COMP3004A Final Exam Winter 2020

DUE DATE:

SUNDAY April' 27th'2020, 11:59 PM

Professor: - Vojislav Radonjic

Submitted by: - Khushal Kumar Singh
Student Id: - 101094697

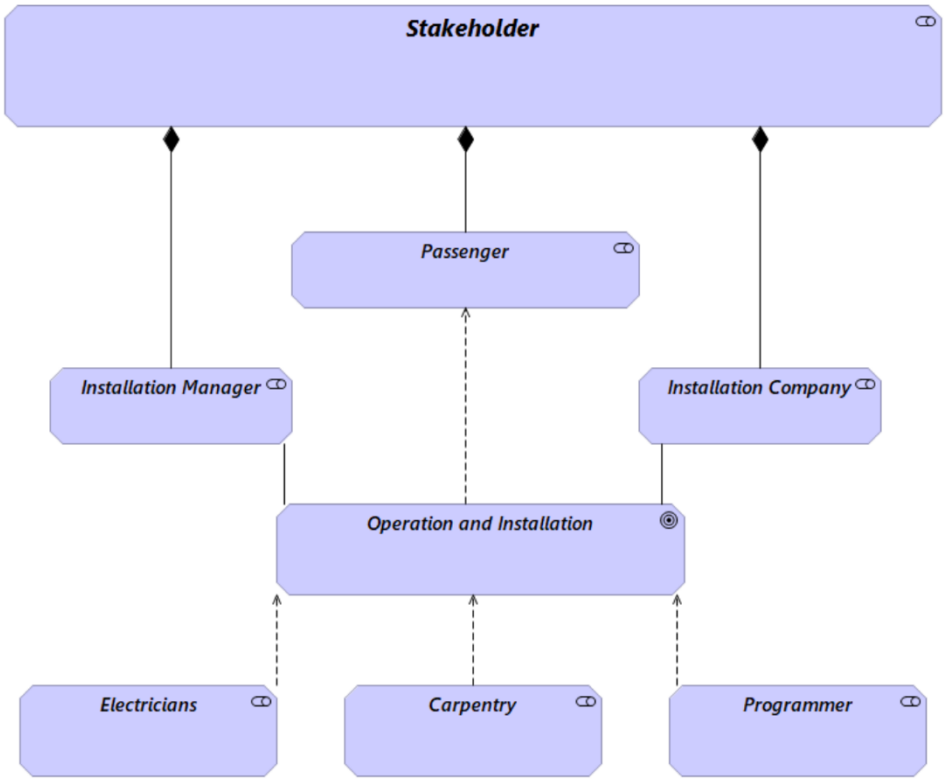
PART I: - THE THERAC-25: 30 YEARS LATER

A. No, we cannot say that software itself is safe or unsafe because it depends on the software requirements. If we run software on outdated machine it is supposed to hang(lag) or create problems, but it we run software on suitable machine or updated machine then it is supposed to work perfectly. The same thing happened with Therac-25 software. The software was used on therac-20 machine, that is why accident happened.
B. Safety in software development comes into play at the beginning of development. As it is increased by satisfying the software requirements, building safety into machines at the time of development.
C. We also cannot say that it is safer to reuse the software just because it has been used and tested extensively, software is safe and unsafe within a specific context. Maybe, the software can be safe in one system but unsafe in another system. Overall, safety is the quality of system in which software is used.
D. Using Object-oriented technology is the approach may or may not lead to safer software because software is context specific. For sure, OOP programming provides a lot of functionality: - Data encapsulation, etc. Also, for instance. Therac-25 software was unsuccessful of therac-20 when tested. But on Therac-25 it was successful.
E. It is better to do error handling first because safety of a software should be maximized, also it should be started with development of software. At each step, security and safety should be ensured.

PART: - II ELEVATOR INSTALLATION PROCESS MODELLING IN ARCHIMATE

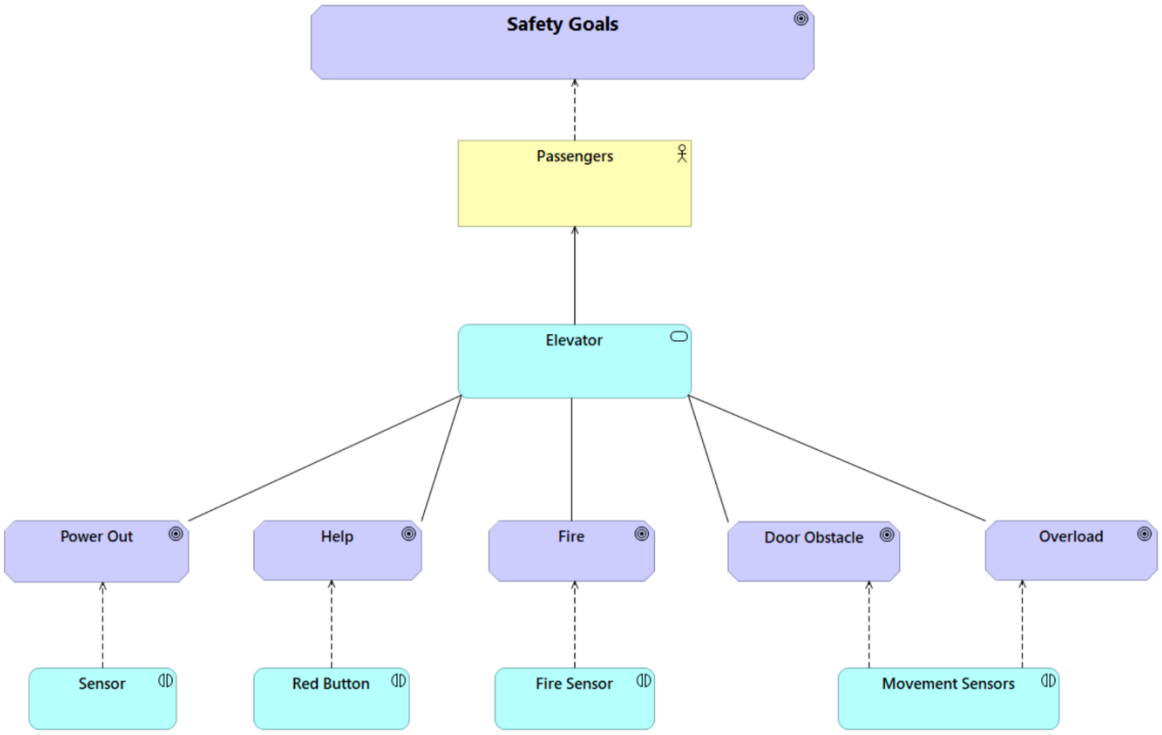
Model the stakeholders, goals, drivers and the process of installing an elevator as presented in the video.

