James Yap – 101276054
COMP 3004 Assignment 3

Use Case 1: Calling the Elevator
Primary Actor: Passenger
Scope: Elevator in use
Level: User goal
Stakeholders and Interests:

- Passenger: A user aiming to access the elevator for transportation.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: The floor is accessible via the elevator, and the user has access.
Minimal Guarantee: The elevator should be made inaccessible if there's a malfunction or repair.
Success Guarantee: The elevator door opens, aligning with the floor level on the user's floor.
Trigger: User presses the elevator call button on a specified floor.
Main Success Scenario:

1. User arrives on floor serviced by the elevators.
2. User presses "up" or "down" button.
3. Button illuminates and remains illuminated until elevator arrives.
4. Elevator rings upon arrival, button light turns off.
5. Elevator and floor doors open together.
6. Doors remain open briefly for entry or exit.
7. Doors close in sync after a bell rings.
8. User enters elevator. Proceed to ***Use Case 2***.

Extensions:

7a. If doors are obstructed, proceed to **Use Case 3**.

Use Case 2: Riding the Elevator
Actor: Passenger in the elevator
Scope: Elevator in use
Level: User goal
Stakeholders and Interests:

- Passenger: Seeks transportation via elevator.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: User boards the elevator.
Minimal Guarantee: The elevator is inaccessible if there's a malfunction or repair. User is safe from any form of physical harm.
Success Guarantee: User reaches their destination promptly and safely.
Trigger: User selects destination floor.
Main Success Scenario:

1. User boards the elevator.
2. Weight sensor checks load.
3. User selects "close" or "open" door or waits for automatic closure.
4. Floor doors and elevator doors close simultaneously.
5. User selects destination floor(s) on the panel.

6. Elevator moves to selected floor.
7. Onboard display reads the current floor number.
8. Audio system indicates floor.
9. Elevator rings as doors open together.
10. User exits, enters, or remains in elevator.

Extensions:

2a. If weight exceeds limit, perform ***Use Case 4***.

4a. If obstacle detected during door closure, perform ***Use Case 3***.

1-10. If "Help" button pressed, perform ***Use Case 5***.

1-10. If "Fire" button pressed or fire alarm received, perform ***Use Case 6***.

1-10. If power outage signal received from ECS, perform ***Use Case 7***.

Use Case 3: Obstacle Detection

Actor: Passenger

Scope: Elevator in use

Level: User goal

Stakeholders and Interests:

- Passenger: Requires elevator service.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: Obstacle detected during door closure.

Minimal Guarantee: Elevator is inaccessible in case of malfunction.

Success Guarantee: User reaches destination promptly and safely.

Trigger: User boards the elevator.

Main Success Scenario:

1. Light sensor detects obstruction during door closure.
2. Control system halts door closure, reopens.
3. Elevator doors remains open until obstruction clears.
4. Doors closes cautiously.
5. Resume normal operation.

Extension:

3a. If repeated obstructions occur, warning is issued via audio and text.

Use Case 4: Elevator Overload

Actor: Passenger

Scope: Elevator in use

Level: User goal

Stakeholders and Interests:

- Passenger: Requires elevator service.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: Elevator load surpasses limit.

Minimal Guarantee: Elevator is inaccessible in case of malfunction.

Success Guarantee: User reaches destination promptly and safely.

Trigger: User(s) enters and overloads elevator.

Main Success Scenario:

1. Sensor detects excessive load.
2. Elevator sends "Overload" signal to the ECS.
3. Door remains open.
4. Elevator alerts passengers with audio cues/instructions and displays message.
5. Elevator waits until weight decreases.
6. Normal operation resumes.

Use Case 5: Intercom Help Request

Actor: Passenger

Scope: Elevator in use

Level: User goal

Stakeholders and Interests:

- Passenger: Requires elevator service.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: User presses "Help" button.

Minimal Guarantee: Elevator is inaccessible in case of malfunction.

Success Guarantee: User reaches destination promptly and safely.

Trigger: User boards the elevator.

Main Success Scenario:

1. User activates "Help" button, notifying control system.
2. User connects with safety service within 5 seconds.
3. Assistance promptly provided by Building Safety Services.

Extension:

2a. If no response, initiate 911 call.

Use Case 6: Fire Alarm

Actor: Passenger

Scope: Elevator in use

Level: User goal

Stakeholders and Interests:

- Passenger: Requires elevator service.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: User activates "Fire" button or fire alarm received.

Minimal Guarantee: Elevator is inaccessible in case of malfunction.

Success Guarantee: User reaches safe floor.

Trigger: User boards the elevator.

Main Success Scenario:

1. User activates "Fire" signal or ECS receives “Fire” signal from the building.
2. Elevator bypasses selected floors, heads to safe floor.
3. Doors open simultaneously for safe evacuation.
4. Elevator alerts passengers of emergency, directs evacuation.

Use Case 7: Power Outage

Actor: Passenger

Scope: Elevator in use

Level: User goal

Stakeholders and Interests:

- Passenger: Requires elevator service.
- Building safety service: Ensures elevator functionality and offers aid if needed.

Precondition: Control system receives "Power Out" alarm from the ECS.

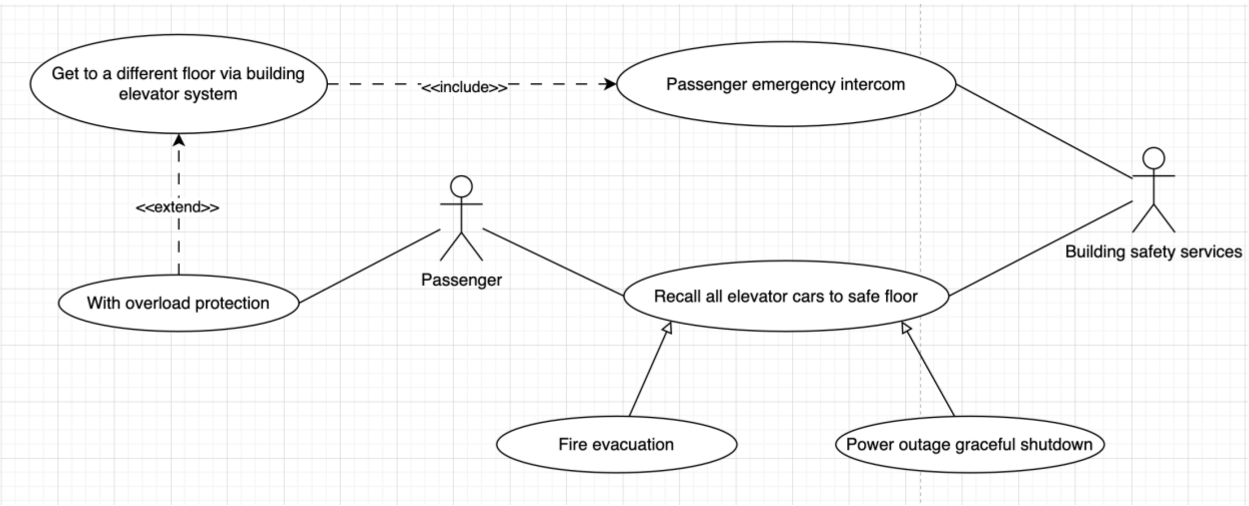
Minimal Guarantee: Elevator is inaccessible in case of malfunction.

Success Guarantee: User reaches safe floor.

