Working in Teams

Big questions

- How do you divide your team into subgroups?
- How will your team make decisions?
- How will everyone communicate and stay in sync?
- How can you motivate all team members to do their share?
  - What will you do if they don't?

Team pros and cons

- Benefits
  - Attack bigger problems in a short period of time
  - Utilize the collective experience of everyone

- Risks
  - Communication and coordination issues
  - Groupthink: diffusion of responsibility; going along
  - Working by inertia; not planning ahead
  - Conflict or mistrust between team members

Communication: powerful, costly!

- Communication requirements increase with increasing numbers of people
- Everybody to everybody: quadratic cost
- Every attempt to communicate is a chance to mis-communicate
- But *not* communicating will *guarantee* mis-communicating

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
  - and the need for bounded yet sufficient communication
- Team organization
  - and results-driven structure
- Reward structure within the team
  - incentives, enjoyment, empowerment (ownership, autonomy)

Team structures

- Tricky balance among
  - progress on the project/product
  - expertise and knowledge
  - communication needs

"A team is a set of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable."

– Katzenbach and Smith

Common SW team responsibilities

- Project management
- Functional management
- Designers/architects
- Developers: programmers, testers, integrators
- Lead developer (“tech lead”)

- These could be all different team members, or some members could span multiple roles.
- Key: Identify and stress roles **and** responsibilities

Organizing by functionality

- Pragmatic Programmer tip:
  "Organize around functionality, not job functions."

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

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

Making decisions

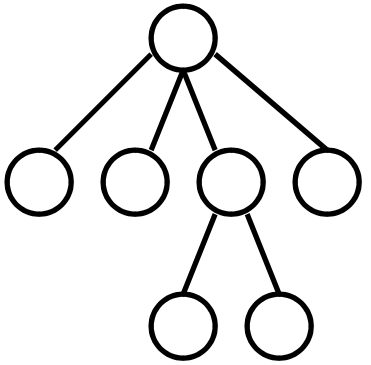
- delegate to subteams when possible
- let everyone give their input
  - even if some of it is off-track
- write down pros/cons of alternatives (weighted?)
  - evaluate cost/benefit/risks
  - How long will it take? How much to learn? Do we have to do it?
- have a clear procedure for resolving disagreement
  - strive for consensus, but if it cannot be achieved, ...
  - majority vote, and PM decides on a tie, ... etc.
- Pareto analysis: find 20% of work that solves 80% of problem
- compromise, compromise, compromise

Once decisions are made

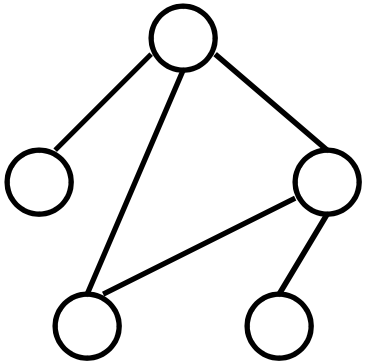
- write down decisions and responsibilities clearly
  - document by email
  - post to a shared wiki or web page
- come up with a set of steps to accomplish the decided goal
  - list specific dates that progress should be made
  - list several smaller milestones
  - agree to communicate and/or meet after 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

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)



Team leadership

- Who makes the important product-wide decisions in your team?
  - One person?
  - All, by unanimous consent?
  - Other options?...
  - Is this an unspoken or an explicit agreement among team members?

Organizing around functionality

- Pragmatic Programmer tip:
  "Organize around functionality, not job functions."

