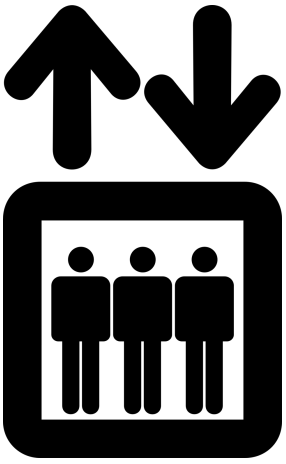
04/10/2024

SYSC 3303 Real Time Concurrent Systems

Final Report

Group 1



Alex Marques	alexandremarques@cmail.carleton.ca	101189743
Arthur Atangana	arthuratangana@cmail.carleton.ca	101005197
Braeden Kloke	braedenkloke@cmail.carleton.ca	100895984
Michael De Santis	michaeldesantis@cmail.carleton.ca	101213450
Victoria Malouf	victoriamalouf@cmail.carleton.ca	101179986

Table of Contents

Overview

As per the project specification, this final report contains the following:

- Team number and team members
- Table of contents
- Breakdown of responsibilities of each team member for each iteration
- All diagrams:
    - UML class diagrams for the three components
    - A State Machine diagram for the scheduler
    - Sequence diagrams showing all the error scenarios
    - Timing diagrams for the scheduler
- Detailed set up and test instructions
- Results from your measurements
- Reflection on your design - what parts do you like and what parts should be redone.

Breakdown of Responsibilities for Each Iteration

Iteration 0 Tasks

Task	Assignee
Report Formatting	Victoria Malouf
Data Analysis	Victoria Malouf
Data Collection	Arthur Atangana
Data Analysis	Arthur Atangana
Data Collection	Michael De Santis
Data Analysis	Michael De Santis
Data Collection	Alexandre Marques
Data Analysis	Alexandre Marques
Data Analysis	Braeden Kloke

Iteration 1 Tasks

Task	Assignee
Project Structure	Victoria Malouf

Floor Model	Victoria Malouf
Elevator Model	Arthur Atangana
Scheduler Model	Arthur Atangana
Parser	Michael De Santis
Dispatcher	Alexandre Marques
Event Handling	Alexandre Marques
Diagrams	Braeden Kloke
Testing	Braeden Kloke

Iteration 2 Tasks

Task	Assignee
System Configuration	Michael De Santis
Class Diagrams	Michael De Santis
Sequence Diagrams	Michael De Santis
State Diagrams	Braeden Kloke
Scheduler Logic	Braeden Kloke
Networking Diagrams	Victoria Malouf
Elevator Utility Test	Victoria Malouf
Floor Logic	Victoria Malouf
State Diagrams	Alexandre Marques
Command/Event Classes	Alexandre Marques
Elevator Logic	Alexandre Marques
Scheduler Logic	Alexandre Marques
Loader Class	Alexandre Marques
Elevator Logic	Arthur Atangana
Scheduler Logic	Arthur Atangana
System Test	Arthur Atangana

Iteration 3 Tasks

Task	Assignee
Scheduler State Machine	Michael De Santis
System Logging	Michael De Santis
System Configuration	Michael De Santis
Scenario Tests	Michael De Santis
Elevator Logic	Michael De Santis
Elevator State Machine	Braeden Kloke
Diagrams	Braeden Kloke
Elevator Logic	Braeden Kloke
Diagrams	Victoria Malouf
Unit Tests	Victoria Malouf
UDP Implementation	Victoria Malouf
System integration	Victoria Malouf
UDP Implementation	Alexandre Marques
Subsystem Separation	Alexandre Marques
Factory Implementations	Alexandre Marques
System Integration	Alexandre Marques
Transceivers	Alexandre Marques
Scheduler State Machine	Arthur Atangana
Elevator Logic	Arthur Atangana
Scenario Tests	Arthur Atangana

Iteration 4 Tasks

Task	Assignee
Parser Fault Handling	Michael De Santis
Scenario Tests	Michael De Santis

Documentation	Michael De Santis
Timing Diagrams	Braeden Kloke
Timer in Scheduler	Braeden Kloke
Elevator Fault Handling	Victoria Malouf
Scenario Tests	Victoria Malouf
Documentation	Victoria Malouf
Elevator Bound Handling	Victoria Malouf
Fault Message Passing	Alexandre Marques
Fault Troubleshooting	Alexandre Marques
System Integration	Alexandre Marques
Elevator Fault Handling	Arthur Atangana
Scenario Tests	Arthur Atangana
Parser Validation	Arthur Atangana
Elevator Bound Handling	Arthur Atangana

