openSF Evolution

Open Simulation Framework Evolution

1st ESA Workshop

Federico Letterio
Javier Martín Avila
Gonzalo Vicario
DEIMOS Space

openSF EVO WS1
ESA/ESTEC
Noordwijk (NL)
23/01/2018

Agenda

9h00 General Presentation of E2E (ESA)

- General architecture (ARCHEO), activities ongoing (BIBLOS, ETC)
- Baseline documentation and process within Earth Observation for E2E
- Rationale for a standard framework and interface

9h30 Presentation/demonstrations openSF Framework (DMS) - General

- Use cases and type of users
- General description of openSF + PE + OSFI + OSFEG (terminology and concepts) and description of new functionalities since version 3.5
- Guidelines for defining configuration files and use openSF (e.g. use of directories or file references and other mistakes commonly observed)
- **Demo** of the openSF installation (linux and OSX)
- **Demo** (linux) of step-by step full nominal process to build/integrate a simulator (create descriptor, models, stages, simulation etc) using openSF
- **Demo** (linux) of example cases delivered with openSF (validation DB)
- **Demo** (linux) of Parameter Editor and planned evolution of it

Agenda

9h30 Presentation/demonstrations openSF Framework (DMS) - Special/specific cases and advanced features

- How-to: Time-driven orchestration, use case, scenario and definition including modification of the configuration files and Module's code to support it
- Iterations and Montecarlo: what can be done and what not
  - Iteration for sensitivity analysis
  - Discussion as very special case on Montecarlo
- DB: GUI functionalities and how to clean/configure it
- How-to: migration a E2E simulator to a newer openSF version

15h00 Exercise

- Hands-on exercise done by participants (e.g. installation or build a chain and run with iteration)

16h00 Q&A

1 Introduction

E2E Mission Performance Chains

Main Software Components (Phase B2/C/D)

- Geometry Module
- Scene Generator
- Instrument Simulator(s) IDS
- Processor Prototypes (L1PP, L2PP)
- Performance Assessment Module

} a.k.a. the OSS

E2E Tasks involves Iterations and repetitions

- Laborious
- Repetitive
- Time consuming
- Parameter Iteration
- Time Iteration
- Data sorting and collection
- Configuration Management
- Coherency of Environment (e.g. orbit)

2 Use Cases & Users' Types

Different Use Cases

- Verification of the System requirements during
  - the design consolidation
  - the instrument/system on-ground characterisation/calibration
- Consolidation/(re-)allocation of performance margins
- Verification of the relevant performance models
- Consolidation of calibration operational profile
- Data level format compatibility verification (AIV/AIT) and debugging
- Ground Segment Test Data Generator (Level 0 from simulator, level 1 from GPP)
- Test and verification environment for Level1 prototype
- Verification of calibration processing
- Blueprint for the Operational L1 processor (including algorithm design and performance)
- Test and verification environment for L2PP
- Support troubleshooting analysis during commissioning

Different openSF Users

Three main user's profiles

- Module developer (code)
  - SW engineer/scientist
  - Develops SW modules and interfaces (E2E ICD compliant)
  - Uses OSFI and OSFEG libraries as support to own code

- E2E simulator integrator (interfaces)
  - SW engineer
  - Integrates the modules into a simulation chain, using all functionalities of openSF
  - Verifies overall coherency/compatibility of modules (configuration files)
  - Packages the final product

- E2E simulator final user (parameters and data)
  - Sat or GS engineer or scientist
  - Will perform many runs simulating various conditions (parameters) or generating different data set with different inputs
  - Will process outputs to get statistics/results/graphs

Different openSF Users

Hybrid profile

- There is the possibility that the same user can cover the several profiles at the same time
  - Module developer integrating and using openSF for validation purposes
  - Integrator that also implements modules
- Overlap of different use cases

Different openSF Users

Hybrid profile

- There is the possibility that the same user can cover the several profiles at the same time
  - Module developer integrating and using openSF for validation purposes
  - Integrator that also implements modules
- Overlap of different use cases

! Risk of forgetting the final user of the E2E simulator !

Orchestration Infrastructure

openSF is an orchestration infrastructure providing

- Standard component interface (as per generic E2E ICD)
- Automatic invocation with the right input of components
- Automatic parameters span (sensitivity)
- Automatic time/scenario span (as satellite flies)
- Automatic collection/organisation of results
- Easy to use GUI
- Multiplatform (Linux + OSX)
- Free maintenance by ESTEC

3 openSF Framework

Open Simulation Framework

openSF is an independent simulation framework providing added-value functionalities to scientific simulations

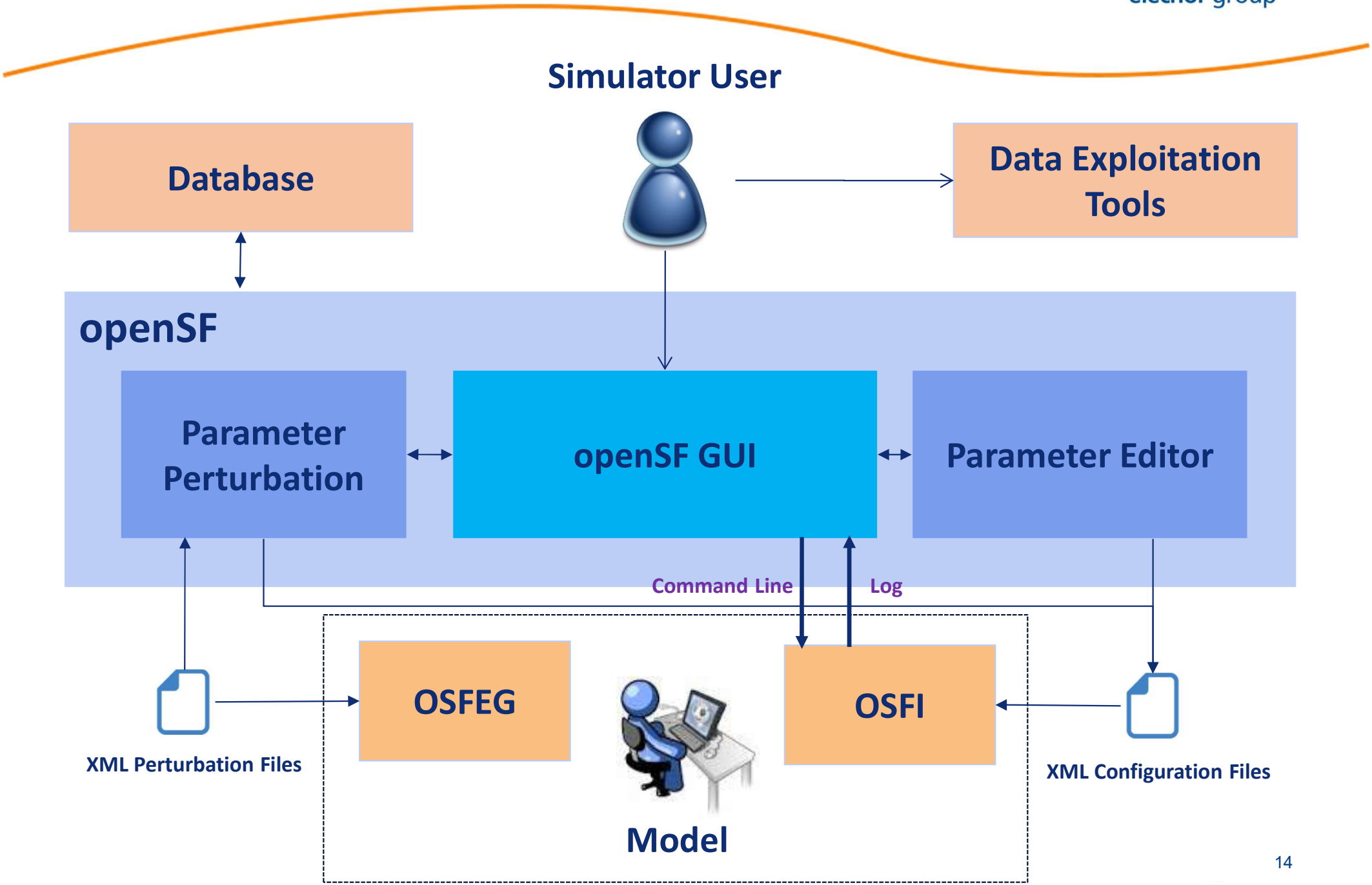
- Intuitive, lightweight and portable GUI (Java)

Scientific models and product exploitation tools can be plugged in the system with ease

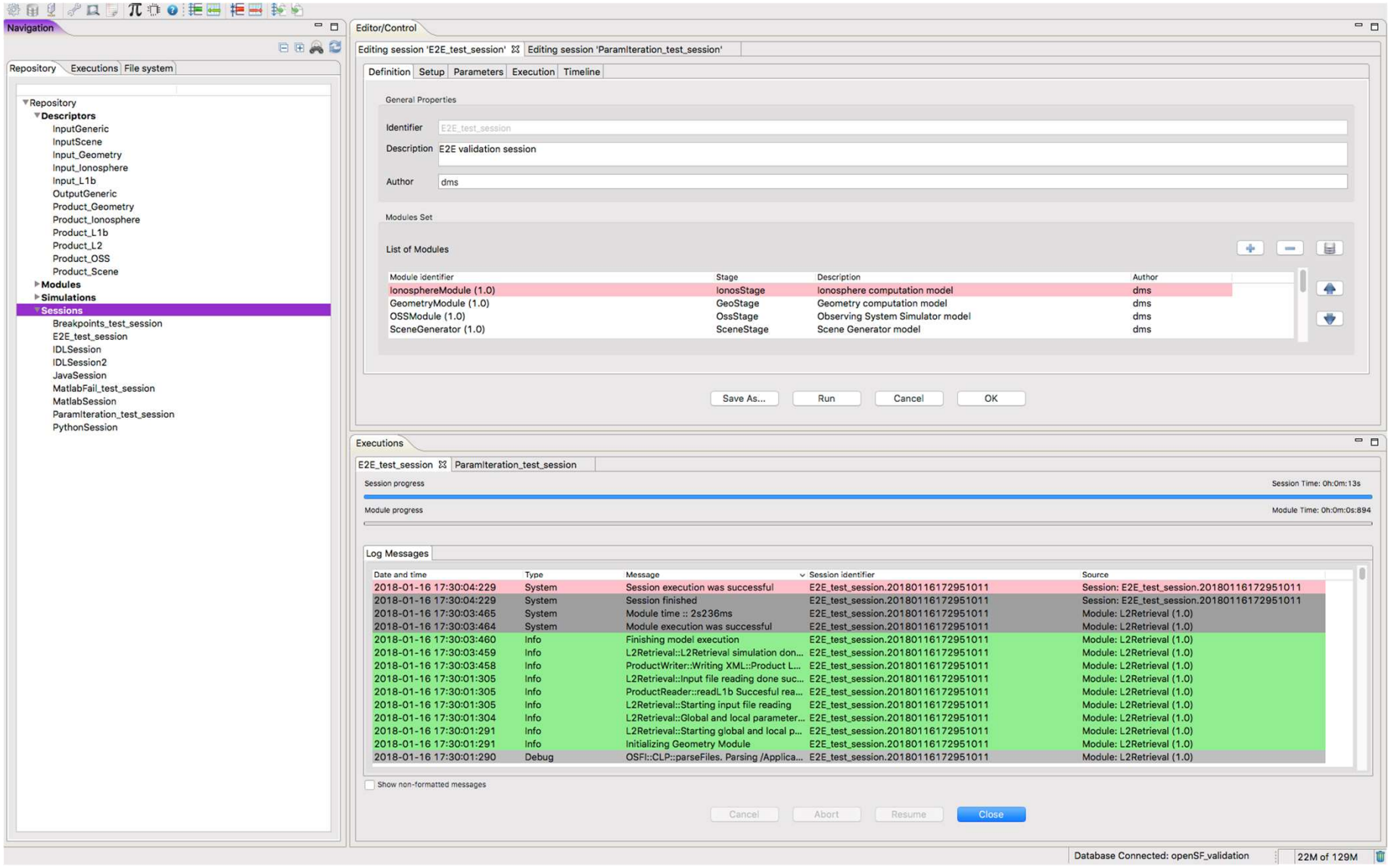
- Simple and non-intrusive interface for scientific models
- Extra applications & libraries for faster development (OSFI, OSFEG) and easier usage (Parameter Editor)

openSF functionalities are aimed to cope with E2E simulators requirements but its architecture made it also a good framework for any processing chain (image processing, …)

