Prodavam Metla Solutions

[Email address]

Abstract

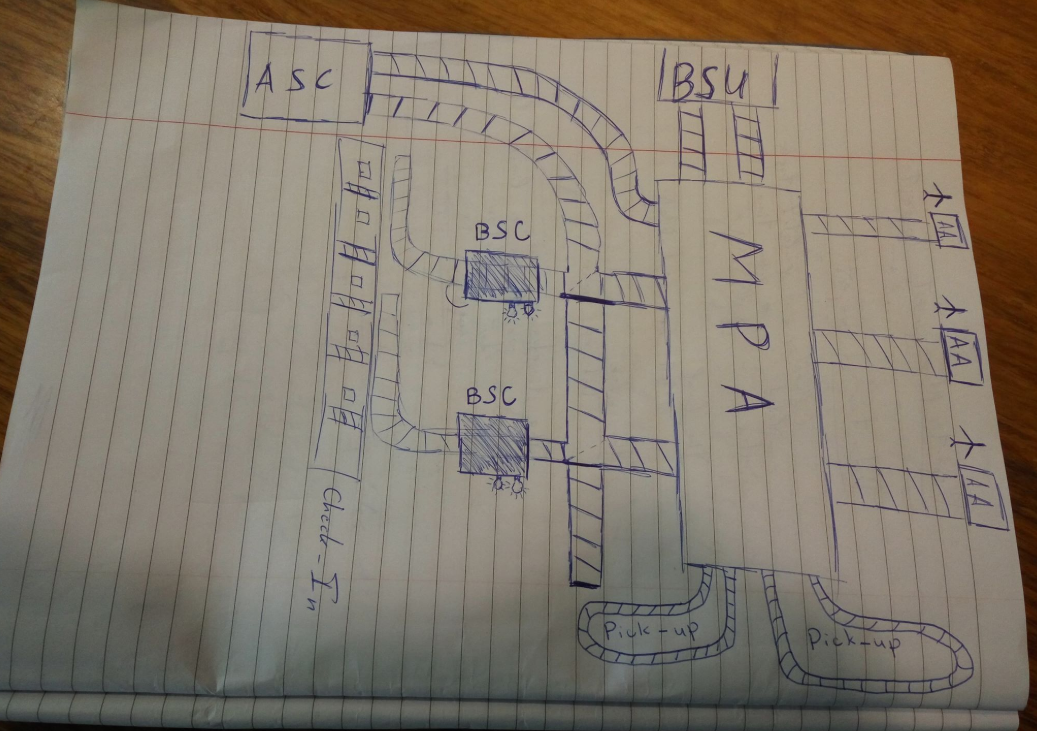
The following document describes the software and visual design of the BHS Simulation Software. Its intention is to provide insight over the working process of Prodavam Metla.

Design Document

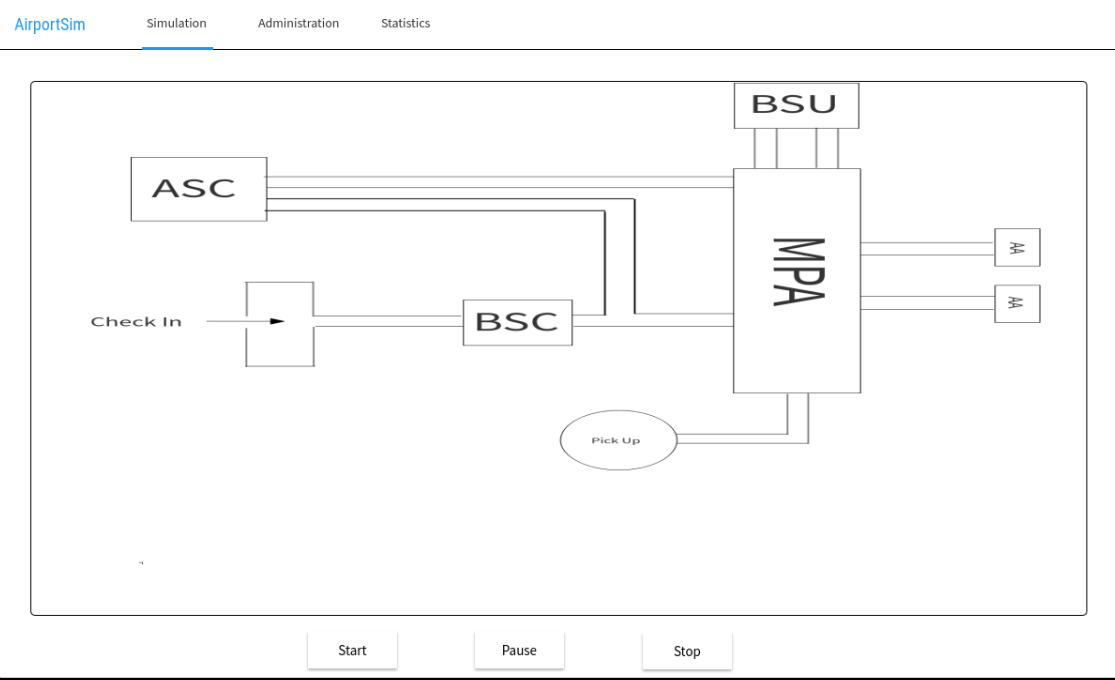
Version 0.1

# Front-End

In our first meetings, the team decided to follow different internet sources in order to understand better the way a baggage is handled at the airport. We came up with a simple drawing of how we imagined the whole process would look like (The abrreviations are explained under Processing Node and Complex Node below) :



After that we turned it into a more simplified wireframe with only one intial check-in and only one pick-up position:



## Iteration Two Changes

## User Interface (UI)

The user interface consists of two windows. A Simulation window and a Statistics Window.

