SYSC 3303 Final Report

Elevator System

Instructor: Dr. Greg Franks

Group L2-4

Omar Agamy 101088548

Mohamed Mahmoud 101040767

Taher Shabaan 101073767

Hassan Hassan 101095571

Ahmed Abdelrazik 101103048

Date: April 14, 2021

Table of Contents

1.0 Breakdown of Responsibilities

Iteration 1:

- **Hassan:** Simulation.java, Scheduler.java, Elevator.java, sequence Diagram
- **Omar:** FileDataTest.java, UserHandler.java, TextFileReader.java, ErrorHandler.java
- **Taher:** Elevator.java, FloorSubsystem.java, TextFileReader.java, InputInformation.java,
- **Ahmed:** FileDataTest.java, DataPathTest.java, Simulation.java, UML class Diagram
- **Mohamed:** DataPathTest.java, input.txt, FloorSubsystem.java, Scheduler.java

Iteration 2:

- **Ahmed:** floor and scheduler subsystems
- **Hassan:** elevator subsystem
- **Taher:** elevator subsystem , unit tests
- **Mohamed:** elevator subsystem unit tests
- **Omar:** UML class, sequence, and state machine diagrams

Iteration 3:

- **Ahmed:** UDP in scheduler subsystem, ElevatorCommunicator, ElevatorSubThread, FloorCommunicator, and FloorSubThread and additions to Scheduler, MessageParser in src/dataSystems, new unit test classes TestMessageParser, TestSchedulerElevatorCommunication, TestSchedulerFloorCommunication.
- **Mohamed:** Added support for Multiple elevators in elevator subsystem . Refactored the code to improve logging messages and make them more clear and understandable.
- **Taher:** UDP in floor subsystem and new test case in floorButton.java called 'testPressFloorButton'.
- **Hassan:** Added more message types and handlers in 'ElevatorSubThread'. scheduler calls in 'Elevator' to utilize UDP.
- **Omar:** New state machine diagrams for elevator, floor, and scheduler systems, updates to overall system class diagram

Iteration 4:

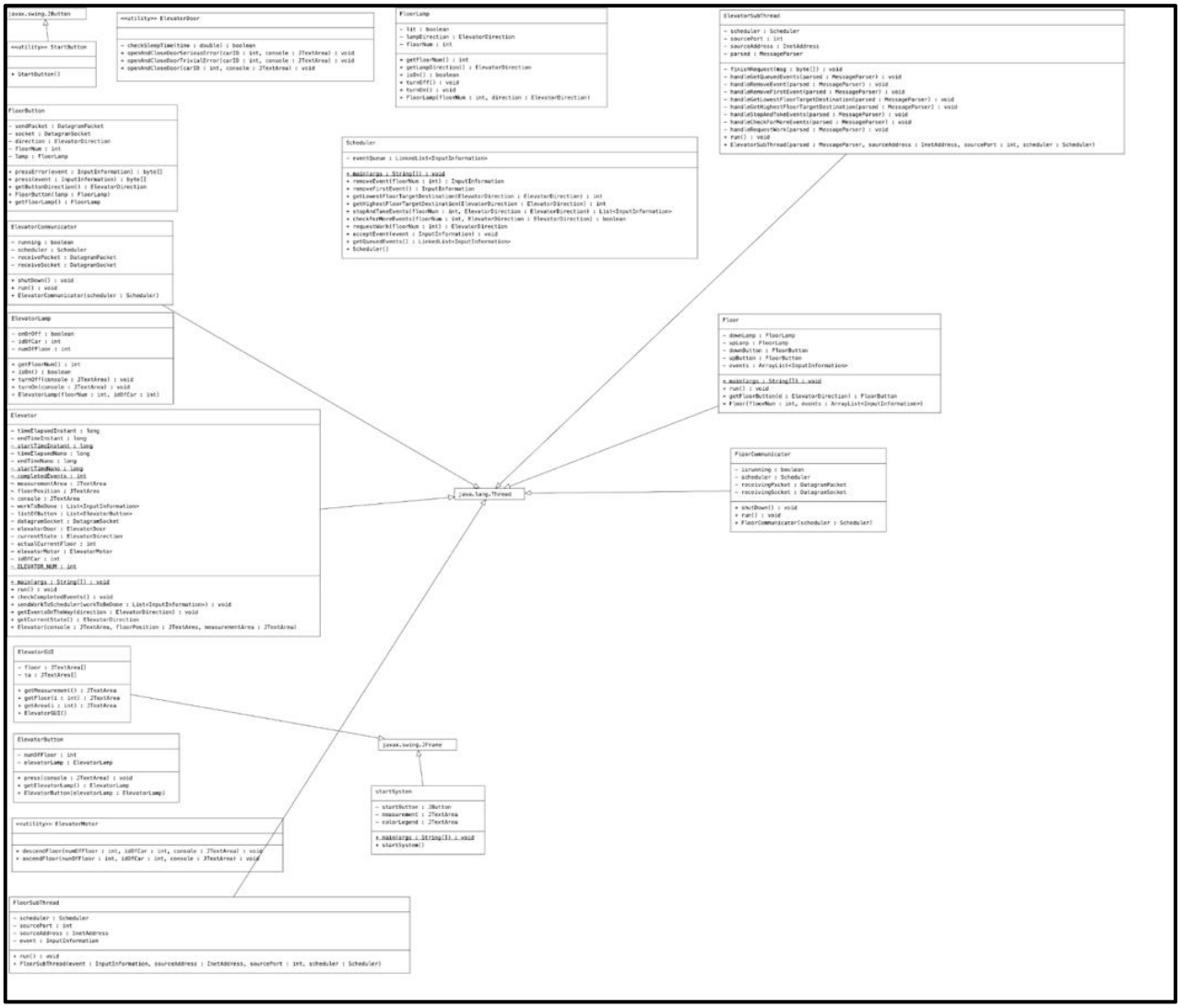
- **Omar:** implemented the transient error, UML class, sequence, and state machine diagrams
- **Mohamed:** implemented the serious errors, elevator subsystem unit tests
- **Ahmed:** timing diagrams, floor and scheduler subsystems
- **Taher:** implemented test cases, elevator subsystem , unit tests
- **Hassan:** implemented the transient error, elevator subsystem

Final submission:

- **Omar:** implemented the 4 consoles and the background color, UML class, sequence, and state machine diagrams.
- **Mohamed:** implemented the consoles4 and the background color, elevator subsystem unit tests.
- **Ahmed:** implemented the elevator current position, floor and scheduler subsystems.
- **Taher:** implemented the elevator current position, elevator subsystem , unit tests.
- **Hassan:** implemented the measurements, elevator subsystem.

