Apollo

User features

**confidential**

Revision number: 1

Date:

Name: P. van der Velde

User features:

Workflow:

* Allow the user to indicate what to solve on each domain. From there the system will have to define the experiment setup. The user should probably define:
  + Materials
  + Physics model
    - Upgrade paths
    - Multi-fidelity
  + Links between regions (although we should be smart here)

With these selections the system should be smart enough to make the appropriate choices. If there are multiple options then make a selection but allow the user to change it (maybe also need to indicate that there are multiple options).

* Define templates for standard experiments?
* Note that users are mainly interested in the data, not the way it is generated.
* An actual thought pattern is roughly like:
  + Define goals --> If possible this should define the post-processing details (which variables to get etc.)
  + Select physics type to solve
  + Determine accuracy and requirements --> Defines solver
  + Define geometry --> modelling accuracy depends on required accuracy etc.
  + Define boundary conditions
  + (optional) define mesh --> should really be part of the solve step
  + Run simulation
  + Get results
* The workflow will largely be context based, i.e. the UI should present options based on the current context, e.g. only present material models in the appropriate moments.

Scenarios:

* Support different scenarios and provide easy setup for each of these scenarios
* Support:
  + What-if / Concept exploration
  + Optimization / Robust design
  + Analysis of design
  + Research

Goals:

* Allow users to specify the desired error bounds on the goal.

Experiments:

* The user should only have to specify the physical properties of the experiment. The system should then be able to suggest which models / boundary conditions are applicable. Obviously the user should be able to amend these suggestions.
* Users should only have to specify the geometry of the problem, not the additional boundaries. The only problem will lie in the differentiation of internal / external & boundaries of the geometry.

Parameters:

* Parameters should always have units
* Parameters editors should be able to deal with different aliases of the parameter and internally do the calculation. The UI should however always show the entered value.
* Allow resetting a parameter to its default value
* Allow the user to specify the tolerances / error bounds on the parameter values

File I/O:

* The user should always be able to get their data out of the system in an appropriate format.

Post-processing:

* The post-processors should allow comparing different cases (experiments) even with geometry changes.
* The post-processors should allow feature tracking and remapping of these features back to the geometry / mesh.
* The post-processors should be available at all times, even when running an experiment
* The post processors should allow multiple visualizations to be active at the same time.
* Allow the following visualizations:
  + Profile (2D graph) on 3D geometry
  + Force vectors in 2D on 3D geometry
  + Calculation of 2D force vectors

User interface:

* Provide easy access to the most current operations. Provide toolbars for the setting of the physical data.
* Allow marking of geometry elements and regions with capabilities. These can then be translated by the system into numerical data. E.g. an element is rigid, a curve is a wall or a symmetry line.

Project explorer:

* The project explorer is used to browse through existing projects / experiments and create new ones. The project explorer cannot be used to visualize any of the data in the project / experiment but it can be used to create new child-experiments etc.
* The project explorer should allow storing a script of the changes to be made so that the changes can be repeated easily over and over again.