The Statistics Window shows different information about the simulation in graphical representation

(charts, graphs etc.)

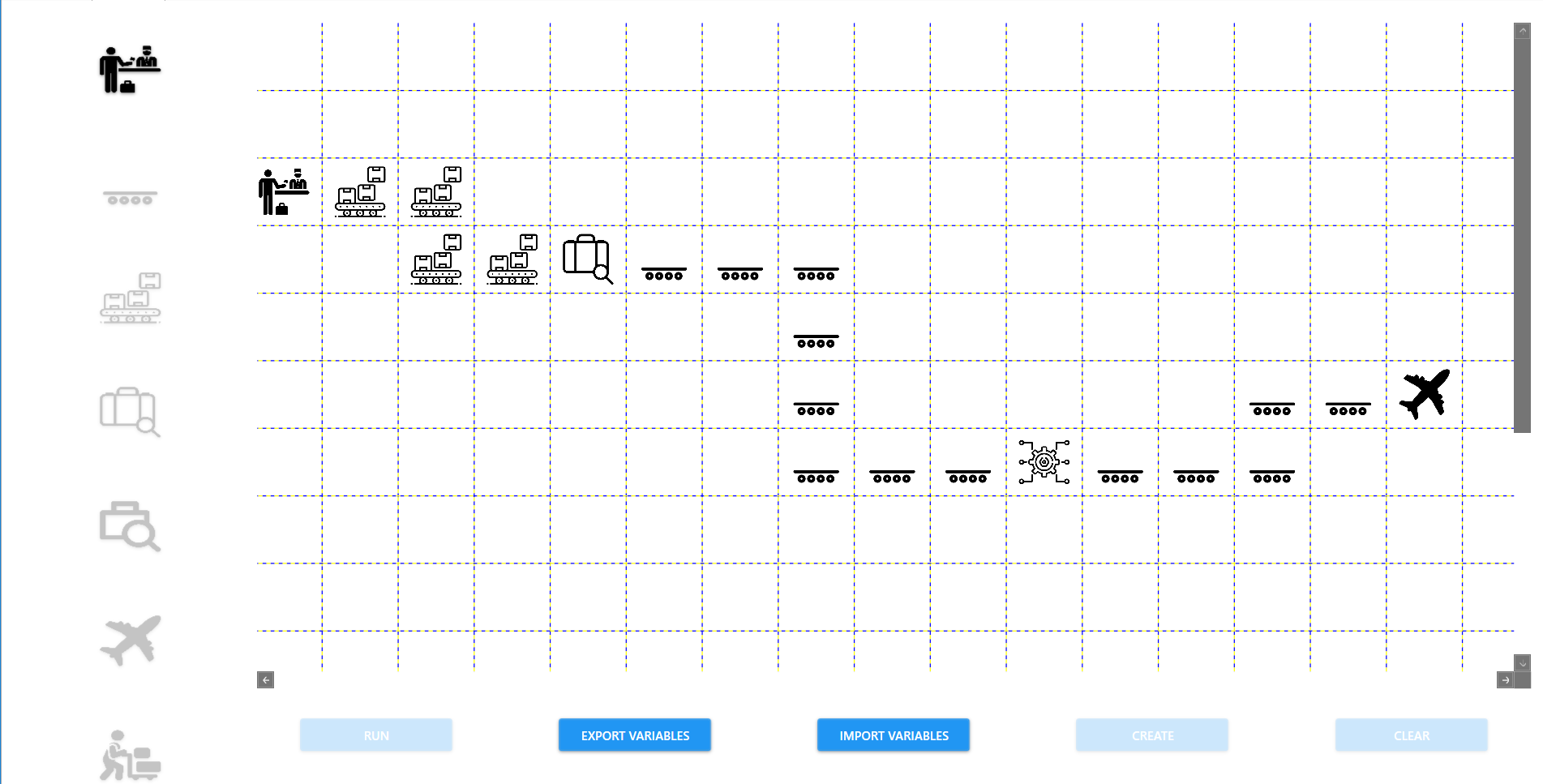
The Simulation window consists of three parts. On the left hand-side we can see different images indicating different elements (check-in area, conveyor, security checks etc.) In the middle of the screen is an empty grid window where the simulation can be built and seen visually. Below that window 5 buttons reside each with specific functionality.

1. The “Run” button starts the simulation and upon finishing shows information in the statistics window. This button is available only when eveything required for the simulation is created in the model
2. The “Export Variables” and “Import Variables” are respectively there to save the currently built simulation(export) for later use and load it again when desired(import). The Export button is available only when at least one simulation has run and the Import is available only when at least one simulation has been exported.
3. The “Create” button has to be clicked everytime a new element is put on the grid. Only in this way the user can signal that he wants a specific element in that specific location
4. The “Clear” button clears the whole grid for a fresh start. If the user wants to remove only one element he can right-click over it to remove. This button is available only when at least one element exists and before the whole model is complete.