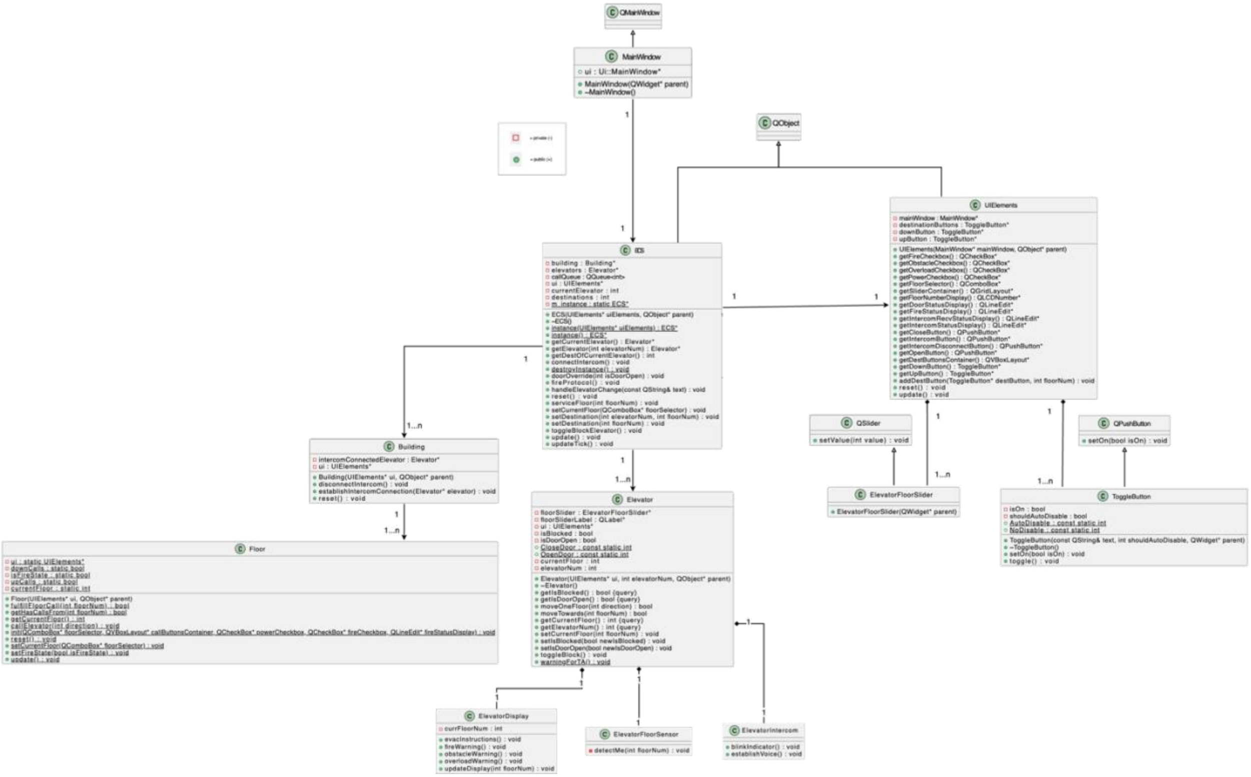
Trigger: User boards the elevator.

Main Success Scenario:

1. Control system detects power outage.
2. Passengers informed of outage.
3. Elevator switches to backup power.
4. Elevator heads directly to safe floor.
5. Doors open for evacuation.


Get to a different floor via building elevator system
<<include>>
Passenger emergency intercom
<<extend>>
Passenger
Building safety services
With overload protection
Recall all elevator cars to safe floor
Fire evacuation
Power outage graceful shutdown

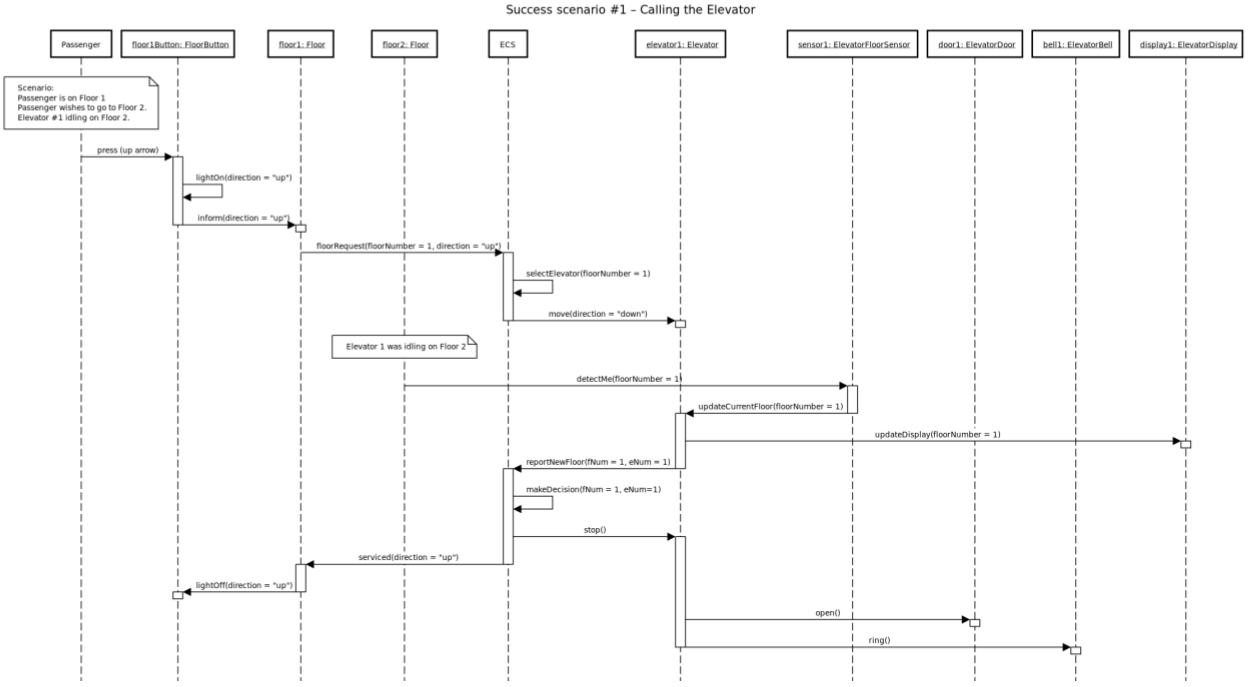
UML Class Diagram (Centralized)


