SPIS Other improvments currently under developments

J. Forest(1), B. Thiebault (1), J.-F. Roussel (2), J.-C. Mateo Velez (2)

SPINE meeting Toulouse

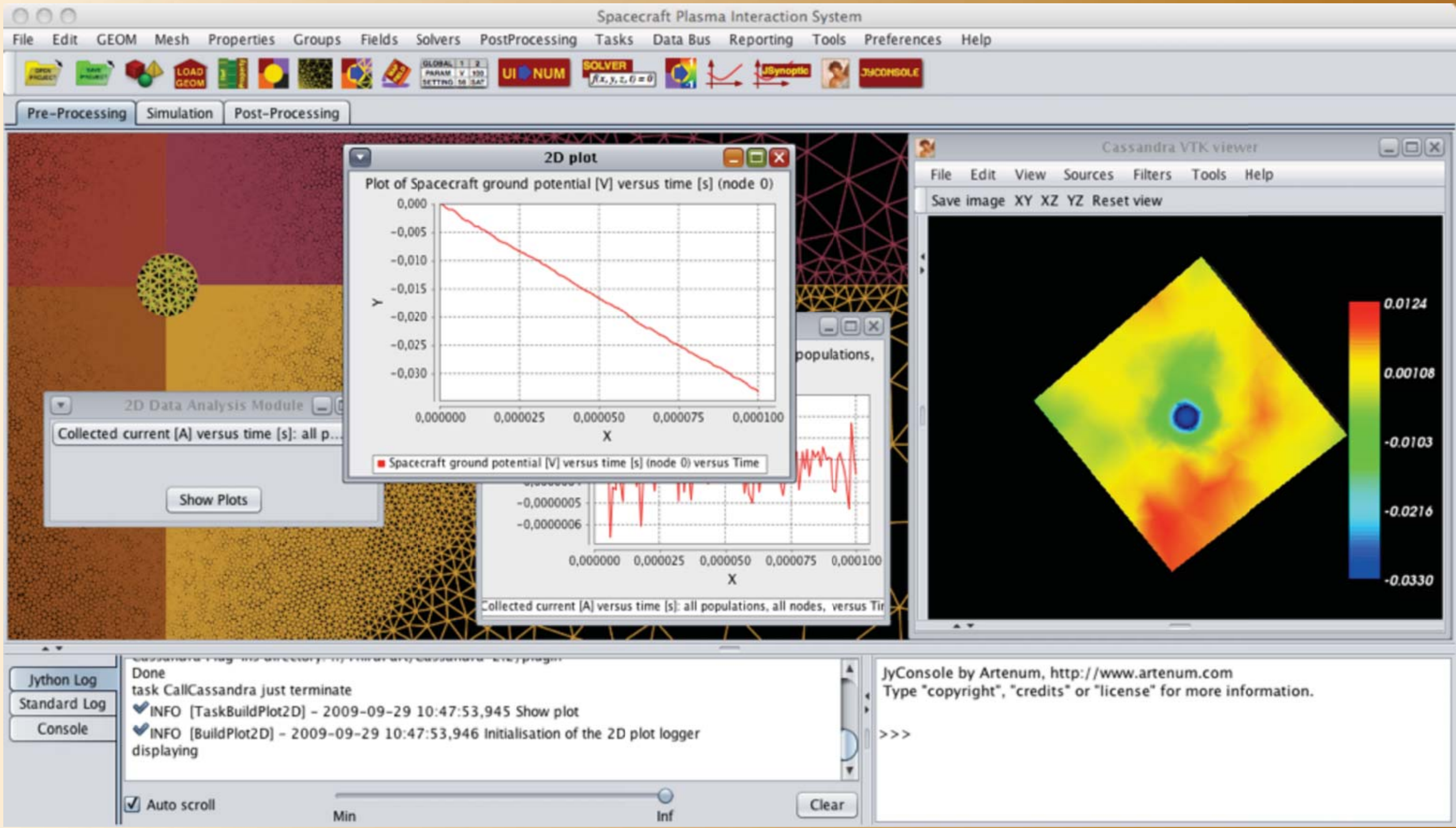
28 Septembre 2009

(1) Artenum, (2) ONERA

Shared effort

- Most of these developments are funded by ESA CNN in the frame of th SPIS Time Dependent evolution
- Also external contributions
  - CETP
  - Artenum's own effort
    - Improvment of JFreeMesh
    - Some elements coming from the Keridwen IME (e.g. Kerwizard...)
    - Cassandra/Cassandra-PCS

