Recitation #1

18-649 Distributed Embedded Systems
Friday 4-Sep-2015
Announcements and Administrative Stuff

◆ Project 1 draft posted
  • Individual work, with the exception of the audit
  • When the web box goes from gray to white it is officially released

◆ Groups will be formed next week after add/drop settles

◆ TA office hours
  • Look for posting next week (Tuesday) with office hours

◆ Course announcements will be made mostly via blackboard
  • You are responsible for these (check it once a day or so)
  • Only critical notices sent by e-mail

◆ Recitations every Friday
  • In this classroom 1:30-2:00 PM weekly
  • After this week there will be team meetings; watch blackboard for details
  • Most team meetings in this classroom 12:30-2:30 PM
  • One team per TA will meet outside that range at a mutually agreeable time
What If You See A Really Great Idea?

◆ Each group shall do all their own development
  • It’s rare that there’s only one way to implement something
  • You’ll definitely see some unique implementations this semester

◆ Copying any portion of another group’s work is cheating
  • Even if you cite where you got it
  • Accidental parallel invention is fine (yes, we can tell the difference)

◆ If you see an idea in someone’s presentation that would make your elevator WAY better (e.g. a better dispatcher algorithm)
  • Modify your own implementation and cite where you got the idea
  • That is a perfectly reasonable thing to do
  • Lines of code are not an “idea” – they are an implementation
  • Do not look at anyone else’s elevator/project code. Ever.

◆ To be safe, the only inter-group idea exchange should be presentations
  • Exchanging ideas WITHIN a group is encouraged at all times!

◆ If you’re not sure, then ask us! Do not assume
Weekly Progress & Individual Contribution

◆ Status reports due every week with your project submission
  • Starts next week

◆ Tell us:
  • How many hours you’ve spent this week (including class time)
    – Be accurate, helps us tune the project load and keep your work to ~12 hrs / week
  • Breakdown of what each team member contributed this week
  • Any other project issues/comments

◆ Your hours report does not affect your grade; be honest!

◆ Starting with Project 2 there are individual contribution requirements
  • At least some of lab is distributed per person
  • Ensures everyone touches the basics
    – Do NOT cover up someone slacking – if they didn’t do the work, don’t give them credit
Some Elevator Lingo

- **Elevator Car** is the compartment passengers ride in

- **Drive** is the motor that moves the car up and down in the shaft
  - Drive speeds for our elevator are simply Stop, Slow, or Fast

- **Hall Call** buttons are the up/down buttons in the hallways at each floor

- **Car Call** buttons are the buttons inside the elevator

- **Door Reversal** sensors check if the doors are blocked by an object or passenger
  - “A door reversal occurs”
    - An object or passenger was in the way when the door tried to close
    - The door stops, *reverses* direction, and reopens

- **Car is overweight** if there are too many passengers in the car
  - One or more passengers must get off for safe operation to continue
Given a set of initial conditions, follow the process to develop a set of requirements for the door controller.

**Process:**

1. Generate a scenario
2. Write requirements using precise words
3. Write requirements using consistent terms
4. Number requirements
5. Testable
6. Traceability for high level requirements
7. Process audit

This process will give you a taste of how to generate requirements.
Process

1. Generate a scenario for the initial conditions
   • A scenario is a short story that corresponds to the initial conditions
   • Pretend you’re an omniscient observer and tell us what you observe happening
     – You would notice the elevator arrives with doors shut, then the doors open
     – You don’t need to include events that are not observable to users
       » Example: Events that don’t happen like door reversals not occurring
     – Suppose the car is overweight when a passenger gets on, what would you observe?

