Software Requirements Specification
Elevator System Controller

Prepared for Dr. Daniel M. Berry

by

Alex Kalaidjian

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1 Purpose

The purpose of this document is to provide a consistent and complete description of the requirements for the software of an elevator controller. The requirements will be presented using textual descriptions to explain concepts, different types of diagrams to illustrate complicated interactions, and tables to relate relevant information.

The intended audience of this document is all of the stakeholders for a project involving the development of elevator controller software. This includes, but is not limited to, software developers, project managers, quality assurance personnel, and customers.

1.2 Scope

The software of the elevator controller is responsible for the safe and efficient operation of all of the other components within the elevator system. The controller's main goal is essentially to handle input signals from other components and respond accordingly with output signals. Two of the main computational obligations of the controller are to have a queuing system to log and process requests from passengers and to navigate the cabs of the elevators between floors in response to those requests. One of the other important objectives of the controller is ensure the safety of the passengers using the elevator system at all times. This is achieved through the interaction with certain components dedicated to monitoring environment attributes that are important to safety.

The controller, however, is not responsible for adapting its behavior to a high volume of requests. It uses the same algorithm to respond to requests regardless of the current state of the environment.

It is important to note that this SRS document only pertains to the requirements of the software of the elevator controller. It does not include requirements for the hardware that it will be deployed on. It also does not include requirements for the other components of the elevator system. Other components are mentioned because the requirements of the controller are indeed based on the interactions between it and the other components of the system; however, the requirements of the other components, as well as the overall elevator system itself, is outside the scope of this document and is therefore not included.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:

Sheave: A component with a groove around its circumference to support and contain a rope or cable and a bearing at its center to permit rotation about a shaft. [2]

Acronyms:

UC: Use Case

SRS: Software Requirements Specification

1.4 Terminology

Elevator doors / doors: Refers to the inner door on an elevator cab and the corresponding outer door on the elevator shaft at the same floor at which the elevator cab is currently stopped at.

Note: This definition is useful because the door opening device on an elevator cab always opens the outer door of the shaft at the same time as the inner door of the cab. There is virtually no need to refer to the inner and outer doors separately.

Position marker: A position marker is a something on the side of an elevator shaft that is typically detected by some kind of sensor on an elevator cab inside of the shaft. There are many position markers all lined up through out the height of the shaft. The sensor reports the detection of these markers to a computer so that it can determine the position of the elevator cab that the sensor is attached to. An example of a position marker is a hole.

Active Summon / Floor Request: An active summon or floor request is a summon or floor request that has not yet been fulfilled by the arrival of an elevator cab. A summon or floor request becomes active when the appropriate button is pressed. It becomes inactive when a cab arrives at the summoned from or requested floor.

1.5 References

[1] How Stuff Works "How Elevators Work"
http://science.howstuffworks.com/elevator.htm, accessed on Nov. 3, 2005

[2] Glossary of Rigging Terms
http://www.sapsis-rigging.com/Glossary.html, accessed on Nov. 3, 2005

[3] Elevator – Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Elevator, accessed on Nov. 3, 2005

1.6 Overview

The rest of this SRS document contains all of the requirements for the elevator controller presented in several ways and organized into different sections.

Section 2 contains general information that is not too specific and is provided as a background for the following sections. It contains descriptions of all of the other elevator components that the controller interacts with, as well as a context diagram that illustrates the entire elevator system. It also lists product functions, constraints, and assumptions about the controller. Section 2 is a good section for customers to read.

Section 3 contains more detail and presents the requirements with many different diagrams that illustrate the functional requirements of the elevator controller. Some of the types of diagrams that are used here include, sequence diagrams, state diagrams, and collaboration diagrams. There is also a part of this section that describes the interface between the controller and the rest of the elevator system as a set of signals. Section 3 is most suitable for developers and testers.

Section 4 of the SRS contains supplementary information required to complete the document's breadth. It includes tables of all of the functional and non-functional requirements, as well as a table for each use case of the elevator controller. Both the requirements and the use cases are cross referenced with each other to provide traceability among both types of artifacts.

2. General Description

The product described in this document is software for an elevator controller. The controller is part of a larger elevator system comprised of several components other than the controller that are required to operate the elevator on a day-to-day basis. The elevator controller is responsible for directing the operation of most of the other components of the system. If it functions correctly, the controller allows passengers to use the elevator system in an intuitive and efficient manner. Note that this particular system is comprised of two elevator shafts, each with their own cab. The elevators provide service to 6 floors.

2.1 Product Perspective

The elevator controller directly interacts with the following other components of the entire elevator system:

Buttons

Summon Buttons: These buttons are on a button panel on the outside of the elevator shafts and are used by potential passengers to call an elevator cab to the floor that the pressed summon button is located on. There are two summon buttons on each floor – one for up, another for down, except on the top floor where there is only down and on the bottom floor where there is only up. The controller interacts with these buttons by receiving press and release signals indicating the requested direction and floor number. It also sends light on/off signals to indicate the status of the buttons.

Floor Request Buttons: This particular elevator controller will be controlling elevator cabs that are in a building with 6 floors. Consequently, each cab has 6 floor request buttons labeled 1 through 6 that passengers can use to direct the elevator cabs to the floor that they would like to go to. These buttons are located on a button panel on the interior of each elevator cab. The controller interacts with these buttons by receiving pressed signals indicating the desired floor number and elevator cab which they were pressed from. It also sends light on/off signals to indicate the status of the buttons.

Open Door Button: This button is on the interior button panel of each cab. A passenger can press this button to open the elevator doors or keep pressing it to keep them open, but only when the elevator cab is stopped at a floor. Some elevator systems also have a close door button, but this one does not. The controller interacts with this button by receiving a signal when it is pressed and when it is released. Both of these signals include the cab from which they came from.

Emergency Stop Button: This button is on the interior button panel of each cab. A passenger can press this button to stop the elevator no matter where it is in a shaft. The controller interacts with this button by receiving a signal from it that indicates that it was pressed, as well as the cab that it came from.

Emergency Bell Button:	This button is on the interior button panel of each cab. A passenger can press this button to sound a bell to alert people outside of the elevator shaft that someone is trapped inside the elevator cab in case of a malfunction. The controller interacts with this button by receiving a signal from it that indicates that it was pressed.
Service Switch:	This feature of the elevator system is used to keep an elevator cab from moving and to keep the elevator doors from either opening or closing. It is useful for loading large items such as furniture into an elevator cab. The controller interacts with the switch by receiving a signal from it when it has been toggled to either AUTO or HOLD mode. AUTO is for normal operation; HOLD is to keep the elevator cab from moving and its doors from opening or closing.

Displays

Floor Number Display:	The interior of each elevator cab has a display that indicates to its passengers which floor the elevator cab is currently on. Some elevator systems have this floor number display on every floor outside of the elevator doors, but this system does not. The controller interacts with this display by sending a signal that tells it which floor number to display.
Direction Display:	The interior of each elevator cab has a display that indicates the current direction of an elevator cab; it is either up or down. The controller interacts with this display by sending it a signal that tells it which direction to display.

Sensors

Load Sensor:	The floor of each elevator cab has a load sensor that keeps track of how much weight there is in the elevator cab at any given time. The controller interacts with the load sensor by receiving a signal from it whenever the total weight that is currently in the elevator cab has changed. The signal indicates the new load inside of the elevator cab and the cab from which the signal is coming from.

Position Marker Sensor: Each elevator cab has a position marker sensor that can detect position makers (defined in section 1.3) on the inside of the elevator shaft. The controller interacts with the floor sensor by receiving a signal from it whenever it detects a position marker. The signal also includes which elevator cab it is coming from.

Bells

Emergency Bell: Somewhere in the elevator system is an emergency bell that is used to alert people outside of the elevator system that someone is trapped inside an elevator cab. The controller interacts with the emergency bell by sending it a signal to ring.

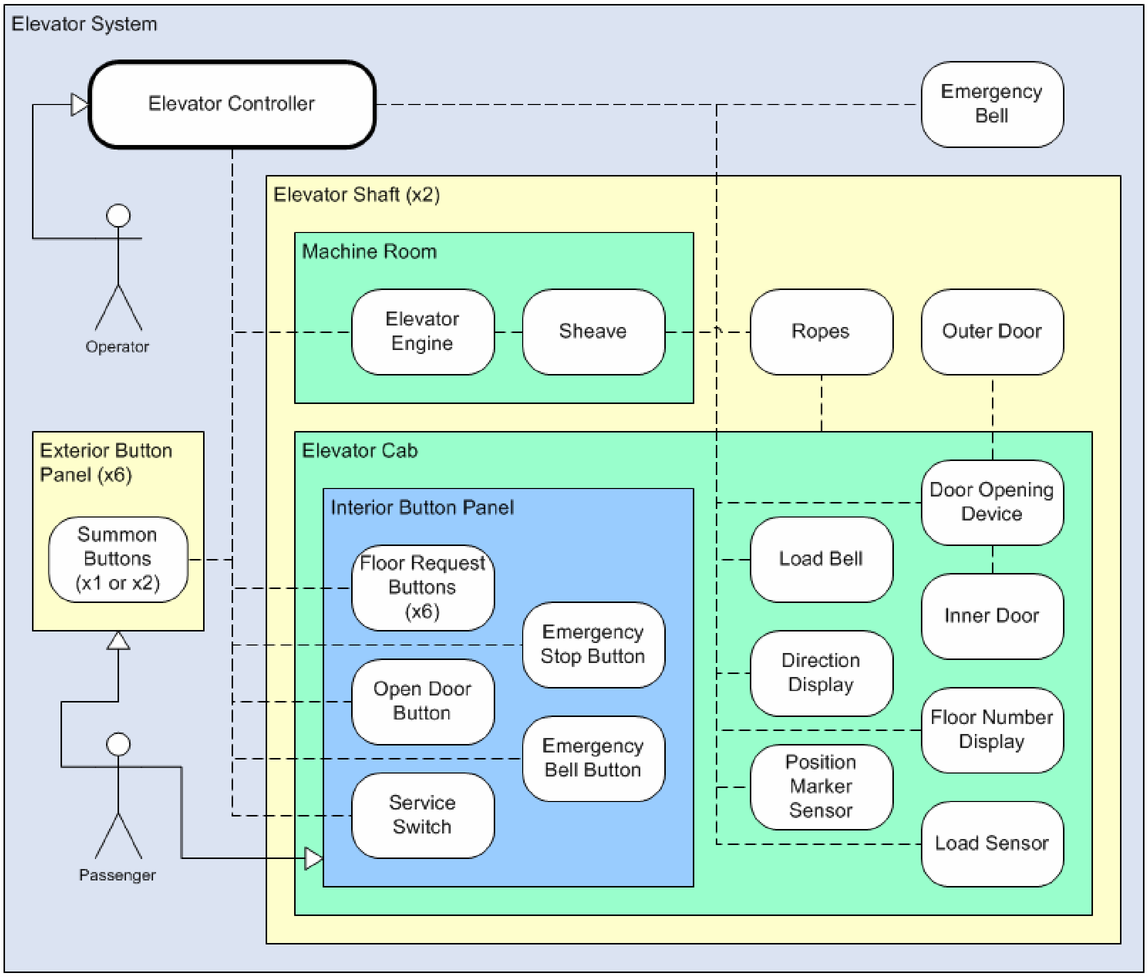
Load Bell: Each cab has a load bell that is used to alert the passengers inside the cab that there is too much weight in it to operate it safely. The controller interacts with the load bell be sending it a signal to ring.

Mechanical

Door Opening Device: On top of each elevator cab is a door opening device. This device opens the inner door of the elevator cab and the outer door of the elevator shaft simultaneously at each floor. The controller interacts with the door opening device by sending signals to open or close the doors and by receiving signals when the doors have been completely opened or closed. The signals that the controller receives also indicate which cab they are coming from.

Elevator Engine: The elevator engine is responsible for moving an elevator cab up and down between floors. As this elevator system uses a roped mechanism, the elevator engine is connected to a sheave which the ropes are looped around. The controller interacts with the elevator engine by sending it a signal that specifies at which speed and in what direction the engine should be going in. A stop signal is simply constructed by setting the speed parameter of the signal to zero.

Figure 1: Context Diagram



2.2 Product Functions

The primary function of the elevator controller is essentially to receive and process a variety of signals from several different components of a whole elevator system. It is able to send signals in response to the ones it receives in order to operate all of the other components in the system. This exchange of signals is how the elevator controller is able to keep the elevators running smoothly on a day-to-day basis.

Here are a few of the following ways the controller interacts with the other components of the elevator system:

- Controls the speed of elevator engines in order to move elevator cabs up and down their respective shafts.
- Queues and processes elevator summons and floor requests from passengers through the signals provided to it by several buttons.

- Processes information sent to it by load sensors in order to ensure that the load of a cab never exceeds the safety limit.
- Processes information sent to it by position marker sensors in order to keep track of where the elevator cabs are at all times, as well as their speed.
- Provides feedback to passengers through the lights on some of the buttons and the floor number and direction displays in each cab.
- Can sound alarm bells that are either invoked by trapped passengers or required to warn of excess load in a cab.
- Controls the operation of the elevator doors of a cab through communication with door opening devices.

2.3 User Characteristics

The users of the elevator controller are essentially the other components of the elevator system that interact with it (see section 2.1). The integral characteristic that these other components must have is being able to send and/or receive signals from the elevator controller. They must use the same protocol or format for signals as the controller does, so that communication between them is feasible. Also, some components, such as the elevator engines and door opening devices, must be able to process and react to the signals sent to it by the controller in a timely fashion. Otherwise, the elevator system may become too slow to be useful.

2.4 General Constraints

The hardware that the elevator controller software will be running on may constrain some design decisions pertaining to timing and performance, as well as signal communication.

Also, certain rules about public safety impose specific requirements on the elevator controller. The following is a list of constraints pertaining to the safety of the elevator system that the controller must manage:

- The inner doors of a cab should not be opened unless the cab is at a correct floor position and is stopped.
- The outer doors of a shaft should not be opened at a particular floor unless there is a cab stopped at that floor.
- An elevator cab should not start to move until its doors are completely closed.
- The acceleration and deceleration of a cab must be gradual enough to prevent the injury of any of the passengers inside.
- If the total weight in a cab has exceeded the safety limit, it should not resume normal operation until the total weight in the cab has been lowered below the safety limit.

- If the controller detects a malfunction in any of the other vital components of the elevator system, it must halt the operation of the affected cabs immediately.

2.5 Assumptions and Dependencies

The following is a list of assumptions made about the elevator system that affect the elevator controller:

- There are 6 floors that the elevator system provides service to.
- There are 2 elevator cabs that the controller is responsible for, each operating in its own, separate shaft.
- The performance of the elevator system must be reasonable; however, the controller does not have to perform special accommodations for high traffic periods during the day or unexpected burst increases in summon requests.
- The other components of the system that the controller must interact with can communicate with the controller by sending and receiving signals that conform to some agreed upon protocol or standard.
- There are a number of emergency functions throughout the elevator system. The elevator controller deals with these functions separately when they occur. There are no global emergency modes that the elevator controller goes into.
    - Emergency Bell Button Pressed
        - This only causes the emergency bell of the system to ring, nothing more.
    - Emergency Stop button Pressed
        - This causes the cab from which the button was pressed from to stop. The controller remains active and keeps accepting requests because the other cab may still be functional and a stopped cab can certainly start answering requests once its normal operation is resumed. Assume that a subsequent press of the emergency stop button after a cab has been stopped causes the cab to resume normal operation.
    - Load Sensor Detects Too Much Weight
        - Likewise, this only affects the cab with too much weight inside, requests are still logged and queued normally by the controller – it does not enter any specific mode of emergency operation.
    - Service Switch
        - Switching a cab into hold mode can be performed manually in the event of an emergency, but not necessarily. Regardless of the context, like the other events mentioned above, the controller still operates normally while taking into account the held cab – it does not enter any specific mode of emergency operation.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Overall System

3.1.1.1 System Sequence Diagrams

Figure 2: Sequence Diagram of UC1 – Process Pressed Signal from Summon Button

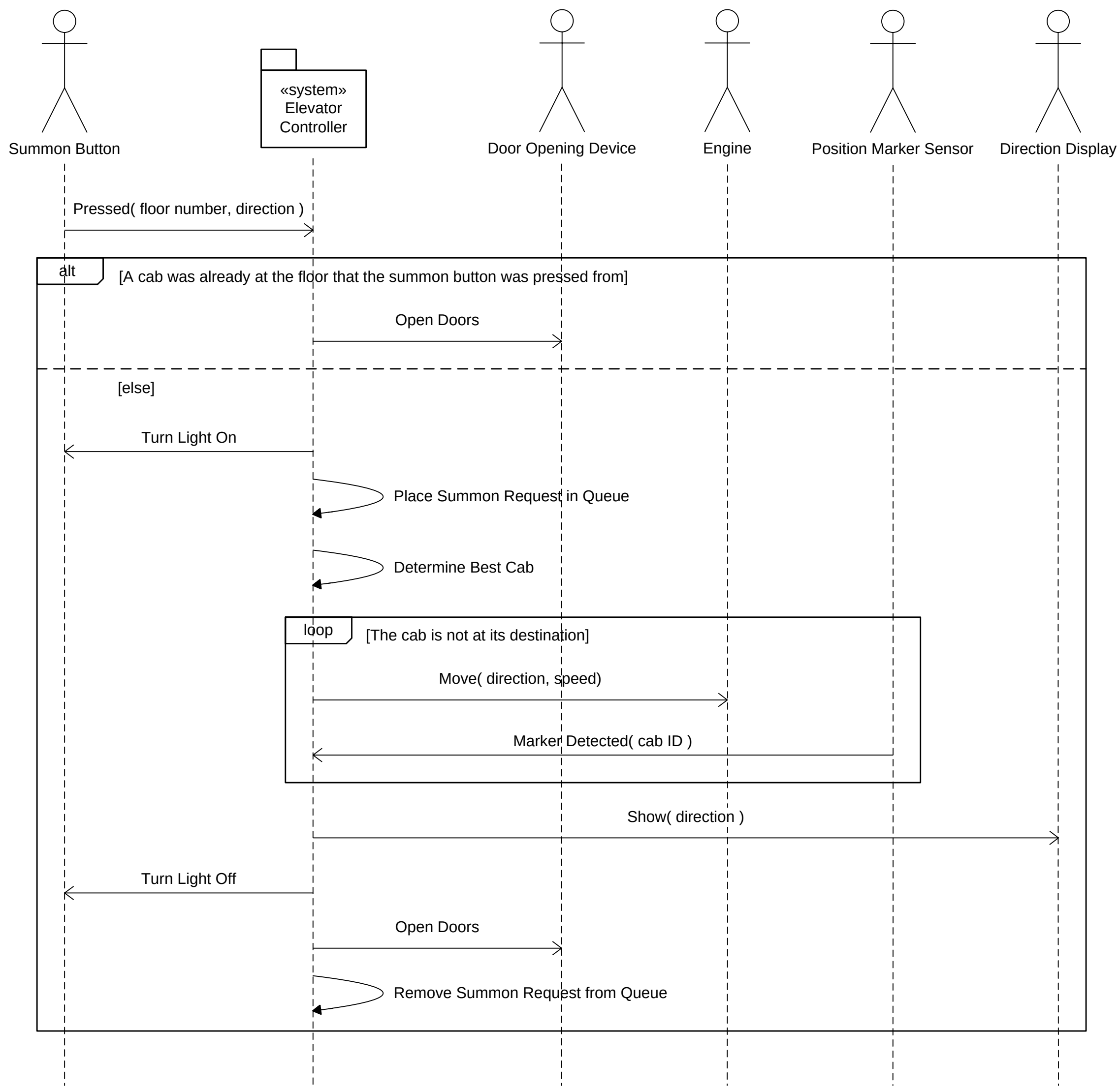


Figure 3: Sequence Diagram of UC2 – Process Released Signal from Summon Button

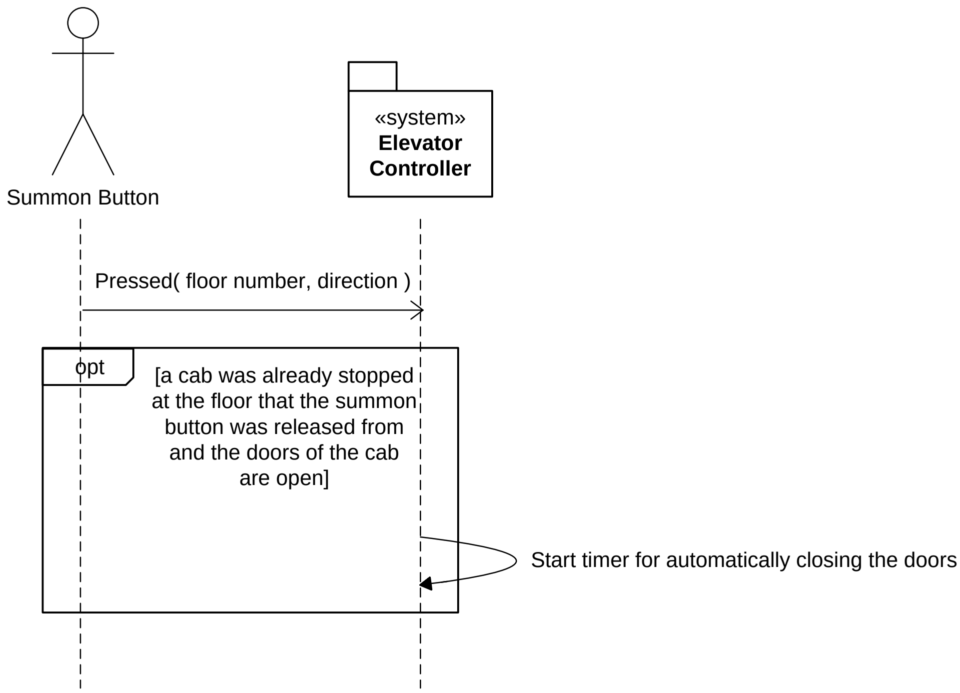


Figure 4: Sequence Diagram of UC3 – Process Pressed Signal from Floor Request Button

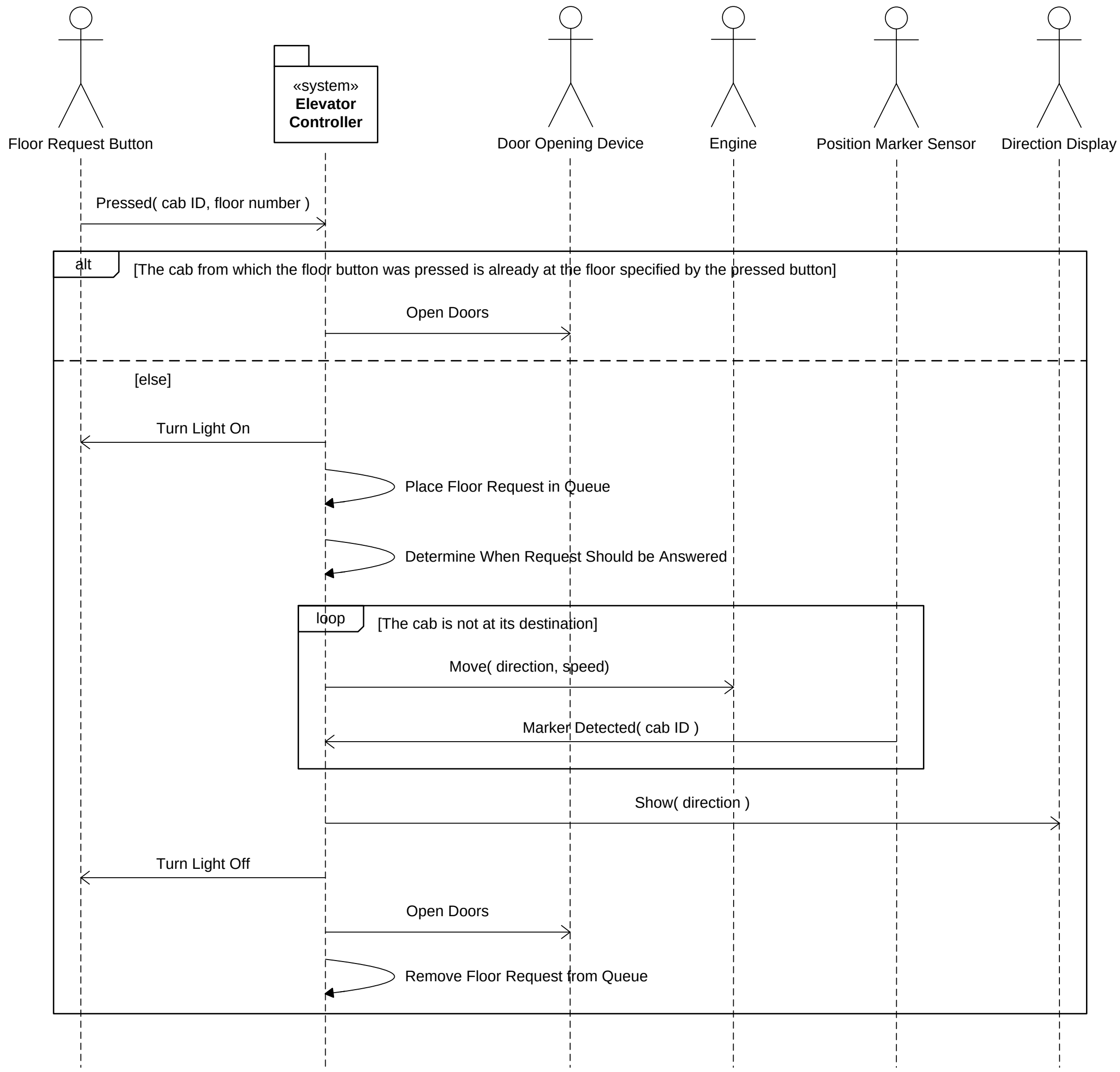


Figure 5: Sequence Diagram of UC4 – Process Pressed Signal from Open Door Button

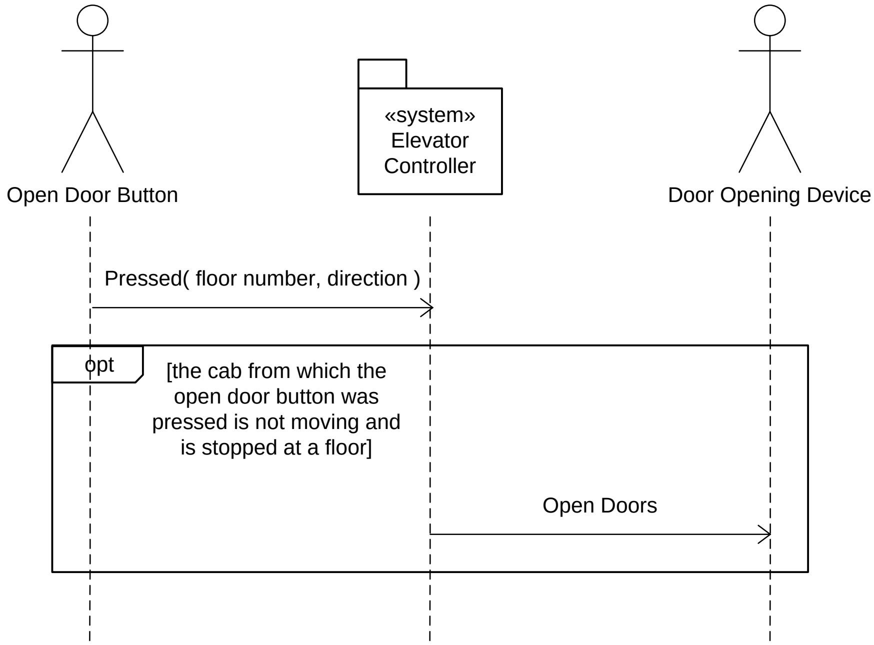


Figure 6: Sequence Diagram of UC5 – Process Released Signal from Open Door Button

