CSE 331

Group Projects and Teams

slides created by Marty Stepp
based on materials by M. Ernst, S. Reges, D. Notkin, R. Mercer, Wikipedia
http://www.cs.washington.edu/331/

Team pros and cons

- Having more people work together has benefits:
  - Attack bigger problems in a short period of time.
  - Utilize the collective experience of everyone.
- Having more people has risks, too
  - Communication issues, conflict, and mistrust.
  - Diffusion of responsibility.
- Successful teams give each member clear responsibilities:
  - Each person knows and is accountable for their work.
  - Monitor each person's individual performance.
- Have an effective communication system:
  - Don't allow a problem to fester until it's too late (a "boiled frog").

Team organization

- Most successful teams break into sub-groups *by functionality*.
  - GUI team? Model team?
  - Strategies team? Testing team?
  - ...

- Possible individual roles:
  - unofficial *team manager* to break ties or make decisions
    (But you are all "equals," so nobody is truly the boss of anybody)
  - *communication manager* or secretary
    (checks up on group members' progress; sends emails to TA)
  - *meeting planner* (coordinates when people will get together)

Leadership and decisions

- How are big decisions made in your team?
  - One person? All, by unanimous consent? Majority vote? ...
- Let everyone give their input (even if some of it is off-track).
  - Write down pros/cons of alternatives in a grid or table (weighted?).
- Have a clear procedure for what to do for a disagreement.
  - Strive for consensus, but if it cannot be achieved, ...
  - Majority vote? Designate a "manager" who decides on a tie? etc.
- Compromise, compromise, compromise.

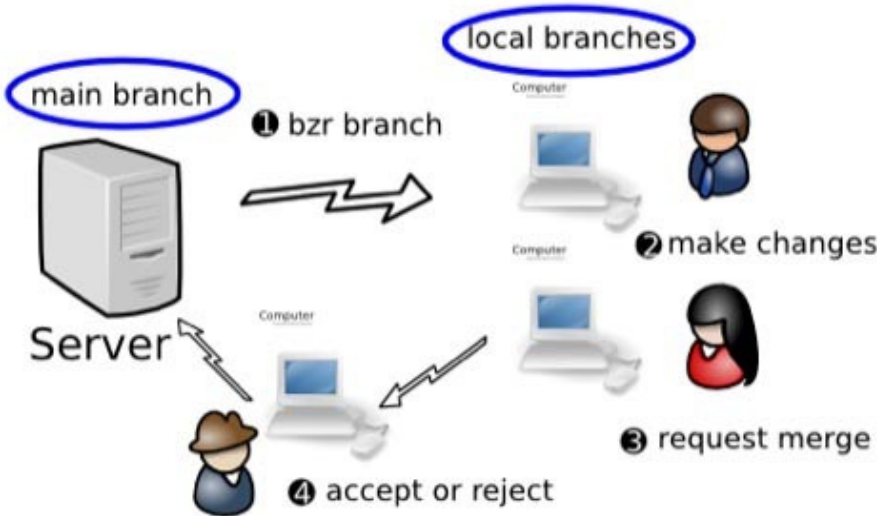
Once decisions are made

- Write down decisions and responsibilities clearly:
  - document by email, or post to a shared wiki or web page
- Come up with a set of steps to accomplish the decided goal:
  - list *specific dates* that progress should be made
  - come up with several small milestones within the overall goal
  - agree to communicate and/or meet after milestones are done
- prioritize and order goals and TODOs:
  - list them by urgency, by due date (or milestone date)
  - make sure to list group member(s) are responsible for which
  - make sure every group member has a significant role

Version Control

- When working on a non-trivial piece of code in a group, *always* use a version control system (svn, git, mercurial, ...).
  - Don't just store the code on someone's computer.
  - Don't email code files to each other.

- Version control benefits:
  - avoids overwriting changes
  - shared access to code
  - safe backup
  - allows branching and merging



Coding nicely with others

- Make sure that the code you check in is **robust** from the start.
  - Add your error-handling and argument-checking early.
  - State your class invariants and preconditions, and check for them.
  - This way, partners using your objects won't have hidden bugs.