Improved / experimental aspect



Underdevelopment efforts

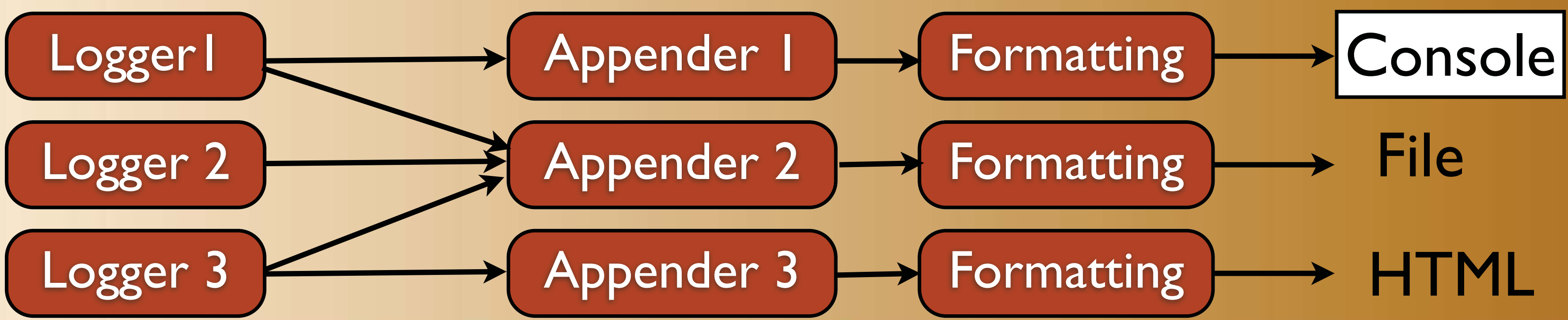
- **Error treatment**
  - Improved error and stack trace by the integration of a generic logger
  - Improved “user level/explicit” error messages
  - Development of a “Mesh Inspector” to analyse the mesh consistency.
  - Develop test plan and perform testing
- **Material parameter input:**
  - Re-factoring of data structure for material data in UI
  - Development of an In/Out module to read external files (e.g. NASCAP-2K)
  - Re-factoring of the interface with NUM and internal refactoring in NUM
  - Update/extension of material parameters list
  - Develop test plan and perform testing
- **Physics improvements**
  - Implement emission of multiple species from a single emitter
  - Implement reflection of particles at boundaries
  - Implement neutral particles
  - Develop test plan and perform testing
- **Simplify the UI for tailored application**
  - Wizard based approaches
  - Dedicated tools

Why an improved logging system ?

- Objectives
  - Introduce a normalised logging system, in order to:
    - Improve the awareness of the user in case of errors and warnings
    - Facilitate the debugging and the users'feedbacks collection
      - Detailed and standardised logging files to facilitate the diagnostic
  - Evolutive, in order to extend the level of information for each error in function on the level of expertise accumulated by the community
  - Offer several levels of verbosity
  - Dynamically configurable
  - Use more standard technics, compliant with the several level of languages used in SPIS (Jython, Java, native...)

Logging system principle and techno: Log4J and Sfl4J

- Use open standards
  - **Abstract logging API: SFL4J**
    - Generic and standardised API upper the really implemented logging API
    - Possibility to change the logging lib without modification of the source code
    - Used by several major projects
    - Improved performances and additional features (e.g profiling...)
  - **Logging and appending concrete implementations: Apache Log4J**
    - The reference !
    - Offer several levels of verbosity (debug, info, warning, error...)
    - Offer a large set of outputs possibilities (console, files, html, etc..)
    - Offer an extensible framework
  - Both compliant with Jython an Java
  - The whole dynamically configurable



A simple implementation

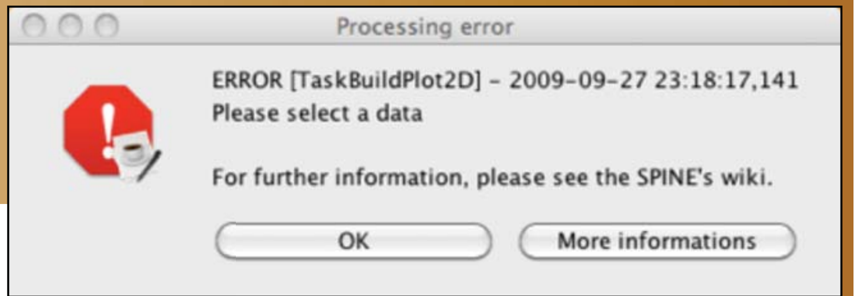
- Create loggers in the relevant classes or group of classes

```
# building of the related logger
self.logger = LoggerFactory.getLogger("TaskLoadProj")
self.logger.info("ProjectLoader2 initialised")
```

- Replace «print» or «System.out.println» by a logging action

```
if "projectInfo" in self.keysList:
    self.logger.info("Loading project's informations")
    try:
        load_dict(sharedProjectInfos, project_file)
        self.logger.info("Done")
    except:                #print >> sys.stderr, "No project file. Pr
        self.logger.warn("No project file. Probably invalid directory.")
```

- Configure the link with the appenders (see next slide)
- Use on of the three man appender/outputs defined
  - Error dialog box with link to the SPINE platform
  - Improved logging console
  - Log file (Saved in the project directory !)




```
INFO  [TaskLoadProj] - 2009-09-27 23:15:01,479 Loading groups
DEBUG [TaskLoadProj] - 2009-09-27 23:15:01,485 loading the new format
DEBUG [TaskLoadProj] - 2009-09-27 23:15:01,487 loading groupsPriorityList
DEBUG [TaskLoadProj] - 2009-09-27 23:15:01,508 loading geoGroup56.py
DEBUG [TaskLoadProj] - 2009-09-27 23:15:01,531 loading geoGroup57.py
WARN  [TaskLoadProj] - 2009-09-27 23:15:01,555 Warning in Material property re-connection
ERROR [TaskBuildPlot2D] - 2009-09-27 23:18:17,141 Please select a data
```

