Milestone 1

Técnicas Avançadas de Programação

1180761 – Jorge Pessoa

1181597 – Rui Alves

1191421 – Luís Neves

## Glossary

1. **Agenda** – Information block containing the existing runways, the aircrafts intending to land and the maximum delay time a regular aircraft can experience.
2. **Aircraft** – Vehicle that will need assignment to a designated runway based on a series of parameters. An Aircraft is characterized with an *ID*, an *Aircraft Class*, a *Target Time,* and an optional *Emergency Delay*. The *Target Time* is the base reference for the ideal time of an aircraft operation to occur, this can be delayed by the maximum time delay defined in the *Agenda*. If an aircraft has an *Emergency Delay* value, it means it’s in an emergency setting and cannot be delayed more than the specified value.
3. **Runway** – Airport runway characterized by an *ID* and a list of *Aircraft Classes* it can handle.
4. Uma imagem com mesa

   Descrição gerada automaticamente**Aircraft Class** – Characterizes an aircraft based on its type of operation (*Landing* or *Taking-Off*) and its weight (*Small, Large* and *Heavy*). This is defined at the start of the project requirements, as we can see in Figure 1.

Figure 1 - Aircraft Class Definition

1. **Separation of Operations** – Set of rules defined at the start of the project requirements that define the mandatory necessary time between operations in a runway. As seen in Figure 2, the different types of operations between different types of aircrafts makes a big difference in the runway assignment algorithm.  
   Uma imagem com mesa

   Descrição gerada automaticamente

Figure 2 - Separation of Operations Table

1. **Schedule XML** – One is created for each assigned *Aircraft*. It is characterized by the Actual Target the *Aircraft* got to land, the associated *Cost* in case there was a delay, a *Runway* and of course an *Aircraft*.

## Domain Model

As seen in Figure 3, the domain model for the present project is relatively simple, where all the above mentioned connections and relations are present.

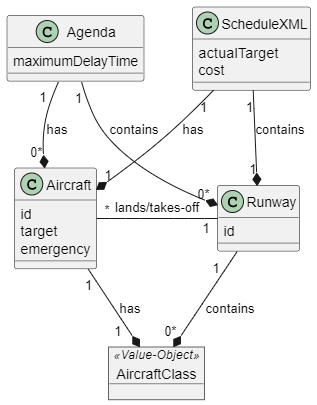


Figure 3 - Domain Model

As seen above, we can extract the main objective of the assignment, which is to elaborate a *Schedule* for each aircraft present in the *Agenda* while taking into account all the necessary restrictions, such as the maximum delay, the specific *Aircraft Classes* a *Runway* accepts and so on.

## Domain Implementation

When thinking of and designing the development process for the project, the team thought of the best practices one can take to assure domain level correctness and validation.

Uma imagem com texto

Descrição gerada automaticamenteTo do so, the team implemented opaque types for domain rules validations, such as the implementation of a *Non Negative Integer*, a *Positive Integer,* or an *Identifier*, as seen in Figure 4, where there’s an example of the *Non Negative Integer* type.

Figure 4 - Opaque Type Implementation

This concept is visible and implemented throughout the algorithms and adjacent methods, as seen in Figure 5.

Uma imagem com texto

Descrição gerada automaticamente

Figure 5 - Aircraft Implementation

To simplify the process of consulting the hardcoded values for the Separation Rules and the Aircraft Classes the team implemented a Utils class to return the necessary value for the required Aircrafts, as seen in Figure 6

.

Uma imagem com texto

Descrição gerada automaticamente

Figure 6 - Utils Class

## Algorithm Implementation

In this section, we will describe the logical process associated with the elaborated algorithm in smaller, more comprehensive steps.

1. Load information from XML file. Using the parsing methods implemented in the respective classes, the process starts by loading the information to create an *Agenda*, as seen in Figure 7.  
     
   Uma imagem com texto

   Descrição gerada automaticamente

Figure 7 - Initial Methods

* 1. Uma imagem com texto

     Descrição gerada automaticamenteThe algorithm then starts by validating the received *Agenda* object and its children to catch any errors in the XML formatting, such as repeated *ID*s, or unassignable *Aircrafts*. As seen in Figure 8.

