Project #8

18-649 Embedded System Engineering
Announcements and Administrative Stuff

◆ Project 8 posted

◆ Project 8 is due Thursday Oct. 23\textsuperscript{th} by 10pm

◆ Presentation template posted – complete for your assigned controller
  – Door Control - 1,4,7,10,13,16, 19, 22
  – Drive Control– 2,5,8,11,14,17, 20
  – Dispatcher– 3,6,9,12,15,18, 21
  • Focus on statecharts through testing
Project 8

◆ Pass all three Project 7 acceptance tests
  • Scripts will pick random seeds, test your design thoroughly!
  • Good idea to create scripts that will run acceptance tests in the background

◆ Start designing fast speed drive and smart dispatcher
  • You will need to add:
    – Scenarios and sequence diagrams
    – Time-triggered requirements
    – Traceability
New Requirements

◆ R-T6: The Car shall only stop at Floors for which there are pending calls.

◆ R-T7: The Car shall only open Doors at Hallways for which there are pending calls.

◆ R-T8: The Car Lanterns shall be use in a way that does not confuse passengers.
  • R-T8.1: If any door is open at a hallway and there are any pending calls at any other floor(s), a Car Lantern shall turn on.
  • R-T8.2: If one of the car lanterns is lit, the direction indicated shall not change while the doors are open.
  • R-T8.3: If one of the car lanterns is lit, the car shall service any calls in that direction first.

◆ R-T9: The Drive shall be commanded to fast speed to the maximum degree practicable.

◆ R-T10: For each stop at a floor, at least one door reversal shall have occurred before the doors are commanded to nudge.
New Requirements - conflicts

If any of the new requirements conflict with any prior requirements given for the controller, your elevator needs to be modified to satisfy the NEW requirements.
## Project Road Map

**Project 8: Advanced Elevator Design**
- Sequence Diagrams and Time Triggered Behaviors for the new elevator design satisfying the new high level requirements RT 6-10
- Clean up Project 7 code
- Write a monitor for RT 6 & 7

**Project 9: Implement Dispatcher**
- Implement Dispatcher and DoorControl
- Write Unit Tests for Dispatcher and DoorControl

**Project 10: Implement DriveControl**
- Implement DriveControl and Lanterns
- Write Unit Tests for DriveControl and Lanterns
- Write a monitor for RT 10

**Project 11: Network Scheduling**
- Adjust Network Traffic for 200 Mb/s
- Pass all your unit tests
- Write some integration tests
- Write a monitor for RT 9

**Project 12: Testing and Validation**
- Write Acceptance Test Generator, and run 100 tests
- Pass all your unit tests, write and pass all integration tests
- Write a monitor for RT 8

**Project 13: Handin**
- Pass all unit, integration, and acceptance tests, with no warnings
- Make portfolio clean and consistent
Fast Drive Speed

- Simulator assumes that car can instantly stop from slow speed

- Need to ramp down speed from fast in time to stop at desired floor
  - Cannot instantly stop from fast speed (engages emergency brake)

- **Commit Point:**
  The elevator position at which you must decide whether to stop at particular floor
  - Occurs when elevator reaches the stopping distance from that floor location
  - Think of it as a “point of no return”
Fast Speed Drive - Commit Point

◆ Stop speed  = 0.00 m/s
◆ Slow speed  = 0.25 m/s
◆ Fast speed  = 5.00 m/s
◆ Constant acceleration/deceleration = 1.00 m/s²

◆ Calculate the maximum stopping distance of the elevator
  •  \( x(t) = x_0 + v_0 t + \frac{1}{2} a t^2 \)
  •  \( v_f^2 - v_0^2 = 2a\Delta x \)

◆ Include slack for:
  • Sensor granularity (CarLevelPosition is in 10 cm increments)
  • Delay of DriveControl control loop
  • Be conservative!!
Only Service Landings with Pending Calls

- Elevator must only stop at floors/hallways that need to be serviced

- DesiredFloor
  - Floor – the floor we intend to go to next
  - Direction – the direction we intend to go after we reach the desired Floor
  - Hallway – which doors should open
Only Service Landings with Pending Calls

- Update desired floor/direction based on current state of hall/car calls
  - When is it OK to update these?

- For example:
  - If the elevator is stopped and opening its doors
    AND there is no pending call at the current floor
    AND there is a pending call at another floor
    THEN:
      - DesiredFloor.Floor must NOT BE current floor by the time the doors are fully open

- What about between floors?

- When should you NOT update these values?

- Above example is not a hard requirement
  - Follow the requirements and do what makes sense for your design
Example

- Suppose car is initially at floor 1 and stopped
  - No calls
  - Desired Floor = (1, stop)
Example

◆ Get a hall call for (8, down)
  • Car begins moving up
    – Current direction = Up
  • DesiredFloor.floor = 8
  • DesiredFloor.direction = Down
    – Where we’re going after servicing floor 8
Get a hall call for (8, down)

Then receive a hall call for (5, up)
  • Dispatcher decides to service floor 5 first
    – Depends on your algorithm
  • Current direction remains Up
  • DesiredFloor.floor = 5
  • DesiredFloor.direction = Up
    – Where we’re going after we service floor 5

How do you decide where to go next?
  • Based on current set of car/hall calls
  • Anything that meets the requirements is OK
    – Example: Sweeping up and down servicing calls in the current direction first
Modifying the Network Interface

◆ You can make ONE of the following modifications to the interface
  • Add mCarPositionIndicator to the input of the Dispatcher and DriveControl, OR
  • Add mDriveSpeed and mCarLevelPosition to the input of the Dispatcher.

◆ For any other modifications you need TA approval

◆ Remember to Completely Update Traceability if you make any changes.
Runtime Monitor

◆ Why monitor?
  • Helps to catch complex corner cases in Drive Control and Dispatcher
  • Helps discover design problems conflicting with high level requirements
  • Finding problems sooner allows for easier fixes

◆ Safety Monitors vs Performance Monitors
  • Performance monitors give a numeric value.
    – How Fast?
    – Number of overweight sensor trips?
  • Safety monitors are boolean.
    – Did we do something wrong?

◆ We monitor high level requirements
  • Safety monitors, or performance?
    – Safety since they answer the boolean question “Did we behave properly?”
High Level Requirement: “The elevator shall never stop at floor six”

State charts should:

- Mirror the actual state of the elevator
- Contain both valid and invalid states
- Throw a warning in invalid states
The monitor is NOT a new controller

◆ Monitor takes mostly physical payloads (few network messages)

◆ `receive()` function executes when the physical payload is sent

```java
public void receive(DriveSpeedPayload msg) {
    checkFastSpeed(msg);
}
```

```java
private void checkFastSpeed(DriveSpeedPayload msg) {
    // Update variables and check for violations
    // If between floors, at some point must go faster than slow speed
    // If reach a new floor and haven’t, then print violation
}
```

◆ Monitor must use `SystemTimer` objects (if you need them)
  • Don’t use Timer objects (only use these in your controllers)
  • This prevents the runtime monitor from contributing to randomness in simulation
Project 8 Monitor

◆ RT 6 & RT 7
  • Pending calls

◆ Run your monitor on project 7 code
  • with proj7acceptance 1.pass

◆ Will you find violations in monitoring proj7?
  • Probably, since the Sabbath elevator doesn’t work this way.
  • Log one of them (seed and timestamp)
  • Log a place where there’s not a violation.
Questions? Come to office hours!