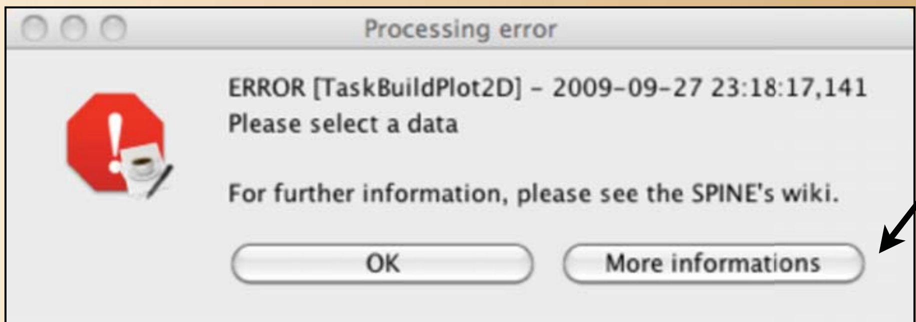
Dynamical configuration

- Links between loggers and appenders can be defined at the runtime through an XML based file (SPIS_ROOT/SpisUI/AuxLib/GUI/ressources/log4j.xml)
- Re-read at each restart of the framework. Does not need any re-compilation.
- Can define:
  - The output format (data, time, source class or logger...)
  - The link logger to appender (i.e select the output directions)
  - The level of verbosity

```
<logger name="TaskLoadProj" additivity="true">
    <priority value ="warn" />
    <appender-ref ref="pythonLog" />
    <appender-ref ref="dialogBox"/>
    <appender-ref ref="file" />
</logger>
```

Problematic of definition of a usefull message

- Most of the error cases in SPIS are due to wrong configurations in amount phases.
- The reasons can be very variable and are generally not very well identified or, at least, not in an systematic manner and is an evolutive situation that:
  - Depends on the studied physics and modelled systems
  - Depends on modelling process
  - Depends on the «functional module» between the seat and the keyboard
  - Cannot address all the cases
- Difficult to give a unique, relevant and general answer (including for the «experts»)
- To solve this problem, it is proposed to
  - Use the community
  - Build-up a system that can be progressively improved
- Error messages generate a error dialog box, where a Web link is available toward a dedicated forum on the SPINE platform, the *SPINE's Online Help*.



Open the default browser to the SPINE Online Help

http://dev.spis.org/projects/spine/home/spis/software/onlinehelp/

Mesh inspector

- Sub-routine directly integrated into JFreeMesh
  - Better integration
  - Better performances
  - Currently implemented sub-routines / tests
    - Face/nodes relative position test
    - Cell/Faces neighbouring test
    - Quality evaluation (volume/height ratio)
    - Barycentre...
- Dedicated GUI still under development and various approaches studied
  - As standalone module in order simple and «automatic» basic tests
  - As plug-ing in Cassandra in order to
    - Visualise and identify the corrupted mesh elements
    - Need to develop the corresponding VTK structure generator

NASCAP material reader/writer

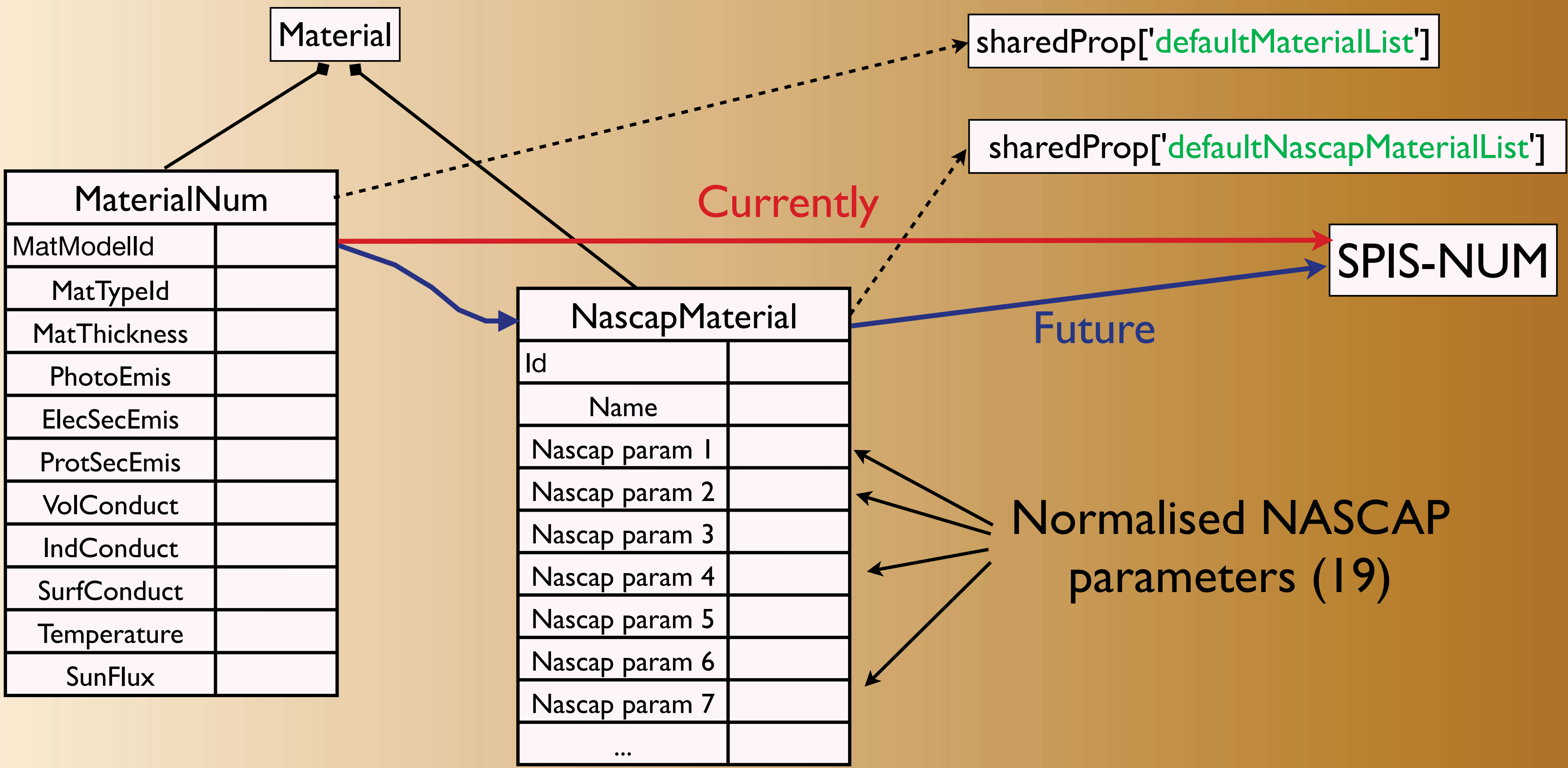
- Development an import/exporter for NASCAP based format based on SAX
  - Developed in Java and based on SAX
  - Integrated into org.spis.imp.io.nascap (i.e lib of main SPIS GUI)
    - NascapXMLReader
    - NascapXMLWriter
- Modification of NascapMaterial as *NascapMaterialFactory* (Modules.Properties)
- Development of a *NascapMaterialImporter* module (Bin) in Jython + a basic GUI (under development)
- Extension will probably lead to a format extension/re-definition
- This is of SPINE normalisation effort


```xml
<Assembly>
    <MaterialProperties Name="Aluminum" Color="16776960">
        <Property Index="0" Value="1.0" />
        <Property Index="1" Value="0.0010" />
        <Property Index="2" Value="-1.0" />
        <Property Index="3" Value="13.0" />
        <Property Index="4" Value="0.97" />
        <Property Index="5" Value="0.3" />
        <Property Index="6" Value="154.0" />
        <Property Index="7" Value="0.8" />
        <Property Index="8" Value="220.0" />
        <Property Index="9" Value="1.76" />
        <Property Index="10" Value="0.244" />
        <Property Index="11" Value="230.0" />
        <Property Index="12" Value="4.0E-5" />
        <Property Index="13" Value="-1.0" />
        <Property Index="14" Value="26.98" />
        <Property Index="15" Value="2699.0" />
        <Property Index="16" Value="17.0" />
        <Property Index="17" Value="18.0" />
        <Property Index="18" Value="1.0E-16" />
        <Property Index="19" Value="20.0" />
    </MaterialProperties>
```


