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.

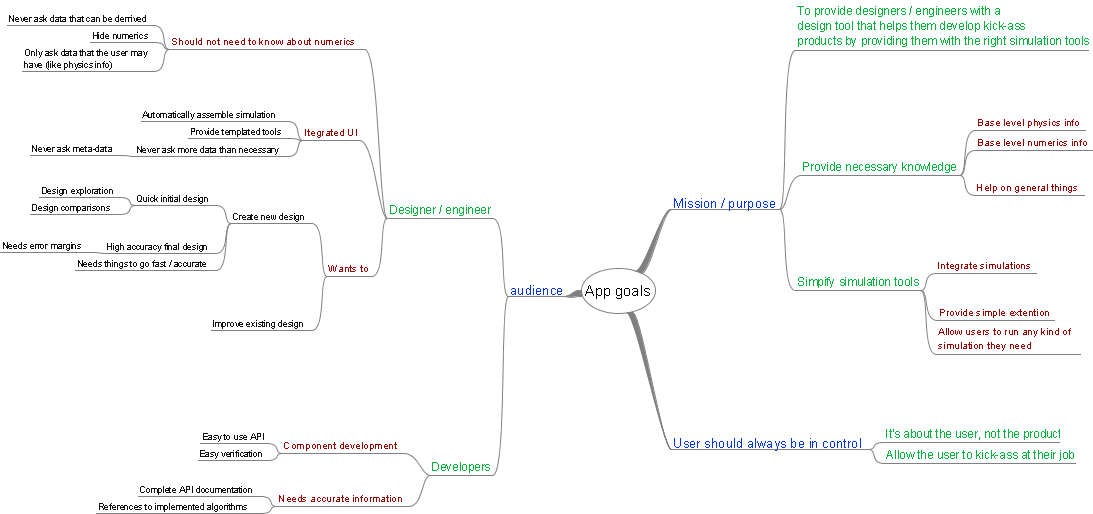


Figure : Mind map of the goals of Apollo

# 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.

Section 4 describes the different types of users which are expected to use Apollo. This is followed Section 5 which provides a limitation of the scope of Apollo by indicating what subjects are not within the scope of Apollo. Section 6 describes the different use cases which are used to determine the usage of Apollo. The use case description is followed by a description of the experiment concept used in Apollo (see Section 7 ). Finally Section 8 provides a description of the way users will work with Apollo.

# 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

* Designer / Engineer
  + Wants to
    - Create new designs
      * Quick initial design
        + Design exploration
        + Comparisons
      * Known accuracy final design
      * Must be fast + accurate
    - Improve existing designs
  + Tool should not be in the way
    - Integrated UI
    - Automatically assembly experiment(s)
    - Never ask more data than required
      * Derive meta-data without user input
      * Never ask user for meta-data
  + Should only need to know about the domain
    - Never ask data that can be derived from other inputs / state
    - Hide numerics, but allow gentle degradation
    - Only ask data that may be known (physics)
* Developer
  + Component development
    - Easy to use API
    - Easy verification
  + Information
    - API documentation
    - Logical API
    - References to implemented algorithms

# What we’re not doing

* Our own geometry engine
* Physics for every possible thing (currently only interested in heat, structures and fluids)

# Use cases

Define the different use cases.

Apollo will allow different simulations and experiments based on the same design geometry to be compared.

Users should always be able to get their data out of Apollo!

The different use cases currently envisioned are:

* Robust design
* What-if research / design
* Optimization of existing or new products
* General research

## Robust design

* 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?
* 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.
* 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.

Cool stuff?

* Comparison of different simulations or simulations & experiments

# Experiments, what and how

* The virtual experiments can be based on one or more physical models.
* Data from an experiment can be compared with data from other experiments based on the same geometry. Apollo should be able to highlight the differences and indicate where these differences were found and which experiment was responsible for the differences.
* When users modify the original geometry Apollo should be able to update the virtual experiments 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

## 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 experiment

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 experiment parameters
  + For real experiments
    - Define data import methods (??) or data location
    - Define data processing methods
  + For virtual experiments