Requirement

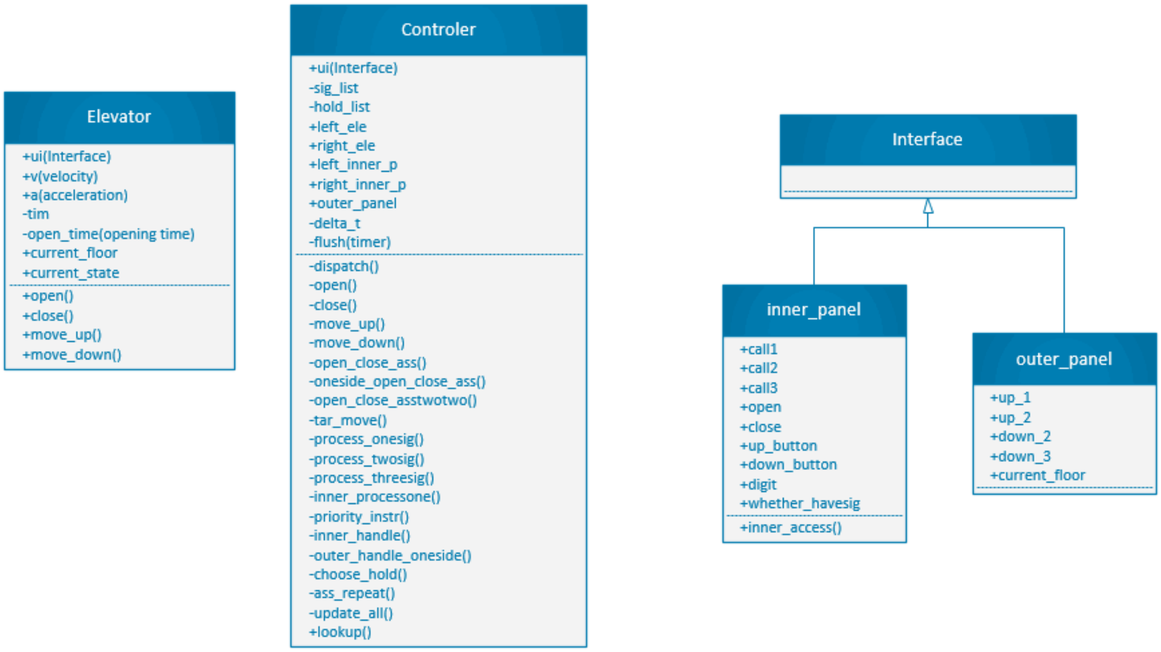
System Objective

In this project, we are developing a software that simulates 2 elevators in a building with 3 floors and design an automated algorithm to schedule and control them according to user requests.

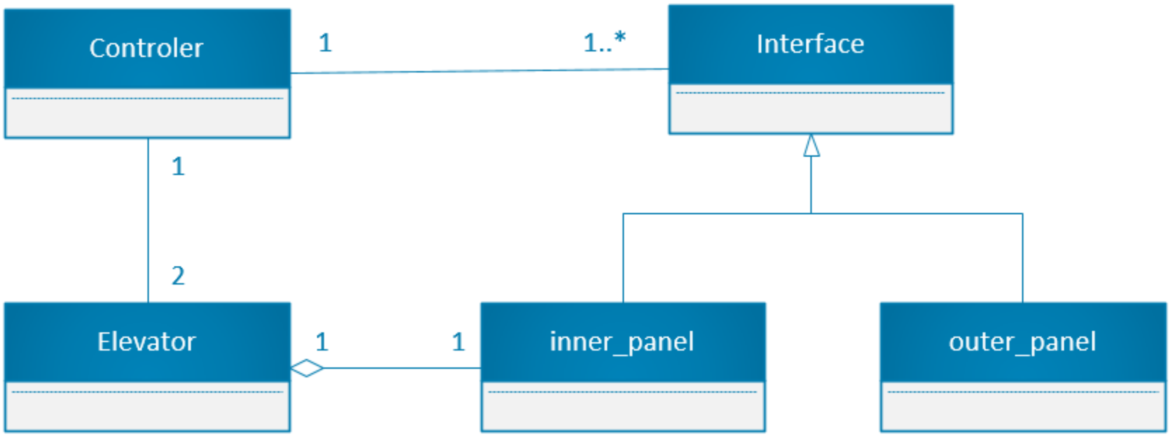
This software offers user the chance to experience double-elevator system inside and outside without waiting reality delay. It's also take care of the graphic interface, display real-time movements of elevators and determine how to process service direction and order of elevators, which ensures every passenger to reach target floor promptly.

