|  |
| --- |
| Fontys University of applied science |
| User Requirements Specification |
| Parcel Handling Simulation |
|  |
| **GDS - Group 3** |
| **9/7/2010** |

|  |  |  |
| --- | --- | --- |
| Document name |  | User Requirements Specification |
| 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 |

**Approval Signatures**

|  |
| --- |
| Approved by: School tutor  Date: |
| Approved by: Client  Date: |

**Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Related pages** | **Brief Description of Change** |
| 1.0 | 2010-09-01 | All |  |
| 2.0 | 2010-09-07 |  | Corrections made in Last meeting with Tutor |
|  |  |  |  |
|  |  |  |  |

**Review and Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Review** | | | |
| **Version** | **Date** | **Reviewed by** | **Position/Role** |
| 1.0 | 2010-09-06 | Peter Boots | School tutor |
| 1.0 | 2010-09-06 | Casper Schellekens | Client |
| 2.0 |  | Peter Boots | School tutor |
| 2.0 |  | Casper Schellekens | Client |

|  |  |  |  |
| --- | --- | --- | --- |
| **Approval** | | | |
| **Version** | **Date** | **Approved by** | **Position/Role** |
|  |  | Peter Boots | School tutor |
|  |  | Casper Schellekens | Client |

**Table of Contents**

[1. Introduction 1](#_Toc271657164)

[1.1 Purpose 1](#_Toc271657165)

[1.2 Assignment: Parcel handling simulation 1](#_Toc271657166)

[1.3 Users 1](#_Toc271657167)

[1.4 Cyclic Development of Application 1](#_Toc271657168)

[1.5 Components 2](#_Toc271657169)

[1.6 Non Functional Requirements 3](#_Toc271657170)

[2. Use Cases 4](#_Toc271657171)

[3. User Interface 12](#_Toc271657172)

[3.1 Simulation Overview 12](#_Toc271657173)

[3.2 Simulation Working Overview 12](#_Toc271657174)

[4. MoSCoW Matrix 13](#_Toc271657175)

# Introduction

## Purpose

This document represents the analysis of functional and user requirements of the Parcel Handling Simulation application that we intend to develop for the client. We will here in state some of the basic requirements that would be made available in the first edition of the application.

## Assignment: Parcel handling simulation

The parcel handling simulation is a distributed application that is intended to resemble the baggage handling systems at the airports. It includes simulation of the conveyors used to transport the parcels or baggage, the inputs, which are the check-in or baggage drop-off desks at the airport, sorters that route the items to their destination and the outputs, which are the destination gates of the parcels.

The user will be able to build the conveyors by drawing lines on the working area. Each parcel will have a set of information within itself such as destination, ID, priority based on urgency of delivery, etc., which would allow the sorters to navigate the parcel to its destination.

Finally, additional features as parcel dimension diversity, storage facility, belt speed etc. can be added to the simulation.

## Users

The users of the fully developed application would be the client.

## Cyclic Development of Application

The way our team will build this application is through the Iterative Application Development (IAD) approach. This means that in developing this application, we will have to repeat certain stages (design, and development) in a cyclic manner.

There would be 3 cycles of development, these would be preceded with the project plan and design phase clarifying the exact technical nature of the application.

Each cycle would involve the addition of functional elements to the application i.e. we would have the realization of the full application with certain functions working in a cycle and then we would add more functional elements to the application in the next IAD. As stated above the idea would be to use 3 IAD’s to achieve our target of a fully functional package handling simulation application.

Further details on the exact constituents of the various phases are in the Project Plan.

## Components

This is a short description of each of the components from which the user can create the simulation layout.

The user can add and remove all of the following elements in the simulation with the exception of the actual payload - the bags. Those are generated automatically by the check-in desks.

**Sorter:**

Object in application that has a function of getting package information and guiding the package further to its correct destination point

**Check-in gate:**

Application component with function of getting the end destination information and other such needed information for use in handling package. Entry point for all packages will enter into the simulation.

