use

LIME TOOLBOX TECHNICAL SPECIFICATION DOCUMENT

ABSTRACT

This document provides the LIME Toolbox technical specifications for the LIME-2 project.

Ramiro González
Carlos Toledano
Javier Gatón
Pieter De Vis
Jacob Fahy
Stefan Adriaensen

UVa, NPL, VITO

27 January 2022

This document was produced as part of the ESA-funded project "Improving the Lunar Irradiance Model of ESA" under ESA contract number:

Signatures and version history

	Name	Organisation	Date
Written by	Carlos Toledano	UVa	08 January 2022
Reviewed by (consortium)		NPL, VITO	27 January 2022
Approved by (ESA)	Marc Bouvet	ESA	05 April 2022

Version history

Version	Date	Publicly available or private to consortium?
0.1	08/01/2022	Public

Contents

1 Introduction

1.2 Purpose and Scope

This document provides the LIME Toolbox technical specifications once the requirements for the toolbox have been agreed with ESA in the KO+2 project meeting (deliverable D-6). It includes the description of the software architecture and the implementation plan.

1.3 Applicable and reference documents

1.3.1 Applicable Documents

The following applicable documents are those specification, standards, criteria, etc. used to define the requirements of this task.

Number	Reference
[AD0]	ESA-EOPG-EOPGMQ-SOW-24. Improving the Lunar Irradiance Model of ESA.

1.4 Glossary

1.4.1 Abbreviations

Abbreviation	Stands For	Notes
ASD	Analytical Spectral Devices	Instrument manufacturer
Cimel	(Not an abbreviation)	Instrument manufacturer, also used as shorthand for instrument itself
EO	Earth Observation	
ESA	European Space Agency	Project customer
FOV	Field of View	
GIRO	GSICS Implementation of the ROLO Model	
GSICS	Global Space Based Inter-calibration System	
GUI	Graphical User Interface	
KO	Kick-off meeting	
LIME	Lunar Irradiance Model of ESA	
NPL	National Physical Laboratory	Project partner
ROLO	RObotic Lunar Observatory	
SoW	Statement of Work	
SRF	Spectral Response Function	
SW	Software	
TBX	Toolbox	
TOA	Top of Atmosphere	
UVa	University of Valladolid	Project partner
VITO	Vlaamse Instelling voor Technologisch Onderzoek; Flemish Institute for Technological Research	Project partner

2 User requirements

Within this project, the WP2 objective is to develop a LIME toolbox allowing to easily run the LIME model and compare its outputs to remote sensing measurements, especially ESA satellite sensors.

The LIME TBX requirements were described in Annex 1 of the SoW (AD0) and agreed with ESA in deliverable D6 (LIME TBX requirement document). The main functionality is to simulate lunar observations for any observer position or satellite sensor (with specified SRF) at any date/time. In that document, the requirements are graded as Critical, Major and Minor, as follows:

- Critical: Core to the software, must be met.
- Major: Improves the software, should be met.
- Minor: Useful, but not critical or major. If it cannot be implemented in a first release, perhaps it can be implemented later.

Note that the TBX must include the developments of WP1 of this project concerning the hyperspectral interpolation in the model, which is very important update of the LIME model with respect to the current version, as well as an algorithm allowing to compute uncertainties for each lunar irradiance simulation across the full spectral range.

3 LIME Toolbox software architecture

According to the abovementioned requirements, the TBX architecture will need to be modular, in order to build the various functionalities in parallel by the most appropriate partner in the consortium, as well as to be able to add modules as they are available to the project. Another important aspect is that it must be able to integrate existing libraries (e.g. NPL punpy module for uncertainty propagation). The programming language will be python.

The main elements or modules of the software are given in the following table, together with the responsible partner. The planned implementation schedule is given in Table 2.

	Led by
Graphical user interface (GUI)	UVa
Regular simulation	UVa
Comparison tool	UVa
Update	UVa
ESA satellite observation simulator	tbd
Main LIME algorithm	all
DOLP	all
EOCFI Adapter	UVa
Spectral integration	NPL
Spectral interpolation	NPL
Propagate uncertainties	NPL
Standard format	NPL
User manual and documentation	all

Table 1. List of LIME TBX modules.

The organisation of these elements is provided in Figure 1, where the relations between them are also indicated. Input and output details are described in Figure 2.

The detailed design of each module will follow the specific requirements provided in deliverable D6 concerning the functionalities, the GUI, inputs and outputs, coding, libraries and accessibility. The interfaces to external elements like the GLOD dataset or EOCFI will be developed too.