STAKEHOLDERS DIAGRAM:



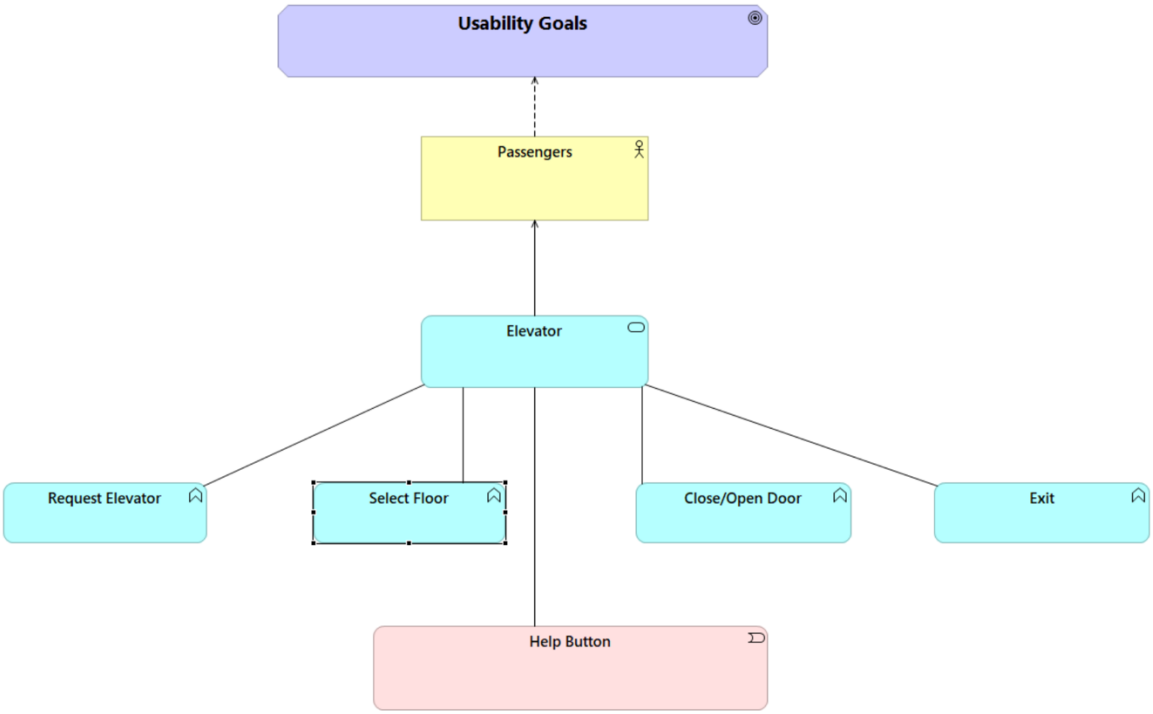
The above diagram represents the stakeholders: these can be the users, the people who built the elevator: masons, carpenters, steel workers, fire service vendors, electricians, steam fitters, and programmers.

SAFETY DIAGRAM:



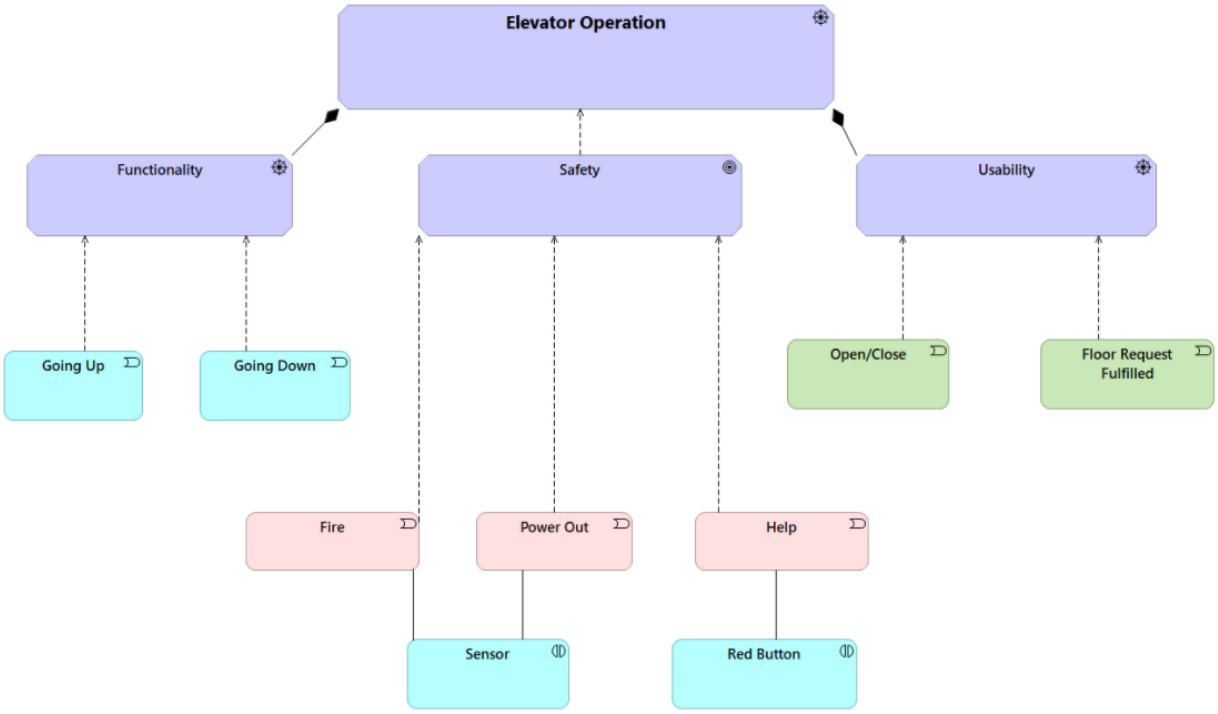
This diagram depicts the safety in functionality, it has been built at the start of the development of ECS. It has fire sensors for detecting smoke, help button to call for help, door obstacle handling, overloading, power out.

USABILITY GOALS: -



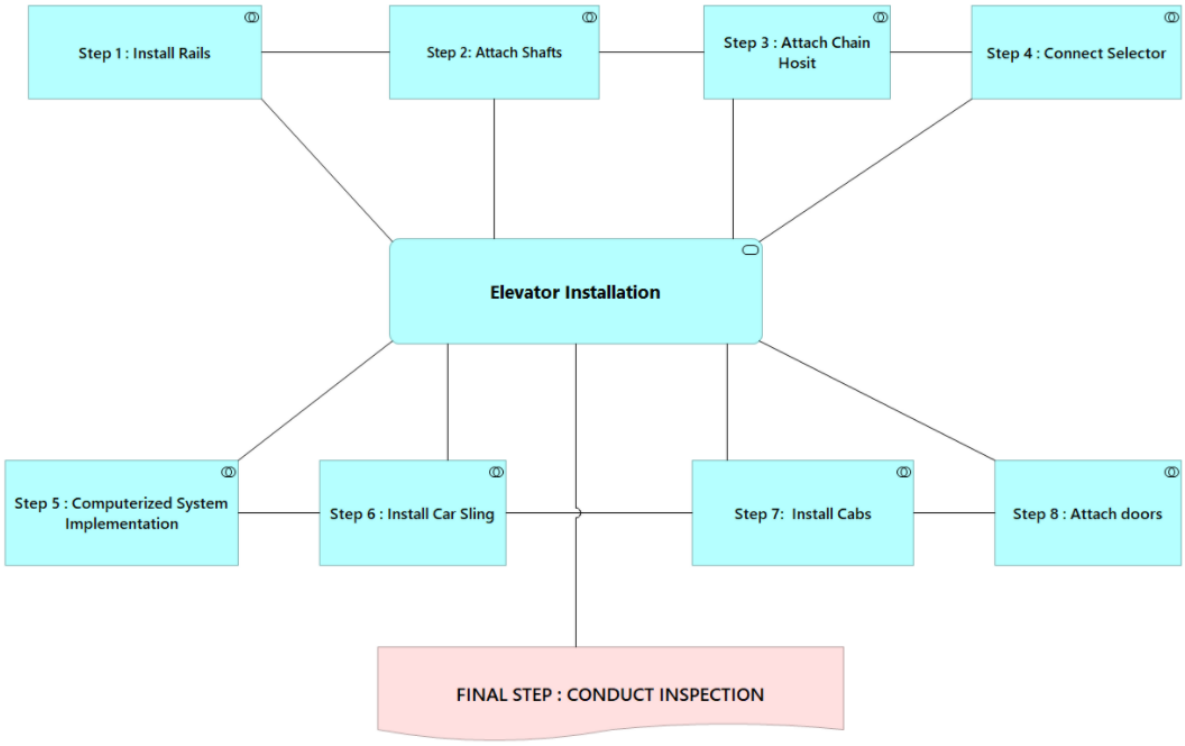
The diagram shows us the usability goals of safety. It enables us to use help button, selecting destination floor, opening and closing door, and requesting elevator using buttons.