## Building the Simulation

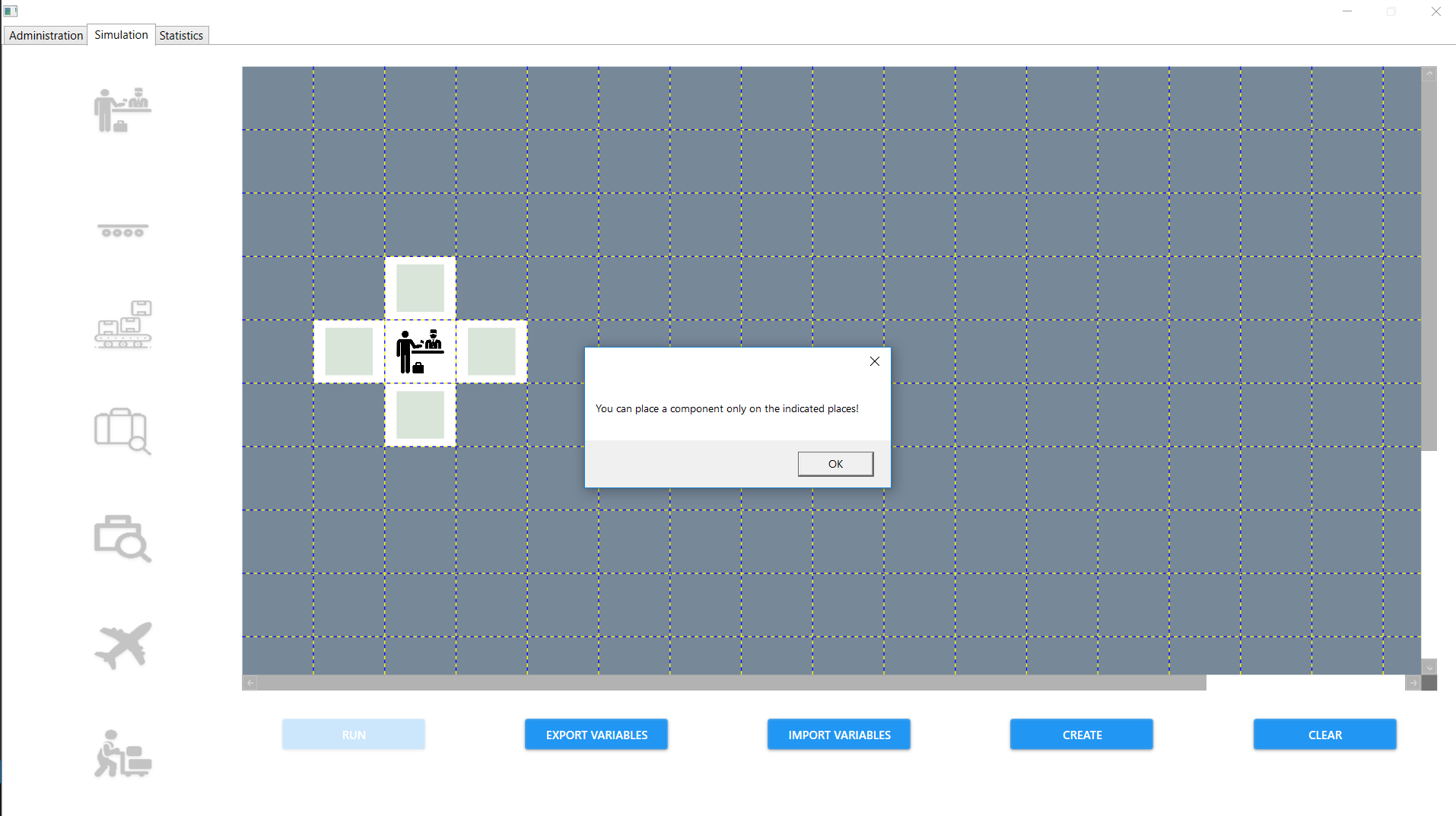
When first building a simulation all elements on the left side are not available except for the ‘Check-In’ element, which initiates the model. To put an element on the grid the user has to left-click the icon and pick a desired location. After creating one or several check-in areas the user is allowed to connect them with conveyors. This structure continues with the Primary Security Check then MPA and at the end the Airport Area - all having conveyors in between them for transportation.

Example of a model (not having all components):