Domain Analysis

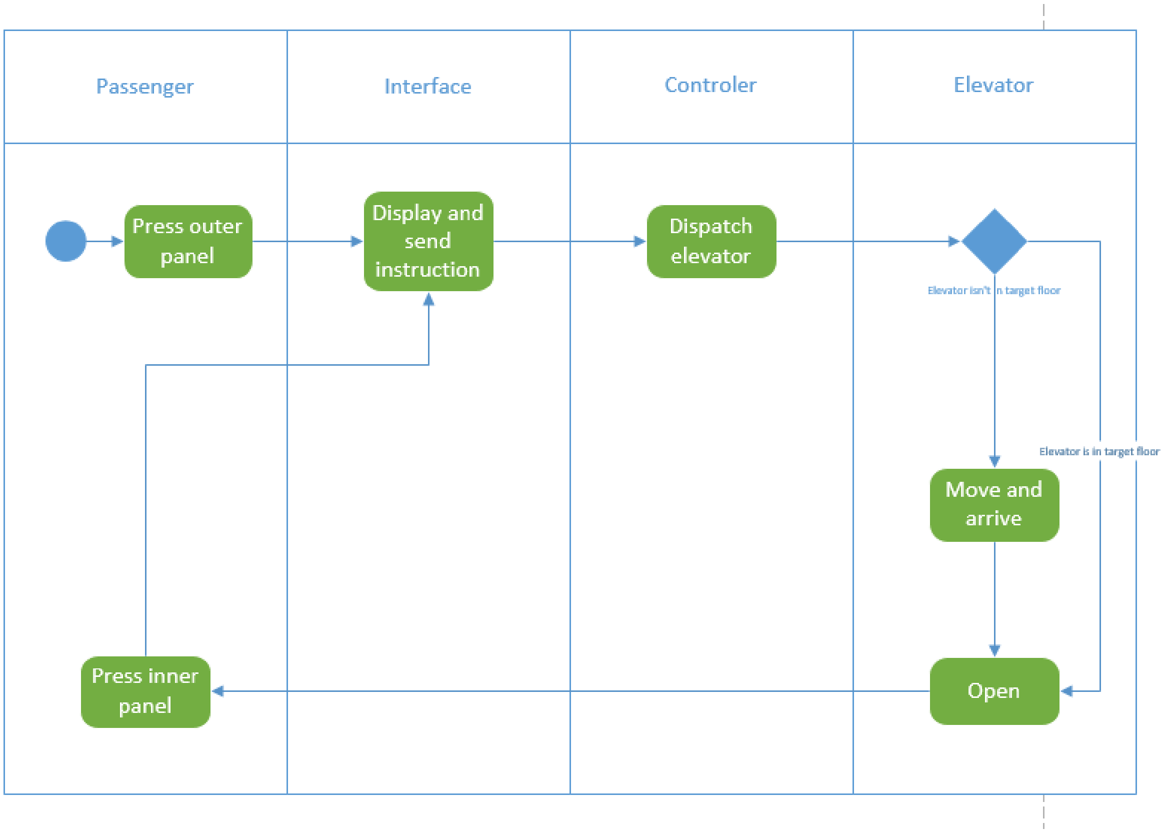
The participants of activities of the system can be categorized into the elevators, a central controller, inner panel and outer panel.



The relationship among different participants are shown as follows:

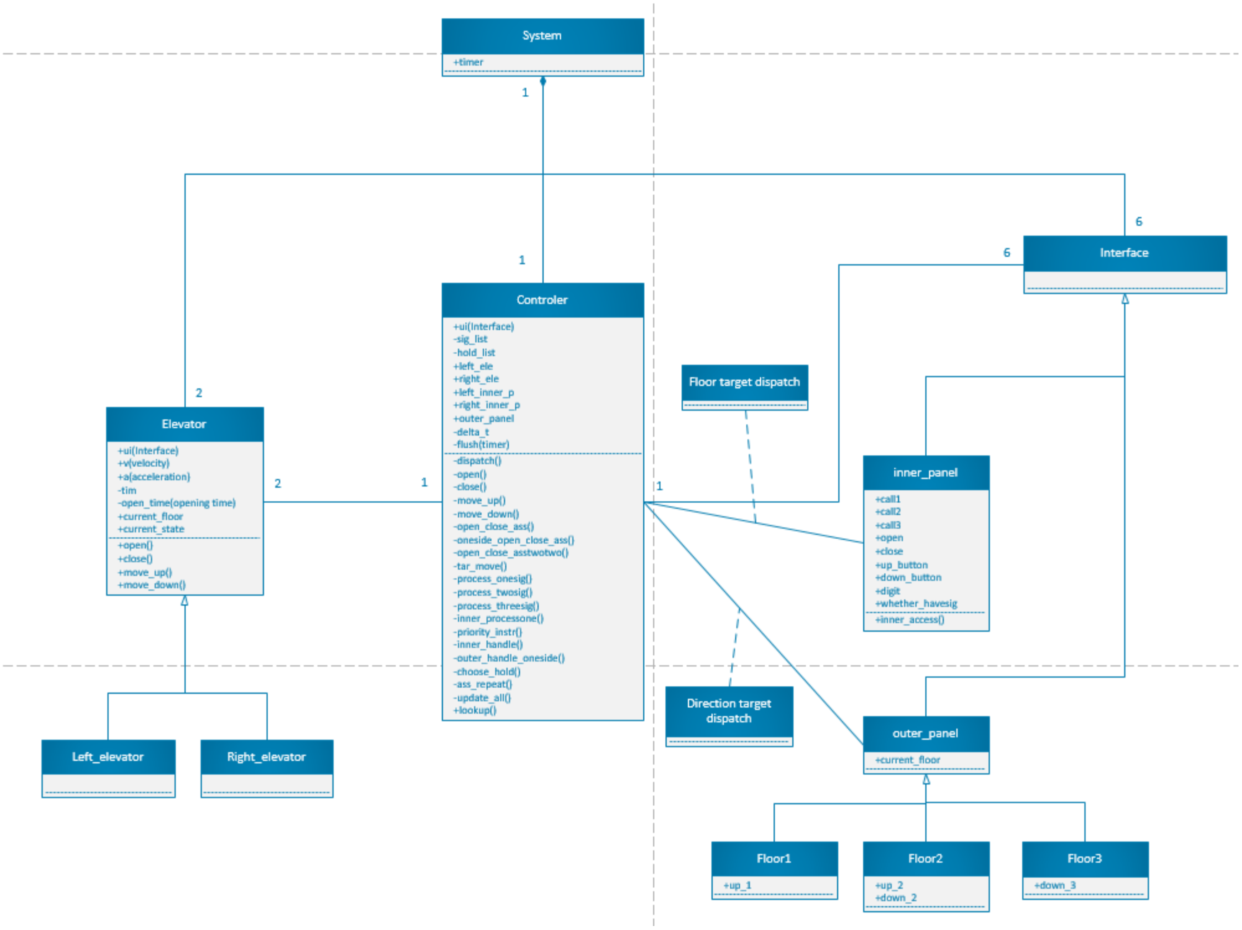


Below is the sequence of events for users use this system:

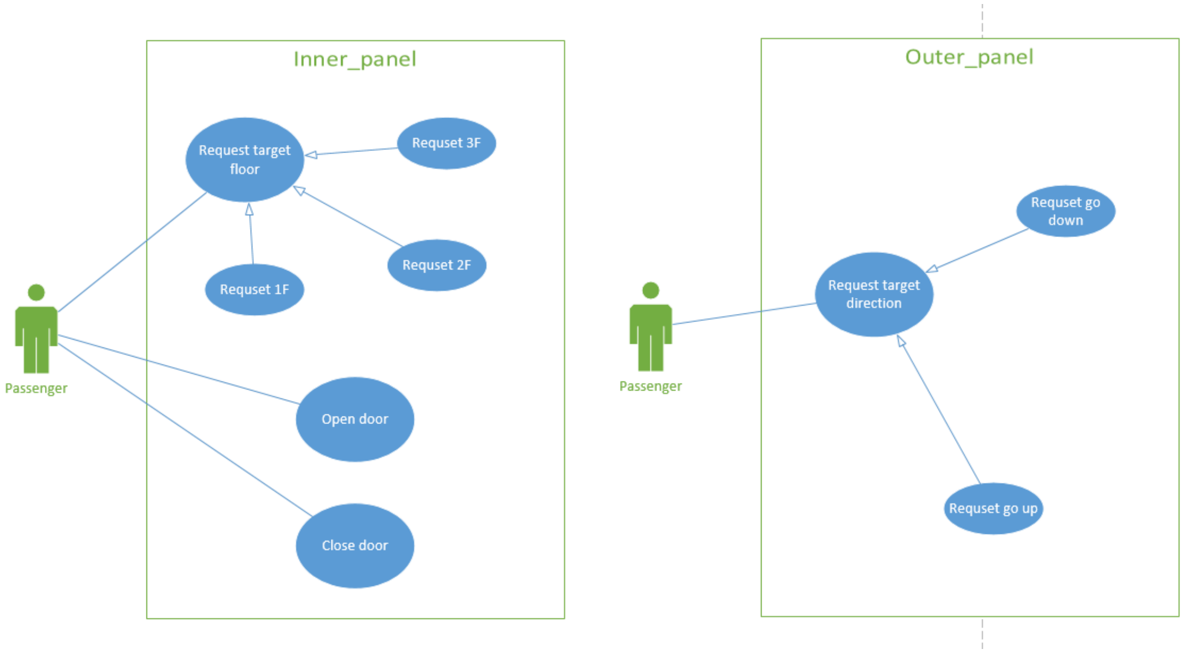


System Architecture

As the class diagram shown below, the whole system architecture is composed of 3 main parts: elevators, user interfaces and the central controller. The controller will link whole system by sending elevator instruction, receive inner and outer panel command with floor target or direction target. The outer panel has three different types according to floor.



User Cases



Software Requirement

R1 Elevator UI

R1.1 User can call an elevator by pushing outer buttons.

R1.1.1 User can call an elevator by pushing first floor "up" buttons.

R1.1.2 User can call an elevator by pushing second floor "up" buttons.

R1.1.3 User can call an elevator by pushing second floor "down" buttons.

R1.1.4 User can call an elevator by pushing third floor "down" buttons.

R1.2 User can control the elevator go to target floor by pushing inner buttons.

R1.2.1 User can control the elevator go to first floor by pushing inner "call_1" buttons.

R1.2.2 User can control the elevator go to second floor by pushing inner "call_2" buttons.

R1.2.3 User can control the elevator go to third floor by pushing inner "call_3" buttons.

R1.3 User can use buttons to control door opening and closing.

R1.3.1 User can use outer buttons to open the door if an elevator is arrive in the floor.

R1.3.2 User can use inner corresponding floor button or open button to open the door if the elevator is arrive in the floor.

R1.3.3 User can use close button to close the door if the elevator reaches the maximum open state at this time.

R1.4 User can see which floor the elevator stays.

R1.4.1 User can see which floor the elevator stays inside an elevator.

R1.4.2 User can see which floor the elevator stays outside an elevator.

R1.5 User can see the elevator moving trace.

R2 Elevator handler

R2.1 When user pushing outer button, there will be a schedule algorithm to choose an elevator responding to the instruction.

R2.1.1 Proximity principle

R2.1.1.1 If two elevators both can be scheduled, choose the one which near to target to respond.

R2.1.2 Current instruction priority principle

R2.1.2.1 If elevator is moving, it shouldn't respond to open/close instruction.

R2.1.2.2 If elevator is moving toward up or down, it shouldn't be interrupted by opposite direction instruction or "has been passed" floor instruction.

R2.1.3 Shortest running distance and energy-saving principle (Additional implement)

R2.2 The elevators should obeys corresponding physical constraints.

R2.2.1 The door should be automatically open as soon as the elevator stopped at the target floor.

R2.2.2 The door should wait 5 seconds when open to Maximum if no other command interrupt.

R2.2.3 The door should be automatically closed if no door-open request interrupts.

R2.2.4 Once the controller receive moving command, the elevator only start after the door is fully closed.

R2.2.5 The elevator height should be no larger than height of 3F, no less than height of 1F.

R2.2.6 The instruction can be canceled by double-click (Additional implement).

R2.2.6.1 For inner panel, the instructions which are not running all can be canceled by double-click.