Deliverable #6

Software Design Document

	Name	Date
Prepared by	Benoît Thiébault, Project Manager	20/08/2012
Checked by	Benjamin Jeanty-Ruard, Software developer	21/08/2012
Approved by	Julien Forest, Expert Consultant	22/08/2012

Reference: ESA-SPISGEO-D6-SDD-2012-08-001	**Version**: 1	**Revision**: 0
Contract number: 4000101174	**Date**: 20/08/2012	

Table of contents

1. INTRODUCTION 4
1.1. Scope of the document 4
1.2. Reference and applicable documents 4
1.2.1. Applicable documents 4
1.2.2. Reference documents 4
1.3. Acronyms and abbreviations 4
2. SOFTWARE OVERVIEW 5
2.1. Background and context 5
2.2. System workflow: the modelling chain and its link to components 5
3. COMPONENTS DESIGN SPECIFICATIONS 9
3.1. Wizard module 9
3.2. Project module 9
3.3. Geometry editor module 10
3.4. Mesh editor module 10
3.5. Group editor module 11
3.6. Electrical circuit editor module 12
3.7. Global parameters module 13
3.8. UI to NUM module 13
3.9. Simulation module 14
3.10. SPIS-NUM 14
3.11. Data mining module 15
3.12. Reporting module 15
3.13. SPIS data model module 16
3.14. Gmsh module 16
3.15. Project converter module 17
3.16. External tools module 17
4. SOFTWARE LIFECYCLE MANAGEMENT 19
5. CONCLUSION 20

Summary

Objectives

The purpose of this document is to describe the.components implemented in SPIS-GEO and their dependencies.

Technical overview

SPIS-GEO application is built on top of Keridwen 2.0, an OSGi-based Integrated Modelling Environment to achieve a modular application, easier to maintain and to integrate with other tools.

A description of the components of this modular application is given as well as their dependencies.

Diffusion

Nom	Organisation
David Rodgers Alain Hilgers Fabrice Cipriani	ESA
Julien Forest Benoît Thiébault Benjamin Jeanty-Ruard Pierre Souquet	Artenum
Jean-Charles Matéo Vélez Jean-François Roussel Pierre Sarrailh	ONERA
Bjarne Andersson Alain Demairé	OHB-Sweden
Patrice Pelissou Marc Sevoz	EADS-Astrium

Changes record

Version	Revision	Date	Author / Modification
1	0	20/08/2012	Benoît Thiébault / Document creation

1. Introduction

1.1. Scope of the document

The purpose of this document is to define a collection of software components and their interfaces to establish a framework for developing SPIS-GEO.

1.2. Reference and applicable documents

1.2.1. Applicable documents

[AD1] Statement Of Work "Simplified MEO/GEO Tools for Spacecraft Charging", TEC-EES/2008.348/DR, issue 1.0, 30/09/2009.

[AD2] Technical and Administrative Proposal, ESA-SPISGEO-PTC-2009-12-001, 20/11/2009

[AD3] Software Requirements Document, ESA-SPISGEO-D3-SRD-2010-11-001.

[AD4] Guide to applying the ESA software engineering standards to small software projects, BSSC(96)2 Issue 1, 1996.

[AD5] Guide to the software architectural design phase, ESA PSS-05-04 Issue 1 Revision 1, March 1995

1.2.2. Reference documents

N/A

1.3. Acronyms and abbreviations

- **GUI**: Graphical User Interface
- **S/C**: Spacecraft
- **SPIS-CORE**: Current Spacecraft Plasma Interactions Software main development branch that is available on spis.org website (version 4.2).
- **SPIS-GEO**: Simplified Standard MEO/GEO Tools for Spacecraft Charging
- **TBD**: To Be Determined
- **UR**: User Requirement
- **WC**: Worst Case
- **W.R.T.:** with respect to

2. Software overview

2.1. Background and context

Initially released in March 2004, the SPIS software has become today the European standard for the modelling and the simulation of the spacecraft plasma interactions. Initially funded on an ESA effort and following an open-source approach in the frame of the SPINE community, SPIS knows today a real and dynamic community life. SPINE counts more than 250 registered members today, inside and outside EU. The average number of posts on the forums overtakes 70 messages per month.

