ECE 18-649
Midterm Project Report

November 1, 2013

Group # 18
Yi Huo
Mustafa Bilgen
Sairam Krishnan
Abhijit Lele

Overview

- Content
  - Project statistics
  - Complete design of Dispatcher
  - Lessons learned

Project Statistics

- 18 sequence diagrams and 146 arcs
- 58 lines of requirements
- 7 state charts, 24 states, 31 arcs
- 1302 lines of non-comment code
- 54 test files
- 52 peer reviews, 40 defects found, 40 defects fixed
- 11 revisions

Scenario

Scenario 9A: Dispatcher computes next desired floor just as doors open.

Pre-Conditions:

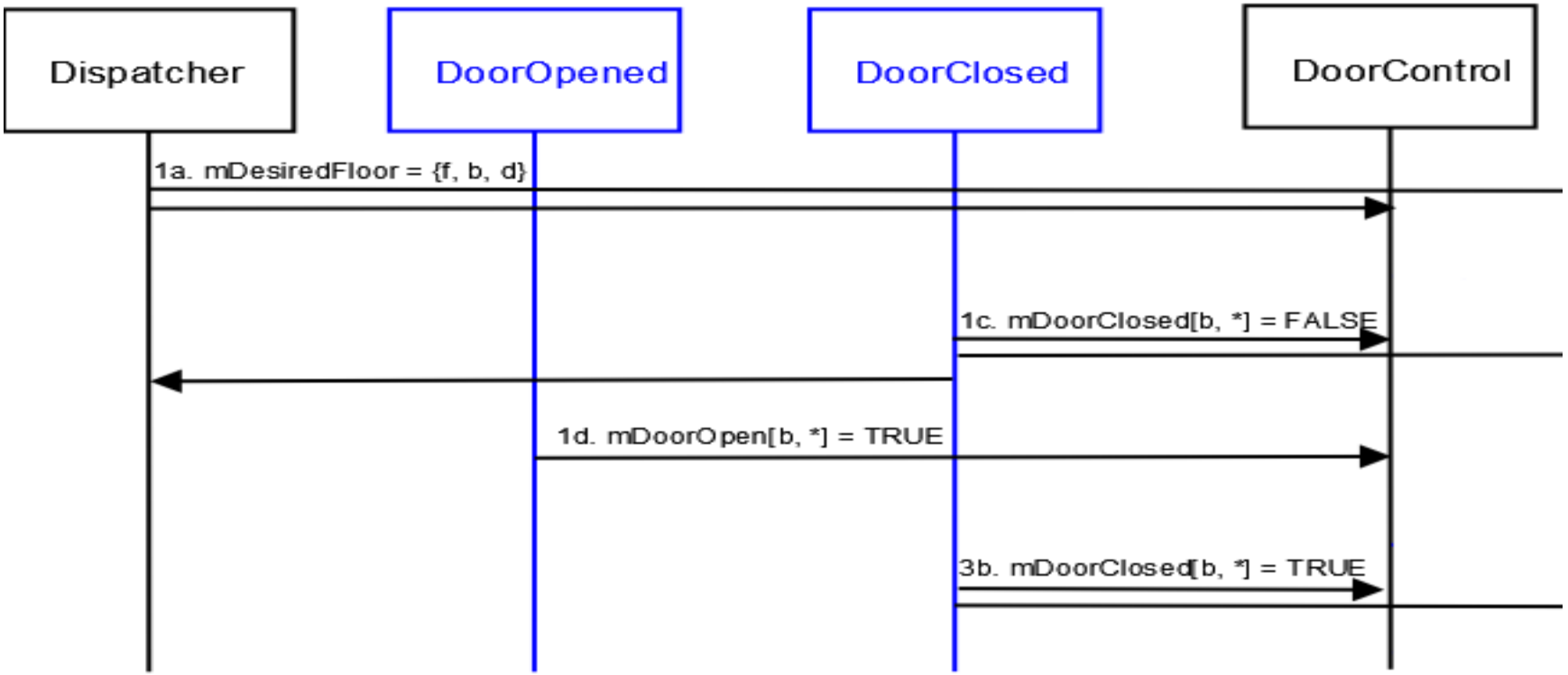
- Car is stopped.
- AtFloor[f,b] was last received as True.
- All doors are closed.

