Good approach, many details still missing.
Let us review some examples in class.

Elevator Control System

Software Requirements Specification (SRS)

SRS Version 3.0

Team #6
13 November 2025

Adrian Faust
Alex Knigge
Bianca Solano
Dustin Ferguson
Jordan Martinez

1 Introduction

1.1 Purpose

This document outlines the software requirements for the control system of a 10-story, four-shaft elevator network. It defines the logical components, their interactions, and the communication required for safe and efficient elevator operation. The purpose of this system is to coordinate elevator movement, manage requests from passengers, ensure door and safety sensor functionality, and maintain reliable operation across multiple modes. The Elevator Control System integrates multiple subsystems including the Passenger call panels, Elevator Cabins, Doors, Supervisor (Control Center), Notifications, and Timer, all to create a unified, automated transportation system.

1.2 Project goals

The primary goal of the Elevator Control System project is to design and implement a safe, reliable, and intelligent control system capable of operating multiple elevators simultaneously within a 10-story building.

Specific goals include:

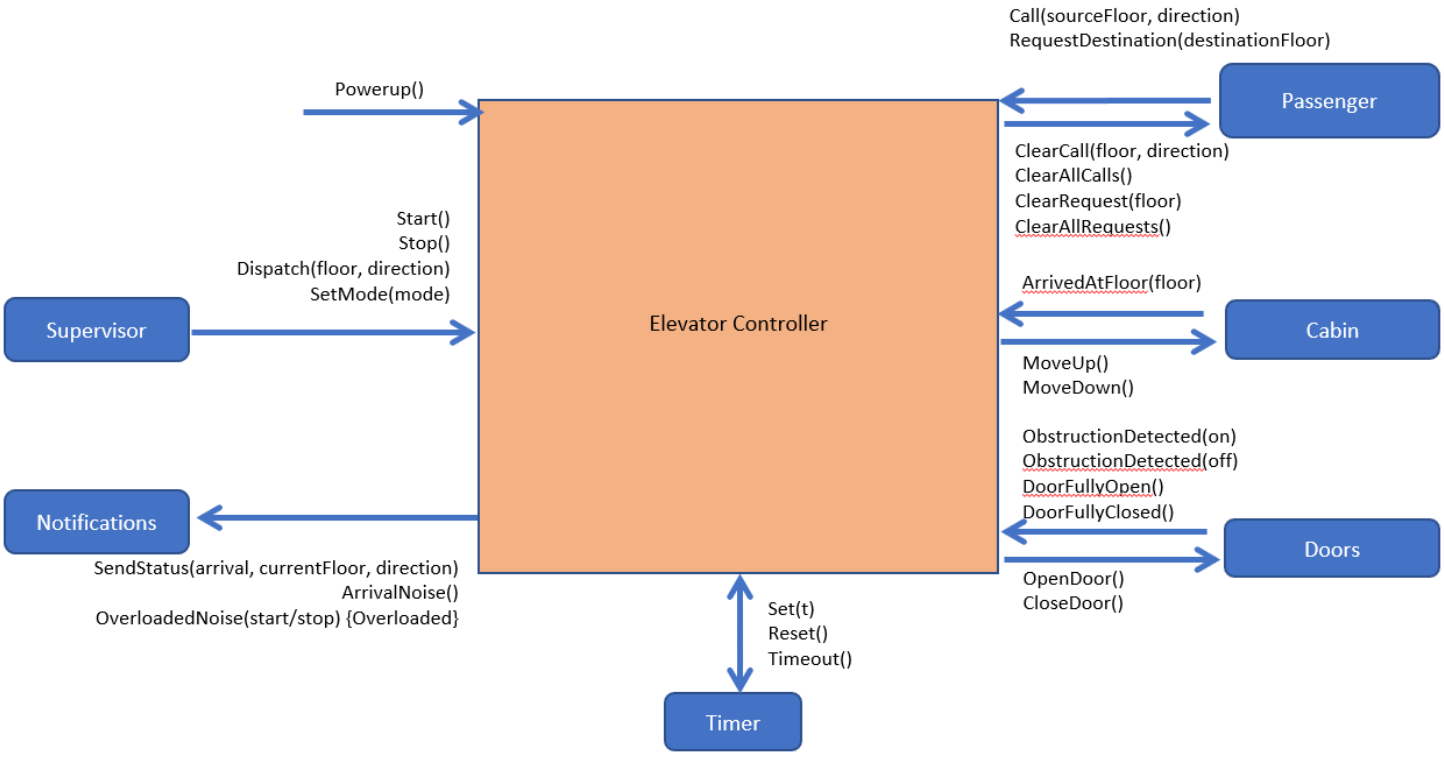
Safety: Maintain passenger safety through door proximity sensors, overload detection, and emergency modes.
Efficiency: Reduce wait times and minimize energy use using an intelligent dispatch algorithm that optimizes elevator movement and load balancing.
Reliability: Ensure continuous operation under varying traffic conditions and modes.
Centralized Control: Enable supervisory monitoring and mode management through a Command Center interface.