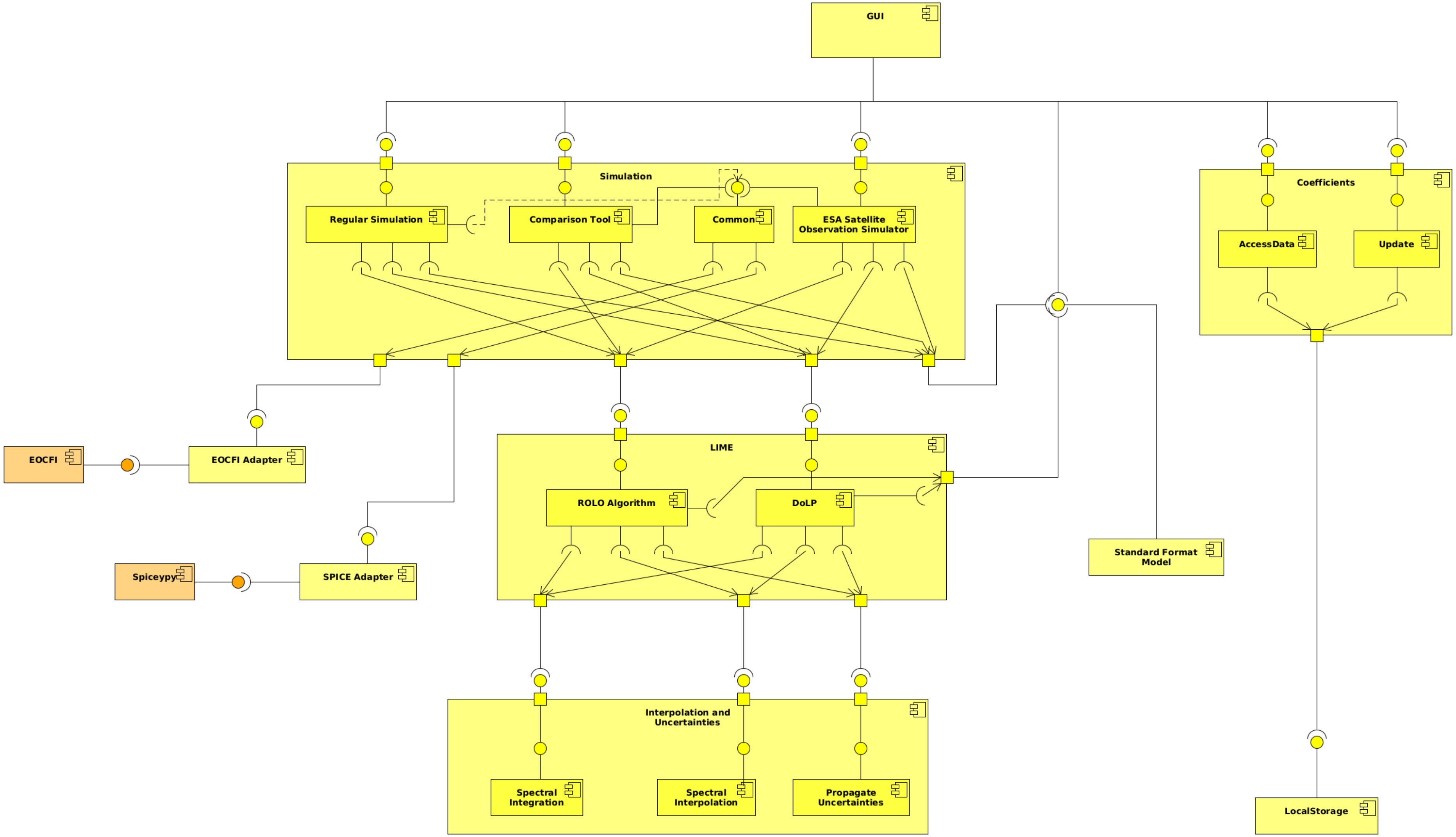
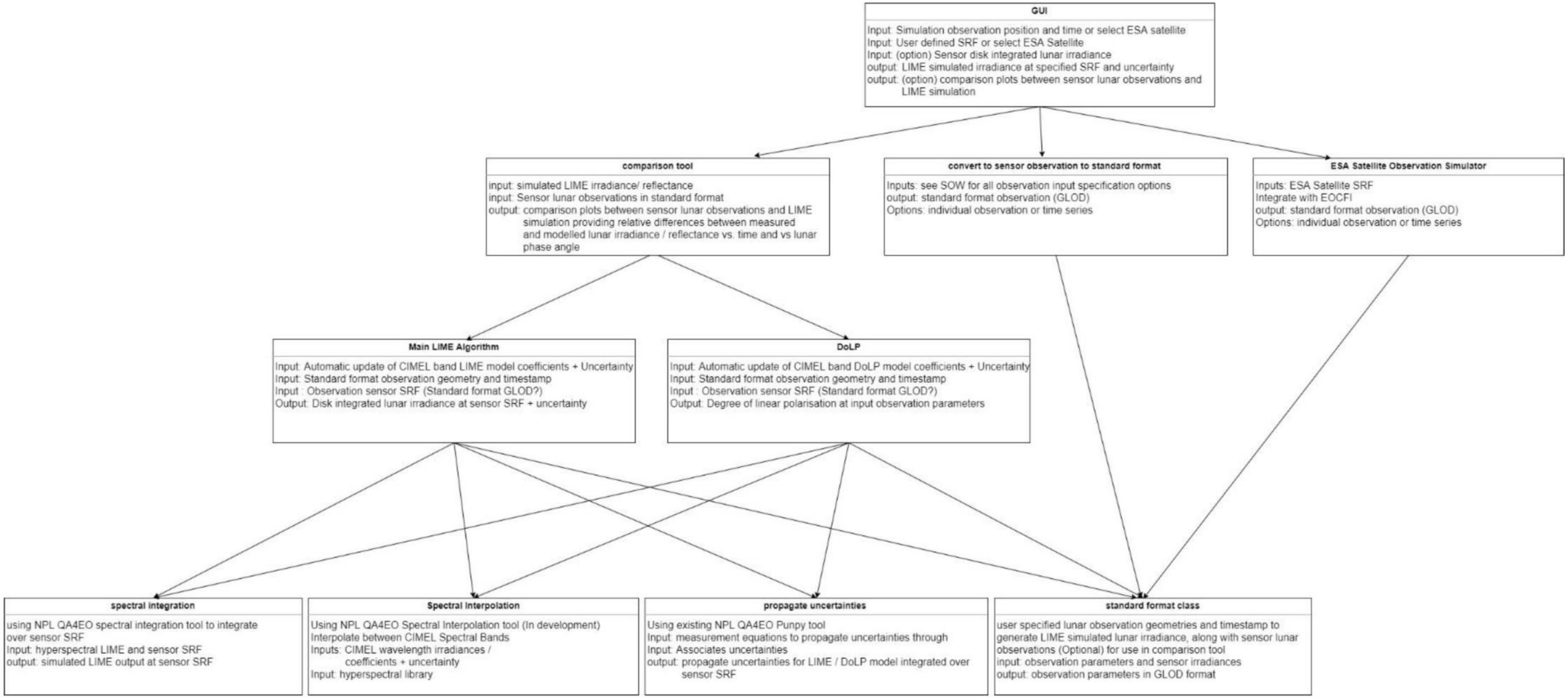


