Preliminary Software Requirements for Elevator Simulator

Lab5 Part 3

Cheryl Huber

# 1.0. Introduction

## 1.1. Purpose

The purpose of this document is to present a preliminary detailed description of Elevator Simulation. It will explain the purpose and features of the simulation, the interfaces, what it will do, and the constraints under which it must operate. This document is intended for Elevator company and will be proposed to their elevator design team so that they may better analyze the minimization of waiting time for people using elevators.

***1.2. Scope of Project***

Elevator Simulation is a software system designed to simulate elevators in multiple buildings. It is intended for the use of elevator designers so that may test their algorithms for passenger pick up. The user can create buildings with elevators and floors. Once these have been created, the user can then create passengers that want to go up or down and what their destination floor is.

## 1.3. Glossary

|  |  |
| --- | --- |
| Term | Definition |
| User | Reviewer/Elevator designer |
| Building | A building that will contain an elevator that needs to be evaluated. |
| Elevator | An elevator that is part of a building and is used to carry passengers to various destinations. |
| Floors | A floor is the desired location within a building that a passenger would like to get to. |
| Passenger | A person that is in a building on a particular floor and would like to go to another floor. |
| Stakeholder | Any person with an interest in the project who is not a developer. |
| Destination | The ultimate floor that a passenger would like to go to. |

## 1.4. References

## IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

## http://en.wikipedia.org/wiki/Software\_requirements\_specification

## 1.5. Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

# 2.0. Overall Description

## 2.1 System Environment

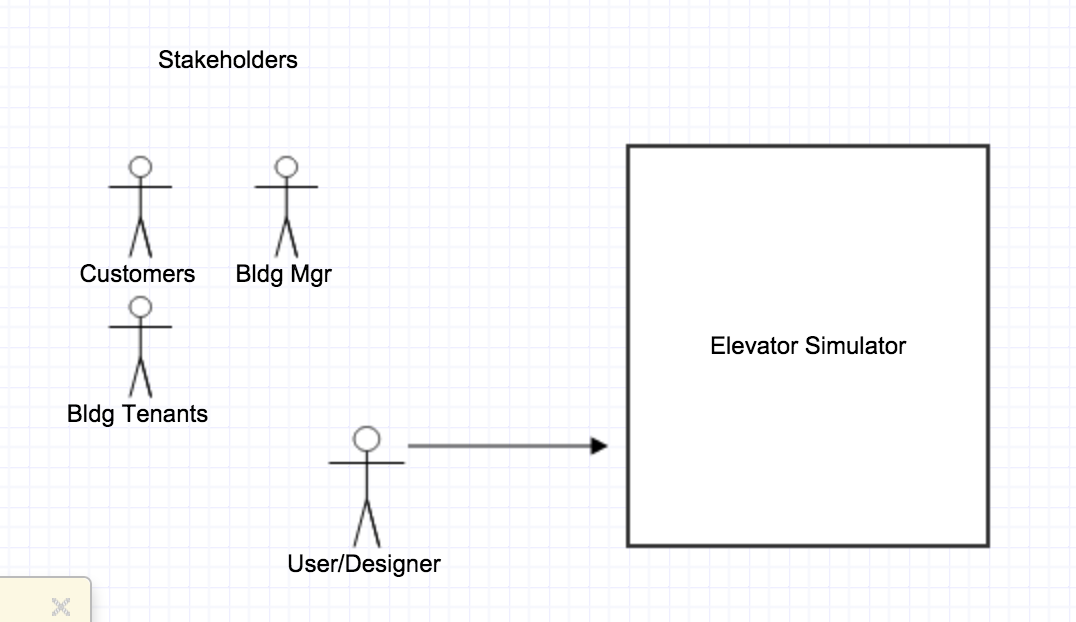


Figure - System Environment

The Elevator Simulator has one active actor and one cooperating system.

The stakeholders are identified as the user/designer, building tenants, building managers and building customers. The user/designer interacts with the Elevator Simulator in order to simulate an elevator while designing his elevator algorithm. The designer’s goal is to create a design that minimizes wait times for all passengers. Once the designer has created his algorithm, then it is implemented in the elevator of the building and the stakeholders will experience the design.