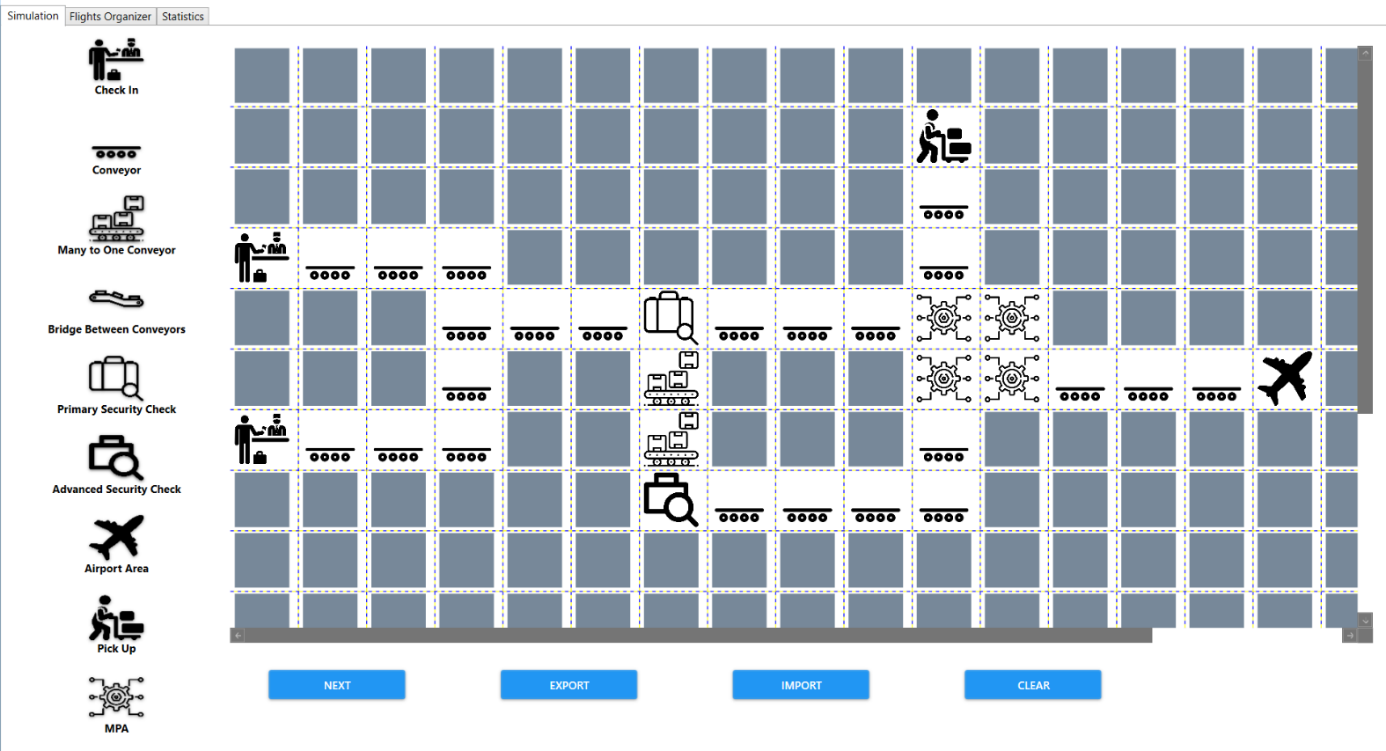
## Constraints

The user has the option to create a new element only if it’s next to a previously created element or anywhere on the map if it’s the first element in the model. This ensures a fool-proof application that will always have a connected model. For example, when creating our very first element we have four paths in which the model can proceed. From here a conveyor can only be created in the 4 squares available.

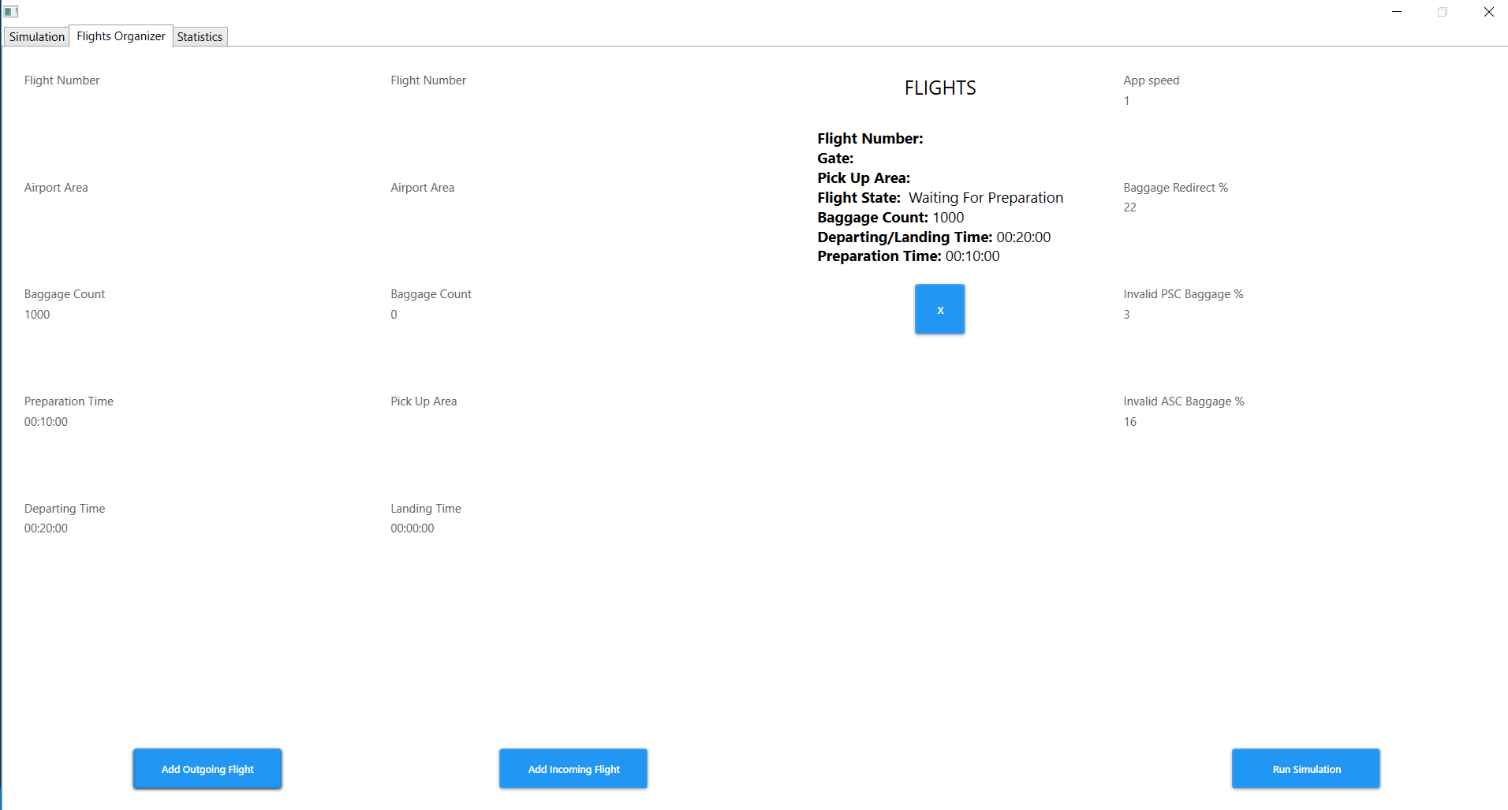


Final Iteration Changes

For the last iteration the button ‘’Run’’ was changed with a grayed-out button named ‘’Next” and the button ‘’Create’’ was removed. The Next button is only available for clicking only after at least one Airport Area is put on the grid, which means at least one full route from check – in to airport area has to exist.

