IMP-0040/1.0	
[MR01] Multiple-repository capabilities	SR-FUN-1100/2.0	
	SR-FUN-1110/2.0	
[MR01] Parameter Editor	SR-FUN-1120/2.0	
	SR-FUN-1130/2.0	
	SR-FUN-1140/2.0	
	SR-FUN-1150/2.0	
	SR-FUN-1160/2.0	

RB identifier	SR identifier	Comment
	SR-FUN-1170/2.0	
[SOW] §3.2 Web access	SR-FUN-1030/1.0	
	SR-FUN-1040/1.0	
	SR-FUN-1050/1.0	
	SR-FUN-1060/1.0	
	SR-FUN-1070/1.0	
	SR-FUN-1080/1.0	
	SR-FUN-1090/1.0	
[MR02].1	SR-FUN-1200/2.2	
	SR-FUN-1210/2.2	
	SR-FUN-1220/2.2	
	SR-FUN-1230/2.2	
[MR02].2	SR-FUN-1180/2.2	
	SR-FUN-1190/2.2	
[MR02].3	SR-FUN-1240/2.2	
[MR02].4	SR-FUN-1250/2.2	
	SR-FUN-1260/2.2	
[MR02].5	SR-FUN-1270/2.2	
[MR02].6	SR-FUN-1290/2.2	
[MR02].7	SR-FUN-1300/2.2	
[MR02].8	SR-FUN-1310/2.2	
[MR02].9	SR-FUN-1360/2.2	
	SR-FUN-1370/2.2	
	SR-FUN-1380/2.2	
	SR-FUN-1390/2.2	
	SR-FUN-1400/2.2	
	SR-FUN-1410/2.2	
	SR-FUN-1420/2.2	
	SR-FUN-1430/2.2	
	SR-FUN-1440/2.2	
[MR02].10	SR-FUN-1320/2.2	
[MR02].11	SR-FUN-1280/2.2	
[MR02].12	SR-FUN-1330/2.2	

RB identifier	SR identifier	Comment
[MR02].13	SR-OPE-0060/2.2	
	SR-OPE-0070/2.2	
[MR02].14	SR-FUN-1340/2.2	
[MR02].15	SR-FUN-1350/2.2	
[CR1].1	SR-FUN-1450/3.0 SR-FUN-1460/3.0 SR-FUN-1470/3.0	
[CR1].2	SR-FUN-1345/3.0	
[CR1].3	SR-FUN-1480/3.0	
[CR1].4	SR-FUN-1490/3.0	
[CR1].5	SR-FUN-1500/3.0 SR-FUN-1510/3.0	
[CR1].6	SR-FUN-1680/3.0 SR-FUN-1690/3.0 SR-FUN-1700/3.0 SR-FUN-1710/3.0 SR-FUN-1720/3.0 SR-FUN-1730/3.0	
[CR1].7	SR-FUN-1520/3.0	
[CR1].8	SR-FUN-1620/3.0 SR-FUN-1630/3.0	
[CR1].9	SR-FUN-1530/3.0	
[CR1].10	SR-FUN-1540/3.0 SR-FUN-1550/3.0 SR-FUN-1560/3.0 SR-FUN-1570/3.0 SR-FUN-1580/3.0 SR-FUN-1590/3.0 SR-FUN-1600/3.0 SR-FUN-1610/3.0	
[CR1].11	SR-FUN-1640/3.0 SR-FUN-1650/3.0 SR-FUN-1660/3.0 SR-FUN-1670/3.0	

5.2. Inverse Traceability

This matrix maps the traceability from the System Requirements to the User Requirements ([AD SOW]). In equivalence with section 5.1, the Comment column is reserved for further explanations.