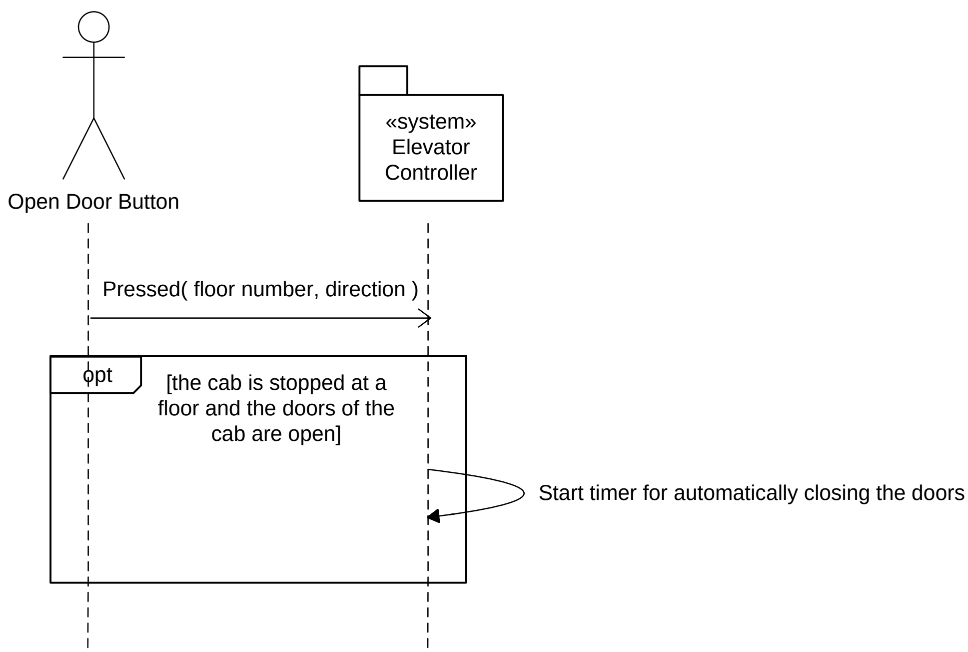


Figure 7: Sequence Diagram of UC6 – Process Pressed Signal from Emergency Stop Button

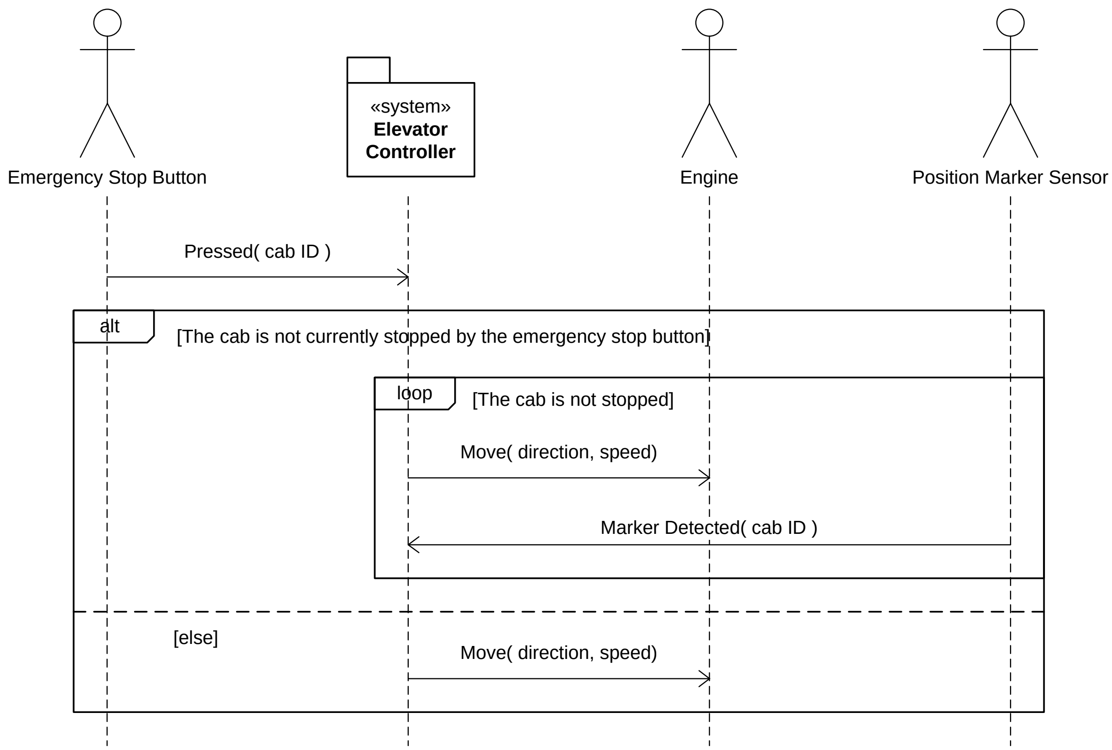


Figure 8: Sequence Diagram of UC7 – Process Pressed Signal from Emergency Bell Button

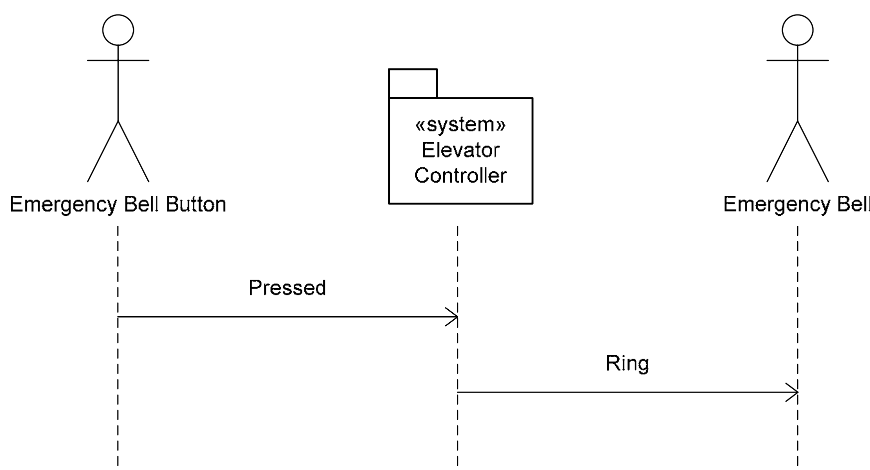


Figure 9: Sequence Diagram of UC8 – Process HOLD Mode Signal from Service Switch

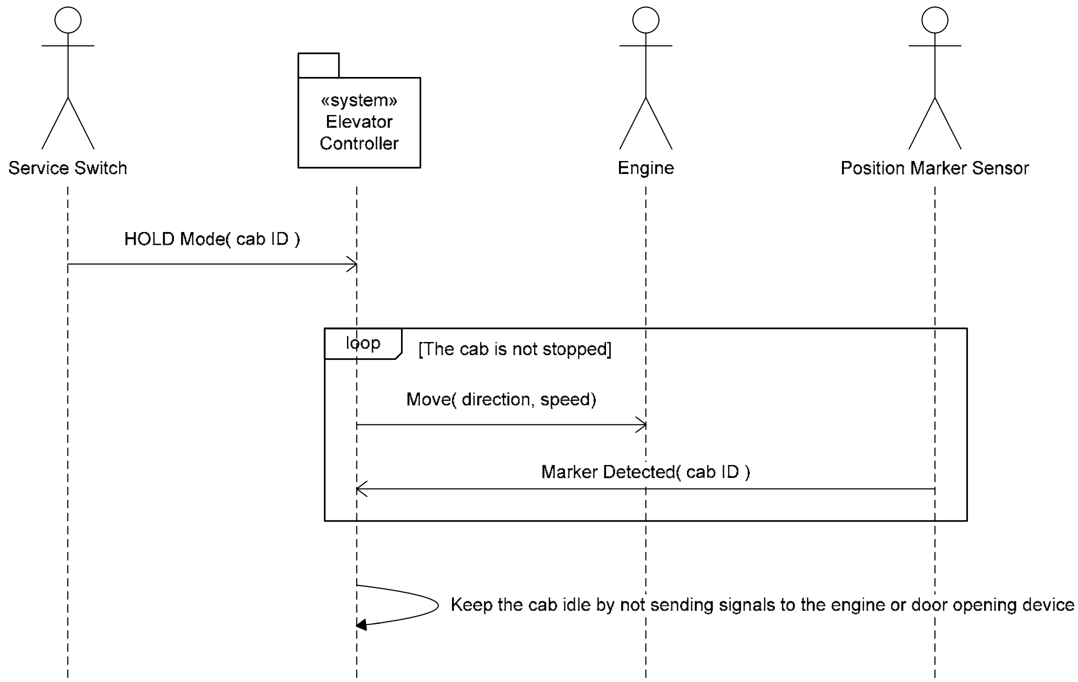


Figure 10: Sequence Diagram of UC9 – Process AUTO Mode Signal from Service Switch

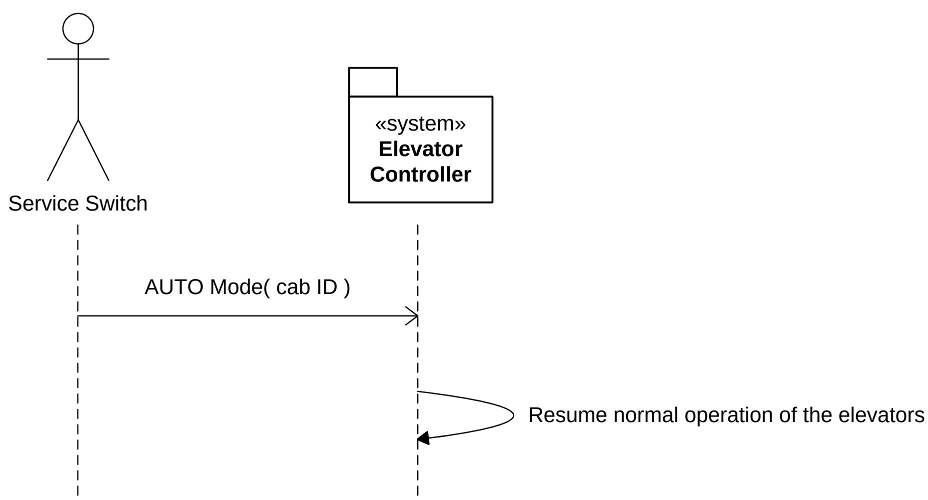


Figure 11: Sequence Diagram of UC10 – Process Weight Changed Signal from Load Sensor

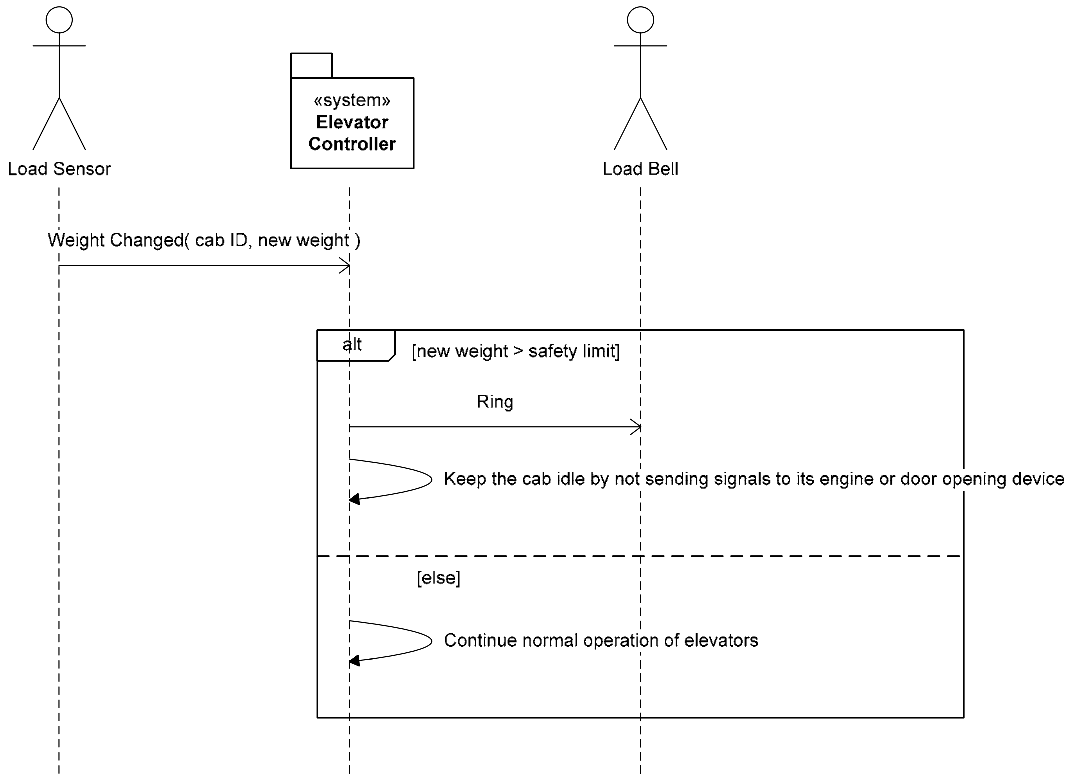


Figure 12: Sequence Diagram of UC11 – Process Signal from Position Marker Sensor

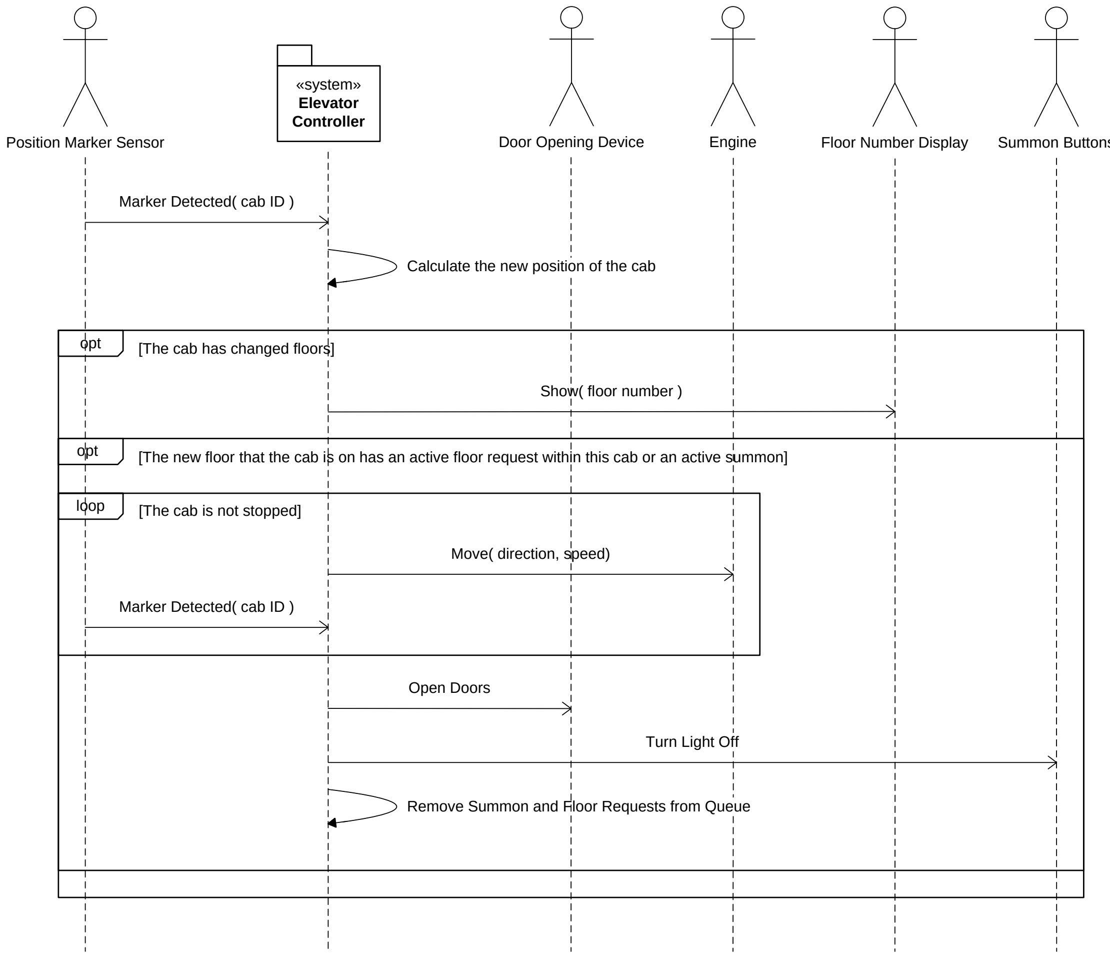


Figure 13: Sequence Diagram of UC12 – Process Turn Off Signal from Operator

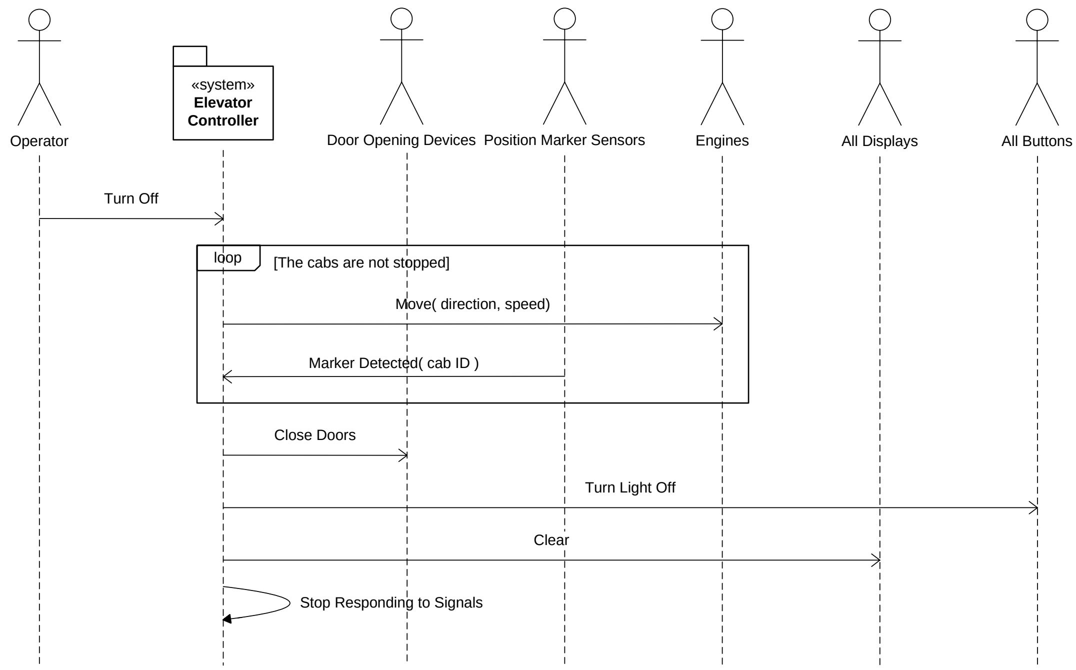


Figure 14: Sequence Diagram of UC13 – Process Turn On Signal from Operator

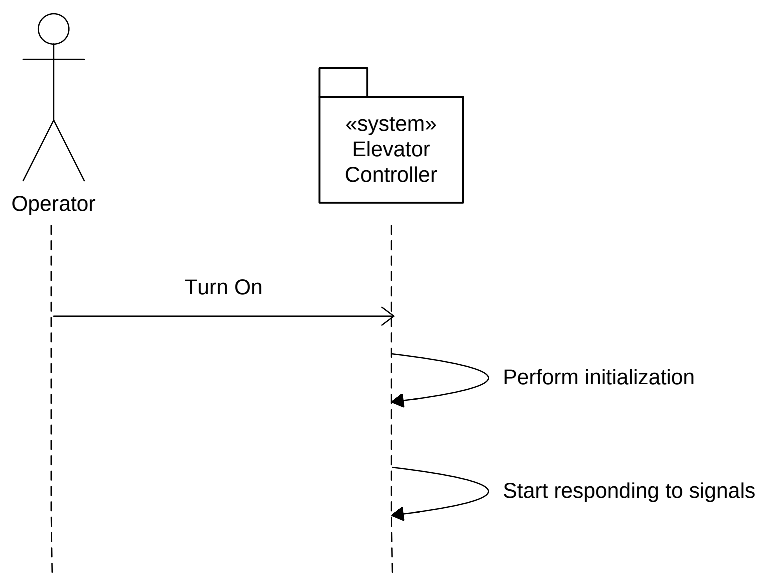


Figure 15: Sequence Diagram of UC14 – Process Doors Opened Signal from Door Opening Device

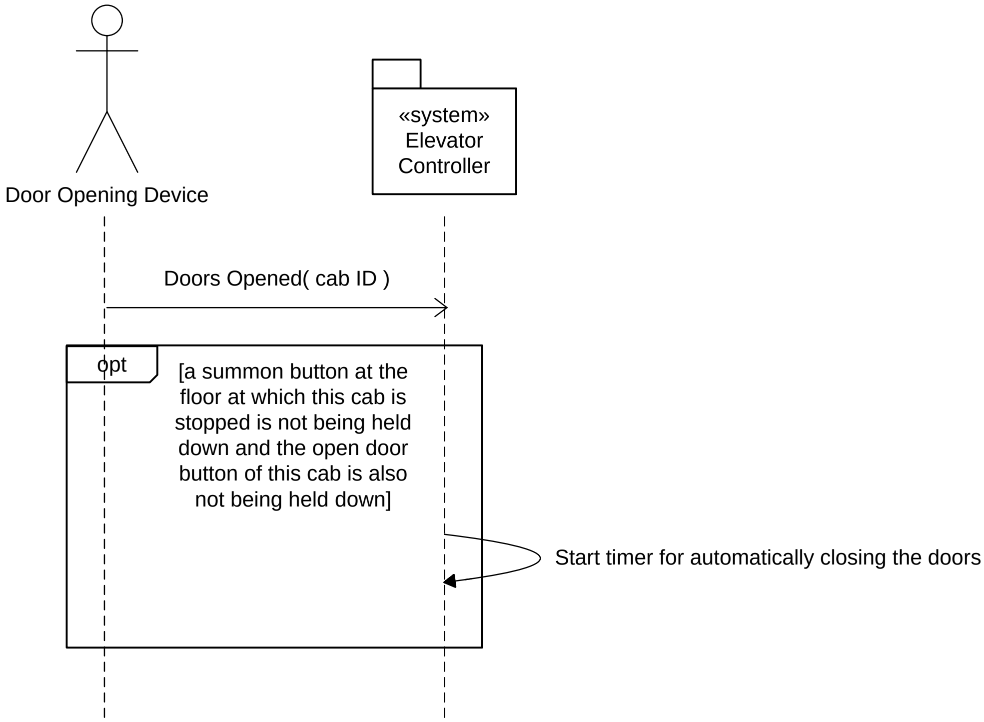
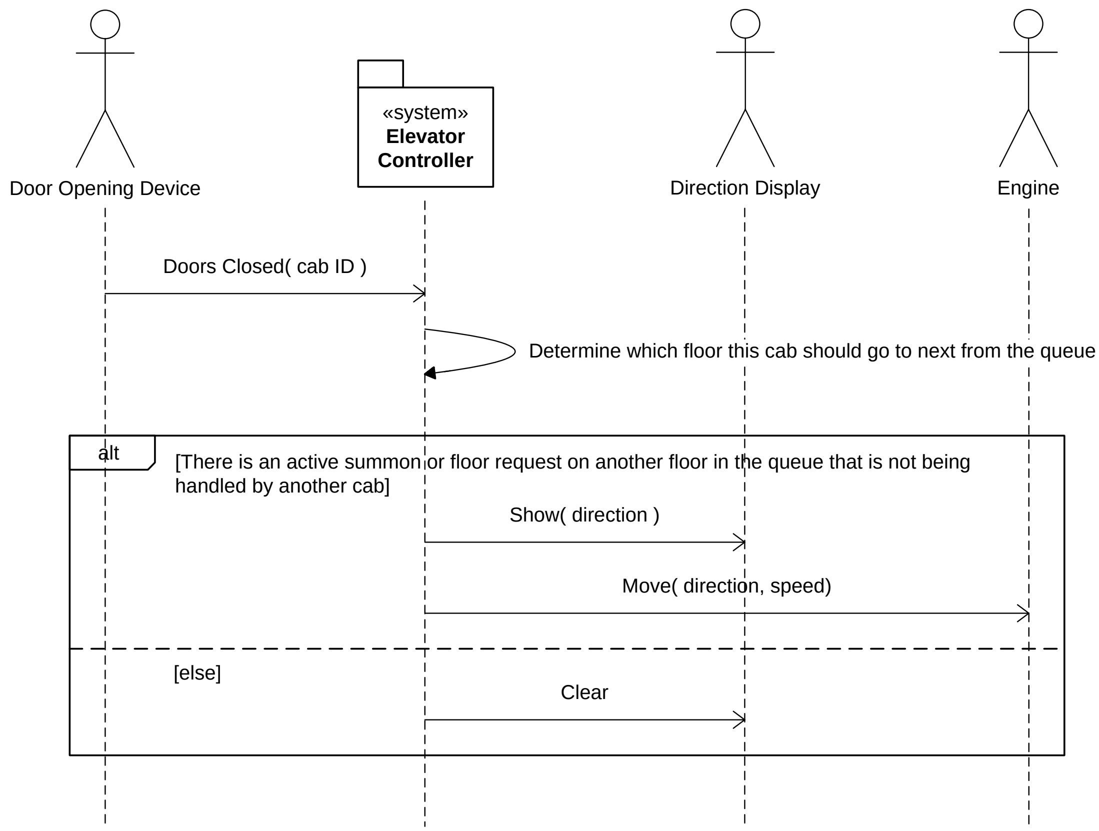


Figure 16: Sequence Diagram of UC15 – Process Doors Closed Signal from Door Opening Device