Originally designed to focus on scientific applications, the application scope of SPIS is largely wider now and is regularly extended to new engineering applications or domains of physics. This includes, for instance, the modelling of electrical propulsion systems, ESD prediction on solar arrays or link with radiation models through deep charging phenomena.

The prediction of the electrostatic charge (absolute and relative) of spacecraft for engineering purposes is also a key issue with modern platforms that are more and more complex, operating high-power and sensitive electronic devices or using modern materials. Differential charging can lead to arcing, dangerous for the electronic payload. Absolute charging can induce disturbances on the radio transmission and/or the positioning systems using electric propulsion. This need is especially critical for GEO and MEO missions, where are located most of the commercial platforms. Moreover, the progressive generalisation of electrical propulsion systems on commercial platforms pushes integrators to perform much more detailed electrostatic analysis before the flight.

Legacy tools currently used in the industry, like NASCAP, cannot address these modern constraints. Thanks to its modularity and the implemented models representing the present state-of-the-art in plasma-surface interactions, SPIS is currently probably the best basis to address these issues in a self-consistent manner. There is a real need of a specific version of SPIS, called SPIS-GEO, to model MEO and GEO missions that would simplify its usage in an engineering context.

2.2. System workflow: the modelling chain and its link to components

In order to perform a simulation with SPIS, the user has to follow a given number of pre- and post-processing steps in the correct order. In SPIS-CORE, the number of the steps to perform the simulation is high but provides the possibility to configure the simulation with the maximum of details, which is the objective of a simulation software targeted to scientists.

In SPIS-GEO, this modelling chain has to be simplified so that engineers can perform the simulation without requiring advanced knowledge in plasma physics or the numerical models. The following steps are implemented to define the simplified modelling chain of SPIS-GEO:

- **Project and study creation or loading:** a new concept has been introduced in SPIS-GEO, the study. A SPIS-GEO project is composed of one or more studies. To a study are associated one CAD model, a set of group properties (e.g. material and physical characteristics allocated to user-defined spacecraft surfaces) and an electric circuit definition. If the user wishes to modify one of these elements, he can create a new study. This first modelling step allows the user to create new project and studies or to load existing ones. The component that implements this function is **org-spis-ui-project**.

- **Geometry edition/creation:** during this step, the user either provides an existing CAD file or creates a new one from the definition of the satellite geometry and defines the global refining coefficient to be applied when meshing the CAD. This function is implemented by component **org-spis-ui-geometry-editor**.

- **Mesh generation and inspection:** the mesh previously generated is displayed and the user can check the mesh statistics. He can also import existing meshes generated by Gmsh outside SPIS. This function is implemented by component **org-spis-ui-mesh**.
- **Local properties allocation:** the user sets the local properties to be applied (material properties, boundary conditions, electrical nodes, electric propulsion, etc.). The component **org-spis-ui-group-editor** is the one implementing this function.
- **Electrical circuit definition: org-spis-ui-electrical-circuit** component allows the user to define the relationship between the different electrical nodes.
- **Environment and numerical settings:** the user has to provide the environment conditions in which he/she wants to model the satellite/plasma interaction as well as numerical settings (such as simulation duration). This is provided by module **org-spis-ui-global-parameters**. If the user wants to run several simulations with different global parameters, he can create different "runs" that are then stored in SPIS project.
- **Simulation loop:** the simulation loop is the phase during which the numerical core of SPIS-GEO is executed to compute the spacecraft/plasma interactions. This is performed by **SPIS-NUM**.
- **Monitoring:** during the simulation loop, the user can visualize the progress of the computation as well as some key results (such as spacecraft potential as a function of time, collected currents as a function of time, etc.). This function is provided by module **org-spis-ui-simulation**.
- **Post-processing:** once the simulation has complete, the user can visualize the results and export them in various formats thanks to module **org-spis-ui-data-mining**.
- **Reporting:** a report generation module, **org-spis-ui-reporting**, offers the possibility to automatically generate a report of the simulation run.

In addition to the modules directly implementing the modelling steps, there are other support modules. The relation between those modules is illustrated Figure 1 below. The dotted arrow illustrate a "use" relationship.