**Destination gate:**

Point of exit of packages from the simulation, sorter uses the position of destination gates to determine direction of packages.

**Parcels:**

Objects being moved from check-in gate to destination gate. The parcels when checked in have values (randomly created) that determine its destination gate or its place of storage. It could also have emergency and/ or weight information. These are the actual payload of the system. On their creation by the check-in desks they are assigned a destination gate, after which they are routed directly to it by the sorters.

**Conveyor segment (straight):**

This component is available in 4 varieties - horizontal, vertical and two diagonals. It connects all other components together into a logically sound baggage handling system.

**Conveyor segment (90 degree turn):**

This component serves the same function as the straight conveyor and allows the user to build more topologically complex conveyors. There are 4 varieties - the four rotations of the base element depending on the direction of movement.

**Conveyor segment (45 degree turn):**

This component facilitates inclusion of sorters in the conveyor system. Again, it has 4 rotations.

## Non Functional Requirements

1. Interface requirements:

- The system must be intuitive. The most common functions should be one click away.

- The user interface should be responsive. It should immediately acknowledge the user input.

- English is used as the language.

2. Operating requirements:

- The system should be a turn-key solution, not requiring installation or additional configuration before it can be used, as much as its distributed nature allows that.

- The application should be able to work on the window OS platform.

3. Performance requirements:

- The system should not limit the number of elements in the simulation.

- The application would be able to work as a distributed application.

- The system should strike an acceptable balance between smoothness of the simulation (refresh rate) and network utilization.

- Visualization of packages in movements is necessary.

4. Lifecycle requirements:

- The development time specified in the project phasing should not be exceeded.

- The system should require zero maintenance.

# Use Cases

|  |  |
| --- | --- |
| ID | **1** |
| Name | **New simulation** |
| Goal | To create a new simulation file |
| Pre-condition | The client program is open  The simulation is not running |
| Main  Success  Scenario | 1. The user selects File->New or the “New simulation…” button on the toolbar 2. The system creates a new empty workspace |
| Extensions | 1a. A simulation file is already open and there are unsaved changes  1. The system presents the user with the choices: Save, Don’t Save, Cancel  2. The user selects one of the three and the system responds accordingly:   |  |  | | --- | --- | | Save | * 1. The system goes to "Save" use case   2. The system returns to MSS, step 2 | | Don't save | * 1. The system closes the current simulation, discarding unsaved changes   2. The system returns to MSS, step 2 | | Cancel (or close button) | The system exits the use case | |
| Post-condition | A new empty workspace is created |
| Author | Kristian Kolev |

|  |  |
| --- | --- |
| ID | **2** |
| Name | **Save As…** |
| Goal | To save the currently open simulation under a user-chosen name and location |
| Pre-condition | The client program is open  A simulation is open |
| Main  Success  Scenario | 1. The user selects File->Save As… or the “Save As…” button on the toolbar 2. The system prompts the user to enter the directory and file name under which to save the simulation 3. The user enters the information 4. The system saves the simulation file |
| Extensions | 1a. The file already exists   1. The system prompts the user if they wish to overwrite the existing file 2. The user makes a choice and the system responds accordingly:  |  |  | | --- | --- | | Yes | The system overwrites the existing file | | No | The system goes back to MSS, step 2 | |
| Post-condition | All changes to the simulation file are saved |
| Author | Kristian Kolev |

|  |  |
| --- | --- |
| ID | **3** |
| Name | **Save** |
| Goal | To save the currently open simulation |
| Pre-condition | The client program is open  A simulation is open |
| Main  Success  Scenario | 1. The user selects File->Save or the “Save simulation…” button on the toolbar 2. The system saves all changes to the simulation file |
| Extensions | 1a. The user has not previously saved the current simulation  1. The system goes to the “Save as…” use case, MSS, step 2 |
| Post-condition | All changes to the simulation file are saved |
| Author | Kristian Kolev |
| ID | **4** |
| Name | **Exit** |
| Goal | To exit the client program |
| Pre-condition | The client program is open |
| Main  Success  Scenario | 1. The user selects File->Exit or the Close button 2. The system exits |
| Extensions | 1a. A simulation with unsaved changes is open  1. The system goes to the “Save” use case, MSS, step 2 |
| Post-condition | The client program exits successfully |
| Author | Kristian Kolev |