Scenario:

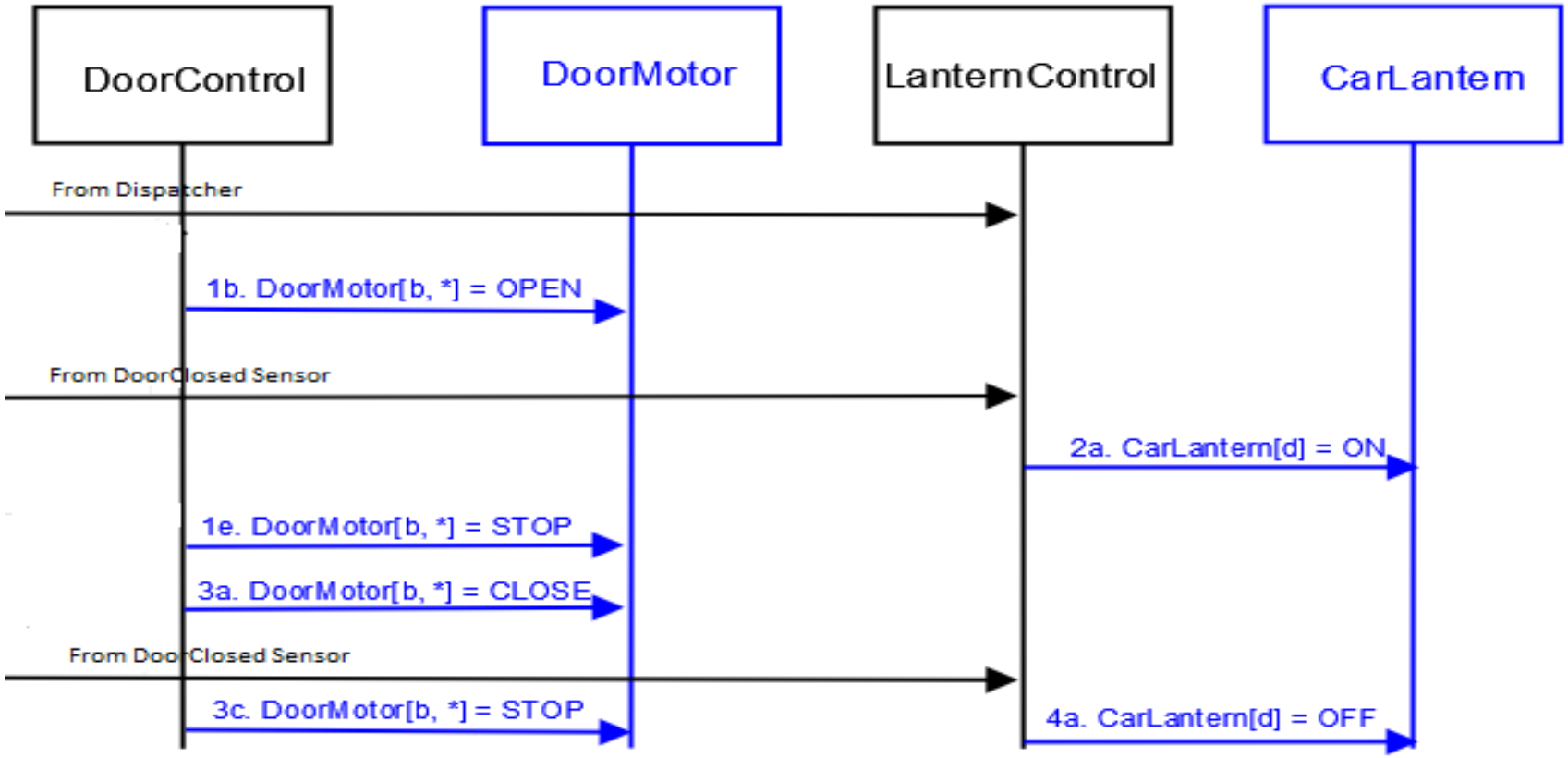
- Doors[**b**,**r**] open.
- CarLantern[**d**] is turned on.
- Doors[**b**,**r**] close.
- CarLantern[**d**] is turned off.

Sequence Diagram

Sequence Diagram 9A:



Sequence Diagram



Dispatcher Requirements

- We have a dispatcher that goes up and down, instead of just up.