SR identifier	RB identifier	Comment
Not implemented	FUN-020	
Not implemented	FUN-220	
Not implemented	FUN-230	
Not implemented	FUN-240	
Not implemented	FUN-260	
Not implemented	FUN-270	
Not implemented	FUN-280	
Not implemented	INT-020	
Not implemented	INT-060	
SR-RES-0050/2.0	CON-110	
SR-FUN-0010/1.0	FUN-010	
	FUN-030	
	FUN-040	
	FUN-110	
	FUN-210	
SR-FUN-0020/1.0	FUN-070	
SR-FUN-0030/1.0	FUN-040	
	FUN-130	
	FUN-150	
	FUN-160	
SR-FUN-0040/1.0	FUN-250	
SR-FUN-0050/1.0	FUN-300	
SR-FUN-0060/1.0	FUN-170	
SR-FUN-0070/1.0	FUN-170	
SR-FUN-0080/1.0	FUN-170	
SR-FUN-0090/1.0	FUN-170	
	INT-040	
	CON-100	
SR-FUN-0100/1.0		DELETED

SR identifier	RB identifier	Comment
SR-FUN-0110/1.0		DELETED
SR-FUN-0120/1.0		DELETED
SR-FUN-0130/1.0	FUN-170	
	INT-030	
	INT-050	
SR-FUN-0140/1.0	FUN-170	
SR-FUN-0150/1.0	FUN-170	
SR-FUN-0160/1.0	FUN-170	
SR-FUN-0170/1.0	FUN-170	
SR-FUN-0180/1.0	FUN-170	
SR-FUN-0190/1.0	FUN-170	
SR-FUN-0200/1.0	FUN-170	
SR-FUN-0210/1.0	FUN-170	
SR-FUN-0220/1.0	FUN-170	
SR-FUN-0230/1.0	FUN-170	
SR-FUN-0240/1.0	FUN-170	
SR-FUN-0250/1.0	FUN-170	
SR-FUN-0260/1.0	FUN-100	
	FUN-170	
	FUN-190	
	FUN-290	
	CON-020	
SR-FUN-0270/1.0	FUN-100	
	FUN-170	
	FUN-190	
	FUN-290	
	CON-020	
SR-FUN-0280/1.0	FUN-100	
	FUN-170	
	FUN-190	
	FUN-290	
	CON-020	
SR-FUN-0290/1.0	FUN-100	

SR identifier	RB identifier	Comment
	FUN-170	
	FUN-190	
	FUN-290	
	CON-020	
SR-FUN-0300/1.0	FUN-100	
	FUN-170	
	FUN-190	
	FUN-290	
	CON-020	
SR-FUN-0310/1.0	FUN-170	
SR-FUN-0320/1.0	FUN-170	
SR-FUN-0330/1.0	FUN-170	
SR-FUN-0340/1.0	FUN-170	
	CON-140	
SR-FUN-0350/1.0	FUN-170	
	CON-140	
SR-FUN-0360/1.0	FUN-170	
	CON-140	
SR-FUN-0370/1.0	FUN-170	
	CON-140	
SR-FUN-0380/1.0	FUN-170	
	CON-140	
SR-FUN-0390/1.0	FUN-170	
	CON-140	
SR-FUN-0400/1.0	FUN-170	
	CON-140	
SR-FUN-0410/1.0	FUN-170	
	CON-140	
SR-FUN-0420/1.0	FUN-170	
	CON-140	
SR-FUN-0430/1.0	FUN-110	
SR-FUN-0440/1.0	FUN-110	
SR-FUN-0450/1.0	FUN-110	

SR identifier	RB identifier	Comment
SR-FUN-0460/1.0	FUN-110	
SR-FUN-0470/1.0	FUN-110	
SR-FUN-0480/1.0	FUN-110	
SR-FUN-0490/1.0	FUN-110	
SR-FUN-0500/1.0	FUN-110	
SR-FUN-0510/1.0	FUN-110	
SR-FUN-0520/1.0	FUN-110	
SR-FUN-0530/1.0	FUN-110	
SR-FUN-0540/1.0	FUN-110	
SR-FUN-0550/1.0	FUN-110	
SR-FUN-0560/1.0	FUN-110	
SR-FUN-0570/1.0	FUN-110	
SR-FUN-0580/1.0	FUN-110	
SR-FUN-0590/1.0	FUN-110	
	FUN-210	
SR-FUN-0600/1.0	FUN-110	
	FUN-210	
SR-FUN-0610/1.0	FUN-030	
	FUN-090	
	FUN-110	
	FUN-210	
SR-FUN-0620/1.0	FUN-110	
	FUN-180	
	FUN-210	
SR-FUN-0630/1.0	FUN-110	
	FUN-210	
SR-FUN-0640/1.0	FUN-110	
	FUN-210	
SR-FUN-0650/1.0	FUN-110	
	FUN-210	
SR-FUN-0660/1.0	FUN-110	
	FUN-210	
SR-FUN-0670/1.0	FUN-110	

