Apollo

User specification

**confidential**

Revision number: 1

Date:

Name: P. van der Velde

# Disclaimer

This specification is not, by any stretch of the imagination, complete. It will need to be revised several times before it is complete. Currently several major parts are either missing or incomplete. This disclaimer will be updated to reflect any change in these sections. Finally a specification is supposed to be a ‘living’ document and therefore never complete. What were you thinking; better learn to live with this fact.

# Goal

The goal of 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.

Figure 1 shows a mind map of the goals for the Apollo project.

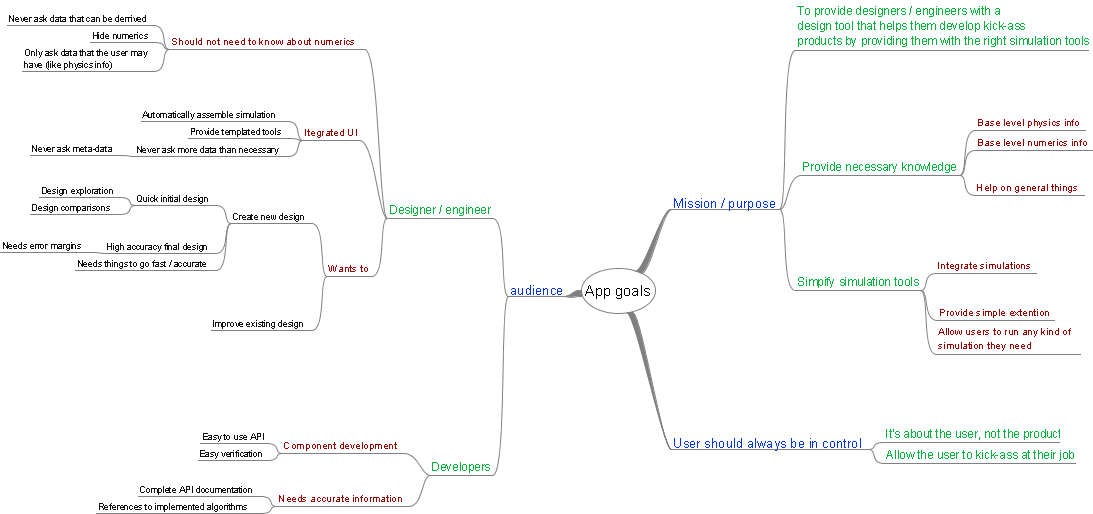


Figure : Mind map of the goals of Apollo

# Introduction

Apollo will support engineers, designers, architects and other professionals who need to evaluate and improve geometry based designs by providing them with a tool that allows them to study the physical behaviour of the design. Apollo will achieve this goal by providing the user with the ability to perform experiments based on one or more physical models on the design. These experiments can be virtual (i.e. simulations) experiments which Apollo can execute or real experiments for which Apollo only stores and processes the data.

During the design process the user may need to investigate different kinds of physical behaviour, ranging from simple cases where only a single physical model plays a role to more complicated cases where multiple physical models interact. Apollo will provide the tools to perform simple and complex experiments as well as providing as many different physical models as possible. However if a physical model is not known to Apollo it should be possible for the user (or others) to add tools that allow Apollo to work with the desired physical model.

Because experiments, both virtual and real, can quickly become complicated one of the main goals of Apollo is to simplify that part of the design process by providing easy setup and execution of virtual experiments and easy processing of data for virtual and real experiments. The idea is to make simple things automatic and complicated things easy. The user interface should be as simple as possible (but no simpler) but still provide access to more powerful features for the complicated scenarios.

# Users

Apollo is aimed at two different groups of users. The first group consists of engineers, architects, designers and other design related professionals who will use Apollo to evaluate the behaviour of different designs. The second group consists of software developers who will develop new tools for Apollo.