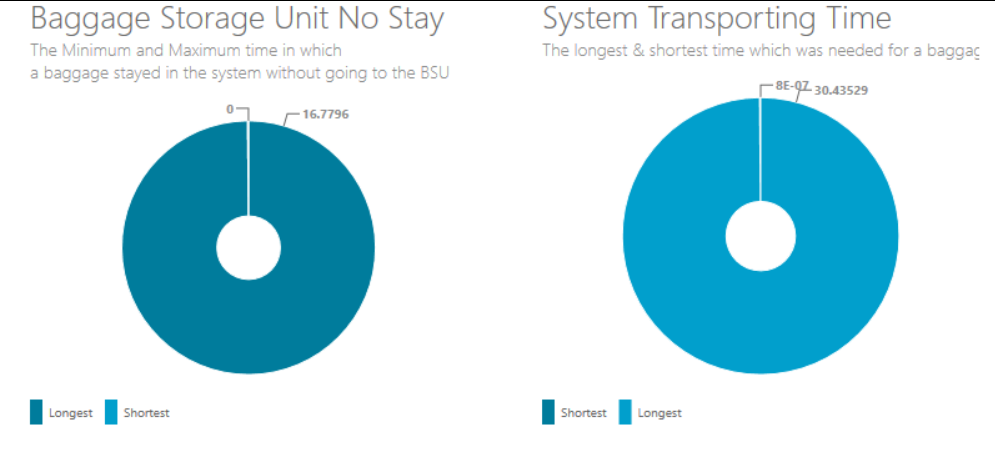


Upon clicking the ‘’Next’’ button the user is redirected to a newly created tab called ‘’Flights”. Here the user can add a flight based on the settings that he has given in the grid and also add different variables on which the statistics would work. For example, fail percentage, processing time required, number of bags on airport etc.

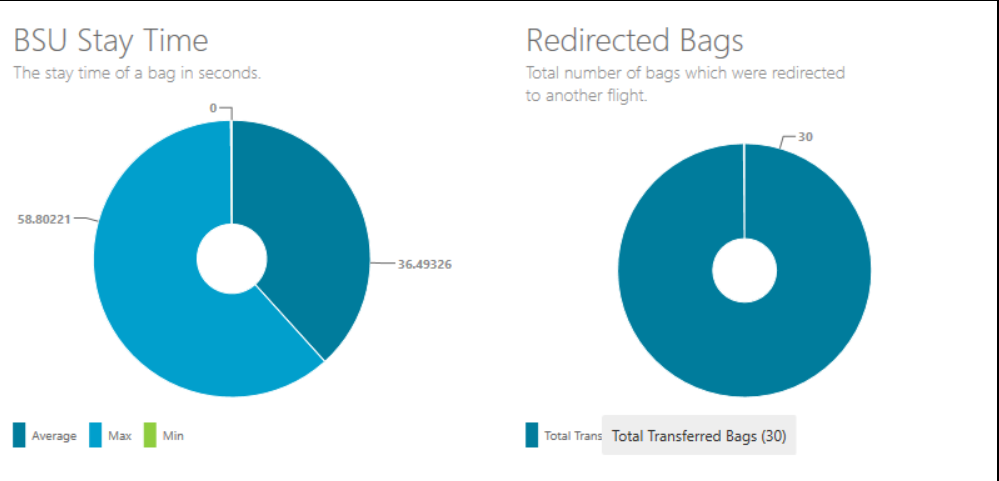


Then the user can click the ‘’Run” button which will transition him to the last tab of the application – the statistics tab where he would be able to see how the stats change while the application is running the whole process. The statistics would be different based on what the user inputs.





In the statistics tab different information can be observed like number of collected/dispatched bags, the total number of bags being transferred and bags delayed, the delays per flight, the minimum maximum and averages BSU stay time and more.



# Class diagram

The class diagram is somewhat daunting on first sight, however, becomes relatively simple once you get the gist of it. The classes can be divided in two sections – “BHS Components” and “Helper classes” (Fig. 1)

## BHS Components

All components are derived from an abstract class called *ChainLink. ChainLink* further spawns two abstract classes – *ProcessingNode* and *TransportingNode.* From ChainLink is derived an additional group of nodes, which do not fall under the scope of neither *ProcessingNode, nor TransportingNode.* For simplicity we are going to name them *Complex nodes.*

A *ChainLink* is the basic outline of a component and allows it only to be connected with other components (ChainLinks).

### TransportingNode

A *TransportingNode* is a component provided with the ability to “move” *Baggage* (imagine a conveyor). TransportingNode parents two classes *OneToOneConveyor* and *ManyToOneConveyor.* (Fig. 3) *OneToOneConveyor* connects one *ProcessingNode/Complex node* to a one other *ProcessingNode/Complex node. ManyToOneConveyor* allows the connection of the one or more components to a single other component.