0: ITOC (Material coated with ITO)
1: CERS (Solar cell material. Cerium doped silicon with MgF2 coating)
2: CFRP (Carbon fibre, conducting, no resin layer)
3: KAPT (Kapton, average values for SEE...)
4: COSR (Optical solar reflector without MgF2 coating. Cerium doped glass type)
5: EPOX (Epoxy. Thin layer of Epoxy resin on (conducting) Carbon fibre)
6: BLKP (Non conductive black paint. SEE yields are as measured for Electrodag 501)
7: BLKH (Non conductive black paint HERBERTS 1002-E. Values updated 3.10.88.)
8: BLKC (Conductive black paint Electrodag 501)
9: PCBZ (White paint PCB-Z assumed to be conductive in space)
10: PSG1 (White paint PSG 120 FD assumed to be conductive in space.)
11: TEFL (Teflon, DERTS measurements of SEE)
12: CONT (Generic Dielectric after 5 years in GEO environment.)
13: GOLD
14: SILV (Silver as from NASCAP library)
15: ALOX (Oxydized Aluminium. SEE yields from DERTS for Aluminium/Kapton)
16: STEE (Steel, SEE sigma +Emax from DERTS, curve shape from CONT material)
17: AL2K (Aluminium according to NASCAP-2k)
18: AU2K (Gold according to NASACP-2k)
19: KA2K (Kapton according to NASACP-2k))
20: TE2K (Teflon according to NASACP-2k)
21: OSR2K (OSR according to NASACP-2k)
22: BK2K (Black Kapton according to NASACP-2k)
23: SC2K (Solar Cells according to NASACP-2k)
24: NP2K (Non-conductive paint according to NASACP-2k)
25: GP2K (Graphite according to NASACP-2k)

Data structure: current status and evolutions

- Current structure based on a «composition» of *Materials*.
- SPIS-UI and SPIS-Num structures close to each other.
- But still Jython based and not a clean object oriented structure