Iteration 5 Tasks

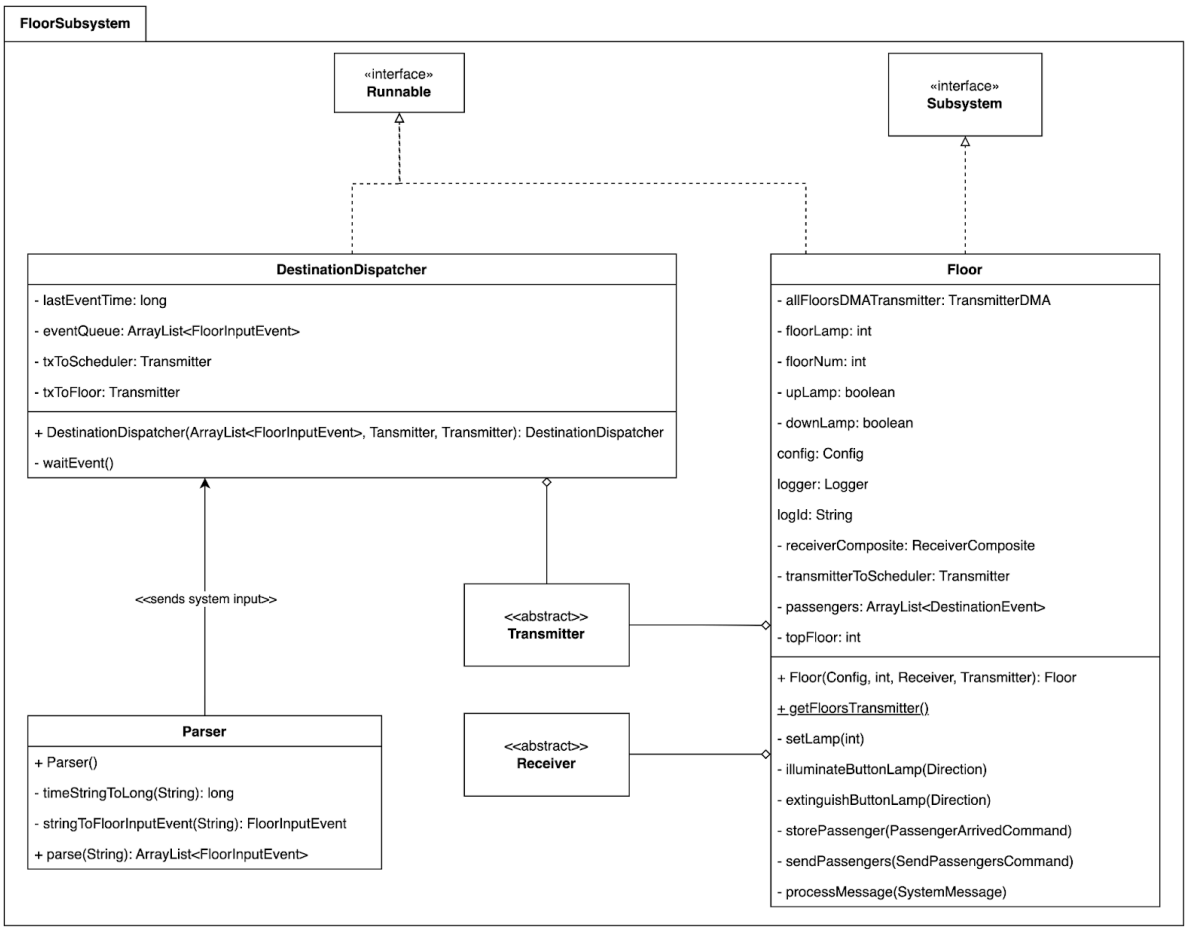
Task	Assignee
Display Console TUI	Michael De Santis
Display Console GUI	Michael De Santis
Scenario Tests	Michael De Santis
Scheduler Statistics	Michael De Santis
Scheduler Statistics	Braeden Kloke
Scheduler Terminal State	Braeden Kloke
Scenario Tests	Braeden Kloke
Diagrams	Victoria Malouf
Passenger Capacity Handling	Victoria Malouf
Unit Tests	Victoria Malouf

Passenger Capacity Handling	Alexandre Marques
Scenario Tests	Alexandre Marques
Passenger Capacity Handling	Arthur Atangana
Scenario Tests	Arthur Atangana
Display Console GUI	Arthur Atangana