### ProcessingNode

*ProcessingNodes* are all components who alter the properties of a *Baggage* or have inconstant *successor (*a *successor* is the next *ChainLink* to which the current one will pass the *Baggage* object). *ProcessingNode* is the common denominator to a number of components: (Fig. 4)

1. *CheckInDesk* – the entry point of a *Baggage* object into the BHS.
2. *Psc (Primary Security Check)* – simulates an X-Ray scanner in the most basic sense.
3. *Asc (Advanced Security Check)* – a simplified model of an area where baggage is sent to be further examined. A *Baggage* is sent to the *Asc* if it fails “scanning” by a *Psc.* If a *Baggage* fails *Asc* scan it’s “swallowed” by the *BagCollector.*
4. *Aa (Airport area) –* an *Aa* is where *Flights* “arrive” and “depart”.
5. *Robot –* robots are used in the BSU (Baggage Storage Unit). Their task is to “sort” baggage to its respective *BaggageBucket.*

### Complex Nodes

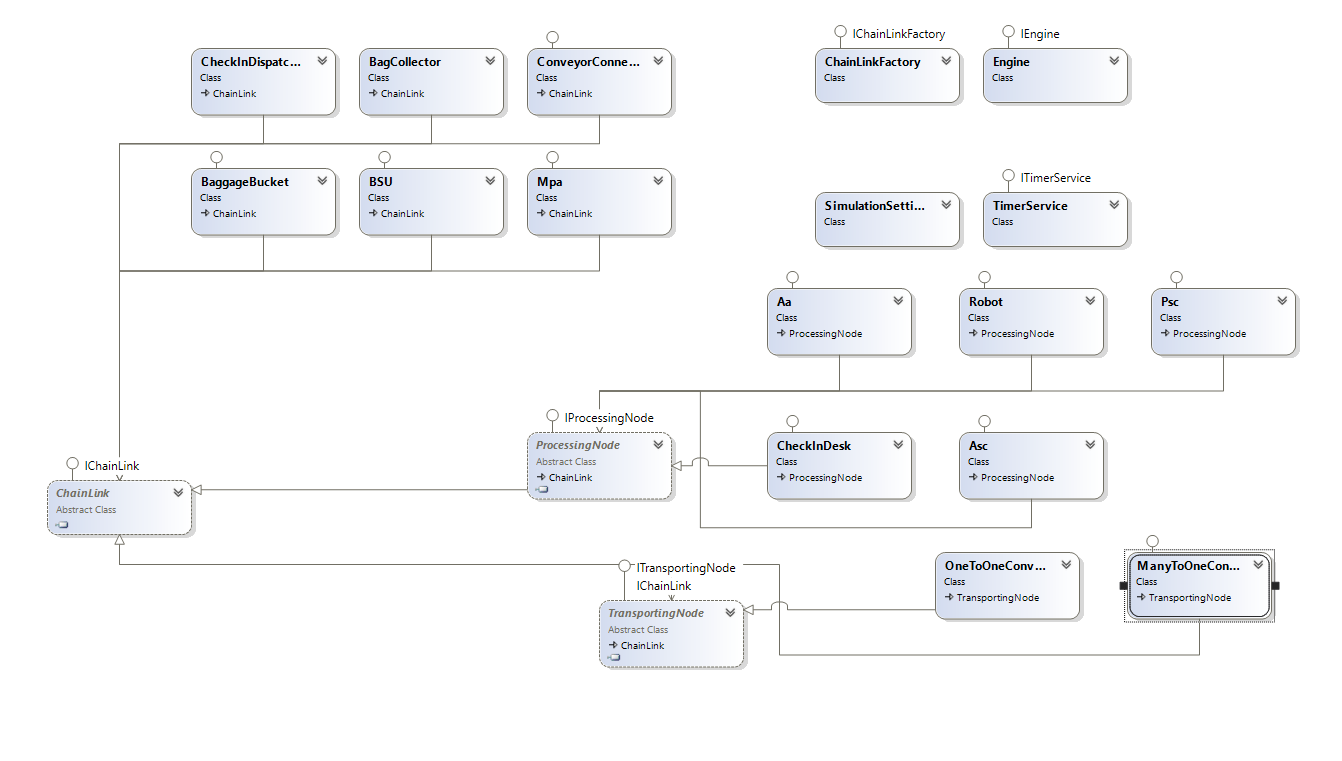
Due to their specifics some components cannot be classified as *TransportingNode* or *ProcessingNode.* They are derived directly from *ChainLink*. Such components are: (Fig. 5)

1. *CheckInDispatcher* – used to simulate the flow of passengers at the *CheckInDesks.*
2. *AADispatcher –* used to simulate the flow of *Baggage* from “arriving” *Flights.*
3. *MPA (Main Processing Area)* – the *MPA* resembles a sorting conveyor. Once *Baggage* passed to the *MPA,* it is distributed to its respective *Destination (Aa, BSU, PickUpArea).*
4. *BSU (Baggage Storage Unit) –* the *BSU* is a simplified model of a storage area used for storing *Baggage* that has arrived in the BHS too early for its *Flight.*
5. *BagCollector –* used as a Black hole that “swallows” all *Baggage* processed by an *Aa, Asc, PickUpArea.*
6. *ConveyorConnector –* A *ConveyorConnector* is not an actual component. It rather serves as an adapter which enables other non-*TransportingNode* components to connect with a *ManyToOneConveyor*.

