Apollo

User features

**confidential**

Revision number: 1

Date:

Name: P. van der Velde

# Goal

The goal for Apollo is to provide engineers and designers with the capability to evaluate and change the behaviour of geometry based designs / objects in different physical situations. Apollo achieves this goal by allowing users to perform different experiments on a design. These experiments of one or more physical models can be virtual (i.e. simulations) experiments which Apollo executes or real experiments for which Apollo only stores and processes the data.

# Features

For Apollo a series of user features are suggested. These user features are described below. Each feature belongs to a certain area.

1. **User interface**
   1. The user interface should provide the user with easy access to the most important operations. Which operations are most important depends on the current context that the application is in. The user interface controls will be designed to provide a look and feel that is consistent with the hosting application. Depending on the hosting application, menu's toolbars, commands and context menus should be provided.
   2. Apollo will provide two different User Interface types, one UI which will integrate with CAD software and one stand-alone UI for project management.
   3. When Apollo is integrated into a CAD program then it should be easy for the user to mark geometry elements and regions with the desired characteristics, whether they are physical or numerical.
   4. The stand-alone UI will provide the user with the ability to browse existing projects and create new ones. Unlike the CAD plug-in the stand-alone UI will not be able to visualize any geometric or experiment data. It will however be able to provide the user access to the project (meta-)data
   5. Both UIs will be able to generate scripts which can be used to create new projects and/or make changes to existing projects.
2. Projects
   1. The user will be able to define a project which is a unit that maintains information about the relations between one or more experiments.
   2. DESCRIBE WHAT A USER CAN DO WITH PROJECTS
3. Experiments
   1. DESCRIBE WHAT AN EXPERIMENT IS
   2. DESCRIBE THE TYPE OF DATA EXPERIMENTS CAN HANDLE
   3. The user only needs to specify the physical and geometric properties for an experiment. Based on the physical and geometric properties Apollo can select or suggest the appropriate processing data / algorithms / methods.
   4. The user only needs to specify the geometry of the object under study. Apollo should be able to determine if additional boundaries need to be created and will automatically do so if required.
4. **Workflow**
   1. The user can easily indicate what the properties, both physical and chemical, are for each domain.
   2. The user can easily specify which 'variables' are of interest
   3. The user can easily specify what the desired error bounds are on the final results. Apollo will ensure that all calculations fall within these error bounds.
   4. The user can easily specify why an experiment is being run. Possible suggestions for running an experiment are:
      1. What-if; also known as concept exploration
      2. Optimization or robust design
      3. Scenario research

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 (Or should the system do this by itself?)
  + 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.

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