Recitation #3

18-649 Embedded System Engineering

Friday 18-Sept-2015

Electrical & Computer ENGINEERING

Carnegie Mellon

Note: Course slides shamelessly stolen from lecture

Announcements and Administrative Stuff

- Project 3 posted
- Groups finalized. Any confusion ask TA's

- Project 2 due Yesterday

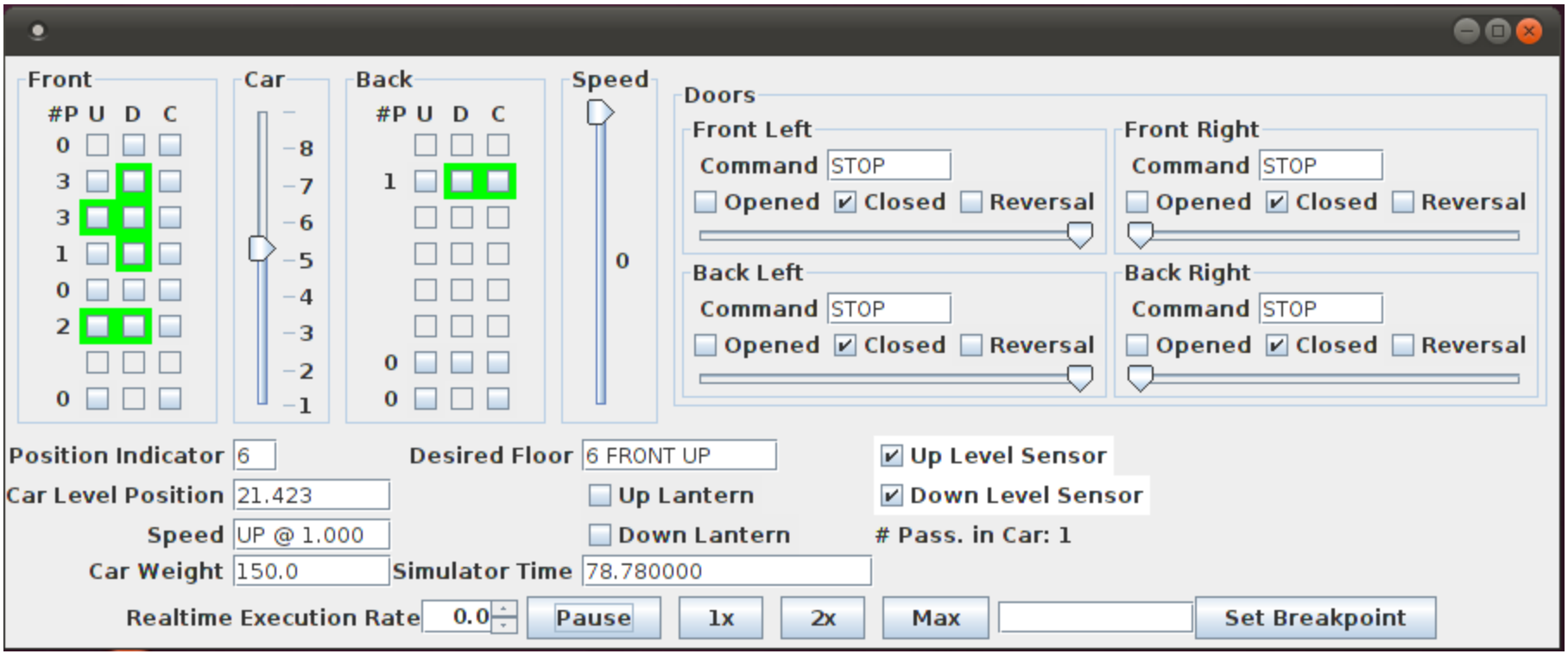
- TA office hours
  - http://www.ece.cmu.edu/~ece649/admin.html#info
  - Monday: **BH237B** 4:30-5:30 (Zach)
  - Wednesday: **BH237B** 8:00-9:00 (Jacob)
  - Thursday: **BH237B** 4:30-5:30 (Shane)

Minimum Requirements Chart

- A way for TAs to check if you fulfilled the minimum requirements for each project.
- ***Shall*** be downloaded and completed for each project.
- Project is not turned in until we have the chart