Figure 8 - Domain Validation

* 1. We then call the *assignRunway* method that takes an *Agenda* object as parameter and defines a tail recursive method that takes a list of Remaining Aircrafts and a list of Assigned Aircrafts. The list of Assigned Aircrafts is initialized as empty.  
     1. We start by iterating over the list of *Aircrafts*
        1. Use a helper method to get runways that can accommodate the current *Aircraft.* Returns error if no runways can accommodate the *Aircraft*.
        2. Iterates over available *Runways*.  
           1. If no Aircrafts have been assigned to the current Runway, assigns aircraft to current runway and calls loop again.
           2. If Aircrafts have already been scheduled, check the availability of the rest of the Runways. If any Runway is empty and available to receive that Aircraft, assign to that one.
           3. If there are scheduled Aircrafts and there are not empty Runways, the algorithm then tries to assign the Aircraft to the Runway that will cause it the least delay using the *findBestRunway* method.

The method starts by elaborating a List of Tuples containing the Runway and the required time to assign the Aircraft to it, after that, it iterates through the list and finds the smallest time value, thus finding the best Runway.

* + - * 1. With the Tuple return of the previous method the algorithm now calculates the delay that will be applied to the aircraft.

If the delay is less than the *Agenda*’s Maximum Delay Time, the algorithm calculates the cost of the delay and the Aircraft’s updated target time.

* + - * 1. The algorithm then checks to see if this is an Emergency Aircraft and if the calculated delay is bigger than the allowed Emergency Delay, the algorithm calls the method *findFitForEmergency*, since the Emergency Aircraft has priority over the rest of the regular Aircrafts.

This method takes as parameters the list of assigned Aircrafts and the Emergency Aircraft to be assigned and defines a tail recursive method inside that takes as parameter the remaining list of Assigned Aircrafts, the list of Aircrafts that were assigned but will be removed and the list of Aircrafts that are deemed untouchable.   
For example, if we want to add an emergency Aircraft to this list that represents a Runway, the red colored n’s will be untouchable since they would alter the emergency Aircraft that is already in that Runway because they were scheduled before it.  
[normal, n, n, Emergency 1, n, n, n] <------ Emergency 2   
On the other hand, the green n’s are alterable since they were scheduled before.

The method then evaluates if the Runway Interdict List does not contain the current Aircraft’s ID. Because if it does, we need to skip to another runway, since that one is already searched and not available for our needs.

Then, the necessary separation time is calculated between the Aircraft in question and the Emergency Aircraft.

We then calculate the time interval the Emergency Aircraft can do. That is, the Emergency Aircraft Target + the Emergency Target. For example, for a Target of 50 with an Emergency Target of 5, the interval would be [50, 55].  
What the method does is calculate if the current Aircraft + it’s Separation Rule is less or within those boundaries. If it’s less than the lower boundary there is no delay to the Emergency Aircraft, if its between the boundaries, it calculates it’s delay and assigns the Emergency Aircraft to the Runway with the updated Target.  
  
If it’s above the higher boundary, the current Aircraft needs to be unassigned (removed from the list of Assigned Aircrafts) and the loop is called again, now with the next Assigned Aircraft to make the same calculations and see if the Emergency Aircraft can be assigned. Worst case scenario, the Emergency Aircraft substitutes every Aircraft and takes first place in the Runway.  
  
When it finds a spot for the Emergency Aircraft, it returns and exits the loop.

After assigning the Emergency Aircraft to a Runway, the regular calculations need to be made, such as the cost.

After that, the algorithm verifies if the removed Aircraft was the first in the Runway, to then assign the Emergency Aircraft as the first.

* 1. When the loop exits, we then catch the correct output and write the list of Schedules to a file.

## Testing

The group was able to complete 18/20 tests that came with the project, making a final score of 90.  
Regarding other tests, the group implemented Unit and Functional Tests.  
As seen bellow, in Figure 9, there’s an example of a Unit Test for the creation of an Aircraft Class.  
  
Uma imagem com texto

Descrição gerada automaticamente

Figure 9 - Unit Tests