# CSE 403
# Lecture 5

Working in Teams

Thanks to Marty Stepp, Michael Ernst, and other past instructors of CSE 403
http://www.cs.washington.edu/403/

# Big questions

- How do you decide who should be project manager?
  - What's the difference between project manager and tech lead?
- How do you divide your team into subgroups?
  Who will work on what, and with whom?
- How will we make decisions about our project?
- How will everyone communicate and stay in sync about important decisions and issues?
- What will we do if someone is not doing their share?
  - How can we motivate team members to prevent this?

# Team pros and cons

- Having more people 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
  - Diffusion of responsibility
  - Working by inertia; not planning ahead
  - Conflict or mistrust between team members

# Issues affecting team success

- Presence of a shared mission and goals
- Motivation and commitment of team members
- Experience level and presence of experienced members
- Team size, need for bounded yet adequate communication
- Team organization
  - and results-driven structure
- Reward structure within the team
  - incentives, enjoyment, empowerment (ownership, autonomy)

# Common roles

- Many software development teams include:
  - **designers** (system design, documents, communication)
  - **developers** ("devs")
  - a **head developer** or **tech lead**
  - **testers**
  - a **project manager** a.k.a. "PM"
    - functional and/or project management responsibilities

- These could be all different team members, or some members could span multiple roles.

- Key: Identify roles **and** responsibilities

# Team leadership

- Who makes the important big decisions in your team?
  - One person?
  - All, by unanimous consent?
  - All, by majority vote?
  - Other options?...

  - Is this an unspoken or explicit agreement among team members?

- *Question:* What are some good/bad ways to make decisions?

# Making decisions

- 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?)
- think about cost/benefit, risk analysis of each proposed idea
  - How long will it take?  How much to learn?  Do we have to do it?
- have a clear procedure for what to do for a tough disagreement
  - strive for consensus, but if it cannot be achieved, ...
  - majority vote, and PM decides on a tie, ... etc.
- stepladder technique: have 2 people decide, then 3, then 4, ...
- Pareto analysis: find 20% of work that solves 80% of problem
- compromise, compromise, compromise

# Once decisions are made

- write down decisions and responsibilities clearly
  - document by email
  - 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
  - list several smaller milestones
  - 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

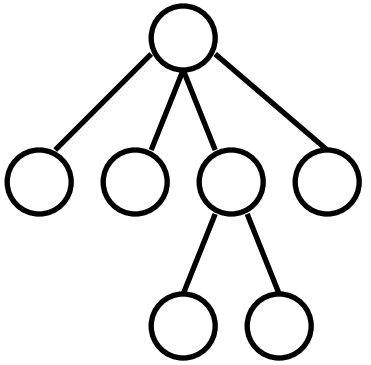
# Kinds of teams

- **problem-resolution**: a focused attack on specific issues
- **creativity**: coming up with and exploring new ideas
- **tactical-execution**: carries out a defined plan

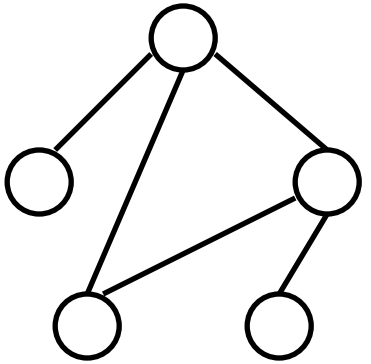
- Some team models
  - **business**: tech lead and a bunch of equal devs
  - **chief programmer / surgical**: lead dev does most of work
  - **skunk works**: just turn the devs loose
  - **feature**: divide work based on features of product
  - **search-and-rescue**: focused on a specific problem
  - **SWAT**: skilled with a particular advanced tool(s)
  - **athletic**: carefully selected people w/ specialized roles
  - **theater**: "director" assigns roles to others

# Team structure models

- Dominion model
  - Pros
    - clear chain of responsibility
    - people are used to it
  - Cons:
    - single point of failure at the commander
    - less or no sense of ownership by everyone



- Communion model
  - Pros
    - a community of leaders, each in his/her own domain
    - inherent sense of ownership
  - Cons
    - people aren't used to it (and this scares them)



# Common positive factors

- Results-driven structure
- Clear roles and responsibilities
  - Each person knows and is accountable for their work
- Monitor individual performance
  - Who is doing what, are we getting the work done?
- Effective communication system
  - Available, credible, tracking of issues, decisions
  - Problems aren't allowed to fester ("boiled frogs")

# Organizing by functionality

- Who will do the ...
  - scheduling? development? testing? documentation (spec, design, write-ups, presentations)? build/release preparation? inter-team communication? customer communication?

- Pragmatic Programmer tip:
  *"Organize around functionality, not job functions."*
  - Don't separate designers from coders, testers from data modelers, etc.

- *Question:* What are some benefits of organizing teams around functionality vs. around job functions/titles?

# 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; 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 403 ... 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 the PM send them a kind but firm email with concerns
  - have an in-person meeting with PM and a few members
  - contact primary customer to let them know about 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 were late on the ABC feature from the last phase so I thought I better email you. When you have time, please tell me when XYZ is done.

-- 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"

# Slackers, continued

- group relations must be handled with care
  - "slacker" may feel singled out by rude or critical messages / meetings
  - There are two sides to every story.
  - goal: to improve problem and maintain positive group relations
  - it is possible to be critical without being harsh, rude, mean
  - it is often helpful to avoid the appearance of a 1-on-1 conflict
    - PM can speak on behalf of other group members
      - "We" are concerned about XYZ, not "I"
  - depersonalize by making it about all group members, not one
    - "We" are a little behind, so let's all meet in person in the lab today

# 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 them to wiki, etc.
    - include the reason(s) why a particular decision was made
  - have 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;  PM should manage 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 discussion is sidetracked, PM should steer group back to 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  (email)