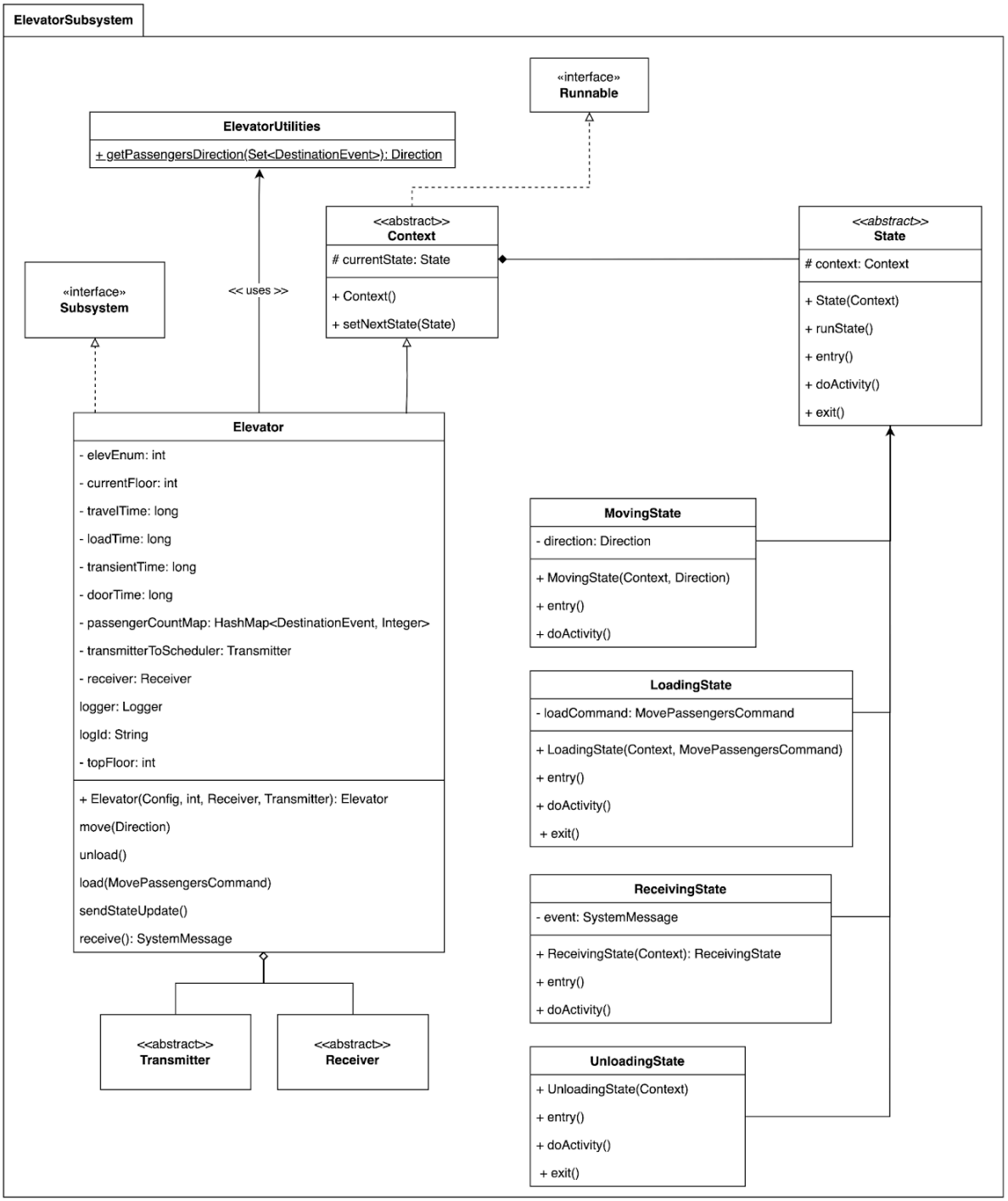
Diagrams

UML class diagrams for the three components

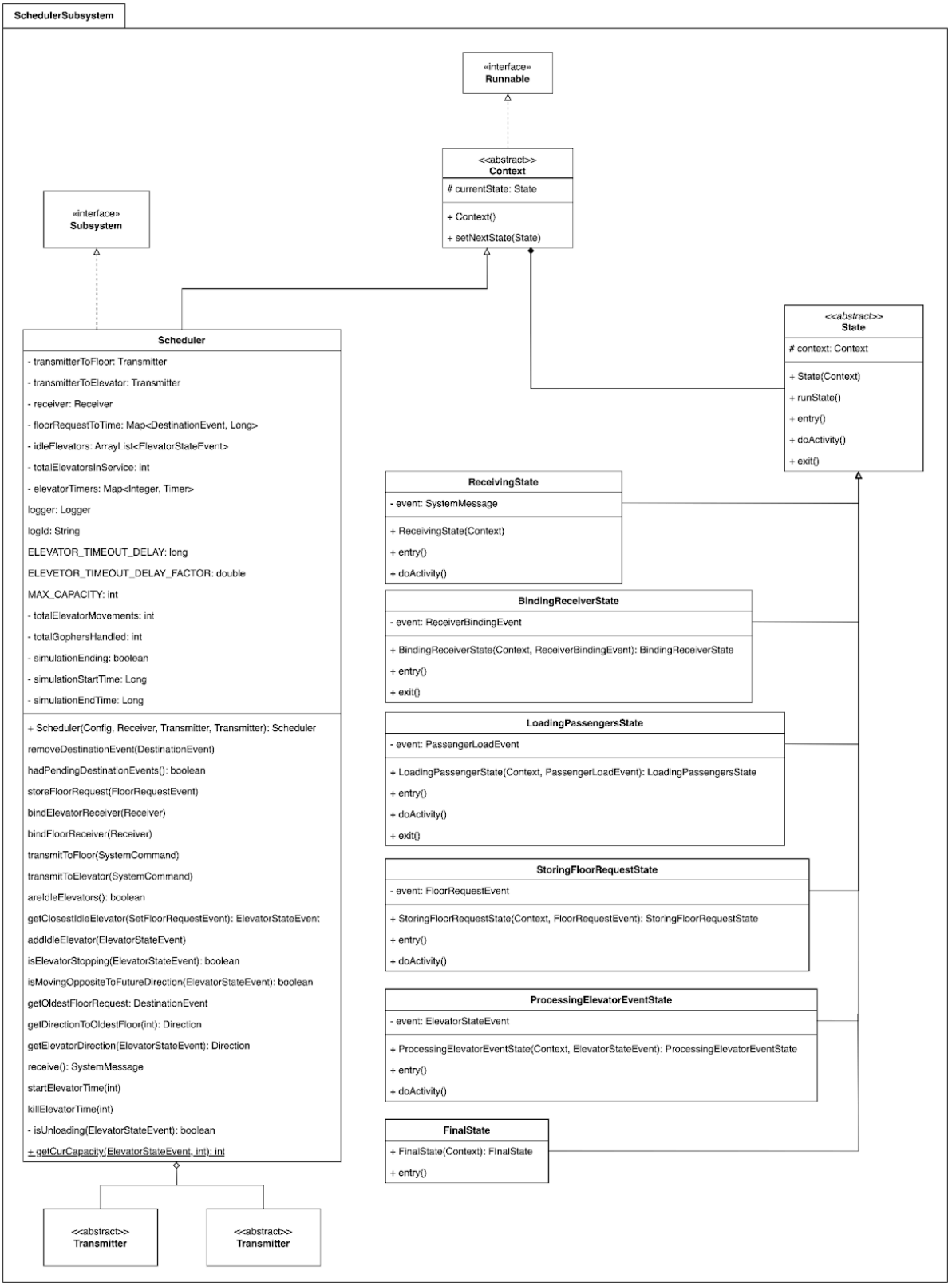
Floor Subsystem



Elevator Subsystem

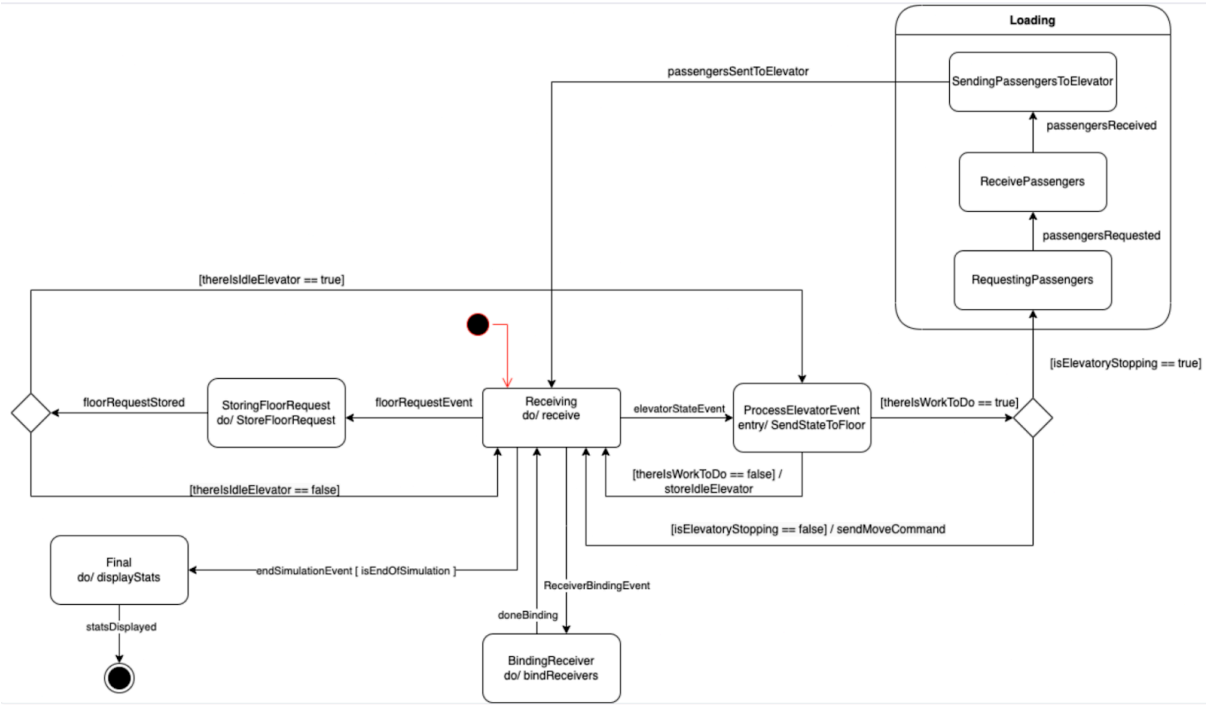


Scheduler Subsystem



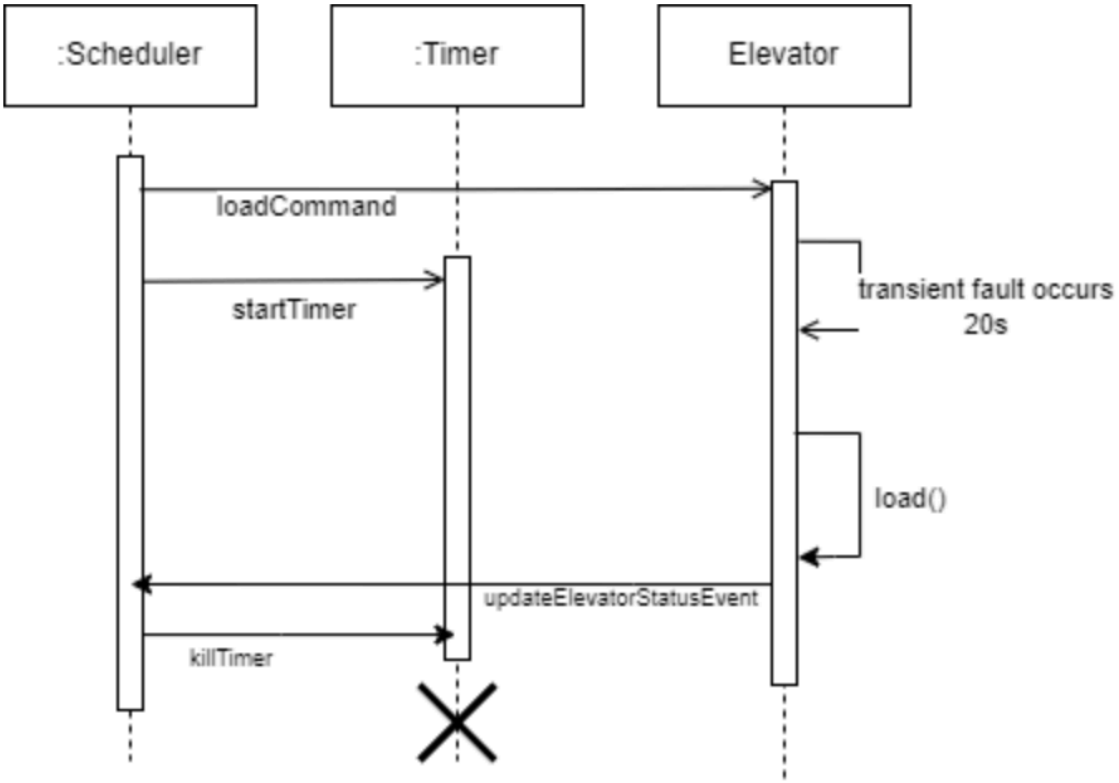
State Machine diagram for the scheduler

Scheduler Subsystem

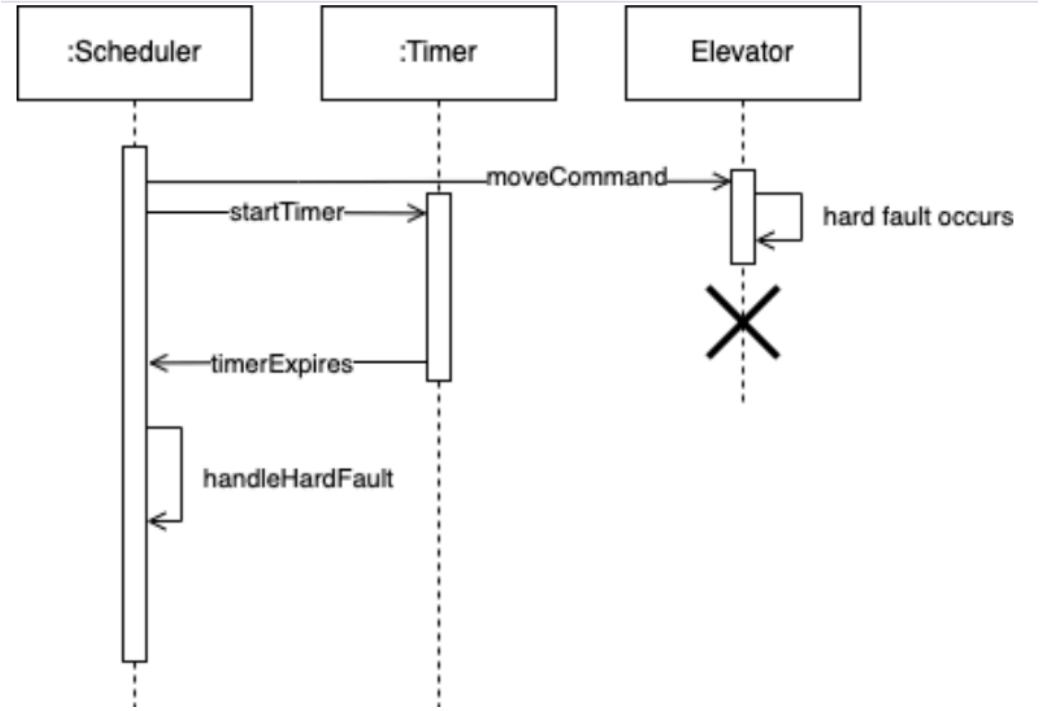


Sequence diagrams showing all the error scenarios

Transient Fault

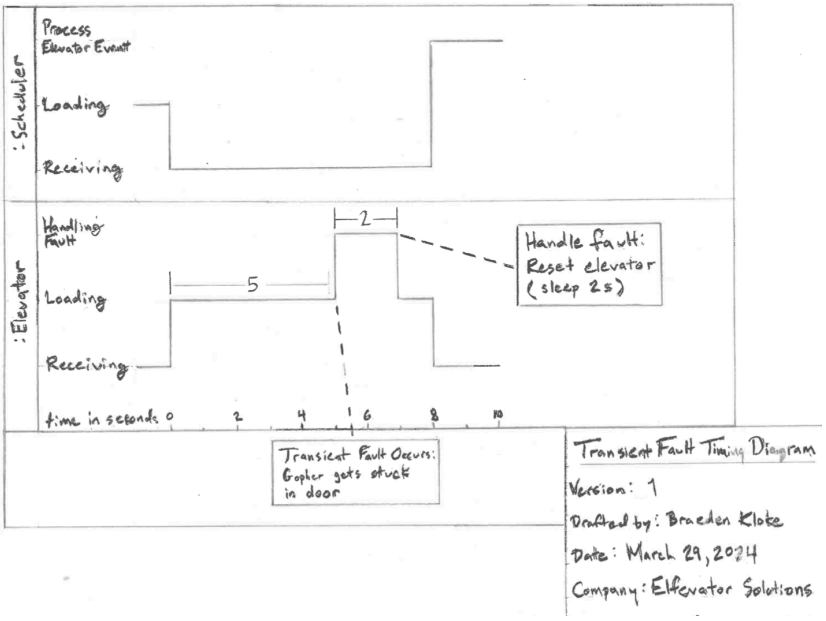


Hard Fault

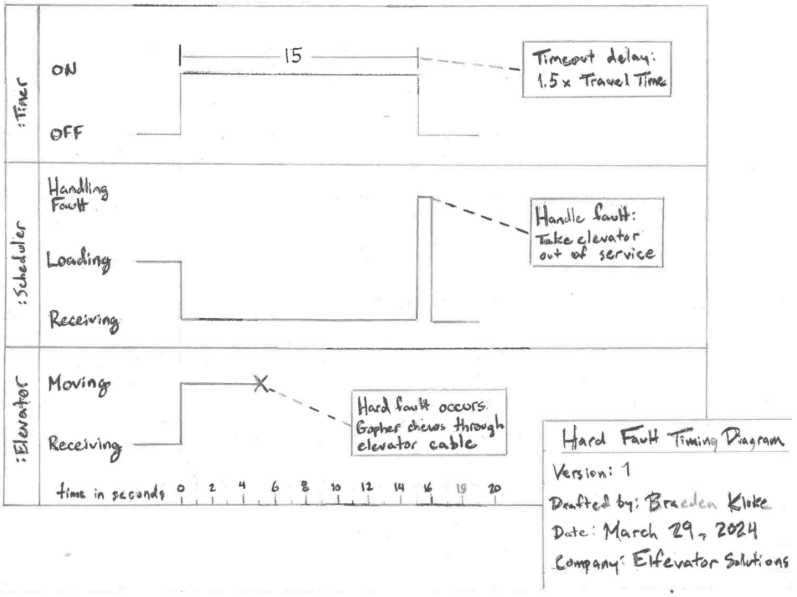


Timing diagrams for the scheduler

Transient Fault



Hard Fault



Detailed set up and test instructions

Set Up Instructions

Usage written for use in IntelliJ 2023 IDE.
Note: Compiled using JDK - 21 Oracle OpenJDK version 21.0.1

1. Retrieve source code (https://github.com/ArthurAtangana/SYSC3303A_Project) in IntelliJ