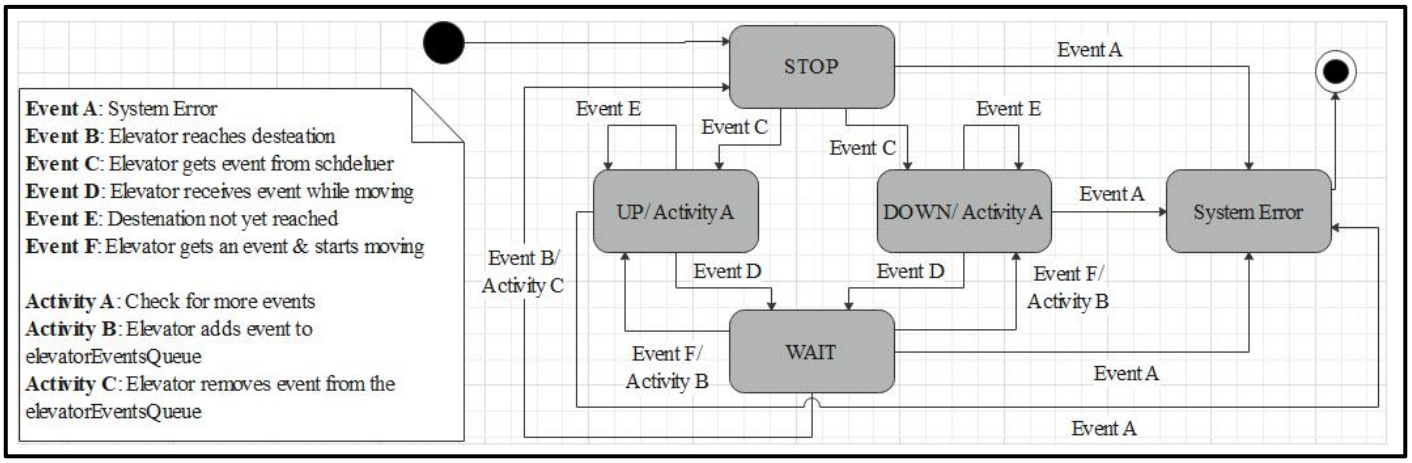
2.0 Diagrams

2.1 UML Class Diagram

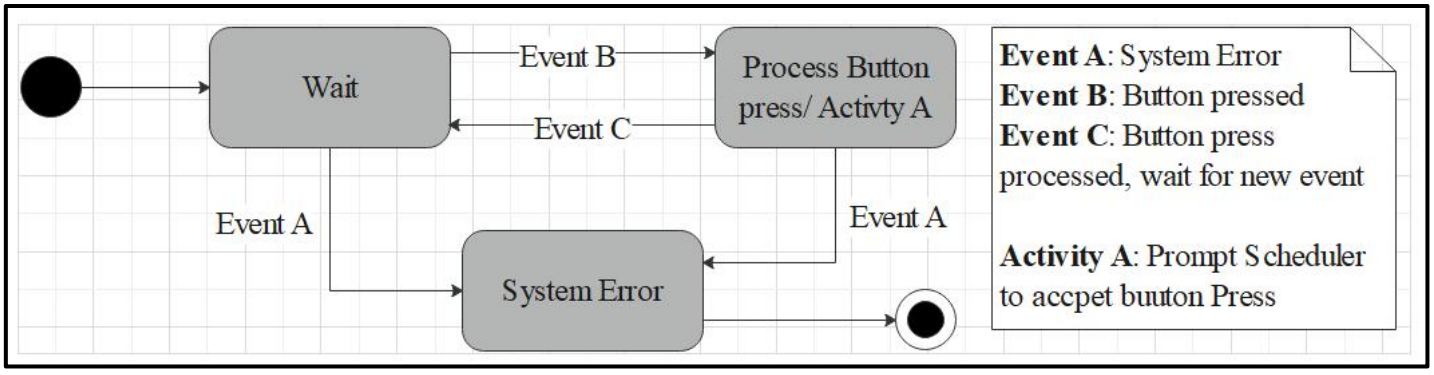


2.2 State Machine Diagrams

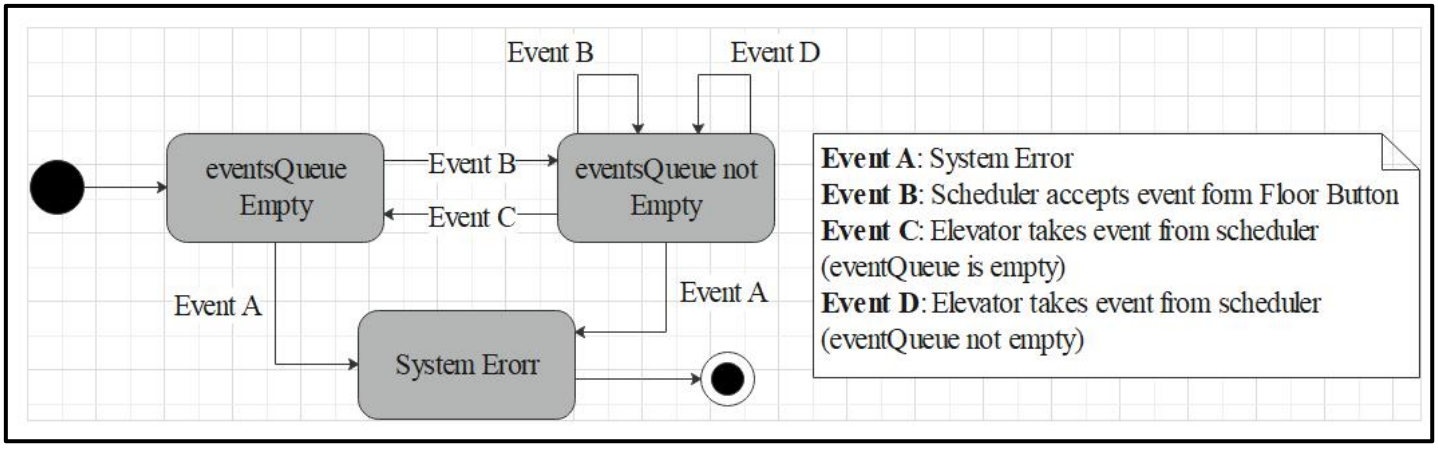
Elevator State machine



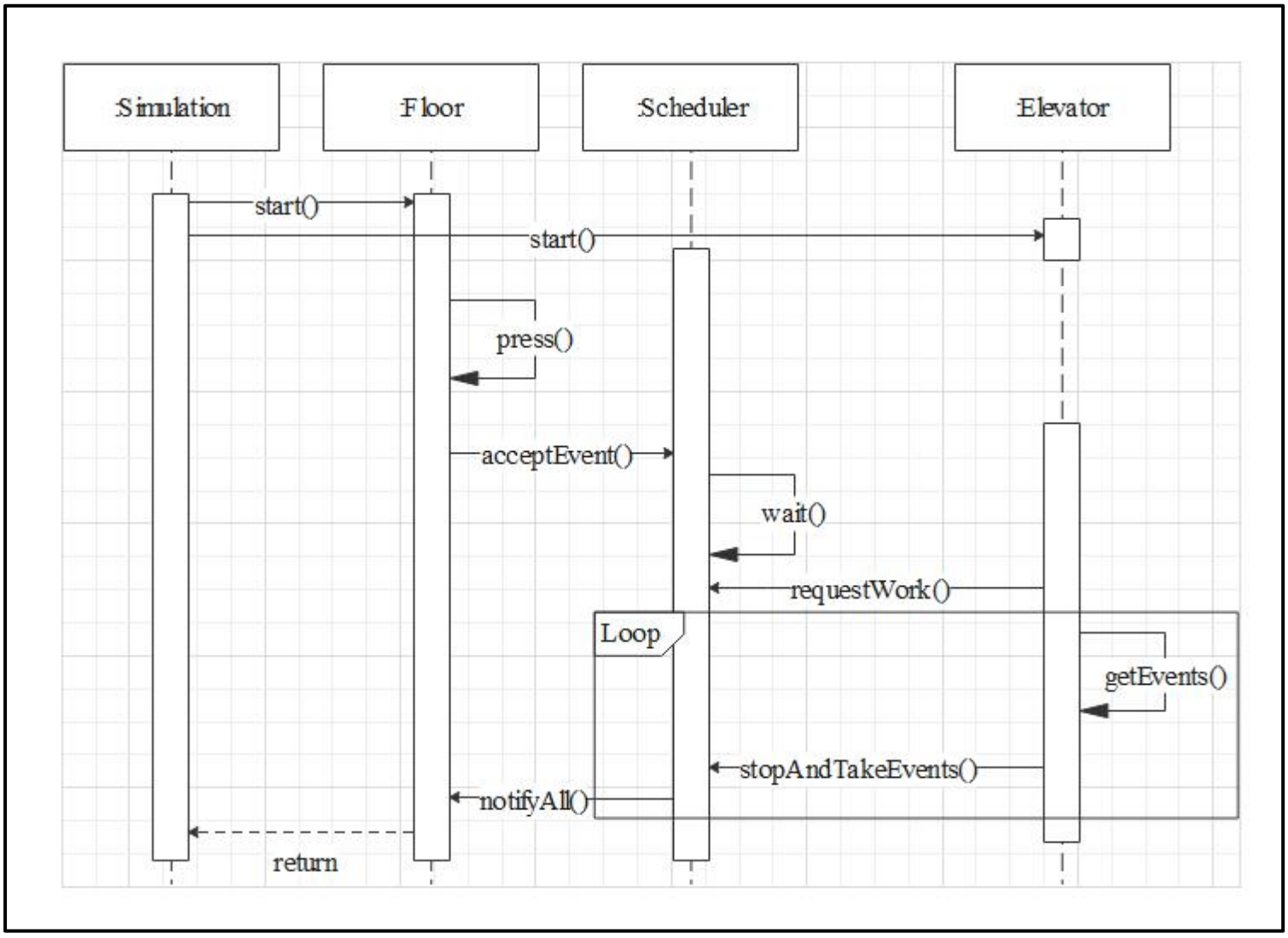
Floor State machine



Scheduler State machine

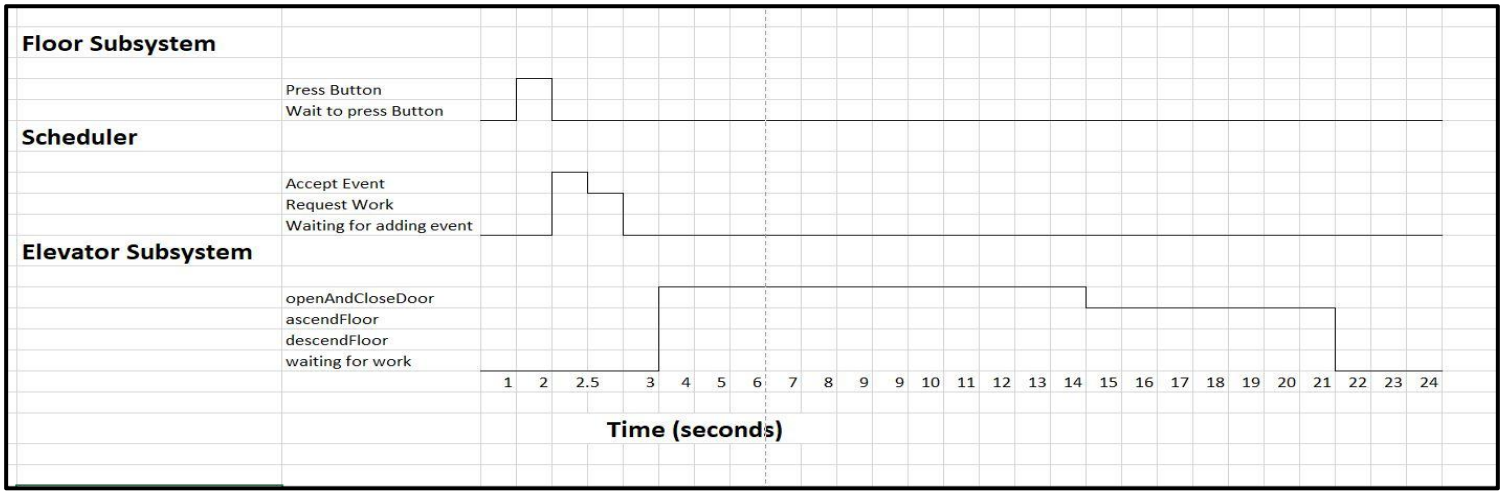


2.3 Sequence Diagram

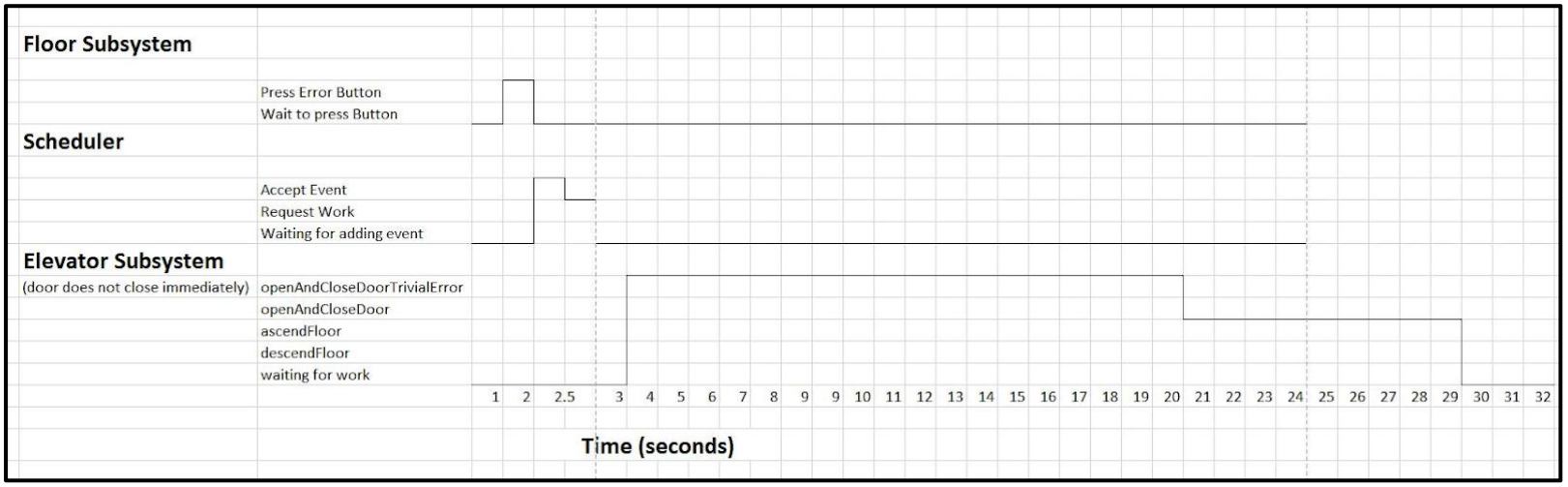


2.4 Timing Diagram

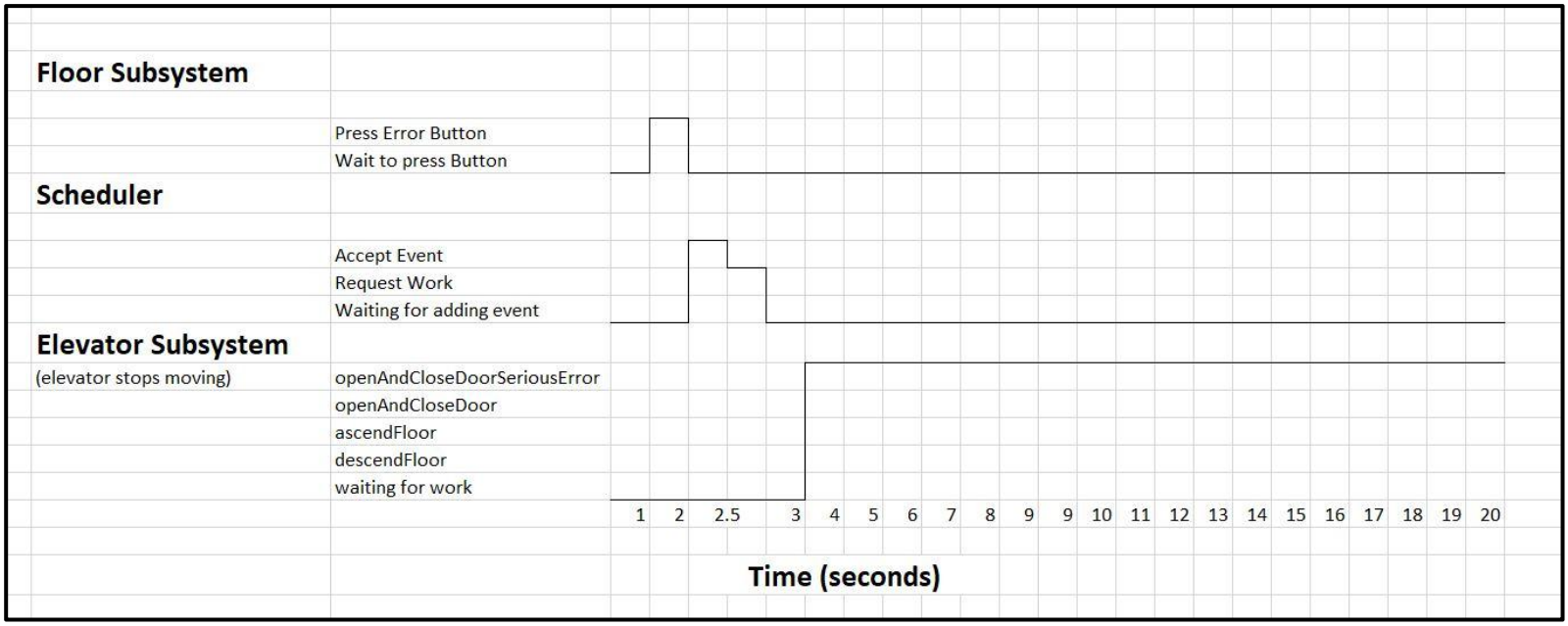
