Team dynamics
CSE 403

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

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
• Developers: programmers, testers, integrators
• Lead developer/architect ("tech lead")

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

Questions when organizing your team

• 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?
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 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:

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 dev's 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**]

- **Chief**: all key decisions
- **Coffler**: chief's assistant
- **Administrator**: manages people, bonuses, resources
- **Editor**: edits chief's documentation
- **Secretaries**: for administrator and for editor
- **Programmer**: keeps all project around
- **Toolsmith**: builds programming tools for chief
- **Testers**: develop and run unit and system tests
- **Language/luke**: programmer and language expert, additional chief
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.

Common factors in good teams

• 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")
• 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?

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, …

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