Figure 1. LIME TBX component diagram.



GUI

Input: Simulation observation position and time or select ESA satellite
Input: User defined SRF or select ESA Satellite
Input: (option) Sensor disk integrated lunar irradiance
output: LIME simulated irradiance at specified SRF and uncertainty
output: (option) comparison plots between sensor lunar observations and LIME simulation

comparison tool

input: simulated LIME irradiance/ reflectance
input: Sensor lunar observations in standard format
output: comparison plots between sensor lunar observations and LIME simulation providing relative differences between measured and modelled lunar irradiance / reflectance vs. time and vs lunar phase angle

convert to sensor observation to standard format

Inputs: see SOW for all observation input specification options
output: standard format observation (GLOD)
Options: individual observation or time series

ESA Satellite Observation Simulator

Inputs: ESA Satellite SRF
Integrate with EOCFI
output: standard format observation (GLOD)
Options: individual observation or time series

Main LIME Algorithm

Input: Automatic update of CIMEL band LIME model coefficients + Uncertainty
Input: Standard format observation geometry and timestamp
Input : Observation sensor SRF (Standard format GLOD?)
Output: Disk integrated lunar irradiance at sensor SRF + uncertainty

DoLP

Input: Automatic update of CIMEL band DoLP model coefficients + Uncertainty
Input: Standard format observation geometry and timestamp
Input : Observation sensor SRF (Standard format GLOD?)
Output: Degree of linear polarisation at input observation parameters

spectral integration

using NPL QA4EO spectral integration tool to integrate over sensor SRF
Input: hyperspectral LIME and sensor SRF
output: simulated LIME output at sensor SRF

Spectral Interpolation

Using NPL QA4EO Spectral Interpolation tool (In development)
Interpolate between CIMEL Spectral Bands
Inputs: CIMEL wavelength irradiances / coefficients + uncertainty
Input: hyperspectral library

propagate uncertainties

Using existing NPL QA4EO Punpy tool
Input: measurement equations to propagate uncertainties through
Input: Associates uncertainties
output: propagate uncertainties for LIME / DoLP model integrated over sensor SRF

standard format class

user specified lunar observation geometries and timestamp to generate LIME simulated lunar irradiance, along with sensor lunar observations (Optional) for use in comparison tool
input: observation parameters and sensor irradiances
output: observation parameters in GLOD format

Figure 2. LIME TBX class diagram. Input and output products are indicated for each module.

3.1 GUI

This module consists of the Graphical User Interface. It will contain the software for the creation of all views and widgets that will allow the user to perform all use cases.

During development this module will probably be subdivided in multiple modules in order to encapsulate its functionalities.

It will use modules from the “Simulation” module in order to perform the main use cases, the “Standard Format Model” module in order to read GLOD format files and spectral response SRF files, and the Update module in order to perform updates.

3.2 Regular Simulation

This module performs the simulations from a location on Earth’s surface and for custom selenographic coordinates.

It exports the following functions:

- **get_eli_from_surface**(srf, latitude, longitude, altitude, datetime):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **latitude**: Geographic latitude of the observer.
        - **longitude**: Geographic longitude of the observer.
        - **altitude**: Altitude of the observer.
        - **datetime**: Date and time at which simulate the extra-terrestrial lunar irradiance.
    - Output:
        - A list containing the simulated extra-terrestrial lunar irradiances for the input data, along with their respective uncertainties.
- **get_elref_from_surface**(srf, latitude, longitude, altitude, datetime):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **latitude**: Geographic latitude of the observer.
        - **longitude**: Geographic longitude of the observer.
        - **altitude**: Altitude of the observer.
        - **datetime**: Date and time at which simulate the extra-terrestrial lunar irradiance.
    - Output:
        - A list containing the simulated extra-terrestrial lunar reflectances for the input data, along with their respective uncertainties.
- **get_polarized_from_surface**(srf, latitude, longitude, altitude, datetime):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **latitude**: Geographic latitude of the observer.
        - **longitude**: Geographic longitude of the observer.
        - **altitude**: Altitude of the observer.
        - **datetime**: Date and time at which simulate the lunar disk degree of polarization.
    - Output:
        - A list containing the simulated lunar disk degrees of polarization for the input data, along with their respective uncertainties.