3.1.1.2 System State Diagrams

Figure 17: High Level System State Diagram

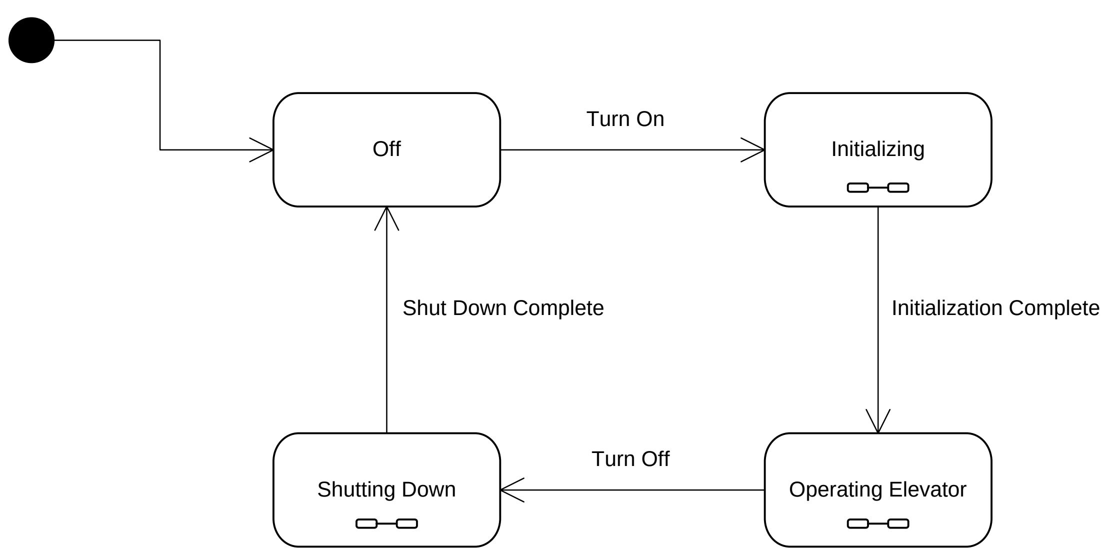


Figure 18: Operating Elevator State Diagram

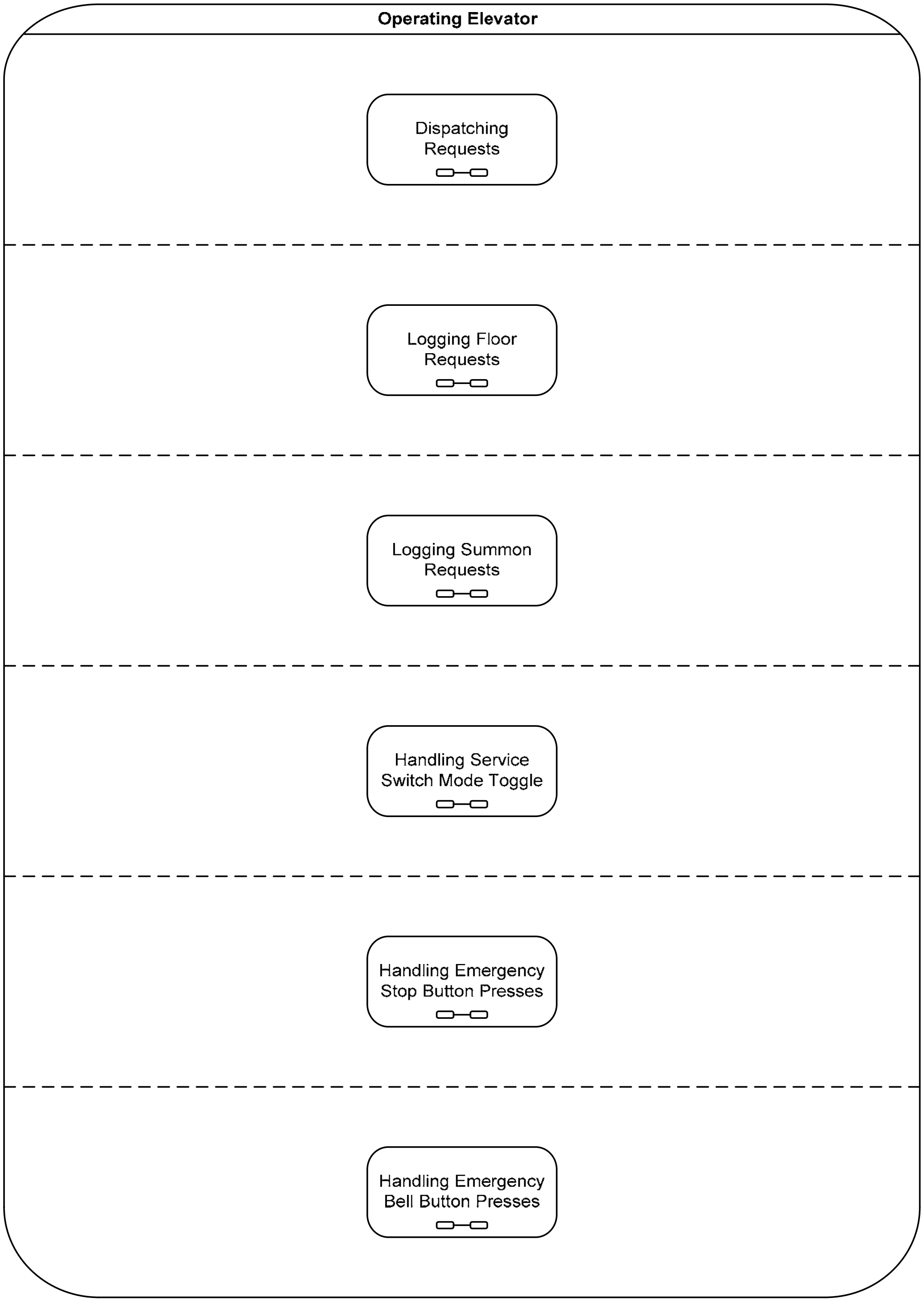
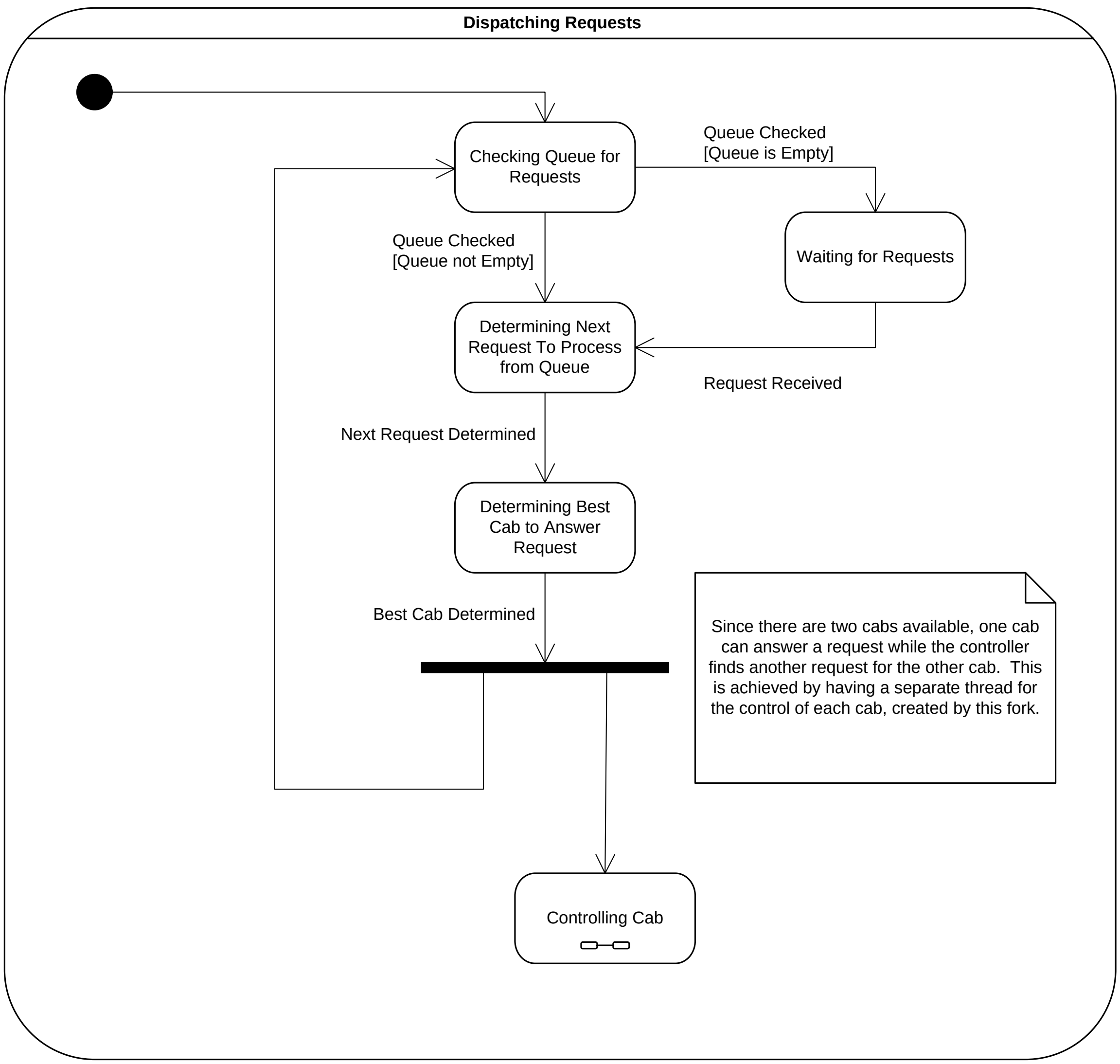


Figure 18 is an important state diagram. It implies that there is a lot of concurrency within the elevator controller. This concurrency is essential because of the fact that most input signals need to be handled as soon as they are received and input signals can not be lost simply because the controller is in one specific state instead of another. After close examination, it is evident that all 6 sub-states shown in the Operating Elevator state must run within the controller concurrently.

Usually, when there is a lot of concurrency within a system or sub-system, data synchronization issues arise. However, the most important data stored in the controller is a queue, and its management is trivial. The two logging sub-states can only add requests to the queue, while the dispatching sub-state only processes and removes requests from it. Since the controller does not enter any specific emergency modes, the other 3 sub-states can operate concurrently without any concerns. For example, even though the emergency stop button is pressed from a cab that should not mean that requests should not be logged. Another cab can still answer requests while one cab is stopped and the stopped cab can resume answering requests from the queue when its normal operation is resumed.

Figure 19: Dispatching Requests State Diagram



To elaborate on the embedded note in Figure 19, the fork in the diagram is very important. It means that after the best cab to answer a request is determined, the current process branches into two new processes. One of the new processes is created to control the selected cab into answering the request. The other process goes back to the start state and checks the queue for more requests. This branch is necessary because of the fact that the elevator controller can operate two cabs simultaneously. While one cab is answering a request, the controller does not wait until that request has been answered to dispatch the next request; it does it as soon as the previous request has been delegated to a cab. Note that if both cabs are currently answering requests, the controller will stay in the state "Determining Best Cab to Answer Request" until one of them becomes free.

Figure 20: Controlling Cab State Diagram

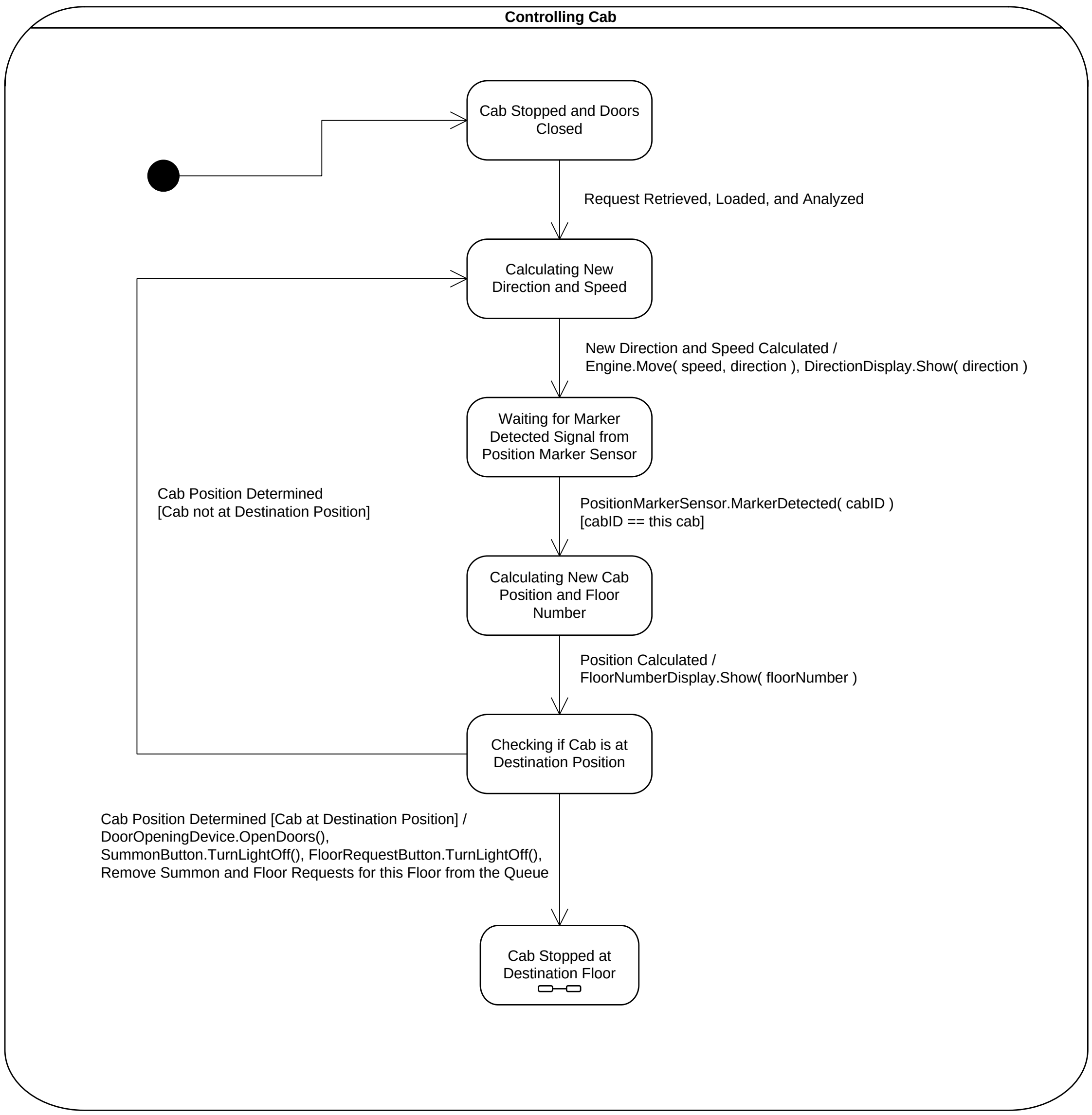
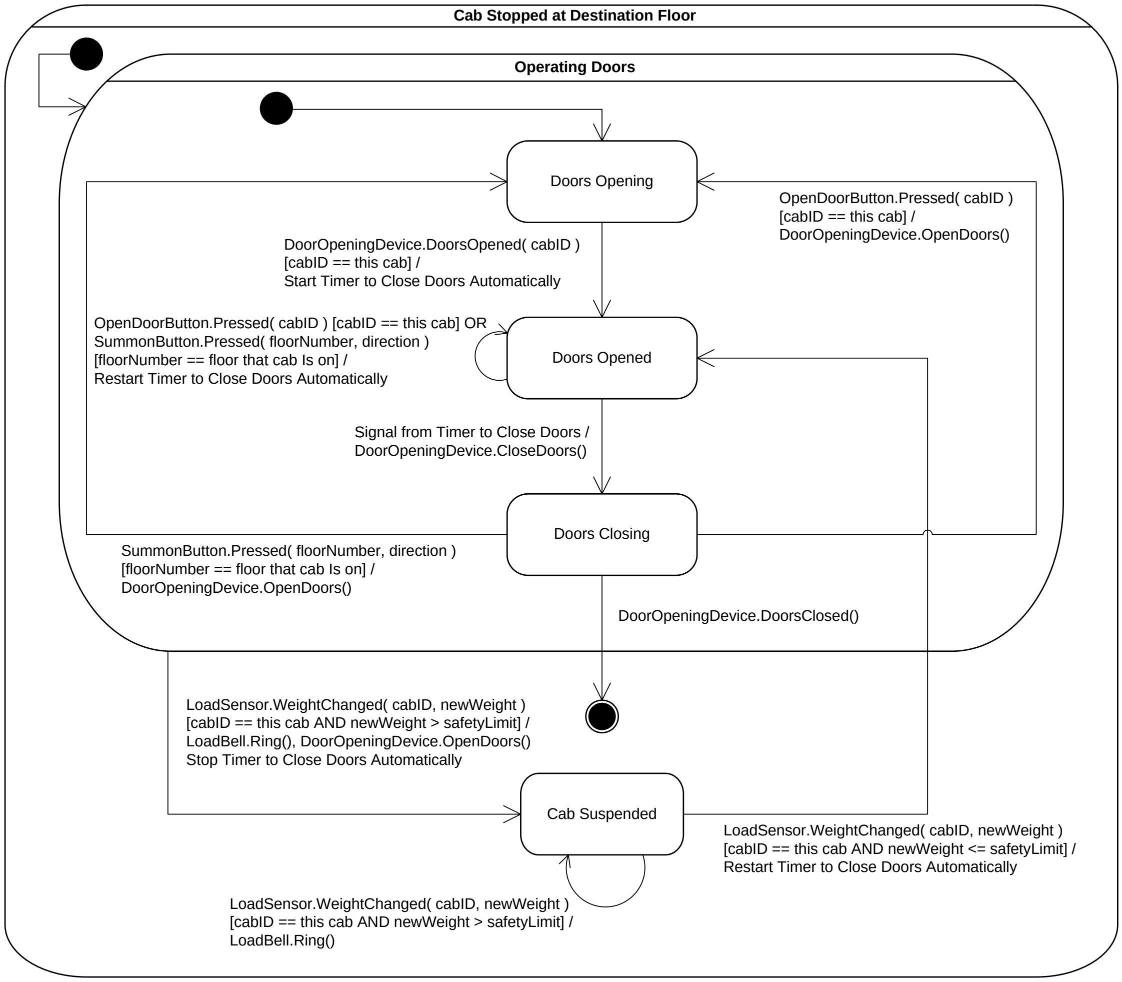


Figure 21: Cab Stopped at Destination Floor State Diagram



There is a tricky situation in the state diagram of Figure 21. You can see that the Cab Suspended state can be entered from any of the three states in Operating Doors, not just Doors Opened. The reason for this is that a passenger could sneak into the elevator cab while the doors are opening or closing and cause the total weight of the cab to exceed the safety limit. So the cab can become suspended while the doors are opening or closing, not just when they are fully opened. The Cab Suspended state only returns to the Doors Opened state because the doors are opened when a cab becomes suspended and remain that way until normal operation resumes.