   - Clone using HTTPS:
     - https://github.com/ArthurAtangana/SYSC3303A_Project.git
   - Clone using SSH:
     - git clone git@github.com:ArthurAtangana/SYSC3303A_Project.git
   - Clone using GitHub CLI:
     - gh repo clone ArthurAtangana/SYSC3303A_Project

2. Add lib folder as a library
   a) Right click lib folder in root directory
   b) Select add as library
   c) Leave settings as default, select ok

3. Run the files in the "Mains" package in this order:
   a) MainDisplayConsole
   b) MainSchedulerUDP
   c) MainElevatorUDP
   d) MainFloorUDP

Test Instructions

All tests are located in directory test.
Tests are categorized in subdirectories labeled:
- unit: Tests isolated components.
- system: Tests all components working together. (eg. Does the simulation correctly get all passengers to their desired floors?)

All resources for tests (eg. input files) are in test/resources.
Only classes that expose public methods are unit tested.
The testing framework used for unit tests is JUnit 5.8.1.

Scenario-based System Testing

To test the system, a variety of scenario tests have been written by modifying a standardized system input file, which is parsed by the FloorSubsystem to create input events at runtime. These scenarios each define specific conditions under which the system will be tested, and are used to test the overall functionality of the system, as well as aid in debugging efforts. Scenario files may be found in the test/resources directory, and each test file includes a doc header and commentary that describes the scenario and its operation. Scenario testing from these files is easily configurable from the system configuration JSON, located in the project res/ directory, by specifying the scenario file as the value of the "inputFilename" property.