## Helper Classes

In addition to our components, we have “Helper classes”. They are used to build the model or provide time management options. Helper classes are defined as follows:

1. *ChainLinkFactory –* Autofac Factory to build *ChainLinks.*
2. *SimulationSettings –* User-customizable settings that are used by the *ChainLinkFactory.*

Figure . Full Class Diagram

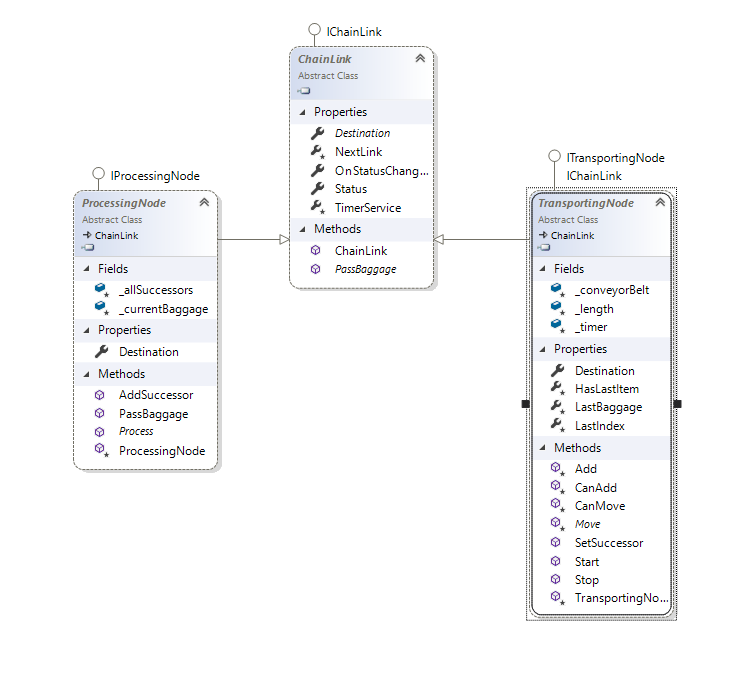