QMainWindow
MainWindow
ui : Ui::MainWindow*
MainWindow(QWidget* parent)
~MainWindow()
= private (-)
= public (+)
QObject
ECS
building : Building*
elevators : Elevator*
callQueue : QQueue<int>
ui : UIElements*
currentElevator : int
destinations : int
m_instance : static ECS*
ECS(UIElements* uiElements, QObject* parent)
~ECS()
instance(UIElements* uiElements) : ECS*
instance() : ECS*
getCurrentElevator() : Elevator*
getElevator(int elevatorNum) : Elevator*
getDestOfCurrentElevator() : int
connectIntercom() : void
destroyInstance() : void
doorOverride(int isDoorOpen) : void
fireProtocol() : void
handleElevatorChange(const QString& text) : void
reset() : void
serviceFloor(int floorNum) : void
setCurrentFloor(QComboBox* floorSelector) : void
setDestination(int elevatorNum, int floorNum) : void
setDestination(int floorNum) : void
toggleBlockElevator() : void
update() : void
updateTick() : void
UIElements
mainWindow : MainWindow*
destinationButtons : ToggleButton*
downButton : ToggleButton*
upButton : ToggleButton*
UIElements(MainWindow* mainWindow, QObject* parent)
getFireCheckbox() : QCheckBox*
getObstacleCheckbox() : QCheckBox*
getOverloadCheckbox() : QCheckBox*
getPowerCheckbox() : QCheckBox*
getFloorSelector() : QComboBox*
getSliderContainer() : QGridLayout*
getFloorNumberDisplay() : QLCDNumber*
getDoorStatusDisplay() : QLineEdit*
getFireStatusDisplay() : QLineEdit*
getIntercomRecvStatusDisplay() : QLineEdit*
getIntercomStatusDisplay() : QLineEdit*
getCloseButton() : QPushButton*
getIntercomButton() : QPushButton*
getIntercomDisconnectButton() : QPushButton*
getOpenButton() : QPushButton*
getDestButtonsContainer() : QVBoxLayout*
getDownButton() : ToggleButton*
getUpButton() : ToggleButton*
addDestButton(ToggleButton* destButton, int floorNum) : void
reset() : void
update() : void
QSlider
setValue(int value) : void
ElevatorFloorSlider
ElevatorFloorSlider(QWidget* parent)
QPushButton
setOn(bool isOn) : void
ToggleButton
isOn : bool
shouldAutoDisable : bool
AutoDisable : const static int
NoDisable : const static int
ToggleButton(const QString& text, int shouldAutoDisable, QWidget* parent)
~ToggleButton()
setOn(bool isOn) : void
toggle() : void
Building
intercomConnectedElevator : Elevator*
ui : UIElements*
Building(UIElements* ui, QObject* parent)
disconnectIntercom() : void
establishIntercomConnection(Elevator* elevator) : void
reset() : void
Floor
ui : static UIElements*
downCalls : static bool
isFireState : static bool
upCalls : static bool
currentFloor : static int
Floor(UIElements* ui, QObject* parent)
fulfillFloorCall(int floorNum) : bool
getHasCallsFrom(int floorNum) : bool
getCurrentFloor() : int
callElevator(int direction) : void
init(QComboBox* floorSelector, QVBoxLayout* callButtonsContainer, QCheckBox* powerCheckbox, QCheckBox* fireCheckbox, QLineEdit* fireStatusDisplay) : void
reset() : void
setCurrentFloor(QComboBox* floorSelector) : void
setFireState(bool isFireState) : void
update() : void
Elevator
floorSlider : ElevatorFloorSlider*
floorSliderLabel : QLabel*
ui : UIElements*
isBlocked : bool
isDoorOpen : bool
CloseDoor : const static int
OpenDoor : const static int
currentFloor : int
elevatorNum : int
Elevator(UIElements* ui, int elevatorNum, QObject* parent)
~Elevator()
getIsBlocked() : bool {query}
getIsDoorOpen() : bool {query}
moveOneFloor(int direction) : bool
moveTowards(int floorNum) : bool
getCurrentFloor() : int {query}
getElevatorNum() : int {query}
setCurrentFloor(int floorNum) : void
setIsBlocked(bool newIsBlocked) : void
setIsDoorOpen(bool newIsDoorOpen) : void
toggleBlock() : void
warningForTA() : void
ElevatorDisplay
currFloorNum : int
evacInstructions() : void
fireWarning() : void
obstacleWarning() : void
overloadWarning() : void
updateDisplay(int floorNum) : void
ElevatorFloorSensor
detectMe(int floorNum) : void
ElevatorIntercom
blinkIndicator() : void
establishVoice() : void
1
1
1
1
1
1...n
1
1...n
1
1...n
1
1...n
1
1...n
1
1
1
1
1
1

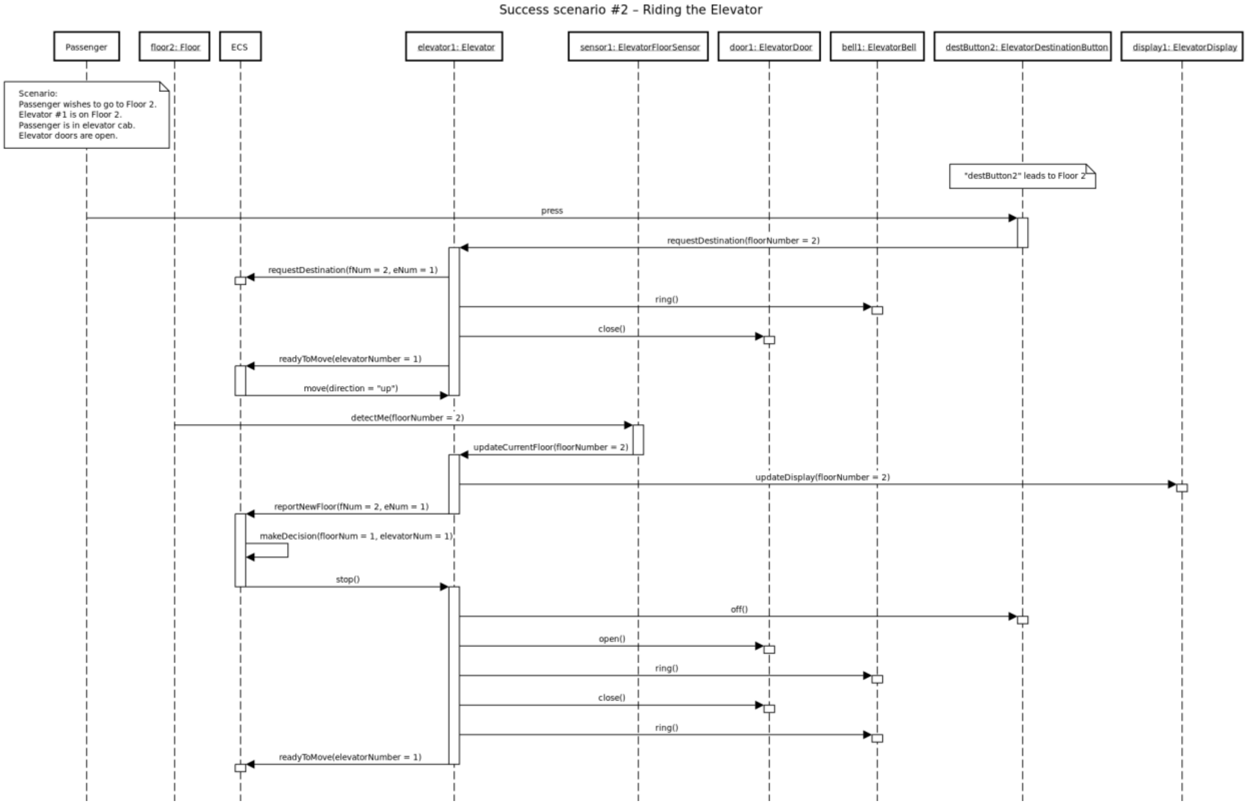
For reference, the full-scale version of this diagram is also included in the submitted zip package.

Sequence Diagram (Centralized) – Success scenario #1

Sequence Diagrams assume there are 3 passengers, 7 floors, and 3 elevators.