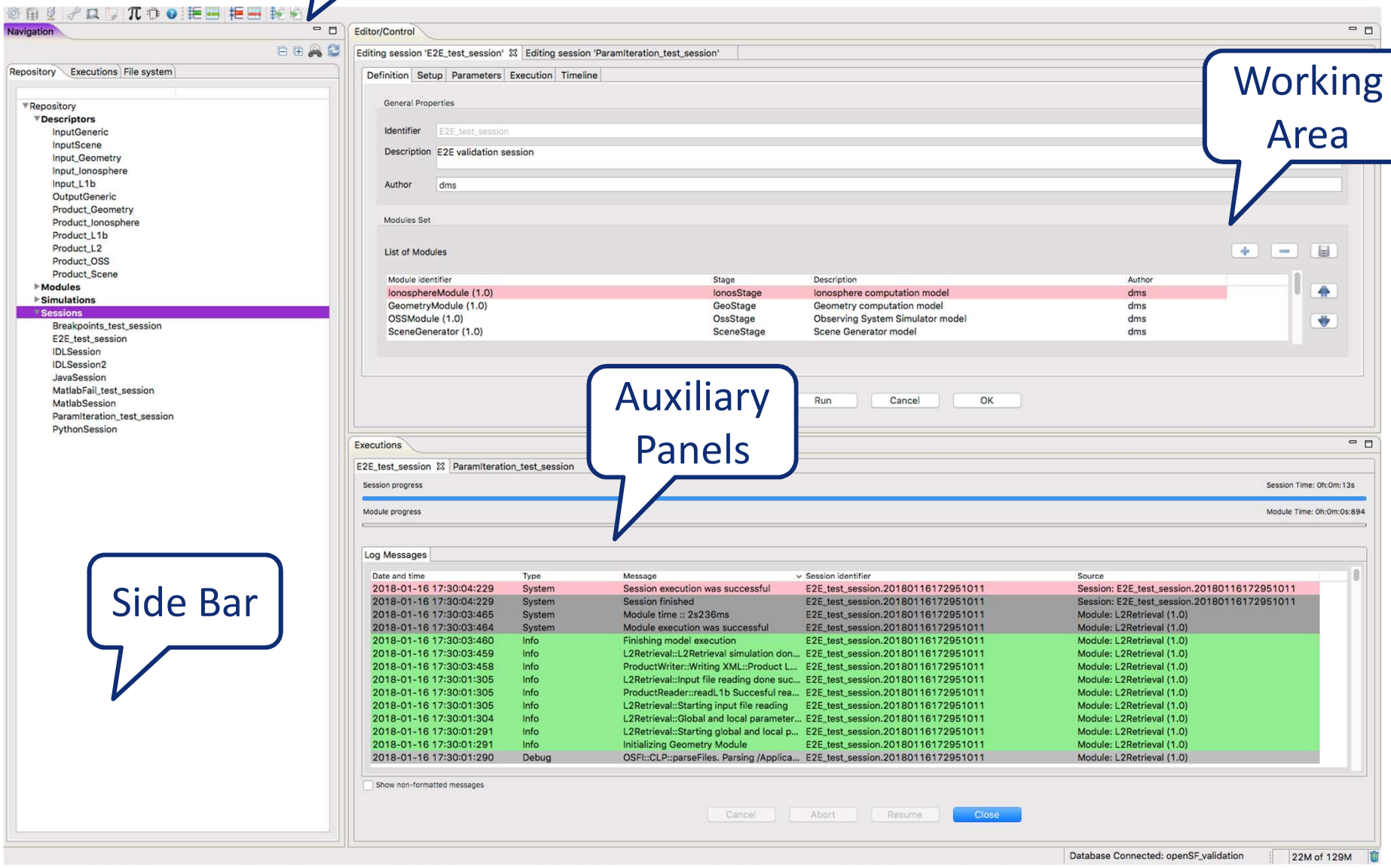
openSF System Context



openSF Graphical User Interface



openSF Graphical User Interface



OSFI - openSF Integration Library

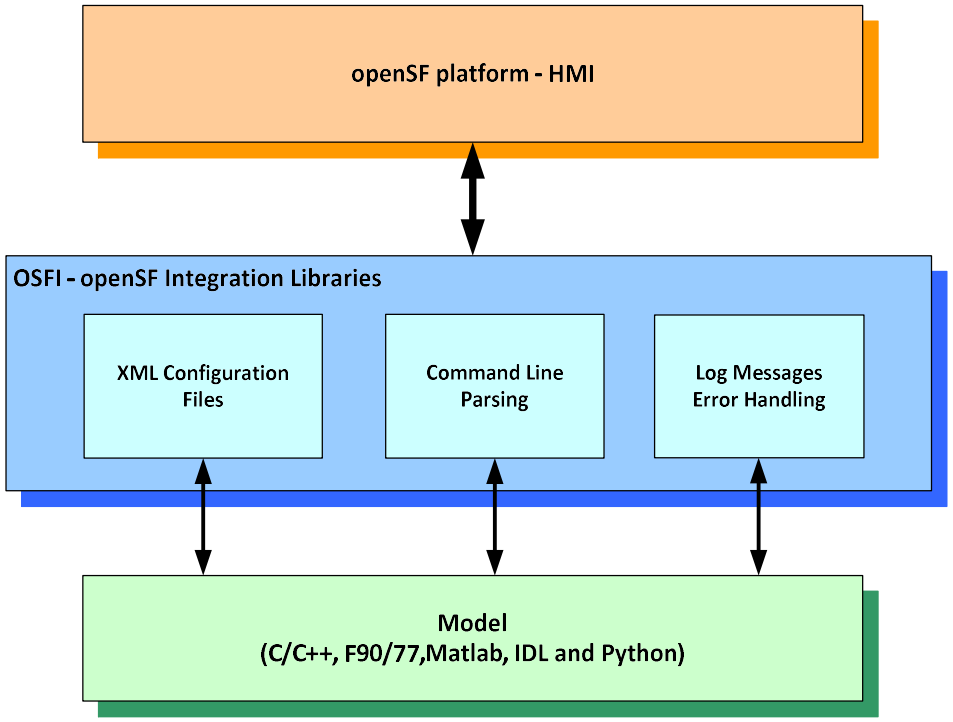
Integration libraries aimed to ease model integration into openSF

Cover the three interface constraints introduced by openSF:

- **Configuration** files parsing
- **Command** line parsing
- **Log** messages generation

Multi-language support

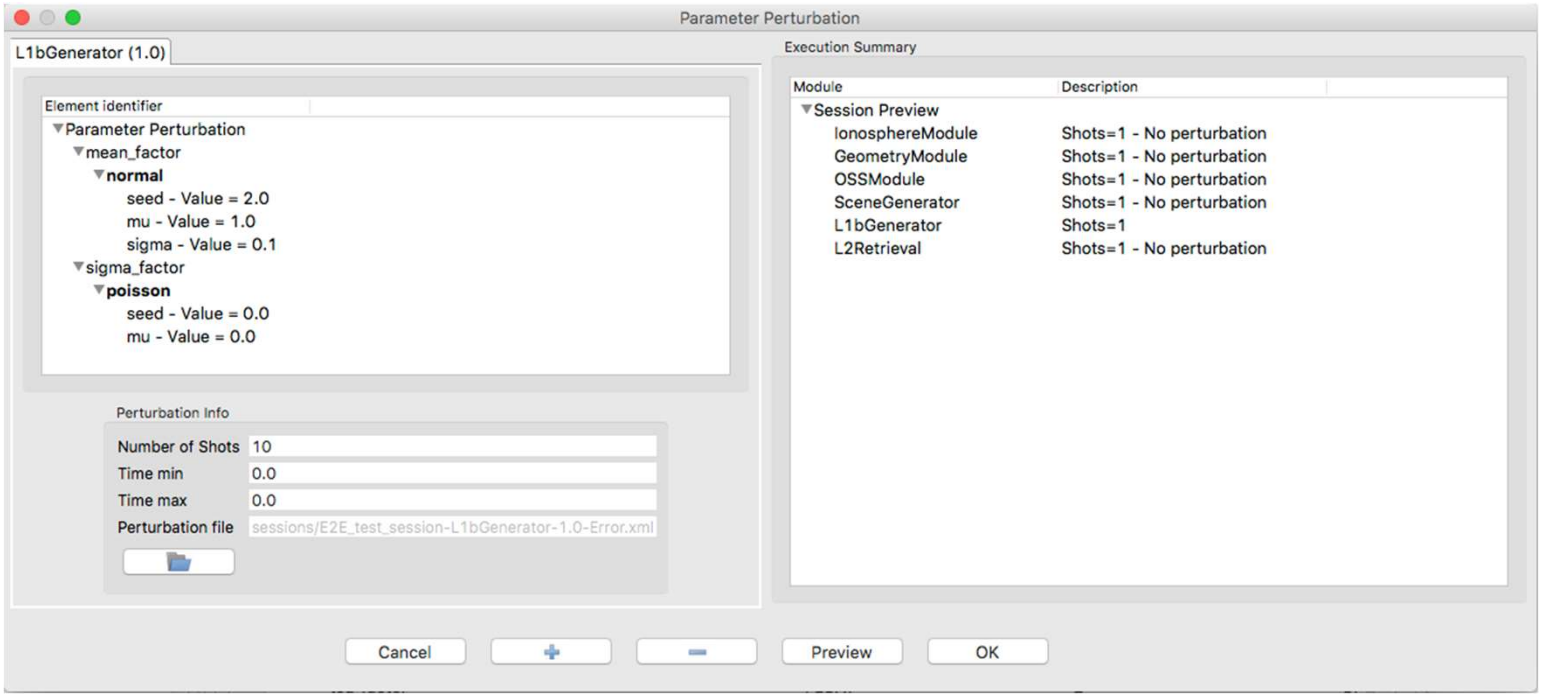
- C
- C++
- Java
- Fortran
- Python
- Matlab



OSFEG - openSF Error Generation Library

Libraries aimed at generating perturbations on input parameters, gives the module developer access to the same functions that perform the equivalent task in openSF

Improved sensitivity analysis approach (simple iteration of parameter values) by combining it with perturbation functions



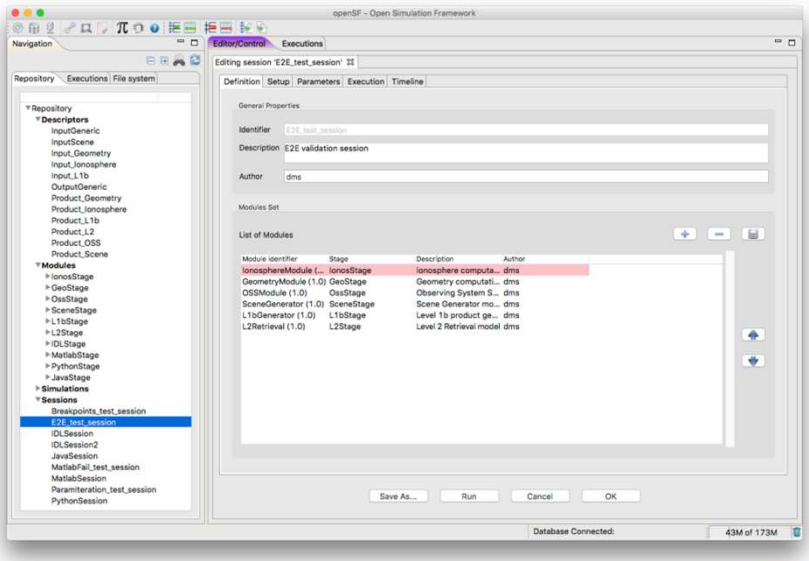
Time Driven Scenario Orchestration

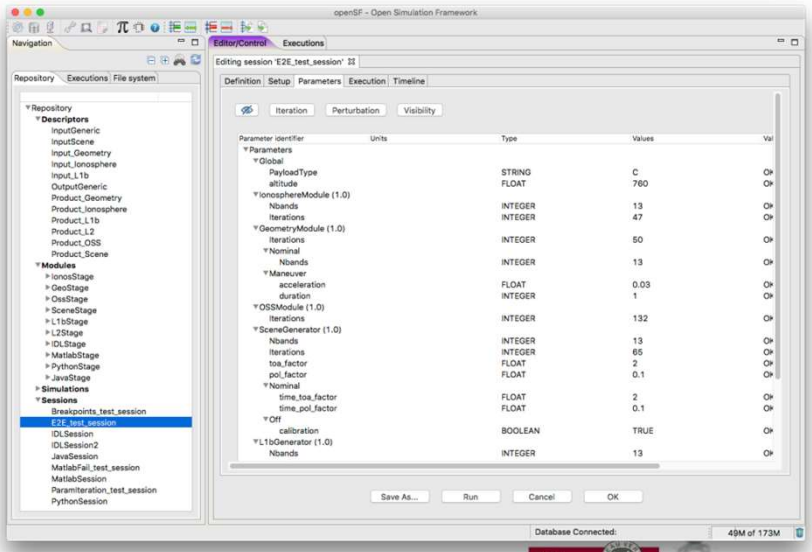
For end user

- Simpler development of Modules (takes care externally of time-changing parameters) and flexible use.
- Define concept of simulated "**time interval**" in a given "**mode**"
- Introduce the concept of **timeline scenario** as sequential concatenation of simulated time intervals.

How ?