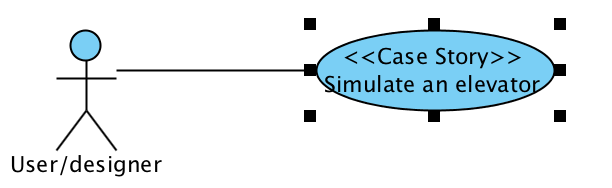
## 2.2 Functional Requirements Specification

This section outlines the use cases for each of the active readers separately. The reader, the author and the reviewer have only one use case apiece while the editor is main actor in this system.

### 2.2.1 User/Designer Use Case

#### Use case: Simulate Elevator for testing

**Diagram:**

****

**Brief Description**

The user/designer accesses the Elevator Simulator and uses it to run test cases and design an elevator that minimizes wait times.

**Initial Step-By-Step Description**

Before this use case can be initiated, the user/designer has already accessed the Elevator Simulator.

1. The user/designer creates a controller.
2. The system creates an elevator and floor model.
3. The user/designer declares an elevator and floor object and assigns it to the controller’s models.
4. The user/designer boards passengers onto the elevator.
5. The user/designer decides how many times the elevator should change floors.
6. The system provides a GUI representation of the elevator running.

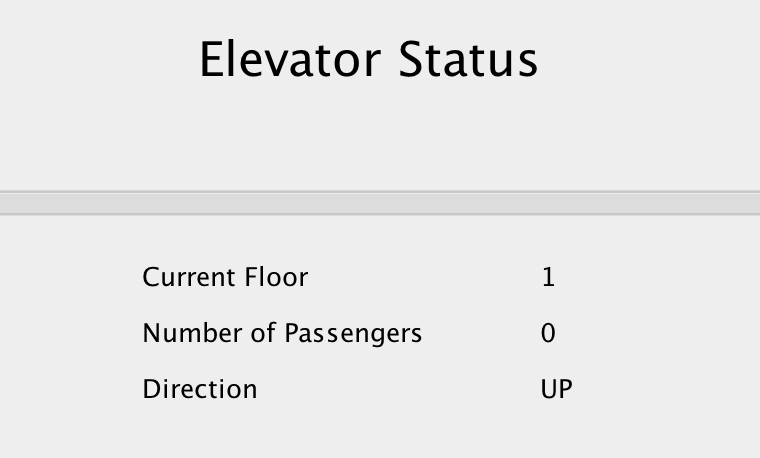


Figure – GUI that displays simulation

**Xref:** Section 3.2.1, User/Designer Use Case

## 2.3 User Characteristics

The user/designer is expected to learn and familiarize himself with the user interface. He will have to know how to board passengers and specify the destination floor. He will also need to determine the number of floors in the building and set the number of floor changes he wants to simulate.

## 2.4 Non-Functional Requirements

The application will need to run on a computer that can run a java application. The user’s screen must be large enough to display the designed GUI. The user/designer must be designing for a building that only has one elevator.

# 3.0. Requirements Specification

## 3.1 External Interface Requirements

There are no external interface requirements

## 3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

### 3.2.1 User/Designer use Case

|  |  |
| --- | --- |
| **Use Case Name** | User/Designer Use Case |
| **XRef** | Section 2.2.1, Search Article |
| **Trigger** | The user/designer opens the simulator |
| **Precondition** | Java Virtual Machine is installed on user/designer’s computer |
| **Basic Path** | 1. The user/designer creates a controller. 2. The system creates an elevator and floor model. 3. The user/designer declares an elevator and floor object and assigns it to the controller’s models. 4. The user/designer boards passengers onto the elevator. 5. The user/designer decides how many times the elevator should change floors. 6. The system provides a GUI representation of the elevator running. |
| **Alternative Paths** | 1. After testing, the user can choose to change the number of floors. 2. Different passenger boarding data can be created. 3. Another elevator for another building can be created. |
| **Postcondition** | The user/designer has fully simulated his test cases. |
| **Exception Paths** | A passenger may not board the elevator with a path that is outside of the bounds of the floors. |
| **Other** | The buildings can only have one elevator. |