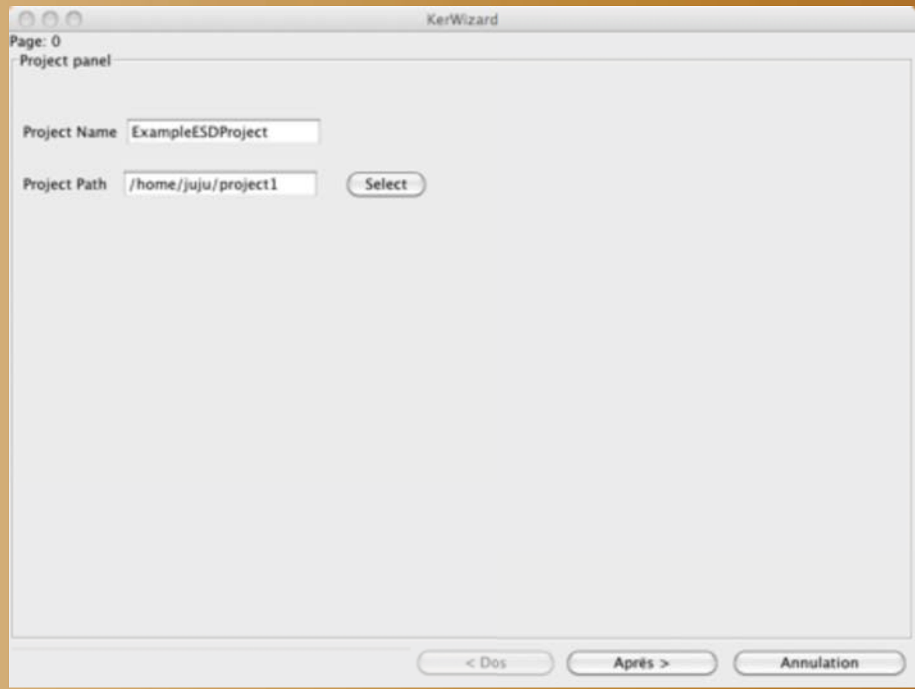


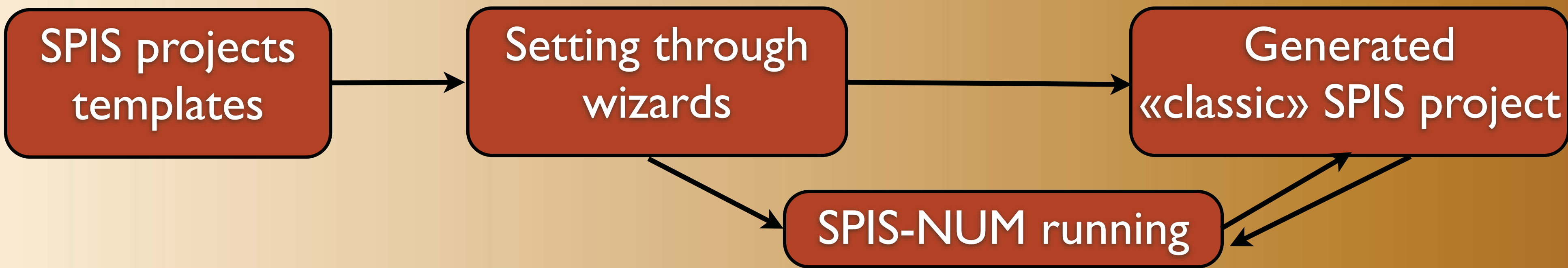
Physics improvments

- Work mainly on SPIS-NUM side
- Tasks to be done:
  - Implementation of multi-species on single emitter
  - Reflecting for particles
  - Implementation of neutral particles
  - Validation and testing

Simplified interfaces for taylored processes

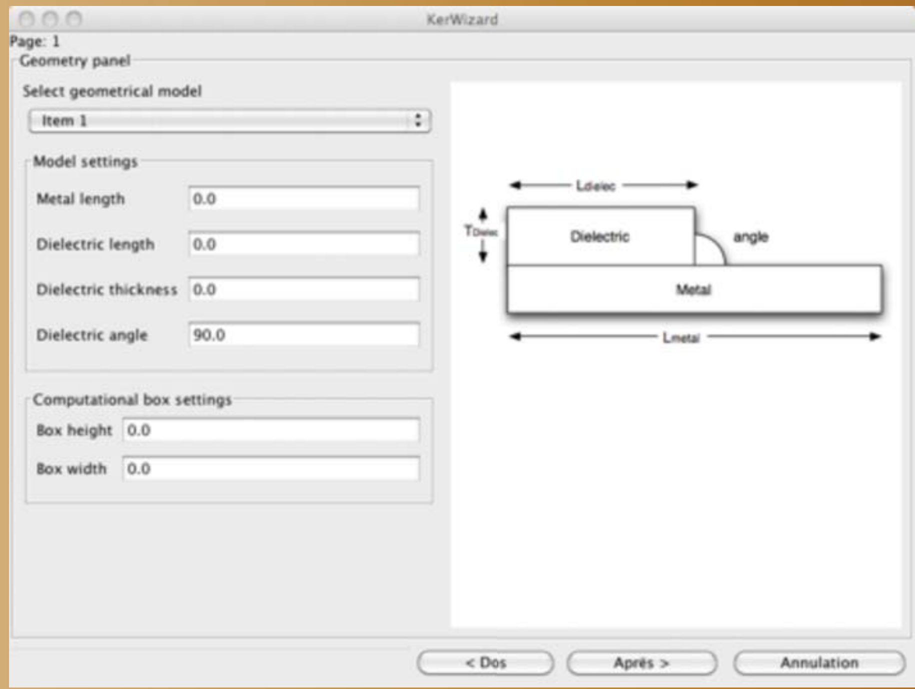
- Introduction of a wizard based approach
  - Start from «template project»
  - Let the access to only «key parameters»
  - Guide the user step-by-step
  - Generate a «fully standard» SPIS project, usable as usual.
- Based on a triple layer architecture
  - A generic wizard engine (KerWizard), runnable as standalone application or SPIS's task.
  - A set of tailored panels, that pilots SPIS-UI
  - SPIS-UI as «piloted model»