Figure 22: Logging Floor Requests State Diagram

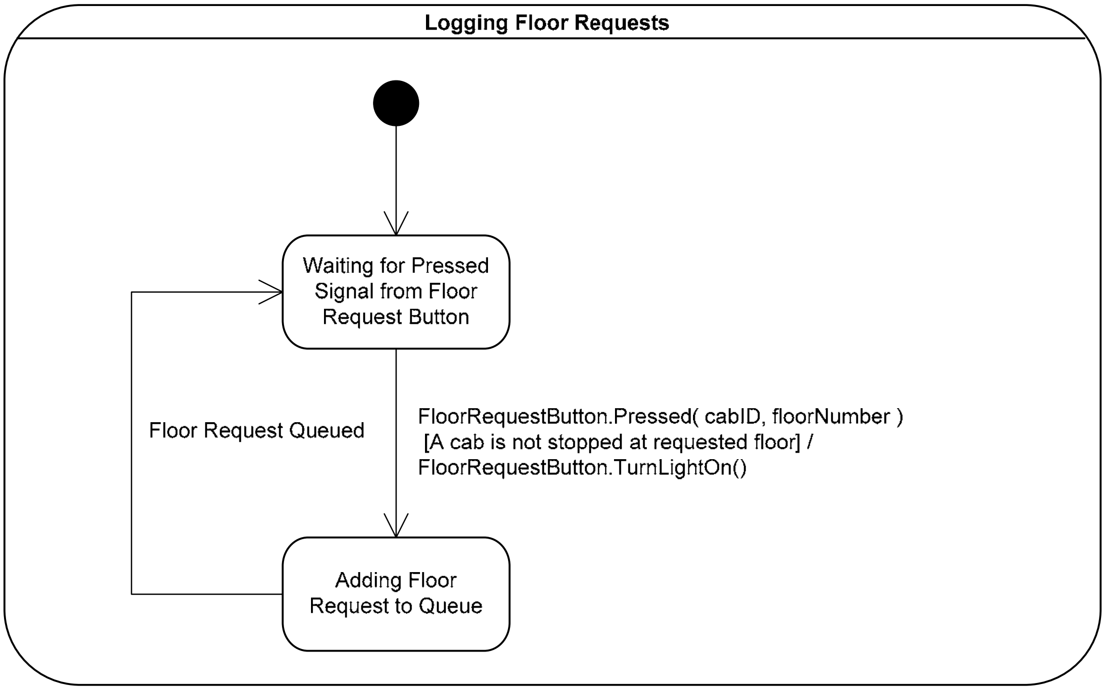


Figure 23: Logging Summon Requests State Diagram

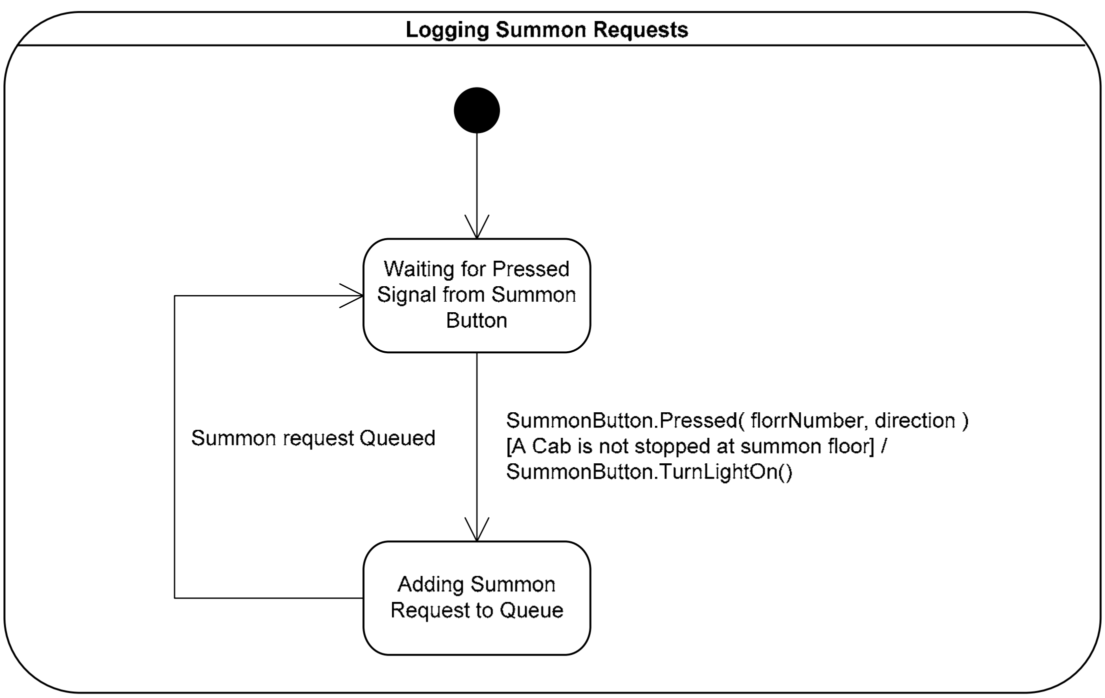


Figure 24: Handling Service Switch Mode Toggle State Diagram

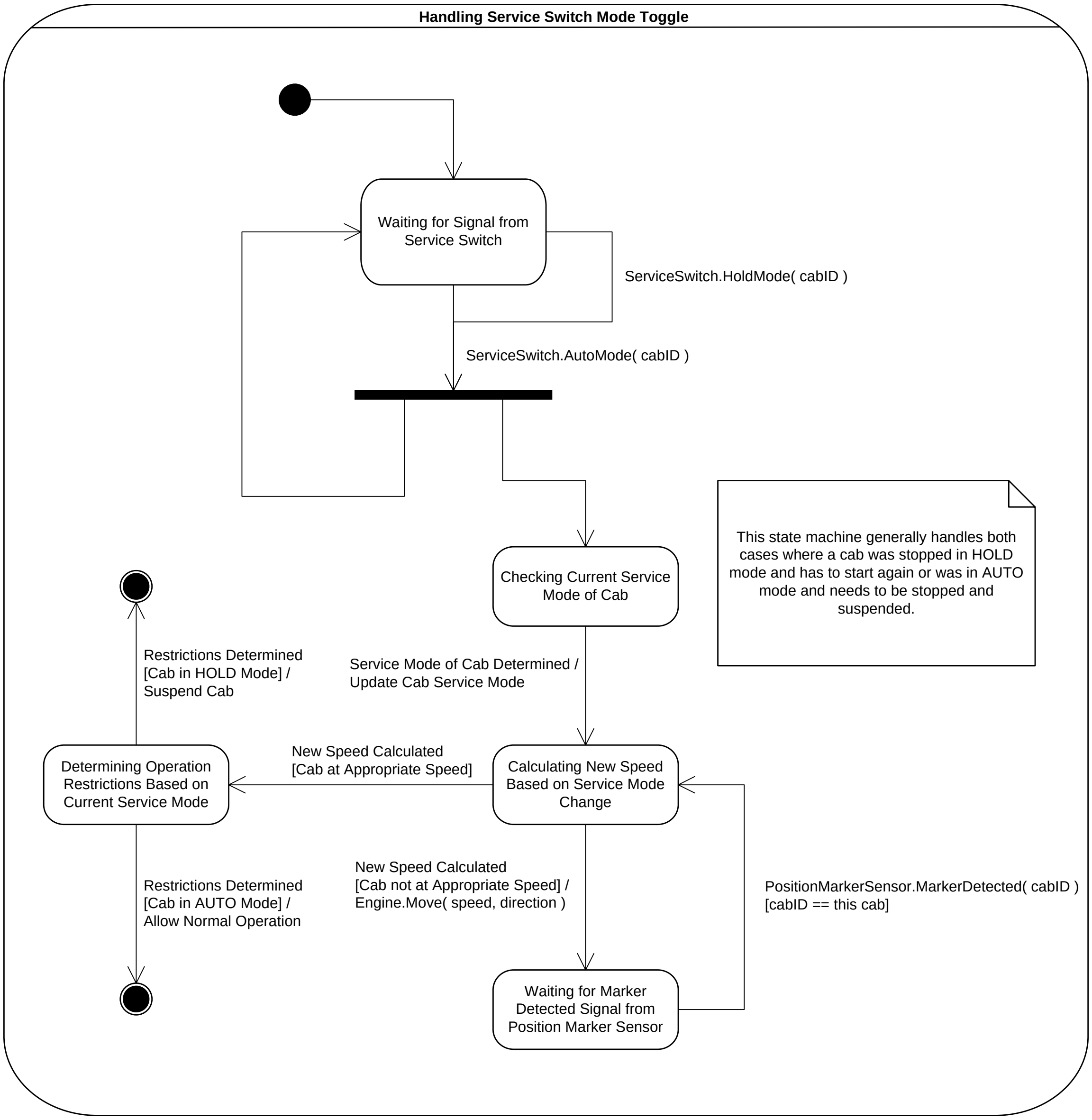


Figure 25: Handling Emergency Stop Button Presses State Diagram

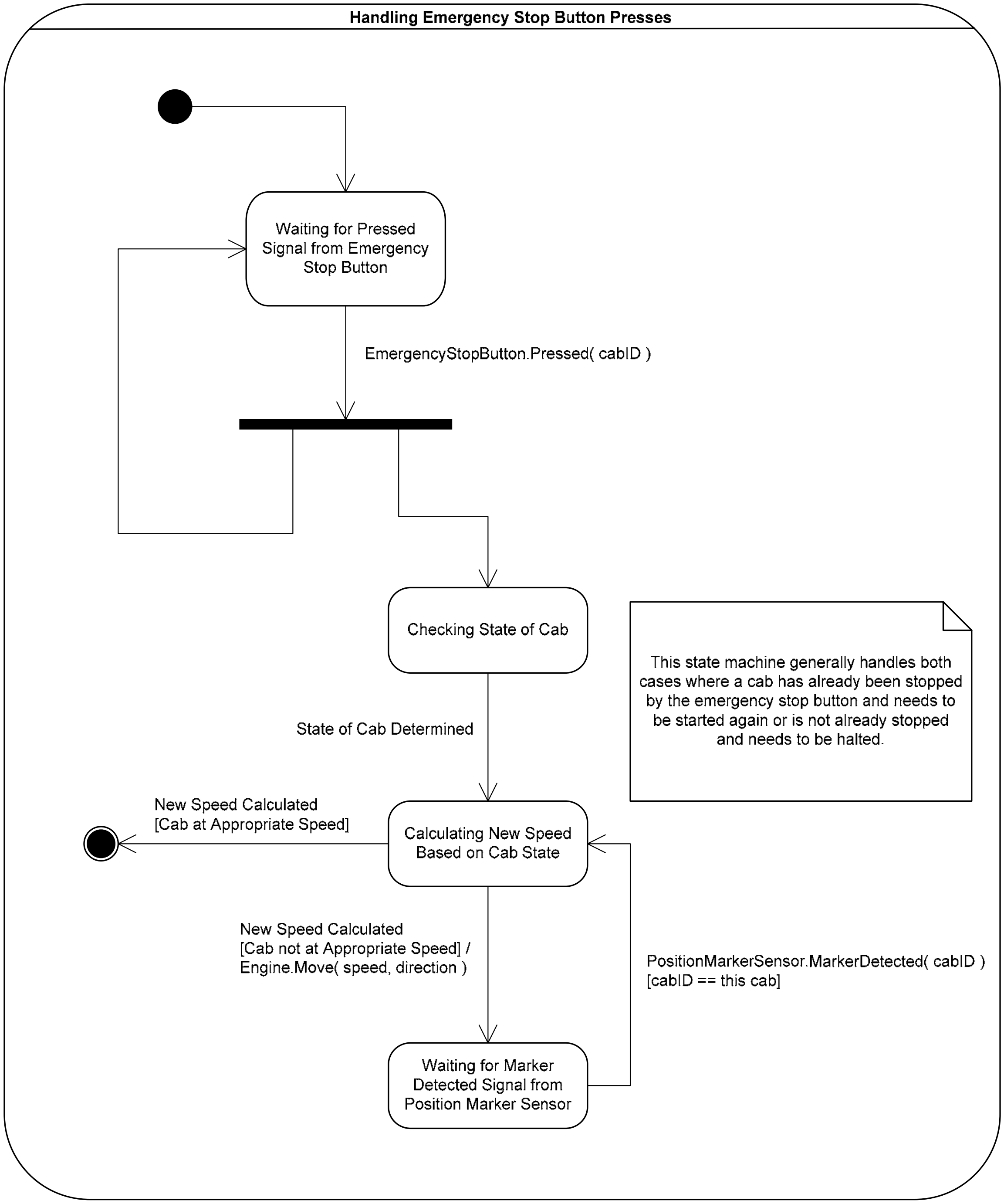
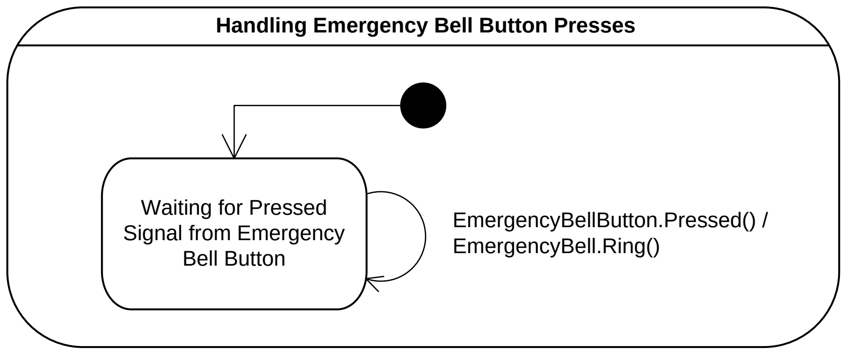
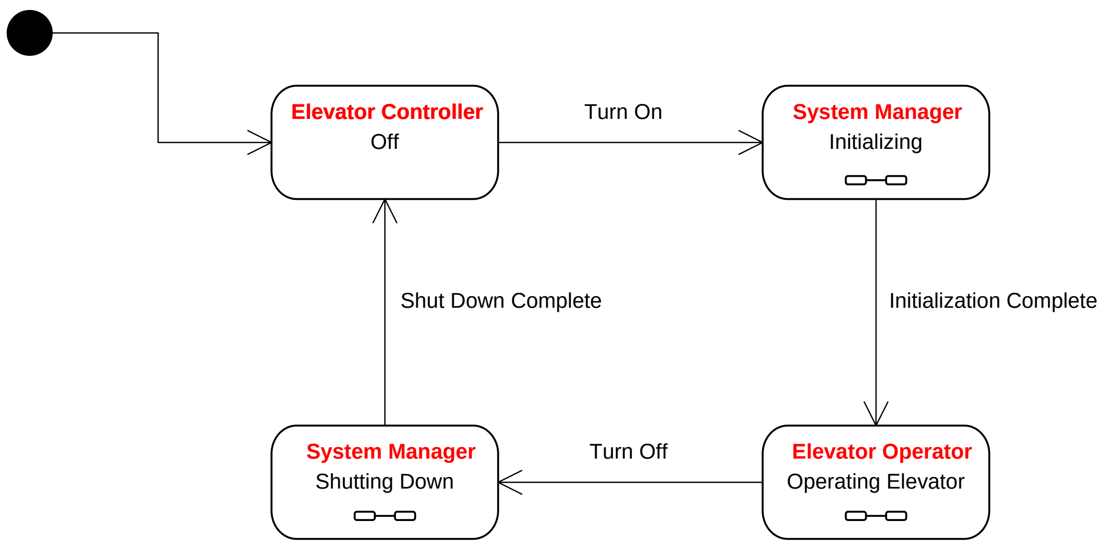


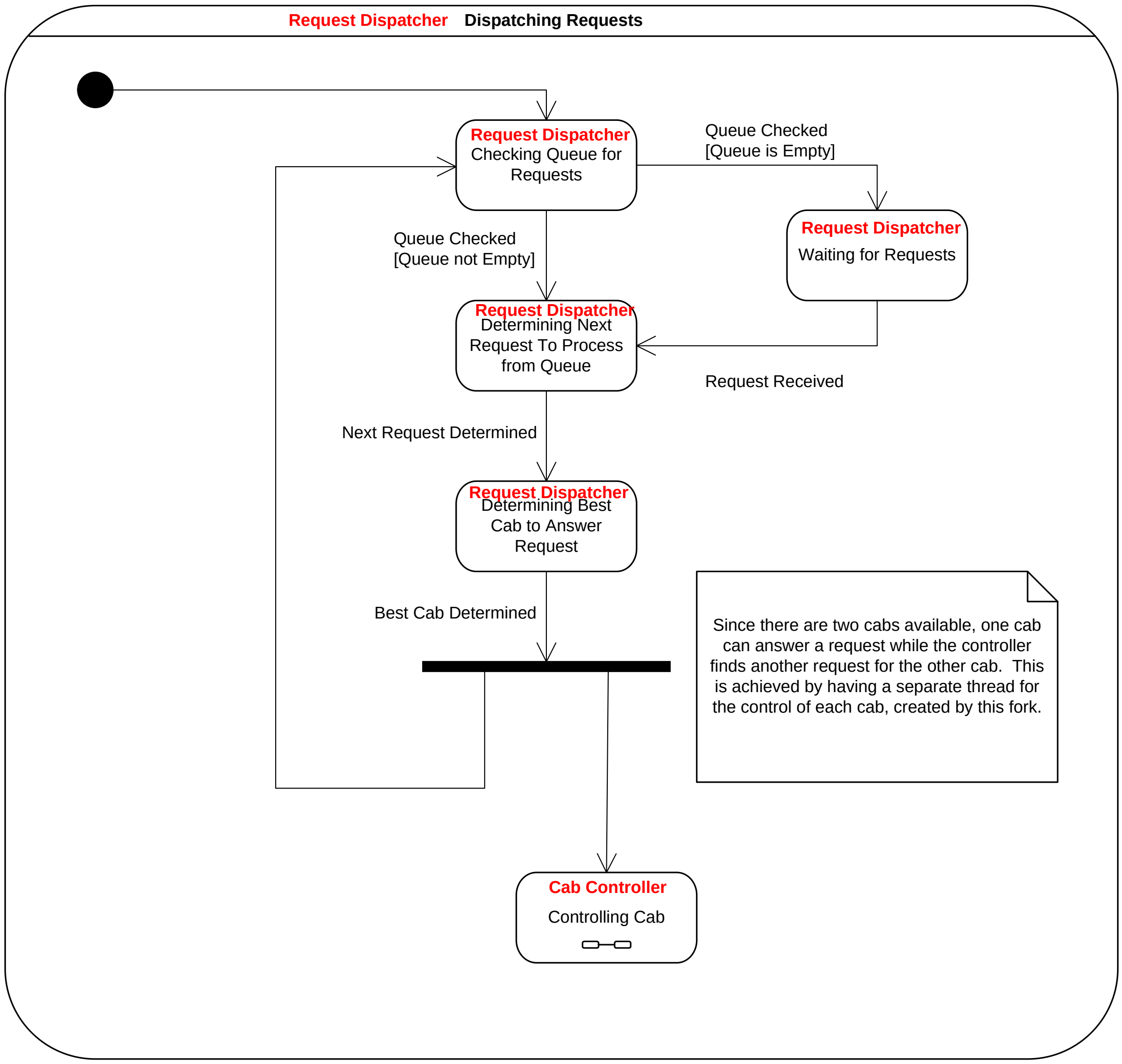
Figure 26: Handling Emergency Bell Button Presses State Diagram

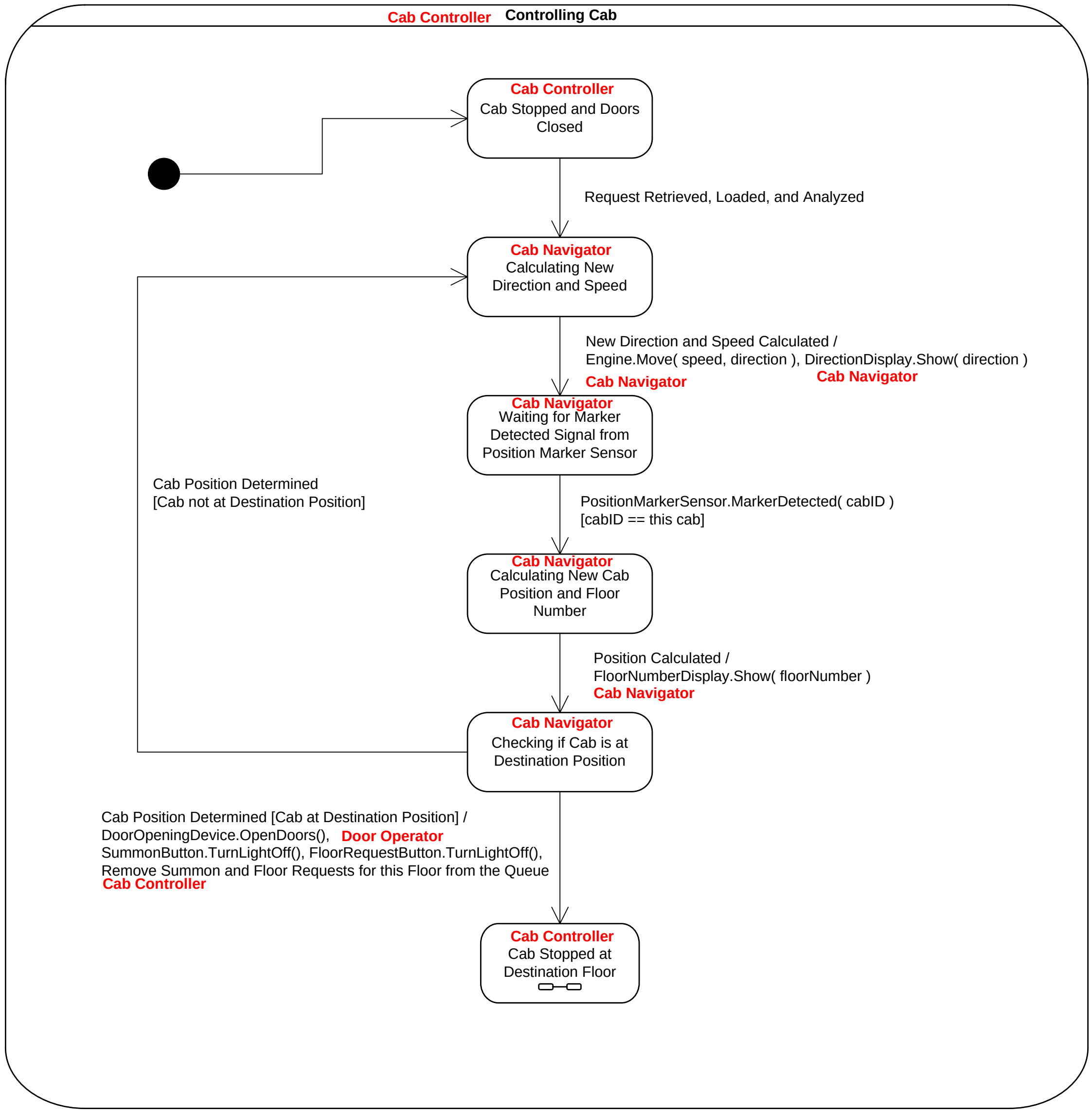


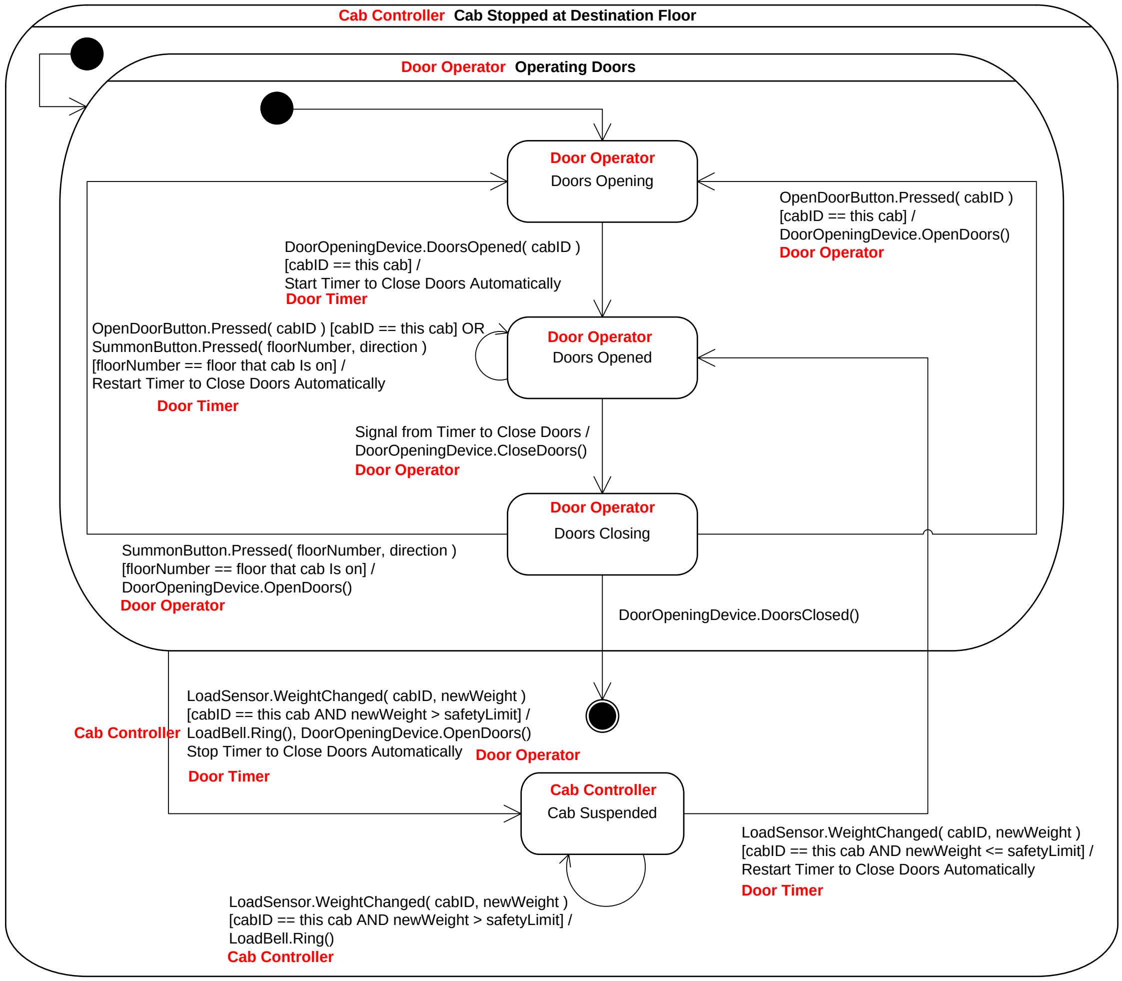
3.1.1.3 System State Diagrams with Concepts

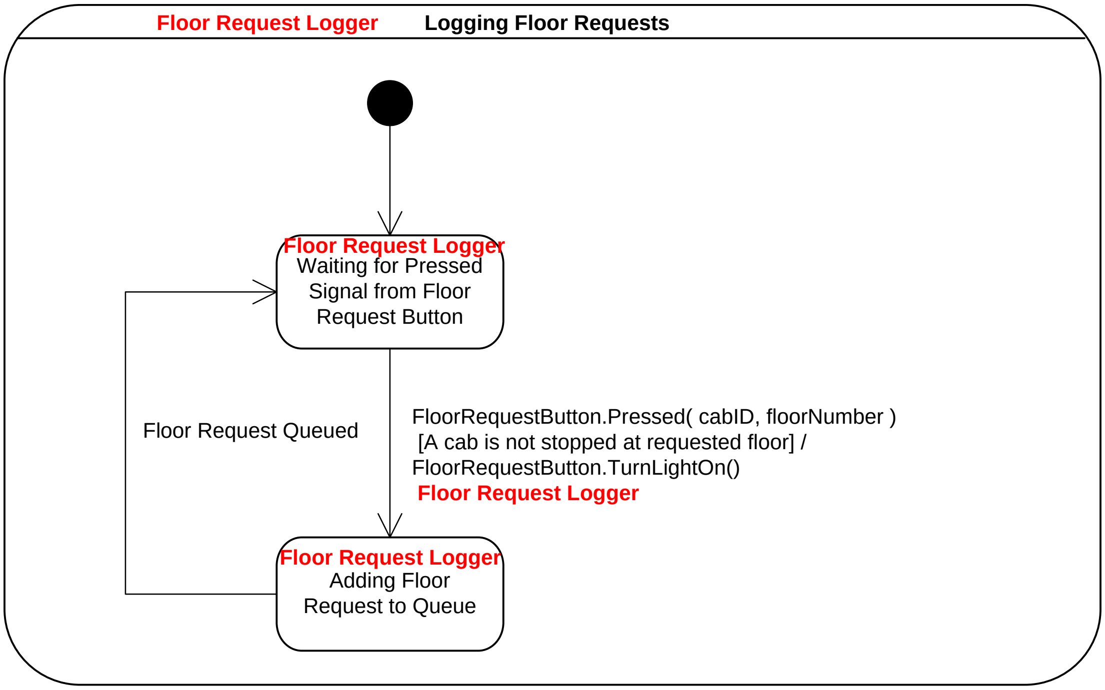


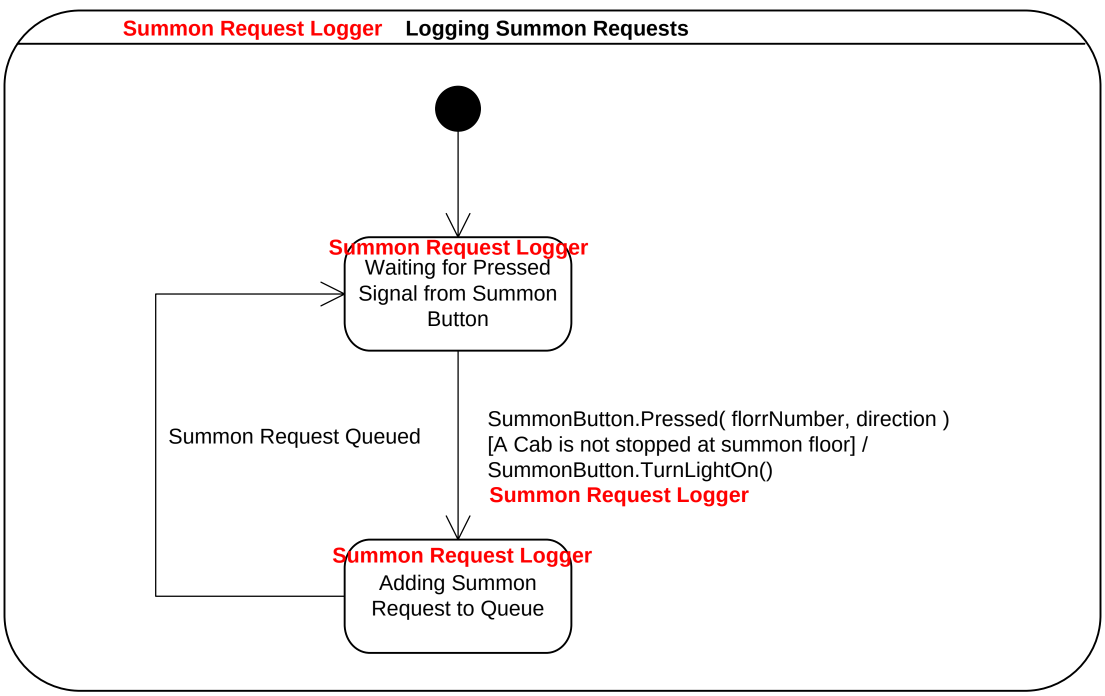

Elevator Operator Operating Elevator
Request Dispatcher
Dispatching Requests
Floor Request Logger
Logging Floor Requests
Summon Request Logger
Logging Summon Requests
Service Switch Handler
Handling Service Switch Mode Toggle
Emergency Stop Handler
Handling Emergency Stop Button Presses
Emergency Bell Handler
Handling Emergency Bell Button Presses

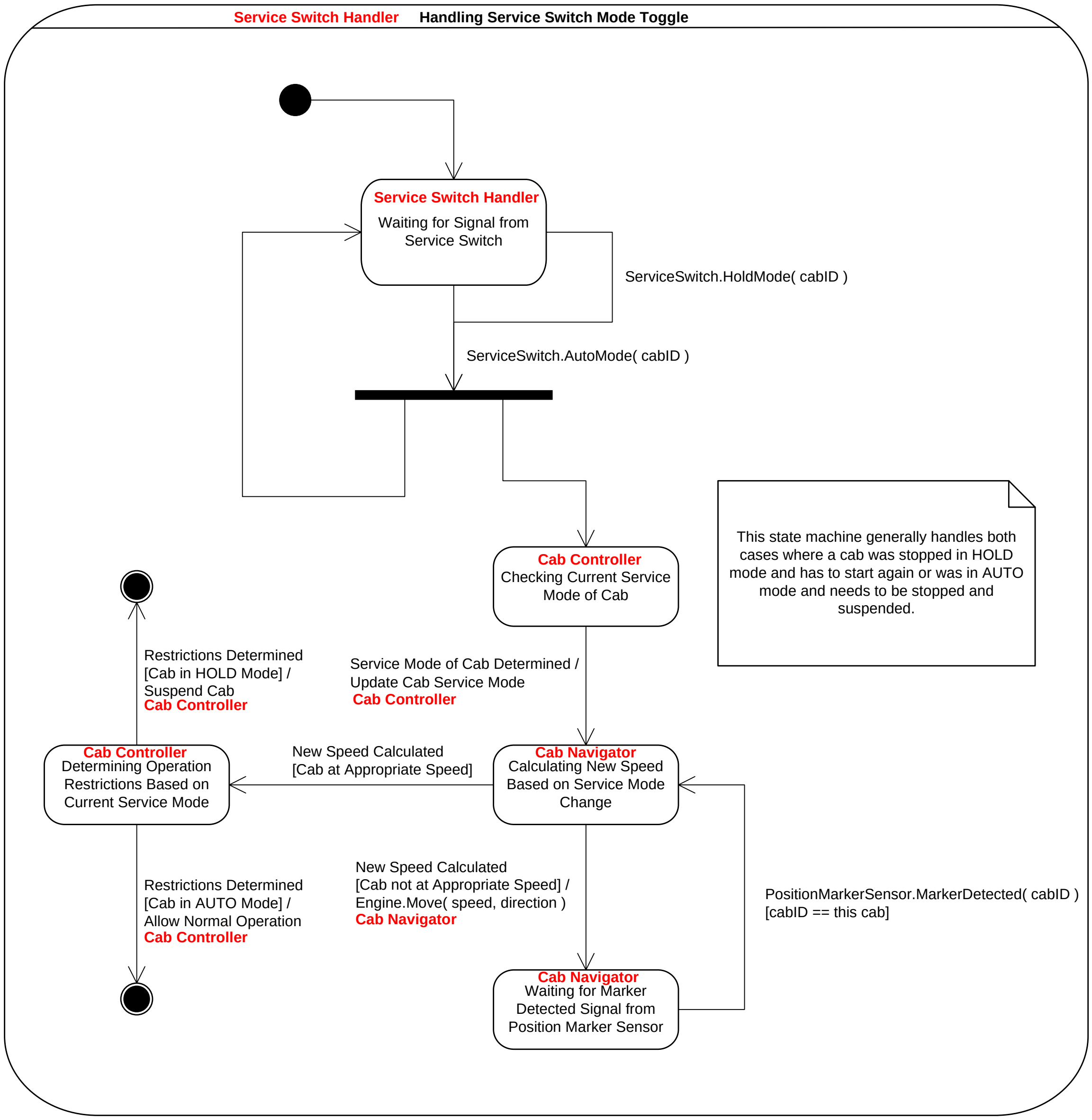

Request Dispatcher Dispatching Requests
Request Dispatcher
Checking Queue for Requests
Queue Checked
[Queue is Empty]
Request Dispatcher
Waiting for Requests
Queue Checked
[Queue not Empty]
Request Dispatcher
Determining Next Request To Process from Queue
Request Received
Next Request Determined
Request Dispatcher
Determining Best Cab to Answer Request
Best Cab Determined
Since there are two cabs available, one cab can answer a request while the controller finds another request for the other cab. This is achieved by having a separate thread for the control of each cab, created by this fork.
Cab Controller
Controlling Cab


Cab Controller Controlling Cab
Cab Controller
Cab Stopped and Doors Closed
Request Retrieved, Loaded, and Analyzed
Cab Navigator
Calculating New Direction and Speed
New Direction and Speed Calculated /
Engine.Move(speed, direction), DirectionDisplay.Show(direction)
Cab Navigator
Cab Navigator
Cab Navigator
Waiting for Marker Detected Signal from Position Marker Sensor
Cab Position Determined
[Cab not at Destination Position]
PositionMarkerSensor.MarkerDetected(cabID)
[cabID == this cab]
Cab Navigator
Calculating New Cab Position and Floor Number
Position Calculated /
FloorNumberDisplay.Show(floorNumber)
Cab Navigator
Cab Navigator
Checking if Cab is at Destination Position
Cab Position Determined [Cab at Destination Position] /
DoorOpeningDevice.OpenDoors(), Door Operator
SummonButton.TurnLightOff(), FloorRequestButton.TurnLightOff(),
Remove Summon and Floor Requests for this Floor from the Queue
Cab Controller
Cab Controller
Cab Stopped at Destination Floor

