ECE 18-649 Mid-Term Project Presentation

Oct 21, 2015

Group # 12

Tom Eliot
Shepard Emerson
Daniel Gorziglia
Daniel Haddox

All

Outline

- Project Statistics
- Controller Design, Dispatcher
  - Scenarios
  - Sequence Diagrams
  - Requirements
  - Statechart
  - Code
  - Testing
- Lessons Learned
- Open Issues

Project Statistics

- 18 Scenarios, 18 Sequence Diagrams
- 37 lines of requirements (controllers only)
- 7 Statecharts (controllers only)
  - 21 States
  - 26 Arcs
- 1870 lines of non-comment code
- 33 test files
- 612 Git commits
- 60 total peer reviews
  - 47 defects found and fixed
- 20 defects found via test, all fixed

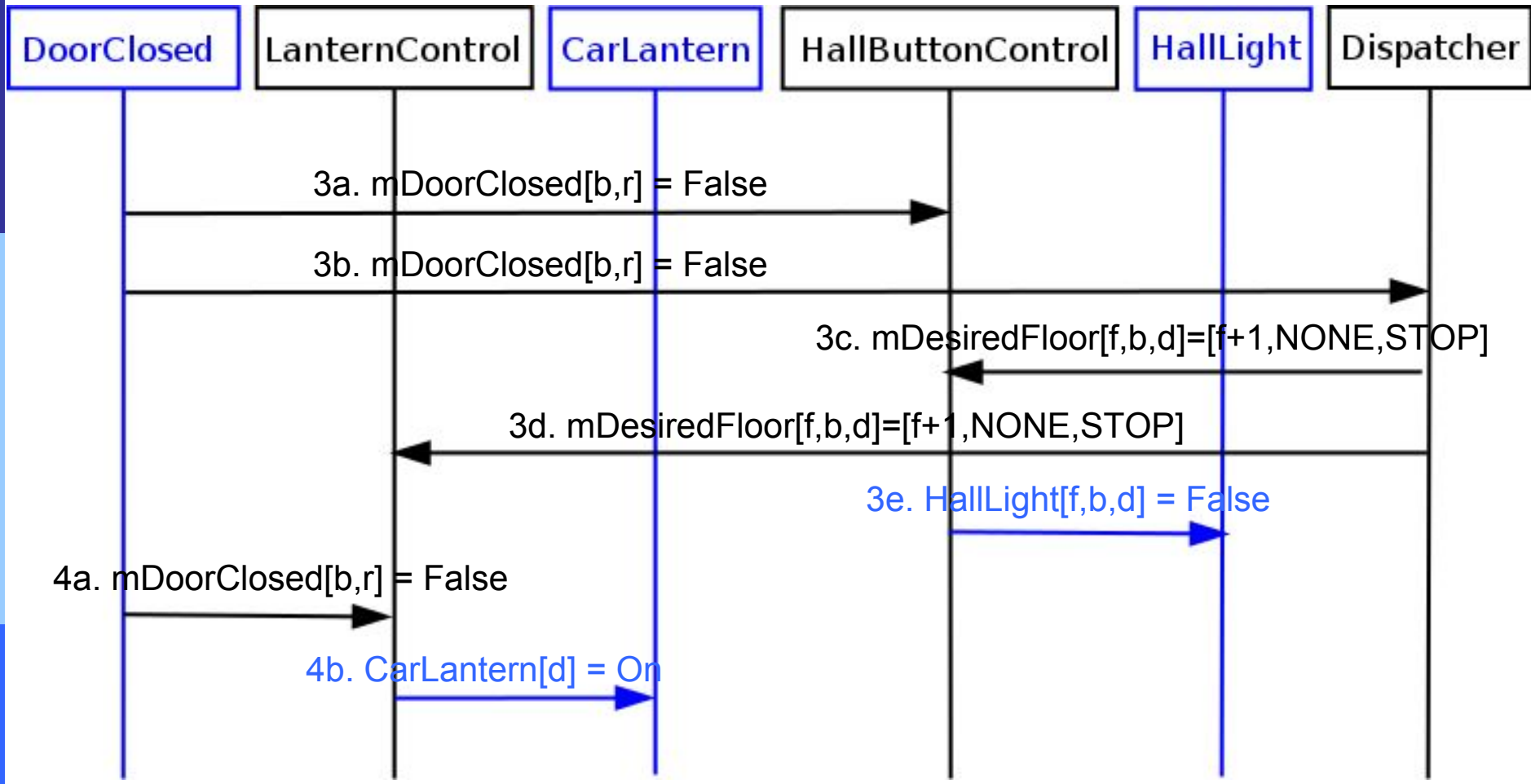
Dispatcher: Scenarios and Sequence Diagrams