DRIVERS:



This diagram shows the functionality of the elevators the safety measures, moving of elevator UP and Down changing floors, and the requests of elevators.

INSTALLATION PROCESS:



The above picture represents the Installation process, it is not just a single process, but it is built by the help of many stakeholders. Setting up needs a lot of hard work and precise measurements are required to make it function.

The Installation of elevator starts with precisely installing rear and spot brackets at rear and top point of the shaft, thereafter plumb line is inserted to line the lower brackets(which are the foundations), but the rear brackets should also be aligned straight and levelled. To be precise, the tolerance of deflection is only allowed to be plus minus $1/64^{th}$ of an inch. This process is repeated from floor to floor with guide rail attached to them with one-ton chain hoist. Effective communication between workers makes it possible to happen.

Secondly, at this point, up and down movement is checked, and oil is put into hydraulics so the piston can work on them. Also, sensors are installed to monitor the doors, and control system is installed during this compressed motion and thereafter, control pump is installed by which car or elevator performs styles cross heads and both the channels. As soon, as the door bug ensures the header of the door along the entrance of each landing floor then dome and ceiling units are installed after the strike and return column and front panel assembler.
Now, almost all the work is completed, and finishing is only left after this. So, the cab door is attached the operator to ensure smoothness. Moreover, Connections measurements are checked and adjusted, buttons and microprocessors are connected with control board. After this, Computer is programmed for function. All the safety measures are checked once again to give it a go for further inspection of certification.

After all this complex process and Evaluator comes for an inspection and performs 150-point test and gives it an authorization certification from regional office.

PART III: - ELEVATOR CONTROL SYSTEM

Use cases that capture the normal and exception-handling

Use case 1: Normal Usage

1. Passenger presses up or down button to go to any floor
2. Button lights up until elevator arrives
3. Passenger waits for elevator
4. Elevator comes at floor
5. Button light turns off
6. Elevator makes sound of ring while opening doors
7. Elevator door opens up
8. Passenger boards the elevator
9. Passenger presses the destination button.
10. Button lights up until the elevator has reached the destination floor.
11. Elevator make a sound of ring before closing doors.
12. Elevator door closes after 7 seconds.
13. Elevator digital screen updates as floor number are increasing or decreasing
14. Elevator arrives at destination floor
15. Button light goes off
16. Elevator makes a sound before opening doors
17. Elevator door opens
18. Passenger exits the elevator

Use case 2: Fire Emergency Use case

1. Passenger presses up or down button to go to any floor
2. Button lights up until elevator arrives
3. Passenger waits for elevator
4. Elevator comes at floor
5. Button light turns off
6. Elevator makes sound of ring while opening doors
7. Elevator door opens up
8. Passenger boards the elevator
9. Passenger presses the destination button

10. Button lights up until the elevator has reached the destination floor.
11. Elevator make a sound of ring before closing doors
12. Elevator door closes after 7 seconds
13. Elevator digital screen updates as floor number are increasing or decreasing
14. Fire alarm goes off
15. All button light goes off
16. Elevator stops at the nearest floor where smoke detector did not detect the smoke or fire.
17. Elevator continuously makes a sound before opening doors
18. Elevator door opens
19. Passenger exits the elevator and takes the stairs

Use case 3: Help button

1. Passenger presses up or down button to go to any floor
2. Button lights up until elevator arrives
3. Passenger waits for elevator
4. Elevator comes at floor
5. Button light turns off
6. Elevator makes sound of ring while opening doors
7. Elevator door opens up
8. Passenger boards the elevator
9. Passenger presses the destination button
10. Button lights up until the elevator has reached the destination floor.
11. Elevator make a sound of ring before closing doors
12. Elevator door closes after 7 seconds
13. Patient presses help button in the middle
14. All button light goes off
15. Elevator dials a call to the building security office for assistance
16. Phone rings for 5 seconds
17. If no one picks up phone, elevator calls 911 emergency no
18. Elevator stops at the ground floor
19. Door opens and remain opened until someone presses the close door button.

Other scenario: - if the button is pressed while the elevator doors are open then it will remain open until someone presses the close door button.

Use case 4: Overload – max weight 600Kg

1. Passenger presses up or down button to go to any floor
2. Button lights up until elevator arrives
3. Passenger waits for elevator
4. Elevator comes at floor
5. Button light turns off
6. Elevator makes sound of ring while opening doors
7. Elevator door opens up
8. 7 Passengers aboard the elevator
9. Elevator continuously beeps and Shows message on digital screen "Overloading"
10. 2 passengers exit the elevator, message removed from screen.
11. Beeping stops
12. Passenger presses the destination button.
13. Button lights up until the elevator has reached the destination floor.
14. Elevator make a sound of ring before closing doors.
15. Elevator door closes after 7 seconds.
16. Elevator digital screen updates as floor number are increasing or decreasing.
17. Elevator arrives at destination floor
18. Button light goes off
19. Elevator makes a sound before opening doors
20. Elevator door opens
21. Passenger exits the elevator

Use case 5: Door Obstacle

1. Passenger presses up or down button to go to any floor
2. Button lights up until elevator arrives
3. Passenger waits for elevator
4. Elevator comes at floor
5. Button light turns off
6. Elevator makes sound of ring while opening doors
7. Elevator door opens up
8. Passenger board the elevator
9. Passenger presses the destination button
10. Button light turns on
11. Passenger put something in front of sensor of doors or stops the door
12. Elevator wait for 7 seconds
13. Obstacle comes in between doors
14. Elevator Door opens
15. Elevator makes sound
16. Elevator wait for 10 seconds
17. Elevator beeps continuously while closing doors
18. Elevator digital screen updates as floor number are increasing or decreasing
19. Elevator arrives at destination floor
20. Button light goes off
21. Elevator makes a sound before opening doors
22. Elevator door opens
23. Passenger exits the elevator

Other scenario: - If the user continuously putting hands in between doors while closing, after every 7 seconds. It will only open back door 3 times. For the fourth time it will continuously beep and close doors whether something come in between and will proceed only after the doors are closed completely.

