Requirements document

Potkuri-group

Helsinki December 12, 2008

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Veera Hoppula
Mikko Kuusinen
Jesse Paakkari
Tobias Rask
Timo Tonteri
Eero Vehmanen

Client

Valentin Polishchuk

Project Masters

Sampo Lehtinen

Homepage

`http://www.cs.helsinki.fi/group/potkuri`

Change Log

Version	Date	Modifications
1.0	12.12.2008	Final version
0.6	10.12.2008	E2 added
0.5	08.12.2008	Small corrections to F5 and Arrival Tree
0.4	29.10.2008	Spelling errors fixed
0.3	7.10.2008	Corrections to several parts
0.2	24.9.2008	Corrections, adding picture to model
0.1	18.9.2008	1st draft

Contents

1 Introduction

What this document is about.

1.1 Overview

This document functions as an agreement between the customer and the project group of what project will produce. Document lists vocabulary, describes interest groups and data used, explains use cases, lists all the requirements and explains system models and architecture that belong to the system. Document is also used as basis for planning.

Purpose of the program is to model and create safe paths for airplanes to land at an airport in changing weather. Goal is to create this to be as visual as possible so the user can see “a movie” of planes following flight routes to the airport. Changing weather in this project means storm centers that the planes have to avoid if storms are too intense. Both planes and weather can be randomly generated or generated by user. Weather data can also be acquired form some real-time weather source like Testbed site of Finnish Meteorological Institute.

2 Vocabulary

Airport Airport is where arrival tree begins. If map is presented as a circle, airport will be in a middle of the circle. If the map will be presented as a fourth of a circle, the airport will be at the center corner.

Arc Arcs are circles (or fourths of circles) at a determined radius distance of the airport. The merge points are located into these arcs.

Arrival tree A binary tree consisting of paths. Has a root at the airport.

dbZ decibels of Z, a measure of rain.

Flight plan Every plane has a flight plan which describes its path.

Map A map from somewhere in the world used in this product.

Merge point A point on the map where two paths merge into one path.

nmi nautical mile (=1,8520km)

Path A route to the airport that should avoid storms.

Plane An airplane that tries to land at an airport along a path avoiding storms.

Storm A set of pixels with dBZ-values above 24 dBZ close each other on the map. Indicated with red color on the map.

User A person using the product to watch animations on aircrafts landing at an airport in presence of hazardous weather systems.

3 Interest Groups

3.1 Direct

The user A person who wants to use the product to watch animations on aircrafts landing at an airport in presence of hazardous weather systems.

3.2 Non-Direct

Real-time source like Helsinki Testbed It is possible that the data for the system will be taken from some weather source from the internet. For example Helsinki Testbed is a project (of many Finnish industry and research institutes such as The Finnish Meteorological Institute and Vaisala Oyj) that provides real time weather data from southern Finland. Testbed is available at http://testbed.fmi.fi.

4 The Data

4.1 Weather Information

The weather information will be read from file. Writing to file may happen from the internet from address http://testbed.fmi.fi/ or the user may give data straight to file. Weather information the program needs is also possible to implement so that user can determine the weather (storm areas, wind direction) via user interface when he or she uses the program. When acted this way, reading from file will not be necessary. Also random storm areas generated by the program are acceptable weather information to be used. It is not required that the program should save any history information about the weather.

4.2 Airplanes

The airplanes may be generated by the program to become in sight regularly or by occurrence, or user may give data about them to file (arrival direction, time from the previous plane, speed). It is also possible to make a program the way that user can be able to determine some or all of these qualities via user interface when using the program.

4.3 Airport

The map will be adjusted so that the airport will be about in a same point in a screen every time when program is used. The airport will be presented as a point (like any other location) in a map.