- Definition of configurable instrument operational scenario (e.g. calibration, observation, standby, off, etc..) represented by "**modes**" corresponding to different **set of parameters** value/configurations
- Minimal impact on openSF interfaces

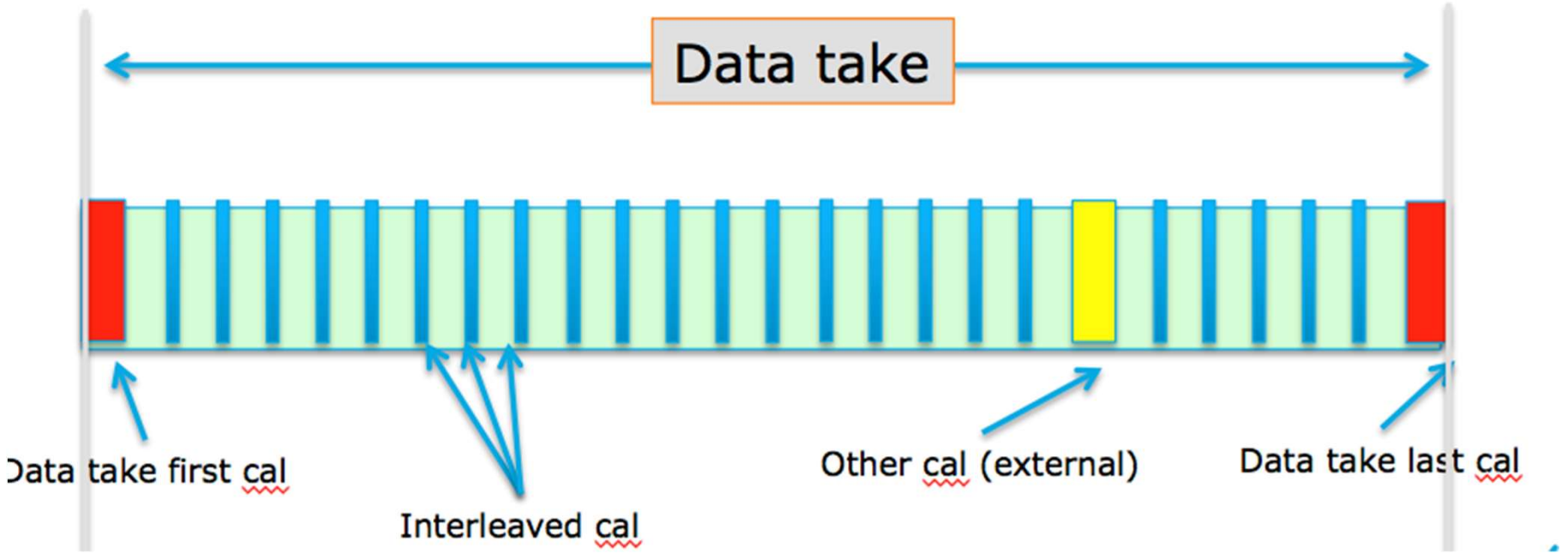




Time Driven Scenario Orchestration (1)

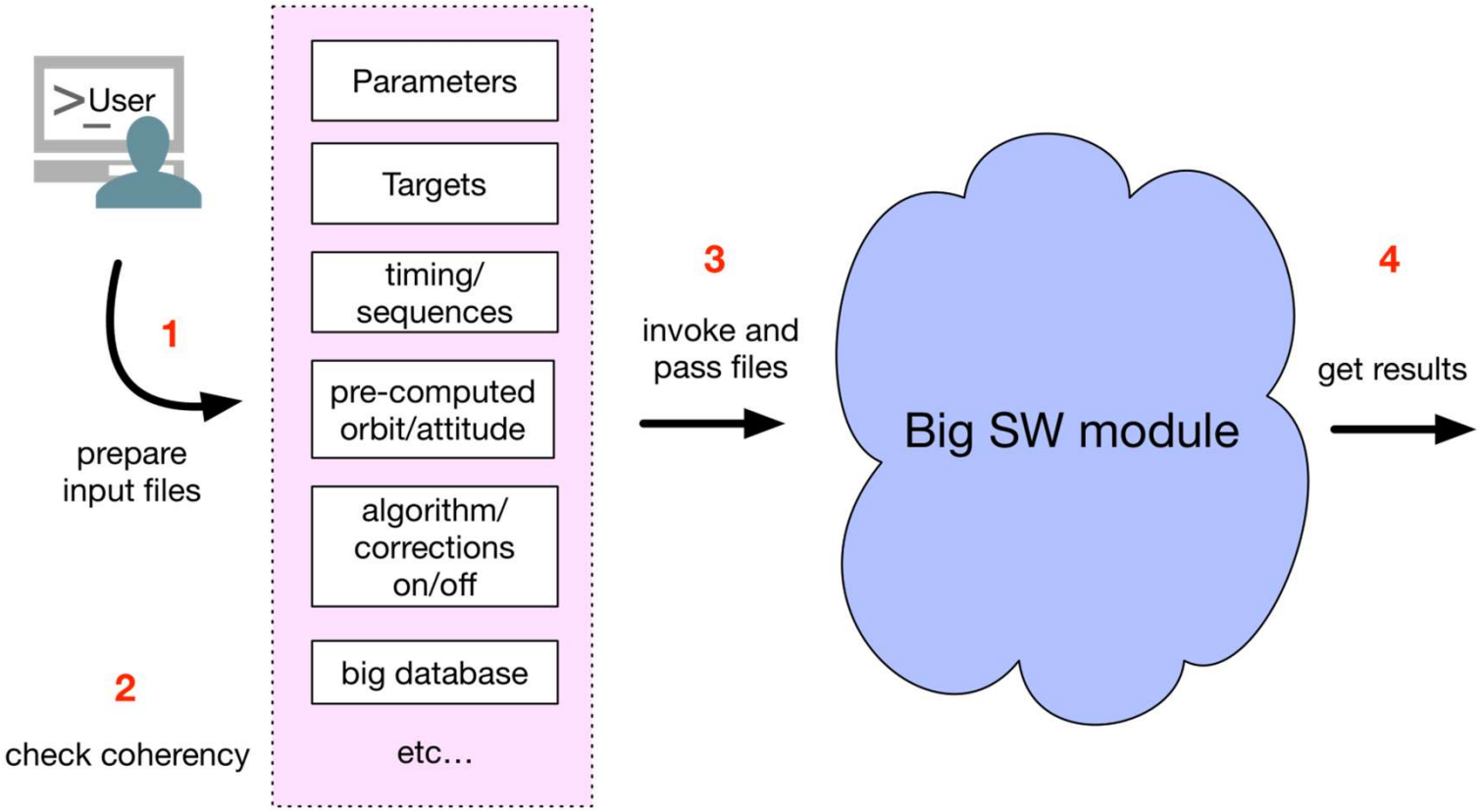
Use case:

Scenario of observation and calibration modes with instrument configuration and target changing with time.



Time Driven Scenario Orchestration (2)

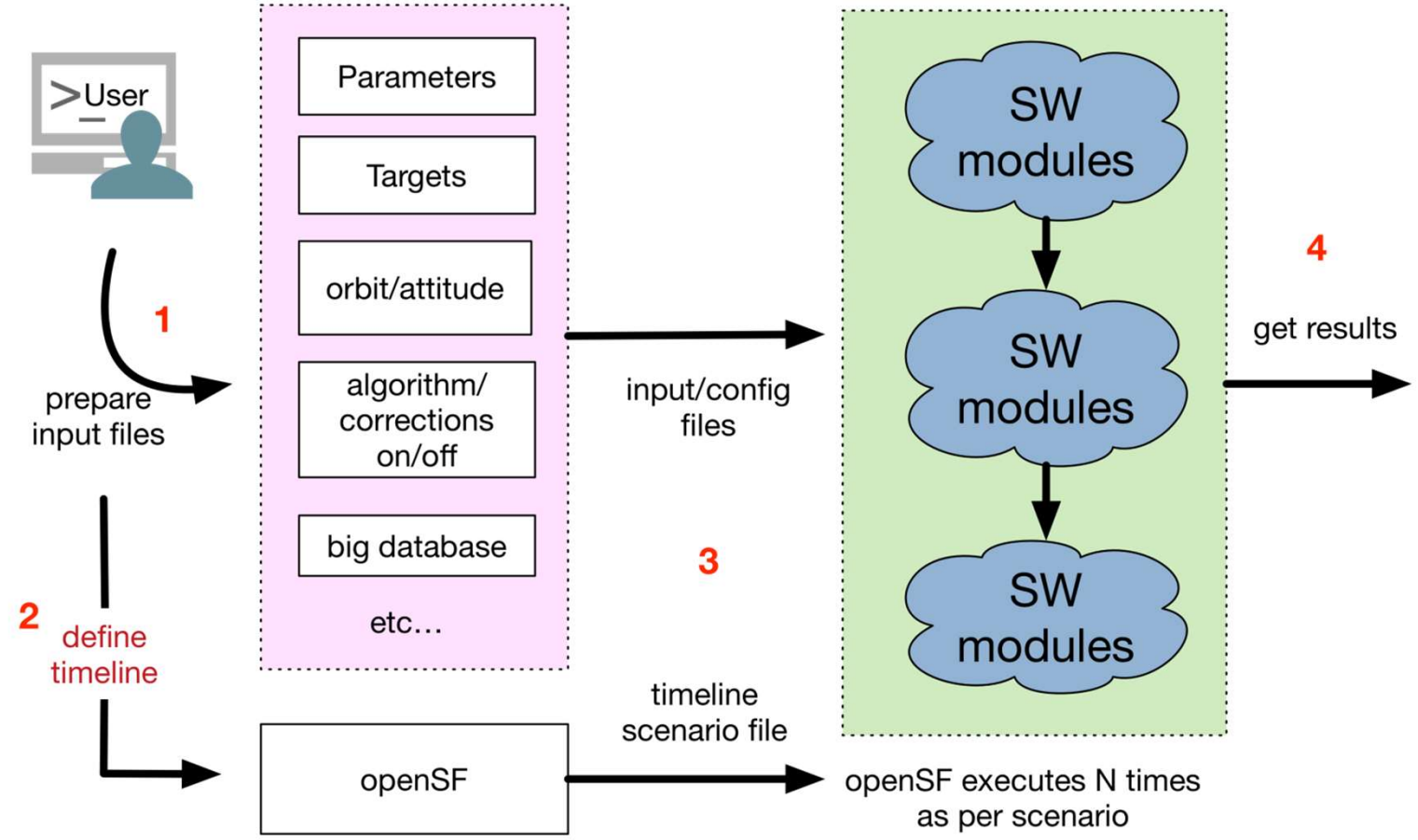
With orchestration internal to module:



Observation/cal scenario change ?? → repeat N times

Time Driven Scenario Orchestration (3)

With mode-aware simulator design



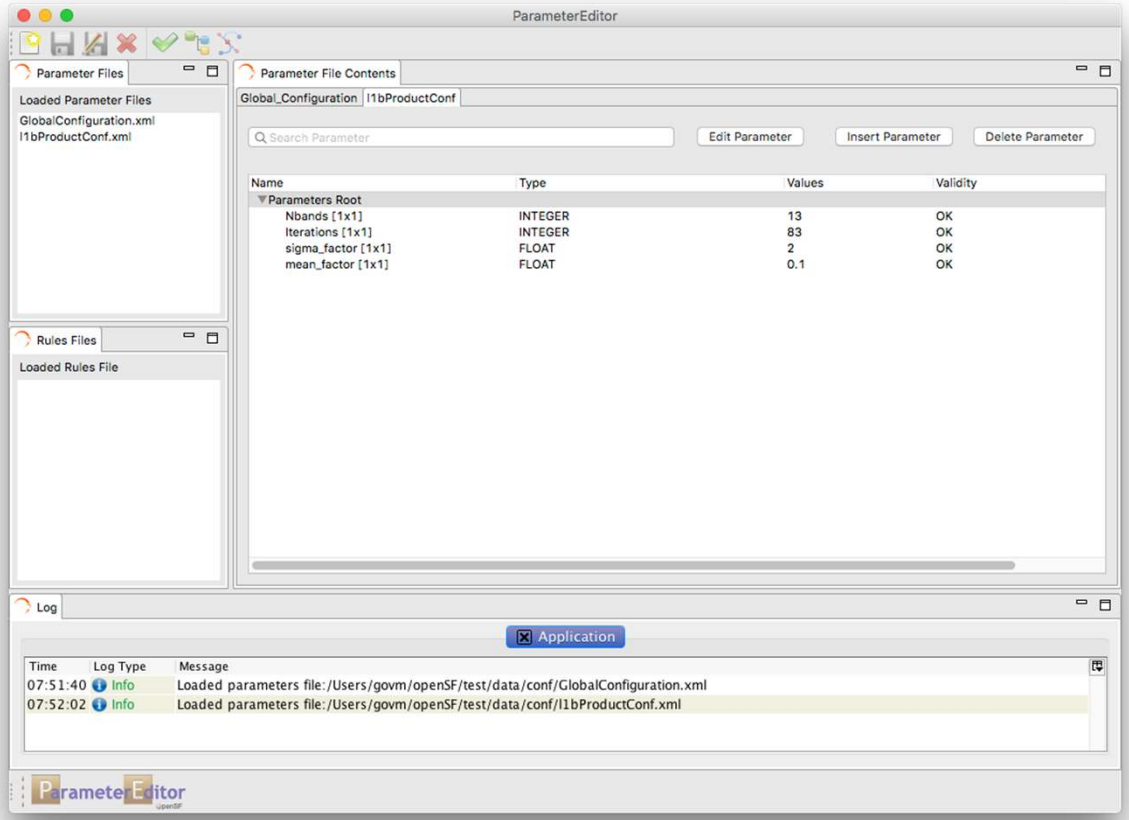
Observation/cal scenario change ?? → change only step 2