Scheduler Timing diagram



Door Error Timing Diagram



Elevator Error Timing Diagram



3.0 Setup and Test Instructions

3.1 Setup Instruction

1- Unzip the submitted file and import it into Eclipse (import as an existing file)
2- When launching Eclipse, open the directory in which you unzipped the file.
3- After this step, close the welcome page then go to File-à Open Projects from the Filesystem and select the project that was unzipped.
4- After this step the project should be available in the Package Explore menu.

3.2 Simulation Instructions single computer

1- Run to src/elevatorSubsystem/startSysem.java
2- Press on the start button that shows up in when the GUI runs
3- The system should be running and the elevators GUI panel should appear after the start button was pressed

3.3 Simulation Instructions multiple computer (More instructions)

1- To change the number of elevators in the system go to dataSystems/Configuration.java and change ELEVATORS_IN_BUILDING
2- To change the number of floors in the system go to dataSystems/Configuration.java and change FLOORS_IN_BUILDING
3- To change the speed in which the elevator goes from a floor to the other in the system go to dataSystems/Configuration.java and change SPEED_FROM_FLOOR_TO_ANOTHER
4- To change the speed in which the elevator doors open in the system go to dataSystems/Configuration.java and change SPEED_OF_DOOR
5- To add or change input events in the system go to src/input.txt

3.4 JUnit Testing Instructions

1- Right click on src/tests and then select Run As then select as a JUnit Test
2- The tests should be running then

4.0 Measurements

Please note that the following assumptions were made, both of which were taken from the project specification: each floor is 4m apart, and the riser (platform on which people stand) is 0.1778m (7 inches) tall.