- **get_eli_from_custom**(srf, distance_sun_moon, distance_observer_moon, selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **distance_sun_moon**: Distance between the Sun and the Moon on the simulation instant.
        - **distance_observer_moon**: Distance between the observer and the Moon at the simulation instant.
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - A list containing the simulated extra-terrestrial lunar irradiances for the input data, along with their respective uncertainties.
- **get_elref_from_custom**(srf, distance_sun_moon, distance_observer_moon, selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **distance_sun_moon**: Distance between the Sun and the Moon on the simulation instant.
        - **distance_observer_moon**: Distance between the observer and the Moon at the simulation instant.
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - A list containing the simulated extra-terrestrial lunar reflectances for the input data, along with their respective uncertainties.
- **get_polarized_from_custom**(srf, selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **distance_sun_moon**: Distance between the Sun and the Moon on the simulation instant.
        - **distance_observer_moon**: Distance between the observer and the Moon at the simulation instant.
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - A list containing the simulated lunar disk degrees of polarization for the input data, along with their respective uncertainties.

3.3 Comparison Tool

This module performs comparisons of lunar observations from a remote sensing instrument.

It exports the following functions:

- **compare_observations**(glod_instrument_observations):
    - Input:
        - **glod_instrument_observations**: Observations from the remote sensing instrument in glod format.
    - Output:
        - The comparison plots shall provide: relative differences between measured and modeled lunar irradiance/reflectance vs. time and vs. lunar phase angle.
        - The comparison plots shall display statistical indicators (mean relative difference, standard deviation of the mean relative difference, temporal trend if applicable, number of comparison samples, etc…)
    - Note: This might change during development as this might not be the most efficient approach.

3.4 ESA Satellite Observation Simulator

This module performs the simulations from an ESA satellite.

It exports the following functions:

- **get_eli_from_satellite**(srf, satellite, datetime):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **satellite**: ESA satellite of which simulate the extra-terrestrial lunar irradiance.
        - **datetime**: Date and time at which simulate the extra-terrestrial lunar irradiance.
    - Output:
        - A list containing the simulated extra-terrestrial lunar irradiances for the input data, along with their respective uncertainties.
- **get_elref_from_satellite**(srf, satellite, datetime):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **satellite**: ESA satellite of which simulate the extra-terrestrial lunar irradiance.
        - **datetime**: Date and time at which simulate the extra-terrestrial lunar irradiance.
    - Output:
        - A list containing the simulated extra-terrestrial lunar reflectances for the input data, along with their respective uncertainties.
- **get_polarized_from_satellite**(srf, satellite, datetime):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **satellite**: ESA satellite of which simulate the lunar disk degree of polarization.
        - **datetime**: Date and time at which simulate the lunar disk degree of polarization.
    - Output:

        - A list containing the simulated lunar disk degrees of polarization for the input data, along with their respective uncertainties.

3.5 Common

This module performs functionalities common to all Simulation submodules.

The exported functions are not defined yet, but there will be common functions as Simulation submodules are very cohesive.

3.6 Access Data

This module accesses the values of all of the locally stored data, like coefficients.

It exports the following functions:

- **get_all_coefficients_irradiance**():
    - Output:
        - All coefficients used in the calculation of the extra-terrestrial lunar irradiance (and reflectance).
- **get_all_coefficients_polarization**():
    - Output:
        - All coefficients used in the calculation of the degree of polarization.

3.7 Update

This module checks for updates of the coefficients and downloads them.

It exports the following functions:

- **check_for_updates**():
    - Output:
        - Boolean value representing if there are updates.
- **download_coefficients**():
    - Output:
        - Boolean value representing if the download was successful.

3.8 Local Storage

This module stores data in local storage.

The exported functions are not defined yet, as the approach for local storage is not decided. There are multiple software patterns available that could be used and depending on the one chosen the exported functions would be different.

3.9 Main LIME Algorithm

This module calculates the extra-terrestrial lunar disk irradiance.

It exports the following functions:

- **get_eli**(wavelengths, distance_sun_moon, distance_observer_moon, selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **wavelengths**: List of wavelengths of which the extra-terrestrial lunar irradiance will be simulated.
        - **distance_sun_moon**: Distance between the Sun and the Moon on the simulation instant.

        - **distance_observer_moon**: Distance between the observer and the Moon at the simulation instant.
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - A list containing the simulated extra-terrestrial lunar irradiances for the input data, along with their respective uncertainties.
- **get_elref**(wavelengths, distance_sun_moon, distance_observer_moon, selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **wavelengths**: List of wavelengths of which the extra-terrestrial lunar reflectance will be simulated.
        - **distance_sun_moon**: Distance between the Sun and the Moon on the simulation instant.
        - **distance_observer_moon**: Distance between the observer and the Moon at the simulation instant.
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - A list containing the simulated extra-terrestrial lunar reflectances for the input data, along with their respective uncertainties.

3.10 DoLP

This module calculates the simulated lunar disk degree of polarization.

It exports the following functions:

- **get_polarized**(wavelengths, distance_sun_moon, distance_observer_moon, selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **wavelengths**: List of wavelengths of which the lunar disk degree of polarization will be simulated.
        - **distance_sun_moon**: Distance between the Sun and the Moon on the simulation instant.
        - **distance_observer_moon**: Distance between the observer and the Moon at the simulation instant.
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - A list containing the simulated lunar disk degrees of polarization for the input data, along with their respective uncertainties.

3.11 EOCFI Adapter

This module calculates the needed data for the simulation relative to ESA satellites using the EOCFI library.