## The designer

The design related professionals will use Apollo to create new designs, possibly based on existing designs. For this task they require that Apollo provides them with tools that support their design process. Apollo should allow users to create several concepts quickly and use an iterative design process to arrive at a final design. This means that Apollo should provide the capability to perform design explorations and comparisons between different concepts. Apollo should allow highly accurate evaluations of the final concepts, by providing tools that allow optimization of the final design while providing certainty levels during each step.

While Apollo provides users with powerful tools it should never the less take a backseat to the user’s final goal. Apollo should never get in the way of the design or analysis work. This means that Apollo should perform as many of the menial tasks automatically and without asking the user for any more information than strictly required. The user should only need to provide data that is rooted in the physical domain of the experiment. From there Apollo should be able to determine all the other required data.

Apollo should ensure that users who only have knowledge of the physical aspects of the experiment can still perform the experiments successfully. This means that Apollo should hide as much of the calculation side of the experiments as possible but still provide advanced users with the possibility to control these parts.

## The developer

Apollo will provide an API (Application Programming Interface) for developers who wish to provide additional tools for Apollo. The API will provide developers with the capability to extend Apollo in a few key areas. The extensibility points of Apollo should allow developers to create additional tooling that provides new experiment sub-elements, data processing and visualization, data import & export filters and additional experiment types.

In order to support development of new tools for Apollo an easy to use API should be provided. This API should guide developers to the best way to develop tools for Apollo, i.e. the API should make the developer ‘fall into the pit of succes’.

# What we’re not doing

While Apollo will be a powerful system there are a number of things it will, by design, not be capable of. Note that this list will probably grow longer and longer as time goes by. The idea for Apollo is to keep it concentrated on a specific area so that it may be good at everything in that specific area.

* Apollo will not define a geometry manipulation API. Geometry manipulation should be done by a Computer Aided Design (CAD) application. To provide geometry storage capability Apollo may be running inside the CAD application as plug-in or side-by-side with the CAD application.
* Apollo will not provide physics modelling for every possible situation. The current interest lies in the fields of heat, structures and fluids. At later stages additional physical models may be added or removed.
* Apollo will not support multiple concurrent users working on the same model. All projects and experiments will only be accessible by a single user at the time.
* Apollo will not allow users to make changes to a running experiment. In order to change or update an experiment the user will have to stop the running of the experiment and make the changes. Apollo may allow users to pause an experiment in order to make changes.

# Use cases

The following use cases will describe the way Apollo can be used by designers. Currently four different use cases have been prepared. These will describe the high level workflow experienced by the designer as they work with Apollo. Parts of the workflow will be described in more detail in later sections.

The different use cases provided below describe:

* Robust design. A workflow that allows the designer to design a product which is able to perform optimally while being tolerant to changes in geometry or environmental conditions.
* What-if research or design. A workflow that allows the designer to investigate the response of a design to changes in geometry or environmental conditions.
* Optimization of existing or new products. A workflow that allows designers to optimize an existing or new design. Note that this workflow is essentially a more simplified version of the robust design workflow.
* General research. A workflow that allows designers to perform general research into the response of a design. Note that this workflow is a more general version of the what-if workflow.

While the workflows are described as separate cases it should be possible for the designer to switch between the different workflows at any point during the design process. The use of workflows while allowing easy switching between workflows will be achieved by providing support for the different workflow scenario’s through suggestions in the user interface. No information will be stored about the actual workflow being followed by the designer.

## Robust design

In robust design the designer is interested in developing a product which is able to perform optimally within a specific range of geometric and environmental conditions.

* Design with tolerances (optimal solution with tolerance to change). User is effectively only interested in the final result. However there could be some interest in learning why this solution is the best so we may need to store the data.
* How to approach? Which actions will the user take to start a robust design?
  + Need:
    - Tolerances for all the input variables
    - Allowed variance of variables (same as optimisation)
  + Getting started
  + Outputs
* Iterations in design changes (or is that what-if, regardless we should support this)
* How to specify the tolerances and how to review the results?
* User might want to ‘butt in’ on the iteration process and control one or more parameters manually, fix their ranges, or even hold them constant.
* Should be useable with all different physics models / solvers and all combinations of these models.
* The final solution should be reached quickly and the results should be accurate. More importantly the accuracy of the solution should be known (i.e. the user should know what the tolerances are on the solution).