|  |  |
| --- | --- |
| ID | **5** |
| Name | **Start Simulation** |
| Goal | To start the simulation |
| Pre-condition | The program is open  Simulation mode is launched |
| Main  Success  Scenario | 1. The user click on start button 2. Simulation starts |
| Extensions |  |
| Post-condition | Simulation is running |
| Author | Antoine Girard |

|  |  |
| --- | --- |
| ID | **6** |
| Name | **Pause Simulation** |
| Goal | To pause the simulation |
| Pre-condition | The program is open  Simulation mode is launched  Simulation is currently running |
| Main  Success  Scenario | 1. The user click on pause button 2. Simulation pauses |
| Extensions |  |
| Post-condition | Simulation paused |
| Author | Antoine Girard |

|  |  |
| --- | --- |
| ID | **7** |
| Name | **Stop Simulation** |
| Goal | To stop the simulation |
| Pre-condition | The program is open  Simulation mode is launched  Simulation is currently running or paused |
| Main  Success  Scenario | 1. The user click on stop button 2. Simulation stops |
| Extensions |  |
| Post-condition | Simulation stopped |
| Author | Antoine Girard |

|  |  |
| --- | --- |
| ID | **8** |
| Name | **Add check-in gate** |
| Goal | To add a check-in gate to the editor |
| Pre-condition | The application is in editor state |
| Main  Success  Scenario | 1. The user selects the check-in gate icon 2. User puts the check-in gate icon on the workspace 3. Application adds check-in gate icon to the workspace |
| Extensions | 1a. User doesn’t select the check-in gate icon correctly  Nothing happens in the workspace.  Go to MSS step 1  2a. User puts check-in gate icon outside the workspace  Nothing happens in the workspace.  Go to MSS step 1 |
| Post-condition | The check-in gate has been added |
| Author | Ibeagha Ginika |

|  |  |
| --- | --- |
| ID | **9** |
| Name | **Add destination gate** |
| Goal | To add a check-out gate to the editor |
| Pre-condition | The application is in editor state |
| Main  Success  Scenario | 1. The user selects the check-out gate icon 2. User puts the check-out gate icon on the workspace 3. Application adds check-out gate icon to the workspace |
| Extensions | 1a. User doesn’t select the check-out gate icon correctly  Nothing happens in the workspace.  Go to MSS step 1  2a. User puts check-out gate icon outside the workspace  Nothing happens in the workspace.  Go to MSS step 1 |
| Post-condition | The check-out gate has been added |
| Author | Ibeagha Ginika |

|  |  |
| --- | --- |
| ID | **10** |
| Name | **Add Parcel** |
| Goal | To add a Parcel to the simulation |
| Pre-condition | The application is running in simulation state |
| Main  Success  Scenario | 1. The user selects the Add Parcel icon 2. System gives a form to fill in Parcel destination, ID number, check-in gate and/or priority 3. When form is filled user accepts choices filled 4. Parcel is added |
| Extensions | 2a. User doesn’t fill the parcel properties form correctly  System gives message that parcel properties form should be correctly filled  Go to MSS step 2 |
| Post-condition | A parcel has been added |
| Author | Ibeagha Ginika |