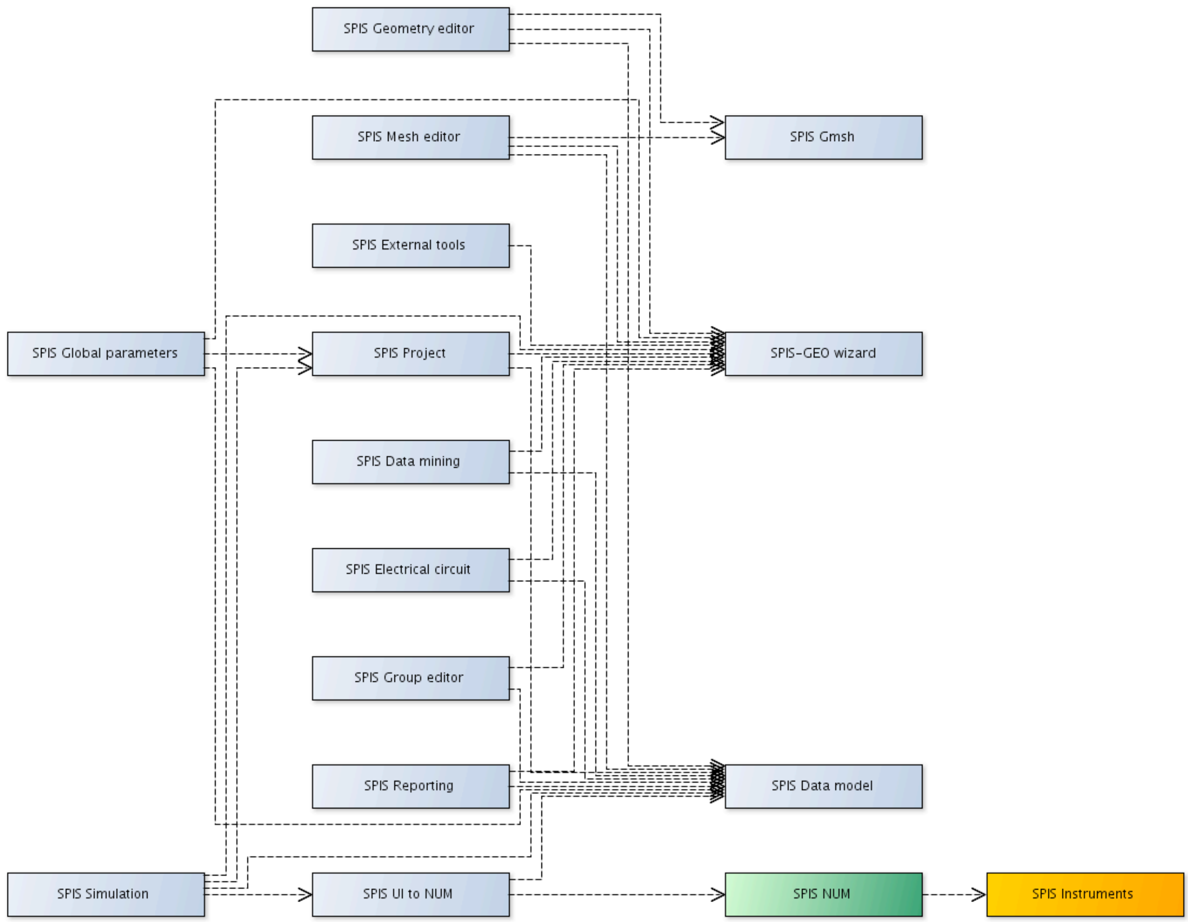

Figure 1: SPIS design simplified overview

SPIS modules also use external libraries such as:

- **Keridwen core**, that implements low-level functions (such as communication between modules, internationalization, user interface docking system)
- **Keridwen modelling**, that provides generic Integrated Modelling Environment functions (such as the global parameters editor, a simple text editor, etc.)
- **Cassandra**, for 3D data visualisation
- **Penelope**, for memory-efficient mesh storage and manipulation
- **Frida**, for group properties allocation
- **ArtTK**, that provides generic GUI widgets

The relationship between SPIS components and external libraries components is detailed on Figure 2.