## What-if research

* Put a design through different parameter studies to see what the differences are. User is normally only interested in the result, i.e. the influences of the different parameters. Possibly this could lead to an optimisation calculation (leaving out the unimportant parameters).
* How is this approached? What actions will be taken, what data is expected and should this data be stored?
* The user may need / want to specify specific variables to keep track of. Allow addition of new variables with their own calculations.
* How will the parameters be specified? How will the parameter range be specified?
* How will the results be presented, graphs, tables, 3d surfaces etc.
* Do users want to compare their results with others? Experiments, simulations?

## Optimization

* Define a design (probably geometry) and optimize its shape or behaviour.
* Which steps will be taken by the user?
* How will the user specify the different optimization variables? It should be really easy. We might also have to indicate a total computation time, so that the user will know how long the total will take (that way the user will probably not define too many variables / steps).
* How will the results be presented? Do we provide an overview of the optimization history and if so what will that overview look like (graphs, tables, movies …).
* Would users want to keep the data generated during the optimization?
* What can an optimization be based on. Simulations only or experimental data as well?
* As with the what-if and robust design tasks the user will probably only care about the final results but they’ll probably want to know that the results are solid (i.e. error margins etc.)

## Research

* General reviewing of a specific model in a specific situation OR several models in a specific situation OR a specific model in multiple situations etc. Need to keep all data and make reviewing of the results easy.
* More than likely that the user will want good control over the exact way the data set is processed.
* How will the model be specified?
* How will the situational settings (physics model etc.) be specified? Should be really easy to set and change
* Need error margins for sure. Probably also need additional error data like convergence checking etc.
* Allow for easy comparison of models / situations. Mapping one to another?

General

* Users are generally not interested in performing detailed error analyses but it could be useful to have these (robust design relies on this, and other formats could use the errors to determine the inaccuracies + error bars etc.). So we’ll have to offer the user an easy way out.
* How are we going to integrate the different scenarios into the system? Apollo itself probably shouldn’t know about these things but the UI should. Should we provide templates? Or …?

Cool stuff?

* Comparison of different simulations or simulations & experiments

# Data sets, what and how

* What is a data set? Data sets can be trouble if we start doing kinematics. People don’t associate kinematics with numerical calculations and thus not with data sets etc.
* What can they be used for
* What can they do
* The virtual data sets can be based on one or more physical models.
* Data from a data set can be compared with data from other data sets based on the same geometry. Apollo should be able to highlight the differences and indicate where these differences were found and which property / parameter was responsible for the differences.
* When users modify the original geometry Apollo should be able to update the virtual data sets and show the differences in results with the original geometry.
* Users can provide Apollo with physical and geometric situations in which they are interested. Apollo will then be able to provide the user with the behaviour of the geometry in these situations.
* Users can provide Apollo with additional expressions for which they wish to see results. Apollo should be able to process these additional expressions and provide the required data to the user.
* Users should be able to request that Apollo modifies the geometric parameters based on certain criteria (robust design, optimization) to achieve optimal performance in certain physical situations.

# Working with Apollo

Show steps, no need to tell a story

## Creating a new project

In the project explorer

In CAD

## Loading an existing project

In the project explorer

In Cad

## Working with a project

## Working with a data set

So what can we do with this???

* Define geometry
  + Define geometric parameters
  + Define geometry elements (walls, symmetry etc.) (??)
  + Define materials
* Indicate physics types
  + Define physics parameters
* Indicate experiment type (virtual, real)
  + Define data set parameters
  + For real experiments
    - Define data import methods (??) or data location
    - Define data processing methods
  + For virtual data sets

Should we do use cases too? If yes then check the “user interface design.txt” file which has some basic use cases. Obviously we’ll have tons of them. Maybe put them in their own document?