It exports the following functions:

- **get_satellite_location**(satellite, datetime):
    - Input:
        - **satellite**: ESA Satellite identifier.
        - **datetime**: Date and time of the simulations.
    - Output:
        - The coordinates of the satellite (altitude included).

3.12 SPICE Adapter

This module contains functions that calculate needed data for the simulation, using the SPICE library.

These data consist of the selenographic coordinates of the observer, the selenographic longitude of the sun, the absolute moon phase angle for the observer, the distance between the observer and the moon, and the distance between the sun and the moon.

It exports the following functions:

- **get_moon_data_from_earth**(latitude, longitude, altitude, datetime):
    - Input:
        - **latitude**: Geographic latitude of the observer.
        - **longitude**: Geographic longitude of the observer.
        - **altitude**: Altitude of the observer.
        - **datetime**: Date and time at which perform the simulations.
    - Output:
        - The previously describe needed data, corresponding to the input data.

3.13 Standard Format Model

This module contains the common dataclasses that are used in order to transfer data between modules, and also the functionalities that read input data from GLOD format files and SRF files.

It exports the following functions:

- **read_glod_file**(filepath):
    - Input:
        - **filepath**: Path of the GLOD format file to read from.
    - Output:
        - List of observations in GLOD format.
- **read_srf_file**(filepath):
    - Input:
        - **filepath**: Path of the GLOD format file to read from.
    - Output:
        - The spectral response function that was contained in the file.

It exports the following classes and dataclasses:

- SRF: Dataclass representing a Spectral response function.
- GLOD: Dataclass representing an observation in something similar to GLOD format.

It will probably export more classes that the ones presented here that will appear in future sprints.

3.14 Spectral Integration

This module will perform the spectral integration (i.e. convolution) of the lunar irradiances (interpolated to ASD resolution) over the SRF. Internally, the matheo package from the NPL CoMet toolkit will be used to perform the integration.

It exports the following function:

- **convolve_srf**(srf, asd_irradiances):
    - Input:
        - **srf**: Spectral Response Function at which compute the simulation.
        - **asd_irradiances**: Lunar irradiances, interpolated to ASD resolution
    - Output:
        - Band-integrated lunar irradiance for the provided SRF

3.15 Spectral Interpolation

This module will perform the spectral interpolation of the lunar reflectances at CIMEL wavelengths (calculated from LIME coefficients) to the lunar reflectances at ASD wavelengths. ASD observations from WP1 serve as a high-resolution reference spectrum along which to interpolate. Internally, the matheo package from the NPL CoMet toolkit will be used to perform this interpolation along a high-resolution example.

It exports the following functions:

- **get_best_asd_reference**(selen_obs_lat, selen_obs_lon, selen_sun_lon, abs_moon_phase_angle):
    - Input:
        - **selen_obs_lat**: Selenographic latitude of the observer.
        - **selen_obs_lon**: Selenographic longitude of the observer.
        - **selen_sun_lon**: Selenographic longitude of the Sun.
        - **abs_moon_phase_angle**: Absolute moon phase angle for the observer.
    - Output:
        - ASD reference spectrum to be used for interpolation and associated uncertainties, relevant for provided geometry

- **get_interpolated_refl**(cimel_refl,asd_reference):
    - Input:
        - **cimel_refl:** lunar reflectances at the cimel wavelengths, calculated from the LIME coefficients
        - **asd_reference**: ASD reference spectrum to be used for interpolation
    - Output:
        - Lunar reflectances, interpolated to ASD resolution

3.16 Propagate Uncertainties

This module will be used within the various other modules to perform the uncertainty propagation. Typically, in the other modules, a measurement function will be defined to calculate the outputs from the inputs. These measurement functions will be python functions that take the input quantities as arguments and return the measurand. The uncertainty propagation itself will use the punpy package from the NPL CoMet toolkit.

It exports the following function:

- **get_uncertainties_standard**(measurement_func,input_qty,u_input_qty,corr_input_qty):
    - Input:
        - **measurement_func:** measurement function through which uncertainties should be propagated
        - **input_qty**: list of input quantities for each variable (values provided as floats or numpy arrays)
        - **u_input_qty**: list of uncertainties for each input quantities (values provided as floats or numpy arrays)
        - **corr_input_qty**: error-correlation for each input quantity (provided as "random", "systematic" or a custom correlation matrix)
    - Output:
        - Propagated uncertainties on the measurand (and associated correlation matrix if relevant)

Additionally, individual functions might be exported for the propagation through some of the measurement functions when manual calculation of the Jacobians can significantly speed up the uncertainty propagation. It might also export additional functions for storing and reading uncertainty information.

4 Implementation plan

The modules are intended to be sufficiently independent to allow the development in parallel for a subsequent integration. The workflow for the module development will be as follows:

- Leader to develop a detailed design for the module
- Discussion among all partners, including interfaces to other modules
- Module implementation in GitLab
- Test developments
- Testing of the module

The diagram with the implementation plan is given in table 2. The progress of the implementation will be revised by the partners on a monthly basis.

A password-protected GitLab repository will be used for the implementation, according to the accessibility requirements. The GitLab *issues* system will be used to track tasks until completion. Regular meetings between the complete development team will be held to monitor progress and identify the next tasks based on the design, and these will be added as GitLab issues.

To assure the quality of the code and used algorithms, different types of tests will be used. They are described in the SW verification plan (deliverable D8).