For example:
"inputFilename": "test/resources/scenario-6-combined-fault-test.txt"

Fault Injection and Handling

The system is equipped to handle both HARD and TRANSIENT faults, where:

TRANSIENT: faults represent a minor fault that affects the quality of service of the system, such as a passenger blocking an elevator door, that may be gracefully recovered from by the system to permit full and continued operation after recovery.

HARD: faults represent a significant fault, such as mechanical failure of an elevator, that requires the elevator to be safely removed from service, leaving the overall system functional but with a reduced number of elevators.

Faults are simulated, as per project specifications, by attaching a fault code to modeled passengers. Faults may be attached to passengers and injected into the system by supplying a scenario input file with fault codes according to the following scheme:

- 0: indicates NO fault; the system will operate as normal.
- 1: indicates a TRANSIENT fault; the system will gracefully handle the fault, and subsequently resume normal operation with all elevators still functional.
- 2: indicates a HARD fault; the system will handle the fault by removing the faulty elevator from service, and subsequently resume normal operation less the removed elevator.

Troubleshooting

IntelliJ can't find your tests?
Ensure you have directory test labelled as your Test Source Folder in Project Structure > Modules.

Test classes don't compile due to missing imports (red squiggles)?
Alt-enter on the invalid imports, and select add JUnit 5.8.1 to classpath, select OK.

Missing libraries?
To add the included libraries in the project's lib/ directory for build, use the following procedure.

1. In IntelliJ, open the Project Structure window (File > Project Structure), and navigate to the Libraries section (Project Settings > Libraries).
2. Click the + icon to add a new library, and select Java from the dropdown menu.
3. In the Select Library Files browser that opens, select this project's lib/ directory and click OK.
4. In the Choose Modules dialog box that opens, ensure that this project's module, SYSC3303A_Project, is selected (this should be the default), and click OK.
5. In the Project Structure window, confirm the changes by clicking OK.

All JAR files in the lib/ directory should now be included in the project for import and build.

Results from your measurements

This section is divided into three subsections. The first subsection contains the simulation results of the demo input file. The second subsection describes what the simulation statistics are and how they are measured. The last subsection describes the data collection and analysis of a real elevator.

Simulation Results

The results from the demo input file are as follows:

- **Total simulation time:** 13 minutes 44 seconds
- **Total elevator movements:** 221
- **Total hard faults handled:** 0

The following image is the display console output after our system runs the demo input file.

```
[1712670114229::INFO::SCHEDULER] Displaying simulation statistics ...
[1712670114243::INFO::SCHEDULER] Total simulation time (HH:MM:SS): 00:13:44
[1712670114244::INFO::SCHEDULER] Total elevator movements: 221
[1712670114244::INFO::SCHEDULER] Total gophers handled (hard faults): 0
```

Simulation Statistics

The following three subsections describe each statistic tracked in this simulator. The final subsection shows the simulation results for the demo input file. All statistics are displayed to console at the end of the simulation.

Total Simulation Time

The simulation starts when the first passenger request is received by the Scheduler.
The simulation ends when all the in-service elevators are idle and there are no more passenger requests. Total simulation time is calculated by subtracting the simulation start time from the simulation end time.

Elevator Movements

Total elevator movements are assumed to mean the total number of floors traveled by all elevators. Each MoveElevatorCommand sent by the Scheduler corresponds to an elevator traveling one floor. Thus, the system tracks the total floors traveled by all elevators by tracking the total number of MoveElevatorCommands sent by the Scheduler.

Hard Faults Handled

Hard faults are simulated with gophers chewing through the elevator cables,

causing the elevator cab to plummet through the Earth's crust and disrupt subterranean ecosystems. No rescue mission is mounted because passengers are assumed to be engineering students. Since gophers are hard faults, the system tracks total gophers handled by tracking total hard faults handled.

Data Collection and Analysis of a Real Elevator

Objective

The objective of Iteration 0 is to collect and analyze data using Carleton's Canal Building elevators to implement functions that return real-world representations of elevator boarding and travel times. This report will state the assumptions and strategies used to estimate the rate of acceleration and deceleration of an elevator car; the maximum speed of the car; and the typical loading and unloading times as a function of the number of passengers.

Data Collection

Principles and procedures used in the collection of empirical data for this design iteration.

Collection Procedures

During data collection, the following actions were taken to ensure consistency and accuracy in measured results:

- All data collection was performed using the same elevator: Canal Building's "ELEV - 1" elevator
- Three team members were present during the collection of empirical data to ensure that all necessary factors were adequately tracked, with one team member each responsible for ensuring consistent time, passenger, and distance measurements
- All time domain measurements were recorded using the same device: a team member's stopwatch on their smartphone with a resolution of 10ms
- All time domain measurements were executed by the same team member according to the same procedure
- Elevator functions were invoked in a consistent way, with no additional or discrepant functionalities introduced by pressing unnecessary buttons

All collected empirical data was recorded in the appended spreadsheet, and only this data was used in subsequent data analysis.

Boarding Data

Boarding and deboarding data were quantified by the ingress or egress of each passenger across the threshold of the elevator. In total, 24 samples were recorded with the number of passengers ranging from one to six. The team chose this range based on elevator capacity. Four samples were collected for each number of passengers ranging from one to six. An assumption was made that passenger directionality was not a contributing factor to the total time taken for the boarding data, as both boarding and deboarding procedures were observed to be similar. The "Open Doors" and "Close Doors" buttons were not pressed during these trials, allowing all time intervals to be managed solely by the elevator scheduler without considering additional user input that might have hastened or delayed boarding times. Boarding intervals were initiated upon the separation of the two elevator doors during the

opening procedure, and terminated by the contact between the two elevator doors during the closing procedure.

