|  |
| --- |
| FONTYS UNIVERSITY OF APPLIED SCIENCE |
| Design Document |
| Parcel Handling Simulation |
|  |
| **GDS - Group 3** |
| **10/8/2010** |

|  |  |  |
| --- | --- | --- |
| Document name |  | Design Document |
| Project name |  | Parcel Handling Simulation |
| Version |  | 2.0 |
| Department |  | ICT |
| Client |  | Casper Schellekens |
| School tutor |  | Peter Boots |

|  |  |  |
| --- | --- | --- |
| Group leader |  | Qian Li |
| Group members |  | Antoine Girard |
|  |  | John Ibeagha |
|  |  | Kritian Kolev |
|  |  | Sebastien Lepage |

**Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Related pages | Brief Description of Change |
| 1.0 | 2010-09-22 | All |  |
| 2.0 | 2010-10-05 |  |  |
|  |  |  |  |
|  |  |  |  |

**Review and Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Review** | | | |
| Version | Date | Reviewed by | Position/Role |
| 1.0 | 2010-10-04 | Casper Schellekens | School tutor |
| 2.0 |  | Casper Schellekens | School tutor |

|  |  |  |  |
| --- | --- | --- | --- |
| **Approval** | | | |
| Version | Date | Approved by | Position/Role |
| Final |  | Casper Schellekens | School tutor |

**Table of Contents**

[1. Introduction 1](#_Toc274249781)

[1.1 Purpose 1](#_Toc274249782)

[1.2 Scope 1](#_Toc274249783)

[1.3 Users 1](#_Toc274249784)

[2. Project Design 2](#_Toc274249785)

[2.1 Class Diagrams 2](#_Toc274249786)

[2.2 Classes and Methods 4](#_Toc274249787)

[2.2.1 Airport Library Class Diagram 4](#_Toc274249788)

[2.2.2 Handling Simulation Class Diagram 5](#_Toc274249789)

[2.3 Sequence Diagrams 5](#_Toc274249790)

# Introduction

## Purpose

This document represents the design and the architecture of the Parcel Handling Simulation project. The design is introduced in detail for the whole group to understand the whole architecture of the application.

## Scope

This document covers the class diagram for the Parcel Handling Simulation application and a clear description of all the methods used in our design. Specifically, this design covers the following aspects:

* UML modelling;
* Custom User Interface components;
* C# programming

## Users

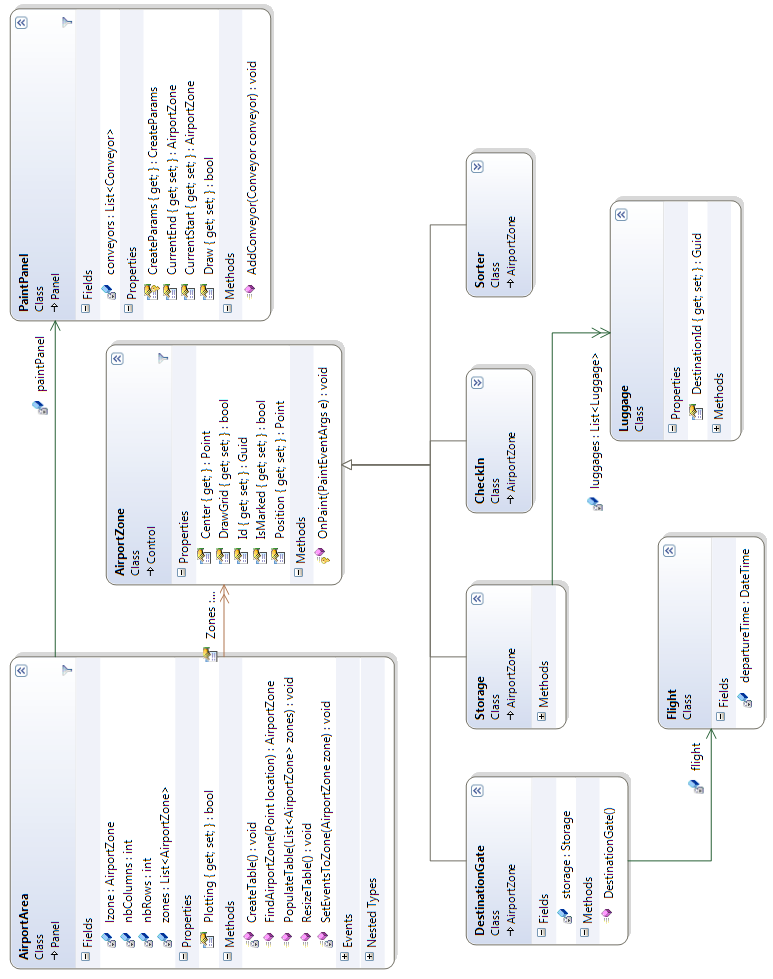
This document is intended to act as a technical tool for software engineers involved in the development of this project. For better understanding of our application, the knowledge required is:

* UML – Unified Modeling Language
* OOD – Object-oriented Design
* C# programming language
* .NET Remoting

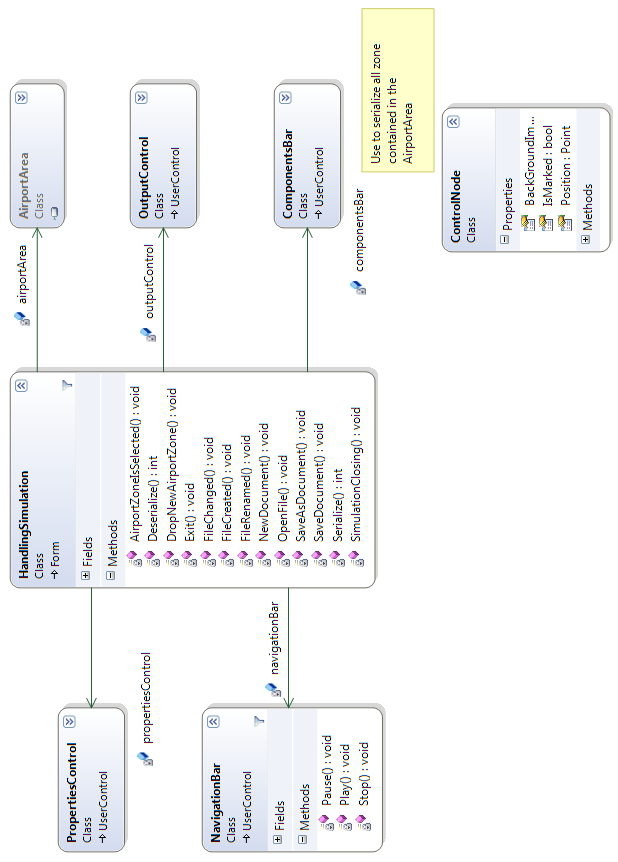
# Project Design

## Class Diagrams

Airport Library:



Handling Simulation:



## Classes and Methods

This chapter describes all the classes and methods which are included in our class diagrams. The Airport Library class diagram shows the classes of the components and methods that the client needs to create a simulation while the Handling Simulation class diagram is intended to explain the methods to handle the simulations such as making a new simulation, saving a simulation and etc.

### Airport Library Class Diagram

**AirportZone** - This class is one the server side and is to take charge of creating the simulation interface of our application. There are 4 classes inherited from it which are DestinationGate, Storage, CheckIn and Sorter. In this class, the method OnPaint with a return type of void is to paint all the components on the interface.

**AirportArea** - This class is on the server side and is to take charge of the activities on the airport zone, which is also medium of painting conveyors and transferring the image on the airport zone. All methods in this class are of type void except method FindAirportZone. CreateTable and ResizeTalbe are functions used to create a new simulation interface and resize the interface area which have no parameters. PopulateTable is to put the new table in the table list in the system, which has a parameter namely List<AirportZone> zones while SetEventsToZone is a function to set all the events into the new interface area like DragDrop, DragEnter, MouseDown, MouseMove with a parameter called AirportZone zone. FindAirportZone has a function of locating where the mouse is pointing at in the interface area and it has a return type of void.

**PaintPanel** - This class is on the server side and is to add conveyors on the airport zone. It connects to the AirportZone through AirportArea. It has one method named AddConveyor of type void. Obviously, it is used to add a conveyor by adding the start point and the end point of the conveyor.

**DestinationGate** - Inherited from AirportZone, this is a class used to add a destination gate on the airport area and connected to the class Flight and Storage. Each destination gate has storage.

**Flight** - This class is in charge of the flight information including which gate the plane will be waiting and the departure time of every flight.

**Storage** - This class is in charge of the storage information which is also related to Class Luggage. It contains the luggage waiting for its flight and the capacity.

**Luggage** - This class takes charge of all luggage with the plane/destination gate it goes to.

**CheckIn** - This class is in charge of the check in gates. The luggage enters the simulation from the check in gates and the luggage information will be generated.

**Sorter** - This class is to take charge of all the sorters in the simulation. It contains the algorithms and methods to identify the luggage ID and configure the route for the luggage to go to its right destination gate/storage.

### Handling Simulation Class Diagram

**HandlingSimulation** - This class is main class in this diagram, which is in charge of the user interface for the client to manage the simulation such as Open a simulation, Save a simulation and etc. It is on the client side, containing all the methods that the user could do with the simulations. There are 2 methods with a int return type while the others are with a void return type, and all the methods in this class has no parameter. Method AirportZoneIsSelected is a function to check whether a simulation interface is selected or not. DropNewAirportZone is a function used to create a new (empty) simulation interface for the system after the user calls the method NewDocument, which is creating a new simulation. Likewise, SimulationClosing is for the system to close the simulation interface after the user calls the method Exit. OpenFile is a function for the user to open an existing simulation while SaveDocument and SaveAsDocument is to save a simulation file.FileCreated, FileChanged and FileRenamed are methods that executed after the SaveDocument and SaveAsDocument methods are called. The only 2 methods with a int type are Serialize and Deserialze. These 2 methods are used to serialize the position of components on the airport zone by the point on the interface.

