Apollo

System features

**confidential**

Revision number: 1

Date:

Name: P. van der Velde

System features:

Experiments:

* An experiment forms the base for all work done. An experiment consists of:
  + Data
  + Generator(s)
  + Visualization
* Experiments can be linked in a Directed Acyclic Graph (DAG).
* Experiments can by copied by the user
* Experiments can be created by the user
* Experiments can create sub-experiments of their own in order to generate data.
* Create sub-experiments when:
  + Large amounts of data are involved
  + The data must remain available for later processing
  + Changing unique (singleton) data, e.g. geometry, which influences the entire system
* Sub-experiments can be hidden from the user, but the user should be able to view them if so desired
* Experiments can request data from sub-experiments
* Experiment can be run directly by the user or when another experiment asks for it. When system controlled experiments can be run
  + As batch
  + Directly
  + In a postponed fashion
* When dealing with long running sub-experiments it is possible to put the parent experiment in ‘sleep’ mode. The parent experiment is woken up when all the data is available
* Sub-experiments should be locked against user changes because they are controlled by other experiments. Users have nothing to do with these. This also allows users to copy an experiment and its sub-experiments
* Experiments can be large so we should only copy the data if there are changes, i.e. copy-on-write.
* Experiments can contain a mixture of:
  + Real experimental data
  + Virtual experimental data, i.e. simulations
  + Theoretical experimental data
* Experi

Generators

* Generators should be able to add system wide commands to the command set
* Generator commands are:
  + Experimental generator
  + Simulation generator
    - Run simulation
    - Stop simulation
    - Pause simulation
  + Theoretical generator
* Generators themselves are stored in plugins
* Generators should be extendible by plug-ins

Simulations

* Allow multiple schedules to exist. These allow running of actions separately from the main schedule
* Only have one main schedule. This is the schedule that is executed when the simulation is run
* Allow running of sub-sets of any schedule
* Only allow creating sub-schedules if:
  + The resulting data can be stored in the originating simulation

Components

* Allow links between components:
  + Coworkers
  + Usage / Encapsulate, e.g. inside a data set where one data object encapsulates an other
  + Construction links, e.g. one element relies on another for construction (not necessarily as co-worker), example: mesh – geometry

Data

* Allow data components to be invalidated when generating data is changed