5 Example Use Cases

Followings are example use case scenarios about the program use:

Basic use case Valtteri, the user of the program, is willing to get information about how many airplanes it is possible to route to the airport in a certain time when weather circumstances are determined in a way it was in a certain day (he has picture information about that day's weather). Valtteri turns the program on and gives the command-line parameter informing the program about the location of the parameter-file. Then he watches a simulation about how given airplanes fly and land to the airport avoiding storm areas. He misses some important parts of the simulation because of interruption and wants to see it again. He repeats the simulation in a bit faster speed because he does not have time and need to watch it all over as slowly as a moment ago again. After this Valtteri changes circumstances to be different, because he wants to get airplanes fly more often. He watches the new simulation and gets the information wanted. He closes the program.

Making parameter information Valtteri, the user of a program is willing to get information about how many planes it is possible to route to to the airport when there are storm areas in some exactly determined locations which are moving in exactly determined way. He writes a text document where he determines slide by slide a location of each storm area. He turns the program on and tells to it in a parameter information to use the document he has written and how many planes he wants to try to send to airport in a certain time. He repeats this until he has seen enough. He closes the program.

6 User Requirements

In the following is the list of requirements made by the customer. Some of the requirements have been specified and/or categorized.

Plane

- Planes and their routes have constant sized circle shaped safety zone around them. Both must have safety zone because planes can't fly too close to each other and routes must have safety zone to storms. Route trees branches can't be closer than this zone.
- Speed of the planes is 400nmi/h.
- Speed of the planes decelerates when approaching the airport. This is optional.
- At the outer limit of map the speed is 400nmi/h but close to the airport the speed has decelerated to 100nmi/h. This is optional.
- Planes can either slow down speed or fly zigzag to avoid collision with other planes.
- When the arrival tree changes, the plane will follow the previous branch given to it to the next merge point.
- Planes can't fly backwards.
- Planes always have to get closer to the airport.
- The program should model approaching planes.

Weather

- Weather can be randomly generated, acquired from some weather data source from the Internet, or acquired from files.
- Storm is presented in map by colored pixels. It may be presented in just one colour.
- User should be able to generate storm centers and their directions and speeds. This is optional.

Map

- Clearness of the visualization is important.
- Graphics don't need to be fancy. Weather is modeled in colors and planes as moving disks. Arrival tree's branches are modeled as colored lines from trees root, which is at the airport.
- Map is divided into 90 degree segments and each segment is divided into about 6 parts. Planes and their route trees will start from these points at the outermost arc of approaching area. This is optional.
- Zooming in map is optional.

Airport

- Airport works always as the root of arrival tree.

Algorithm

- When weather changes collide with some branch of the arrival tree, the tree will be changed. Otherwise the arrival tree will not change.
- Approaching planes can be randomly generated in the outer arc of the area. From there the planes are directed to closest merge point and then guided to airport according to arrival tree.
- How approaching planes are generated should be easily modified. This is optional.
- In situation when airport is blocked by intense storm, the most simple case is to force planes to land even through the storm. Preferably planes should be put to circle around the airport or send to another airport, but both of these cases are optional.

Arrival Tree

- Arrival tree for planes is a binary tree.
- Root of the tree is at the airport and the leaves start from where the planes enter map. One merge point can only have two children. Trees branches can't cross each other. There are about four levels in this tree.
- Radius for the approaching area is about 150nmi. This area is divided into smaller inner areas. The radius es for inner areas might be about 60nmi, 30nmi and 10nmi. Each inner area will have merge points for arrival trees. The amount of merge points can be fixed but merge points in three arcs (plus outermost arc) would be fine.
- Arrival tree should be checked again every time the weather is updated.

Merge points

- Merge points are not constant.
- Merge points change when forced by the the weather changes and can be even blocked by storms.
- Merge points on same level of tree don't need to be at the same exact distance from airport. It can vary a little.
- When plane is given a merge point it will have to be closer to the airport than the previous one. Only exception to this can be when storm blocks the airport.

General

- Program should basically be a movie.
- There does not have to be a scroll bar available for time line.

- Purpose of the program is to count and model as safe as possible way to approaching planes to airport through changing weather conditions.
- Basic idea of program is to get answers to question "how many airplanes it is possible to route to the airport in a certain time when weather circumstances are in a certain way."