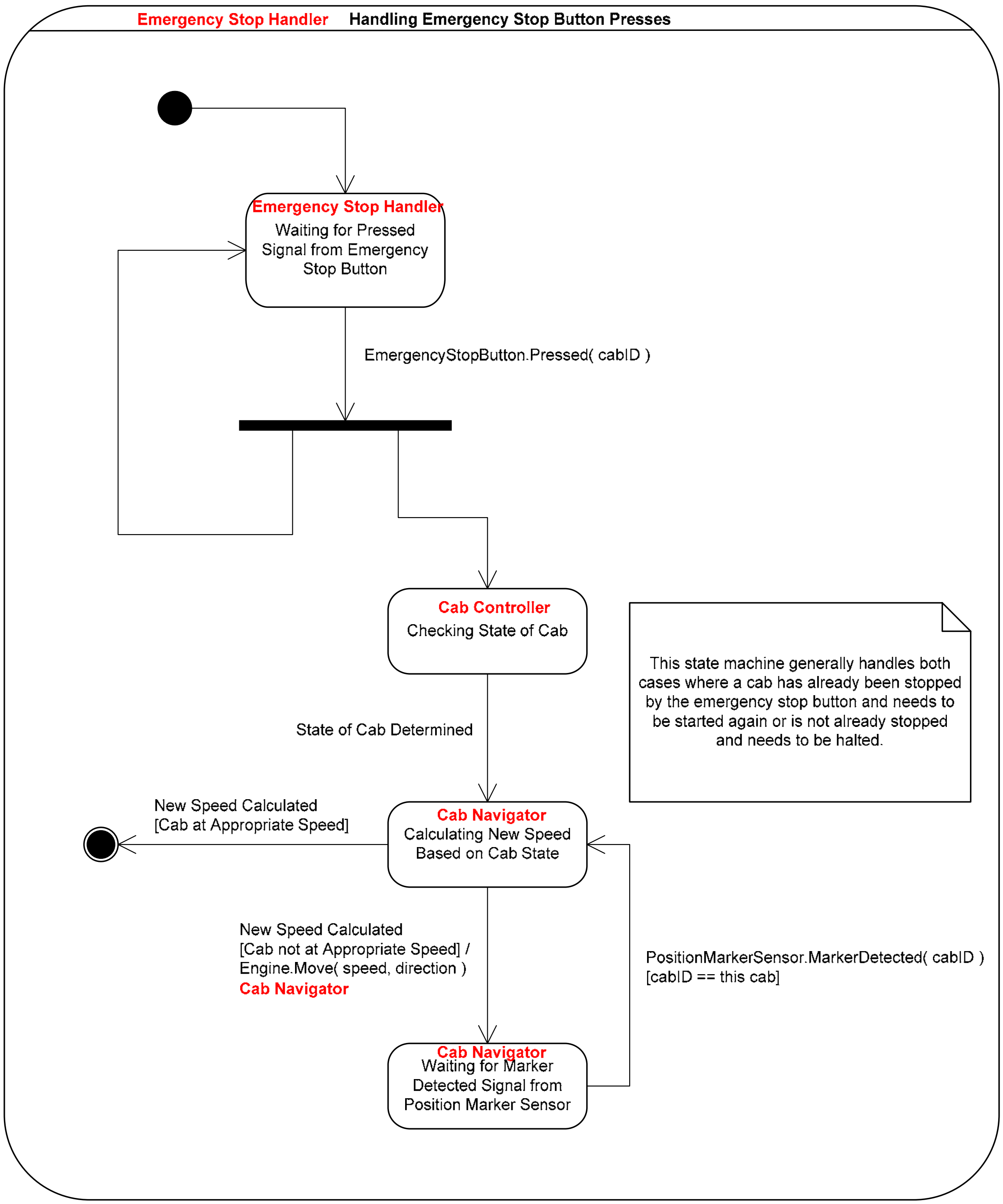

Cab Controller Cab Stopped at Destination Floor
Door Operator Operating Doors
Door Operator
Doors Opening
OpenDoorButton.Pressed(cabID)
[cabID == this cab] /
DoorOpeningDevice.OpenDoors()
Door Operator
DoorOpeningDevice.DoorsOpened(cabID)
[cabID == this cab] /
Start Timer to Close Doors Automatically
Door Timer
OpenDoorButton.Pressed(cabID) [cabID == this cab] OR
SummonButton.Pressed(floorNumber, direction)
[floorNumber == floor that cab Is on] /
Restart Timer to Close Doors Automatically
Door Timer
Door Operator
Doors Opened
Signal from Timer to Close Doors /
DoorOpeningDevice.CloseDoors()
Door Operator
Door Operator
Doors Closing
SummonButton.Pressed(floorNumber, direction)
[floorNumber == floor that cab Is on] /
DoorOpeningDevice.OpenDoors()
Door Operator
DoorOpeningDevice.DoorsClosed()
LoadSensor.WeightChanged(cabID, newWeight)
[cabID == this cab AND newWeight > safetyLimit] /
Cab Controller
LoadBell.Ring(), DoorOpeningDevice.OpenDoors()
Stop Timer to Close Doors Automatically
Door Operator
Door Timer
Cab Controller
Cab Suspended
LoadSensor.WeightChanged(cabID, newWeight)
[cabID == this cab AND newWeight <= safetyLimit] /
Restart Timer to Close Doors Automatically
Door Timer
LoadSensor.WeightChanged(cabID, newWeight)
[cabID == this cab AND newWeight > safetyLimit] /
LoadBell.Ring()
Cab Controller

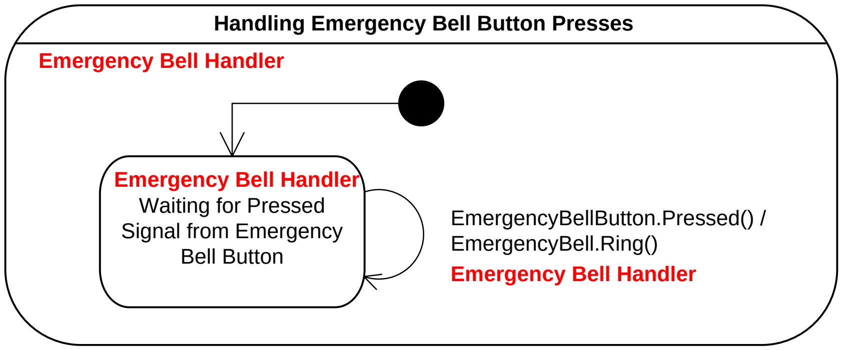

Floor Request Logger
Logging Floor Requests
Floor Request Logger
Waiting for Pressed Signal from Floor Request Button
Floor Request Queued
FloorRequestButton.Pressed(cabID, floorNumber)
[A cab is not stopped at requested floor] /
FloorRequestButton.TurnLightOn()
Floor Request Logger
Floor Request Logger
Adding Floor Request to Queue


Summon Request Logger
Logging Summon Requests
Summon Request Logger
Waiting for Pressed Signal from Summon Button
Summon Request Queued
SummonButton.Pressed(florrNumber, direction)
[A Cab is not stopped at summon floor] /
SummonButton.TurnLightOn()
Summon Request Logger
Summon Request Logger
Adding Summon Request to Queue



Service Switch Handler
Handling Service Switch Mode Toggle
Service Switch Handler
Waiting for Signal from Service Switch
ServiceSwitch.HoldMode(cabID)
ServiceSwitch.AutoMode(cabID)
This state machine generally handles both cases where a cab was stopped in HOLD mode and has to start again or was in AUTO mode and needs to be stopped and suspended.
Cab Controller
Checking Current Service Mode of Cab
Restrictions Determined [Cab in HOLD Mode] / Suspend Cab
Cab Controller
Service Mode of Cab Determined / Update Cab Service Mode
Cab Controller
Cab Controller
Determining Operation Restrictions Based on Current Service Mode
New Speed Calculated [Cab at Appropriate Speed]
Cab Navigator
Calculating New Speed Based on Service Mode Change
PositionMarkerSensor.MarkerDetected(cabID) [cabID == this cab]
New Speed Calculated [Cab not at Appropriate Speed] / Engine.Move(speed, direction)
Cab Navigator
Restrictions Determined [Cab in AUTO Mode] / Allow Normal Operation
Cab Controller
Cab Navigator
Waiting for Marker Detected Signal from Position Marker Sensor


Emergency Stop Handler
Handling Emergency Stop Button Presses
Emergency Stop Handler
Waiting for Pressed Signal from Emergency Stop Button
EmergencyStopButton.Pressed(cabID)
Cab Controller
Checking State of Cab
This state machine generally handles both cases where a cab has already been stopped by the emergency stop button and needs to be started again or is not already stopped and needs to be halted.
State of Cab Determined
New Speed Calculated
[Cab at Appropriate Speed]
Cab Navigator
Calculating New Speed Based on Cab State
New Speed Calculated
[Cab not at Appropriate Speed] /
Engine.Move(speed, direction)
Cab Navigator
PositionMarkerSensor.MarkerDetected(cabID)
[cabID == this cab]
Cab Navigator
Waiting for Marker Detected Signal from Position Marker Sensor


Handling Emergency Bell Button Presses
Emergency Bell Handler
Emergency Bell Handler
Waiting for Pressed Signal from Emergency Bell Button
EmergencyBellButton.Pressed() /
EmergencyBell.Ring()
Emergency Bell Handler

3.1.1.4 System Collaboration Diagram

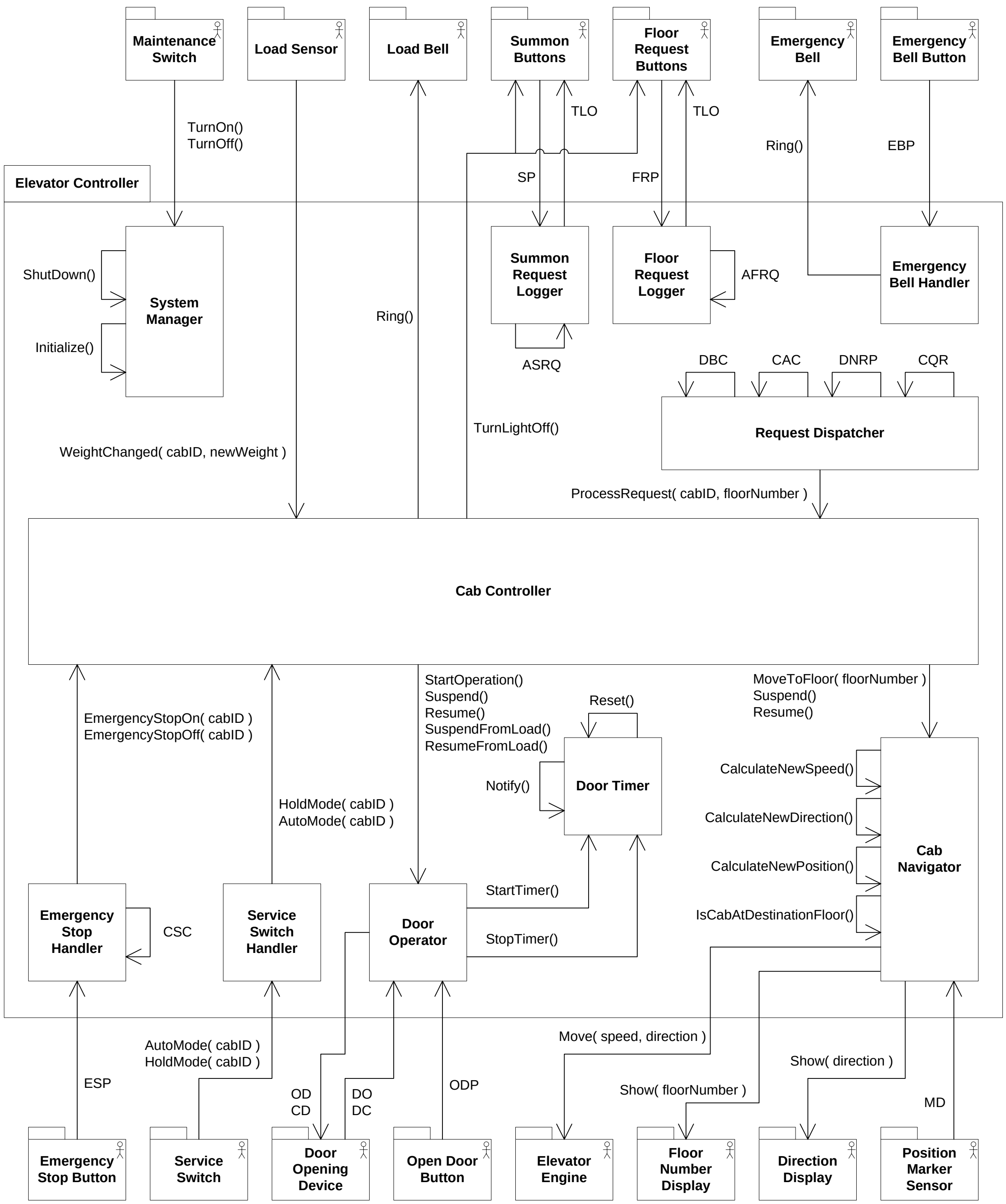
Figure 27: System Collaboration Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

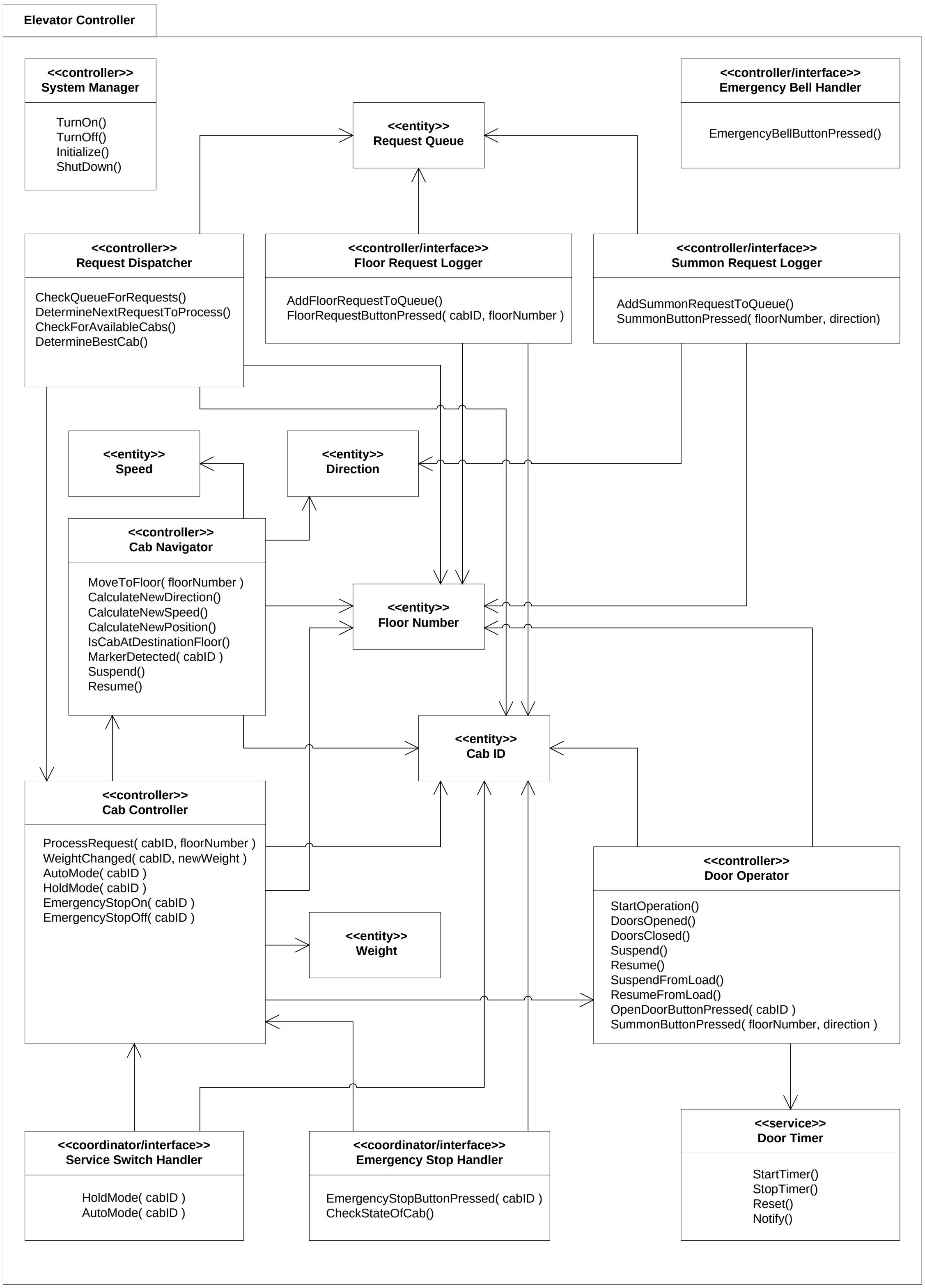
ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()