## 3.3 Detailed Non-Functional Requirements

### 3.3.1 Logical Structure of the Data in a Class Diagram

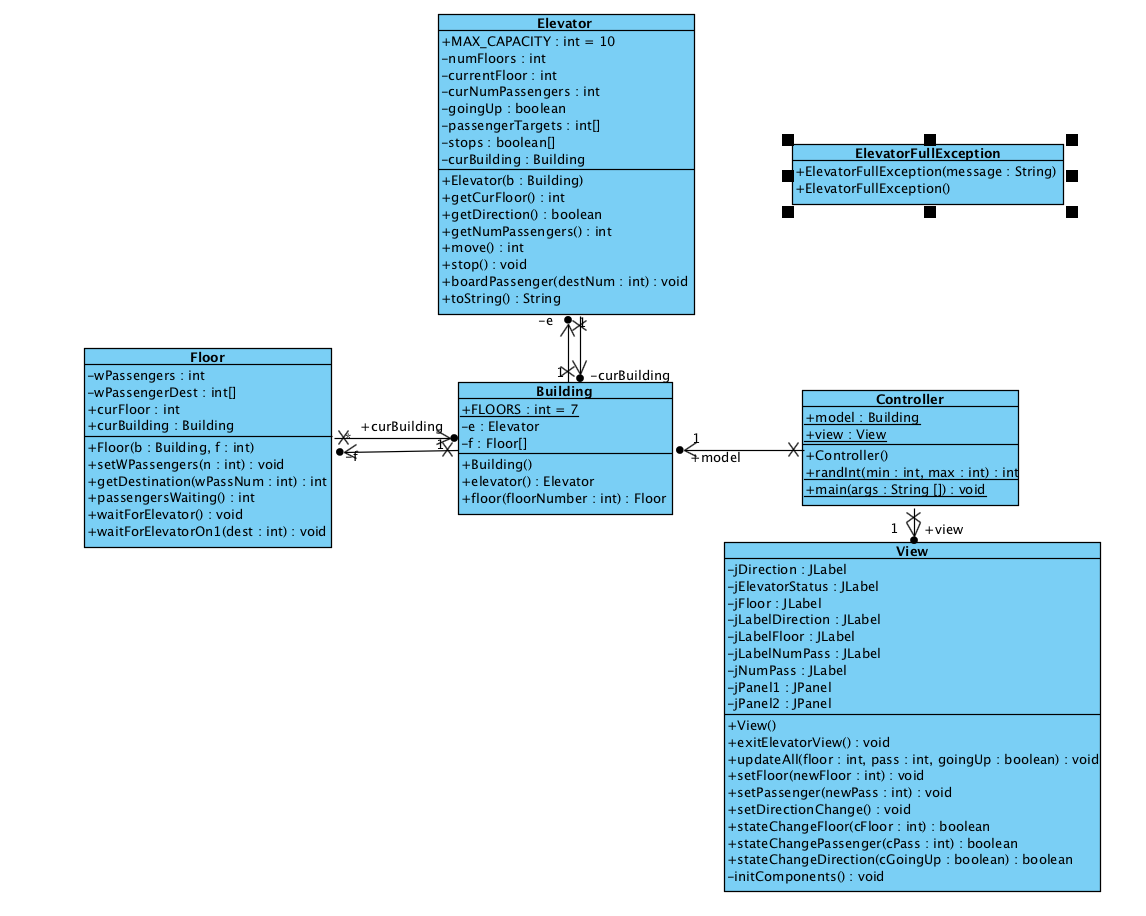


Figure - Logical Structure of the Elevator Simulator

The data descriptions of each of these data entities is as follows:

**Building Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| e | Elevator | The building’s elevator |  |
| f | Floor | Current floor |  |
| FLOORS | Integer | Total # floors |  |

**Floor Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| wPassengers | Integer | # of passengers waiting |  |
| wPassengerDest | Integer | Corresponding destination of a waiting passenger | Array |
| curFloor | Integer | Current floor |  |
| curBuilding | Building | Building that the floor is a part of |  |

**Controller Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Model | Building | Building that is created at instantiation of controller |  |
| View | View | GUI view |  |

### 3.3.2 Security

This is a stand alone simulator and there are no security issues.