Working in Groups

CSE 331, Spring 2011
Section 9
What this *is*

- A collection of our personal experiences & observations from working on group software projects
  - (Capstone classes, software engineering projects, internships)
- Maybe some advice
What this is *not*

- A definitive guide to everything that could go wrong (or right!)

- A replacement for the valuable things you will learn struggling through your own group project(s)
  - There's nothing like first-hand experience!
The beginning of a project

- The most important part!!

- Early decisions about the program structure / architecture will have a big impact later, so consider them carefully

- Early investment in the development "infrastructure" will pay off as the project gets larger
  - setting up a source code repository
  - scheduling meetings
  - deciding group roles / dividing up responsibilities
  - setting up automatic building & testing
  - coming up with a development plan
  - defining the interfaces / interactions between parts of the program
How to divide your project into tasks

- Try to avoid one member's tasks depending on the completion of another member's tasks
Dividing up tasks

- Give everyone a specific task
  - This way nobody gets confused about what they are supposed to do

- Write down who is doing what task and give the list to all group members
  - Forgetting what you agreed to do is not an excuse
Set mini deadlines

- Help you keep on schedule
- Demonstrates your progress to the rest of the group
Meet in person!

- Meet with your group, do not leave all contact to email
- Sit down and talk about the overall goal, and do the high-level design together.
- It is not necessary to hammer out every last detail, but sketch out the interfaces so you know how each piece interacts when you split up the work
  - Once you have defined interfaces, try not to change them!
Pair programming

- Interesting/useful for particularly complex feature.
  - One person will 'drive' and the other will act as a second pair of eyes, catch things the other misses and offer alternate approaches
Communication with group members

- Once you've divided up responsibilities, a lot of your communication will be about the interaction between your parts of the project
  - One reason to keep the interfaces / interactions between parts of the program simple

- Consistency is very important
  - It takes a lot of time and energy to communicate "unusual" nuances of your interface
  - Better to just keep things consistent
Disagreements

- Disagreements happen all the time
  - A normal part of the design process
  - It doesn't mean your team is "dysfunctional" or "failing"

- Try not to take it personally!
  - They're not attacking you - just explaining a different perspective of a complex problem
  - Think of disagreements as opportunities to learn

- At some point, you need to resolve your disagreement and get back to work
  - An arbitrary decision is better than being stuck forever
  - Have a plan for resolving disagreements, e.g. "after X minutes of debate we will take a vote" or "flip a coin"
Now, your turn!
The actual program

● What have you done so far?
● What do you expect to have done by Saturday?
● What extra feature(s) are you thinking of adding?
● What has been the trickiest technical part? *(Besides SVN and design decisions)*
Yahtzee project discussion (part 2 / 4)

The development process

- What's your development strategy?
  - What code did you start working on first?
  - What are you saving for later?

- How are you developing the code?
  - Pair programming? Solo?

- Something that you like about your work / design so far?
Working in a group

- How is your group organized?
  - Roles? Functionality?
- Any team issues or challenges you didn't expect?
- How are you handling problems / code conflicts / disagreements?
- Best / worst thing about group work so far?
Questions

● Questions about the design of this project? (Or your particular design?)
● Questions about requirements or features?