SR identifier	RB identifier	Comment
	FUN-210	
SR-FUN-0680/1.0	FUN-030	
	FUN-110	
	FUN-210	
	CON-010	
SR-FUN-0690/1.0	FUN-110	
	FUN-210	
SR-FUN-0700/1.0	FUN-110	
	FUN-210	
SR-FUN-0710/1.0	FUN-110	
	FUN-170	
	FUN-210	
SR-FUN-0720/1.0	FUN-110	
	FUN-210	
SR-FUN-0730/1.0	FUN-110	
	FUN-210	
SR-FUN-0740/1.0	FUN-110	
	FUN-210	
SR-FUN-0750/1.0	FUN-110	
	FUN-210	
SR-FUN-0760/1.0	FUN-130	
	FUN-210	
SR-FUN-0770/1.0	FUN-070	
	FUN-210	
SR-FUN-0780/1.0	FUN-170	
	CON-060	
SR-FUN-0790/1.0	FUN-170	
	CON-050	
SR-FUN-0800/1.0	FUN-170	
	CON-050	
SR-FUN-0810/1.0	FUN-170	
	CON-050	
	CON-060	

SR identifier	RB identifier	Comment
SR-FUN-0820/1.0	FUN-050	
	CON-060	
SR-FUN-0830/1.0	FUN-050	
SR-FUN-0840/1.0	FUN-210	
SR-FUN-0850/1.0	FUN-210	
SR-FUN-0860/1.0	FUN-210	
SR-FUN-0870/1.0	FUN-210	
SR-FUN-0880/1.0	FUN-210	
SR-FUN-0890/1.0	FUN-210	
SR-FUN-0900/1.0	FUN-210	
SR-FUN-0910/1.0	FUN-210	
SR-FUN-0920/1.0	FUN-070	
	FUN-210	
SR-FUN-0930/1.0	FUN-070	
	FUN-210	
SR-FUN-0940/1.0	FUN-160	
	FUN-210	
SR-FUN-0950/1.0	FUN-160	
	FUN-210	
SR-FUN-0960/1.0	FUN-160	
	FUN-210	
SR-FUN-0970/1.0	FUN-160	
	FUN-210	
SR-FUN-0980/1.0	FUN-160	
	FUN-210	
SR-FUN-0990/1.0	FUN-040	
	FUN-130	
	FUN-150	
	FUN-210	
SR-FUN-1000/1.0	FUN-130	
	FUN-140	
	FUN-210	
SR-FUN-1010/1.0	FUN-120	

SR identifier	RB identifier	Comment
SR-FUN-1020/1.0	FUN-120	
SR-FUN-1030/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1040/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1050/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1060/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1070/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1080/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1090/1.0	[SOW] Section 3.2	Web site access
SR-FUN-1100/2.0	[MR01]	Multiple-repository capabilities
SR-FUN-1110/2.0	[MR01]	Multiple-repository capabilities
SR-FUN-1120/2.0	[MR01]	Parameter Editor
	CON-030	
	CON-040	
SR-FUN-1130/2.0	[MR01]	Parameter Editor
	CON-030	
	CON-040	
SR-FUN-1140/2.0	[MR01]	Parameter Editor
	CON-030	
	CON-040	
SR-FUN-1150/2.0	[MR01]	Parameter Editor
SR-FUN-1160/2.0	[MR01]	Parameter Editor
SR-FUN-1170/2.0	[MR01]	Parameter Editor
SR-FUN-1180/2.2	[MR02].2	
SR-FUN-1190/2.2	[MR02].2	
SR-FUN-1200/2.2	[MR02].1	
SR-FUN-1210/2.2	[MR02].1	
SR-FUN-1220/2.2	[MR02].1	
SR-FUN-1230/2.2	[MR02].1	
SR-FUN-1240/2.2	[MR02].3	
SR-FUN-1250/2.2	[MR02].4	
SR-FUN-1260/2.2	[MR02].4	
SR-FUN-1270/2.2	[MR02].5	
SR-FUN-1280/2.2	[MR02].11	