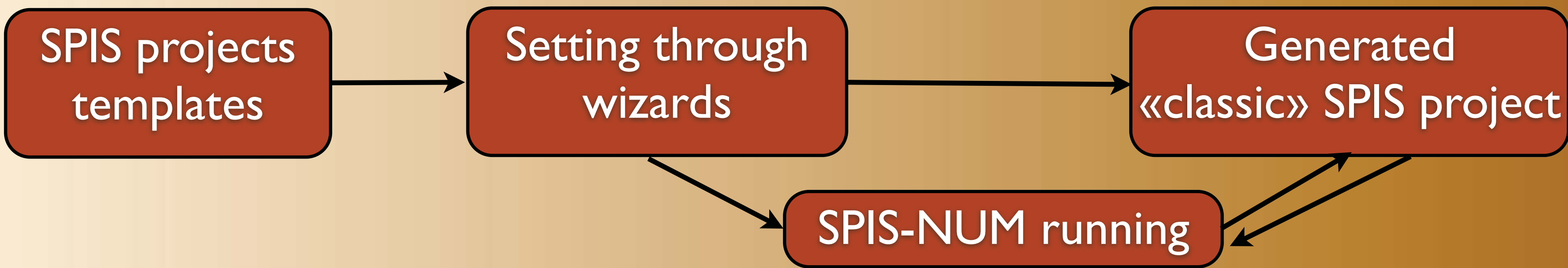




Simplified interfaces for taylored processes

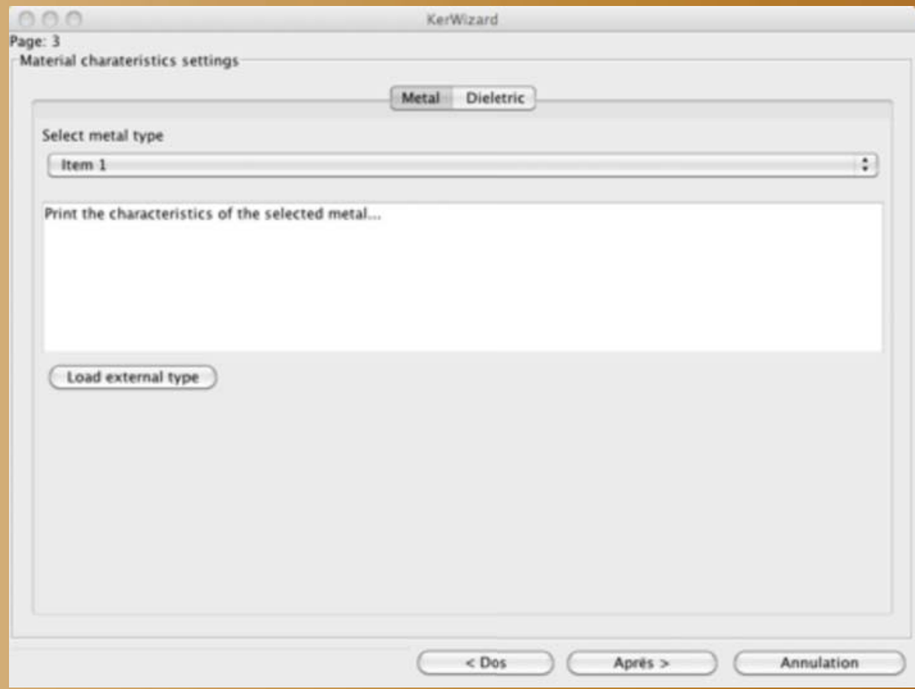
- Introduction of a wizard based approach
  - Start from «template project»
  - Let the access to only «key parameters»
  - Guide the user step-by-step
  - Generate a «fully standard» SPIS project, usable as usual.
- Based on a triple layer architecture
  - A generic wizard engine (KerWizard), runnable as standalone application or SPIS's task.
  - A set of tailored panels, that pilots SPIS-UI
  - SPIS-UI as «piloted model»