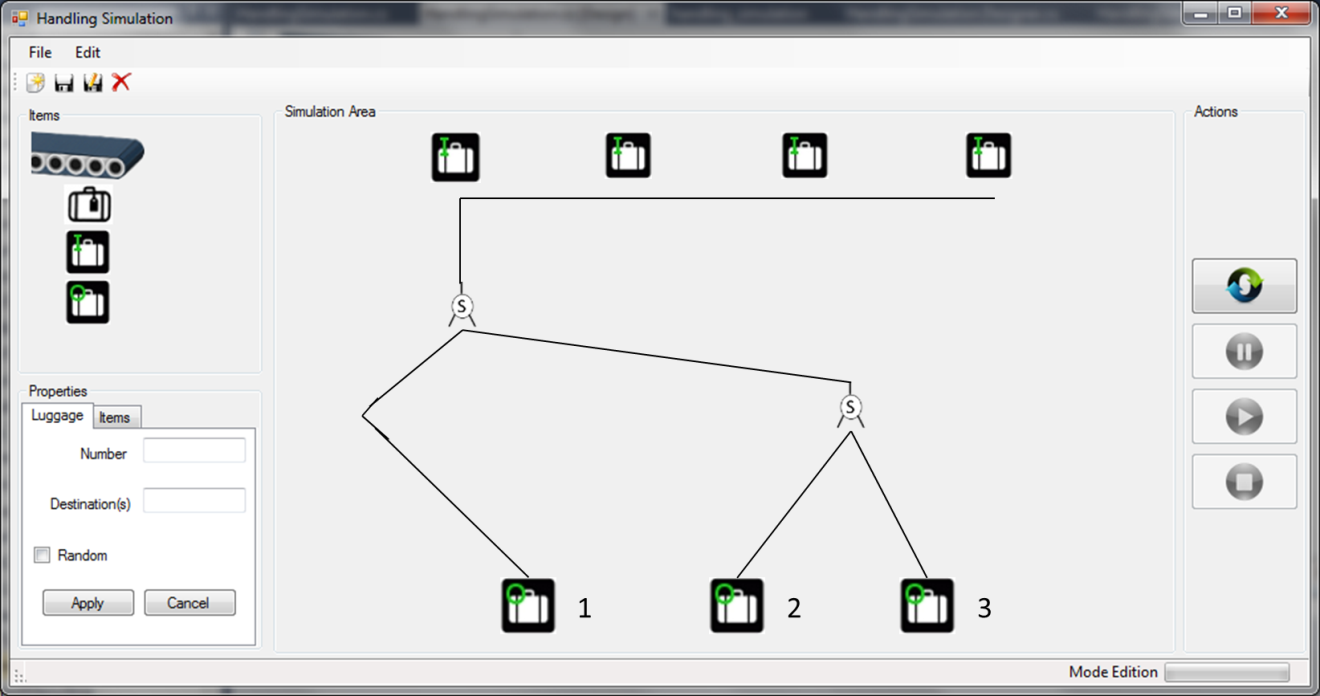
|  |  |
| --- | --- |
| ID | **11** |
| Name | **Remove Object** |
| Goal | In case where the user want to remove an added item in the simulation. |
| Pre-condition | The simulation is not started |
| Main  Success  Scenario | The selected item doesn’t displayed anymore |
| Extensions |  |
| Post-condition |  |
| Author | Sébastien Lepage |

|  |  |
| --- | --- |
| ID | **12** |
| Name | **Clear all items** |
| Goal | In case where the user wants to reinitialize the simulation and make a new simulation with a different number and type of items. |
| Pre-condition | The simulation is not started |
| Main  Success  Scenario | 1. Click on the Edit tab 2. Select “Clear all items”   The number of items is 0. |
| Extensions |  |
| Post-condition | The user has to create a new scenario with new items. |
| Author | Sébastien Lepage |