Travel Data

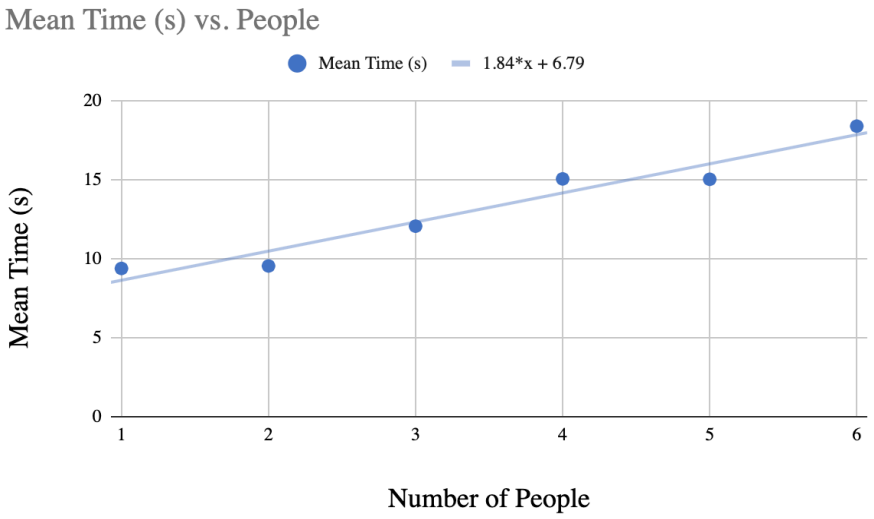
Travel data was collected taking into account the number of floors traveled. In total, 48 samples were recorded, with the number of floors traveling from one to six. The team chose this range based on the fact that the Canal Building has seven floors accessible by elevator, allowing a maximum traversal of six consecutive floors in a single, unidirectional trip. Eight samples were collected for each number of floors traversed, with four samples in the ascending direction and four in the descending direction. Floor requests for travel intervals were made during the preceding boarding intervals so that the elevator would begin servicing the request immediately upon entering the travel interval; this was done to ensure that no additional and variable delay was introduced by beginning a travel interval without a destination floor selected. Travel intervals were initiated upon contact between the two elevator doors during the closing procedure, and terminated by the initial separation of the two elevator doors during the opening procedure.

Data Analysis

Data analysis was performed on the empirical data collected according to the procedures described above.

Linear Regression Analysis of Boarding Data

Graph 1 below shows the plotted data of boarding time as a function of the number of passengers.



Graph 1: Linear Regression Analysis of Boarding Data

Acceleration, Deceleration, and Maximum Speed Calculations

To calculate acceleration, deceleration, and maximum speed, the following assumptions were considered:

- Acceleration is constant
- Acceleration equals deceleration
- Elevator spends 50% of travel time accelerating and 50% travel time decelerating.
- Floors are 4 meters apart.

Using the collected data for travel time and the above assumptions, we can derive acceleration and calculate the mean acceleration to be 0.250 m/s^2. An example derivation of acceleration can be found in Appendix A.

Using the above assumptions, peak speed is reached halfway through each elevator trip. We can then calculate the peak speed for each sample using the derived velocity function for that sample. From this set of peak speeds, we select the maximum. The maximum peak speed is our best estimation of the maximum speed of the elevator car.

A summary of the above calculations are as follows:

- Acceleration / Deceleration: 0.250 m/s^2
- Maximum Speed: 2.469 m/s

Results

The final results arrived at through the data collection and data analysis procedures described above, including relevant values and possible sources of error.

Summary of Values

The values arrived at in fulfillment of this iteration's objectives are as follows:

1. Average Magnitude of Acceleration/Deceleration:
   0.250 m/s2
2. Maximum Speed:
   2.469 m/s
3. Boarding/Deboarding Time (as a function of people, *n*):
   boardingTime = 1.84n + 6.79

Statistical Confidence in Derived Values

To determine a level of confidence in our results, we calculated confidence intervals on our collected data. The confidence interval is used to estimate the range in which a specified parameter will fall. For our calculations, we used 5% significance to determine upper and lower limits. The formula for our confidence interval calculations is:

$$CI = \overline{x} \pm z\frac{s}{\sqrt{n}}$$

Where:

- $\overline{x}$ is the sample mean
- z is the confidence level value
- s is the standard deviation
- n is the sample size

Sample calculation of confidence interval for the time it takes 2 people to board/deboard an elevator:

- $\overline{x}$ = 9.533 s
- z = 1.96
- s = 0.601 s
- n = 4

$$CI = 9.533s \pm 1.96\frac{0.601s}{\sqrt{4}}$$

$$CI = 9.533 \pm 0.589$$

The confidence interval of boarding time in seconds of 2 passengers with 95% confidence is 8.94 < x < 10.12 seconds.

Error

Various sources of error may have been introduced in the collection of empirical data. Discrete measurements, such as floors traveled and people boarding and deboarding, were simple to observe and record with confidence. However, time measurements relied on a human agent to visually observe start and stop criteria for each interval (ie. the opening and closing of elevator doors), and then manually start and stop the stopwatch. This procedure is imprecise and introduces errors in the time required to observe and manually start and stop recording. Additionally, trials did not factor in a variety of typical and atypical elevator behaviors that may increase or decrease boarding times, such as passengers with mobility issues; passengers using the "Open Doors" or "Close Doors" buttons; passengers carrying freight that may slow down boarding; passengers manually holding doors open to wait for other passengers; and a variety of other predictable and unpredictable behaviors. Travel measurements also did not factor in the number of people in the elevator, nor their cumulative weight, which may have some implications for acceleration and deceleration rates. Another source of error is in the assumption of exactly 4m between floors. No distance measurements were taken to verify this distance or provide a more accurate measurement; however, this assumption was prescribed by the design requirements, and distance measurements were deliberately not gathered in favor of the prescribed value.

Conclusions

The procedures for empirical data collection and analysis were adequate to meet the objectives of Iteration 0. All necessary data values were obtained empirically, and all required analysis was performed to confidently arrive at representative derived values. Although there are sources of error contributing to the derived values, this error is acknowledged and documented and will be considered in future project iterations.

