Model-based Optimization and Visualization of Aircraft Noise

Product Planning

Contents

1 Introduction

The goal of our project is to create a standalone software application which can be used by researchers from the Air Transport & Operations department at TU Delft for model-based optimization and visualization of aircraft noise. To achieve a good development process and a successful software product careful planning is required.

In the second chapter of this document an analysis of our program is provided by describing the context, problem and stakeholder inputs. Based on this the features needed for the product are described in a high level backlog and the steps needed to complete the project are defined in a roadmap. Here you can find the major release schedule and goals.

Chapter 3 contains a detailed release plan specifying our milestones per release. In chapter 4 a definition of 'done' is given on backlog items, sprints and releases. This will enable us to know when a feature is done and when the corresponding sprint item can be closed.

2 Product

This chapter gives a high level overview of the product. Paragraph 2.1 gives a brief analysis of the program and high level overview of the product backlog. In paragraph 2.2 the steps needed to create this program will be described in a roadmap of major releases. Here the major releases of the program are given in a clear overview.

2a Program Overview

Analysis context
Aircraft and airport noise are complex subject matters which have been studied for decades and are still the focus of many research efforts nowadays. Also at the department of Air Transport & Operations (ATO) at TU Delfts Faculty of Aerospace Engineering. ATO has three research aims: 1) To develop radical new ways to optimise aircraft operations for efficiency, safety, cost and environmental impact; 2) To extend the analysis to an airline fleet and network level to include capacity and resilience; 3) To synthesize these to include operational safety at an airline and ATM level. To support their research findings at conferences, they need a visualization application that does this.

Analysis problem The ATO research department needs a program that represents a creative and efficient way to minimize and visualize aircraft noise along simulated and real flight routes. This requires the implementation of a mathematical model for aircraft noise and aircraft trajectory design. The model will be deployed by the research team to predict aircraft noise along a particular trajectory (flight route) in order to be able to minimize the produced noise over populated areas around airports. In order to implement this model the parameters that are required for the sound calculation need to be derived mathematically. Besides that, the program should also be able to visualize the noise produced along the simulated trajectory pictured on a real map. This requires an implementation of noise contours, which are noise footprints whose shape indicate areas of constant noise. Noise contours are a new subject to the research group and havent been implemented in relation with optimization regarding noise reduction before so this will be a challenging topic. The visualization should also show the effects of the produced aircraft noise on population annoyance.

Analysis stakeholder inputs The application needs to be able to read in two arbitrary data (.dat extension) or text files indicating the input trajectory (including selected airplane model) and the grid that needs to be used. These files will be given as input to the sound model. The sound model will then calculate the noise levels produced along the trajectory and the output of the sound model will be entered into the optimization model or will directly be sent to the visualization component.

Additionally, the stakeholder will deliver us some papers and extra explanation on the mathematical concepts behind the optimization model and contouring algorithm.

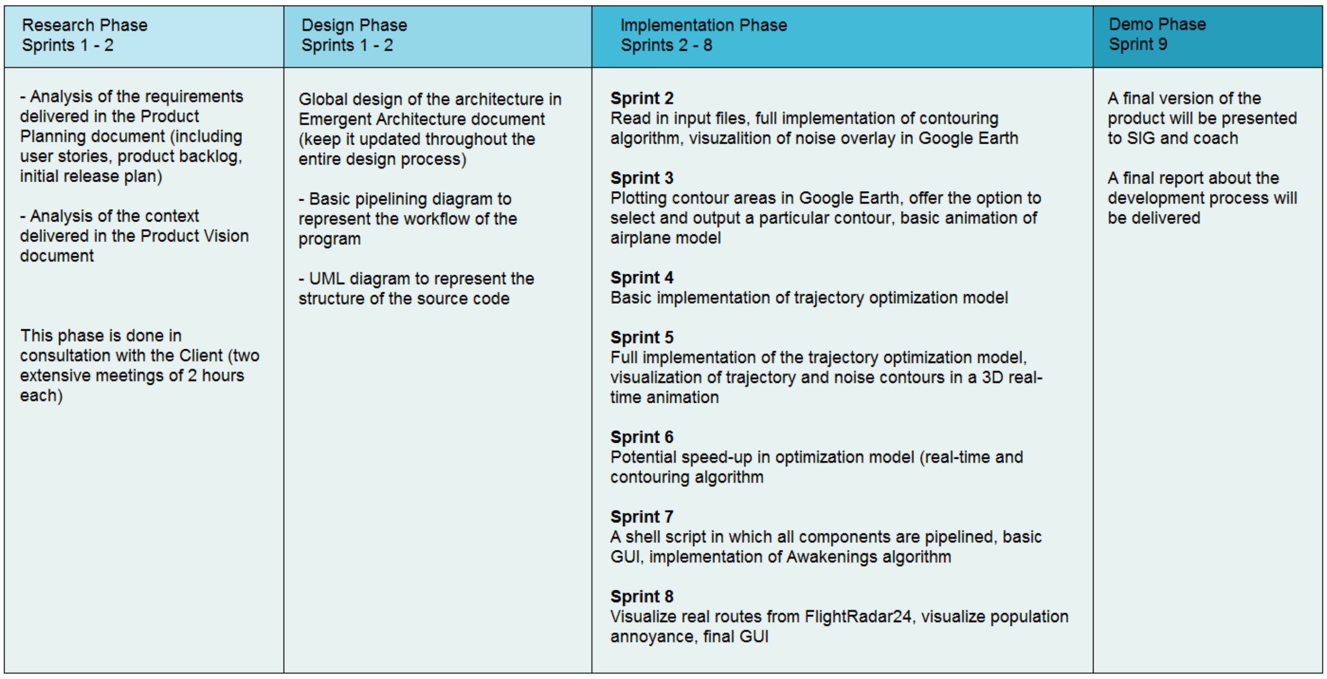
Product requirements Our product should be able to do the following tasks:

- The program can calculate noise values in real time.
- The program can calculate and output the actual noise contours that are produced along a particular trajectory in real time.
- The program has the option to optimize a trajectory in an iterated manner to minimize the produced noise over populated areas that are affected. The optimization algorithm should be based on the area of the produced noise contours instead of the actual noise values.

- The program can visualize the (optimal) trajectory together with the noise contours in real-time 3D animation mapped on Google Earth.
- The program can calculate and visualize the effects of produced noise on population annoyance using the awakenings algorithm.
- The program should save the results of the visualization and, if requested, the intermediate calculated values in a particular format and directory specified by the user.
- The program should execute all the processes above (corresponding to the computation of noise values, noise contours, noise minimization and the visualization of noise contours) in an automated and pipelined manner.
- The program should offer all these tasks in a graphical user interface.

2b Roadmap

In the roadmap below major releases of the program are spread across different phases of software development and they are given in a clear overview.



Research Phase Sprints 1 - 2	Design Phase Sprints 1 - 2	Implementation Phase Sprints 2 - 8	Demo Phase Sprint 9
- Analysis of the requirements delivered in the Product Planning document (including user stories, product backlog, initial release plan) - Analysis of the context delivered in the Product Vision document This phase is done in consultation with the Client (two extensive meetings of 2 hours each)	Global design of the architecture in Emergent Architecture document (keep it updated throughout the entire design process) - Basic pipelining diagram to represent the workflow of the program - UML diagram to represent the structure of the source code	**Sprint 2** Read in input files, full implementation of contouring algorithm, viszualition of noise overlay in Google Earth **Sprint 3** Plotting contour areas in Google Earth, offer the option to select and output a particular contour, basic animation of airplane model **Sprint 4** Basic implementation of trajectory optimization model **Sprint 5** Full implementation of the trajectory optimization model, visualization of trajectory and noise contours in a 3D real-time animation **Sprint 6** Potential speed-up in optimization model (real-time and contouring algorithm **Sprint 7** A shell script in which all components are pipelined, basic GUI, implementation of Awakenings algorithm **Sprint 8** Visualize real routes from FlightRadar24, visualize population annoyance, final GUI	A final version of the product will be presented to SIG and coach A final report about the development process will be delivered

3 Release Plan

This chapter shows an initial release plan with milestones. For a detailed overview of the user stories describing the features of our program we refer you to section 3c of our research report.

3a Initial release plan (milestones, MRFs per release)

The milestones are spread across different sprints and described below with the corresponding goals. All the releases are continually tested with system testing throughout the entire project. Please note that a sprint corresponds with one week.

Sprint 1: 18/04/2016 - 24/04/2016
This release will contain at least the following features:

- A Product Planning document containing a roadmap, overall planning and user stories that are prioritized in consultation with the client.
- A set-up of the Emergent Architecture document containing a pipeline diagram representing the workflow of our program and the way in which the noise, optimization and visualization models are connected.