3.1.1.5 System Conceptual Diagram

Figure 28: Conceptual Diagram



3.1.2 Concept State Diagrams

Figure 29: System Manager State Diagram

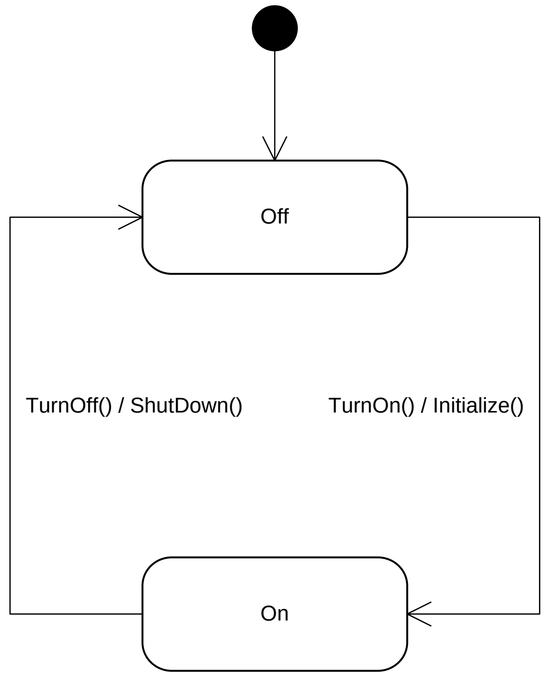


Figure 30: Summon Request Logger State Diagram

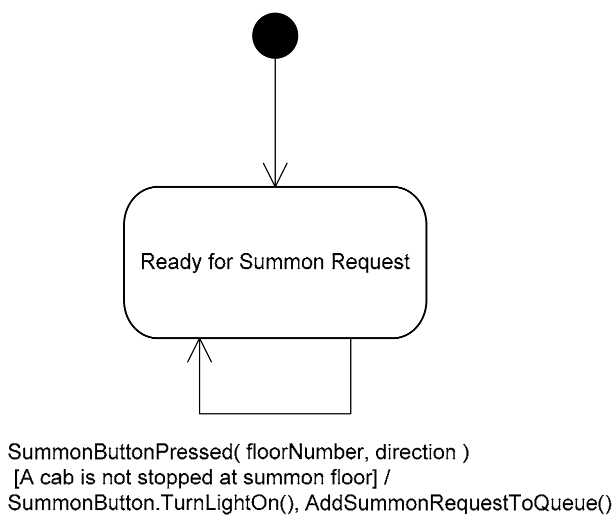


Figure 31: Floor Request Logger State Diagram

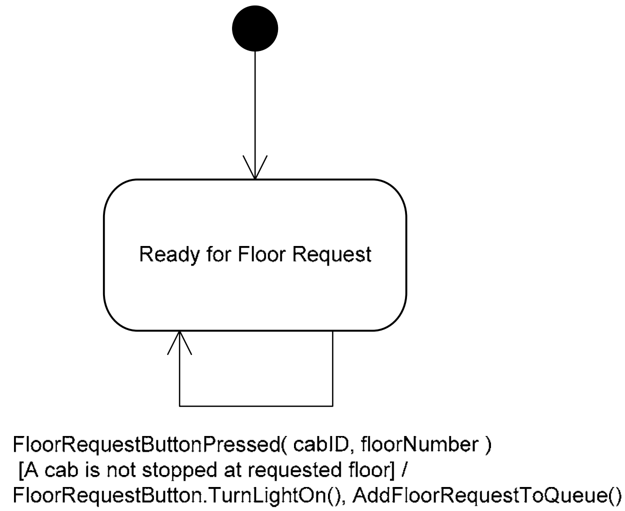


Figure 32: Request Dispatcher State Diagram

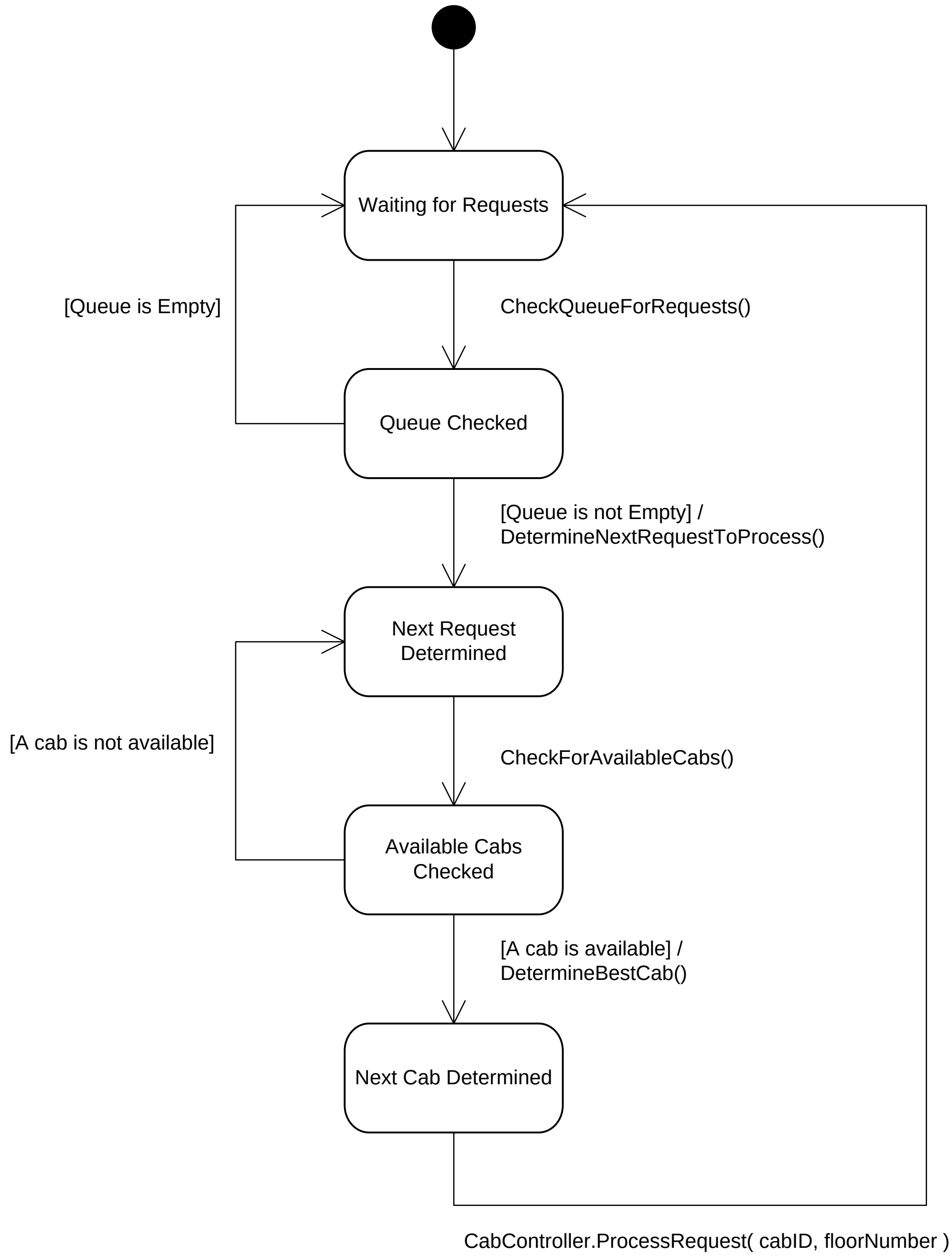


Figure 33: Cab Controller State Diagram

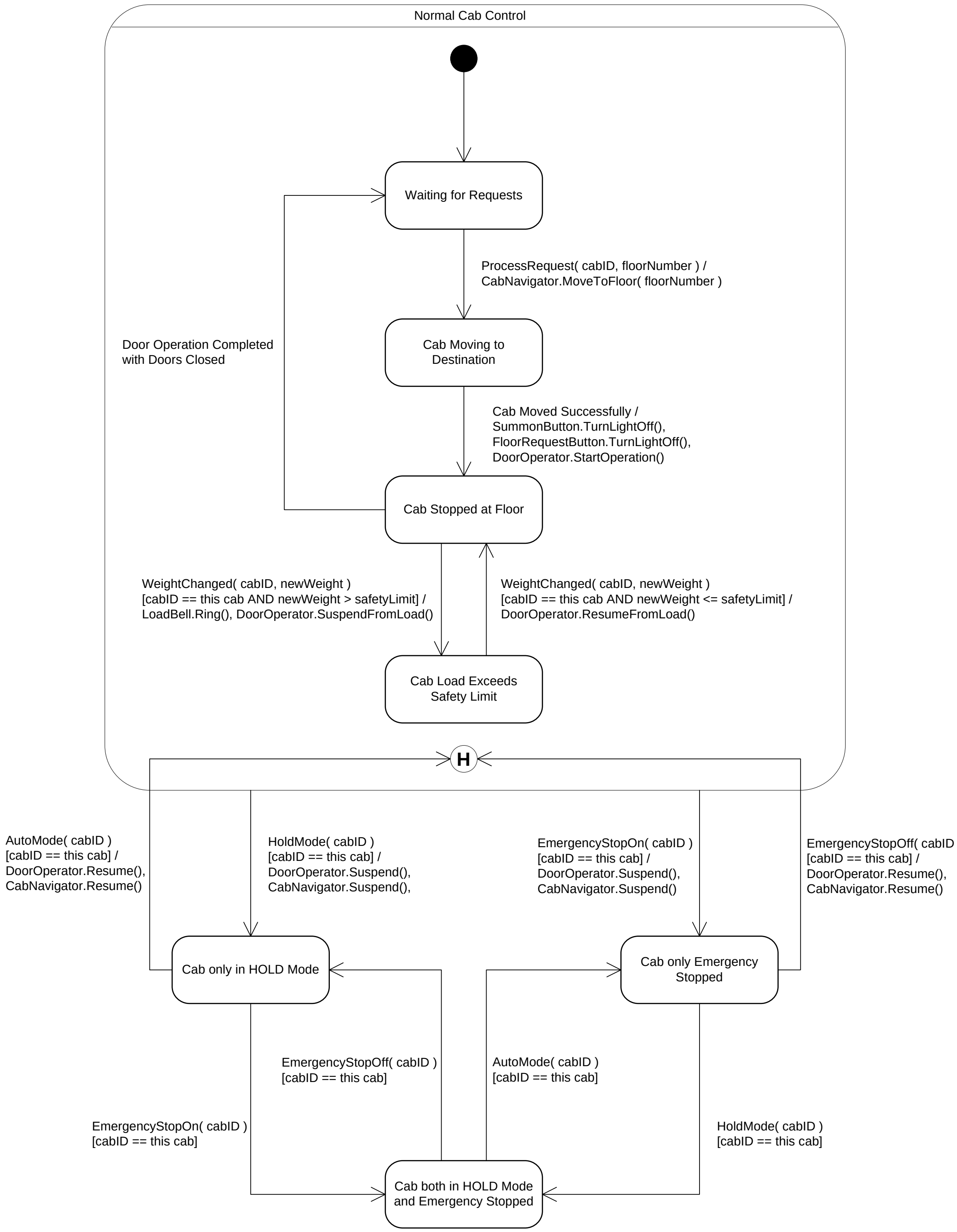


Figure 34: Cab Navigator State Diagram

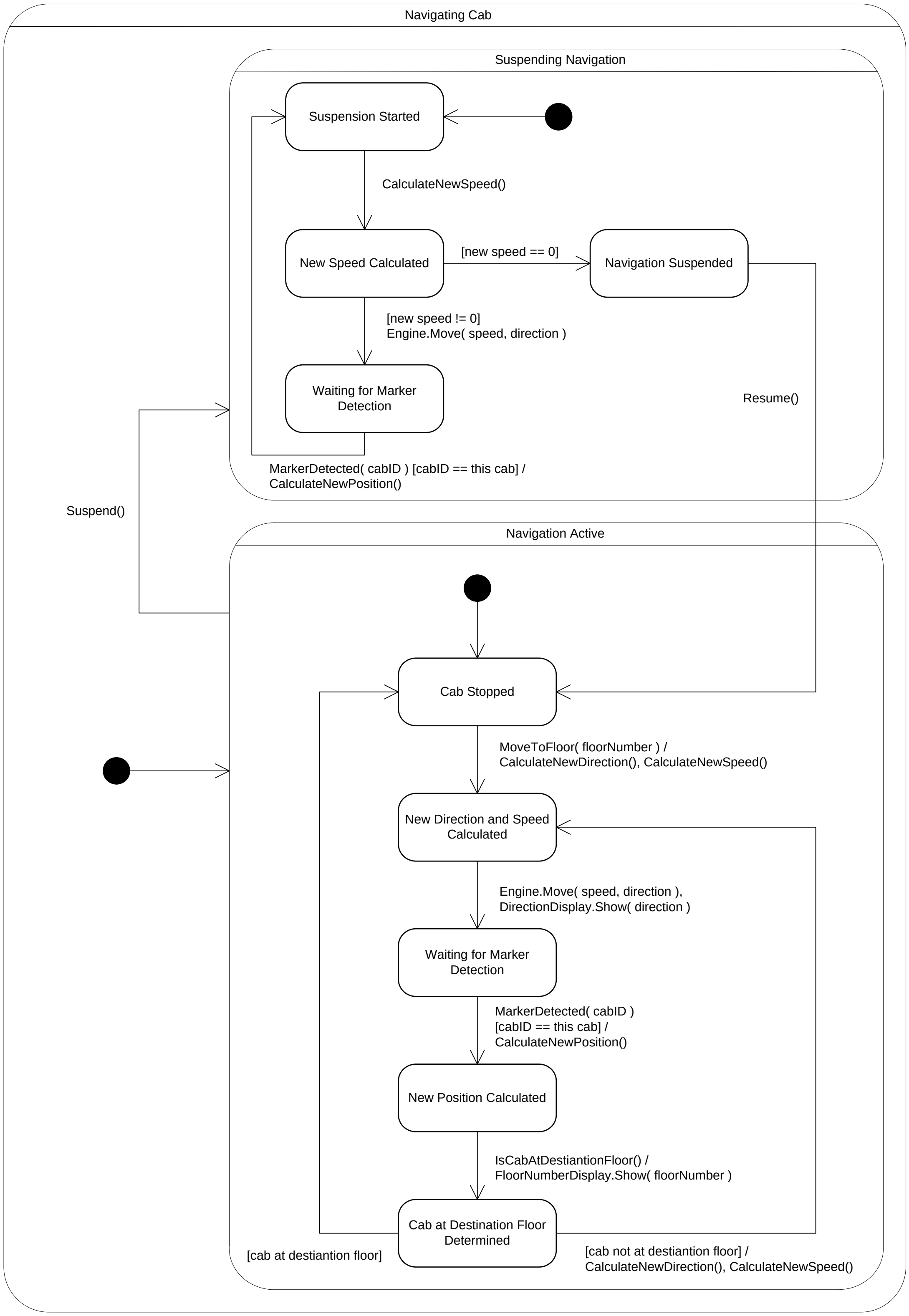


Figure 35: Door Operator State Diagram

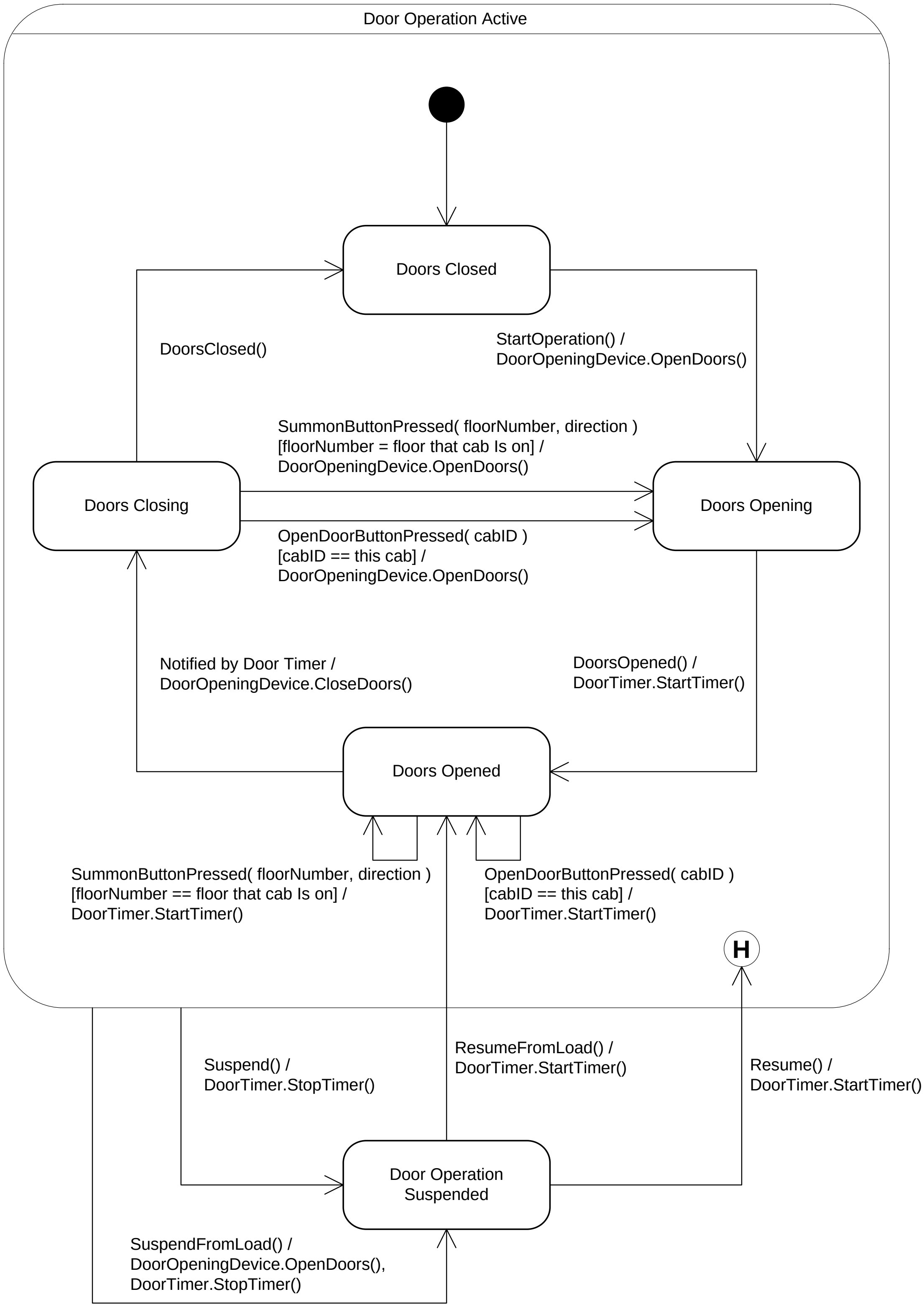


Figure 36: Door Timer State Diagram

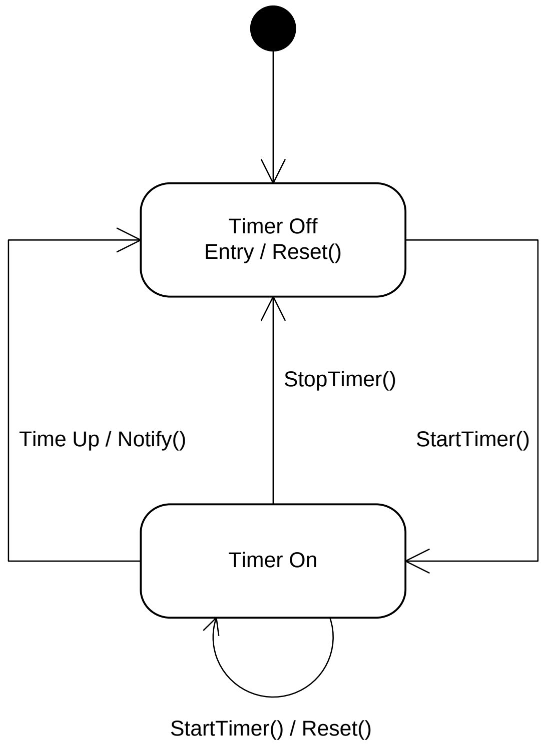


Figure 37: Service Switch Handler State Diagram

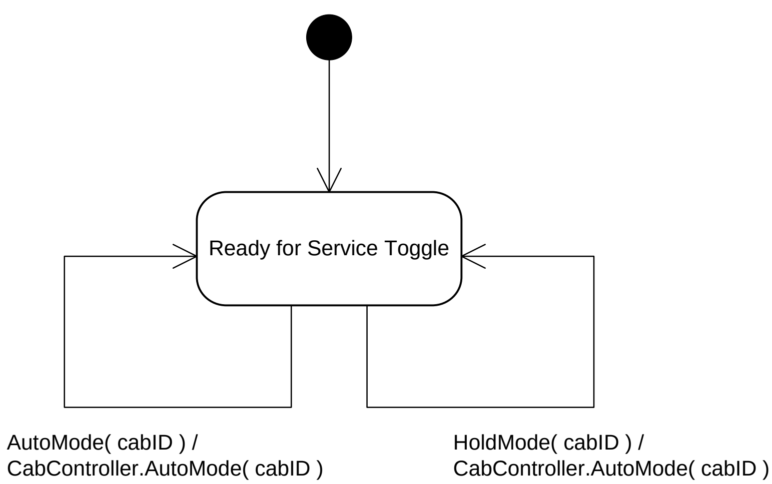


Figure 38: Emergency Stop Button Handler State Diagram

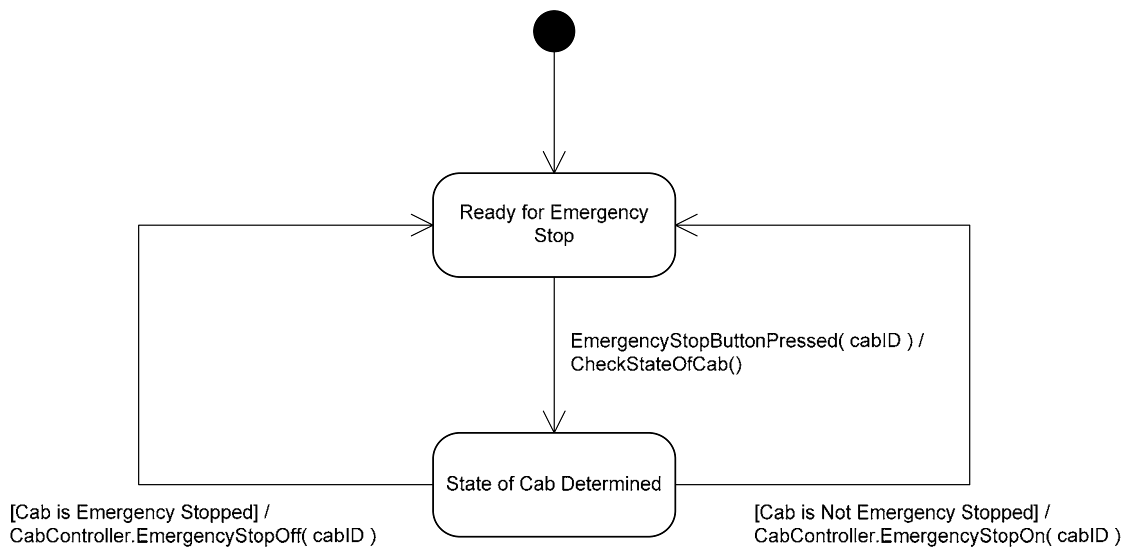
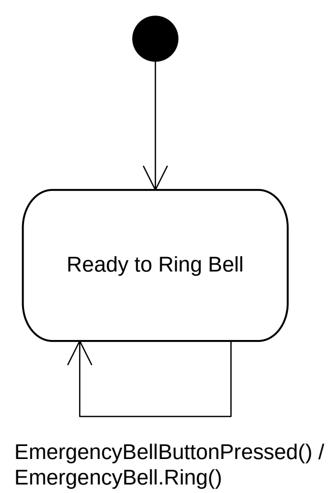


Figure 39: Emergency Bell Handler State Diagram



3.1.3 Collaboration Sequence Diagrams

Figure 40: Summon Button Pressed Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

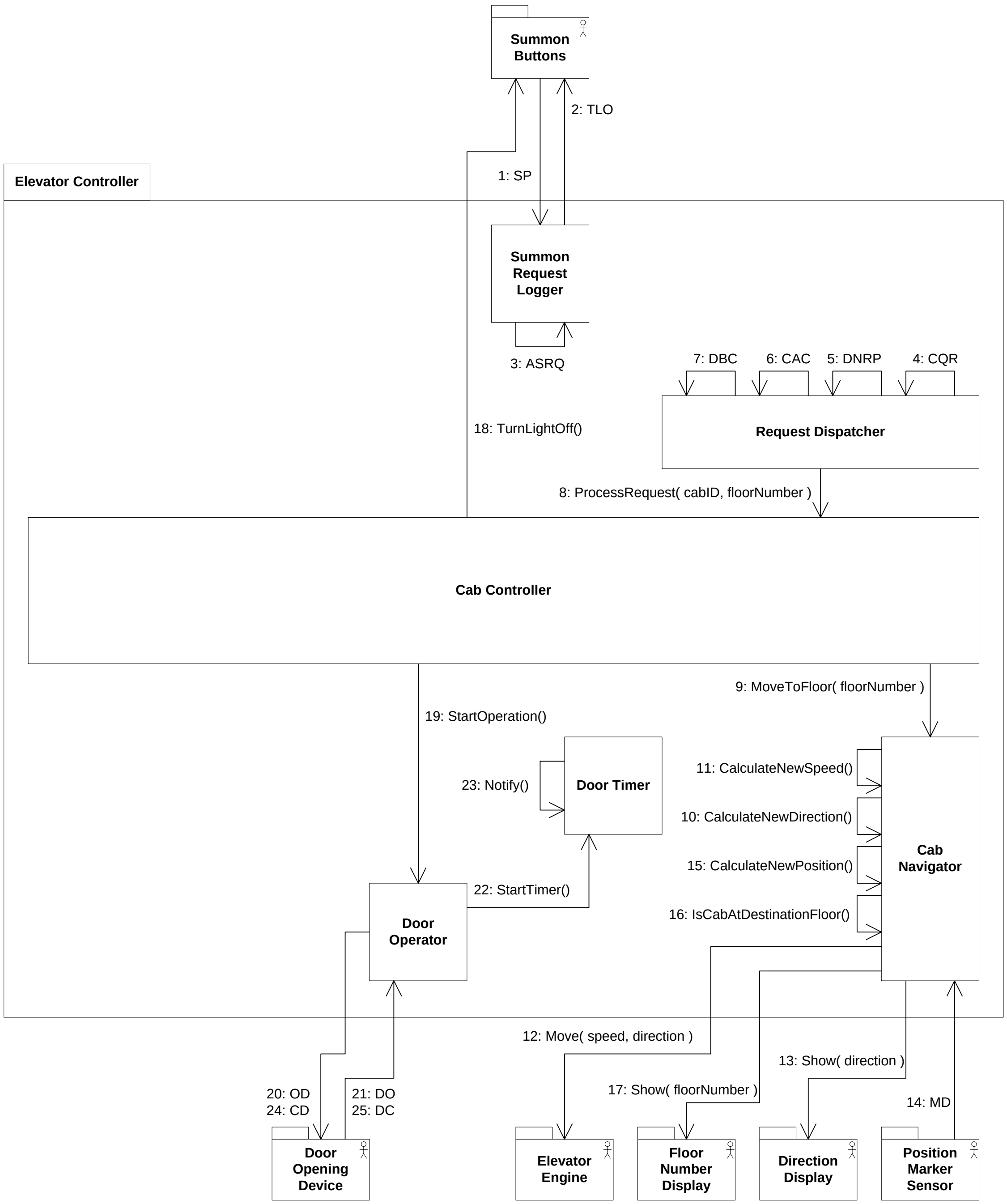


Figure 41: Floor Request Button Pressed Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

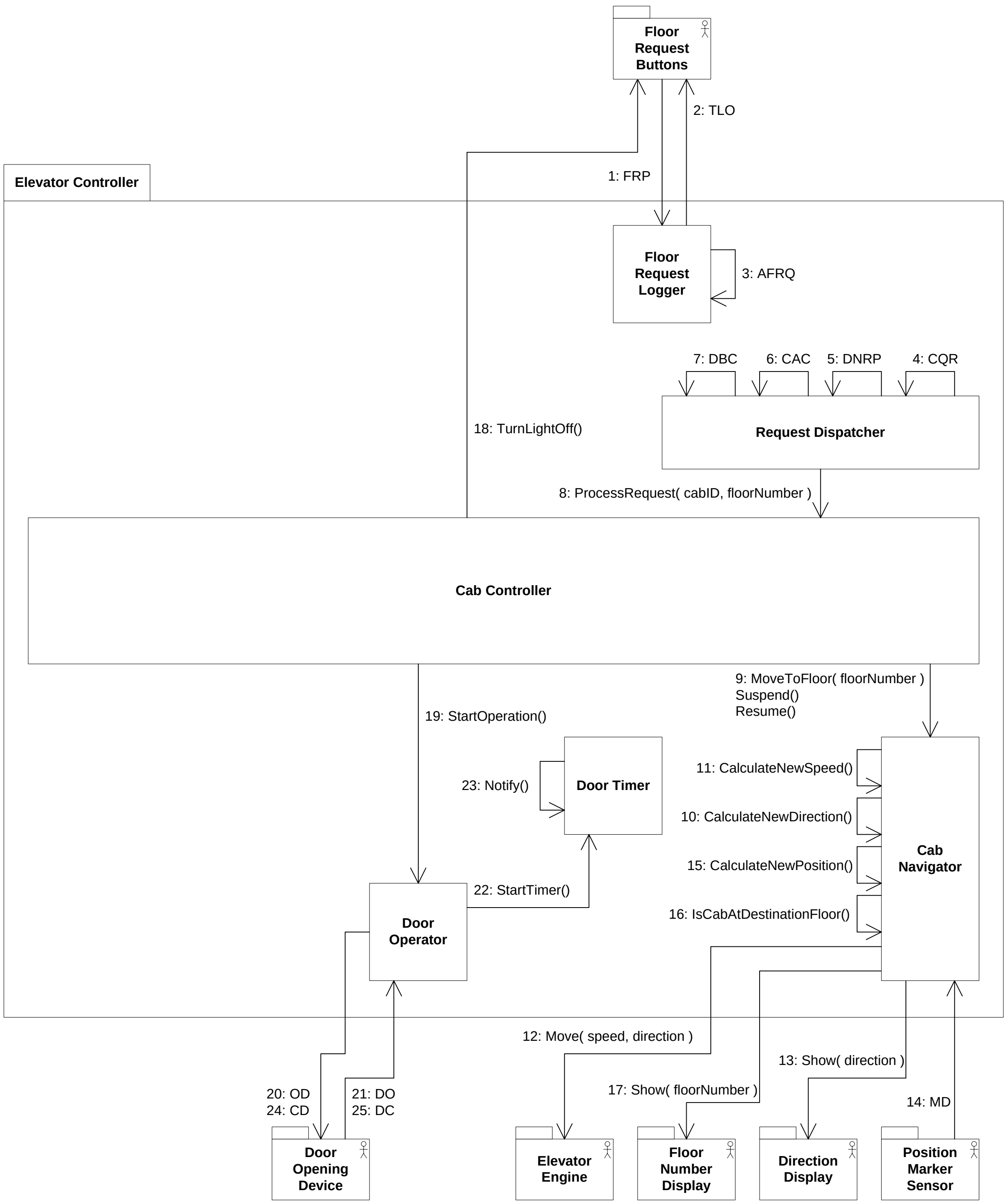


Figure 42: Open Door Button Pressed Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

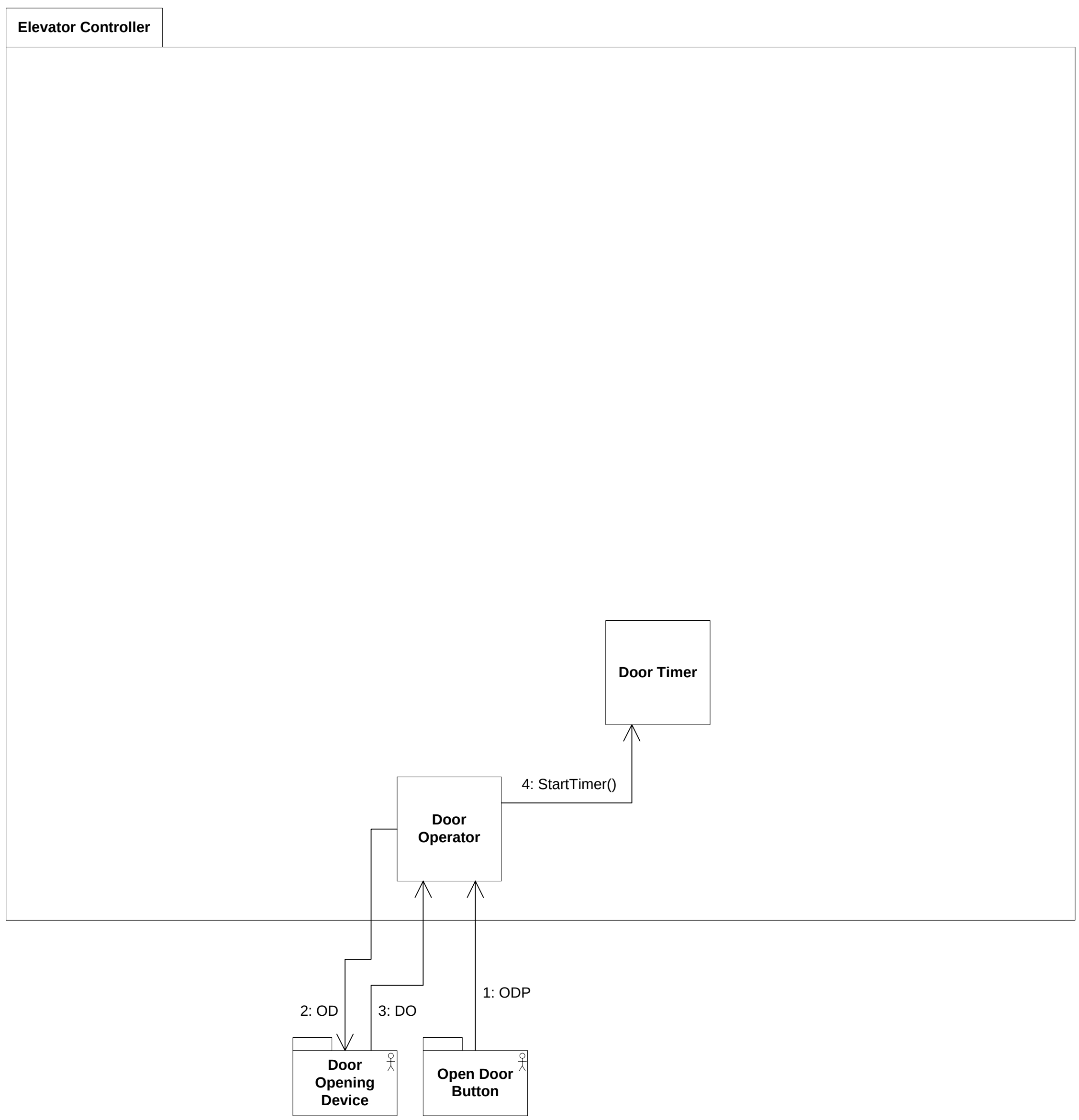


Figure 43: Cab Emergency Stopped Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

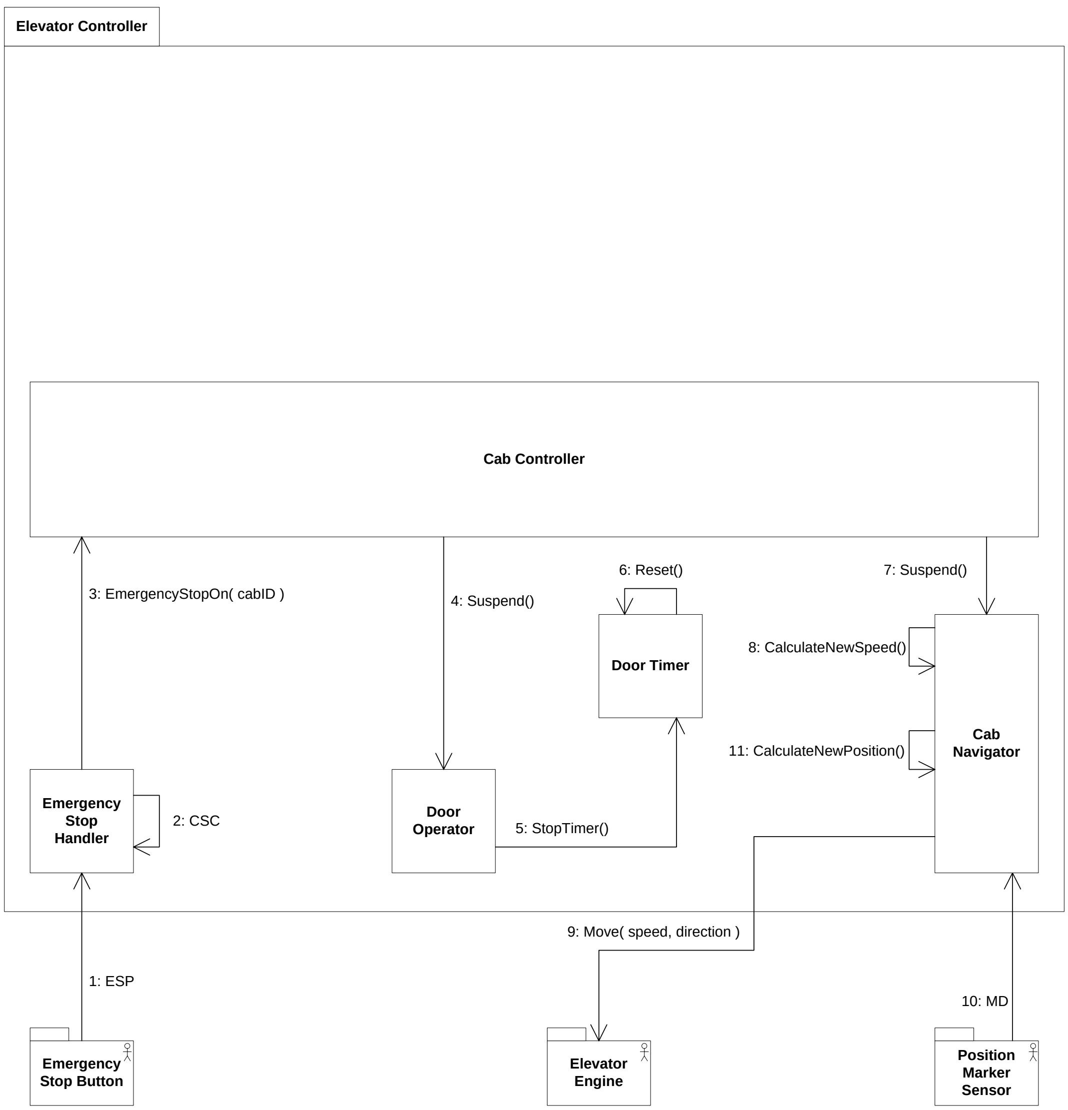


Figure 44: Cab Started from Emergency Stop Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