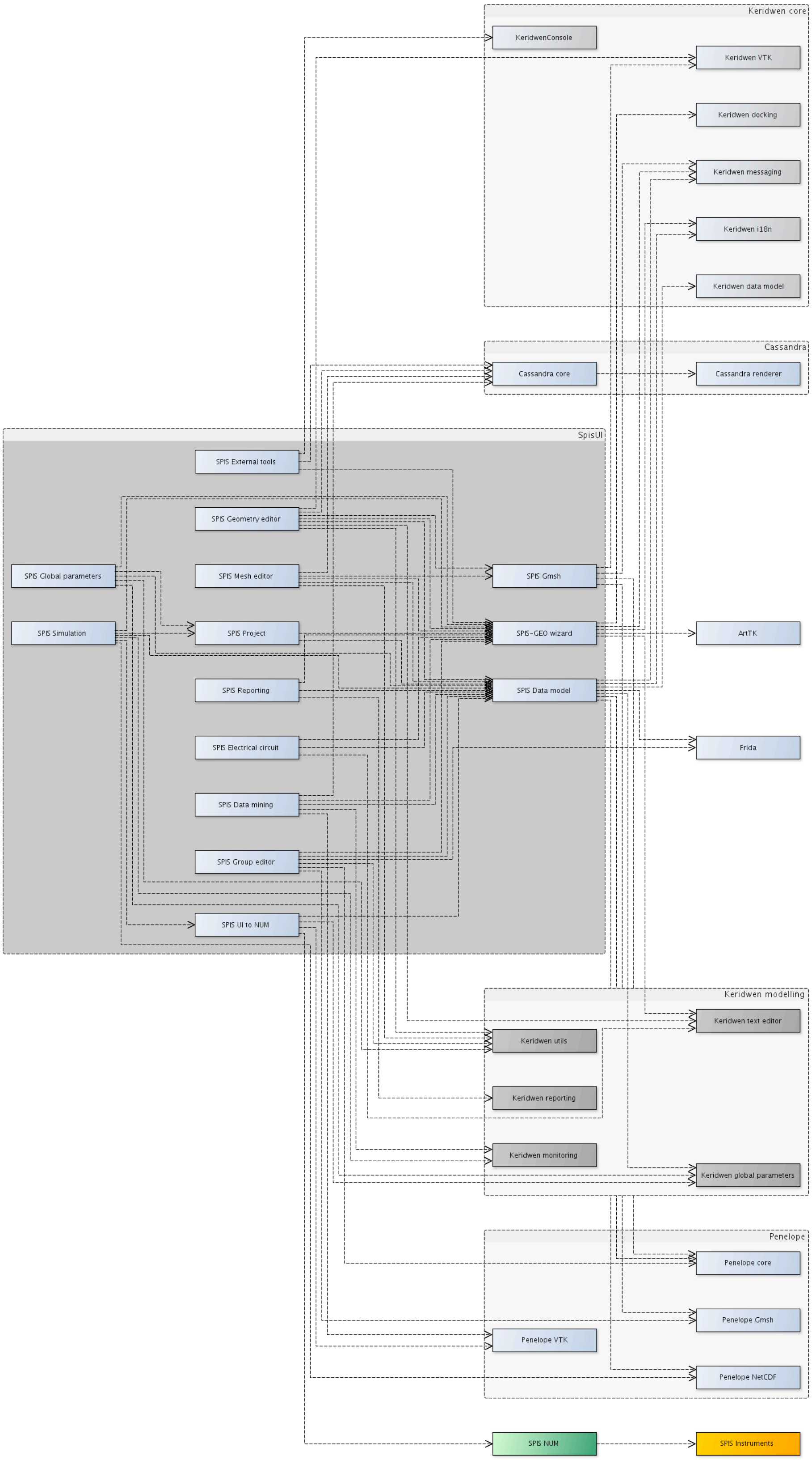

Figure 2: SPIS components interactions with external libraries

3. Components design specifications

3.1. Wizard module

Id	org-spis-ui-geo-wizard
Type	Support module
Purpose	Provide to other modules the possibility to dynamically register their GUI in SPIS main user interface and docking system. This module also provides graphical components (icons or dialogs for instance) that are reused by modelling chain modules.
Dependencies	Internal: • None External • ArtTK • Keridwen core messaging • Keridwen docking system • Keridwen internationalisation • Keridwen text editor
Interfaces	• Allows modules to register/unregister • Changes the currently displayed “perspective”. A perspective is the graphical user interface of a modelling step component • Provides various graphical components
Processing	N/A
Data	N/A