Sprint 2: 25/04/2016 - 01/05/2016
This release will contain at least the following features:

- A Research Report in which the problem, context and possible solutions are analysed. This will be discussed with the client.
- An implementation of the contouring algorithm (refining the grid, finding switch points and clustering points with a similar noise level)
- An implementation of the algorithm that converts Rijksdriehoekscordinaten to WGL coordinates (long/lat)
- Basic visualization of noise contours with an overlay on Google Earth

Sprint 3: 02/05/2016 - 08/05/2016
This release will contain at least the following features:

- Extended visualization of the noise contours plotted/ mapped in Google Earth
- An implementation of the spline interpolation algorithm to smoothen out the contour lines
- The option to output actual noise data
- The option to turn on or off particular noise contours in the visualization

Sprint 4: 09/05/2016 - 15/05/2016
This release will contain at least the following features:

- Basic implementation of the trajectory optimization model (point-mass calculation)

Sprint 5: 16/05/2016 - 22/05/2016
This release will contain at least the following features:

- Full implementation of the trajectory optimization model (added: operational constraints)
- Visualization of the input trajectory and noise contours in a 3D real-time animation in Google Earth (link all visualization components together)

Sprint 6: 23/05/2016 - 29/05/2016
This release will contain at least the following features:

- Speed-up in the trajectory optimization model (real-time)
- Potential speed-up of the contouring algorithm

Sprint 7: 30/05/2016 - 05/06/2016
This release will contain at least the following features:

- A shell script in which all operations are pipelined and performed in an automated manner

- First version of the GUI containing containing a menu bar and tabs for the import of files, noise model (with raw noise level data as output) and optimization model.
- An implementation of the Awakenings algorithm to calculate population annoyance
- Emergent Architecture document presenting the final state of the architecture design.

Sprint 8: 06/06/2016 - 12/06/2016
This release will contain at least the following features:

- The option to insert a real flight route from FlightRadar24 and visualize this with a 3D animation in Google Earth
- Final version of the GUI representing the three models (noise, contouring, optimization) and all possible user operations (added: visualization tab).
- A visualization of population annoyance in Google Earth (linked with the animation)

Sprint 9: 13/06/2016 - 19/06/2016
This release will contain at least the following features:

- Solved bugs and other problems from the previous sprint(s).
- Final product containing at least all must-have and should-have requirements.
- Final report about the developed, implemented, and validated software product.

3b Definition of Done

This chapter describes the definition of done so that we both will have the same end goals. This will enable us to know when a feature is done and when the corresponding sprint item can be closed.

Our definition of done focuses on three levels: backlog items (features), sprints and releases.

4.1 Level 1: Backlog items We consider a backlog item as done when it has an test coverage of at least 75% for non-GUI elements. These tests can be divided into unit tests and other automated tests. For a feature to be merged (using a pull request) into the release version of the product, it has to be approved by the other team member. Their approval will be based on the test coverage, code readability, documentation and the overall code quality. Documentation should be added on every class and method.

4.2 Level 2: Sprints

Every sprint corresponds to one week. At the end of a sprint, all the items on the sprint plan of the current sprint have to be finished, according to the definition of done for backlog items. A new release of our product should be submitted to the version control server (Github) master. All the unit tests should pass and the system should be improved based on added or extended features. There shouldn't be any bugs/errors left in the system and the program should behave and look like the client wanted. We should also have written a sprint reflection for that sprint and a new sprint plan for the coming sprint.

4.3 Level 3: Releases

A release version of our product is done at the end of a sprint. The release version should contain the features that should have been implemented for the corresponding sprint. This also includes the minimal test coverage and other conditions described in the definition of done for a sprint. It is also important to note that the potential feedback of the client for that week should be processed before a release is completed. In the final release we should have implemented all must haves since the program can't function without these mandatory features. Most should haves (50 to 60%) and some could haves (30 to 40%), which are defined in 3.1, have to be implemented. As explained in chapter 3, these features are not necessary for the user and the system to work but our goal is to implement them partially since it would be nice to include them. It would make our program more user friendly. Lastly, the final release and other major releases should be approved by the client, the coach and obviously by ourselves too. This means that it should be simple and efficient to use for the client and well documented and designed by us. The SIG test is also an important part of this. They will determine if the code meets the standards and if it is clearly structured and documented.

As developers we also strive for maintainability and extendability, so that the system can be easily maintained, improved and updated by us or other people after our final release. This will also be taken into account throughout the entire project.