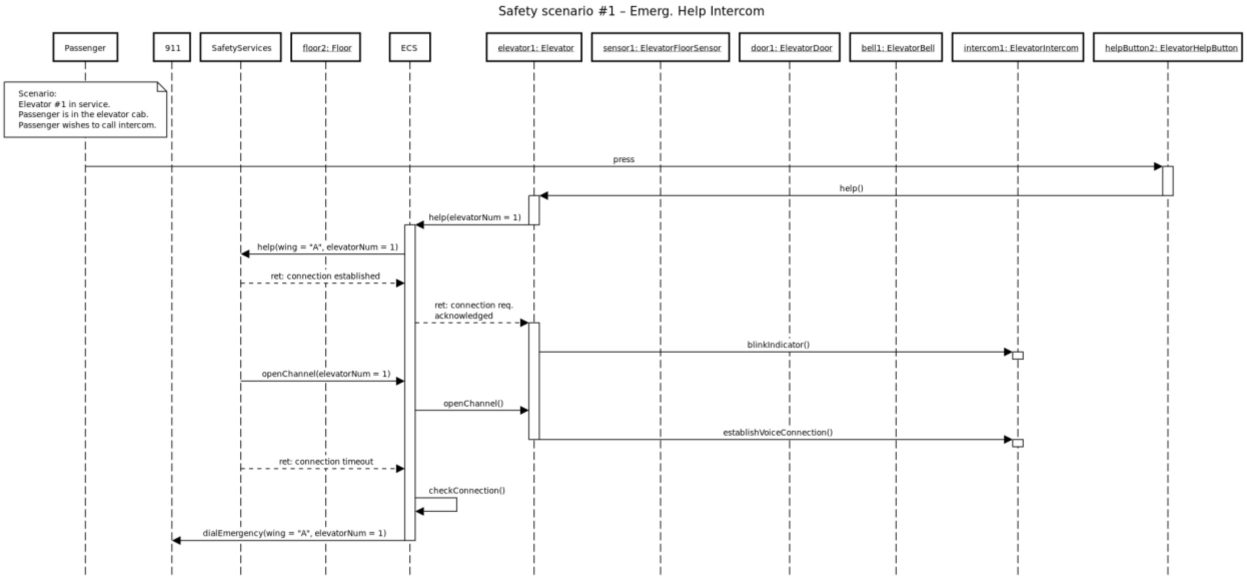

Success scenario #1 – Calling the Elevator
Passenger
floor1Button: FloorButton
floor1: Floor
floor2: Floor
ECS
elevator1: Elevator
sensor1: ElevatorFloorSensor
door1: ElevatorDoor
bell1: ElevatorBell
display1: ElevatorDisplay
Scenario:
Passenger is on Floor 1
Passenger wishes to go to Floor 2.
Elevator #1 idling on Floor 2.
press (up arrow)
lightOn(direction = "up")
inform(direction = "up")
floorRequest(floorNumber = 1, direction = "up")
selectElevator(floorNumber = 1)
move(direction = "down")
Elevator 1 was idling on Floor 2
detectMe(floorNumber = 1)
updateCurrentFloor(floorNumber = 1)
updateDisplay(floorNumber = 1)
reportNewFloor(fNum = 1, eNum = 1)
makeDecision(fNum = 1, eNum=1)
stop()
serviced(direction = "up")
lightOff(direction = "up")
open()
ring()

Sequence Diagram (Centralized) – Success scenario #2


Success scenario #2 – Riding the Elevator
Passenger
floor2: Floor
ECS
elevator1: Elevator
sensor1: ElevatorFloorSensor
door1: ElevatorDoor
bell1: ElevatorBell
destButton2: ElevatorDestinationButton
display1: ElevatorDisplay
Scenario:
Passenger wishes to go to Floor 2.
Elevator #1 is on Floor 2.
Passenger is in elevator cab.
Elevator doors are open.
"destButton2" leads to Floor 2
press
requestDestination(floorNumber = 2)
requestDestination(fNum = 2, eNum = 1)
ring()
close()
readyToMove(elevatorNumber = 1)
move(direction = "up")
detectMe(floorNumber = 2)
updateCurrentFloor(floorNumber = 2)
updateDisplay(floorNumber = 2)
reportNewFloor(fNum = 2, eNum = 1)
makeDecision(floorNum = 1, elevatorNum = 1)
stop()
off()
open()
ring()
close()
ring()
readyToMove(elevatorNumber = 1)

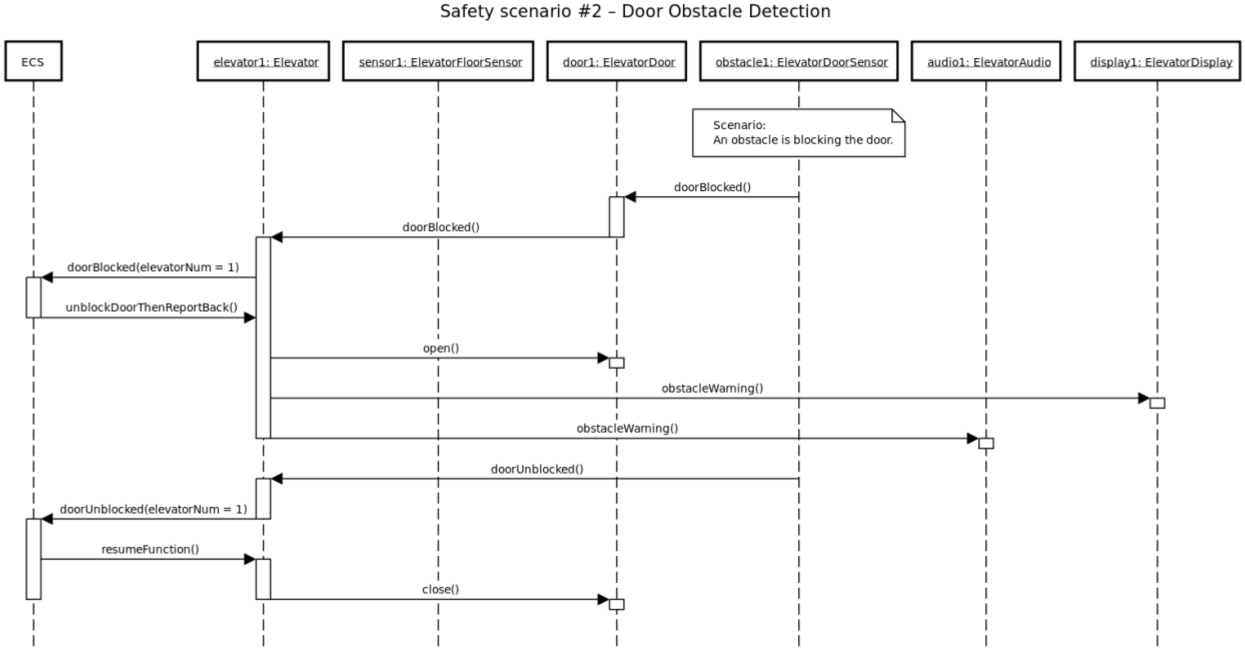
Sequence Diagram (Centralized) – Safety scenario #1

Help Button Intercom


Safety scenario #1 – Emerg. Help Intercom
Passenger
911
SafetyServices
floor2: Floor
ECS
elevator1: Elevator
sensor1: ElevatorFloorSensor
door1: ElevatorDoor
bell1: ElevatorBell
intercom1: ElevatorIntercom
helpButton2: ElevatorHelpButton
Scenario:
Elevator #1 in service.
Passenger is in the elevator cab.
Passenger wishes to call intercom.
press
help()
help(elevatorNum = 1)
help(wing = "A", elevatorNum = 1)
ret: connection established
ret: connection req. acknowledged
blinkIndicator()
openChannel(elevatorNum = 1)
openChannel()
establishVoiceConnection()
ret: connection timeout
checkConnection()
dialEmergency(wing = "A", elevatorNum = 1)