GUI Overview

- Great debugging tool for later projects
- Good mental concept of the elevator for design projects



Project 3 Overview

- Write requirements for an *event-triggered* system
  - DoorControl [b, r]
  - CarPositionControl
  - Dispatcher

  } These are done for you already

  - DriveControl
  - LanternControl [d]
  - HallButtonControl [f, b, d]
  - CarButtonControl [f, b]

  } You specify requirements for these

- Traceability
  - Requirements to sequence diagrams
  - Sequence diagrams to requirements

  - *ALL SEVEN* controllers need to be included in traceability

The Magic Formula for Event-Triggered Systems

- Behavioral requirements
  - (ID) <message received> shall result in <message transmitted> …
    and/or <variable value assigned> …
  - OR
  - (ID) <message received> and <variable value tested>
    shall result in <message transmitted> …
    and/or <variable value assigned> …
  - Account for all possible messages received; OK to restrict by value
    - E.g., <message received> with value V shall result in …
  - Account for all possible messages that need to be transmitted outbound
  - Make sure all variables are set as required in right hand sides
  - *EXACTLY ONE* received message per requirement (network serializes messages; simultaneous reception of messages is impossible)
  - OK to have: multiple messages transmitted; multiple variables assigned

From Sequence Diagrams to Requirements

- For each controller
  - Find all sequence diagrams that include that controller
  - Identify all incoming /outgoing arcs for the controller in a diagram
  - Note any variables that need to be tested or set

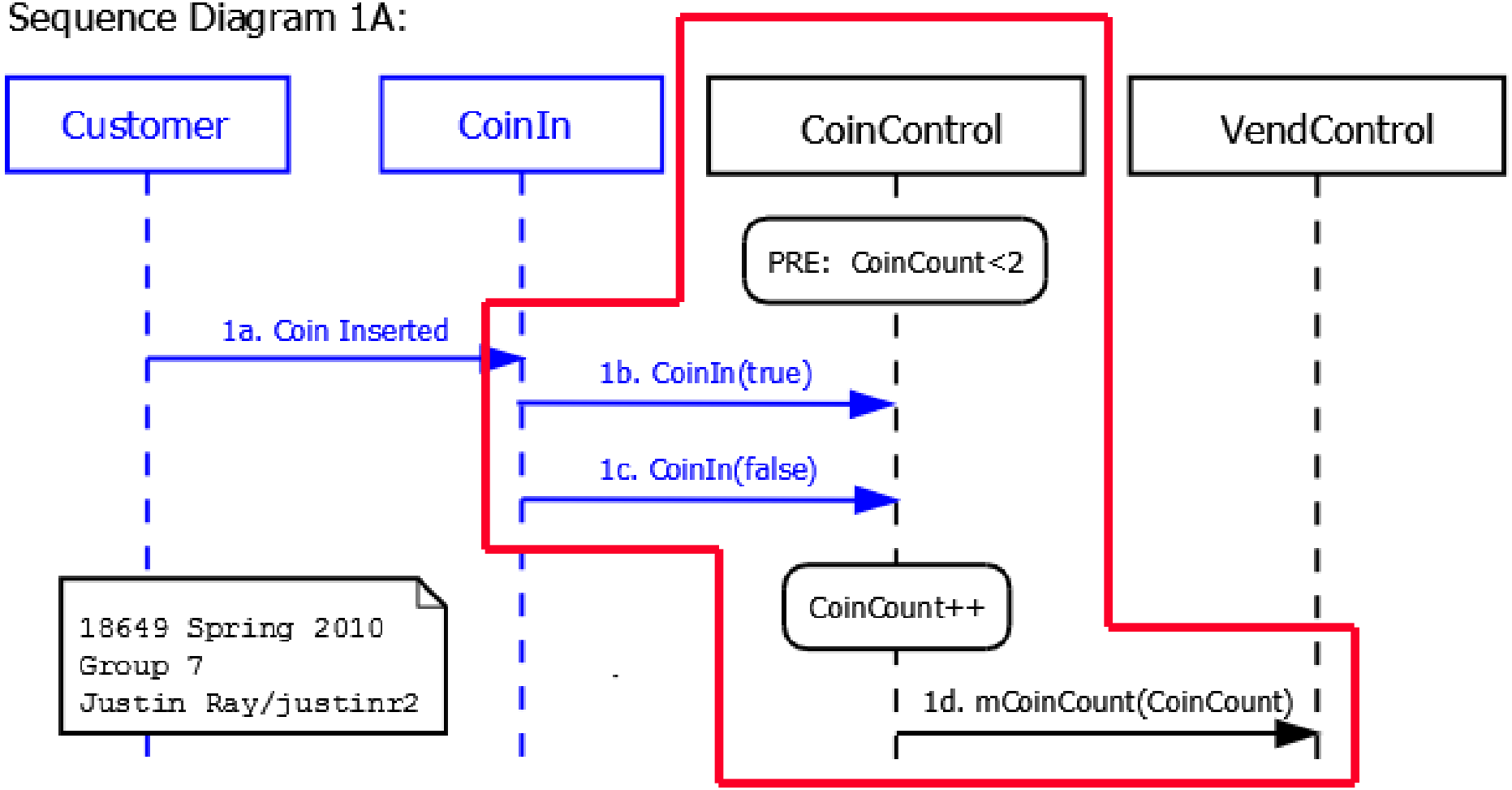
- Gives you a behavior that you've defined in that sequence diagram
  - Incoming message arcs trigger the event (or cause variables to be set)
  - Outgoing messages are the resulting transmissions from the event
  - Test and set variables as appropriate