7 System Requirements

System requirements are conditions related to system. Certain functionality can be required from system after user has accomplished a use case. System requirements can be totally invisible to user. Priority for each requirement is given in scale from one to three. One (1) meaning on the scale that the requirement is obligatory. Two (2) meaning that the requirement is essential but the system is partially usable even with out it. Three (3) meaning that the requirement is only implemented if there is enough time for it.

7.1 Functional Requirements

Functional requirements are closely related to user actions or their consequences.

F1 - Planes have a location at certain time on map
Description: Planes can not disappear.
Priority: 1

F2 - Planes never fly over intense storm
Description: Arrival tree can't go over too intense storm.
Priority: 1

F3 - Arrival tree's branches never cross
Description: Arrival tree's branches have certain safety zone that no other branch can touch.
Priority: 1

F4 - Weather data is acquired from file
Description: Weather data is read from file.
Priority: 1

F5 - The program shows visually planes, arrival tree and storms
Description: Planes will be shown as discs moving along arrival tree. Arrival tree is shown as colored lines. Storm maybe colored by one colour.
Priority: 1

F6 - Parameters for program are in text file
Description: As many as possible parameters of the program will be stored in separate text file.
Priority: 1

F7 - User can create weather conditions
Description: User is able to input storm centers and wind speed manually in text file or in bitmap form
Priority: 2

F8 - Map can be zoomed
Description: User is able to zoom map in and out.
Priority: 3

F9 - User is able to give amount and arriving place of the planes
Description: Parameters are given in text file that describe the above variables.
Priority: 3

F10 - Wind has direction and speed
Description: Wind has direction in degrees and speed can not be negative.
Priority: 1

F11 - Weather data is generated randomly
Description: Weather data is generated randomly.
Priority: 2

F12 - User is able to give storm centers, their intensity and wind speed
Description: Parameters are given in text file that describe the above variables.
Priority: 3

F13 - Weather data is acquired from Testbed
Description: Weather data is acquired from Finnish Meteorological Institute's Testbed site.
Priority: 3

F14 - Program stores every weather data picture from Testbed to hard drive
Description: Every picture of weather data received from Testbed is stored.
Program can use this data to simulate weather data if no Internet connection is available.
Priority: 3

7.2 Non-Functional Requirements

Non-functional requirements are properties of system that the user can't see or the user can't directly use. User interface requirements are non-functional.

N1 - Program works fluently
Description: There should no visual pauses when planes, routes and storms are drawn to user.
Validation: Planes move towards airport continuously, arrival tree changes when storm comes to it's way and storms are updated on map always after some suitable time.
Priority: 2

N2 - Program is highly visual
Description: Planes, arrival tree and storms are displayed in a way that they are easily distinguishable. User is able to clearly see what every mark and object means.
Validation: Every member of our group and the customer must agree that the program is highly visual.
Priority: 1

7.3 Environment

Environmental requirements describe programs relation to operating environment, surrounding world and interfaces.

E1 - Program runs on relatively modern laptop computer
Description: Program must work on laptop that is no more than two years old.
Priority: 1

E2 - Program requires Java 6.x runtie environment
Description: Java runtime environment version 6.x or greater must be installed on system.
Priority: 1

8 System Models

The program consists of five major parts namely parameters, weather data, plane position information, route planning and graphics.

Parameters Program uses parameters to direct execution of the program. These parameters are saved in system and can be edited before program is run.