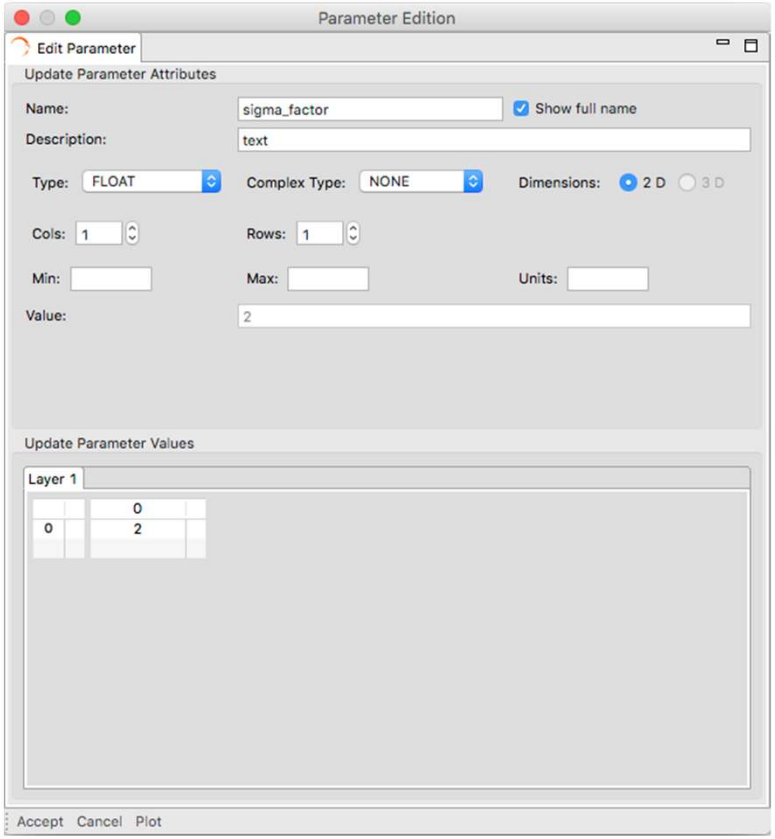
Graphical tool for editing openSF configuration files

- User-friendly interface to create, edit and delete parameters avoiding the manual XML text editing
- Enhanced consistency checking of parameters, including range, type and dimension check
- Enhanced editing with excel-like interface for vectors/matrix parameters and plot capabilities
- Built-in XML text editor with syntax and errors highlighting
- Interface for rules and constraint definition connecting parameters that can be located in different configuration files

Parameter Editor

Graphical tool for editing openSF configuration files





Parameter Editor Evolution

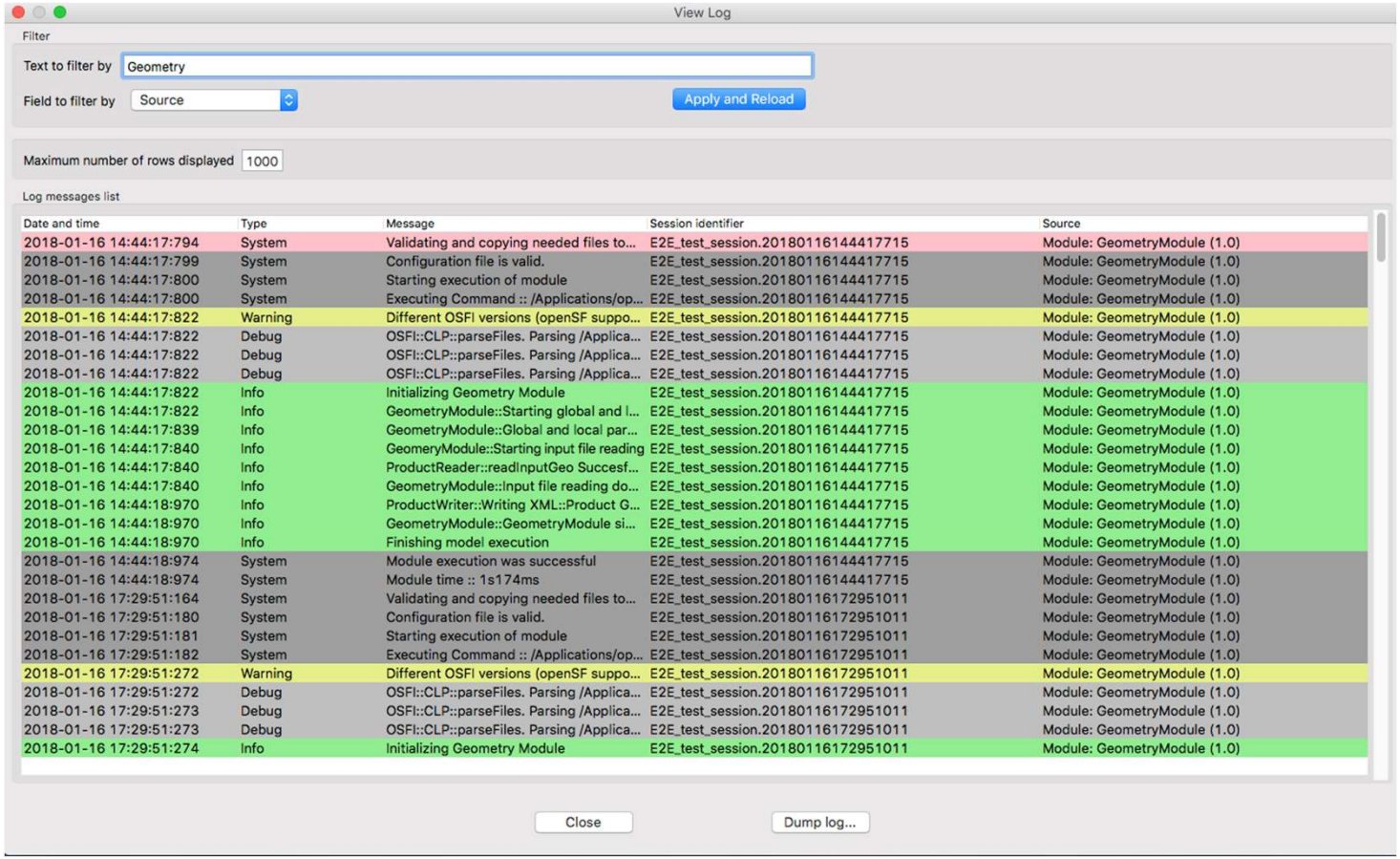
Evolution of PE to be released in 1st quarter 2018

- Eclipse RCP porting
- Improved UI
- XSD validation
- Copy/paste of parameters
- Remove/insert of columns/rows
- In-line syntax for edition

Log window

Color coded logs, intercepted from executed modules

Filtering capability



Date and time	Type	Message	Session identifier	Source
2018-01-16 14:44:17:794	System	Validating and copying needed files to...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:799	System	Configuration file is valid.	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:800	System	Starting execution of module	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:800	System	Executing Command :: /Applications/op...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:822	Warning	Different OSFI versions (openSF suppo...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:822	Debug	OSFI::CLP::parseFiles. Parsing /Applica...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:822	Debug	OSFI::CLP::parseFiles. Parsing /Applica...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:822	Debug	OSFI::CLP::parseFiles. Parsing /Applica...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:822	Info	Initializing Geometry Module	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:822	Info	GeometryModule::Starting global and l...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:839	Info	GeometryModule::Global and local par...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:840	Info	GeomeryModule::Starting input file reading	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:840	Info	ProductReader::readInputGeo Succesf...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:17:840	Info	GeometryModule::Input file reading do...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:18:970	Info	ProductWriter::Writing XML::Product G...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:18:970	Info	GeometryModule::GeometryModule si...	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:18:970	Info	Finishing model execution	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:18:974	System	Module execution was successful	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 14:44:18:974	System	Module time :: 1s174ms	E2E_test_session.20180116144417715	Module: GeometryModule (1.0)
2018-01-16 17:29:51:164	System	Validating and copying needed files to...	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:180	System	Configuration file is valid.	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:181	System	Starting execution of module	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:182	System	Executing Command :: /Applications/op...	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:272	Warning	Different OSFI versions (openSF suppo...	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:272	Debug	OSFI::CLP::parseFiles. Parsing /Applica...	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:273	Debug	OSFI::CLP::parseFiles. Parsing /Applica...	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:273	Debug	OSFI::CLP::parseFiles. Parsing /Applica...	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)
2018-01-16 17:29:51:274	Info	Initializing Geometry Module	E2E_test_session.20180116172951011	Module: GeometryModule (1.0)

openSF Recent Evolution

From openSF 3.5 to openSF 3.7.1

- Migration of GUI to Eclipse RCP
- Improvement to Log filtering and sorting
- Enhancement of Timeline management
- Support to Python 3.x
- Live session execution progress status
- Upgrade COTS (→ MySQL 5.7)
- Improved interface (i.e. menu bar and contextual menus)
- Compliance to OSX guidelines
- Remove dependencies between openSF and OSFI, upgrade to xerces 3.2.0 and new OSFI distribution strategy (only source)
- OSFI Fortran command line parsing problem fixed

openSF Recent Evolution

From openSF 3.5 to openSF 3.7.1

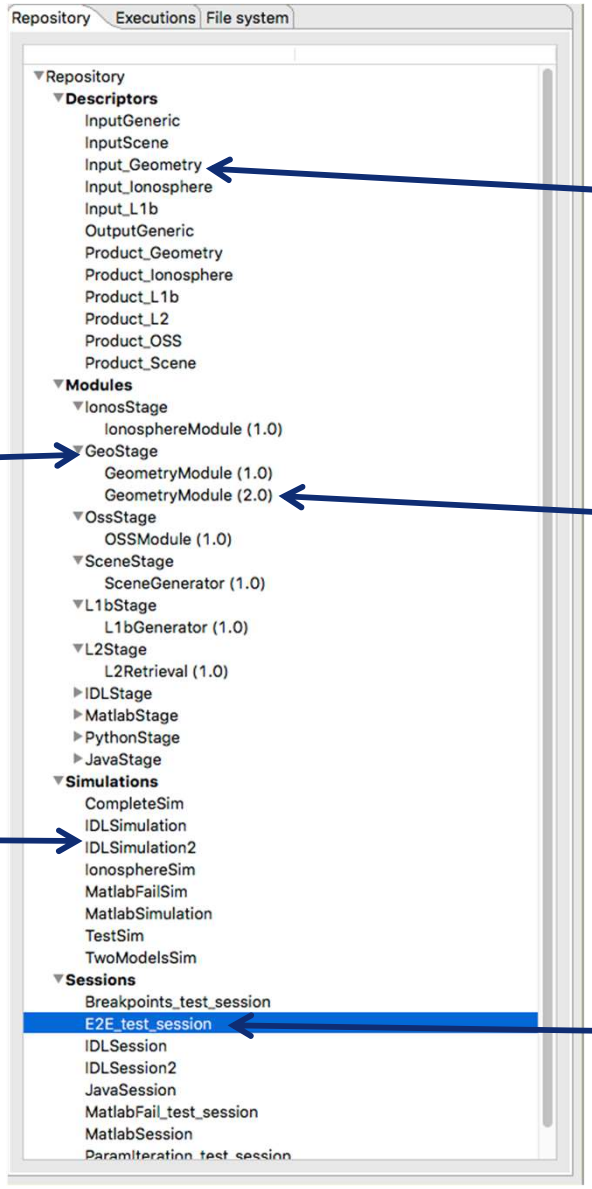
- DB export to XML
- Improved DB management at start-up
- Implemented Log by session
- openSF internal log messages moved to dedicated log files
  - Mitigates problems with “talkative” modules causing slowdowns due to the parsing of a huge openSF.log file
- User friendly installation process
  - Possibility to provide MySQL tools path during installation
  - Server/database created during installation selected by the user
- Bug and other fixes

4 Framework Concepts

Basic openSF elements

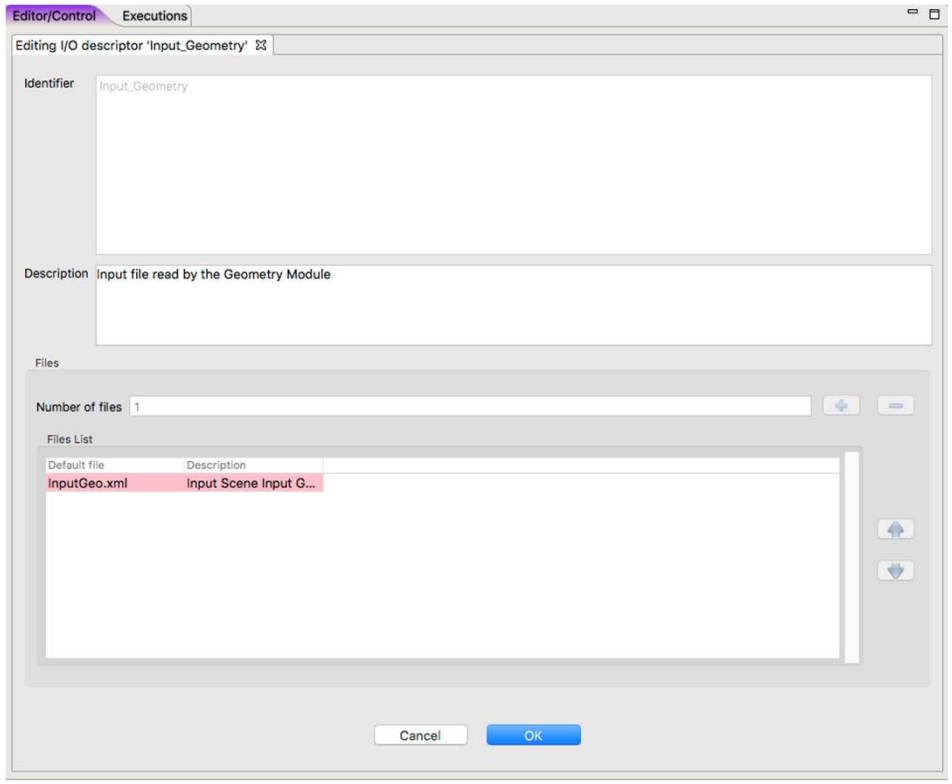
- Session
  - Element that openSF runs
  - Comprises multiple executable blocks called **modules**
  - Modules are grouped in **stages**
- Module
  - Executable entities that perform the calculations
  - Inputs and outputs are defined by **descriptors**
  - Behaviour modified by the user with **configuration files**
- Stage
  - It is a collection of modules
  - Defines a phase in a simulation process
- Simulation
  - Most abstract element
  - Template for the session definition, groups chains of modules commonly used together