Relevant Scenarios:

- 3A: Passenger in hallway, has pressed hall call
  - Dispatcher must update value of floor in DesiredFloor message before the doors fully open
    - Required to turn Car Lantern on, and Hall Light off
- 7A: Passenger reads car direction
  - Update value of floor as above to turn on Car Lantern
- 9A: Dispatcher cycles doors
  - DoorControl receives updated DesiredFloor message when the doors cycle

Daniel Haddox

Sequence Diagram 3A



Dispatcher Design - Requirements

State Variables

- Target: an integer Floor number for desired Floor, initialized to Lobby = 1.
- CurrentHallway: shorthand notation for the hallway of whichever mAtFloor[f, b] is True, if any. If CurrentHallway is invalid it has a value of None.

Constraints

11.1 Target shall be a valid Floor number from 1..MaxFloor inclusive.

11.2 The desired direction d of mDesiredFloor(f, b, d) shall not be Up when f = MaxFloor.

11.3 The desired direction d of mDesiredFloor(f, b, d) shall not be Down when f = 1.

Daniel Gorziglia

Dispatcher Design - Requirements

Time-Triggered Requirements

11.4 mDesiredFloor.f shall always be set to Target.

11.5 mDesiredFloor.d shall always be set to Stop.

11.6 Whenever any mDoorClosed [b, r] is False, then

11.6.1 Target shall be set equal to ((f mod MaxFloors) + 1)

11.6.2 mDesiredFloor.b shall be set to b, where f, b is whichever mAtFloor[f,b] is True

11.7 If all mAtFloor[f, b] are False AND any mDoorClosed [b, r] is False (which means doors are not closed between floors!); then