SR identifier	RB identifier	Comment
SR-FUN-1290/2.2	[MR02].6	
SR-FUN-1300/2.2	[MR02].7	
SR-FUN-1310/2.2	[MR02].8	
SR-FUN-1320/2.2	[MR02].10	
SR-FUN-1330/2.2	[MR02].12	
SR-FUN-1340/2.2	[MR02].14	
SR-FUN-1345/3.0	[CR1].2	
SR-FUN-1350/2.2	[MR02].15	
SR-FUN-1360/2.2	[MR02].9	
SR-FUN-1370/2.2	[MR02].9	
SR-FUN-1380/2.2	[MR02].9	
SR-FUN-1390/2.2	[MR02].9	
SR-FUN-1400/2.2	[MR02].9	
SR-FUN-1410/2.2	[MR02].9	
SR-FUN-1420/2.2	[MR02].9	
SR-FUN-1430/2.2	[MR02].9	
SR-FUN-1440/2.2	[MR02].9	
SR-FUN-1450/3.0	[CR1].1	
SR-FUN-1460/3.0	[CR1].1	
SR-FUN-1470/3.0	[CR1].1	
SR-FUN-1480/3.0	[CR1].3	
SR-FUN-1490/3.0	[CR1].4	
SR-FUN-1500/3.0	[CR1].5	
SR-FUN-1510/3.0	[CR1].5	
SR-FUN-1520/3.0	[CR1].7	
SR-FUN-1530/3.0	[CR1].9	
SR-FUN-1540/3.0	[CR1].10	
SR-FUN-1550/3.0	[CR1].10	
SR-FUN-1560/3.0	[CR1].10	
SR-FUN-1570/3.0	[CR1].10	
SR-FUN-1580/3.0	[CR1].10	
SR-FUN-1590/3.0	[CR1].10	
SR-FUN-1600/3.0	[CR1].10	

SR identifier	RB identifier	Comment
SR-FUN-1610/3.0	[CR1].10	
SR-FUN-1620/3.0	[CR1].8	
SR-FUN-1630/3.0	[CR1].8	
SR-FUN-1640/3.0	[CR1].11	
SR-FUN-1650/3.0	[CR1].11	
SR-FUN-1660/3.0	[CR1].11	
SR-FUN-1670/3.0	[CR1].11	
SR-FUN-1680/3.0	[CR1].6	
SR-FUN-1690/3.0	[CR1].6	
SR-FUN-1700/3.0	[CR1].6	
SR-FUN-1710/3.0	[CR1].6	
SR-FUN-1720/3.0	[CR1].6	
SR-FUN-1730/3.0	[CR1].6	
SR-IMP-0040/1.0	CON-090	
	CON-160	
SR-IMP-0070/1.0	CON-080	
SR-IMP-0080/1.0	CON-150	
SR-IMP-0090/1.0	CON-130	
SR-IMP-0100/1.0	CON-070	
SR-INS-0010/1.0	FUN-200	
	CON-090	
SR-INT-0010/1.0	INT-010	
SR-INT-0020/1.0	FUN-080	
	INT-030	
	INT-040	
SR-INT-0030/1.0	FUN-080	
SR-OPE-0020/1.0	FUN-060	
SR-OPE-0030/1.0	FUN-060	
SR-OPE-0040/1.0	FUN-060	
SR-OPE-0050/1.0	FUN-050	
SR-OPE-0060/1.0	[MR02].13	
SR-OPE-0070/1.0	[MR02].13	
SR-RES-0040/1.0	CON-100	

SR identifier	RB identifier	Comment
SR-RES-0050/2.0	CON-110	
SR-SAF-0080/1.0	CON-120	

End of document