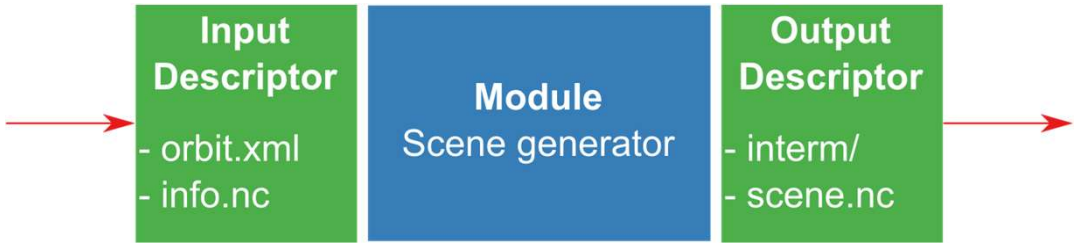
Basic openSF elements



Input/output module interface

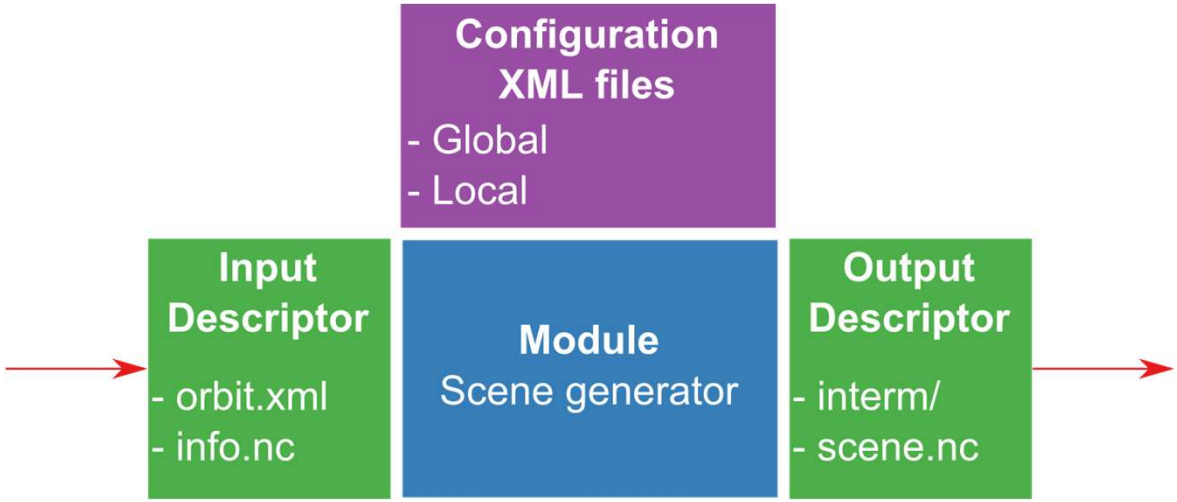
- Descriptors define input and output interfaces for modules
- One input and one output descriptor per module
- A descriptor represents a file/folder or a set of them





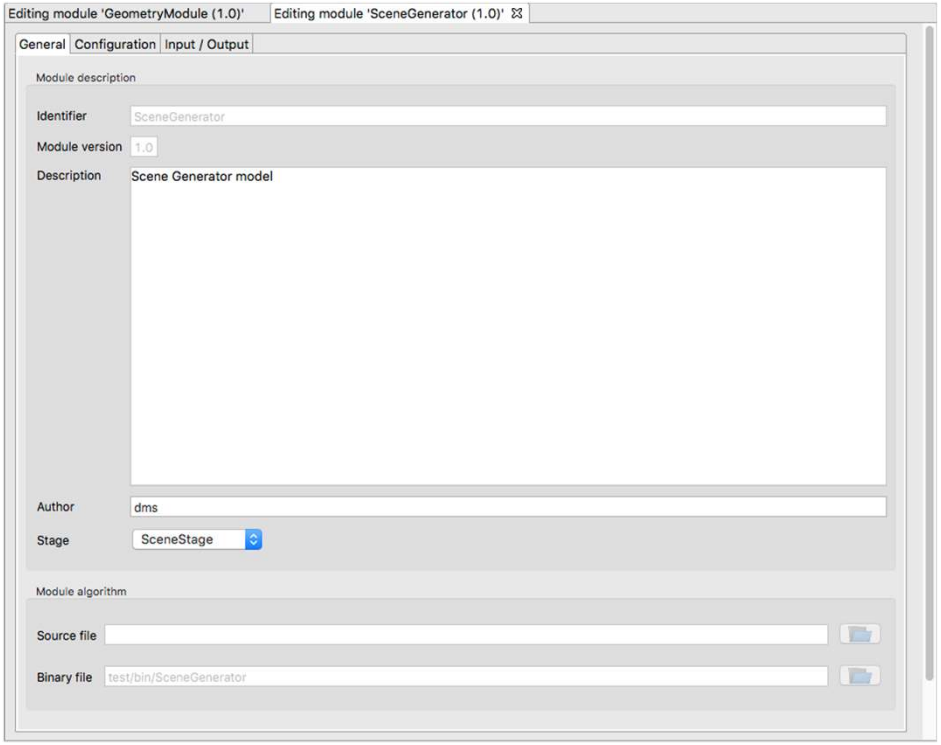
Modules, input/output and configuration

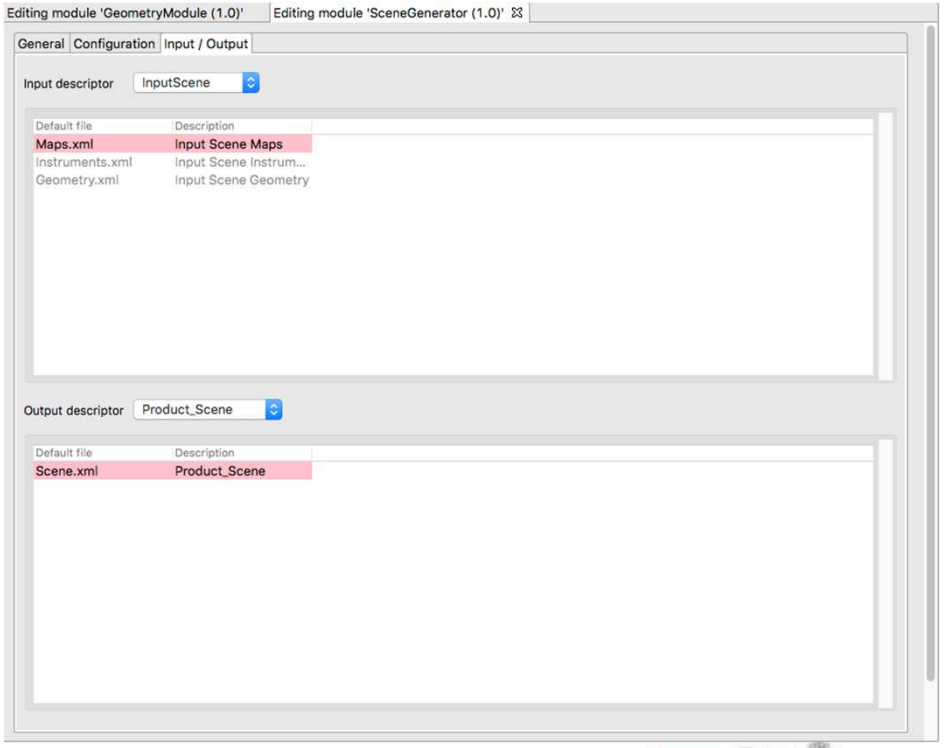
- Module: executable entity of the simulation
- Defined with
  - Input descriptor
  - Output descriptor
  - 2 configuration XML files
    - *Global configuration file: parameters shared in the simulation*
    - *Local configuration file: module specific parameters*



Modules, input/output and configuration

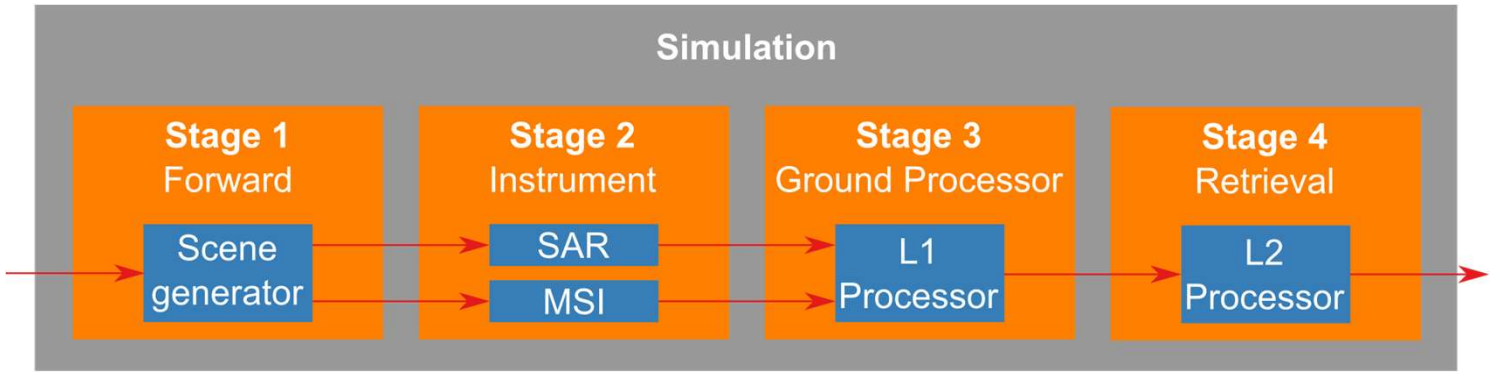
- Module
  - Input descriptor
  - Output descriptor
  - Configuration XML files





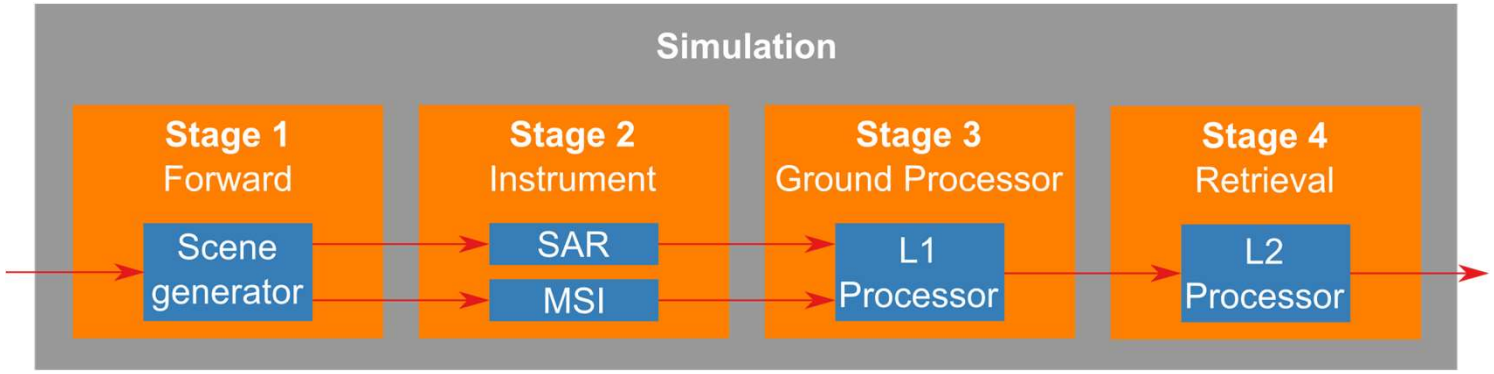
Stages

- Defines a phase in a simulation process
  - Easier to define meaningful E2Es simulations
- A module must be associated to a stage
  - Multiple modules can be associated to the same stage
- The stage order specifies the logic of the simulation