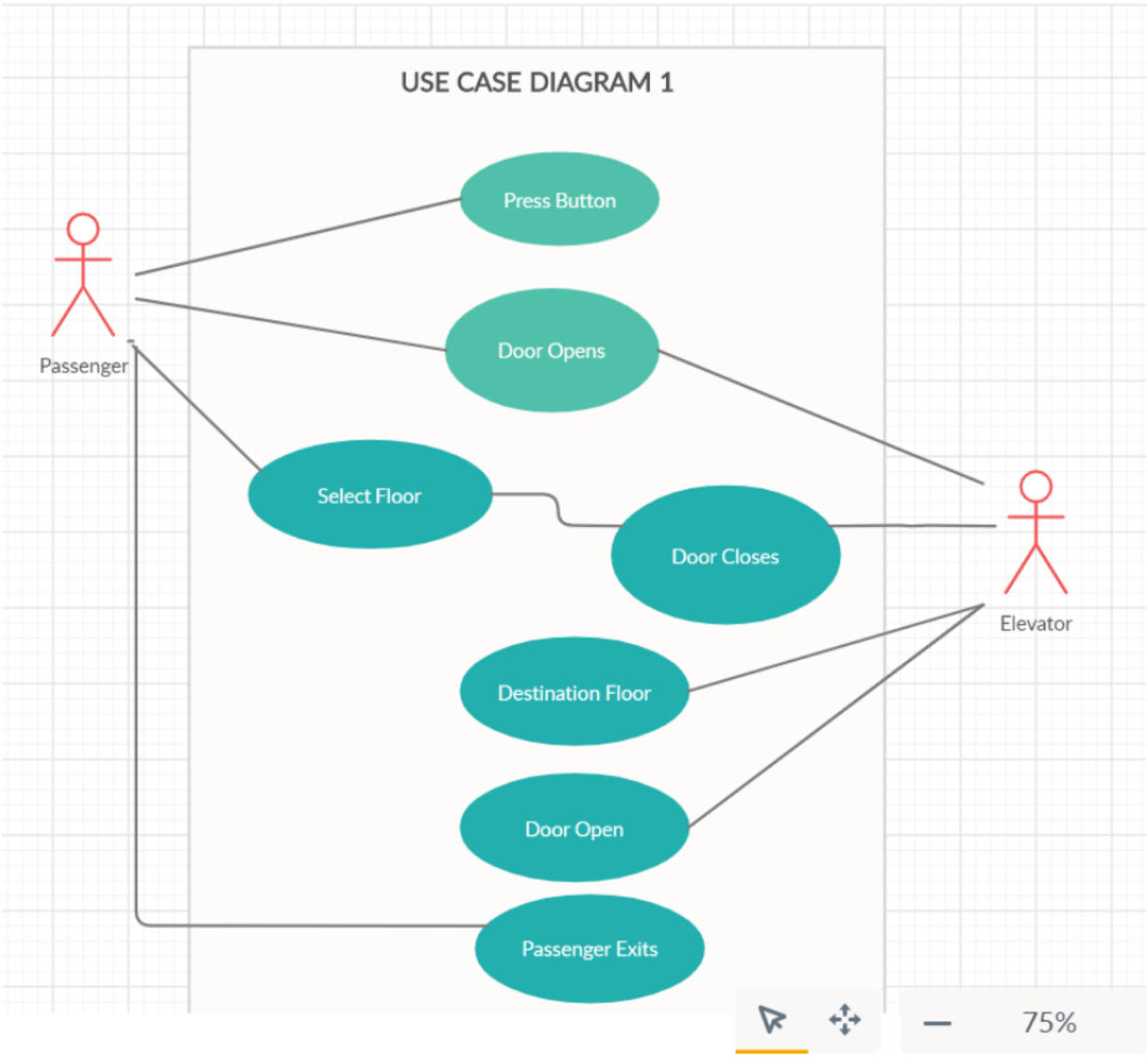
Use case 6: Power shut down

1. Passenger presses up or down button to go to any floor
2. Button lights up until elevator arrives

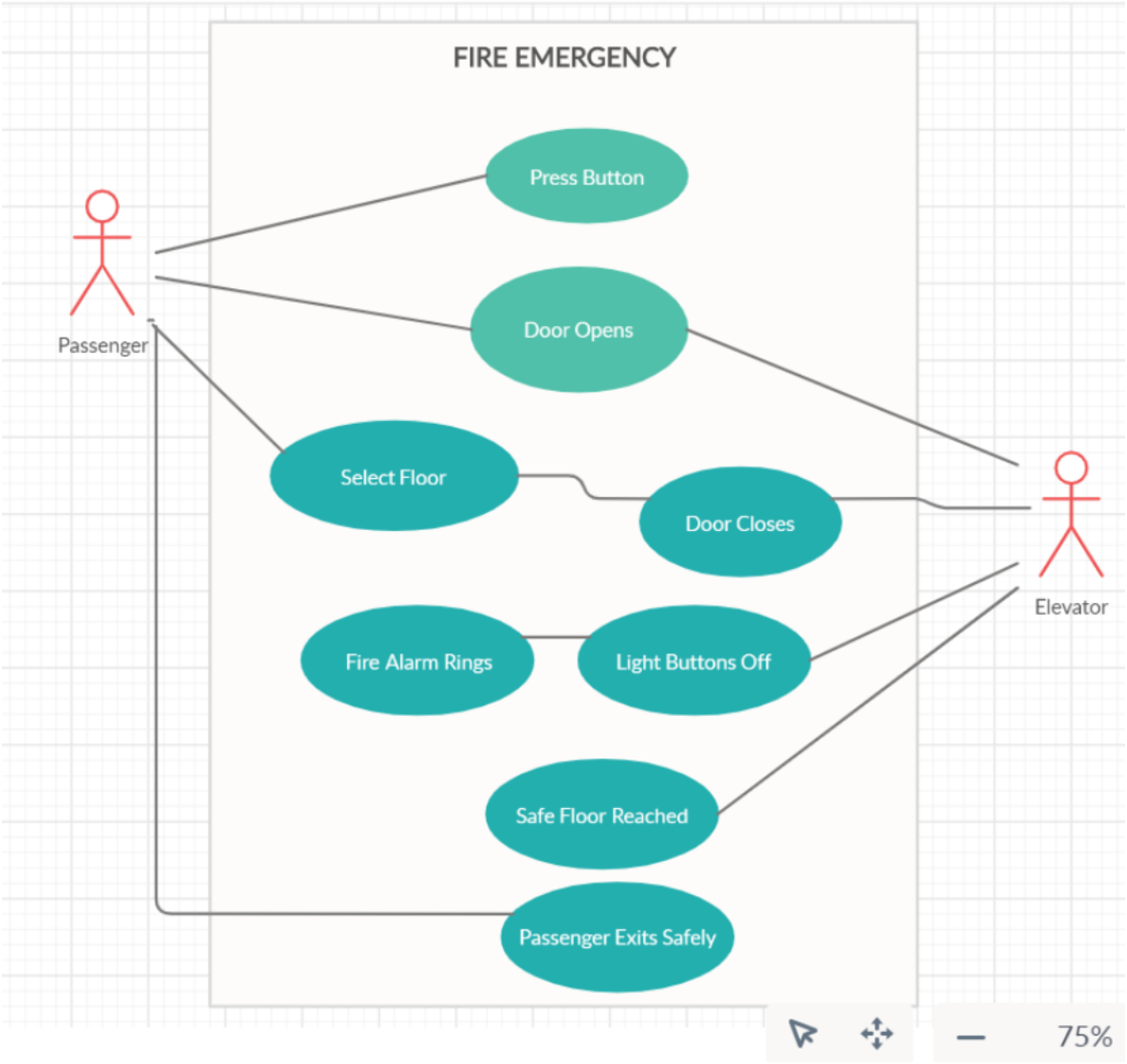
3. Passenger waits for elevator
4. Elevator comes at floor
5. Button light turns off
6. Elevator makes sound of ring while opening doors
7. Elevator door opens up
8. Passenger board the elevator
9. Passenger presses the destination button
10. Button light turns on
11. Passenger put something in front of sensor of doors or stops the door
12. Elevator wait for 7 seconds
13. Power cuts off, elevator stops.
14. Button light turns off
15. Backup battery activated
16. Message shown on digital screen “POWER OUT PLEASE WAIT”
17. Audio message is also played.
18. Elevator arrives at safe/default floor
19. Button light goes off
20. Elevator makes a sound before opening doors
21. Elevator door opens
22. Passenger exits the elevator