Concerning the documentation, a user manual will be produced. This user documentation will be hosted online as well as be delivered as a pdf. The GitLab continuous integration tools will be used for this purpose. The python code itself will be documented using `docstrings' and each partner will be responsible for the documentation of their own modules. A final document will be produced as deliverable D10.

All deliverables related to the TBX are listed as Gantt diagram of table 2. According to that, a first functional version of the TBX will be presented to ESA by end of 2022.

The implementation is divided in tasks, explained in Annex I, and the time management is described in the Gantt diagram of table 3.

Deliverables	Oct-21	Nov-21	Dec-21	Jan-22	Feb-22	Mar-22	Apr-22	May-22	Jun-22	Jul-22	Aug-22	Sep-22	Oct-22	Nov-22	Dec-22	Jan-23	Feb-23	Mar-23
Month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
D-6: LIME TBX requirement document 1		■																
D-7: LIME TBX technical specification document 1				■														
D-8 LIME TBX verification plan 1				■														
D-9: LIME TBX verification report 1															■			
D-10: LIME TBX user manual 1															■			
SW-3: LIME TBX															■			

Table 2. LIME TBX related deliverables.

Task	Hours	Planned Start Week	Planned End Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1.1.1	65	3	5			■	■	■																																			
1.1.2	17	5	5					■																																			
1.1.3	9	1	1	■																																							
1.1.4	9	1	1	■																																							
1.1.5	17	1	1	■																																							
1.1.6	41	1	2	■	■																																						
1.1.7	41	2	3		■	■																																					
1.1.8	17	6	6						■																																		
1.1.9	17	6	6						■																																		
1.1.10	25	1	1	■																																							
1.1.11	41	2	3		■	■																																					
1.1.12	41	3	5			■	■	■																																			
1.1.13	41	7	7							■																																	
1.1.14	25	8	8								■																																
1.1.15	9	8	8								■																																
1.1.16	25	8	9								■	■																															
1.1.17	25	9	9									■																															
1.2.1	41	10	11										■	■																													
1.2.2	25	11	11											■																													
1.2.3	17	10	10										■																														
1.2.4	25	12	12												■																												
1.2.5	9	12	12												■																												
1.2.6	1	12	12												■																												
1.2.7	25	13	13													■																											
1.2.8	25	13	13													■																											
2.1.1	25	14	14														■																										
2.1.2	17	14	14														■																										
2.1.3	9	15	15															■																									
2.1.4	17	15	15															■																									
2.1.5	25	15	16															■	■																								
2.2.1	17	16	16																■																								
2.2.2	9	16	16																■																								
2.2.3	9	16	16																■																								
2.2.4	17	17	17																	■																							
2.2.5	17	17	17																	■																							
3.1.1	41	17	18																	■	■																						
3.1.2	17	18	19																		■	■																					
3.1.3	9	19	19																			■																					
3.1.4	17	19	19																			■																					
3.1.5	25	19	20																			■	■																				
3.2.1	17	20	20																				■																				
3.2.2	9	20	20																				■																				
3.2.3	9	20	21																				■	■																			
3.2.4	17	21	21																					■																			
3.2.5	17	21	22																					■	■																		
4.1.1	17	15	16															■	■																								
4.1.2	17	14	14														■																										
4.1.3	17	14	14														■																										
4.1.4	17	14	15														■	■																									
4.1.5	17	15	15															■																									
4.2.1	9	16	18																■	■	■																						
4.2.2	9	17	17																	■																							
4.2.3	17	17	17																	■																							
4.2.4	17	17	18																	■	■																						
5.1.1	25	19	19																			■																					
5.1.2	17	19	19																			■																					
5.1.3	17	20	20																				■																				
5.1.4	25	20	20																				■																				
5.1.5	17	21	21																					■																			
5.1.6	25	21	21																					■																			
5.1.7	25	22	23																						■	■																	
5.2.1	5	23	23																							■																	
5.2.2	5	23	23																							■																	
5.2.3	25	23	24																							■	■																
5.2.4	17	24	24																								■																
5.2.5	17	24	24																								■																
6.1.1	17	25	25																									■															
6.1.2	41	26	27																										■	■													
6.1.3	9	25	25																									■															
6.1.4	25	25	26																									■	■														
6.1.5	17	26	27																										■	■													
6.1.6	25	27	27																											■													
6.2.1	17	28	28																												■												
6.2.2	9	28	28																												■												
6.2.3	17	28	28																												■												
6.3.1	17	29	29																													■											
6.3.2	17	29	29																													■											
6.3.3	17	29	30																													■	■										
6.3.4	17	30	30																														■										
6.4.1	9	31	31																															■									
6.4.2	25	31	31																															■									
6.4.3	9	31	32																															■	■								
6.4.4	5	32	32																																■								
6.4.5	17	32	32																																■								
6.4.6	9	32	33																																■	■							

Table 3. LIME TBX Gantt chart for implementation.

5 Annex I: Product Backlog

5.1 Epic stories

Number	Name	Description
1	Simulate from Earth	As a user I want to be able to simulate lunar observations for any observer position on Earth.
2	Simulate with custom selenographic coordinates	As a user I want to be able to simulate lunar observations for a custom observer and solar selenographic latitude and longitudes.
3	Simulate from a satellite position	As a user I want to be able to simulate lunar observations from a satellite position, for at least the satellites defined in **RNF104**.
4	Simulate for a spectral response	As a user I want to be able to simulate the lunar observations for a user defined instrument spectral response.
5	Compare from a remote sensing instrument	As a user I want to be able to perform comparisons of lunar observations from a remote sensing instrument to the LIME output.
6	Download updated coefficients	As a user I want my application to update itself so I always have the latest version and I don't have to take care of it myself.