4.1 Maximum Speed/Acceleration

Trial	Split Time (s)	Accumulated Time (s)	Speed (m/s)	Acceleration (m/s^2)
1	2.50	2.50	1.607535703	
	1.43	3.92	2.817001408	0.86344768
	3.07	6.99	1.318189542	-0.4943698908
	2.51	9.49	1.61	0.1176575528
	6.11	15.60	0.6556644845	-0.154819396
2	2.91	2.91	1.379410345	
	1.17	4.08	3.419803419	1.743256473
	2.97	7.04	1.352351351	-0.6985635363
	2.28	9.41	1.763114537	0.1809629454
	6.32	15.62	0.6339244216	-0.178895581
3	2.51	2.51	1.6	
	1.29	3.81	3.127	1.182270544
	2.99	6.78	1.337792644	-0.5977282144
	2.38	9.17	1.687763714	0.1470466686
	6.4	15.48	0.6349206350	-0.1668531028
4	2.58	2.58	1.544401543	
	1.31	3.90	3.076923079	1.178862718
	2.99	6.89	1.337792643	-0.5816489749
	2.39	9.28	1.680672270	0.1440670702
	6.45	15.8	0.6211180125	-0.1645270586

- **Maximum speed:**
    - The speed for split 2 is the highest
    - Split 2 mean: 3.109406985 m/s
    - Split 2 median: 3.1009615286 m/s
    - With 95% Confidence, the split 2 speed mean is between 2.86 m/s and 3.34 m/s based on the 4 samples recorded (95% Confidence Interval: 3.12 m/s ± 0.241)
- **Acceleration:**
    - Again, the acceleration between split 1 and 2 is the highest
    - Split 1-2 mean: 1.239159454 m/s^2
    - Split 1-2 median: 1.18051672 m/s^2
    - With 95% Confidence, the acceleration between split 1 and 2 mean are between 0.879 m/s^2 and 1.6 m/s^2 based on the 4 samples recorded (95% Confidence Interval: 1.24 m/s ± 0.364)

4.2 Average Opening/Closing Time

Trial	Loading time
1	12.75
2	13.24
3	12.73
4	12.91

Mean: (12.75 + 13.24 + 12.73 + 12.91) / 4 = 12.9075

Median: 12.75 + ((12.91 - 12.73) / 2) = 12.84

Mode is not applicable - to get multiple results with the same value would require an unrealistic amount of trials

With 95% Confidence, the loading time mean is between 12.8 seconds and 13.2 seconds based on the 4 samples recorded (95% Confidence Interval: 12.9 s ± 0.24)

5.0 Reflection on Design

5.1 Parts of The Design That We Like

Starting with what was done well in our design, we consider networking was done quite well. The scheduler handles the requests done from elevators and floors smoothly and it is done concurrently by spawning multiple different threads that are waiting for actions to be done. The scheduler's behavior is as effective as it was before introducing the UDP networking. This was shown when the system was first tested under UDP client server conditions as there were no shown differences in how the system acted before and after the networking was introduced.

The simultaneousness of the scheduler additionally ended up being successful all through the span of the task, as it generally scaled well to increases in the quantity of lifts in the framework, number of floors in the structure, and number of occasions added. Furthermore, the item chain of importance executed for the info occasions was very viable, as abstracting traveler occasions and blunder occasions under a typical interface simplified it to send over UDP. The regular interface was set up as a serializable class, which made it conceivable to simply send the occasions as bytes over the UDP associations and parse them back into objects on the less than desirable end. We also followed the principle of composition over inheritance which means our classes have instances of other classes instead of just extending those classes. That was done to provide more functionality to the classes. That was shown when the Elevator class had instances of other classes. When doing the GUI, the start button accelerated the approach of running the system by making a start button that would run the whole system in the GUI.

5.2 Parts of The Design That Could Be Better

Some aspects of the design could have been done a little better and from those aspects is the logic of schedule events. In the current implementation of the system, the events are scheduled between both the scheduler and elevator subsystem. What would be better if the scheduler subsystem contained all the logic of scheduling the events. One more advancement that could be done is to the floor class which only sends the scheduler one event at a time. It is not the most realistic implementation as the system should also handle multiple passengers who will be arriving at the elevator at the same time but going to different destinations. By combining the events into one event and then breaking it down after, the system could be more realistic but to do so the input file should be reformatted to allow more than one destination at a time. Finally, the GUI could have had a little better representation of the elevator, although the GUI does show the system thoroughly, it would have been nice to have a table that would pinpoint the location of the elevator.