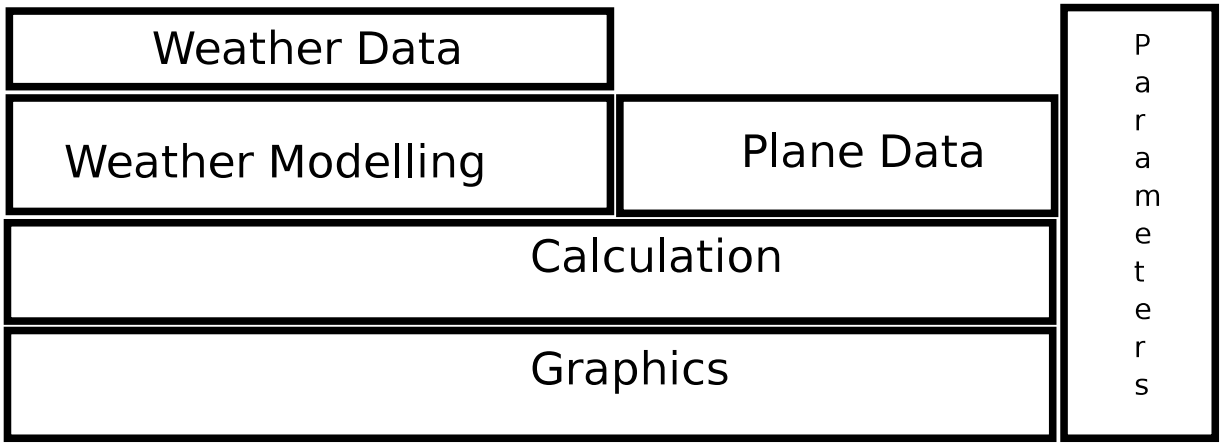
Weather data Handles feeding weather data from some text/picture file or from some weather source from the internet.

Weather Modelling Handles calcutions related to the weighted net calculated from the weather data.

Airspace Stores and handles information about the planes in the airspace, including, but not limited to plane position, speed, plane type and destination merge point.

Calculation Handles the calculations involved in planning the merge point distribution and plane movements along the routes between the merge points.

Graphics Handles visually displaying the information from weather data, plane position and route planning.



9 System Architecture

This part introduces you to the architecture of the system. The chapter has been shared into two parts. First the environment and then the architecture itself.

9.1 Environment

Program may take required data from some weather source from the internet like Helsinki Testbed, but this is is optional. The primary option is to use some image or text data, in *static file(s)*.

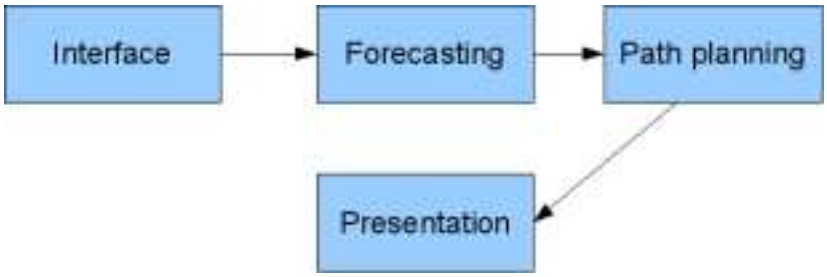
9.2 Architecture

The program has at least three parts. The outermost is an interface. It reads weather data input in text or image format. The input can be made by the user or downloaded from a weather service site in the internet. Interface uses *static files*. The interface converts the image into a weighted graph.

Another part is the one that does the actual calculating of arrival trees. It uses graph to make routes for the aircrafts. It also takes to count the merge points and decides where the planes should meet. As the weather possibly changes, the calculation is re-run. So the *flight paths* will be always up to date.

It is also possible that the program tries to analyze the graph and guess where the storms are moving. (That would be one more part to the system.) This would include calculating wind speeds and storm coordinates. This would also create an array that could be used to find the best possible *arrival tree*.

Graphical presentation of the map, planes and paths are drawn to the screen in the last part of the system. It creates a simple window that presents the whole programs operation to the user.



10 System Life Cycle

10.1 Getting Started

Read more about getting started from user's guide.

10.2 Maintenance

Group Potkuri plans and pilots a program, which is described by this requirement document. The document is constructed from customer's demands of the task, and requirement document has gain customer approval. There will be a Test Plan, which is a model for actual testing.

Programs structure is designed to support further development and furthermore transfer to other environments. With this in mind, the project will be well documented. Group Potkuri will not be responsible for further development or maintenance.

10.3 Uninstalling / Removing

To uninstall software just delete programs main directory. Read more about uninstalling from user's guide.