Sequence Diagram (Centralized) – Safety scenario #2

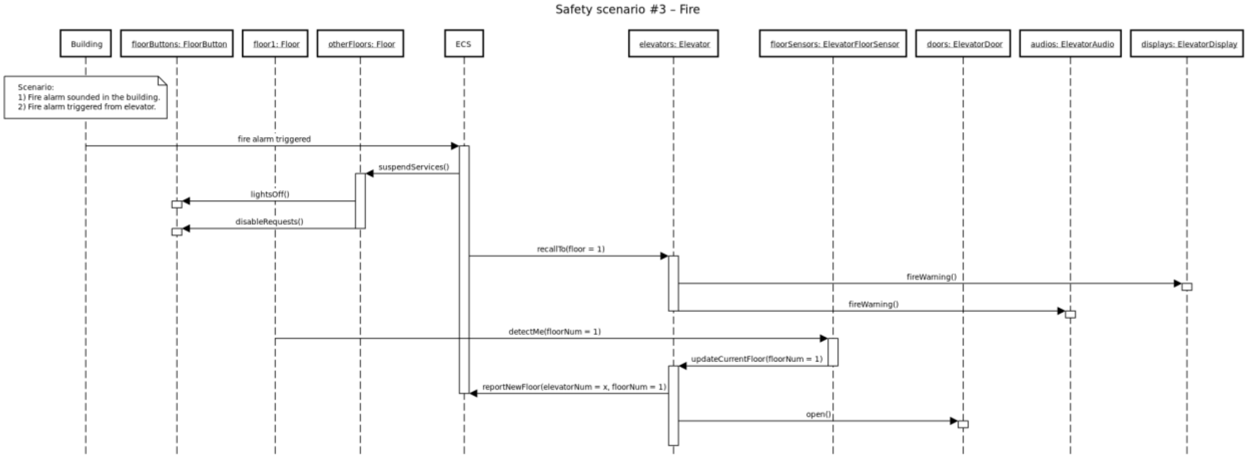
Door obstacles


Safety scenario #2 – Door Obstacle Detection

ECS
elevator1: Elevator
sensor1: ElevatorFloorSensor
door1: ElevatorDoor
obstacle1: ElevatorDoorSensor
audio1: ElevatorAudio
display1: ElevatorDisplay
Scenario:
An obstacle is blocking the door.
doorBlocked()
doorBlocked()
doorBlocked(elevatorNum = 1)
unblockDoorThenReportBack()
open()
obstacleWarning()
obstacleWarning()
doorUnblocked()
doorUnblocked(elevatorNum = 1)
resumeFunction()
close()

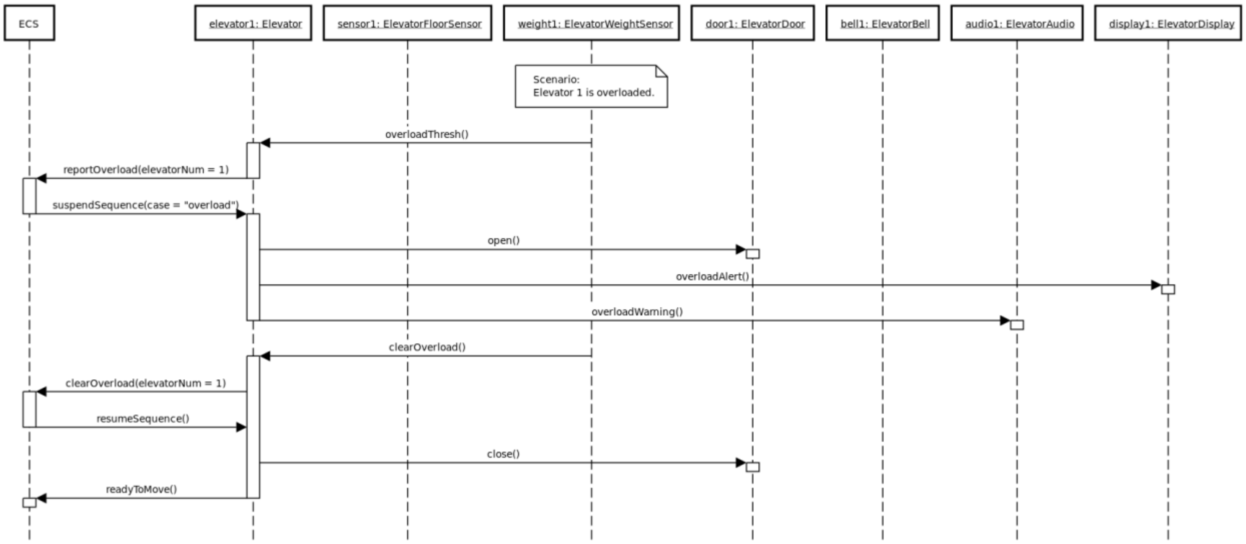
Sequence Diagram (Centralized) – Safety scenario #3

Fire


Safety scenario #3 – Fire

Sequence Diagram (Centralized) – Safety scenario #4

Overload

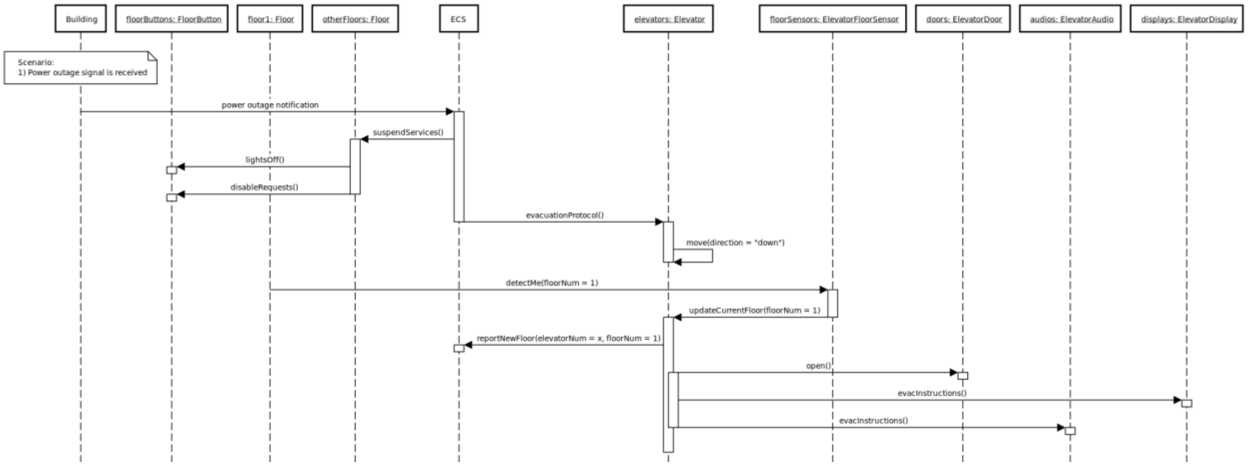


Safety scenario #4 – Elevator Cab Overload

ECS
elevator1: Elevator
sensor1: ElevatorFloorSensor
weight1: ElevatorWeightSensor
door1: ElevatorDoor
bell1: ElevatorBell
audio1: ElevatorAudio
display1: ElevatorDisplay
Scenario:
Elevator 1 is overloaded.
overloadThresh()
reportOverload(elevatorNum = 1)
suspendSequence(case = "overload")
open()
overloadAlert()
overloadWarning()
clearOverload()
clearOverload(elevatorNum = 1)
resumeSequence()
close()
readyToMove()