5.2 User stories

1.1	Simulate single observation from Earth	
1.2	Simulate time series of observations from Earth	
2.1	Simulate single observation with custom selenographic coordinates	
2.2	Simulate time series of observations with custom selenographic coordinates	
3.1	Simulate single observation from a satellite position	
3.2	Simulate time series of observations from a satellite position	
4.1	Input a user-defined instrument spectral response SRF	
4.2	Visualize the user defined spectral response used	
5.1	Compare to a remote sensing instrument	
5.2	Export data	
6.1	Perform manual update	

6.2	Perform automatic update	
6.3	Select coefficients version	
6.4	Version management and metadata	

5.3 Tasks

S-1.1 Simulate single observation from Earth

1.1.1	GUI input for single observation from Earth	Create the GUI view that allows the direct input of the data and the option to choose an input file instead.
1.1.2	Input file to model parameters	Creating a function that reads the input file and extracts the needed parameters.
1.1.2-B	Execution from command line	Execute the story from command line without the use of the GUI.
1.1.3	Selenographic coordinates	Creating a function that calculates the selenographic coordinates of the location at the time
1.1.4	Selenographic longitude of the sun	Creating a function that calculates the selenographic longitude of the sun at the time
1.1.5	Moon phase angle	Creating a function that calculates the moon phase angle at the time for that location
1.1.6	Simulate lunar disk irradiance	Implement the main LIME algorithm that simulates the lunar disk irradiance or reflectance in the spectral range of 400nm to 2500nm
1.1.7	Simulate lunar disk degree of polarization	Create the module for the calculation of lunar disk degree of polarization in the spectral range of 400nm to 2500nm
1.1.8	Calculate lunar disk irradiance associated uncertainty	Create a function that calculates the uncertainty based on the original data uncertainty and the interpolation uncertainty for the lunar disk irradiance
1.1.9	Calculate lunar degree of polarization associated uncertainty	Create a function that calculates the uncertainty based on the original data uncertainty and the interpolation uncertainty for the lunar degree of polarization
1.1.10	Adapt NPL spectral integration libraries to the project	Adapt the existing NPL libraries and modules in order to create a new module that performs spectral integration.
1.1.11	Adapt NPL interpolation libraries to the project	Adapt the existing NPL libraries and modules in order to create a new module that interpolates.
1.1.12	Adapt NPL uncertainty libraries to the project	Adapt the existing NPL libraries and modules in order to create a new module that returns the associated uncertainty of the previous NPL modules.

1.1.13	GUI output for single observation from Earth	Create the GUI view that shows the calculated output to the user and lets them start the process of exporting it.
1.1.14	Export output to GLOD format files	Create a function that exports the calculated data to a file following the GLOD format.
1.1.15	Export uncertainty output to NetCDF format files	Create a function that exports the calculated data uncertainties to a NetCDF file.
1.1.16	Create user manual for user story	Creation of the user manual entry for this user story
1.1.17	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-1.2 Simulate time series of observations from Earth

1.2.1	GUI input for time series of observations from Earth	Create the GUI view that allows the selection of an input file.
1.2.2	Input file to model parameters of time series of observations	Expand the function created on **1.1.2** and let it read the input file and extract the needed parameters for multiple observations.
1.2.2-B	Execution from command line	Execute the story from command line without the use of the GUI.
1.2.3	Adapt to multiple observations	Create new functions that use the functions created in **1.1** so now it processes multiple observations.
1.2.4	GUI output for time series of observations from Earth	Create the GUI view that shows the calculated output to the user and lets them start the process of exporting it.
1.2.5	Export output to GLOD format files	Create a function that exports the calculated data to a file following the GLOD format, based on **1.1.14**.
1.2.6	Export uncertainty output to NetCDF format files	Create a function that exports the calculated data uncertainties to a NetCDF file, based on **1.1.15**.
1.2.7	Create user manual for user story	Creation of the user manual entry for this user story
1.2.8	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-2.1 Simulate single observation with custom selenographic coordinates

2.1.1	GUI input for single observation with custom selenographic coordinates	Create the GUI view that allows the direct input of the data and the option to choose an input file instead
2.1.2	Input file to model parameters of time series of observations	Expand the function created in **1.1.2** and let it read the input file and extract the needed parameters for selenographic coordinates.
2.1.2-B	Execution from command line	Execute the story from command line without the use of the GUI.

2.1.3	GUI output for single observation with custom selenographic coordinates	Create the GUI view that shows the calculated output to the user and lets them start the process of exporting it.
2.1.4	Create user manual for user story	Creation of the user manual entry for this user story
2.1.5	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-2.2 Simulate time series of observations with custom selenographic coordinates

2.2.1	GUI input for time series of observations with custom selenographic coordinates	Create the GUI view that allows the selection of an input file.
2.2.2	Input file to model parameters of time series of observations	Expand the function expanded on **2.1.2** and let it read the input file and extract the needed parameters for multiple observations for selenographic coordinates.
2.2.2-B	Execution from command line	Execute the story from command line without the use of the GUI.
2.2.3	GUI output for time series of observations with custom selenographic coordinates	Create the GUI view that shows the calculated output to the user and lets them start the process of exporting it.
2.2.4	Create user manual for user story	Creation of the user manual entry for this user story
2.2.5	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-3.1 Simulate single observation from a satellite position