The use case diagram that relates these use cases (from step 1).

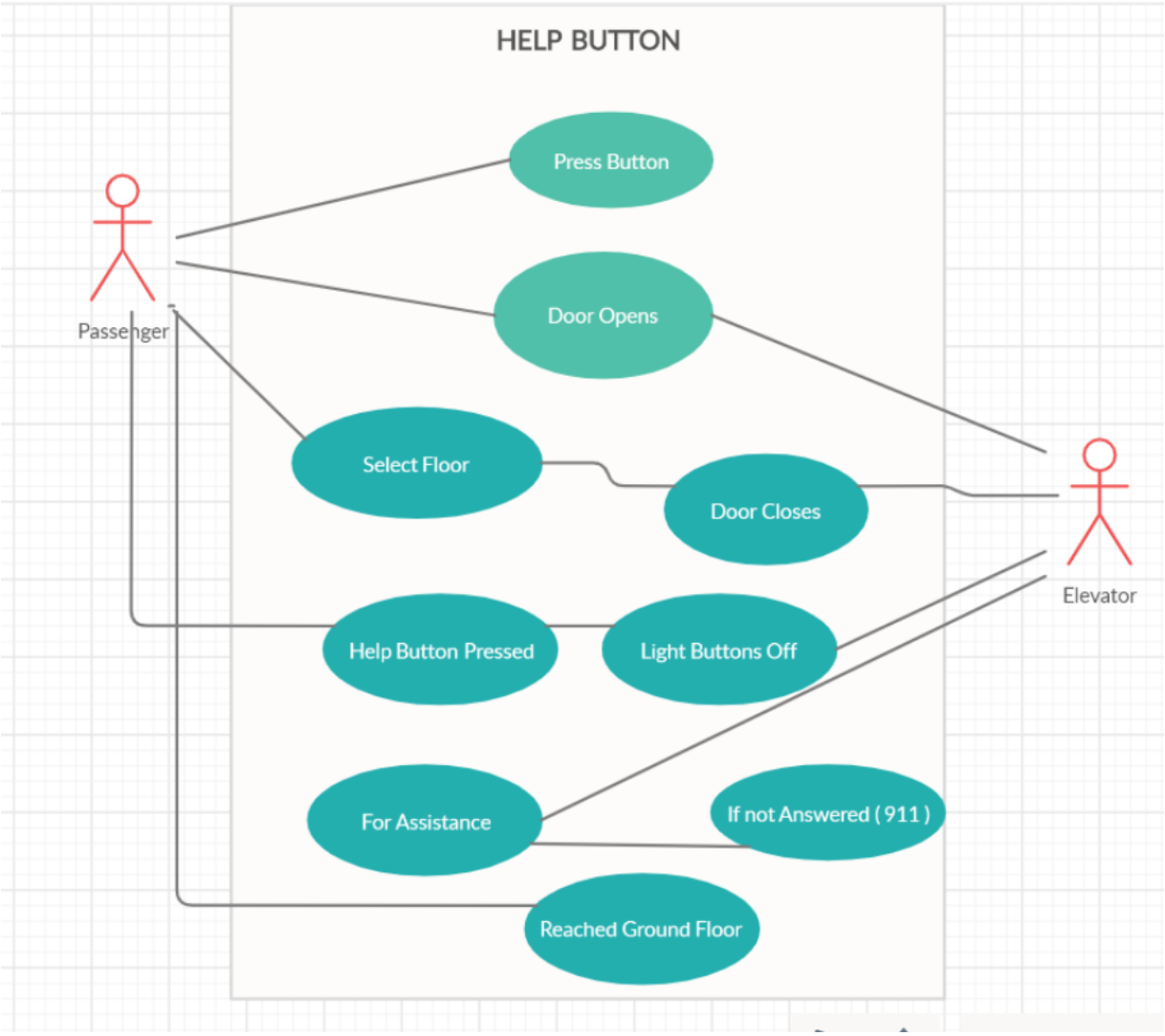
Explanation of use case 1: - It is the normal daily use case, when passengers just want to go to their destination floors and the functionality of Elevator. These are the steps that are going to take place whenever passengers want to go somewhere using elevator in normal circumstances.



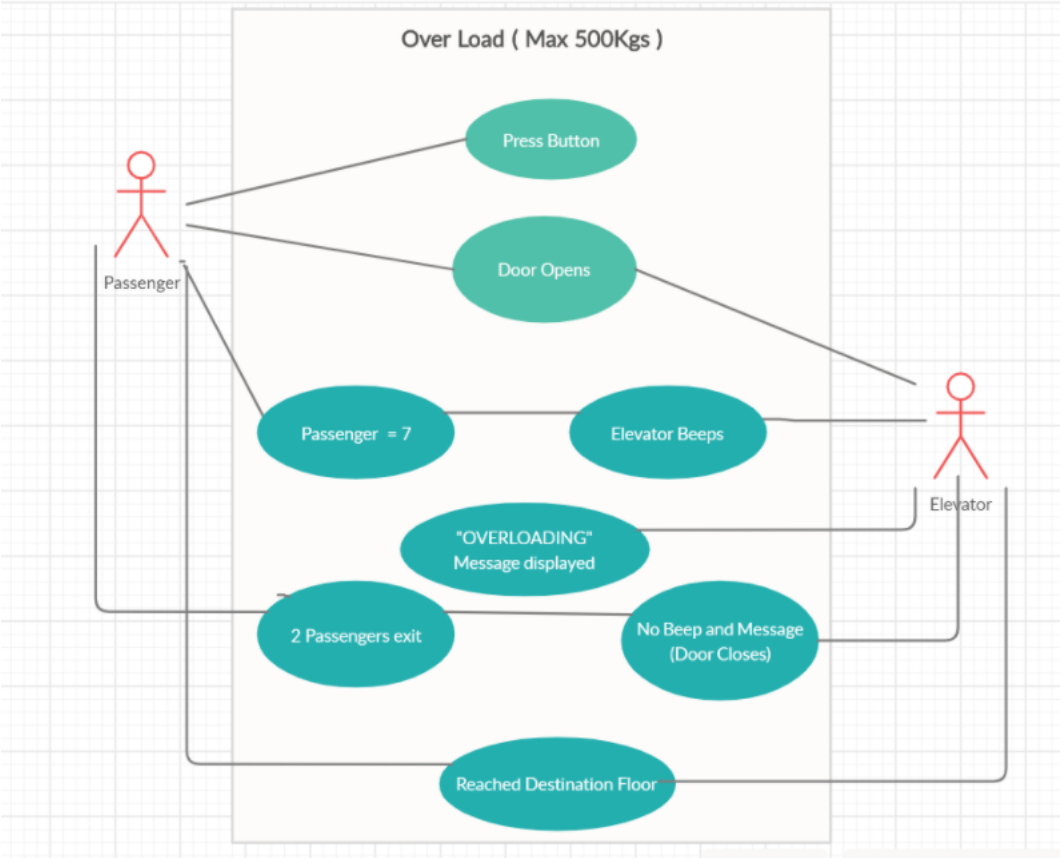
Explanation of use case 2: - This use case will be used when there is an emergency situation in building such as fire or smoke detectors beeps, these are the steps or functionality that elevator will follow at that time.