2. Generate 2 - 5 requirements for the door controller based on the scenario
   • Use SHALL and SHOULD as per lecture

3. Revise the requirements in step 2 to use consistent terms
   • Use the terms defined in the interface section with appropriate notation
   • If you don’t understand the interface, come to office hours and ask
   • Don’t forget to consider messages like mDriveSpeed and mAtFloor
Process (Cont)

- **Number the requirements**
  - Very useful for traceability, audit, and communication about design documents

- **Write a test case for each requirement**
  - Describe test inputs and described expected outputs that confirm the requirement is met

- **Traceability to high level requirements**
  - How does each of your requirements support and/or not violate the high level requirements?
  - Check ALL high level requirements for EACH of your requirements

- **If you get lost:**
  - Look at the Pepsi Machine example on the course web site
  - Come to office hours and ask
  - Take your best shot; this is to get you ready for lectures with more detail
Audit

Once finished steps 1-6, exchange with another classmate to audit

- You are responsible for finding another classmate (perhaps in your group)
- The other classmate must also be finished
- This is the *only* non-independent part
- Once you get feedback, fix your mistakes (you’ll get a better grade)
  - Grade on following process, NOT on perfect pre-review work product

Audit Questions

- Step 1: Is there a scenario?
- Step 2: Do the requirements use only shall and should, or the appropriate words from the lecture?
- Step 3: Do the requirements use the terms given in the Interface section to describe sensors, actuators and controllers on the elevator?
- Step 4: Are the requirements numbered?
- Step 5: Is there a test given for each of the requirements?
- Step 6: Traceability - Does *each* requirement either support or not contradict *each* of the high-level requirements?
Project 1 - Part 2 - Play with the Simulator

- Download and exercise the elevator
  - Follow the directions on the web

- Requires you to make a small correction to testlight.java
  - Mostly to ensure you can figure out how to edit/compile/run

- Run an acceptance test and see the passenger is delivered
Staff E-Mail Usage

◆ E-mail only for administrative issues and bugs in infrastructure
  (ece649-staff@lists.andrew.cmu.edu)
  - Inform staff immediately about administrative issues via e-mail
    - Example: I can’t access my AFS submission directory or We have group problems
  - Send all e-mail to staff, not an individual TA
    - If one of us is offline, someone else can respond more quickly

◆ E-mail staff about errors in assignments or the simulator
  - We endeavor to provide a bug-free simulator, but there may be occasional bugs
  - Read the admin.html page and find the checklist to use before reporting bugs
  - Reasonable bug reports receive high priority; don’t skip the checklist
  - Yes, the checklist is long. You’ll find almost all bugs are in your code.

◆ Don’t expect technical support over e-mail
  - Includes bugs in your code, assignment clarifications, etc.
  - We do not support tools. At all. Don’t ask. Pick tools you are comfortable with.

◆ Feel free to ask for clarifications during class, recitation, or office hours
  - Ask about individual/group project guidance during office hours
Please use this check-list before submitting an e-mail regarding the course project:

◆ Check blackboard to see if an answer has been posted.
◆ Re-read the assignment to make sure you are reading it correctly.
◆ Look at the grading checklist to see if it has relevant information.
◆ **Look at the Pepsi machine example**
◆ Find a reasonable way that doesn’t violate requirements
◆ If you simply don't understand, skip the e-mail and come to office hours.
◆ Look again at examples; OK to look at simulator code too
◆ Include the URL of the document and specifically tell us the defect.
◆ Please follow the bug report instructions in the Project FAQ. If we can’t reproduce it, probably we can’t fix it.
◆ Wait 5 minutes before sending. Seriously. “Oops, found it" e-mails waste a lot of everyone's time.
◆ Start your e-mail with "I've used the e-mail question checklist, and I think the following is an issue:" or we might reply “did you use the checklist?”
Suggestions and Reminders

◆ Read the next-week project assignment before you show up for recitation each week

◆ For project 1, read the interface carefully!
  • This will help clarify what terms like “door reversal” or “hall call” mean
  • If confused, look at the Pepsi machine and take your best guess

◆ Read Project FAQ directions for submissions directions!!!
  • Has checklist to avoid making mistakes in hand-ins – use it!
  • Before assignment is done, put a test file in the AFS submission directory
  • Let us know early if you have submission issues

◆ Start early with assignments
◆ Start thinking about groups

◆ Dates: Team requests are due Wednesday evening by 5 PM
  Lab Hand-ins are due Thursdays at 10 PM
  Be ready for Friday TA meetings
Questions?