3.1.1	GUI input for single observation from satellite position	Create the GUI view that allows the direct input of the data and the option to choose an input file instead, selecting a concrete ESA satellite for at least the satellites defined in **RNF104**.
3.1.2	Input file to model parameters of single observation from a satellite position	Expand the function created on **1.1.2** and let it read the input file and extract the needed parameters from a satellite position.
3.1.2-B	Execution from command line	Execute the story from command line without the use of the GUI.
3.1.3	GUI output for single observation from a satellite position	Create the GUI view that shows the calculated output to the user and lets them start the process of exporting it.
3.1.4	Create user manual for user story	Creation of the user manual entry for this user story
3.1.5	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-3.2 Simulate time series of observations from a satellite position

3.2.1	GUI input for time series of observations from satellite position	Create the GUI view that allows the selection of an input file while selecting a concrete ESA satellite for at least the satellites defined in **RNF104**.
3.2.2	Input file to model parameters of time series of observations	Expand the function expanded on **3.1.2** and let it read the input file and extract the needed parameters for multiple observations from a satellite position.
3.2.2-B	Execution from command line	Execute the story from command line without the use of the GUI.
3.2.3	GUI output for time series of observations from a satellite position	Create the GUI view that shows the calculated output to the user and lets them start the process of exporting it.
3.2.4	Create user manual for user story	Creation of the user manual entry for this user story
3.2.5	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-4.1 Input a user-defined instrument spectral response SRF

4.1.1	GUI input for instrument spectral response	Create the GUI view that allows the input of the user-defined instrument spectral response directly and the option to import it from a file.
4.1.2	Input file to model parameters of SRF	Create a function that reads the input file containing the SRF and transforms it to model data.
4.1.2-B	Execution from command line	Execute the story from command line without the use of the GUI.
4.1.3	Create user manual for user story	Creation of the user manual entry for this user story
4.1.4	End-to-end testing of user story	Creation of end-to-end tests for this user story
4.1.5	Calculate the band integrated model output	

S-4.2 Visualize the user defined spectral response used

4.2.1	Modify GUI so user can ask to be shown a plot of the user-defined spectral response	Visualize the user defined spectral response used for the spectral integration of the LIME output into a sensor spectral band.
4.2.2	Visualization of user-defined spectral response.	Create the GUI view that shows the plot of the user-defined spectral response.
4.2.3	Create user manual for user story	Creation of the user manual entry for this user story
4.2.4	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-5.1 Compare to a remote sensing instrument

5.1.1	GUI input for comparing from a remote sensing instrument	Create the GUI view that lets the user input the needed data and select the input file.

5.1.1-B	Execution from command line	Execute the story from command line without the use of the GUI.
5.1.2	Process the GLOD format file	Create the functions that read the GLOD format file and transform it to model data
5.1.3	Comparison of relative differences	Implement comparison functions that calculate the relative differences between measured and modeled lunar irradiance/reflectance vs. time and vs. lunar phase angle.
5.1.4	Comparison of statistical indicators	Implement functions that calculate statistical indicators (mean relative difference, standard deviation of the mean relative difference, temporal trend if applicable, number of comparison samples, etc…)
5.1.5	Comparison plots	Implement functions that display the elements calculated with tasks **5.1.3** and **5.1.4** in the GUI
5.1.6	Create user manual for user story	Creation of the user manual entry for this user story
5.1.7	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-5.2 Export data

5.2.1	GUI options for exporting plots	Add GUI elements that let the user export the plots as they wish
5.2.2	Export plot as jpg or pdf	Develop a function that exports the plot as a JPG file or PDF file
5.2.3	Export data as ASCII, GLOD, NetCDF	Develop a function that exports the data as ASCII, GLOD or NetCDF
5.2.4	Create user manual for user story	Creation of the user manual entry for this user story
5.2.5	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-6.1 Perform manual update

6.1.1	GUI options for updating	Add GUI elements that let the user ask for a manual update
6.1.2	Set up repository for storing the model coefficients	
6.1.3	Coefficient update check	Implement a function that performs an update check
6.1.4	Update coefficients	Download the updated coefficients and save them
6.1.5	Create user manual for user story	Creation of the user manual entry for this user story
6.1.6	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-6.2 Perform automatic update

6.2.1	Perform automatic check on start	Create a function that calls the check when the application is just started
6.2.2	Create user manual for user story	Creation of the user manual entry for this user story
6.2.3	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-6.3 Select coefficients version

6.3.1	GUI option for choosing coefficients version	Add the GUI elements that let the user choose a coefficient version.
6.3.2	Use the selected coefficients	Modify functions so now the coefficients used for the calculations are the selected ones
6.3.3	Create user manual for user story	Creation of the user manual entry for this user story
6.3.4	End-to-end testing of user story	Creation of end-to-end tests for this user story

S-6.4 Version management and metadata

6.4.1	List all version data and metadata that we need to keep track of	Create a list of all version data and metadata that we need to keep track of
6.4.2	Set up internal data objects that keep track of the information	Set up internal data objects that keep track of the information
6.4.3	Add version and metadata to exported data files	Add version and metadata to exported data files
6.4.4	Add version and metadata to exported plots	Add version and metadata to exported plots
6.4.5	Create user manual for user story	Creation of the user manual entry for this user story
6.4.6	End-to-end testing of user story	Creation of end-to-end tests for this user story