1.3 Organization

This document is organized as follows:

1. **Introduction:** The current section, which introduces the document.
2. **Logical Diagram:** Presents an overview of the main logical components of the elevator system and their data flow relationships.
3. **Logical Interfaces:** Defines input and output behaviors for each subsystem, including Passenger, Cabin, Doors, Supervisor, Notifications, and Timer.
4. **System State:** Describes the various operating states of the system (Idle, Moving, Door Open, Emergency, etc.).
5. **State Transitions:** Explains how the system transitions between states in response to events and conditions.
6. **Conclusion:** Summarizes key design goals and project direction.
7. **Definition of Terms:** All acronyms and special definitions (if any) used in this document.

2 Logical Diagram



3 Logical Interface

The **Elevator Controller** is the central unit coordinating all subsystems: ***Passenger, Cabin, Doors, Supervisor, Notifications,*** and ***Timer.*** Each interface defines a **set of input events** (what the controller *receives*) and **output actions** (what the controller *sends*), enabling consistent communication between modules.

3.1 Passenger

The **Passenger interface** handles all user requests from the hallway and cabin buttons.

Input events:

- Call(floor, direction) - Passenger presses up/down button on a floor
- RequestDestinations(floor) - Passenger selects a destination inside the cabin.

Output actions:

- ClearCall(floor, direction) - Remove the active hall call after it is served.
- ClearAllCalls() - Clears all calls in the case of Fire mode.
- ClearRequest(floor) - Clears a cabin request once the passenger reaches the destination.
- ClearAllRequests() - Clears all requests in the case of Fire mode.

Relation:
Realized by the exterior buttons on the floor, and the interior cabin buttons.

3.2 Cabin

The **Cabin interface** controls movement commands and floor-arrival feedback between the elevator car and the controller.

Input events:
- ArrivedAtFloor(floor) - Triggered when sensors detect the cabin has reached a floor.

Output Actions:
- MoveUp() - Command motor to move the cabin upward.
- MoveDown() - Command motor to move the cabin downward.

Relation:
Realized by the motors on the cabin, and floor sensors.

3.3 Doors

The **Doors interface** ensures doors open and close safely and synchronously with cabin motion.

Input events:
- ObstructionDetected(on) - Safety signals from sensors for when there is an obstruction.
- ObstructionDetected(off) - Safety signals from sensors for when there is no obstruction.
- DoorFullyOpen() - Boolean to indicate the door is fully open.
- DoorFullyClosed() - Boolean to indicate the door is fully closed.

Output actions:
- OpenDoor() - Command to open doors.
- CloseDoor() - Command to close doors.

Relation:
Realized by the motorized door assemblies, safety edge sensors, and proximity detectors that confirm door positions.

3.4 Supervisor

The **Supervisor** serves as the central controller and coordination unit for the elevator system when the elevator controller is in “Centralized Mode”.

Input events:
- Start() - Initializes elevator operation.
- Stop() - Stops elevator operation.

- Dispatch(floor, direction) - Assigns target floor and direction for service.
- SetMode(mode) - Handles mode transitions (fire, centralized, normal).

Relation:
Realized by the main control software or supervisory processor responsible for dispatch and safety management.

3.5 Notifications

The **Notifications interface** provides visual and auditory feedback to passengers about elevator state.

Output actions:
- SendStatus(arrival, currentFloor, direction) - Updates display with elevator status.
- ArrivalNoise() - Play sound when the elevator arrives.
- OverloadedNoise(start/stop) - Activates overload alarm if cabin is overweight.

Relation:
Realized by indicator lights, floor displays, chimes, and communication modules connected throughout the system.

3.6 Timer

The Timer interface supports tracking time-based events. It ensures doors close automatically.

Input conditions:
- Set(t) and Reset() commands from controllers.

Output actions:
- Timeout signal when a preset duration elapses (door dwell time).

Relation:
This logical interface is realized by internal hardware or software timers within the control processors.

4 System State

The system state represents the collection of data values the elevator controller must maintain to operate correctly. It defines the essential variables such as mode, current floor, destination, and door status that describe the elevator's present condition. These stored values allow the controller to make decisions and transition between operational behaviors.