3.2. Project module

Id	org-spis-ui-project
Type	Modelling step module
Purpose	This module provides the project and study creation and loading functions. It also allows saving the project.
Dependencies	Internal: • Wizard module • SPIS Data model module External • Keridwen modelling utils
Interfaces	• Saves the project
Processing	N/A
Data	• Works with SPIS 5 projects only (read/write)

3.3. Geometry editor module

Id	org-spis-ui-geometry-editor
Type	Modelling step module
Purpose	This module offers the possibility to import CAD files, to edit them or to create new ones. It also generates the mesh with the given refinement coefficient.
Dependencies	Internal: • Wizard module • SPIS Data model module • Gmsh module External • Keridwen modelling utils • Keridwen VTK module • Cassandra core • Cassandra renderer
Interfaces	N/A
Processing	Creates a mesh from the given CAD files
Data	Input: • CAD files (Gmsh format) Output: • Mesh file (Gmsh .msh 2.2 format)

3.4. Mesh editor module

Id	org-spis-ui-mesh
Type	Modelling step module
Purpose	This module allows the user to inspect the generated mesh and its statistics. The user can also import mesh files generated by Gmsh outside SPIS.
Dependencies	Internal: • Wizard module • SPIS Data model module • Gmsh module External • Keridwen modelling utils • Keridwen VTK module • Cassandra core • Cassandra renderer
Interfaces	N/A
Processing	N/A
Data	Input: • Mesh files (Gmsh .msh 2.2 format) Output: • Mesh object (Penelope format)

3.5. Group editor module

Id	org-spis-ui-group-editor
Type	Modelling step module
Purpose	This module provides the user interface to load and edit properties catalogues and to allocate those properties to CAD groups.
Dependencies	Internal: • Wizard module • SPIS Data model module External • Keridwen modelling utils • Penelope core • Penelope Gmsh • Frida
Interfaces	N/A
Processing	In SPIS-GEO, the notion of properties corresponds to the definition of local parameters, like boundary conditions, used in the definition the model. This includes material, physical and numerical properties. The purpose of the pre-processing part is to attribute these properties to each specific parts of the spacecraft or the computational domain, convert them into fields (i.e. DataFields) mapped on the mesh that the numerical model can understand. In order to properly define the needed initial and boundary conditions, the various properties should be attributed to their respective Physicals in order to build properties groups. This is done using the Group Editor. The Frida library offers a very versatile and adaptable model to attribute all types of data or properties to groups. Groups are gathered into a central groups list. A Group can contain several Properties. Each Property can contain several Characteristics. A Characteristic can be a scalar (integer, float, double), a vector or all other types of data (object). A Property can have a sub-property, that itself contains its own characteristics. Such an approach allows to build-up or enrich properties by composition of existing ones. The same Characteristic can have different values when it is linked to different Properties, in order to set different values for the corresponding field.

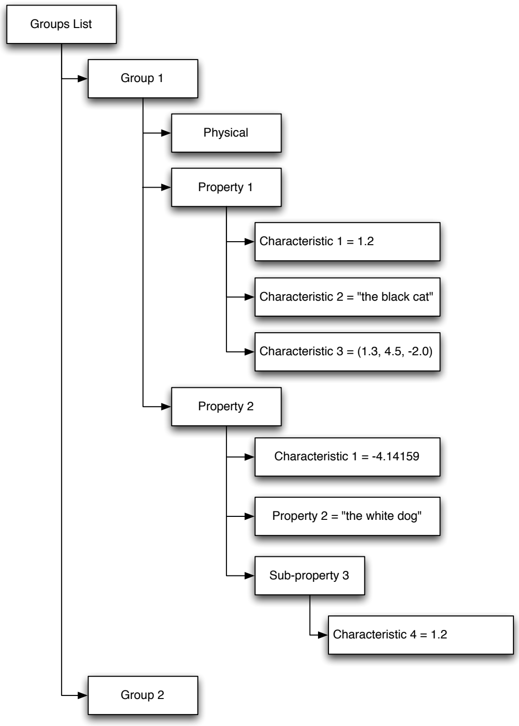

Figure 3: Groups, Properties and Characteristics structures in Frida library.