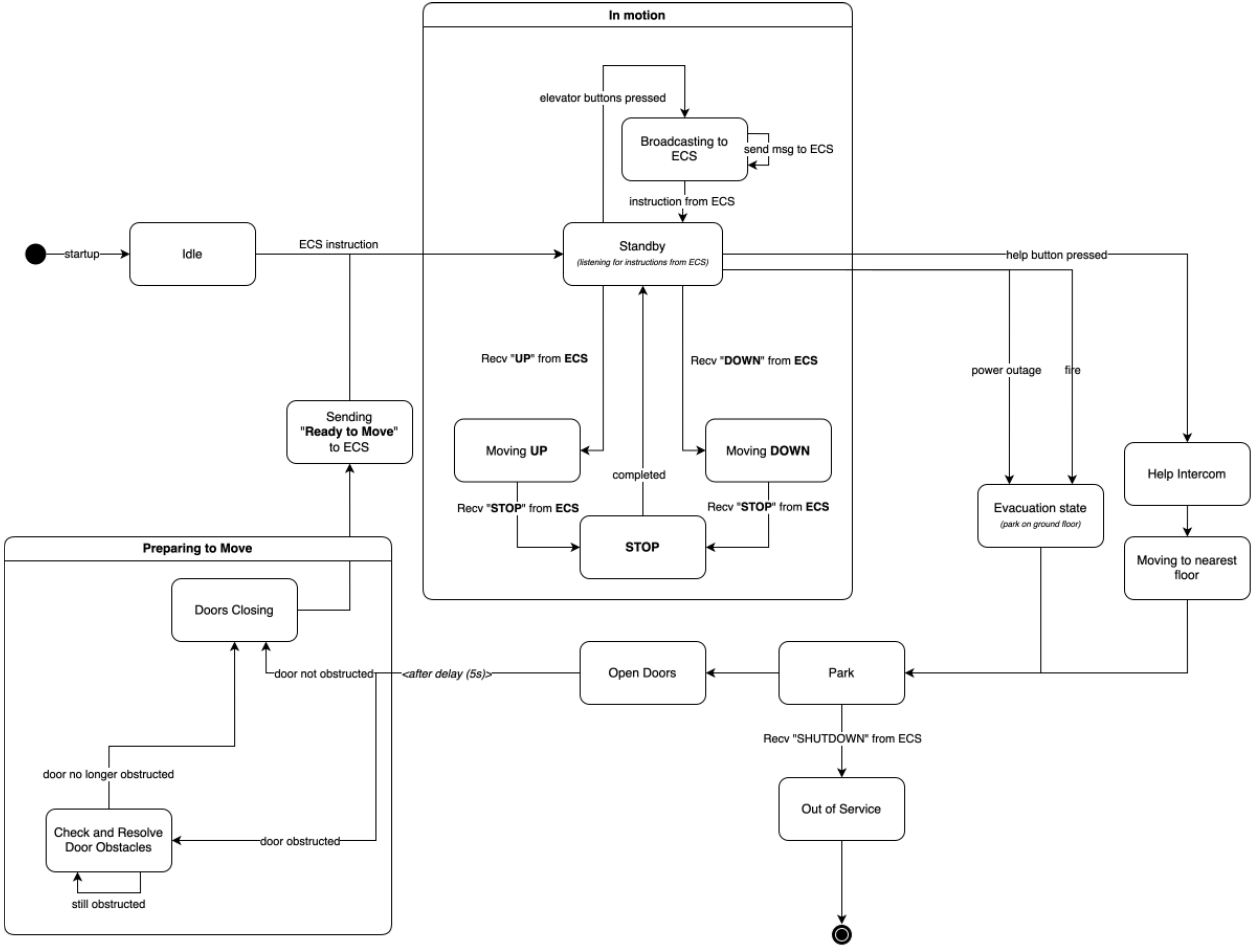
Sequence Diagram (Centralized) – Safety scenario #5