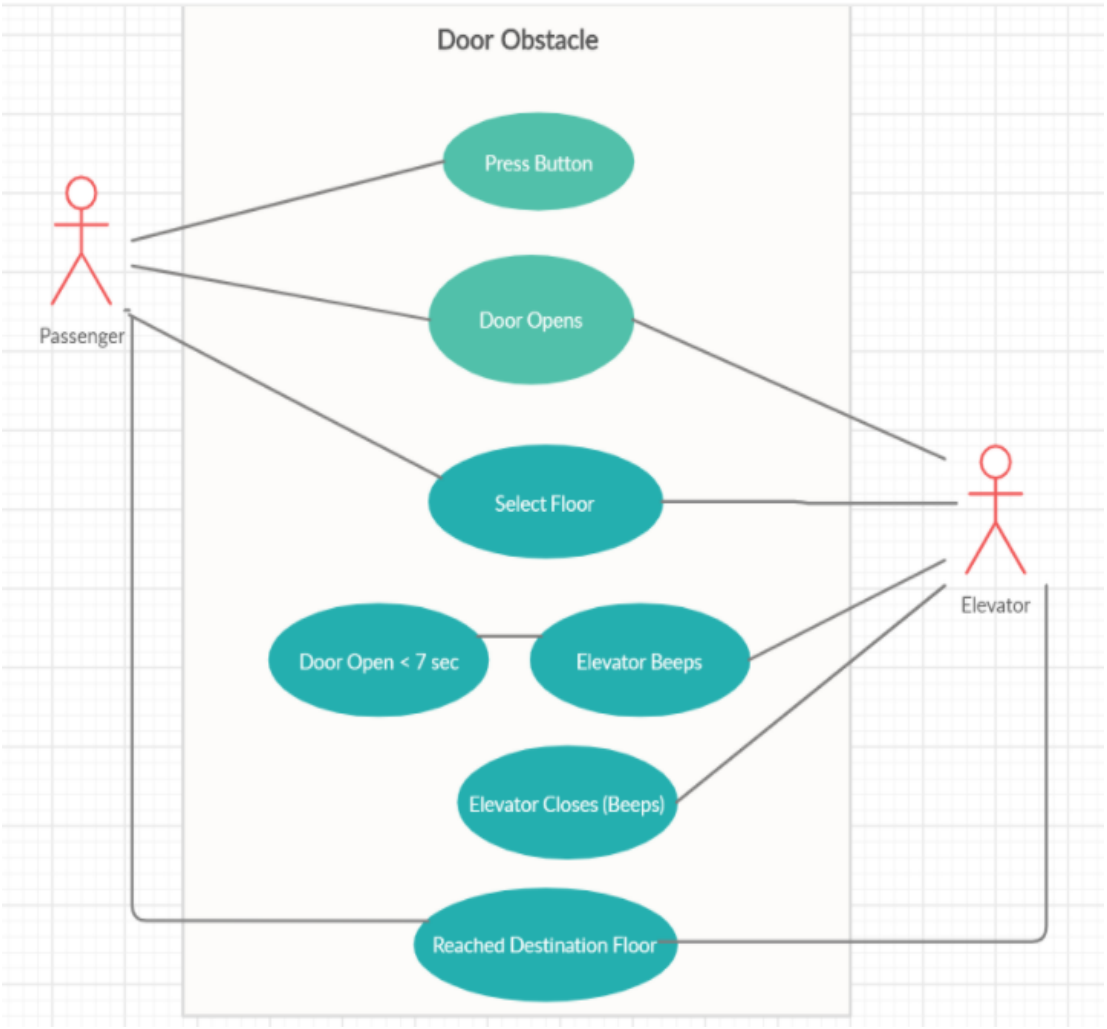
Explanation of use case 3: - This will happen when someone will press the help button. It can be anyone like injured person, disable person who needs instant help. In this type of scenarios these steps are performed.



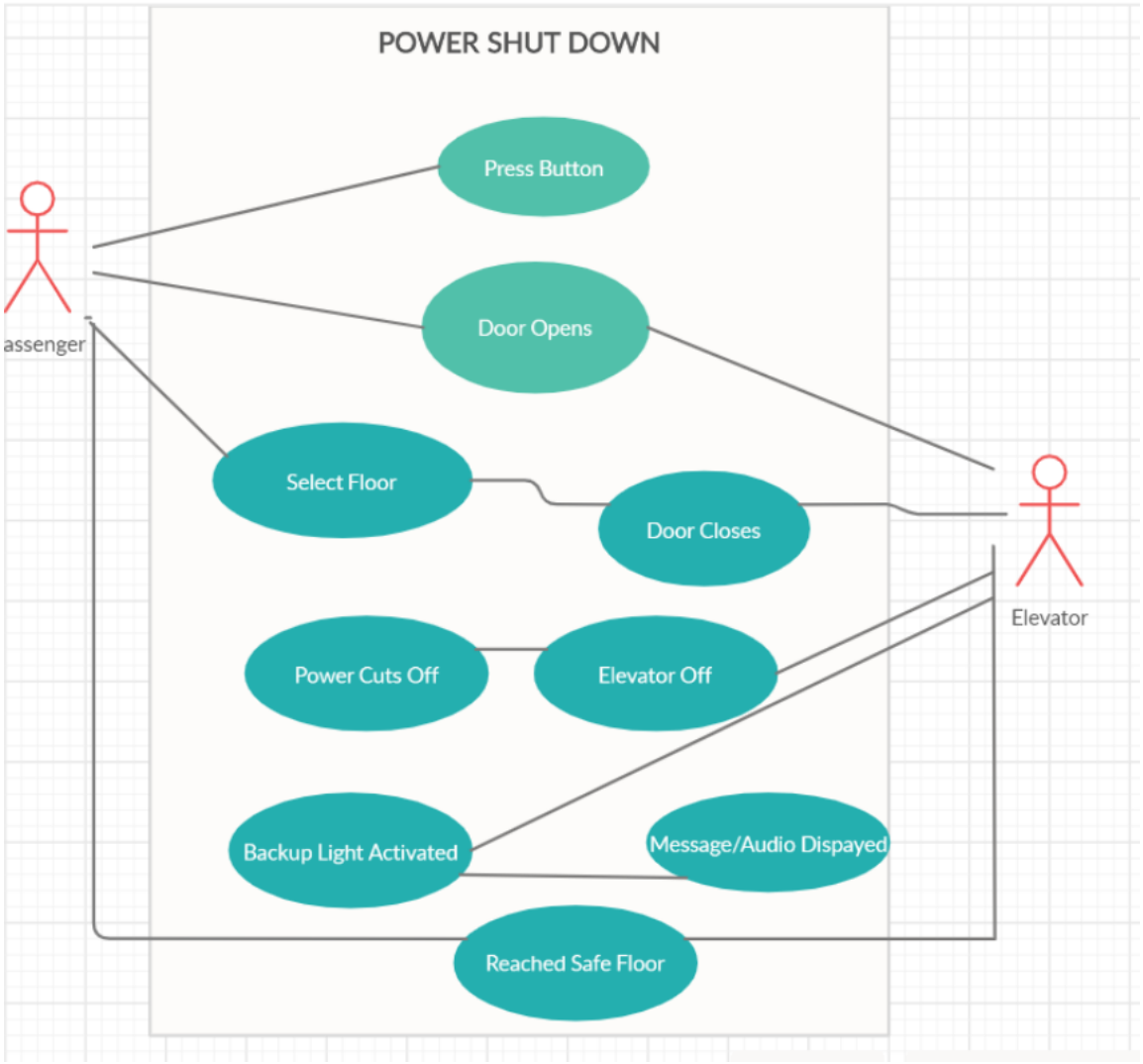
Explanation of use case 4: - Whenever a lot of people will enter the elevator and the maximum limit of weight is increased then the elevator will beep continuously until and unless the weight is less than the limit. In that case these steps will be followed by elevator.