	The Groups Editor provides a facility to help selecting appropriate property types for each given type of group, with a mechanism of group operators making the link between each group type and the corresponding supported properties. Properties and Characteristics can be edited and modified dynamically. Frida includes a Properties/Characteristics mapping function that deploys the values of Characteristics on the Mesh and generates the corresponding DataFields. Characteristics are deployed according the given group priority. All information (groups, properties and characteristics) is saved under the form of NetCDF files.
Data	Input: • Mesh files (Gmsh .msh 2.2 format) or object (Penelope format) Output: • Datafields with mapped properties (Penelope object format or NetCDF file format)

3.6. Electrical circuit editor module

Id	org-spis-ui-electrical-circuit
Type	Modelling step module
Purpose	This module provides a text editor to create the electrical circuit file determining the electrical relationship between electrical nodes.
Dependencies	Internal: • Wizard module • SPIS Data model module External • None

Interfaces	N/A
Processing	N/A
Data	Output: • Circuit.txt ASCII text file

3.7. Global parameters module

Id	org-spis-ui-global-parameters
Type	Modelling step module
Purpose	This module allows the user to set the global parameters of the simulation. The user can also import and export the global parameters and add/delete existing global parameters.
Dependencies	Internal: • Wizard module • Project module • SPIS Data model module • UI to NUM module External • Keridwen global parameters • Keridwen modelling utils
Interfaces	N/A
Processing	Once the global parameters have been set, this module converts the data from UI format to NUM format.
Data	Input : • Global parameters XML files Output: • Global parameters XML files of Keridwen object

3.8. UI to NUM module

Id	org-spis-ui-ui2num
Type	Support module
Purpose	This modules converts the data from UI format to NUM format
Dependencies	Internal: • SPIS-NUM • SPIS Instruments • SPIS Data model module External • Keridwen core messaging • Keridwen global parameters • Penelope XML plugin • Penelope VTK plugin
Interfaces	• Converts UI data to NUM data

Processing	• Retrieving the datafields and identifying the different meshes to build for SPIS-NUM • Mapping between ids from Penelope mesh and ids from SPIS Num meshes. • Creation of SPIS-NUM mesh object for the spacecraft from the datafields. • Creation of SPIS-NUM mesh object for the external boundary from the datafields. • Creation of SPIS-NUM mesh object for the computational volume from the datafields. • Conversion of the global parameters from Keridwen module to global parameters defined in SPIS Num • Creation of local parameters defined in SPIS-NUM from datafields defined on Penelope Mesh. • Creation of NumTopFromUi from local parameters, global parameters, SPIS-NUM meshes and paths used by SPIS Num.
Data	Input: • Mesh object in Penelope format • Datafields list created by the group editor module • Global parameters created by the global parameters module • Directory where are stored files used by SPIS Num • Directory where will be stored output created by SPIS Num Output: • NumTopFromUi, which is the object used by SPIS-Num to launch the simulation

3.9. Simulation module

Id	org-spis-ui-simulation
Type	Modelling step module
Purpose	The purpose of this module is to execute, monitor and control the numerical kernel
Dependencies	Internal: • Wizard module • Project module • SPIS-NUM • SPIS Data model module External • Keridwen monitoring • Penelope NetCDF module
Interfaces	N/A
Processing	N/A
Data	Input: • Results of the UI to NUM module Output: • Simulation results (Datafields written in NetCDF files)

3.10. SPIS-NUM

Id	org-spis-num

Type	Simulation kernel
Purpose	The numerical kernel performs the simulation
Dependencies	Internal: • SPIS Instruments External • None
Interfaces	• Initialize the simulation • Adding/removing instruments for monitoring • Launch and pause the simulation • Retrieve simulation results
Processing	Cf. SPIS-NUM user manual
Data	Input: • NumTopFromUi, as created by UI 2 NUM Output: • Simulation results