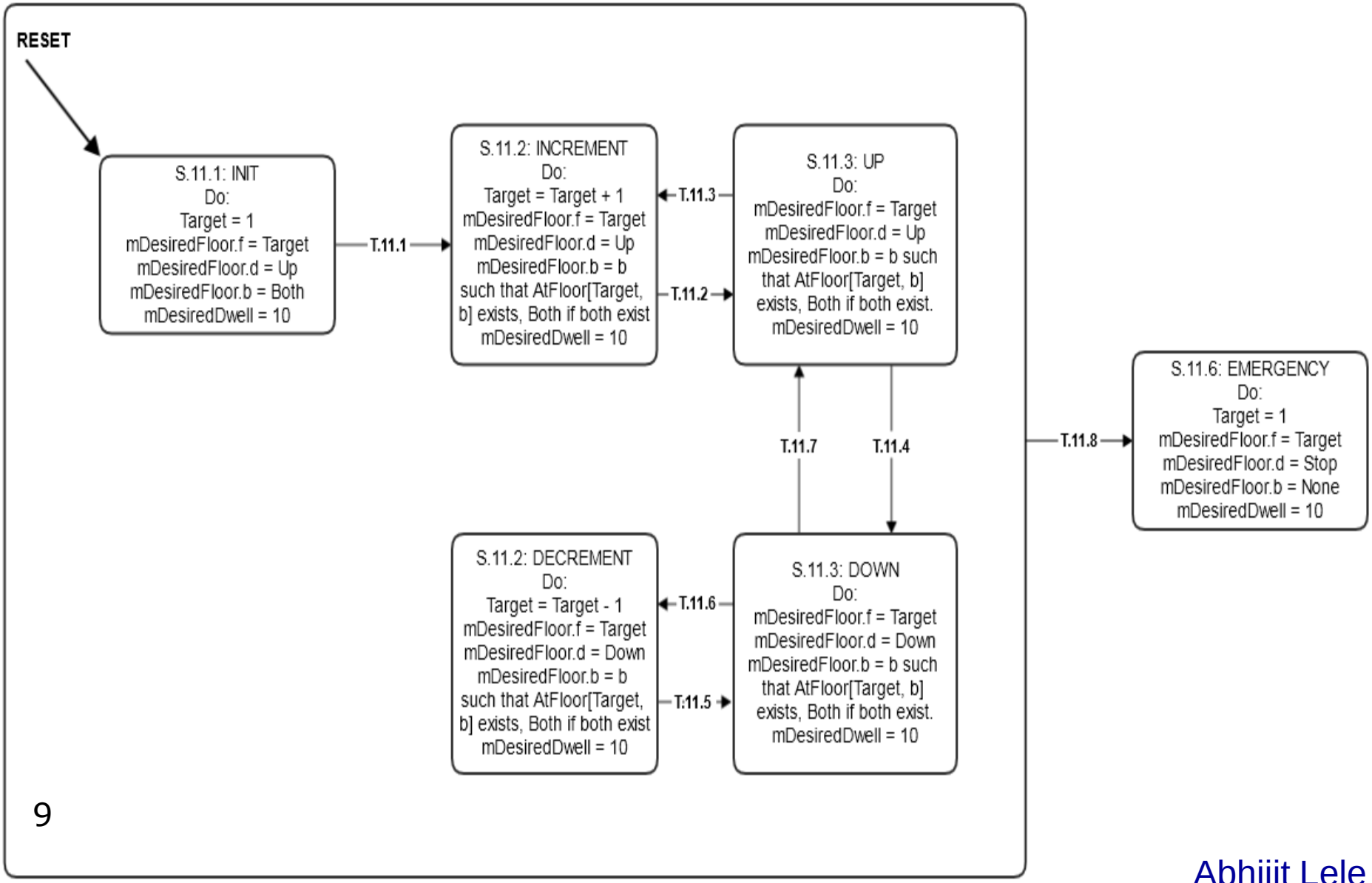
- 11.4 mDesiredFloor.f shall always be set to Target.
- 11.5 If Target == 8 and all mDoorClosed[*, *] are TRUE, mDesiredFloor.d shall be set to DOWN.
- 11.6 If Target == 1 and all mDoorClosed[*, *] are TRUE, mDesiredFloor.d shall be set to UP.
- 11.7 Whenever any mDoorClosed [b, r] is False and at least one mAtFloor[*, *] is True, then
  - 11.7.1 If mDesiredFloor.d is UP, Target shall be set equal to (f + 1), where f whichever mAtFloor[f, *] is True
  - 11.7.2 If mDesiredFloor.d is DOWN, Target shall be set equal to (f - 1), where f is whichever mAtFloor[f, *] is True
  - 11.7.3 mDesiredFloor.b shall be set to b, where b is whichever AtFloor[Target, b] exists.

Dispatcher Requirements

- We have a dispatcher that goes up and down, instead of just up.

- 11.8 If all mAtFloor[f, b] are False AND any mDoorClosed [b, r] is False (which means doors are not closed between floors!); then
  - 11.8.1 Target shall be set to 1
  - 11.8.2 mDesiredFloor.b shall be set to None
- 11.9 If two AtFloor[Target, b] sensors exist, then mDesiredFloor.b shall be set to Both.
- 11.10 mDesiredDwell shall always be set to a constant appropriate value for door open dwell.