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

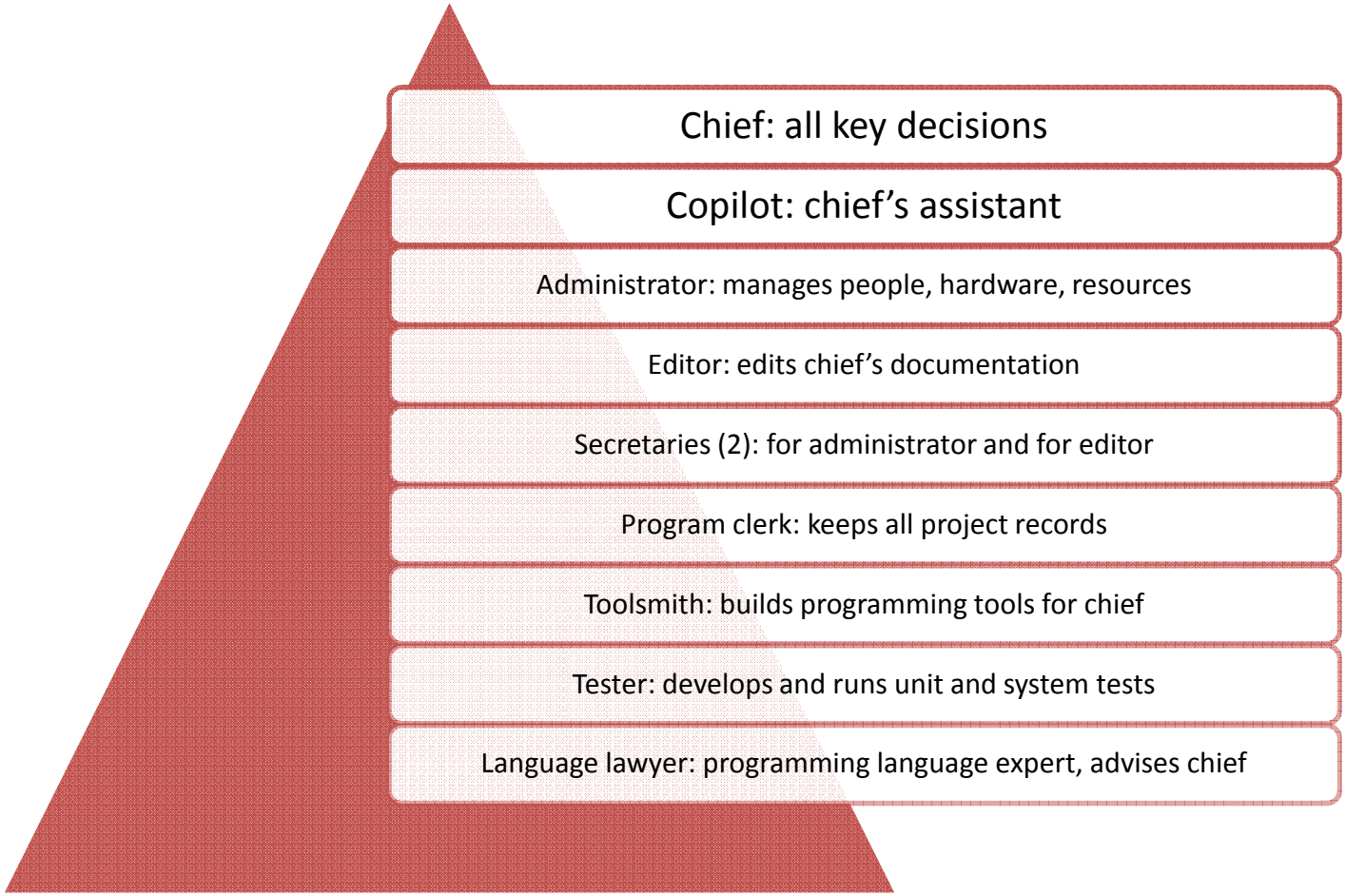
- Who will do the ...
  - scheduling? development? testing? documentation (spec, design, write-ups, presentations)? build/release preparation? inter-team communication? customer communication?

Kinds of teams

- **problem-resolution**: a focused attack on specific bugs, problems, 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
  - **skunkworks**: turn the devs loose
  - **feature**
  - **search-and-rescue**: focused on a specific problem
  - **SWAT**: skilled with a particular advanced tool(s)
  - **Professional Athletic**: carefully selected people w/ very specialized roles
  - **Theater**: "director" assigns roles to others

Surgical/Chief Programmer Team

[Baker, Mills, Brooks]



Microsoft's team structure

[microsoft.com]

- **Program Manager**. Leads the technical side of a product development team, managing and defining the functional specifications and defining how the product will work.
- **Software Design Engineer.** Codes and designs new software, often collaborating as a member of a software development team to create and build products.
- **Software Test Engineer.** Tests and critiques software to assure quality and identify potential improvement opportunities and projects.

Toshiba Software Factory [Y. Matsumoto]

- Late 1970's structure for 2,300 software developers producing real-time industrial application software systems (such as traffic control, factory automation, etc.)
- Unit Workload Order Sheets (UWOS) precisely define a software component to be built
- Assigned by project management to developers based on scope/size/skills needed
- Completed UWOS fed back into management system
- Highly measured to allow for process improvement

Results-driven structure

- Clear roles and responsibilities
  - Each person knows and is accountable for their work
- Monitor individual performance, hold people accountable
  - Who is doing what, are we getting the work done?
- Effective communication system
  - Available, credible, tracking of issues, decisions
- Fact based decisions
  - Focus on the facts, not the politics, personalities, …

Motivation

- What motivates you?
  - Achievement
  - Recognition
  - Advancement
  - Salary
  - Possibility for growth
  - Interpersonal relationships
    - Subordinate
    - Superior
    - Peer
  - Status
  - Technical supervision opportunities

- Company policies
- Work itself
- Work conditions
- Personal life
- Job security
- Responsibility
- Competition
- Time pressure
- Tangible goals
- Social responsibility

Other?

De-motivators

- What takes away your motivation?
  - Micro-management or no management
  - Lack of ownership
  - Lack of effective reward structure
    - Including lack of simple appreciation for job well done
  - Excessive pressure and resulting "burnout"
  - Allowing "broken windows" to persist
  - Lack of focus in the overall direction
  - Productivity barriers
    - Asking too much; not allowing sufficient learning time; using the wrong tools
  - Too little challenge
  - Work not aligned with personal interests and goals
  - Poor communication inside the team

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

How to run an effective meeting

- have an agenda of topics to discuss (email this ahead of time)
- each member/subgroup reports its progres
- get an update on every current work item
- take notes on decisions made and post them to wiki, etc.
- 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

Meeting gotchas

- be professional: don't tolerate lateness, lack of forus
  - 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, stop 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)