3.11. Data mining module

Id	org-spis-ui-data-mining
Type	Modelling step module
Purpose	The purpose of the data mining module is to extract the data produced by the simulation kernel and to visualise it. It can also export the data in different formats
Dependencies	Internal: • Wizard module • Project module • SPIS Data model module External • Cassandra core • Cassandra renderer
Interfaces	N/A
Processing	N/A
Data	Input: • Simulation results in the form of Datafields NetCDF files Output: • CSV export for 2D data • VTK export for 3D data

3.12. Reporting module

Id	org-spis-ui-reporting
Type	Modelling step module

Purpose	
Dependencies	Internal: • Wizard module • SPIS Data model module External • Keridwen reporting module • Keridwen text editor
Interfaces	N/A
Processing	N/A
Data	Input: • Preprocessing data • Simulation data • Postprocessing data Output: • OpenOffice report

3.13. SPIS data model module

Id	org-spis-ui-model
Type	Support module
Purpose	This module is used to store in memory all the data managed by SPIS. It also contains the various reader/writer called when the project is saved.
Dependencies	Internal: • None External • Keridwen core data model • Keridwen core messaging
Interfaces	• Adding/delete nodes in the data tree • Saving and loading the projects • State management of the whole application
Processing	
Data	

3.14. Gmsh module

Id	org-spis-ui-gmsh
Type	Support module
Purpose	This module provides methods to control Gmsh from other modules
Dependencies	Internal: • None External

	• Penelope core • Keridwen Gmsh • Keridwen VTK • Keridwen core messaging • Gmsh
Interfaces	• Mesh .geo file in 1, 2 or 3D • Generate VTK unstructured grid from Penelope mesh or .msh file
Processing	N/A
Data	Input: • Gmsh CAD file (.geo) • Gmsh .msh 2.2 formatted file Output: • Gmsh .msh 2.2 formatted file • Penelope Mesh object • VTK unstructured grid object

3.15. Project converter module

Id	org-spis-ui-project-converter
Type	Support module
Purpose	This modules provides a function to convert a legacy project (from SPIS-CORE) to SPIS 5 project
Dependencies	Internal: • Wizard module • SPIS Data model module External • Penelope JFreeMesh module
Interfaces	• Converts a legacy project to a SPIS 5 project
Processing	N/A
Data	Input: • Legacy SPIS project with global parameters saved in globalParameters.xml file at the root of the project Output: • SPIS 5 project

3.16. External tools module

Id	org-spis-ui-tools
Type	Support module
Purpose	This module is used
Dependencies	Internal: • Wizard module

	External • Keridwen core messaging • Keridwen VTK module • Keridwen console • Cassandra core • Cassandra rendering
Interfaces	• Launch Cassandra • Launch Gmsh • Launch JyConsole/JRosetta
Processing	N/A
Data	N/A

4. Software lifecycle management

As all SPIS releases, SPIS-GEO is hosted on the SPINE community platform at the following address:

http://www.spis.org

A registration as SPINE member may be needed before downloading the software. Registration demands are available at the following address:

http://dev.spis.org/projects/spine/home/captcha

The core elements of SPIS are open-source software (see the Licenses directory or file). However some additional components might be diffused under closed-source and/or proprietary license and/or have diffusion restriction, please check the licenses terms before any use and if necessary, please contact the ESA contact officer or the SPIS SDAB contact person at the following address:

contact@spis.org

The dependencies of SPIS modules are automatically managed by Maven [http://maven.apache.org]. The maven repository used for SPIS is located at [http://maven.artenum.com].

SPIS is also developed in the frame of a continuous integration chain. This means that every night, automatically, SPIS is compiled, its tests are run and quality checks are performed. This allows the developers to be warned early when bugs and regressions are introduced. The continuous integration server is available here [http://hudson.artenum.com] and the quality control server is here [http://sonar.artenum.com].

5. Conclusion

SPIS-GEO application is proposed to be built on top of Keridwen 2.0, an OSGi-based Integrated Modelling Environment to achieve a modular application, easier to maintain and to integrate with other tools.

A description of each of the components implemented for this modular application has been given.