Dispatcher State Chart



Dispatcher Transition Table

Transition #	Guard Condition
T.11.1	mDoorClosed[*,*] == FALSE && mAtFloor[1,*] == TRUE
T.11.2	Always
T.11.3	mDoorClosed[*,*] == FALSE && mAtFloor[Target,*] == TRUE
T.11.4	Target == MAX_FLOOR && all mDoorClosed[*,*] == TRUE
T.11.5	Always
T.11.6	mDoorClosed[*,*] == FALSE && mAtFloor[Target,*] == TRUE
T.11.7	Target == 1 && all mDoorClosed[*,*] == TRUE
T.11.8	mDoorClosed[*,*] == FALSE && all mAtFloor[*,*] == FALSE

Dispatcher Code

```
case STATE_INIT:
        //State actions for INIT
        target = 1;
        mDesiredFloor.set(target, Direction.UP, Hallway.BOTH);
        mDesiredDwell[Hallway.FRONT.ordinal()].set(DEFAULT_DWELL);
        mDesiredDwell[Hallway.BACK.ordinal()].set(DEFAULT_DWELL);

        //Transitions
        if(T.11.1) {
          newState = State.STATE_INCREMENT;
        }


case STATE_INCREMENT:
        //State actions for INCREMENT
       if (Elevator.hasLand ing())
       {
         Assign h to landing sides : FRONT/ BACK / BOTH
        }
        mDesiredFloor.set(target, Direction.UP, h);
        mDesiredDwell[Hallway.FRONT.ordinal()].set(DEFAULT_DWELL);
        mDesiredDwell[Hallway.BACK.ordinal()].set(DEFAULT_DWELL);

        //#transition 'T.11.2'
        newState = State.STATE_UP;
```

Dispatcher Code

```
case STATE_UP:
            if (Elevator.hasLand ing())
            {
              Assign h to landing sides : FRONT/ BACK / BOTH
             }
             mDesiredDwell[Hallway.FRONT.ordinal()].set(DEFAULT_DWELL);
             mDesiredDwell[Hallway.BACK.ordinal()].set(DEFAULT_DWELL);
            if (T.11.3) {
                newState = State.STATE_INCREMENT;
             }
             else if (T.11.4){
               newState = State.STATE_DOWN;
             }
case STATE_DECREMENT:

           //State actions for DECREMENT
           target--;
           if (Elevator.hasLand ing())
          {
            Assign h to landing sides : FRONT/ BACK / BOTH;
           }
           mDesiredFloor.set(target, Direction.DOWN, h);
           mDesiredDwell[Hallway.FRONT.ordinal()].set(DEFAULT_DWELL);
           mDesiredDwell[Hallway.BACK.ordinal()].set(DEFAULT_DWELL);

           //#transition 'T.11.5'
           newState = State.STATE_DOWN;
```

Dispatcher Code

```
case STATE_DOWN:
        if (Elevator.hasLand ing())
        {
          Assign h to landing sides : FRONT/ BACK / BOTH
        }
        mDesiredFloor.set(target, Direction.DOWN, h);
        mDesiredDwell[Hallway.FRONT.ordinal()].set(DEFAULT_DWELL);
        mDesiredDwell[Hallway.BACK.ordinal()].set(DEFAULT_DWELL);
        if (T.11.6){
            newState = State.STATE_DECREMENT;
        }
        else if (T.11.7){
            newState = State.STATE_UP;
        }

case STATE_EMERGENCY:
        target = 1;
        mDesiredFloor.set(target, Direction.STOP, Hallway.NONE);
        mDesiredDwell[Hallway.FRONT.ordinal()].set(DEFAULT_DWELL);
        mDesiredDwell[Hallway.BACK.ordinal()].set(DEFAULT_DWELL);

        newState = state;
        break;
```

Dispatcher Testing

- Single unit test:
- Simulates one cycle of going up and down
- 7.73 simulation seconds
- 129 assertions, all passed

- Three integration tests:
- Simulates arriving at floor, emergency condition, and setting mDesiredDwell
- 31 total assertions, all passed

Lessons Learned

- Consistency facilitates communication of ideas.
  - Use same software for sequence diagrams and state charts.
  - Use a single naming/style convention.
- Version control is your friend.
  - Great way of keeping track of changes to your project and reverting to previous versions in emergencies.
- Coordinate interactions with GitHub, especially at the last minute!
  - Can accidentally merge or override changes.

Lessons Learned

- Most of your time should be spent on design and architecture rather than on implementation and testing.
  - If you have a solid design, then you should only have minor errors in implementation.
- Distributed embedded systems based on the concept of specialization of labor
  - Dispatcher does not have to do everything.
- Leveling
  - Can’t directly transition the Drive from Slow to Level; must stop the car first.

Lessons Learned

- What Worked Well
  - Scenarios & Use Cases
    - Helped us visualize different elevator behaviors
  - State Charts
    - Great guide to implementation
  - Unit Tests and Integration Tests
    - Great way to test state charts and sequence diagrams
  - Acceptance Tests
- What Didn’t Work Well
  - Unit and Integration Tests
    - More errors due to timing issues within the tests themselves rather than with the implementation.