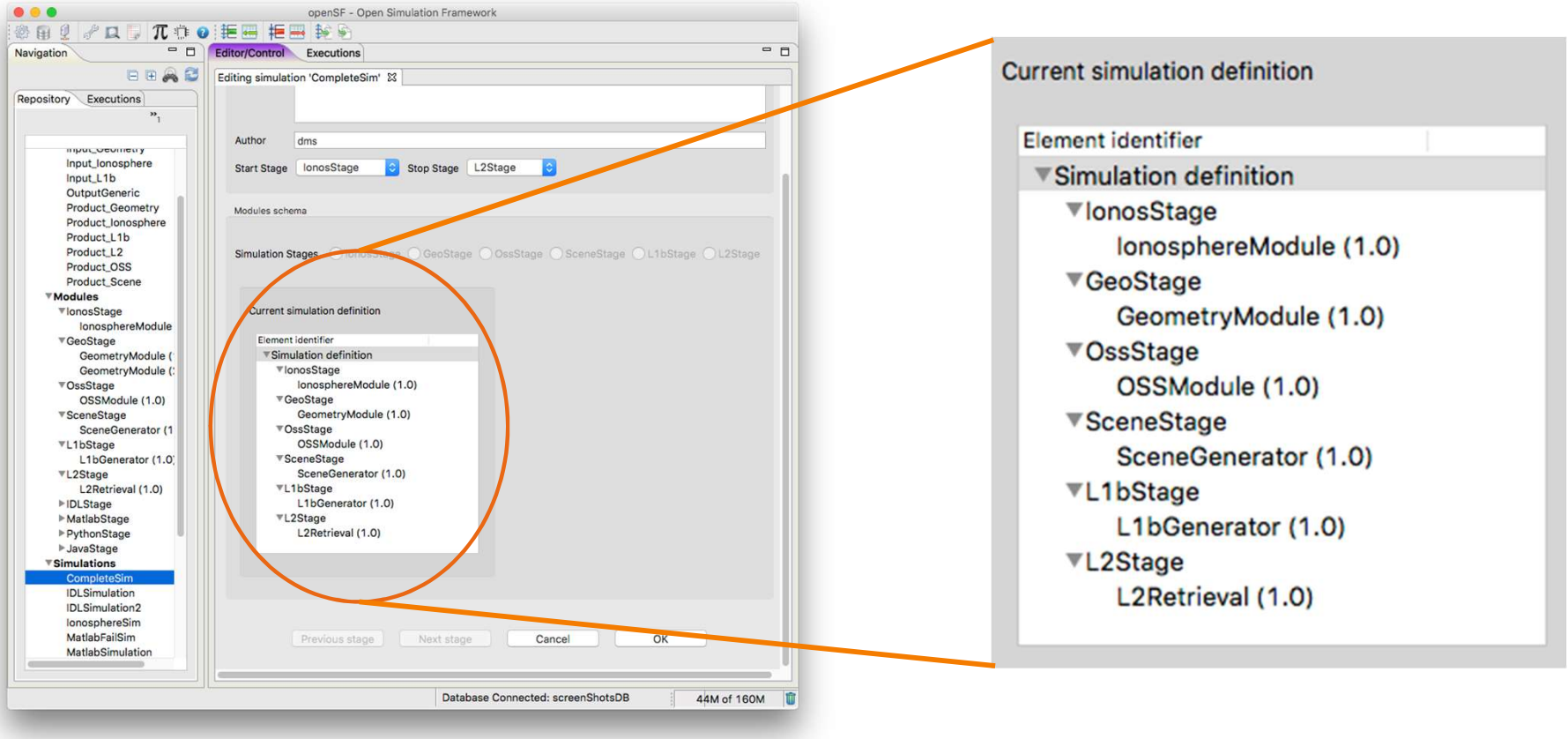
Simulations

- List of modules that is run sequentially
  - A simulation is used as a **template for the session definition**
  - Useful to have pre-defined meaningful sequences, but not necessary – **sessions can be built by adding modules one by one**.
- Orchestration
  - The execution sequence is defined by stages order



Simulations

- Pre-defined meaningful sequence



Sessions

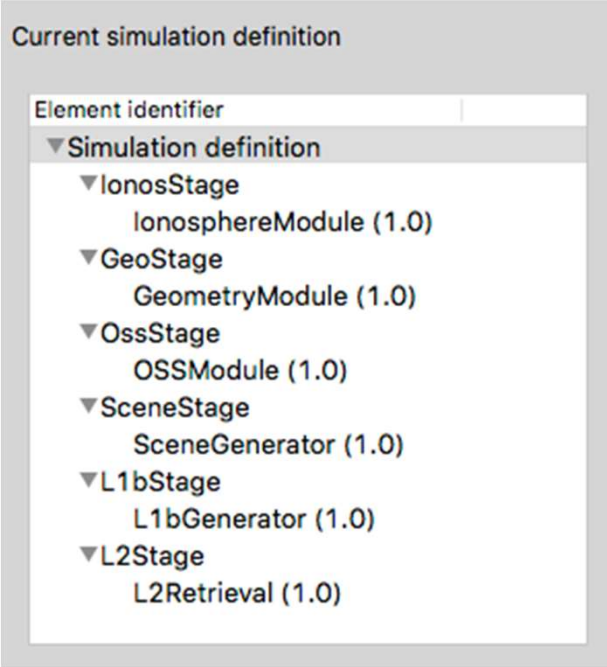
- Module chain that is actually run
  - Can be created by
    - ***Adding simulations***
    - *Adding modules one at a time*
  - Is a collection of modules

- Simulation vs session
  - **The session is the only openSF element that is run**
  - Simulations are useful for having pre-defined meaningful sequences
  - Simulations are just templates → cannot be run

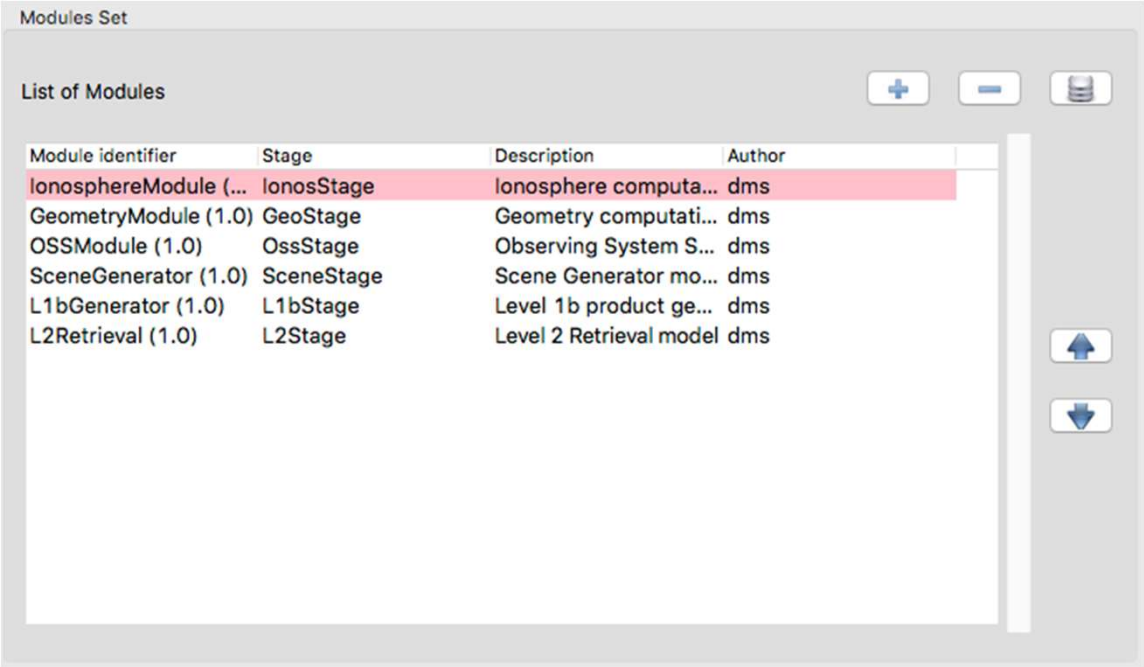
Sessions

- Simulation vs session

Simulation



Session



Other Elements

Tools

- Outputs can be associated to tools
  - A tool is an external executable that performs an action given a file
  - Tools are associated to a file extension
    - *E.g. Gimp to .tiff , Gnuplot to .gp*

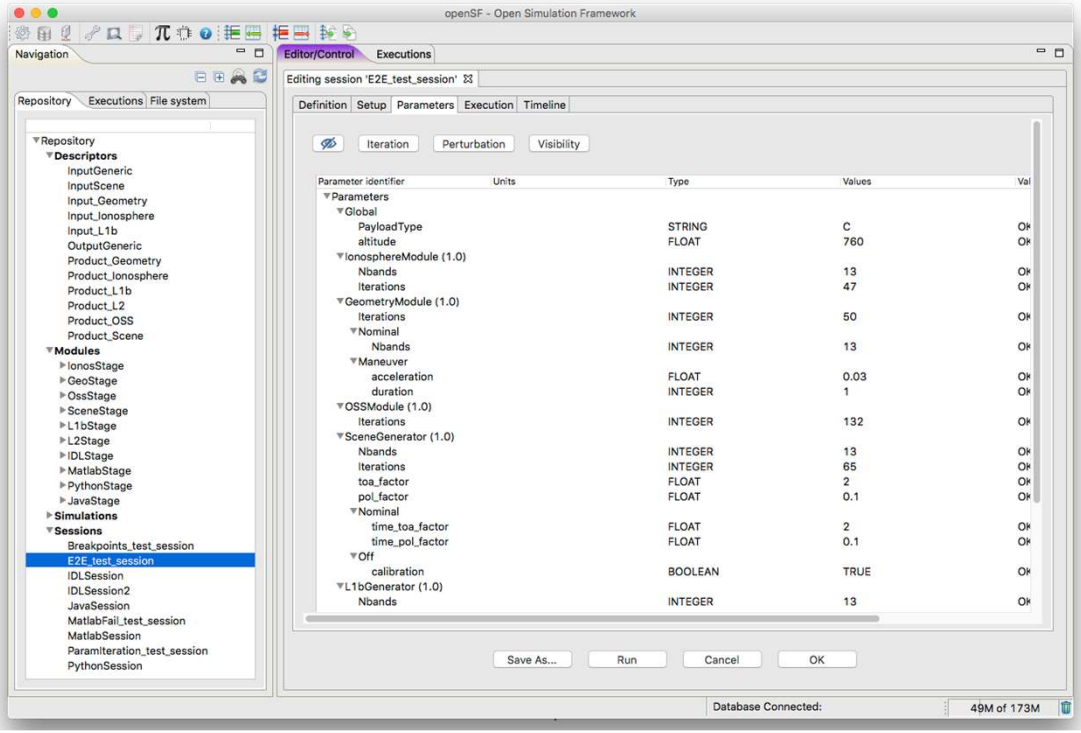
Repository

- OpenSF workspace associated to a defined processing chain
  - Comprises stages, descriptors, modules, simulations, sessions, executions and tools
- Each repository is stored in a MySQL database

Example Sessions

The example sessions are distributed within the default database

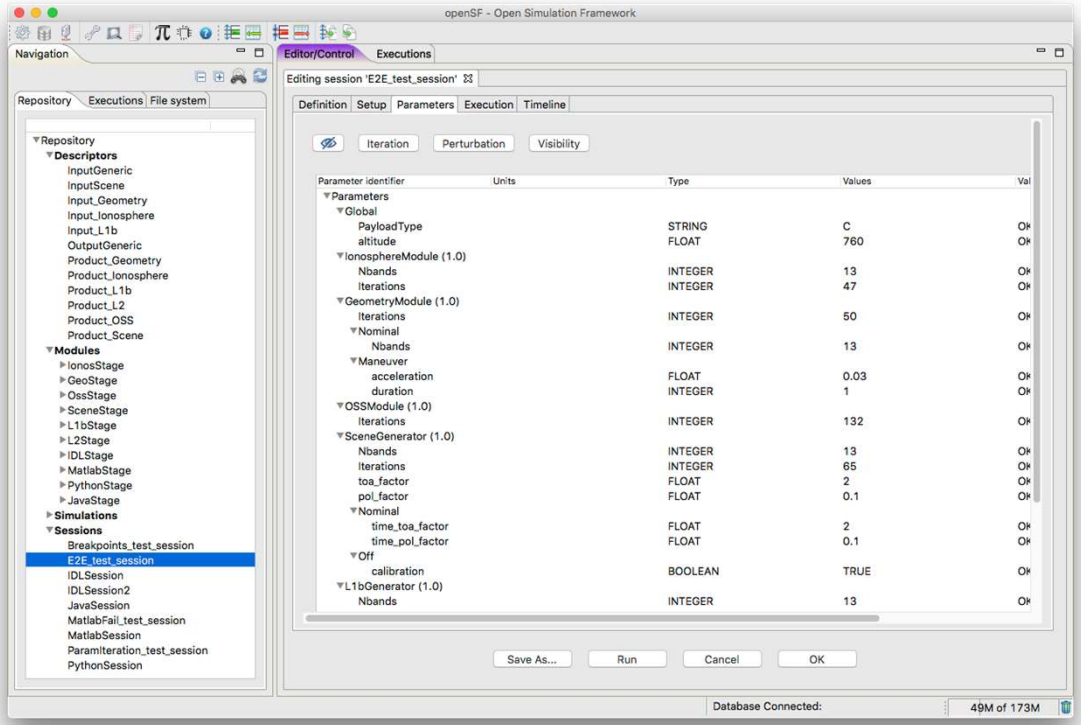
- Purpose
  - Shows how to structure a complex openSF database
  - Installation sanity check
- Location
  - At test\data\database