Figure . ChainLink and its children

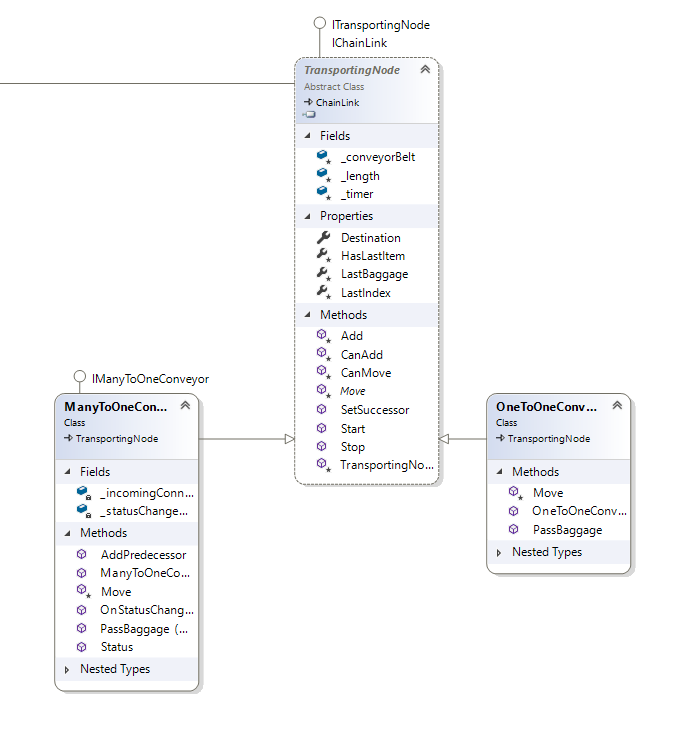


Figure . TransportingNode and its children

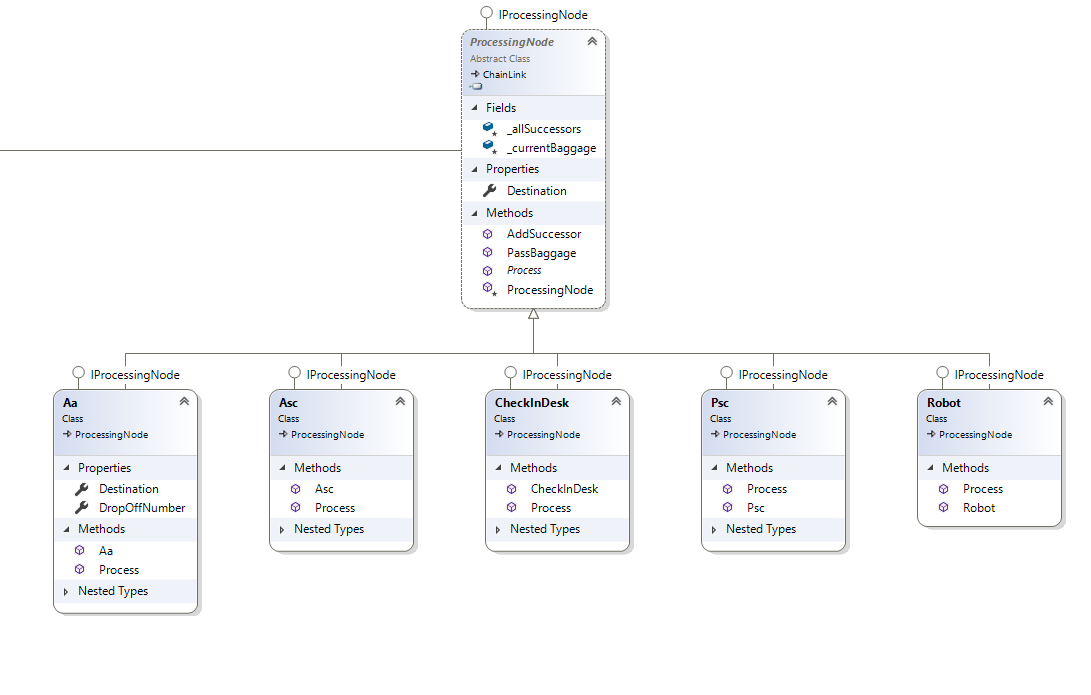


Figure . ProcessingNode and its children

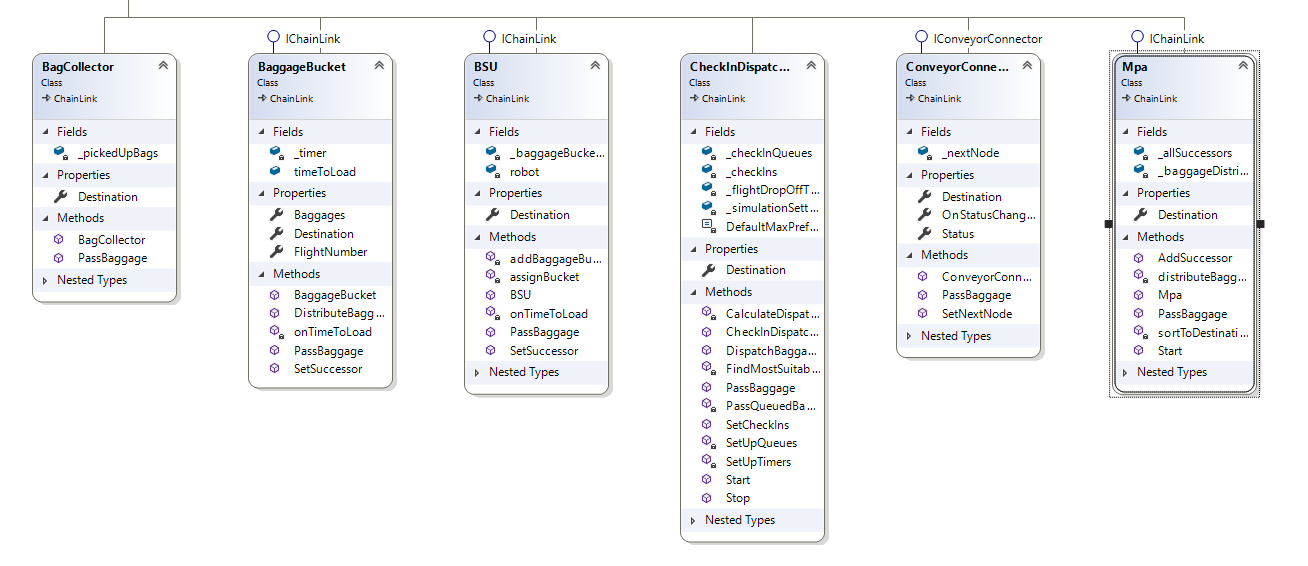


Figure . Complex nodes

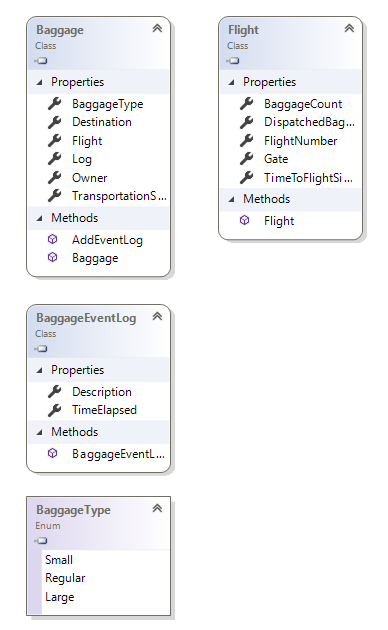


Figure . Baggage and Flight

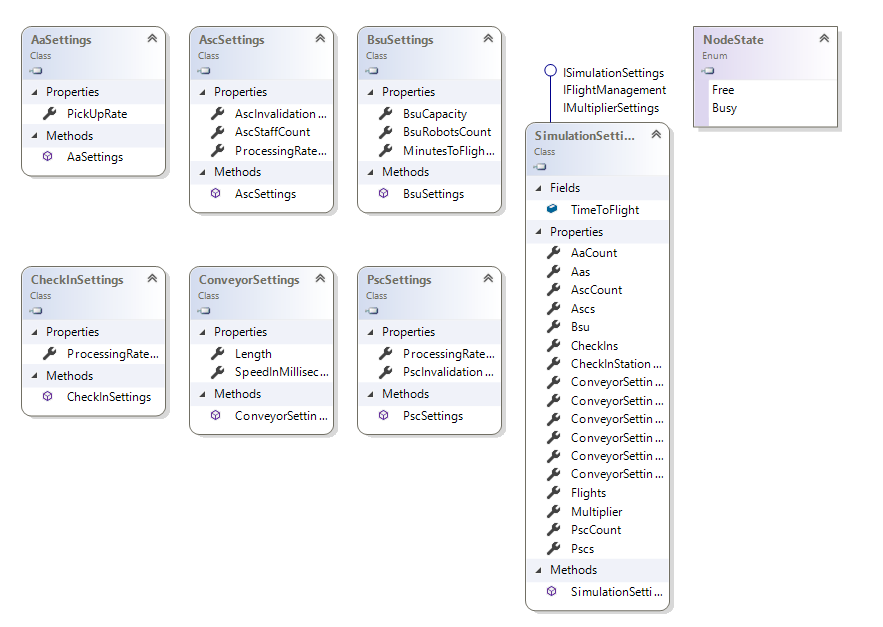


Figure . Settings