Explanation of use case 5: - this is the case when door obstacles happen while closing of elevator doors. Those are the steps that will be followed in that case.

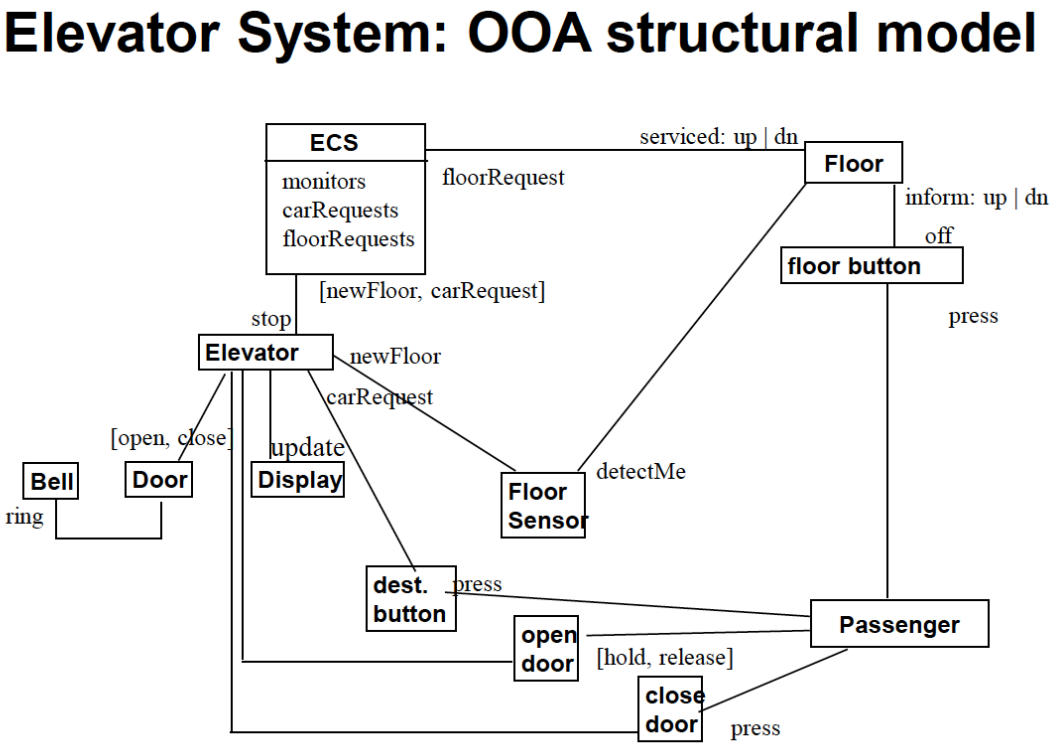


Explanation of use case 6: - This is one of the main use cases, as power can be cut off anytime and passengers can be very scared. In that case backup protocol of elevator will be followed. Below diagram describes that.



Design of a control system that realizes the required behavior (see below). Include enough detail so that it is clear to a programmer how to proceed to a C++ implementation. Use any UML models you see fit.

Here we have taken a snippet of elevator control system from lecture slides:

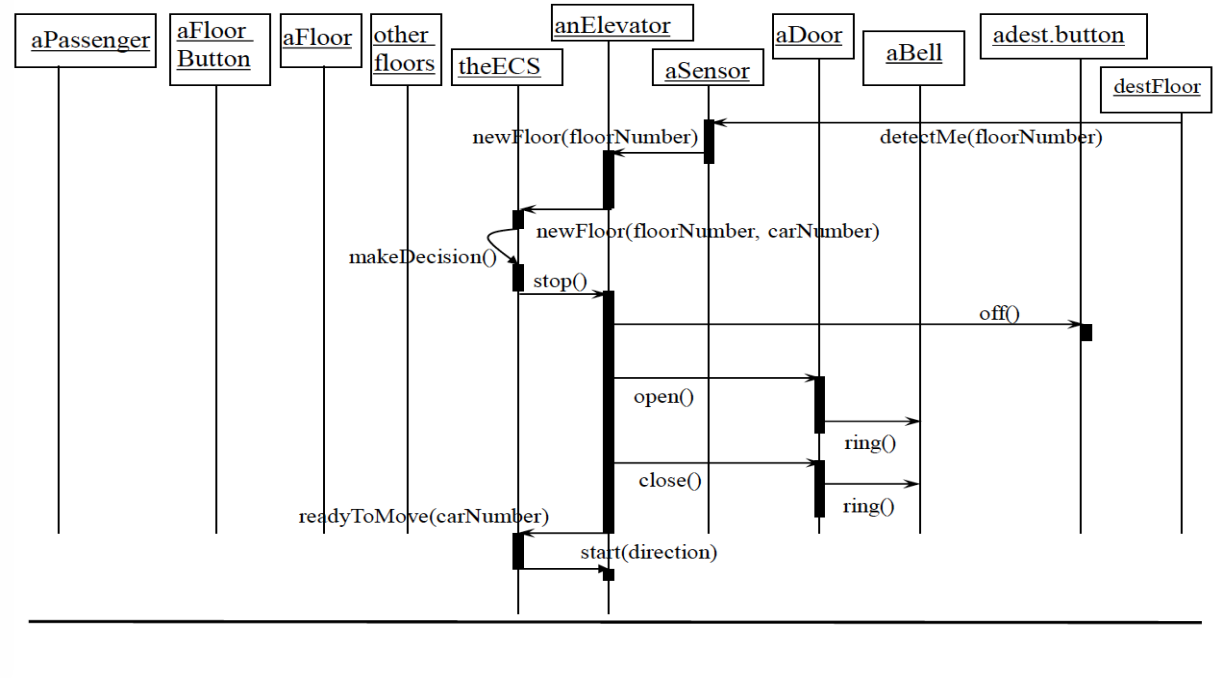


Based on this we have made UML, the classes required to perform each function at least all that fall into use cases.
The Elevator control system basically is the control class, it controls the emergency procedures, and elevator i.e. it has object of emergency class and elevator class. The ECS class has monitor variable which monitors the elevator and accept requests from passengers when they press any kind of button. Thereafter, we have elevator class, in which it has the elevator number because we have m elevators. It also has door status variable which will tell if the door has been closed after the default time or not. It takes input from floor button and updates the display. It also has an object of Emergency class if the emergency protocols are

needed then Emergency class controls the procedures and takes steps accordingly.