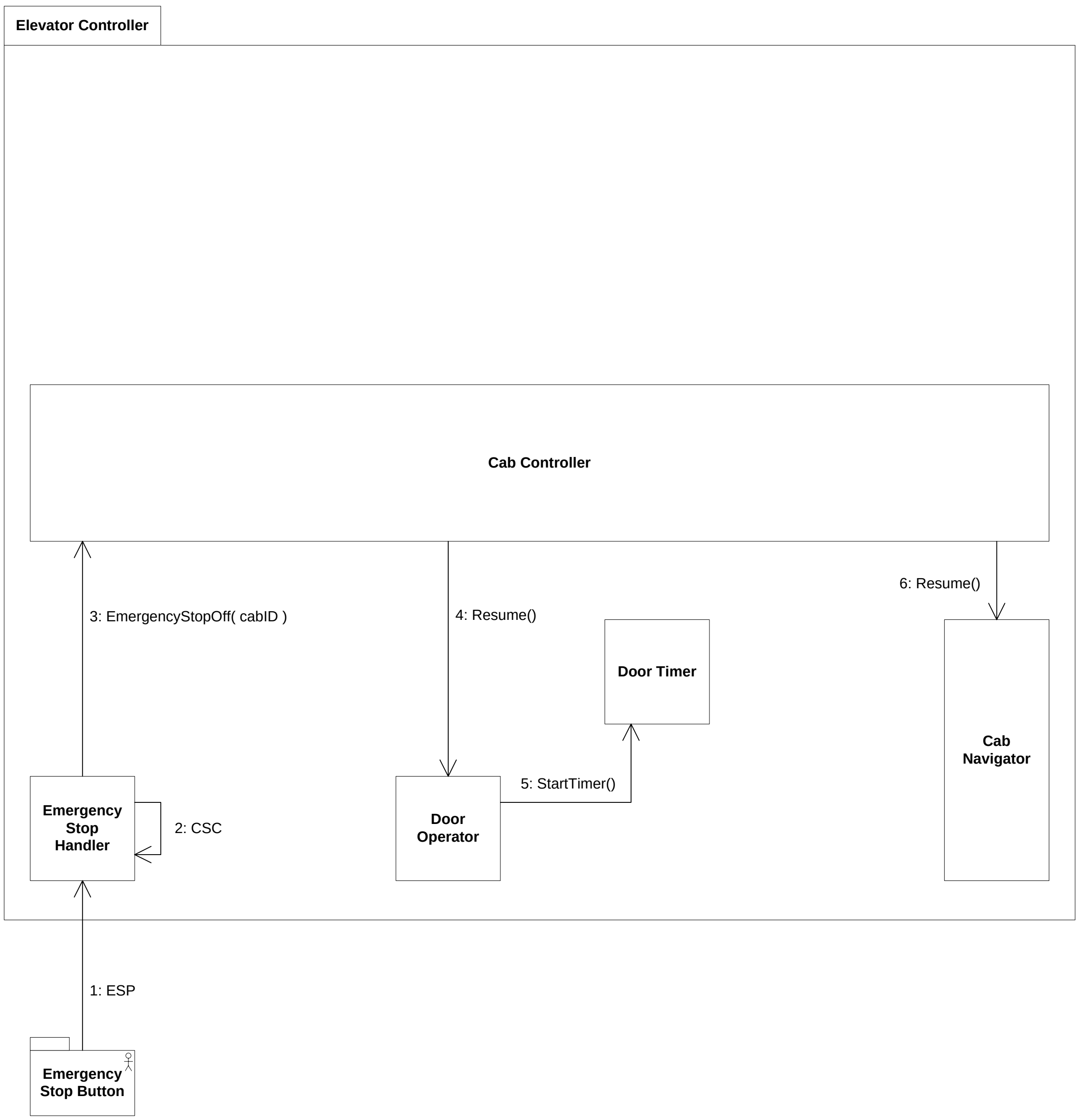


Figure 45: Emergency Bell Button Pressed Collaboration State Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

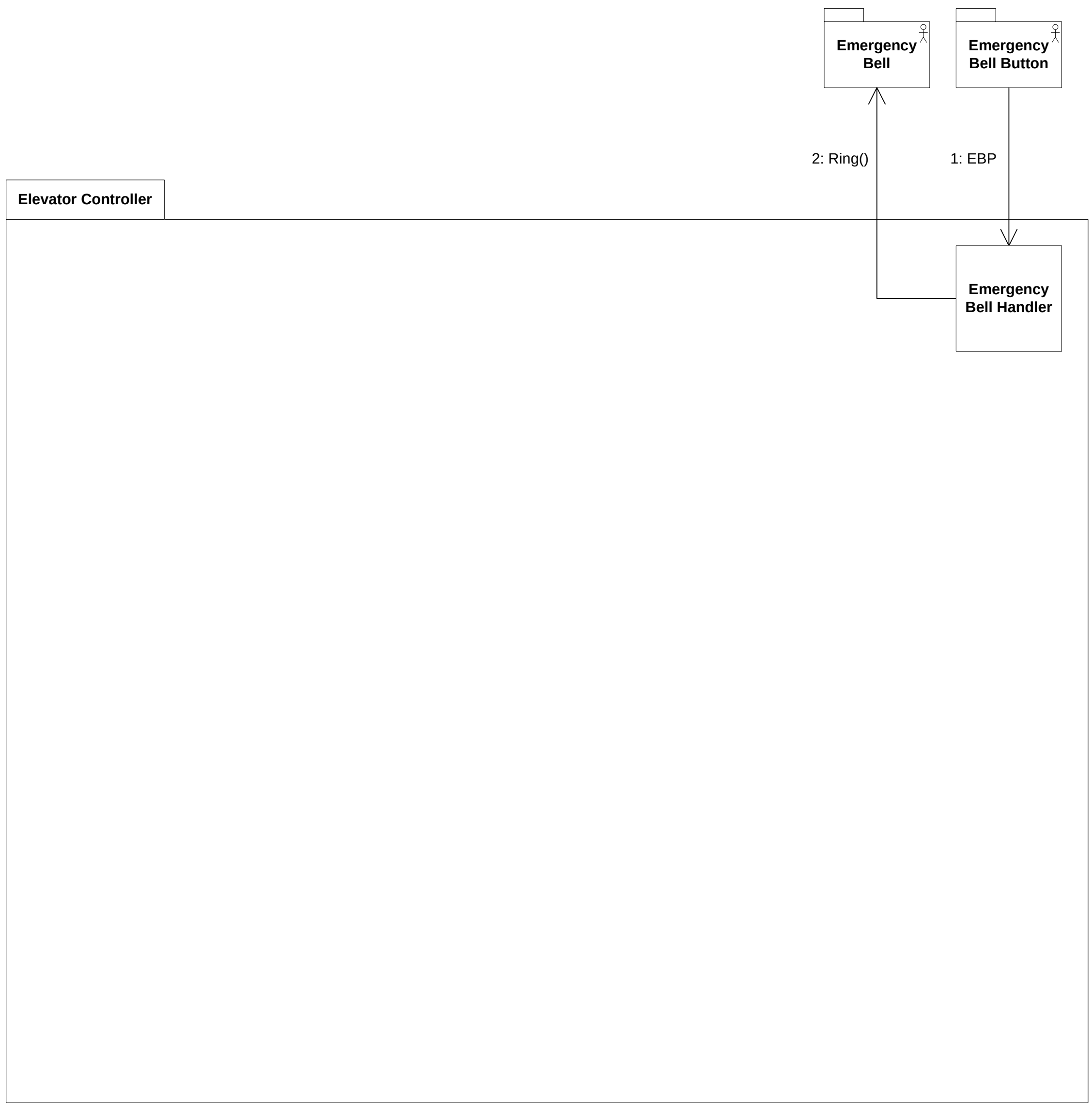


Figure 46: Service Switch Hold Mode Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)
ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)
CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

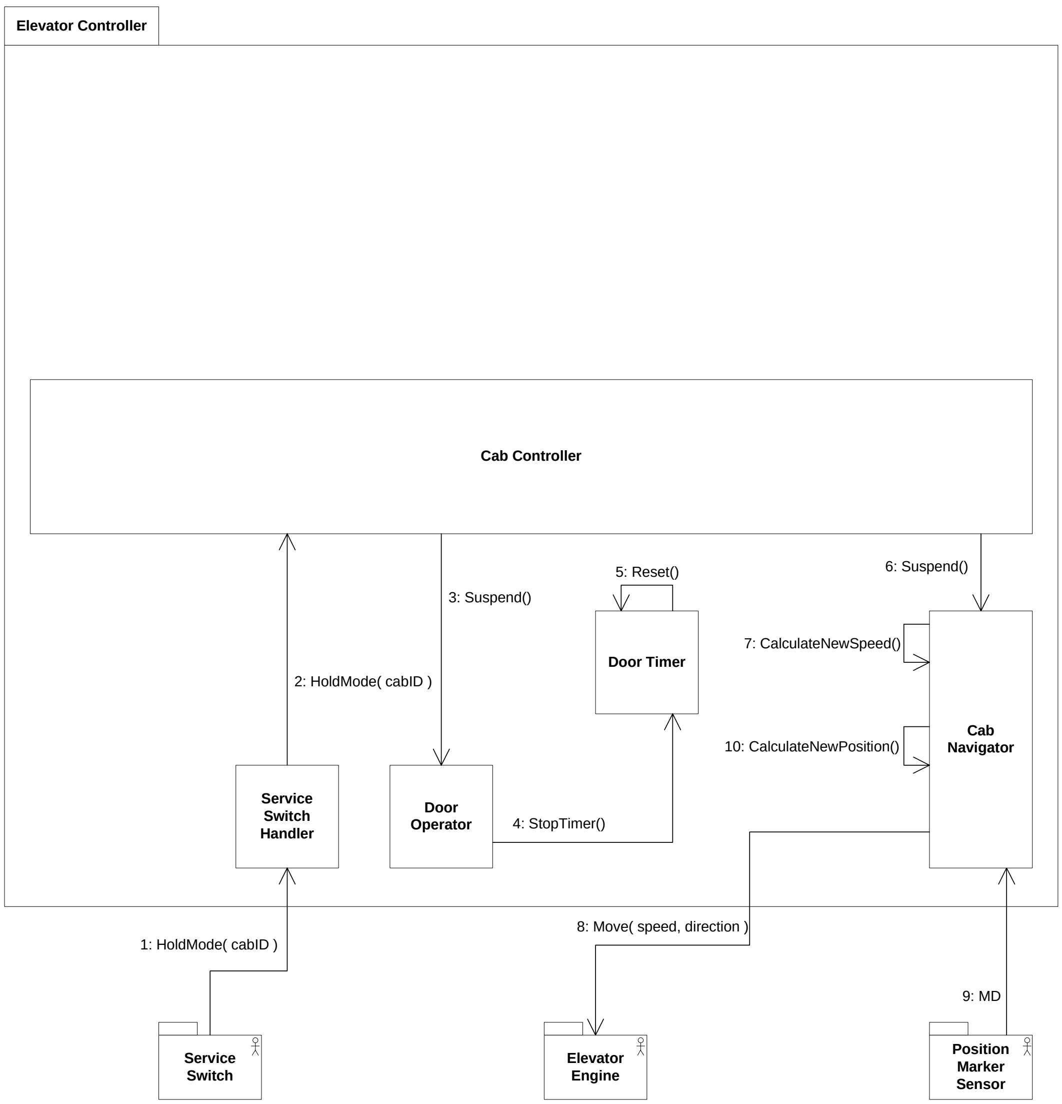


Figure 47: Service Switch Auto Mode Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

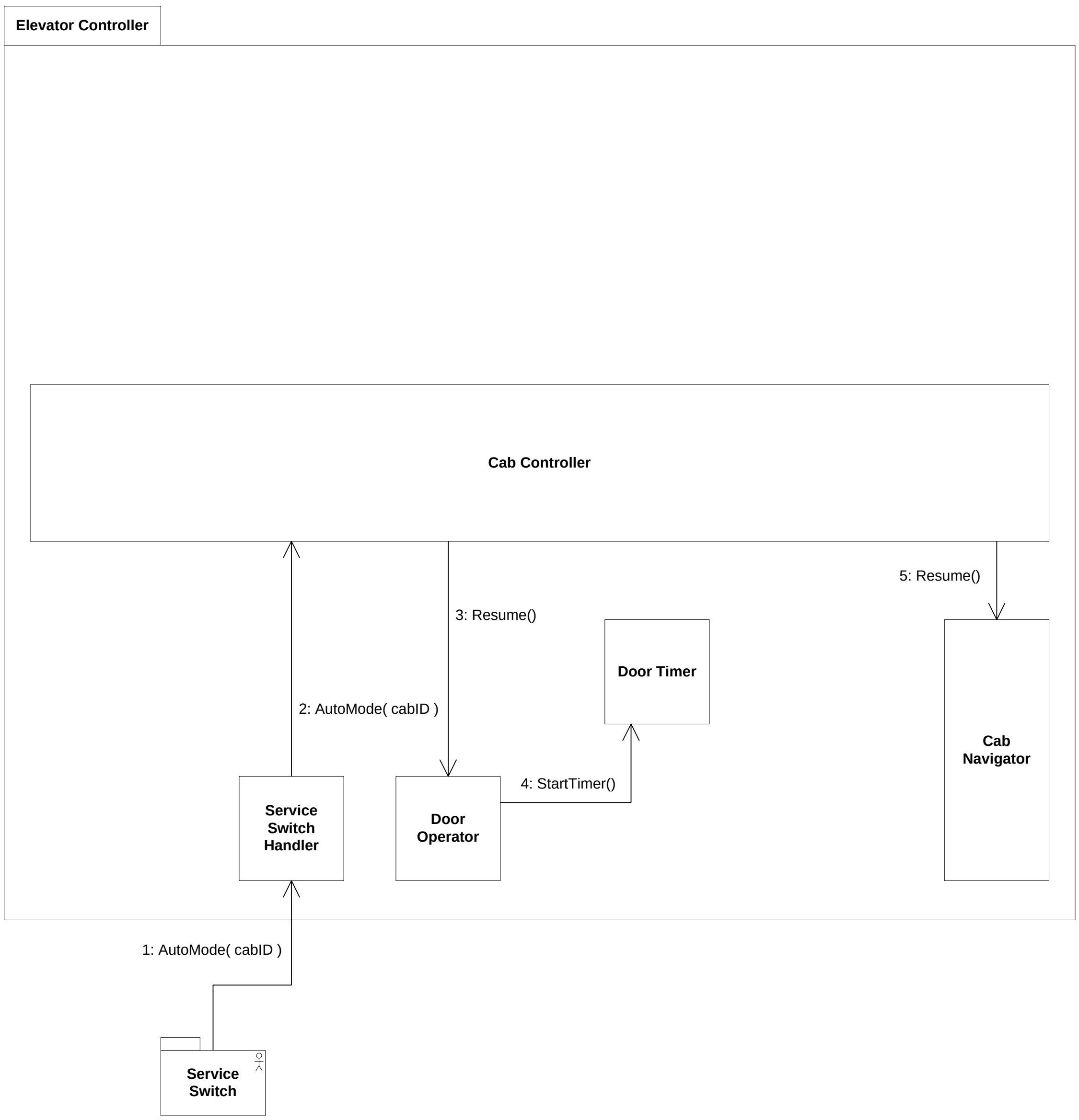


Figure 48: Load Sensor Collaboration Sequence Diagram

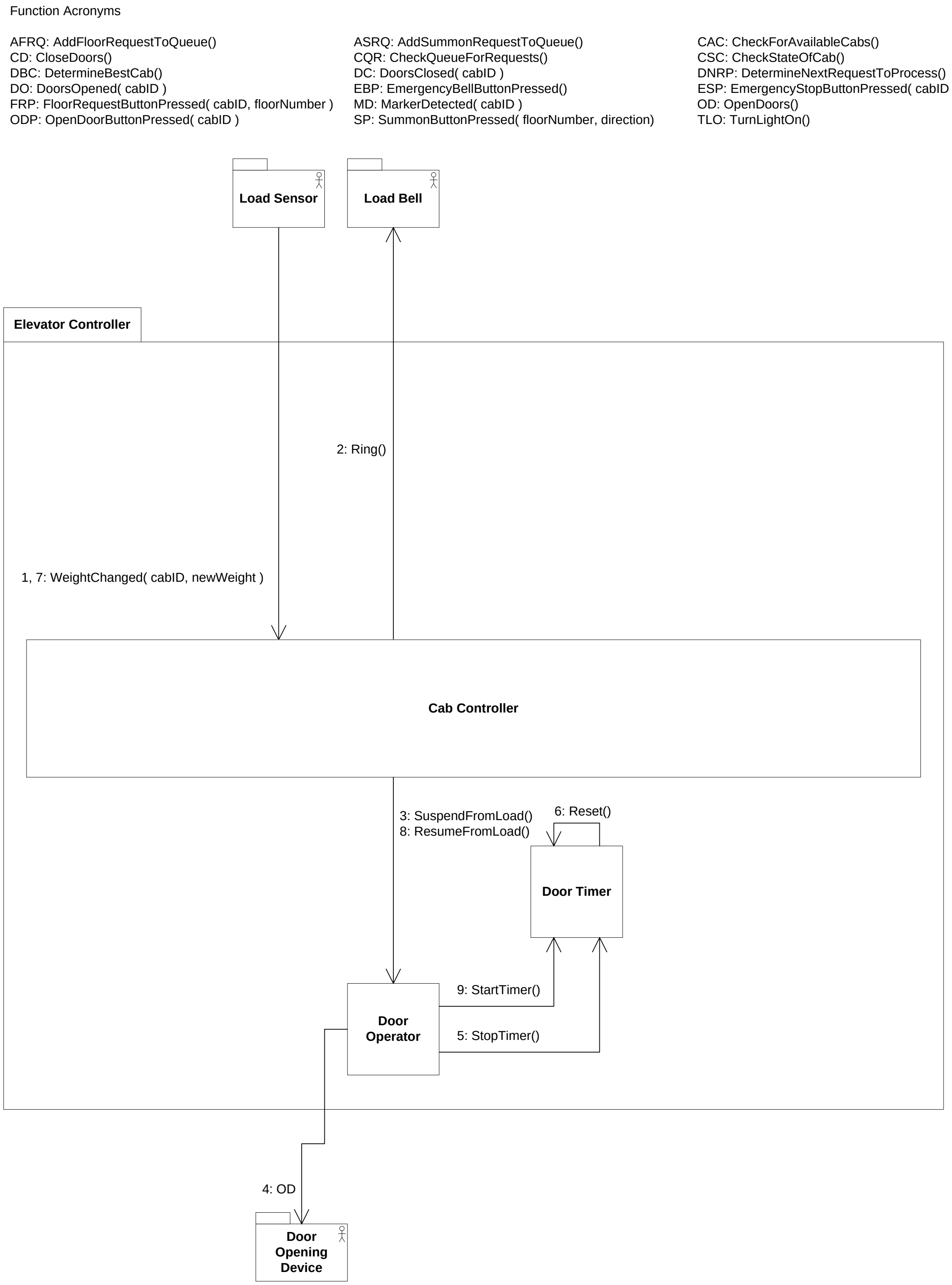


Figure 49: Maintenance Switch Off Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()

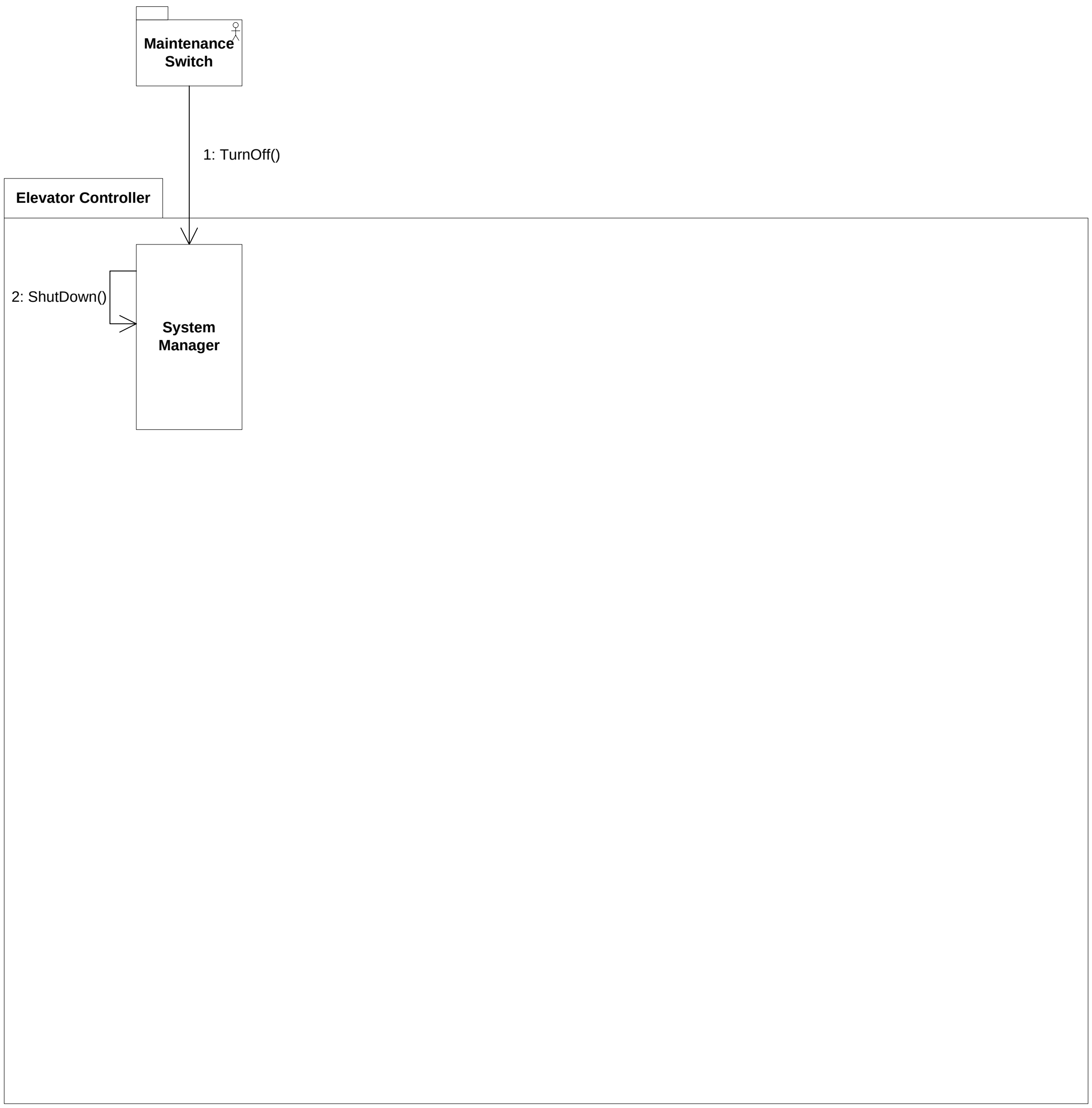


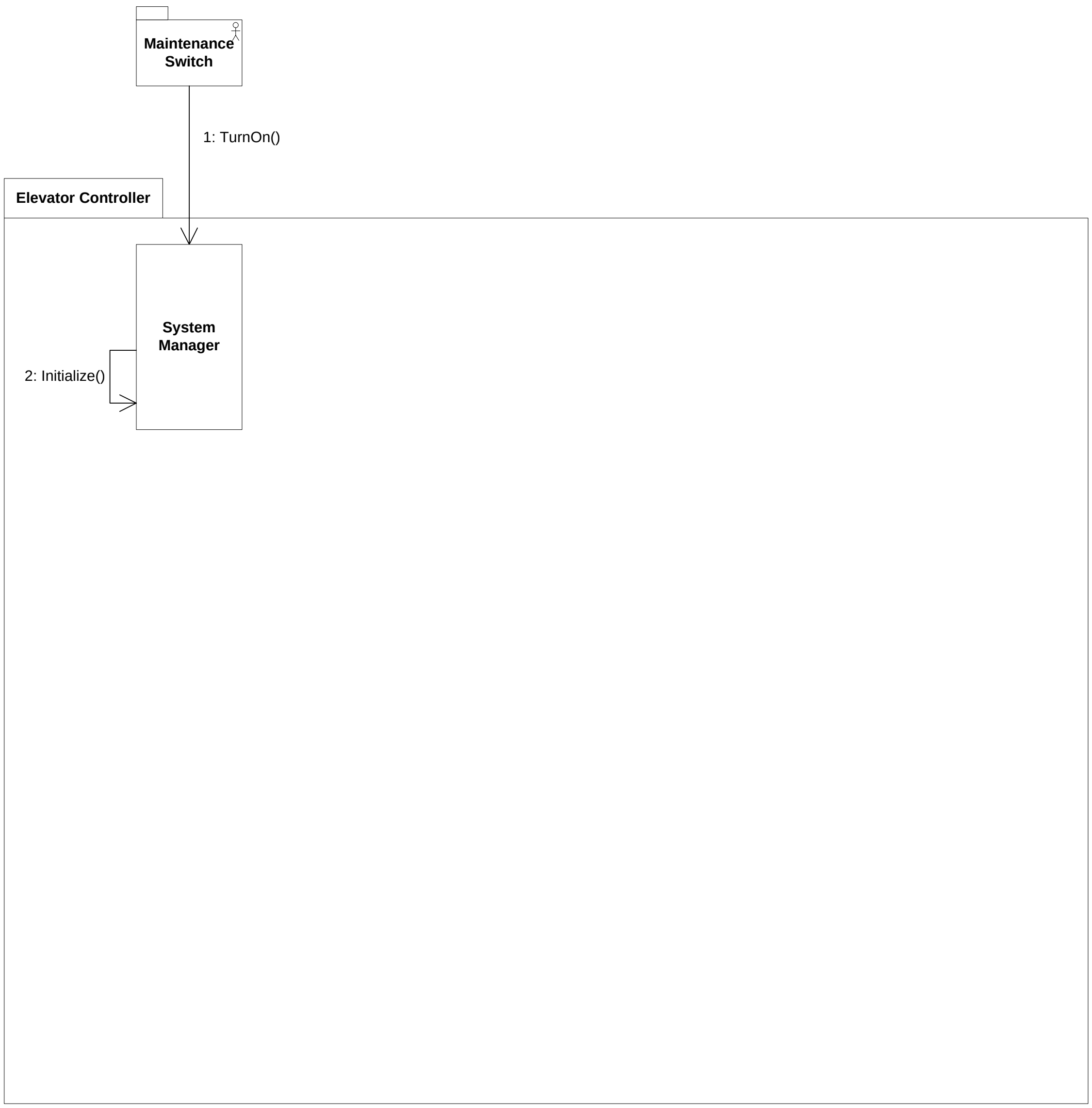
Figure 50: Maintenance Switch On Collaboration Sequence Diagram

Function Acronyms

AFRQ: AddFloorRequestToQueue()
CD: CloseDoors()
DBC: DetermineBestCab()
DO: DoorsOpened(cabID)
FRP: FloorRequestButtonPressed(cabID, floorNumber)
ODP: OpenDoorButtonPressed(cabID)

ASRQ: AddSummonRequestToQueue()
CQR: CheckQueueForRequests()
DC: DoorsClosed(cabID)
EBP: EmergencyBellButtonPressed()
MD: MarkerDetected(cabID)
SP: SummonButtonPressed(floorNumber, direction)

CAC: CheckForAvailableCabs()
CSC: CheckStateOfCab()
DNRP: DetermineNextRequestToProcess()
ESP: EmergencyStopButtonPressed(cabID)
OD: OpenDoors()
TLO: TurnLightOn()



3.2 External Interface Requirements

3.2.1 User Interfaces

Since the elevator controller is an embedded software system, it does not have a visual interface that provides prompts or feedback to its users.