Elevator controller:

- **Operating mode –** Normal, Centralized, or Fire mode
- **Status -** Elevator is on/off
- **Current floor -** Elevator current location
- **Direction -** Direction the elevator is headed in Up/Down/None
- **Destination floor, direction -** Elevator destination location and the direction the elevator has to go in order to reach its destination floor.
- **Door status –** open/closed door
- **Door obstruction** – On/off depending on if an obstruction is detected.
- **Pending calls –** 3 Boolean arrays of pending calls for that elevator. One for button pressed in cabin (1... 10) and two for button pressed on floor (up/down) arrow.
    - requests[1..10]
    - upCalls[1..10]
    - downCalls[1..10]

5 State Transitions

The elevator controller responds to inputs from passengers, the cabin, the doors, and the supervisor. Based on these inputs and the current system state, the controller updates internal variables and issues commands that transition the system between operational states. The following subsections define all state transitions, including expanded clarifications and safety behaviors.

5.1 Passenger Inputs

Call(sourceFloor, direction)

Passenger presses an up or down button on a floor.

- If in Fire Mode: ignore the call.
- If in Normal Mode: set the corresponding boolean in upCalls[] or downCalls[].
- If elevator is Idle: set direction based on relative floor and issue MoveUp() or MoveDown().
- Ignore duplicate calls for the same floor and direction.

RequestDestination(destinationFloor)

Passenger selects a destination from inside the cabin.

- If in Fire Mode: ignore.
- If in Normal Mode: set requests[destinationFloor] = true.
- If elevator is Idle: determine direction and begin motion.

5.2 Cabin Input

ArrivedAtFloor(floor)

Sensor indicates that the cabin has reached a floor.

- Update currentFloor = floor.
- If floor is not a destination: sendStatus(false, currentFloor, direction).
- If floor is a destination:
    - sendStatus(true, currentFloor, none)
    - clearCall(currentFloor, direction) and/or clearRequest(currentFloor)
    - arrivalNoise()
    - OpenDoor()
- If this was the highest pending request in the current direction and more requests exist in the opposite direction: reverse direction.
- If no pending calls remain: transition to Idle.

5.3 Door Inputs

ObstructionDetected(on)

An obstruction triggers the safety sensor.

- Immediately issue OpenDoor().
- Reset and set the obstruction timer for 5 seconds.
- If obstruction reappears during door closing, stop closing and reopen, restarting the timer.

ObstructionDetected(off)

Obstruction timer expires with no obstruction present.

- Issue CloseDoor().

DoorFullyOpen()

Door position sensor indicates the door is fully open.

- If obstruction is active: do nothing until obstruction timeout.
- If obstruction is not active: set door timer (10 seconds) → CloseDoor() on timeout.
- Reset timer whenever the door is commanded to open.

DoorFullyClosed()

Door position sensor indicates the door is fully closed.

- If destination queue is empty: transition to Idle.
- If cabin is overweight:
  - OverloadedNoise(start)
  - OpenDoor()
  - After weight returns to normal:
    - OverloadedNoise(stop)
    - CloseDoor()
- If no weight issues:
  - Select next destination floor.
  - After timer(5) expires: issue MoveUp() or MoveDown().

5.4 Supervisor Inputs

Start()

System begins operation.

- Initialize all state variables.
- ClearAllCalls(), ClearAllRequests().
- Transition to Idle.

Stop()

System halts operation.

- If elevator is moving: stop at nearest safe floor.
- Clear all calls and requests.
- OpenDoor().
- Transition to Idle.

Dispatch(floor, direction)

Supervisor assigns the elevator to a specific floor.

- Insert the floor into the head of the destination queue.

- If Idle: determine direction and begin movement.

setMode(mode)

Supervisor changes the operational mode.

setMode(fire):

- ClearAllCalls(), ClearAllRequests().
- Dispatch elevator to floor 1.
- Ignore all passenger inputs until mode changes.
- Upon arrival at floor 1: open doors and remain there.

setMode(normal):

- Resume standard request handling.

5.5 Destination and Direction Logic

- After serving a floor, the controller clears all hall and cabin requests for that floor.
- The next destination is selected by prioritizing floors in the current travel direction.
- If no requests remain in the current direction but requests exist in the opposite direction: reverse direction.
- If no requests exist at all:
    - Transition to Idle.
    - Set direction to none.

5.6 Timer Behavior

- Any OpenDoor() or CloseDoor() command resets the door timer.
- A timeout event triggers reevaluation of obstruction, overload, and pending destinations before the next action is chosen.
- A door-open timeout with no obstruction present triggers CloseDoor(), unless weight limits prevent closure.

---

6 Conclusion

This SRS has defined the full operational and logical requirements for a 10-story, 4-shaft Elevator Control System. This detailed key logical interfaces and subsystems, which include the Passenger, Cabin, Doors, Supervisor, Notifications, and Timer. This also defines and establishes the possible system states, and the state transitions that drive the system's behavior. This document will guide the development of an elevator system capable of meeting the primary goals of passenger safety, efficiency, and reliability with the capability of centralized control.

7 Definition of terms

No terms to define.