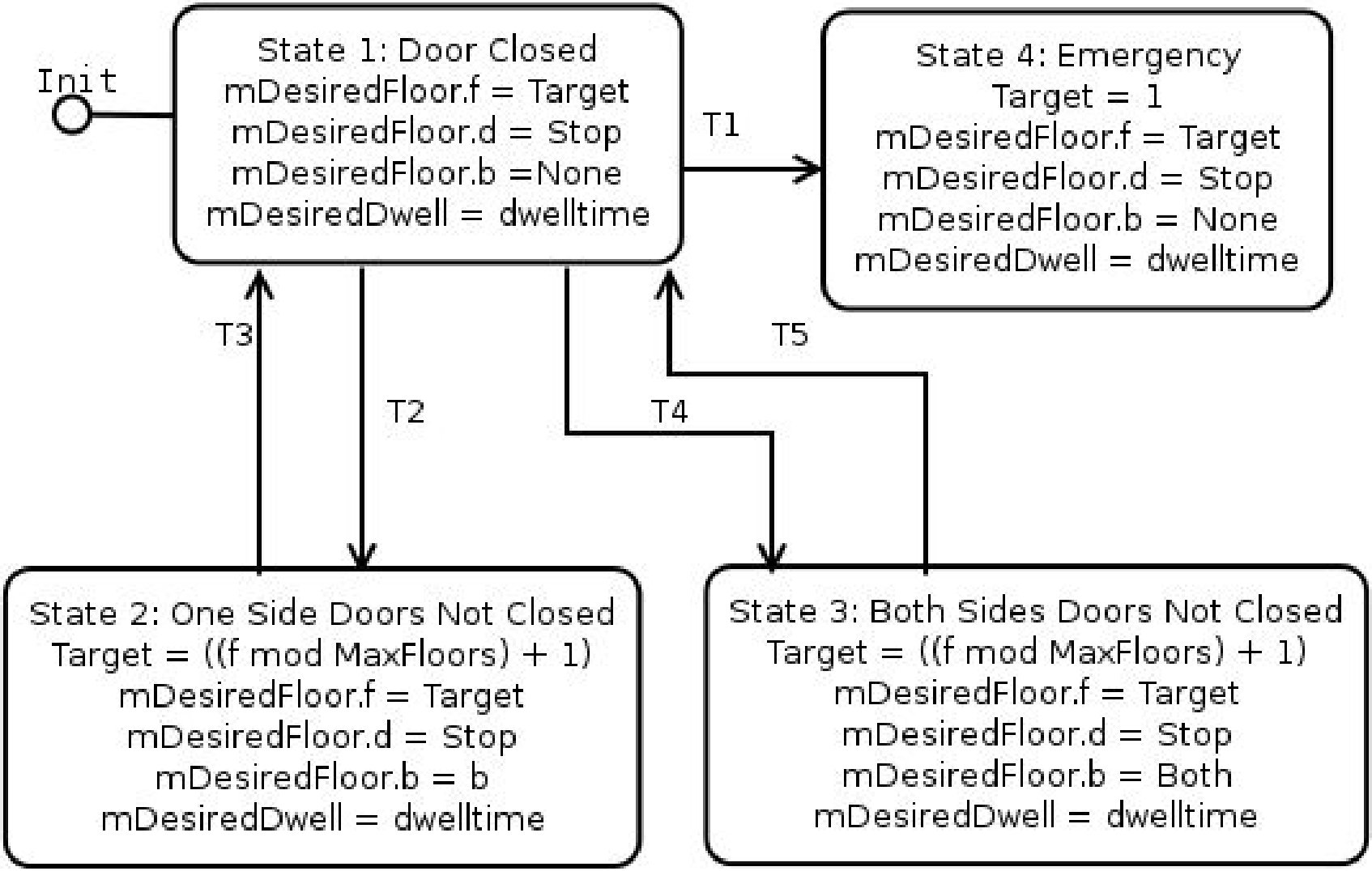
11.7.1 Target shall be set to 1

11.7.2 mDesiredFloor.b shall be set to None .

11.8 If two mAtFloor[f, b] values are True with the same value f, then mDesiredFloor.b shall be set to Both.

11.9 mDesiredDwell shall always be set to a constant appropriate value for door open dwell.

Dispatcher Design - Statechart



Dispatcher Design - Statechart Transitions

Transition	Condition
11.T.1	mAtFloor[*,*] == False AND mDoorClosed[*,*] == False
11.T.2	mDoorClosed[*,*] == False AND ~(mAtFloor[f,*] == True
11.T.3	mDoorClosed[*,*] == True
11.T.4	mDoorClosed[*,*] == False AND mAtFloor[f,*] == True
11.T.5	mDoorClosed[*,*] == True

Dispatcher Design - Code

- No surprises, implementation follows statechart
- Based on TestLight template
- High Level
  - Instantiate messages, mailboxes and translators
  - TimerExpired state chart
  - Switch case to set outputs in a state
  - Update new state based on transition conditions

Dispatcher Design - Code

Dispatcher.java

```
public void timerExpired(Object callbackData) {
  State newState = state;
  switch (state) {
    // State 1: Door closed
    case STATE_DOOR_CLOSED:
      // Set network messages
      mDesiredFloor.set(target, Direction.STOP, Hallway.NONE);
      mDesiredDwellFront.set(dwelltime);
      mDesiredDwellBack.set(dwelltime);
      //#transition '11.T.1'
      if (atFloorFrontTrue() == -1 && atFloorBackTrue() == -1 &&
          allDoorsClosed() == false) {
        newState = State.STATE_EMERGENCY;
      if …
    else
      newState = state;
    break;
```

Dispatcher Design - Testing

- Unit test
  - Instantiates Dispatcher
  - Exercises all transitions with injections
  - Tests outputs in each state with assertions
  - Make sure floors used in testing exist!
- Applicable integration tests
  - Test dispatcher in system
- Regression Testing
  - Run all tests before commit

Dispatcher Design - Unit Test

dispatcher.cf

```
Dispatcher 8 50ms true;
```

dispatcher_test1.mf

```
...
;#transition '11.T.2'
1.0s I 0s N DOOR_CLOSED_SENSOR_[FRONT][LEFT]_CAN_ID DoorClosed
FRONT LEFT = false
;#state '11.S.2'
1.5s A N DESIRED_FLOOR_CAN_ID DesiredFloor : getFloor == 2
1.7s A N DESIRED_FLOOR_CAN_ID DesiredFloor : getDirection == STOP
1.9s A N DESIRED_FLOOR_CAN_ID DesiredFloor : getHallway == FRONT
...
```

Lessons Learned

- Working in the same room has its benefits and drawbacks
- Give at least 4 periods between message injections and insertions
- You are given utility.java, defines.mf
  - Including some helper functions we re-wrote
- Create your own hand in checklist
- Be explicit with order of operations in code, use parentheses
- Writing scripts to automate testing and documentation helps a lot

Open Issues

- Open Issues
  - Our state names for DriveControl should read “One Hall Doors not closed”, “Both Halls Doors not cloded”, etc, not “One Side Doors not closed” which is incorrectly describing the left or right door, rather than front or back hall
- We anticipate that our biggest challenge moving forward will be implementing the fast speed DriveControl and properly calculating commit points

Thank you

Questions?