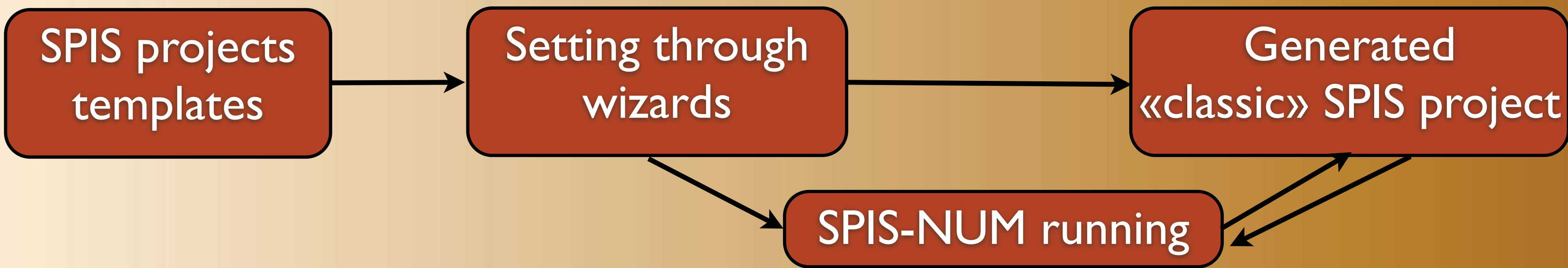




Simplified interfaces for taylored processes

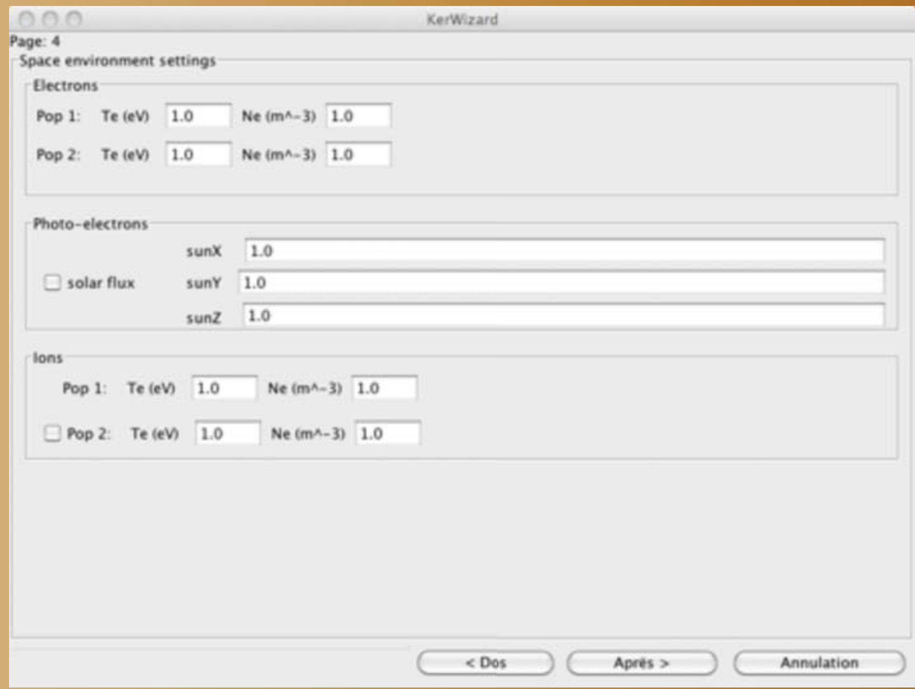
- Introduction of a wizard based approach
  - Start from «template project»
  - Let the access to only «key parameters»
  - Guide the user step-by-step
  - Generate a «fully standard» SPIS project, usable as usual.
- Based on a triple layer architecture
  - A generic wizard engine (KerWizard), runnable as standalone application or SPIS's task.
  - A set of tailored panels, that pilots SPIS-UI
  - SPIS-UI as «piloted model»

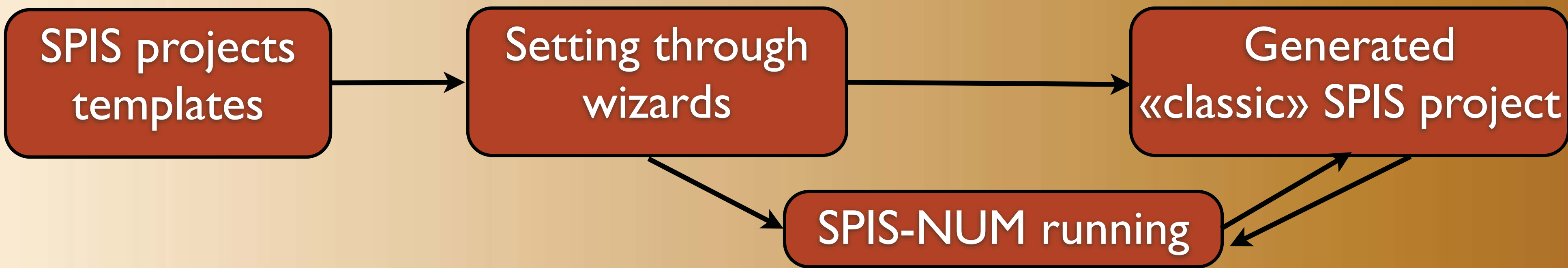




Simplified interfaces for taylored processes

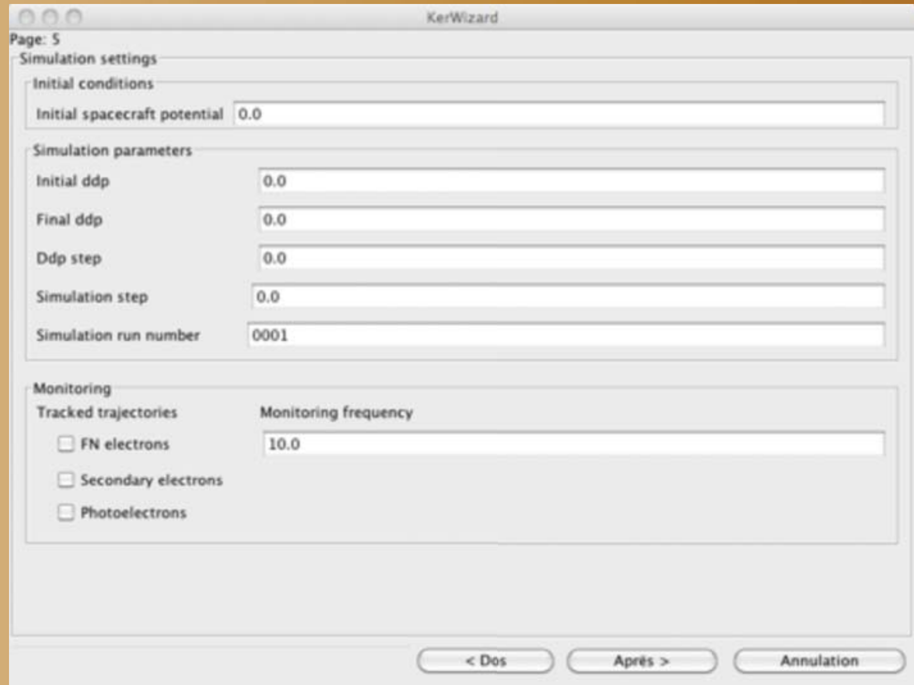
- Introduction of a wizard based approach
  - Start from «template project»
  - Let the access to only «key parameters»
  - Guide the user step-by-step
  - Generate a «fully standard» SPIS project, usable as usual.
- Based on a triple layer architecture
  - A generic wizard engine (KerWizard), runnable as standalone application or SPIS's task.
  - A set of tailored panels, that pilots SPIS-UI
  - SPIS-UI as «piloted model»