- Use **Shall** and **Should**

Soda Machine Example - CoinControl

- Scenario 1A: Customer inserts a coin when the cost of a soda has not been reached

Sequence Diagram 1A:



Note: SodaCost = 2 coins

Example Requirement

- Incoming arcs (and values)
  - CoinIn (true)
- Variables
  - CoinCount
- Outgoing arcs (and values)
  - mCoinCount (CoinCount)

- Example requirement (you might come up with something different):

  RCC.1 - If CoinIn is received as true then,

  RCC.1.a - CoinCount shall be incremented and

  RCC.1.b - mCoinCount shall be set to CoinCount
- Anything you need to be careful about with the above requirement?

- Check out the soda machine design for more example
  - Disclaimer: Soda machine is in development, it may have occasional bugs

An Elevator Example

- Sample Scenario 2A:
  - Passenger is in the car and elevator is not at the desired destination floor

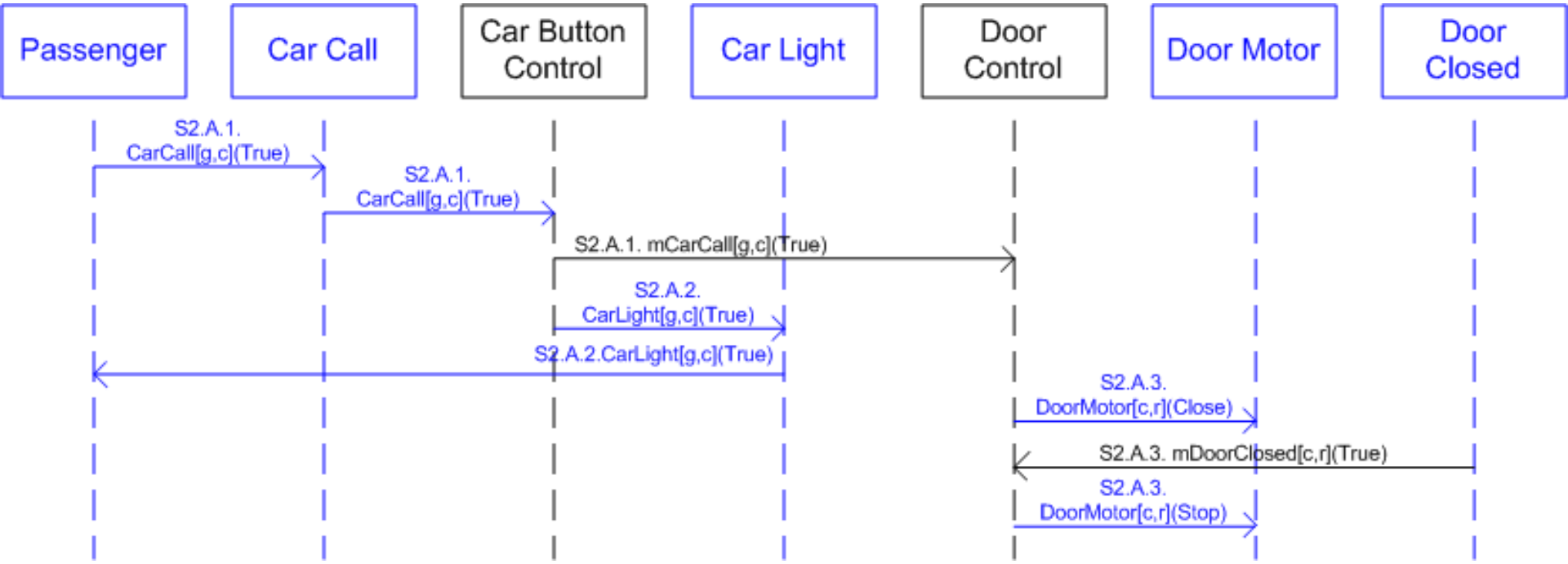
- Pre-Conditions:
  - Car is at floor f, with at least one Door[b,r] open.
  - Passenger is in the car and elevator is not at the desired destination [g,c], where f != g. Also, b might not equal c.
  - Car call button for desired destination is not lit.