Power outage

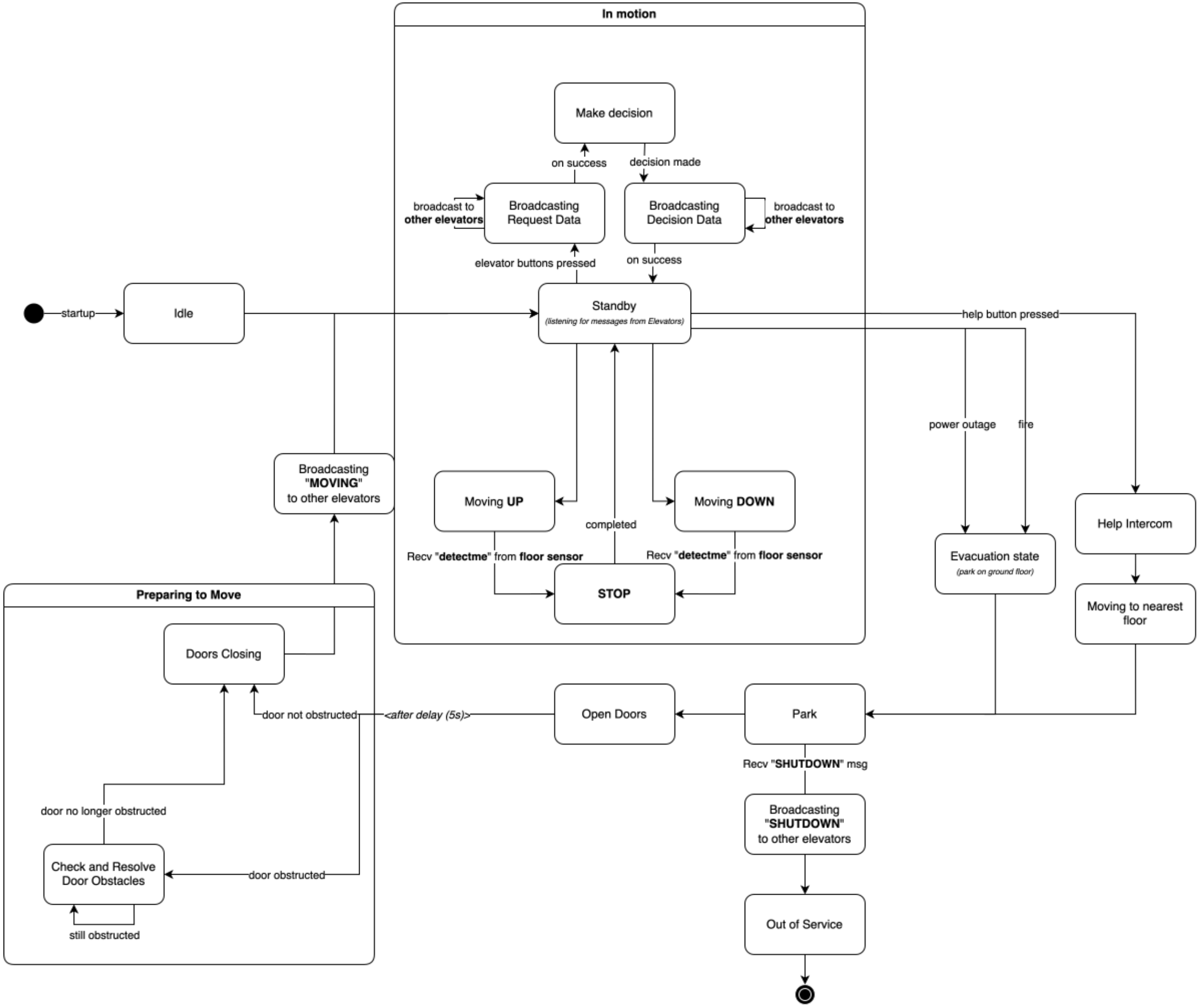


Safety scenario #5 – Power Outage

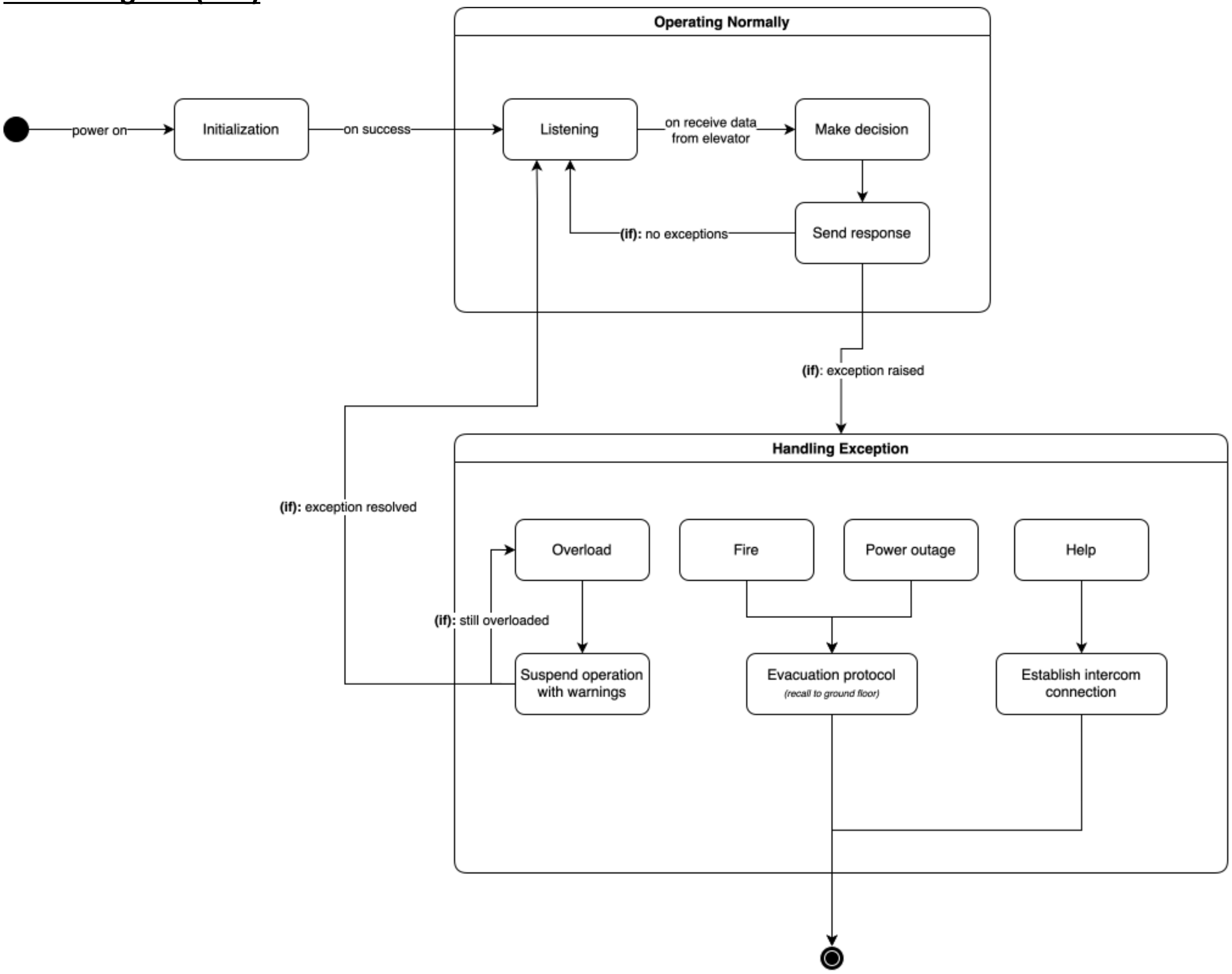
State Diagram (Centralized)



State Diagram (Distributed)



State Diagram (ECS)



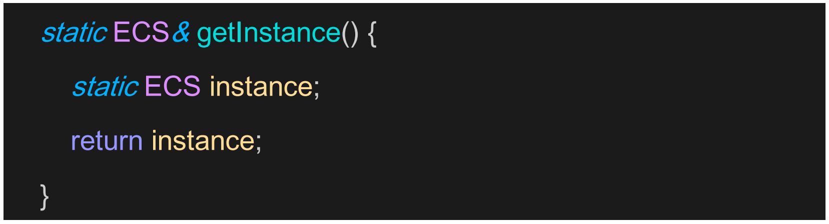
Design Decision Discussion (Design Patterns)

Centralized

- “**Mediator**” pattern used from the Gang of Four Design Patterns.
- Mediator is used because elevators communicate with each other in well-defined but complex ways. The resulting interdependency of elevators and sensors become messy and hard to understand.
- Having a centralized “Elevator Control System (ECS)” abstracts the communication between elevator cars (colleagues) and allows for easier maintenance as elevators are modular and can be independently modified/fixed without detrimental impact to the rest of the system. (Colleague decoupling)

In the centralized elevator system, the ECS acts as the mediator between elevators. Elevator components (such as sensors) are considered part of the elevators’ entity and communicate with the ECS through their container elevators. (This is called a composition relationship)
On top of that, the ECS follows the “**Singleton**” Creational Pattern. Since the ECS acts as the mediator, there should always only be a single instance of the ECS at any given time.

For example, this is how the implementation could look like *(as shown in the UML)*:



```
static ECS& getInstance() {
    static ECS instance;
    return instance;
}
```