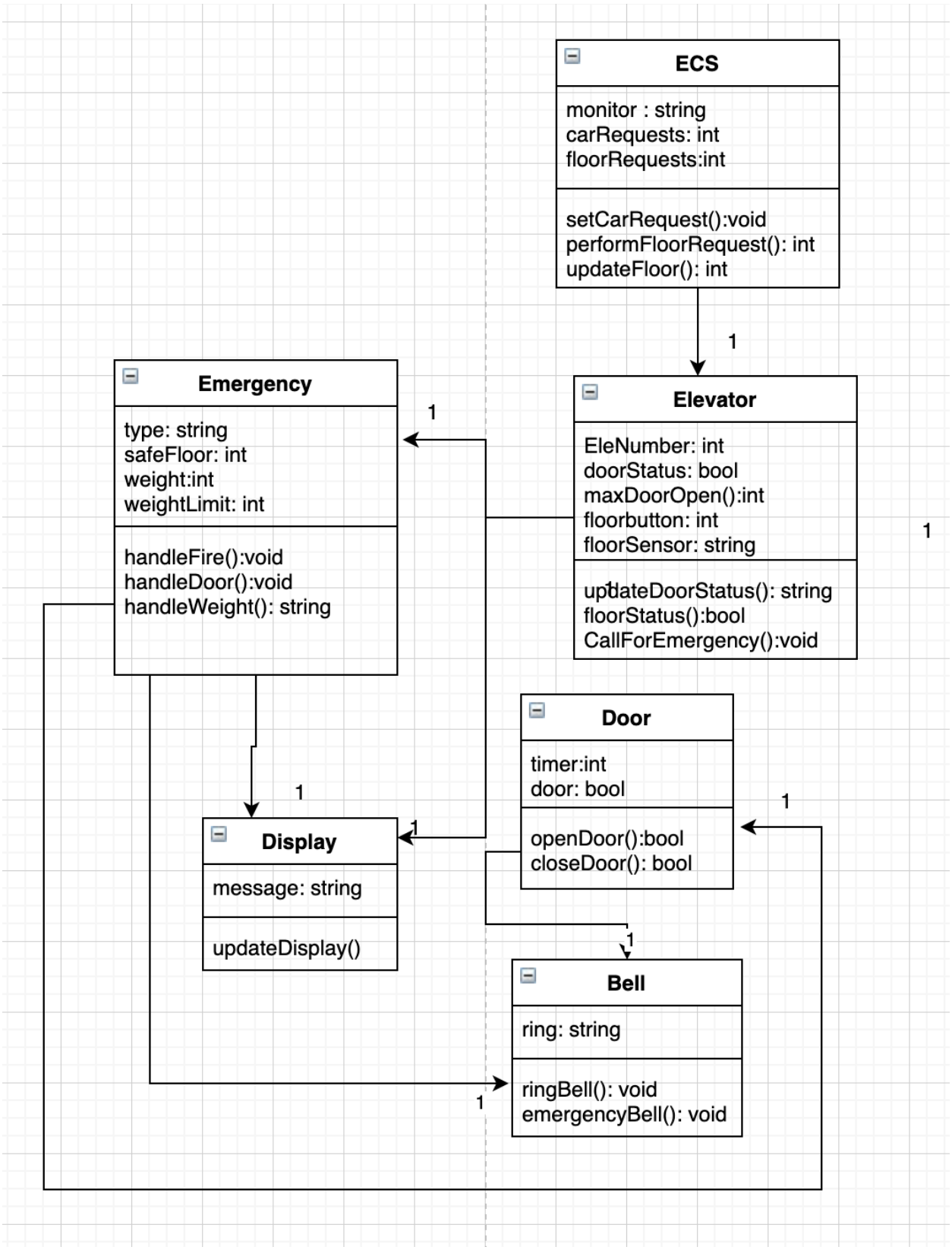
We also have emergency class, which has a type as soon as it is called its constructor will take an string argument as type of emergency whether it is help or call 911, or fire emergency. It has all the handler functions of emergency.

Moreover, we have Display, door, and Bell classes that will update the display, when called, it will show the message on screen. Doors will be responsible for opening, closing of doors, and Bell class will be responsible for and audio output function.



Below we have a UML, which depicts every class and variables that are shown in details.

UML Diagram:


ECS
monitor : string
carRequests: int
floorRequests:int
setCarRequest():void
performFloorRequest(): int
updateFloor(): int
1
Emergency
type: string
safeFloor: int
weight:int
weightLimit: int
handleFire():void
handleDoor():void
handleWeight(): string
1
Elevator
EleNumber: int
doorStatus: bool
maxDoorOpen():int
floorbutton: int
floorSensor: string
updateDoorStatus(): string
floorStatus():bool
CallForEmergency():void
1
1
Door
timer:int
door: bool
openDoor():bool
closeDoor(): bool
1
1
Display
message: string
updateDisplay()
1
Bell
ring: string
ringBell(): void
emergencyBell(): void
1

Define classes and their interfaces in C++.

Elevator Control System (ECS) Class Definition:

```
#define ECS
#ifndef ECS

#include "Elevator.h"

class ECS{

  private;
    Elevator * ele; //elevator instance
    string monitor;
    int carRequests, floorRequests;

  public:

    void setCarRequest();
    int performFloorRequest();
    int updateFloor();

};
#endif
```

Elevator Class definition:

```
#define ELEVATOR
#ifndef ELEVATOR

#include "Display.h"
#include “Door.h”


class Elevator{

  private;
        int EleNumber;
        bool doorStatus;
        int maxDoorOpen;
        int floorbutton;
        string floorSensor;
        Emergency * emer;
        Display * d;

  public:

        string updateDoorStatus();
        bool floorStatus();
        void CallForEmergency();
};
#endif
```

Emergency Class Definition:

```
#define EMERGENCY
#ifndef EMERGENCY

#include "Display.h"
#include “Door.h”


class Emergency{

  private;

                string type;
                int safeFloor;
                int weight;
                int weightLimit;
                Display * dis;
                Bell * b;

  public:
                void handleFire();
                void handleDoor();
                void handleWeight();

};
#endif
```

Display Class Definition:

```
#define DISPLAY
#ifndef DISPLAY

class Display{

  private;
            string message;
  public:

            void updateDisplay();

};
#endif
```

Door Class Definition:

```
#define DOOR
#ifndef DOOR

class Door{

  private;
            int timer;
            bool door;
            Bell * ringIt;
  public:

            bool openDoor();
            bool closeDoor();
};
#endif
```

Bell Class Definition:

```
#define BELL
#ifndef BELL

class Bell{

  private;
        string ring;

  public:
        void ringBell();
        void emergencyBell();

};
#endif
```