Conclusion and Design Reflections

Reflection on your design - what parts do you like and what parts should be redone

Through the iterative design process, component and system designs changed substantially. Design goals for each iteration were informed by iteration requirements, as well as course content. With each iteration, regression tests were performed to determine the continued functionality of each system component, and in many instances these tests also exposed system vulnerabilities and inefficiencies. As our progression through the course content also exposed new ideas, technologies, and design patterns, we spent much time in improving upon existing system components in order to incorporate our increased knowledge.

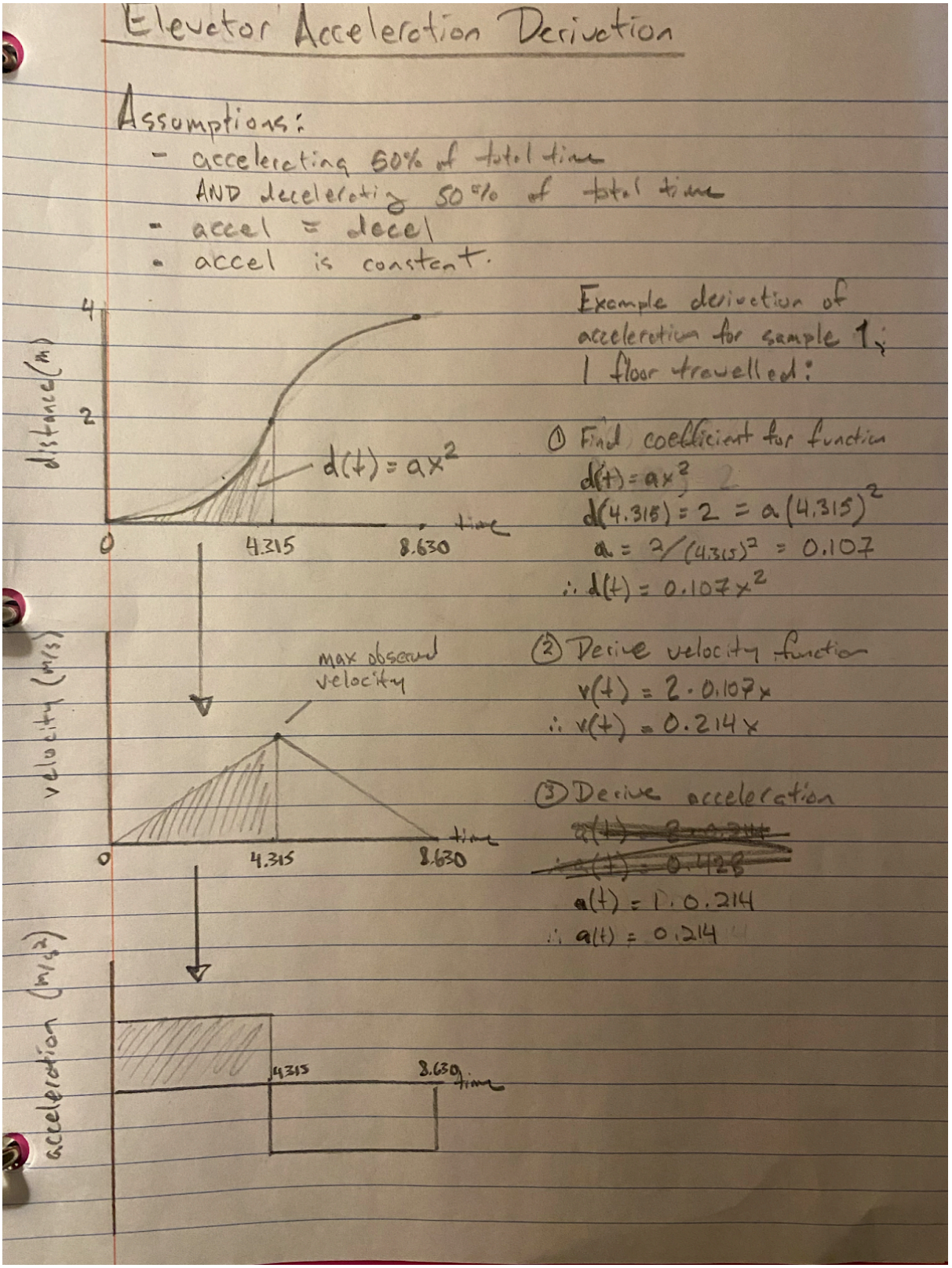
One part of our system that we consider exemplary of a good design is our system's configurability. By providing mechanisms to easily configure and reconfigure our system as needed, this not only helped ease our progression through evolving iteration requirements, but also in quickly troubleshooting a variety of scenarios. Most of our system parameters (debug verbosity, load times, travel times, passenger capacity, etc.) were easily configurable in a single JSON file that was ingested at load time. This feature allowed us to quickly and easily configure the system, and propagate that configuration throughout subsystems without the need for any manual code changes.

Another strength of our system is its configurable monitoring. A centralized logging system provides each subsystem with a means of printing its own logs, as well as aggregating logs to a central display console. Every logged event contains a subsystem timestamp (with millisecond resolution), a subsystem and thread identifier (eg. elevator number, floor number), as well as detailed event descriptions. Additionally, we implemented configurable verbosity, allowing us to suppress all output; print only important system events (INFO level events); or print both major and minor system events for debugging purposes (INFO and DEBUG level system events). In addition to this text-based logging via the console, we also implemented a Graphical User Interface (GUI) that clearly and visually shows the position and state of all elevators. This GUI refreshes in real-time, and allows a precise and clear visualization of all aspects of the system, including system events, states, and faults.

One aspect of our design that we could have improved upon is the readability of our codebase. While our system functions as expected and meets requirements, we did encounter difficulties in managing the complexity of the system across all developers trying to simultaneously work on the codebase. If we were given the opportunity to improve upon this aspect of our codebase, we may consider reducing system complexity, providing additional documentation, and reducing coupling between system components.

Overall, this project provided a stimulating and informative interaction with real-time systems, and allowed us to realize our knowledge through design, implementation, and testing.

Appendix A: Example Acceleration Derivation



Appendix B: Team Photo