3.2.2 Hardware Interface-Application Program Interface

This SRS only pertains to the requirements of the software of an elevator controller, not the hardware that it runs on. Therefore, the details of hardware interface requirements are out of the scope of this document. That being said, the elevator controller hardware must provide a means of sending and receiving communication signals from other components of the elevator system in order for the software to be able to function properly. These signals are listed in the section below. They are simply invoked or handled by the controller software based on if the signal is input or output, with respect to the controller.

3.2.3 Communications Interface

Table 1: Input Signals

Component	Signals
Maintenance Switch	TurnOn() TurnOff()
Summon Buttons	Pressed(floorNumber, direction) Released(floorNumber, direction)
Floor Request Buttons	Pressed(cabID, floorNumber)
Open Door Button	Pressed(cabID) Released(cabID)
Service Switch	AutoMode(cabID) HoldMode(cabID)
Emergency Stop Button	Pressed(cabID)
Emergency Bell Button	Pressed()
Door Opening Device	DoorsOpened(cabID) DoorsClosed(cabID)
Position Marker Sensor	MarkerDetected(cabID)
Load Sensor	WeightChanged(cabID, newWeight)

Table 2: Output Signals

Component	Signals
Elevator Engine	Move(direction, speed)
Door Opening Device	OpenDoors() CloseDoors()
Emergency Bell	Ring()
Load Bell	Ring()
Direction Display	Show(direction)
Floor Number Display	Show(floorNumber)
Summon Buttons	TurnLightOn() TurnLightOff()
Floor Request Buttons	TurnLightOn() TurnLightOff()

4. Reference Tables and Descriptions

4.1 Functional Requirements Table and Traceability Document

Table 3: Functional Requirements

ID	Category	Name	Description	Details/ Constraints	Related Requirements/ Use Cases	Where Specified
F1[E]: Cab Responds to Summon			An engine eventually moves a cab to a floor with an active summon that was invoked by a summon button press.	The most appropriate cab is chosen based on the direction of the summon button pressed.	UC1	Fig. 2
F2[E]: Cab Responds to Floor Request			After a cab has an active floor request from one of its floor request buttons being pressed, its engine eventually moves it to the floor specified by the number of the button that was pressed.	The elevator cab may stop at other floors first.	UC3	Fig. 4
F3[E]: Cab Moves Right After Doors Close			After the doors of a cab close, it starts to move to its next destination floor as soon as possible.	The cab has a future destination floor from an active summon or floor request.	UC15	Fig. 16

F4[E]: Open Door Button Opens Doors	When a cab's open door button is pressed, its door opening device opens its doors.	The elevator cab must be stopped at a floor and the doors must not already be open. If the doors are in the middle of closing, they start to open.	F5, UC4	Fig. 5
F5[E]: Open Door Button Held Down	A cab's door opening device will not close its doors while the cab's open door button is pressed down.	None	F4, UC5	Fig. 6
F6[E]: Doors Open at Floor with Active Summon	A cab's door opening device opens its doors when it reaches a floor that has an active summon.	None	F7, UC11	Fig. 12
F7[E]: Doors Open at Requested Floor	A cab's door opening device opens its doors when it reaches a floor that had an active floor request on it which was specified by the press of a floor request button within the cab.	None	F6, UC11	Fig. 12
F8[E]: Summon Button Opens Doors	When the summon button is pressed on a floor where a cab has stopped and the doors are closing or have just closed and the cab has not yet started to move, the door opening device of that cab will open its doors.	The summon button acts like the open door button.	F9, UC1	Fig. 2
F9[E]: Summon Button Held Down	When the summon button is held down after it has been pressed on a floor where a cab has stopped and the doors are open, the door opening device will not close the doors.	The summon button acts like the open door button.	F8, UC2	Fig. 3
F10[E]: Doors Close Automatically	After the doors of a cab have been open for 3 seconds, its door opening device closes them.	The open door button of the cab or the summon button on the same floor as where the cab is stopped were not pressed within the 3 second time span. The load sensor must not have sent a signal that the weight inside the cab was over the safety limit within the 3 sec time span, either.	UC2, UC5, UC14	Fig. 3, Fig. 6, Fig. 15
F11[E]: Emergency Stop Button Pressed to Stop Cab	The cab from which an emergency stop button is pressed is stopped by its engine.	The cab was not already stopped by the emergency stop button.	UC6	Fig. 7
F12[E]: Emergency Bell Button	The emergency bell of the elevator system is rung when the emergency bell button is pressed from a cab.	None	UC7	Fig. 8
F13[E]: Service Switch	A cab operates normally when its service switch is in AUTO	None	F12, UC9	Fig. 10

AUTO	mode.			
F14[E]: Service Switch HOLD	When its service switch is in HOLD mode, a cab's engine does not move it and its door opening device keeps its doors in the state they were in right before the service switch was turned to HOLD mode from AUTO mode.	None	F13, UC8	Fig. 9
F15[E]: Load Bell Rings because of Excess Load	When the total weight inside of a cab exceeds the safety limit, its load bell rings.	None	F16, UC10	Fig. 11
F16[E]: Cab is Immobilized from Excess Load	When the total weight inside a cab exceeds the safety limit, its engine does not move it from the floor it is currently on and its door opening device keeps its doors open.	None	F15, UC10	Fig. 11
F17[E]: Summon Button Light On	After a summon button has been pressed, its light turns on.	The summon button's light was off.	F18, UC1	Fig. 2
F18[E]: Summon Button Light Off	After an engine has stopped its cab at a floor, the lights of all of the summon buttons on that floor are turned off.	A summon button's light was on.	F17, UC11	Fig. 12
F19[E]: Floor Request Button Light On	After a floor request button has been pressed, its light turns on.	The floor request button's light was off.	F20, UC3	Fig. 4
F20[E]: Floor Request Button Light Off	After an engine has brought its cab to a floor with an active floor request that was invoked by the press of a floor request button within that cab, that floor request button's light turns off.	The floor request button's light was on.	F19, UC11	Fig. 12
F21[E]: Direction Displayed	After the direction that a cab is moving in, or will move in, changes, its direction display is updated to reflect the new direction.	No direction is displayed if a cab is not moving and it has no requests to fulfill.	UC1, UC3	Fig. 2, Fig. 4
F22[E]: Floor Number Displayed	After a cab's position has changed to a different floor, regardless if it has stopped at that floor or not, the cab's floor number display is updated to reflect the new current floor.	None	UC11	Fig. 12
F23[I]: Elevator Controller On	The elevator controller can be turned on.	The elevator controller must be off.	F24, UC13	Fig. 14
F24[I]: Elevator Controller Off	The elevator controller can be turned off.	The elevator controller must be on.	F23, UC12	Fig. 13
F25[I]:	The cab from which an	The cab was previously	UC6	Fig. 7

Emergency Stop Button Pressed to Start Cab Again	emergency stop button is pressed is started by its engine.	stopped by the emergency stop button.		

4.2 Non-Functional Requirements Table and Traceability Document

Table 4: Non-Functional Requirements

ID	Category	Name	Description	Details/ Constraints	Related Requirements
NF1[M]: Cab Acceleration			For the safety of the passengers inside, cabs must not accelerate or decelerate faster than 3 m/s^2.	None	None
NF2[M]: Inner Door Opening			For the safety of the passengers inside, the inner doors of a cab should not be opened unless the cab is stopped at a correct floor position – they should never open while a cab is moving.	None	NF3, NF4
NF3[M]: Outer Door Opening			For the safety of potential passengers, the outer doors of a shaft should not be opened at a particular floor unless there is a cab stopped at that floor in that shaft.	None	NF2, NF4
NF4[M]: Cab Movement			For the safety of the passengers inside, a cab should not move until its doors are completely closed.	None	NF2, NF3
NF5[M]: Cab Immobilized from Failed Component			When a vital component of the elevator system fails, the operation of the affected cabs is halted.	None	None
NF6[M]: Response Time			Signals from other components of the elevator system must be processed in less than 500 milliseconds.	None	None
NF7[M]: Cab Selection			The cab that can process a summon request in the quickest amount of time must be chosen to answer that request.	Decision is based on each cab's currently active floor requests.	NF8
NF8[M]: Floor Request Order			The order in which a cab answers its active floor requests minimizes the amount of changes in direction the cab has to perform.	None	NF7
NF9[M]: Cabs Stop at Correct Position			The cabs must always stop at the correct position on each floor so that the outer doors of the shaft and the inner doors of the cab line	Only refers to stops under normal operation, not for	None

	up properly and passengers can get in and out safely.	emergency or service.	
NF10[M]: Complete Signal Processing	Every input signal received from another component must be processed regardless of what state the controller is in or if it the signal will cause any reaction – no signals can be lost or ignored.	The controller must be on to receive signals. It is not responsible for signals that are lost during transmission.	None

4.3 Use Case Descriptions and Diagrams

Table 5: Use Case 1 – Process Pressed Signal from Summon Button

Name: Process Pressed Signal from Summon Button	**ID:** UC1
Goal: To process a pressed signal from a summon button. **Event:** A summon button sends a signal to the controller indicating that it was pressed. **Precondition:** None **Postcondition:** The cab's doors are open at the original floor or the summon request is answered. **System:** Elevator Controller **Actors:** Summon Button, Door Opening Device, Engine, Direction Display, Position Marker Sensor **Overview:** This use case describes what happens when a summon button sends a signal to the controller indicating that it was pressed. **References:** F1, F8, F17, F21 **Related Use Cases:** UC2	
Typical Process Description	
Initiator Actions	**System Response**
1. Summon Button: Send pressed signal to the controller	
	if (a cab was already at the floor that the summon button was pressed from) 2. Send a signal to the cab's door opening device to open the doors
	else 2. Send a signal to turn on the light of the summon button that was pressed
	3. Queue the summon request
	4. Determine the most appropriate cab to answer the summon
	5. Send a series of signals to an engine to move a cab to the floor that the summon came from when its turn comes up – use received signals from the position marker sensor
	6. Send a signal to that cab's direction display to update it with the cab's current direction
	7. When a cab arrives at a floor with a pending summon, whether the summon was next in the queue or not, send a signal to all summon buttons on that floor to turn off their lights
	8. Send a signal to the cab's door opening device to open its doors
	9. Remove the summon request from the queue

Table 6: Use Case 2 – Process Released Signal from Summon Button

Name: Process Released Signal from Summon Button	**ID:** UC2
Goal: To process a released signal from a summon button. **Event:** A summon button sends a signal to the controller indicating that it was released. **Precondition:** None **Postcondition:** If necessary, the timer for closing doors automatically is started. **System:** Elevator Controller **Actors:** Summon Button **Overview:** This use case describes what happens when a summon button sends a signal to the controller indicating that it was released. **References:** F9, F10 **Related Use Cases:** UC1	
Typical Process Description	
Initiator Actions	**System Response**
1. Summon Button: Send released signal to the controller	
	if (a cab was already stopped at the floor that the summon button was released from AND the doors of the cab are open)
	2. Start the internal timer that is responsible for closing the elevator doors automatically

Table 7: Use Case 3 – Process Pressed Signal from Floor Request Button

Name: Process Pressed Signal from Floor Request Button	**ID:** UC3
Goal: To process a pressed signal from a floor request button. **Event:** A floor request button sends a signal to the controller indicating that it was pressed. **Precondition:** None **Postcondition:** The doors of the cab are open at the original floor or the floor request is answered. **System:** Elevator Controller **Actors:** Floor Request Button, Door Opening Device, Engine, Position Marker Sensor **Overview:** This use case describes what happens when a floor request button sends a signal to the controller indicating that it was pressed. **References:** F2, F19, F21 **Related Use Cases:** None	
Typical Process Description	
Initiator Actions	**System Response**
1. Floor Request Button: Send pressed signal to the controller	
	if (the cab from which the floor button was pressed is already at the floor specified by the pressed button) 2. Send a signal to the cab's door opening device to open the doors
	else
	2. Send a signal to turn on the light of the floor request button that was pressed
	3. Queue the floor request

	4. Determine when this floor request should be answered
	5. Send a series of signals to the engine to move the cab to the floor that the floor request specified when its turn comes up – use received signals from the position marker sensor
	6. Send a signal to that cab's direction display to update it with the cab's current direction
	7. When the cab arrives at the floor that was requested, send a signal to the corresponding floor request button to turn off its light
	8. Send a signal to the cab's door opening device to open the doors
	9. Remove the floor request from the queue

Table 8: Use Case 4 – Process Pressed Signal from Open Door Button

Name: Process Pressed Signal from Open Door Button	**ID:** UC4
Goal: To process a pressed signal from an open door button. **Event:** An open door button sends a signal to the controller indicating that it was pressed. **Precondition:** None **Postcondition:** If appropriate, the doors of the cab are open. **System:** Elevator Controller **Actors:** Open Door Button, Door Opening Device **Overview:** This use case describes what happens when an open door button sends a signal to the controller indicating that it was pressed. **References:** F4 **Related Use Cases:** UC5	
Typical Process Description	
Initiator Actions	**System Response**
1. Open Door Button: Send pressed signal to the controller	
	if (the cab from which the open door button was pressed is not moving AND is stopped at a floor)
	2. Send a signal to the cab's door opening device to open its doors

Table 9: Use Case 5 – Process Released Signal from Open Door Button

Name: Process Released Signal from Open Door Button	**ID:** UC5
Goal: To process a released signal from an open door button. **Event:** An open door button sends a signal to the controller indicating that it was released. **Precondition:** None **Postcondition:** If necessary, the timer for closing doors automatically is started. **System:** Elevator Controller **Actors:** Open Door Button **Overview:** This use case describes what happens when an open door button sends a signal to the controller indicating that it was released. **References:** F5, F10	

Related Use Cases: UC4	
Typical Process Description	
Initiator Actions	**System Response**
1. Open Door Button: Send released signal to the controller	
	if (the cab is stopped at a floor AND the doors of the cab are open)
	2. Start the internal timer that is responsible for closing the elevator doors automatically

Table 10: Use Case 6 – Process Pressed Signal from Emergency Stop Button

Name: Process Pressed Signal from Emergency Stop Button	**ID:** UC6
Goal: To process a pressed signal from an emergency stop button. **Event:** An emergency stop button sends a signal to the controller indicating that it was pressed. **Precondition:** None **Postcondition:** The cab is either stopped or started again depending on its previous state. **System:** Elevator Controller **Actors:** Emergency Stop Button, Engine, Position Marker Sensor **Overview:** This use case describes what happens when an emergency stop button sends a signal to the controller indicating that it was pressed. **References:** F11, F25 **Related Use Cases:** None	
Typical Process Description	
Initiator Actions	**System Response**
1. Emergency Stop Button: Send pressed signal to the controller	
	if (the cab is not currently stopped by the emergency stop button) 2. Send a series of signals to the engine of the cab from which the emergency stop button was pressed to stop the cab – use received signals from the position marker sensor
	else 2. Send a signal to the engine of the cab from which the emergency stop button was pressed to start moving the cab again

Table 11: Use Case 7 – Process Pressed Signal from Emergency Bell Button

Name: Process Pressed Signal from Emergency Bell Button	**ID:** UC7
Goal: To process a pressed signal from an emergency bell button. **Event:** An emergency bell button sends a signal to the controller indicating that it was pressed. **Precondition:** None **Postcondition:** The emergency bell has been rung. **System:** Elevator Controller **Actors:** Emergency Bell Button, Emergency Bell **Overview:** This use case describes what happens when an emergency bell button sends a signal to the controller indicating that it was pressed.	

References: F12 **Related Use Cases:** None	
Typical Process Description	
Initiator Actions	**System Response**
1. Emergency Bell Button: Send pressed signal to the controller	
	2. Send a ring signal to the emergency bell

Table 12: Use Case 8 – Process HOLD Mode Signal from Service Switch

Name: Process HOLD Mode Signal from Service Switch	**ID:** UC8
Goal: To process a HOLD mode signal from a service switch. **Event:** The mode of a service switch is toggled to HOLD. **Precondition:** None **Postcondition:** The cab's normal operation has been held and it is stopped. **System:** Elevator Controller **Actors:** Service Switch, Engine, Position Marker Sensor **Overview:** This use case describes what happens when the mode of a service switch is toggled to HOLD. **References:** F14 **Related Use Cases:** UC9	
Typical Process Description	
Initiator Actions	**System Response**
1. Service Switch: Send signal to the controller that the mode is toggled to HOLD	
	if (the cab is not already stopped) 2. Send a series of signals to the engine of the cab from which the service switch was toggled to HOLD to stop the elevator – use received signals from the position marker sensor
	3. Do not send any signals to the cab's engine or door opening device in order to keep the cab idle

Table 13: Use Case 9 – Process AUTO Mode Signal from Service Switch

Name: Process AUTO Mode Signal from Service Switch	**ID:** UC9
Goal: To process an AUTO mode signal from a service switch. **Event:** The mode of a service switch is toggled to AUTO. **Precondition:** None **Postcondition:** The cab is running under normal operation. **System:** Elevator Controller **Actors:** Service Switch, Engines, Door Opening Devices **Overview:** This use case describes what happens when the mode of a service switch is toggled to AUTO. **References:** F13 **Related Use Cases:** UC8	
Typical Process Description	
Initiator Actions	**System Response**
1. Service Switch: Send signal to the controller that the mode is toggled to AUTO	
	2. Start sending signals to the engine and door

	opening device of the switch's cab to resume its normal operation

Table 14: Use Case 10 – Process Weight Changed Signal from Load Sensor

Name: Process Weight Changed Signal from Load Sensor	**ID:** UC10
Goal: To process a weight changed signal from a load sensor. **Event:** The load sensor sends a signal to the controller indicating a new total weight inside of a cab. **Precondition:** The elevator cab which the load sensor is sending the signal from is stopped at a floor and its doors are open. **Postcondition:** It is ensured that the cab is not running with a load that exceeds the safety limit. **System:** Elevator Controller **Actors:** Load Sensor, Load Bell **Overview:** This use case describes what happens when the load sensor sends a weight signal to the controller. **References:** F15, F16 **Related Use Cases:** None	
Typical Process Description	
Initiator Actions	**System Response**
1. Load Sensor: Send signal to the controller indicating a new total weight inside of a cab	
	if (the weight indicated by the signal is above the safety limit) 2. Send a signal to the cab's load bell to ring
	3. Do not send any signals to the cab's engine or door opening device in order to keep the cab stopped and its doors open until another weight signal is received indicating that the new weight in the cab is below the safety limit
	else 2. Continue normal operation of elevators

Table 15: Use Case 11 – Process Detection Signal from Position Marker Sensor

Name: Process Detection Signal from Position Marker Sensor	**ID:** UC11
Goal: To process a detection signal from a position marker sensor. **Event:** A position marker sensor sends a signal to the controller indicating that it detected a position marker. **Precondition:** The cab is moving. **Postcondition:** The cab's speed and direction is appropriate to what its current position is. **System:** Elevator Controller **Actors:** Position Marker Sensor, Summon Buttons, Floor Number Display, Door Opening Device, Engine **Overview:** This use case describes what happens when the position marker sensor sends a signal indicating that it detected a position marker. **References:** F6, F7, F18, F20, F22 **Related Use Cases:** None	
Typical Process Description	
Initiator Actions	**System Response**
1. Position Marker Sensor: Send signal to the controller indicating that it detected a position	

marker	
	2. Calculate the new position of the cab
	if (the new position of the cab has changed what floor it is on)
	3. Send a signal to the cab's floor number display to update the displayed floor number
	If the new floor that the cab is on has an active floor request within this cab or an active summon
	4. Send a series of signals to the cab's engine to stop it – use received signals from the position marker sensor
	5. Send a signal to the cab's door opening device to open its doors
	6. Send signals to the summon button lights and floor request button lights for this floor to turn off
	7. Remove any summons or requests for this floor from the queue

Table 16: Use Case 12 – Process Off Signal from Operator

Name: Process Off Signal from Operator	**ID:** UC12
Goal: To process an off signal sent by an operator of the elevator system. **Event:** An operator of the elevator controller sends it an off signal. **Precondition:** The controller is on. **Postcondition:** The controller is off. **System:** Elevator Controller **Actors:** Operator, Engine, Summon Buttons, Floor number Displays **Overview:** This use case describes what happens when an operator turns the controller off. **References:** F24 **Related Use Cases:** UC13	
Typical Process Description	
Initiator Actions	**System Response**
1. Operator: Send signal to the controller to turn it off	
	2. Send a series of signals to both engines to stop their cabs – use received signals from the position marker sensors
	3. Send signals to the door opening devices to close their doors
	4. Send signals to all summon buttons and floor request buttons to turn off their lights
	5. Send signals to the floor number displays and direction displays to clear them
	6. Stop responding to signals from other components in the elevator system

Table 17: Use Case 13 – Process On Signal from Operator

Name: Process On Signal from Operator	**ID:** UC13
Goal: To process an on signal sent by an operator of the elevator system. **Event:** An operator of the elevator controller sends it an on signal.	

Precondition: The controller is off. **Postcondition:** The controller is on. **System:** Elevator Controller **Actors:** Operator **Overview:** This use case describes what happens when an operator turns the controller on. **References:** F23 **Related Use Cases:** UC12	
Typical Process Description	
Initiator Actions	**System Response**
1. Operator: Send signal to the controller to turn it on.	
	2. Perform any needed initialization
	3. Start responding to signals from other components

Table 18: Use Case 14 – Process Doors Opened Signal from Door Opening Device

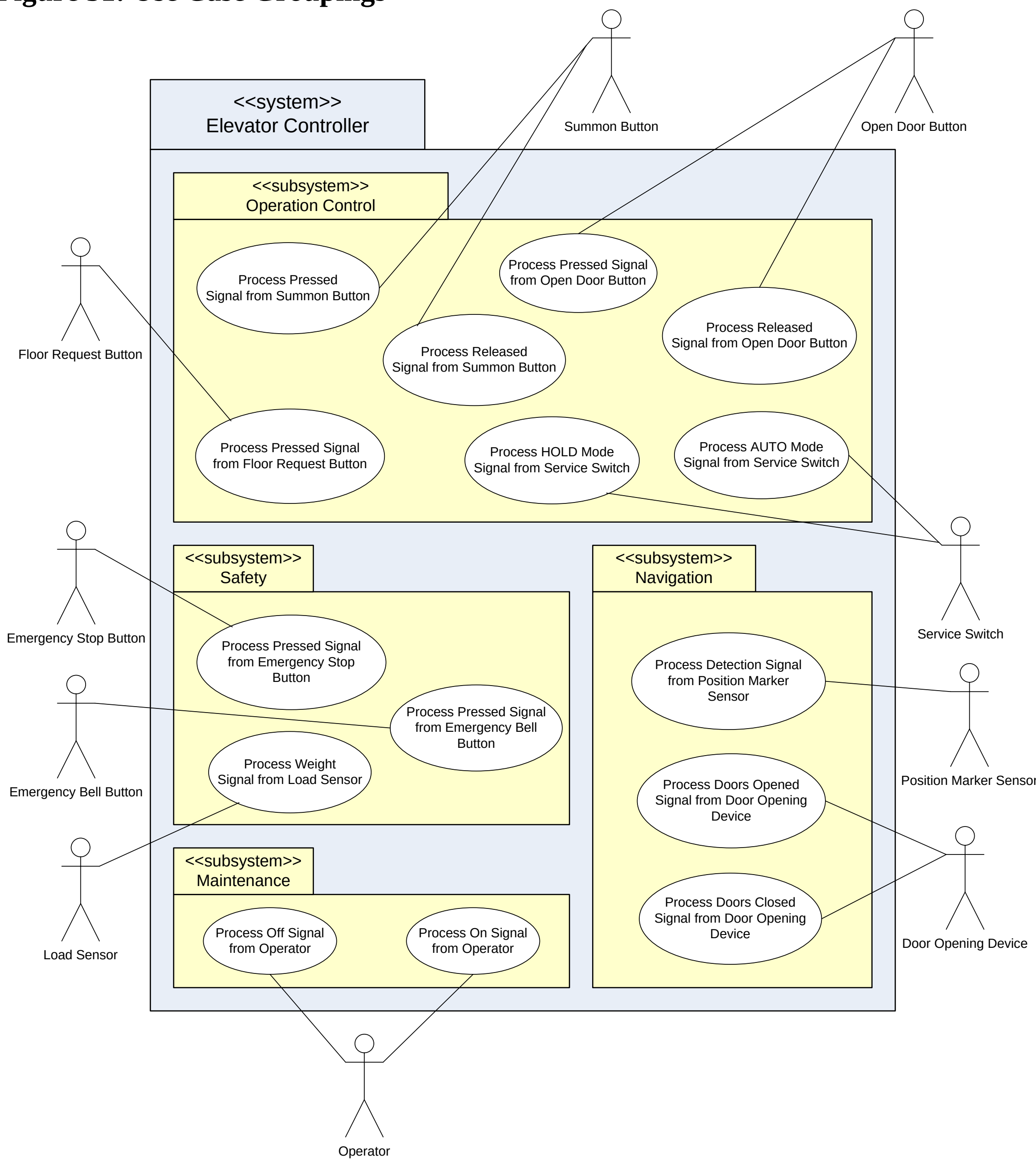
Name: Process Doors Opened Signal from Door Opening Device	**ID:** UC14
Goal: To process a doors opened signal from a door opening device. **Event:** A door opening device sends a signal that its doors have successfully been opened. **Precondition:** The cab is stopped. **Postcondition:** If necessary, the timer for closing doors automatically is started. **System:** Elevator Controller **Actors:** Door Opening Device, Summon Buttons, Open Door Button **Overview:** This use case describes what happens when a door opening device sends a signal that its doors have successfully been opened. **References:** F10 **Related Use Cases:** UC15	
Typical Process Description	
Initiator Actions	**System Response**
1. Door Opening Device: Send signal to the controller indicating that the doors of this cab have successfully been opened	
	if (a summon button at the floor at which this cab is stopped is not being held down AND the open door button of this cab is also not being held down)
	2. Start the internal timer that is responsible for closing the elevator doors automatically

Table 19: Use Case 15 – Process Doors Closed Signal from Door Opening Device

Name: Process Doors Closed Signal from Door Opening Device	**ID:** UC15
Goal: To process a doors closed signal from a door opening device. **Event:** A door opening device sends a signal that its doors have successfully been closed. **Precondition:** The cab is stopped. **Postcondition:** The cab has either started to answer a summon or floor request or is idle. **System:** Elevator Controller **Actors:** Door Opening Device, Direction Display, Engine	

Overview: This use case describes what happens when a door opening device sends a signal that its doors have successfully been closed. **References:** F3 **Related Use Cases:** UC14	
Typical Process Description	
Initiator Actions	**System Response**
1. Door Opening Device: Send signal to the controller indicating that the doors of this cab have successfully been closed	
	2. Determine which floor this cab should go to next from the queue
	if (there is an active summon or floor request on another floor in the queue that is not being handled by another cab) 3. Send a signal to the direction display to update its displayed direction
	4. Send a signal to the cab's engine to start moving it in the proper direction and speed
	else 3. Clear the direction display so it does not display any direction

Figure 51: Use Case Groupings



4.4 Index