- Scenario:
  - S2.A.1. Passenger presses car call button for desired destination [g,c].
  - S2.A.2. Car call button for destination [g,c] is lit. Passenger sees button light up.
  - S2.A.3. Doors close fully.

- Post-Conditions:
  - Elevator has not yet arrived at destination [g,c].
  - Passenger is in the car.
  - All doors are closed.
  - Car call button light for desired destination [g,c] is on.

Elevator Example

- Scenario 2A: Passenger is in the car and elevator is not at the desired destination floor (this ignores the dispatcher)



- What's an event-triggered requirement for Car Button Control?

- Note these are just examples, yours will likely look different
  - There is no single correct answer

Some Requirement Guidance

- Keep them short and concise
  - All but the most complex **should** be less than 25 words,
    - 50 words borders on excessive
  - All requirements **shall** be less than 100 words
  - Don’t ramble; avoid ambiguity.
    - Another team mate might have to implement that requirement later!
- Use English
  - Each requirement **shall** be a complete English sentence
  - Not a line of code!
- Each requirement **shall** have exactly one verb
  - You’ll likely end up with multi-part requirements
    - Refer back to the CoinCount example
- Explicitly record all variables you use in requirements

Traceability

- Trace all seven controllers
  - Another teammate must trace the controller requirements you wrote
  - The excel template is in the portfolio
- Complete forward traceability
  - Each sequence diagram message maps to at least one requirement
  - Ensures you didn't leave out any behaviors
- Complete backward traceability
  - Each requirement maps to at least one sequence diagram message
  - Ensures no spurious or unwanted behaviors
- But what if you realize something important is missing?!
  - Add the missing requirement or sequence diagram message if necessary
  - Its OK to go back and fix sequence diagrams
    - We require a working elevator and complete documentation!
  - Now's a good time to get familiar with that issue log

Peer Reviews

- **For each project we want you to do at least one peer review per person**
  - For this project, we want you to review requirements for each controller
  - Just do the four controllers you wrote requirements for

- **Peer review procedure**
  - Reviews shall be performed by someone other than the primary author of the "artifact".
    - "Artifact" is a diagram, set of requirements, statechart, etc.
  - Reviews should be performed by a team member who did not contribute at all to creating the artifact (an independent reviewer)
  - The reviewer looks at the artifact and creates a review sheet
    - We give you an Excel template, but you can use something else comparable
    - The review sheet records that the review happened, and lists any problems found
    - Use a separate review sheet for *every* review (so there will be many such sheets by the end of the semester
  - When the review is completed, it's added to a web page that lists all reviews for your project, accumulated over the semester

Questions?