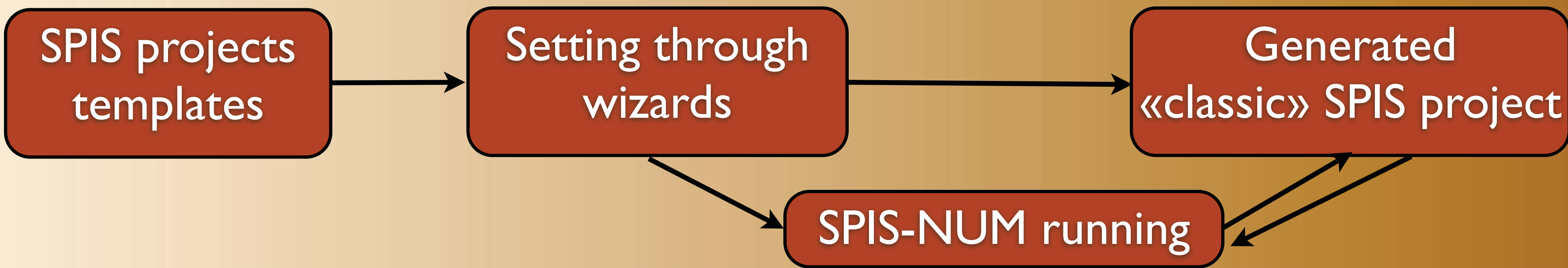




Simplified interfaces for taylored processes

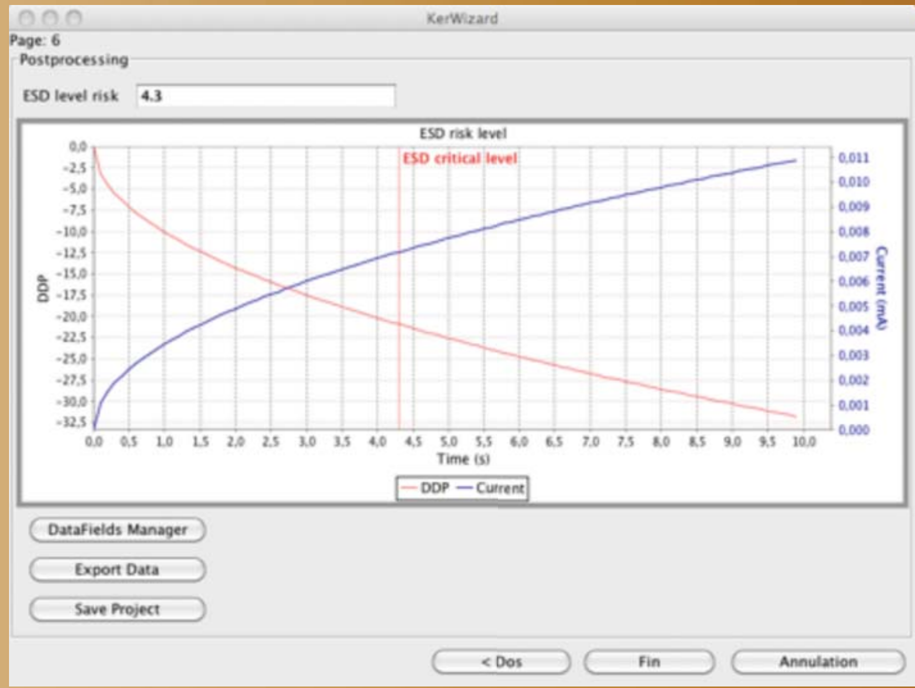
- Introduction of a wizard based approach
  - Start from «template project»
  - Let the access to only «key parameters»
  - Guide the user step-by-step
  - Generate a «fully standard» SPIS project, usable as usual.
- Based on a triple layer architecture
  - A generic wizard engine (KerWizard), runnable as standalone application or SPIS's task.
  - A set of tailored panels, that pilots SPIS-UI
  - SPIS-UI as «piloted model»

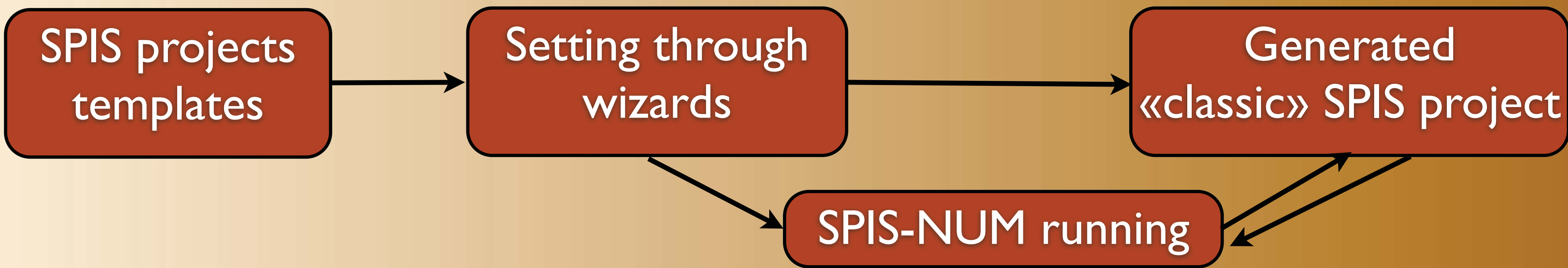




Simplified interfaces for taylored processes

- Introduction of a wizard based approach
  - Start from «template project»
  - Let the access to only «key parameters»
  - Guide the user step-by-step
  - Generate a «fully standard» SPIS project, usable as usual.
- Based on a triple layer architecture
  - A generic wizard engine (KerWizard), runnable as standalone application or SPIS's task.
  - A set of tailored panels, that pilots SPIS-UI
  - SPIS-UI as «piloted model»





Conclusion

- The work must go on.