Distributed

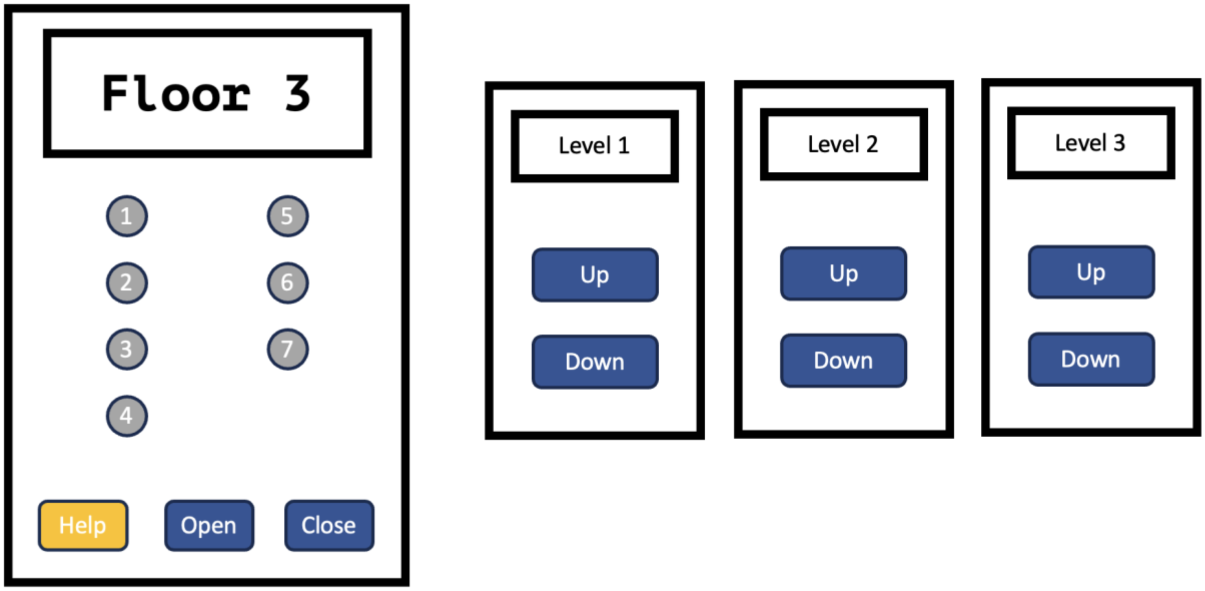
- Interactions between floors and elevators: **Observer Pattern**

In my version of the distributed elevator system, the Observer pattern facilitates communication between different components of the system. Each elevator act as observers and subscribe to updates from floors (floor sensors and elevator call buttons). When notified of a new request, the elevators evaluate themselves to figure out whether they are the best candidate to respond based on their current state (e.g., position, load).

Centralized vs Distributed

- Comparing the Centralized and Distributed systems, the elevators in the centralized system has a more surface-level and **limited** view of the state of the entire system, while the elevators in the distributed system have a more **complete** picture as to the current state of the system as a whole at any given time.
- As a result, it becomes easier to incorporate special features in individual elevators in the distributed system. The trade-off is the significant increase in complexity of inter-elevator communication (as compared to having the "ECS" mediator in Centralized).

GUI Sketch



C++ Files

Note to TA: Source code is included in src/*

Requirements Traceability Matrix

ID	Requirement	Related Use Case	Fulfilled By	Test
1	User presses "up" or "down" button.	Calling the Elevator (UC1)	Elevator, ECS, Floor, UIElements, Building	Click the “UP” or “DOWN” buttons in the “Floor Controls” section.
2.	Button illuminates and remains illuminated until elevator arrives.	Calling the Elevator (UC1)	Elevator, ECS, ToggleButton, Floor, UIElements	Observe that the button becomes disabled and turns blue.
3.	Elevator rings upon arrival, button light turns off.	Calling the Elevator (UC1)	Elevator, ECS, Floor, Elevator	Bell ring message displayed in Application Output.
4.	Elevator and floor doors open together; Doors remain open briefly for entry or exit.	Calling the Elevator (UC1)	Elevator, ECS, UIElements, MainWindow	On the right pane (Elevator Controls section), the Door Status display reflects the current door state.
5.	Doors close in sync after a bell rings; Sensors in doors ensure that doors are not obstructed before closing. *(If repeated obstructions occur, warning is issued via audio and text.)*	Calling the Elevator (UC1), Obstacle Detection (UC3)	Elevator, ECS, Elevator Display, UIElements, Floor	qInfo output prints door status whenever there’s a change. As well, the Door Status display shows a warning if the doors are obstructed.
6.	User boards the elevator; Weight sensor checks load.	Riding the Elevator (UC2)	Elevator, ECS, UIElements, ElevatorDisplay	Door Status display shows a warning if the elevator is overloaded.
7.	User selects "close" or "open" door or waits for automatic closure; Floor doors and elevator doors close simultaneously.	Riding the Elevator (UC2)	Elevator, ECS, ElevatorDoor, UIElements, MainWindow	If the buttons in the Door Control section are clicked, the elevator holds its movements, and a status update is printed in the console.

8.	User selects destination floor(s) on the panel; Elevator moves to selected floor.	Riding the Elevator (UC2)	Elevator, ECS, ToggleButton, UIElements, Elevator, Floor	The destination buttons are programmatically generated using code, instantiating the custom ToggleButton class, made by extending QPushButton. (programmatically generated to generate the right number of floors based on the defs.h configuration dynamically)
9.	Onboard display reads the current floor number; Audio system indicates floor.	Riding the Elevator (UC2)	Elevator, ECS, ToggleButton, UIElements, Elevator, Floor, ElevatorDisplay	Every 1000 milliseconds (unless specified otherwise in defs.h using the TIMER_DELAY config var), the QTimer signal executes a slot function in the ECS called updateTick. updateTick is responsible for calling the other update functions from other components (cascading).
10.	Elevator rings as doors open together; User exits, enters, or remains in elevator.	Riding the Elevator (UC2)	Elevator, ECS, ToggleButton, ElevatorDisplay, UIElements	Ring is printed to console, Elevator Door displays door status.
11.	Sensor detects excessive load and notifies elevator control system (centralized only).	Elevator Overload (UC4)	Elevator, ECS, UIElements, QDebug	ECS checks if the Overload checkbox is activated, and holds the elevators if true.
12.	Door remains open. Elevator alerts passengers with audio cues/instructions and displays message.	Elevator Overload (UC4)	Elevator, ECS, UIElements, ElevatorDisplay, ToggleButton, Floor, QDebug	Elevator checks a private member variable called "isDoorOpen", and holds its current status if true. If blocked, elevator will

				send a signal to ECS to notify.
13.	Elevator waits until weight decreases, normal operation resumes.	Elevator Overload (UC4)	Elevator, ECS, QTimer, UIElements	On overload clear, elevator is sent a message by ECS and returns to moving state, following the previously set destionation.
14.	User activates "Help" button, notifying control system; User then connects with safety service within 5 seconds. 911 is dialed if no response is received within 5 seconds.	Intercom Help Request (UC5)	Elevator, ECS, Building, Floor, UIElements, QTimer, ToggleButton	Help button in intercom panel attempts to connect to Building Safety Services. This is done by checking a checkbox in the Building Control Panel. The disconnect button becomes activated if a channel is open.
15.	User activates "Fire" signal or ECS receives “Fire” signal from the building *(centralized only)*. Elevator bypasses selected floors, heads to safe floor.	Fire Alarm (UC6)	Elevator, ECS, Building, Floor, UIElements, QTimer, QCheckBox	A checkbox in the Building panel sends a signal to the ECS, which then sends a signal to all floors, activated the fire evac protocol.
16.	Doors open simultaneously for safe evacuation. Elevator alerts passengers of emergency, directs evacuation.	Fire Alarm (UC6)	Elevator, ECS, UIElements, ElevatorDisplay, QLineEdit	Door Status Display shows doors are opening, and qInfo panel alerts the user to disembark.
17.	Passengers informed of power outage. Elevator switches to backup power and heads directly to safe floor; Doors open for evacuation.	Power Outage (UC7)	Elevator, ECS, UIElements, ElevatorDisplay, QLineEdit, Floor, Building, QCheckBox	A checkbox in the Building Control section activates the power outage protocol. All floor call buttons and destination buttons are disabled by the ECS, and elevators are sent a signal to recall to the safe floor (as defined in the defs.h file)