Example Sessions

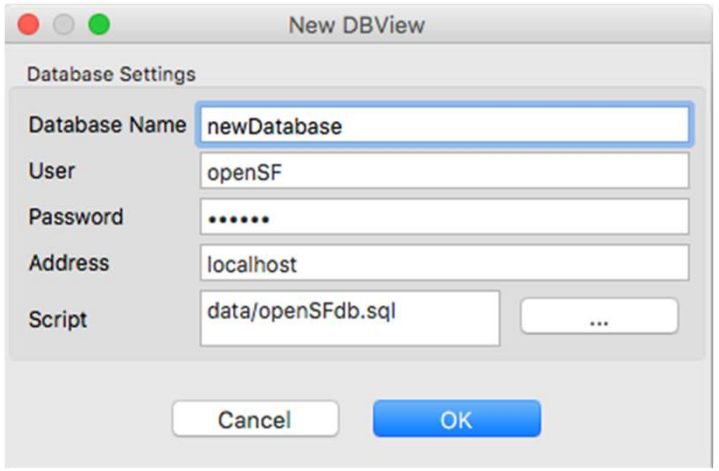
Sessions included

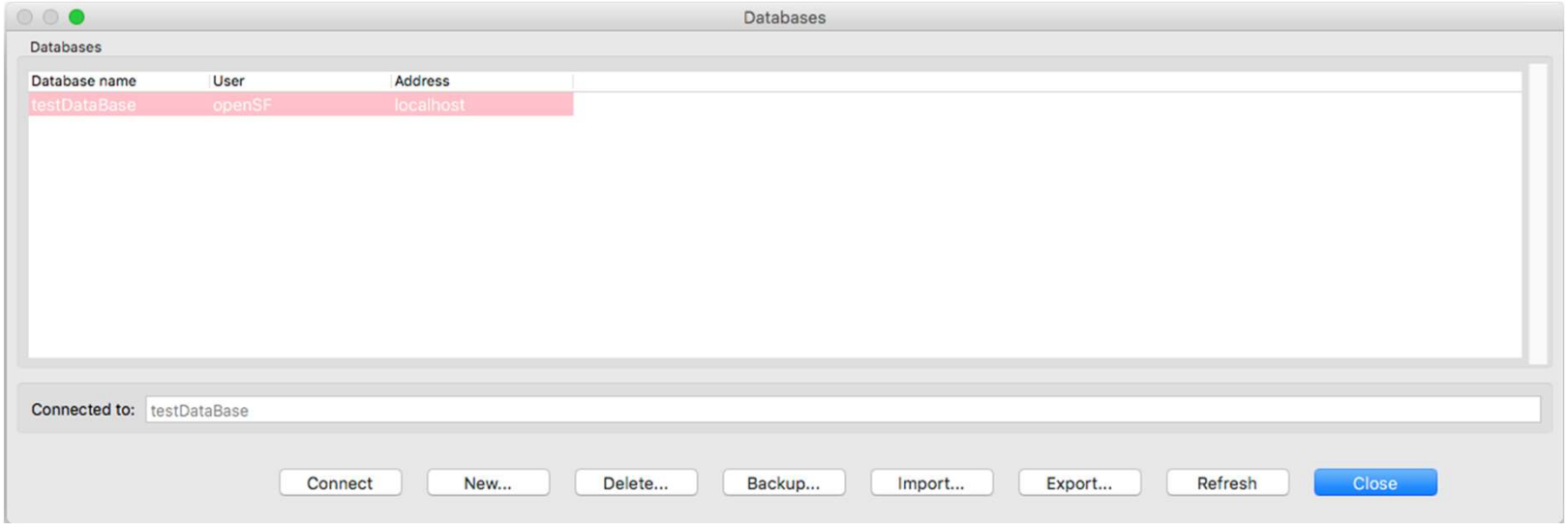
- E2E session with multiple modules
- Dedicated simple sessions:
  - IDL, Matlab, Python, Java, breakpoints



Database functionalities

- From openSF the user can
  - Connect to an existing database
  - Create a new database
    - *Empty or from a existing SQL file*
  - Backup database
    - *Dump to SQL file*
  - Import/export database from/to xml file





Installation and database setup

- Installation
  - Get the MySQL 5.6/5.7 installer from the official website
  - Start the MySQL server and check its status

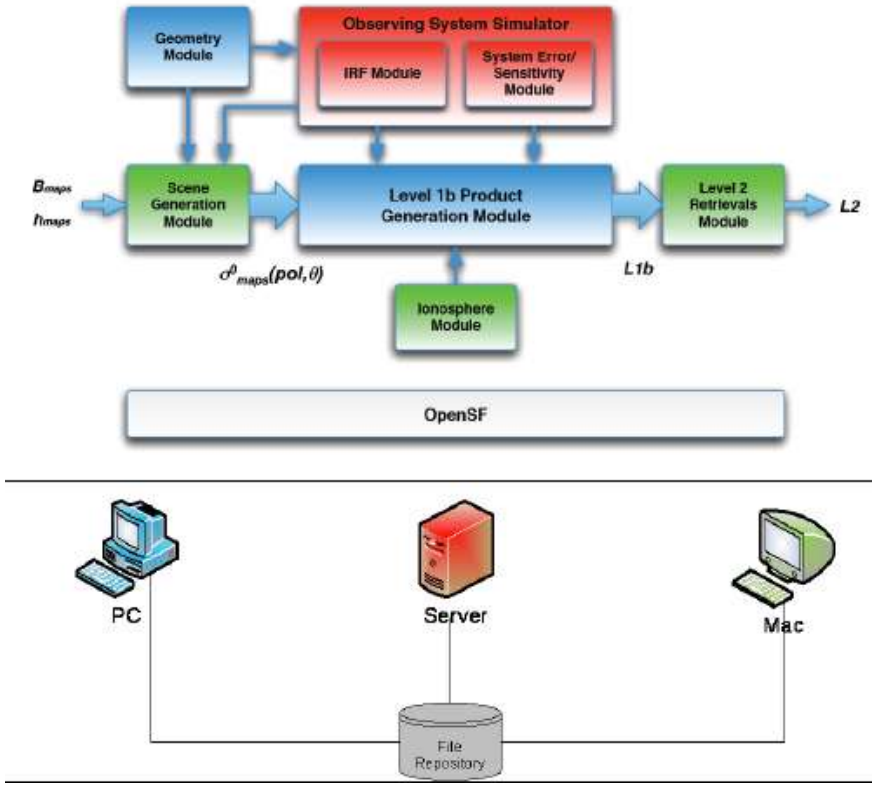
- Set database
  - Create a user with all privileges
    - *Check: https://dev.mysql.com/doc/refman/5.7/en/adding-users.html*

Upgrade procedure

- Database side
  - The installer allows an existing database from openSF v3 to be upgraded
  - The upgrade is performed by openSF itself on opening the DB, so multiple databases can be upgraded by a single installation
  - Recommendation: make a backup/export the DB before upgrading
- Modules side
  - The format of configuration files was updated in 2016 → check compliance with current E2E generic ICD and update older modules if necessary
  - Until v3.7.1, each version of openSF had an associated version of OSFI, which was forced upon the modules through (DY)LD_LIBRARY_PATH
    - *Also, the OSFI shared libraries do not enforce backwards binary compatibility → module compiled against OSFI 3.2 may not execute if run against OSFI 3.3, etc.*
    - *This means that every new version required rebuilding the modules*
  - OSFI binaries are not distributed with openSF. The module developer/integrator is responsible for providing runtime requirements avoiding any future incompatibility
    - *With this new version, modules have to be rebuilt one last time*
    - *Further updates should no longer need module re-build*

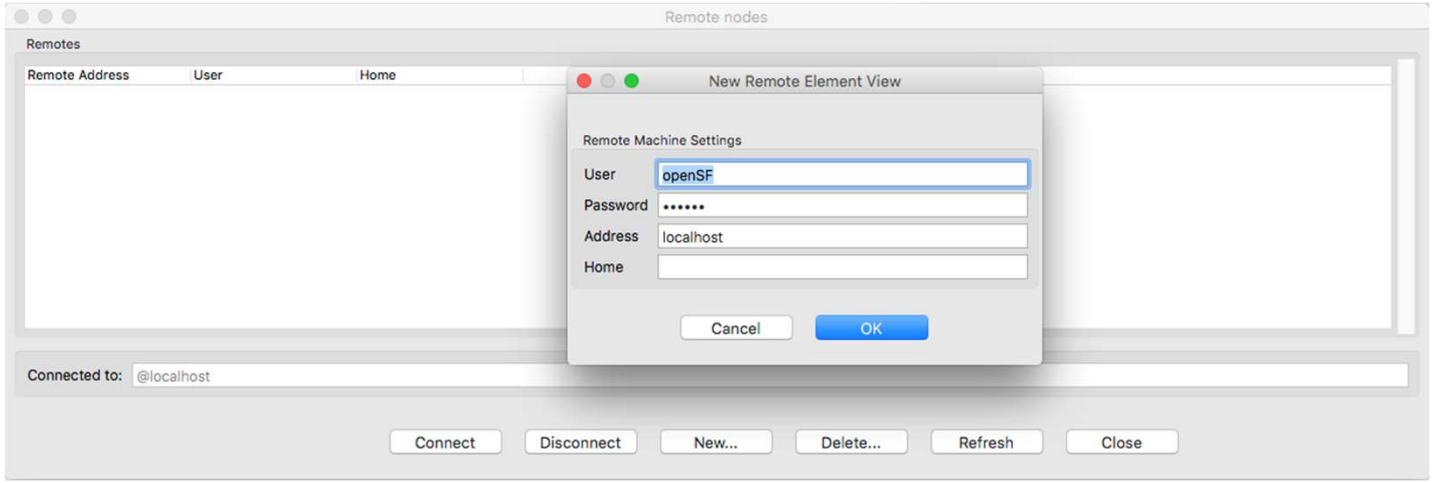
Multi-node simulation

- Remote execution of multiple modules in a session run
- The user can choose which machine to use for each module to execute



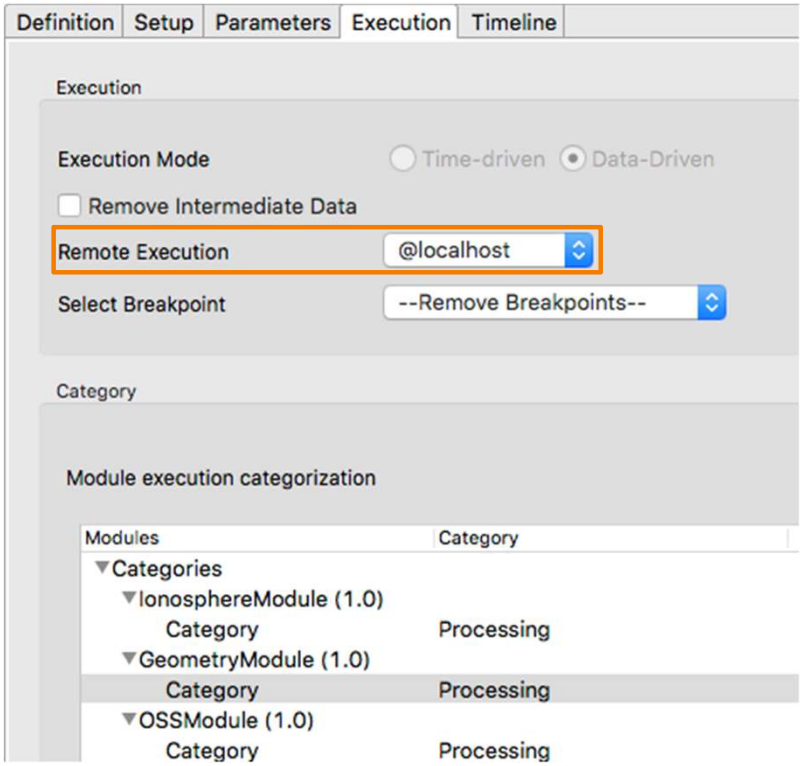
Remote machine management

- New remote machine
  - Provide IP, user, password and openSF location
- Connect to a new machine
  - Session products generated in the file system of a remote machine
  - Session system folder used by openSF is located in a remote machine
- Other operations
  - Disconnect, delete, refresh etc



Remote execution

- Remote machines selection
  - Single machine for the whole session
  - Machine selection module by module

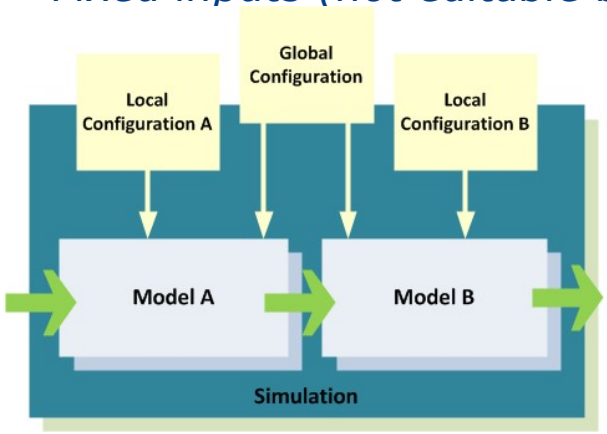


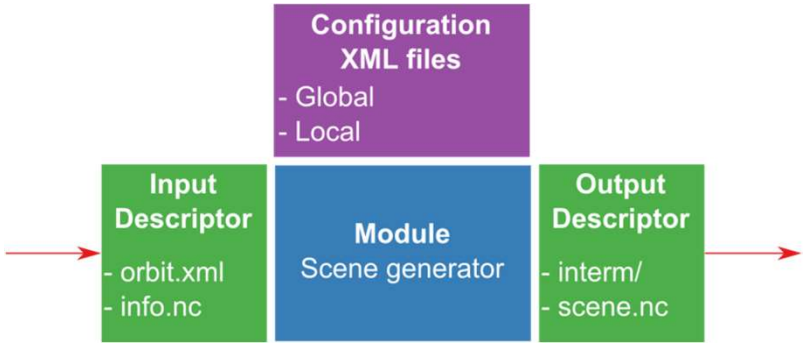
5 Module Developer Tips

Module Inputs

Each module has two input sources

- Configuration files
  - Two configuration files per module
    - *Global configuration file: **parameters** shared in the simulation*
    - *Local configuration file: model specific **parameters***
  - **Parameters**: module inputs that are editable by the user via GUI and usable automatically
- Input files
  - Can be of two types
    - *Outputs of previous modules*
    - *Fixed inputs (not editable by the user via openSF): e.g. data files, folders*