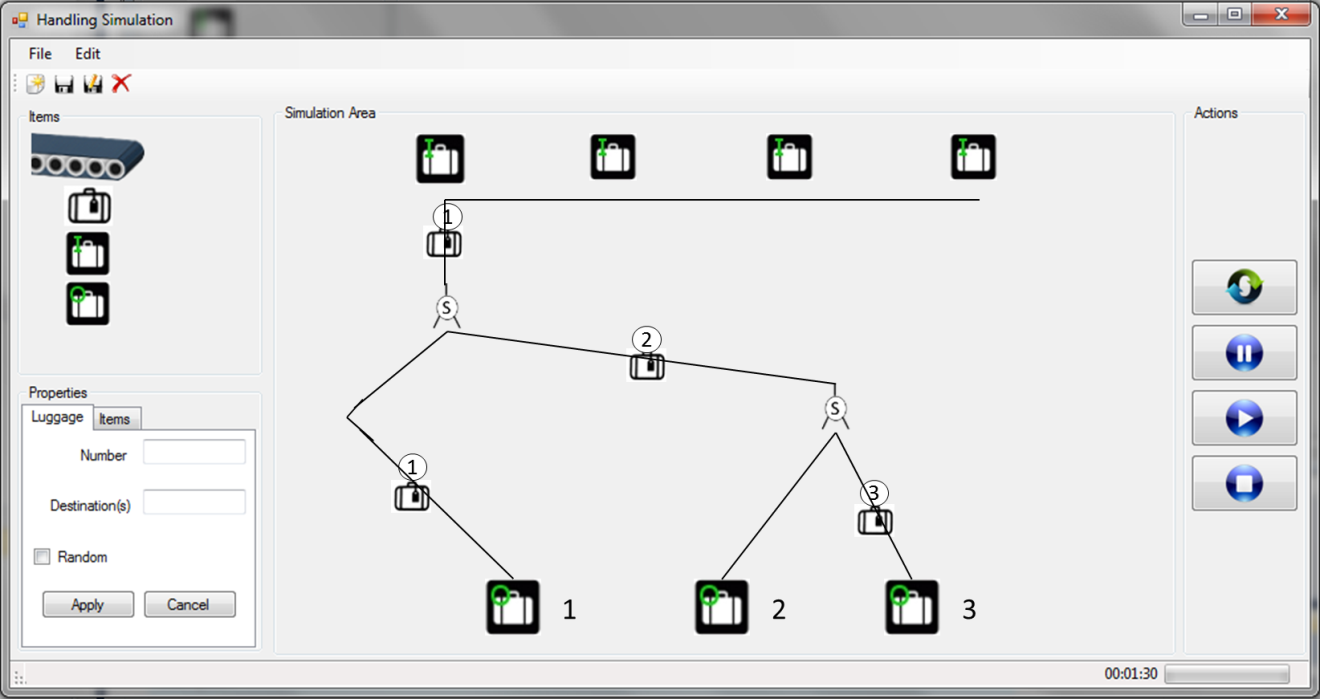
|  |  |
| --- | --- |
| ID | **13** |
| Name | **Set Speed** |
| Goal | Define the applications’ speed |
| Pre-condition | Application is in edit state |
| Main  Success  Scenario | 1. Click on the Set Speed button 2. Increase or decrease speed |
| Extensions |  |
| Post-condition | The speed is updated for the next simulation. |
| Author | Sébastien Lepage |

# User Interface

## Simulation Overview



## Simulation Working Overview



# MoSCoW Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Functions | M | S | C | W |
| Distributed System Functionality |  |  |  |  |
| New Simulation |  |  |  |  |
| Save |  |  |  |  |
| Save as |  |  |  |  |
| Exit |  |  |  |  |
| Start Simulation |  |  |  |  |
| Stop Simulation |  |  |  |  |
| Pause Simulation |  |  |  |  |
| Add check-in gate |  |  |  |  |
| Add destination gate |  |  |  |  |
| Add parcel |  |  |  |  |
| Remove object |  |  |  |  |
| Clear all items |  |  |  |  |
| Set speed |  |  |  |  |

\* **M** - MUST have this.

\* **S** - SHOULD have this if at all possible.

\* **C** - COULD have this if it does not affect anything else.

\* **W** - WON'T have this time but WOULD like in the future