- Make **check-ins** of appropriate size and frequency.
  - Don't work for days and then check in 1,000 lines of code.
  - But don't check in every few new lines you write.
  - Don't check in code that is broken, doesn't compile, or will cause another part of the app to break.
  - Resolve any merge conflicts carefully if necessary.
  - Always write descriptive check-in text so others know what you did.

"Extreme" programming

- **pair programming**: Two programmers working together at the same computer at the same time.
  - supposed to sit close to each other
  - take turns "driving" at the keyboard (very important!)
  - nobody types any code until both people agree and understand

- Benefits:
  - Doesn't produce 2x the code.
  - But produces *better* code.
  - Weaker programmers learn.
  - Stronger programmers avoid writing hackish code.

Achieving productivity

- How can you get the most out of your team members?
  - Give them specific, small, attainable goals that they can visualize.
  - Have frequent communication and updates.
  - Meet in person to work as much as possible.
  - Put people in small teams (~2); minimize work done "solo."
  - Build good team camaraderie.
- What can block people or stop them from making progress?
  - technical confusion: *How do I start implementing that feature?*
  - unclear responsibilities: *Oh, am I supposed to do that?*
  - unclear due dates: *When was I supposed to have that done?*
  - lack of milestones: *But it's not due until next Friday!*
  - laziness: *Time to work on code ... Ooh look, World of Warcraft!*

Dealing with slackers

- What do you do if a group member is slacking off instead of working?

  - Check up on them frequently.
  - Give them less "solo" work; put them in a sub-team of 2-3.
  - Have them meet more in person (harder to slack in front of you).

- If the problem persists, then what?
  - Anonymous feedback
  - Have others send them a kind but firm email with concerns.
  - Have an in-person meeting with a few members.
  - Contact us to let them know about the potential issue.

Email question

- What's wrong with this email?

"Hey Jim, I was wondering if you finished the XYZ feature you were assigned yet? You really messed up the ABC feature last week so I thought I better email you. It's due the day after tomorrow at 8am, you know! When you have time, please tell me if XYZ is done, okay?

-- Ralph"

Being quantitative

- Be **quantitative** and **specific**:
  - Use specific, incremental goals, not just for things to be "done."
  - List particular dates that results are expected.
  - Give an expected date/time to reply to a communication.
  - Don't be accusatory; offer support, help, gratitude as appropriate.
  - Remind about upcoming deadlines, meetings, key points.

- possibly better email:

  "Hey Jim, how is your work on the XYZ going? It's due a week from Friday. Like we talked about at our last meeting, we are hoping to have the rough sketch of the first 2/3 of it by Sunday so we can go over it together. Please let me know by tomorrow night how much progress has been made. If you have any questions or need some assistance along the way, please let me know. We'll all meet Saturday in person and you can give us another update at that time. Thanks! -- Ralph"

Running a meeting

- How to run an effective meeting:
  - Bring an agenda of topics to discuss (best to email this ahead of time).
  - Each member/subgroup reports its progress.
  - Get an update on every current work item.
  - Take notes on decisions made and post or them to everyone.
  - Have a whiteboard/paper handy for sketching out ideas.
  - Keep everyone's attention (maybe ban laptops / cell phones, etc).
  - Walk away with a clear plan of action, set of TODOs, etc.

Meeting gotchas

- Don't tolerate people showing up late, not being on task, etc.
  - Punctuality is expected; have an on-task discussion.
- Don't ignore a group member's input
  - Even if you don't go forward with their idea, at least listen to it.
- Don't let it run too long (people hate long meetings).
  - If a discussion is sidetracked, steer the group back to the agenda.
  - If still running over, find a stopping point and save some for next time.
  - Agree to discuss some issues over email if necessary.
- Don't "meet just to meet"
  - if you have nothing to discuss, make it a "work meeting" in the lab or cancel the meeting altogether (no one will complain)
- Don't use meetings for one-way flow of information (use email).