Configuration files

- Should contain those parameters **editable** by the user
- Agree with end-user (e.g. engineer and scientist) the parameter list to allow openSF parameter iteration (e.g. to support error sensitivity iteration)
  - Module specific parameters → local configuration file
  - Parameters shared between modules → global configuration file
- Use the **Parameter Editor** to create the configuration files

Configuration files

- Should contain those parameters **editable** by the user
- Agree with end-user (e.g. engineer and scientist) the parameter list to allow openSF parameter iteration (e.g. to support error sensitivity iteration)
  - Module specific parameters → local configuration file
  - Parameters shared between modules → global configuration file
- Use the **Parameter Editor** to create the configuration files

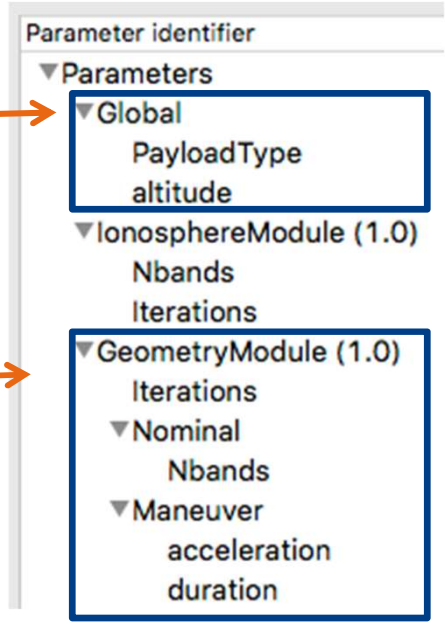
If the parameters are **not** defined in configuration files, **the user cannot edit them via openSF UI** negating the time-saving benefit of automation

Configuration files

- Global configuration file
  - Shared parameters
  - Avoid inconsistencies between modules
- Local configuration file
  - One per module
  - Module specific parameters

Configuration files

- Global configuration file
  - Shared parameters
  - Avoid inconsistencies between modules
- Local configuration file
  - One per module
  - Module specific parameters



Module Inputs

Input files

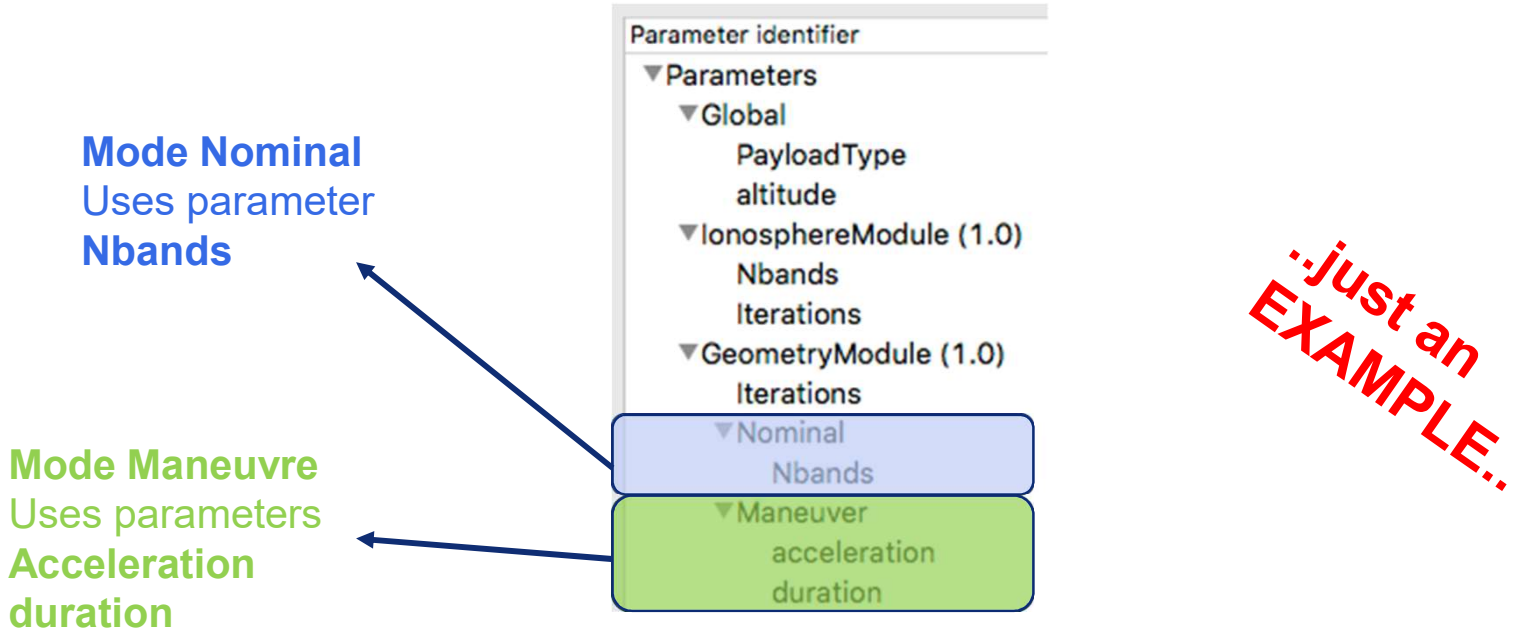
- Usually outputs from previous modules, building the chain
- An exception can be a file needed from a module that is provided by a source external to the simulation chain

Large input files

- All inputs are **copied** into the session folder by openSF
- Do not pass large input files to the module via descriptor (e.g. ephemerides, atmosphere models, etc.)
- Workaround: define a **parameter** in the configuration file or an input file specifying the **location** of the large input file (parameter type file/folder)
  - DON'T: input file /ephdata/de430.bsp
  - DO: configuration parameter "eph.spk_file = /ephdata/de430.bsp"

Time orchestration

- Very useful/time saving for end user but must be thought from beginning
- Module must be coded to be "**mode**" aware (as taken from Global Configuration File):



Module Outputs

Logging

- Should follow E2E generic ICD format. If not

→ NO filtering and no sorting of messages and no use of GUI

- Recommendation: use the Logger module from OSFI

Output files

- Unlike inputs, they are **not copied** (between modules or at the end) so generating a large dataset is not a problem
- If the number of output files is not known at module definition time (e.g. it is parameter-dependent), a **folder** can be designated as an output too.
  - However, openSF cannot check whether it contains the right files before calling the next module or at the end

Other Bad Practices (for users)

Incorporating geometric/orbital calculation within modules and not as standalone module→

- prevents use of external generation of geometrical/orbital inputs

Reuse of filename for input and output and across modules →

- does not allow openSF to check that file is available.

Use of CCDB folder as a file-system tree input in simulation →

- opaque use of information in modules, hidden selection of inputs, must rely only on logging to know actual input. Recommend to select and extract explicitly the input files from it and pass them to modules.

6 Integrator Tips

Parameters Perturbation and Iteration

Execute a session multiple times

- The user can execute a session manually changing the value of some parameters

Automatic Parameter iteration

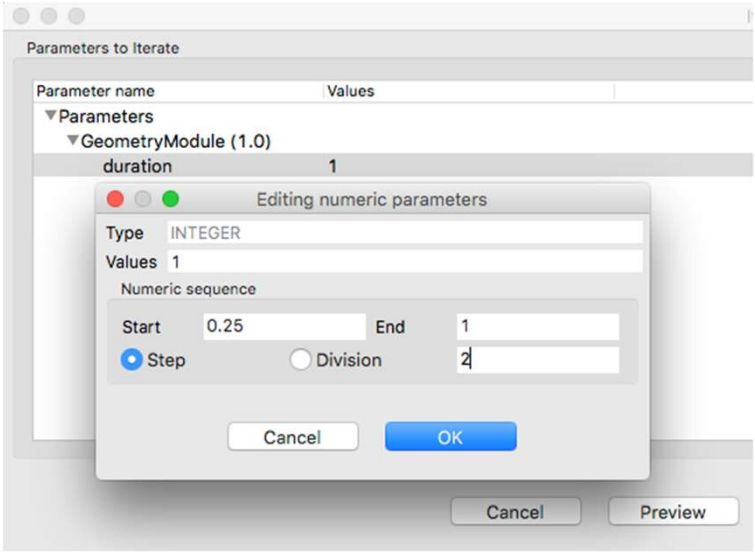
- Array of parameter values provided by the user
  - Manually entering a set of values
  - Minimum and maximum value and step/number of elements
- Useful for sensitivity analysis

Automatic Parameter perturbation

- For complex variations of parameter values
  - Analytical: polynomial, sinusoidal, etc
  - Binary operation: addition, root, etc
  - Custom function
  - Random: normal, Poisson, uniform, etc
- User specifies number of shots for each parameter
- Useful for Monte Carlo analysis

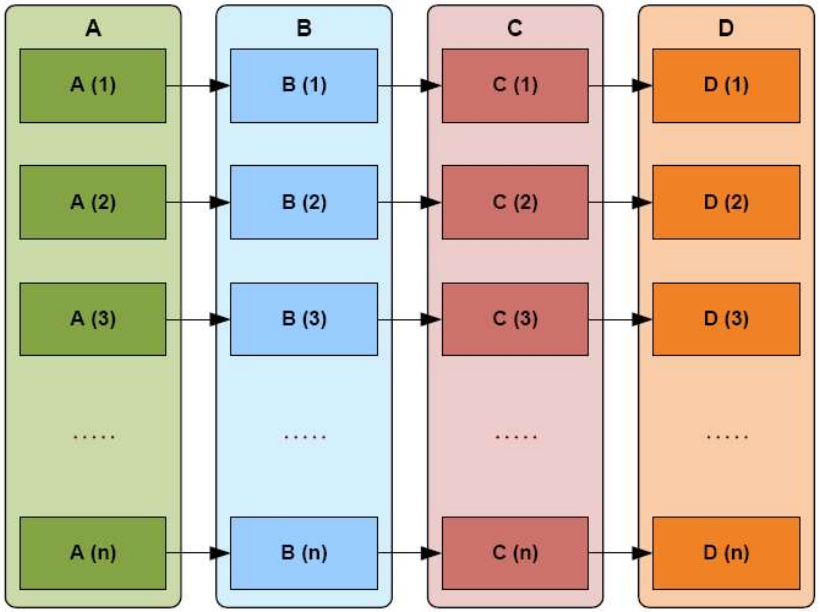
Iterative session

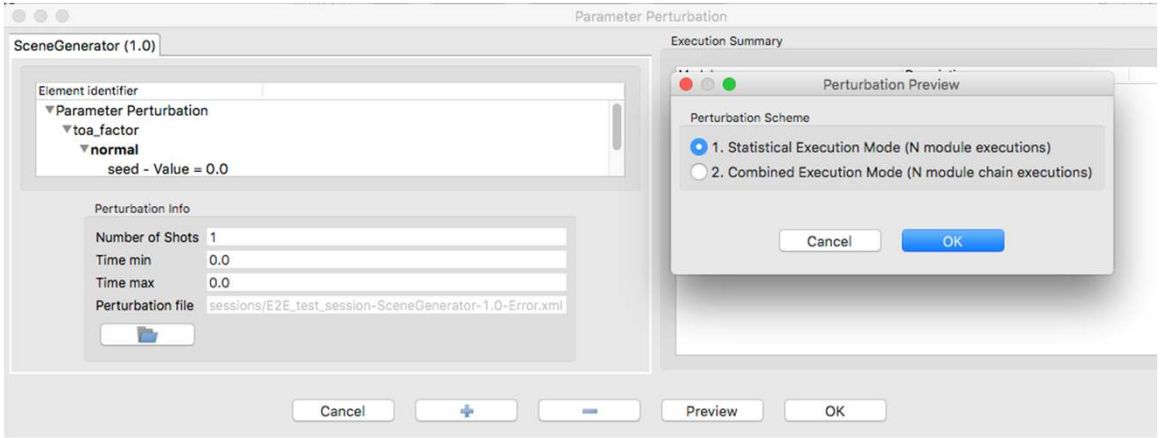
- Oriented to perform **sensitivity analysis**
- Automatically executes a session multiple times with different parameter values
  - 2 parameters with 2 values each → 4 executions
- Each execution changes the value of only one parameter
- Results of each execution stored in separated session folders
  - The user has to collect the results



Perturbed execution modes → Combined Mode

- Oriented to perform **Monte Carlo** analysis of **simulation chain**
- Executes a session as many times as shots specified
- All parameters are perturbed at the same time for each simulation
- All perturbations shall have the same number of shots
- Results of each execution stored in separated session folders
  - The user has to collect the results





7 Demo

openSF installation (linux and OSX)

Step-by step full nominal process to build/integrate a simulator

- create descriptor, models, stages, simulation, etc. using openSF

Example cases delivered with openSF (validation DB)

Parameter Editor

Definition of an example full simulator architecture

- Definition of the stages
- Creating modules
  - I/O descriptors
  - Configuration file
  - Module object
- Putting together a session
  - Changing configuration for a single instance vs. “save as…”
- Making everything run
  - Single shot
  - Multiple shots (iteration)

8 Exercise

Repeat the demo with the definition of a full simulator

- Definition of the stages
- Creating modules
  - I/O descriptors
  - Configuration file
  - Module object
- Putting together a session
  - Changing configuration for a single instance vs. “save as…”
- Making everything run
  - Single shot
  - Multiple shots (iteration)

9 Q&A

Thank you

www.deimos-space.com