**NavigationBar** - This class is used for the user to navigate the simulation. It is on the client side. NavigationBar contains 3 methods: Pause, Play, Stop so the user can start, pause or stop the simulation. The methods here are of type void and have no parameter. The methods are created as buttons of Actions on the interface.

**PropertiesControl** - This class is in charge of editing the properties of components in the simulation. The methods are used to modify the properties.

**ComponentsBar** - This class takes charge of adding components to create a simulation. It contains methods to drag the components into the zone and move the components inside the zone.

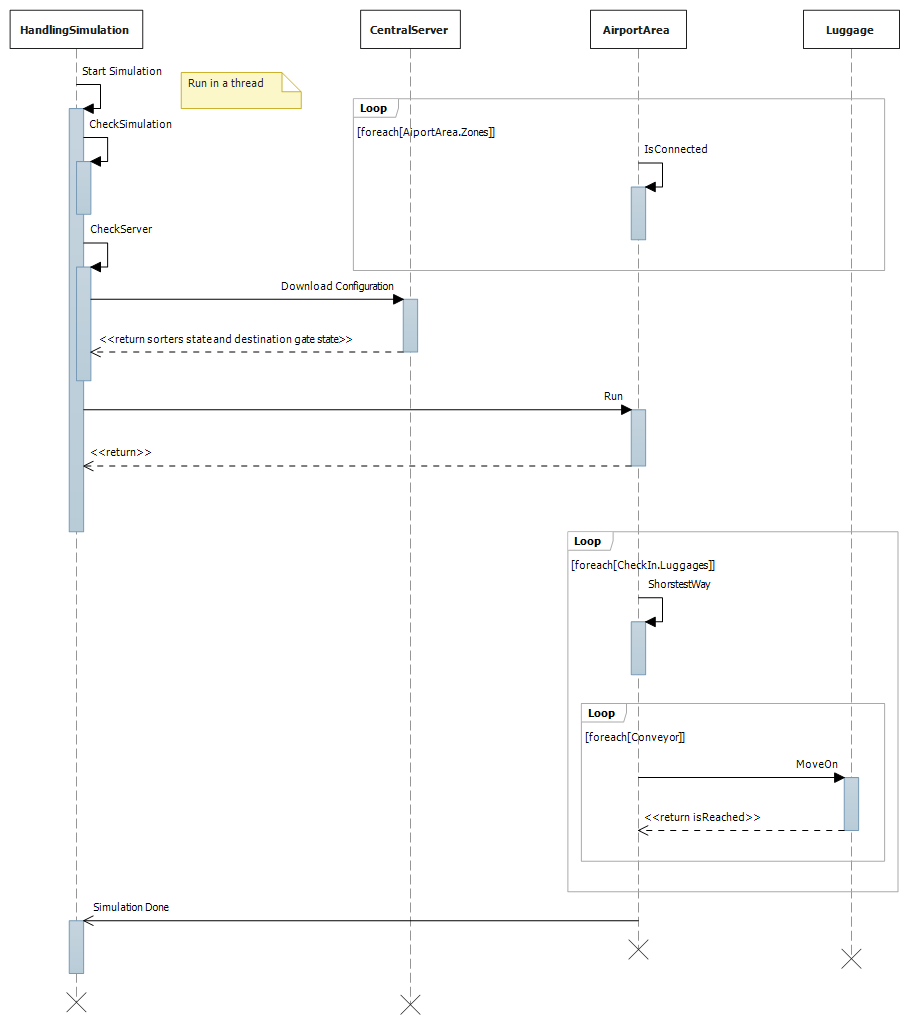
**OutputControl** - This class is in charge of checking whether a simulation is complete. It includes methods which check if there are conveyors between 2 different objects and whether the simulation is able to run. The errors will be shown in the list to remind users to modify the simulation.

**AirportArea** - This class has been introduced in the previous class diagram.

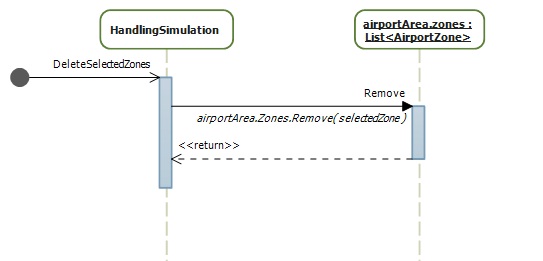
**ControlNode** - This is a class used to serialize all zone contained in the AirportArea.

## Sequence Diagrams

**Start Simulation**



**Remove Object**



First, the user clicks on the component he wants to remove. Then, he clicks on the red cross of the toolbox. The consequence of this action is simply to call the Remove method of the AirportZone list, which is a member of the AirportArea.

**Add Component**

