Lecture 22: Managing People

Organizational Structures

Building high Performance teams

Discussion:

How to make team assignments work in undergraduate courses?

Starting point

You have a project

You have been given a team

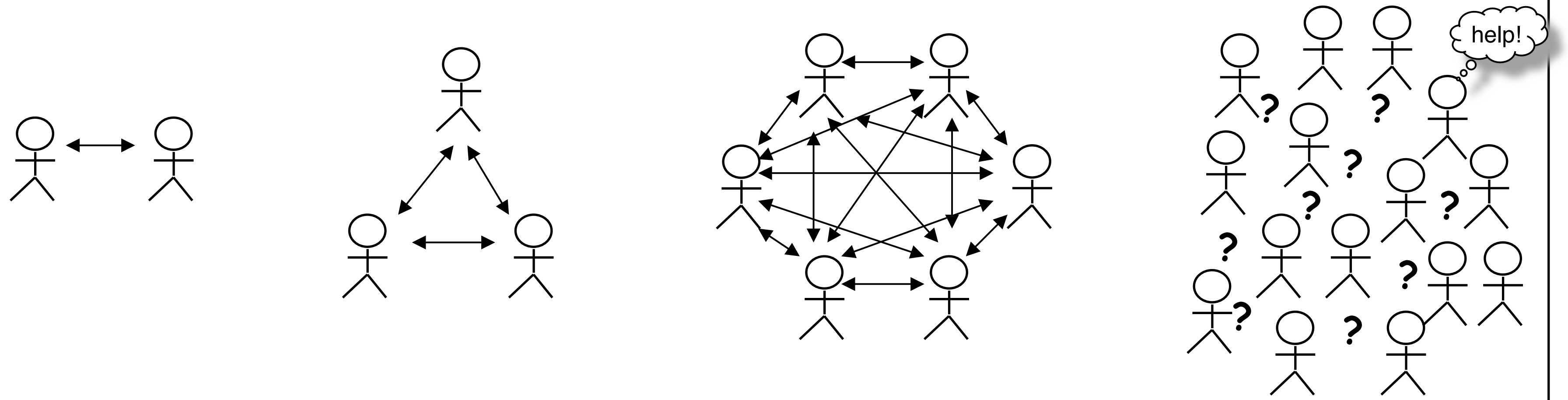
- a mixed set of skills
- a mixed set of motivations

Problem:

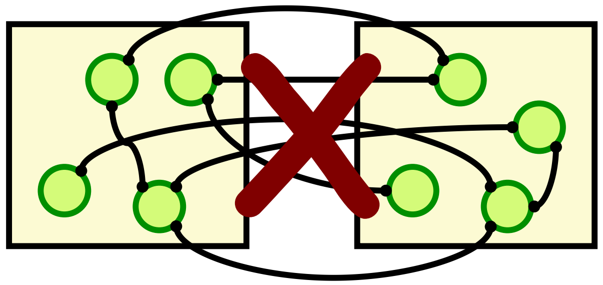
- How do you get everyone to work together?
- …and get the job done?

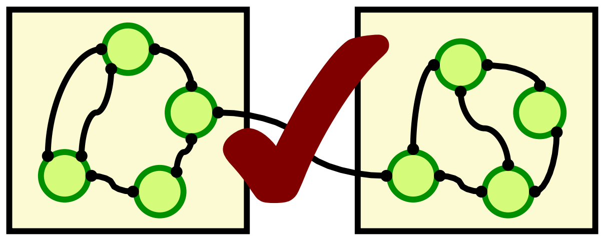
Scaling up…

Communication overhead is exponential

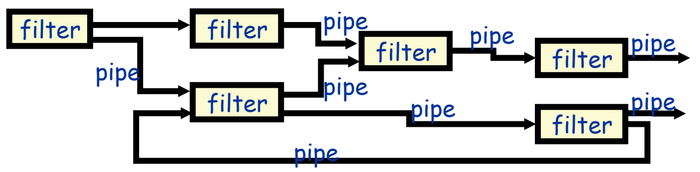


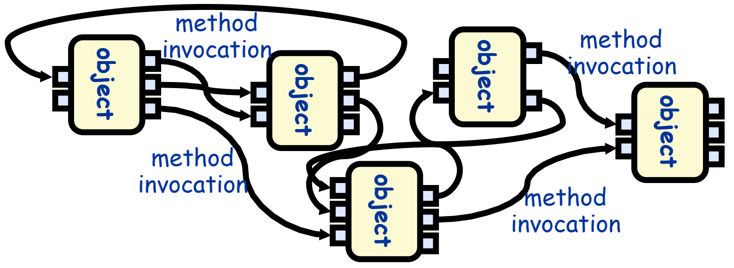
Exploit Modularity:





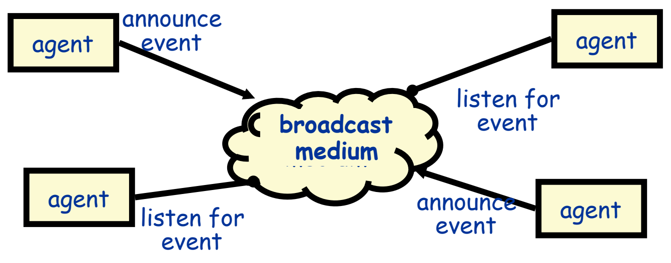
Team Organization

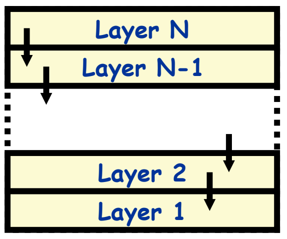


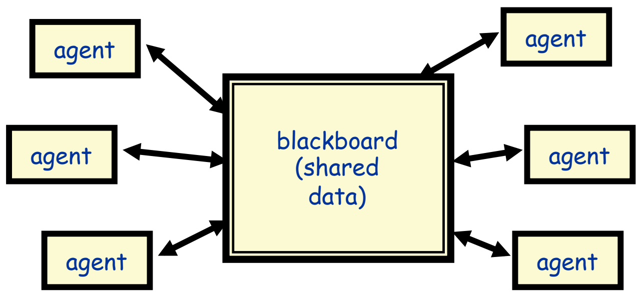


Conway's Law:

The structure of the software reflects the structure of the organisation that built it







Coordination Mechanisms

Direct supervision

- simple structure - little formalization

Standardization of work processes

- "machine bureaucracy"
- e.g. mass production and assembly

Standardization of work outputs

- "divisionalized form"
- e.g. each division has performance targets

Standardization of worker skills

- "professional bureaucracy"
- e.g. hospitals, law firms,…

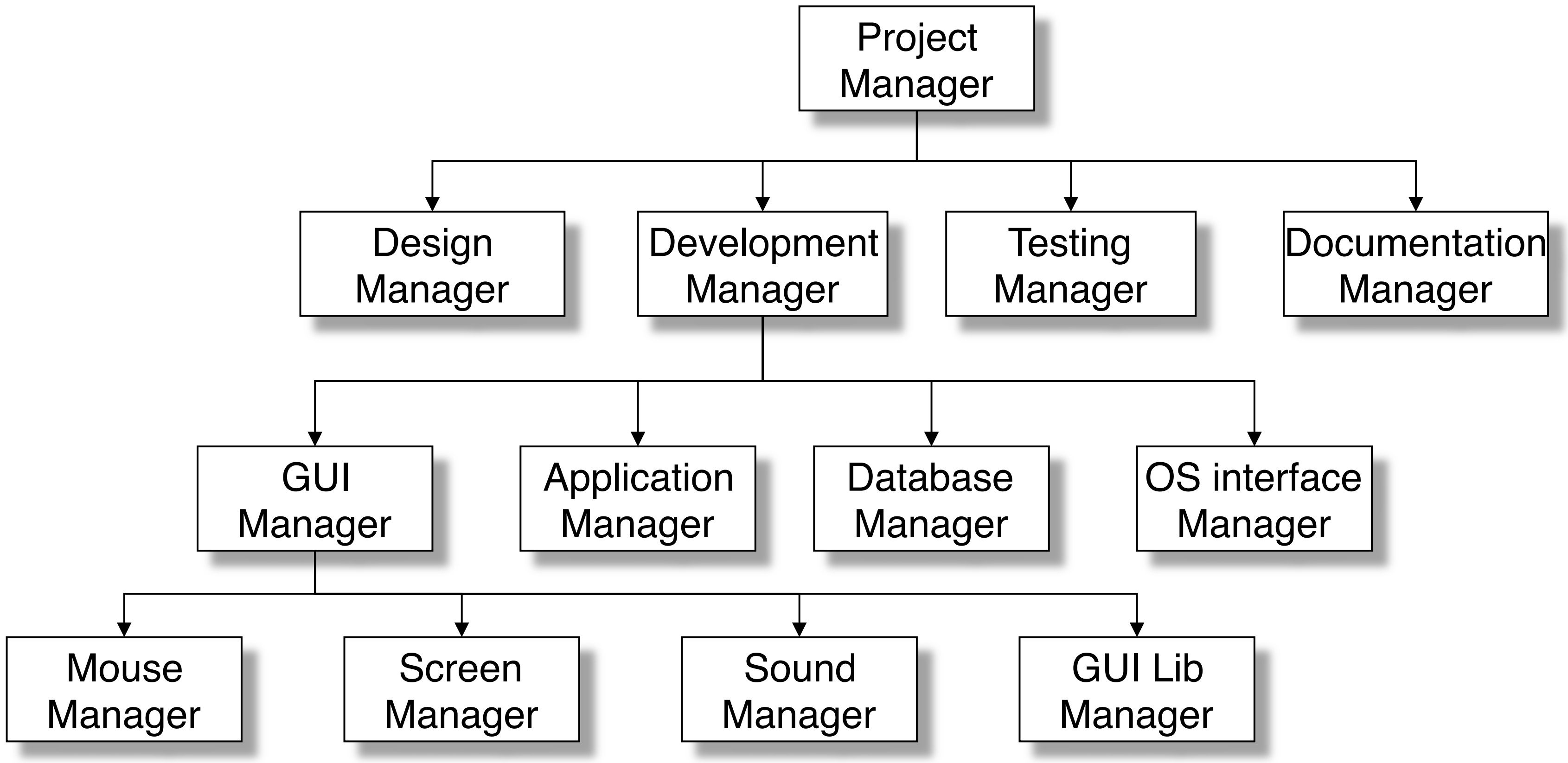
Mutual adjustment

- "adhocracy"
- e.g. skunkworks, high innovation, open source teams

Hierarchical Teams

Team structure = top down decomposition

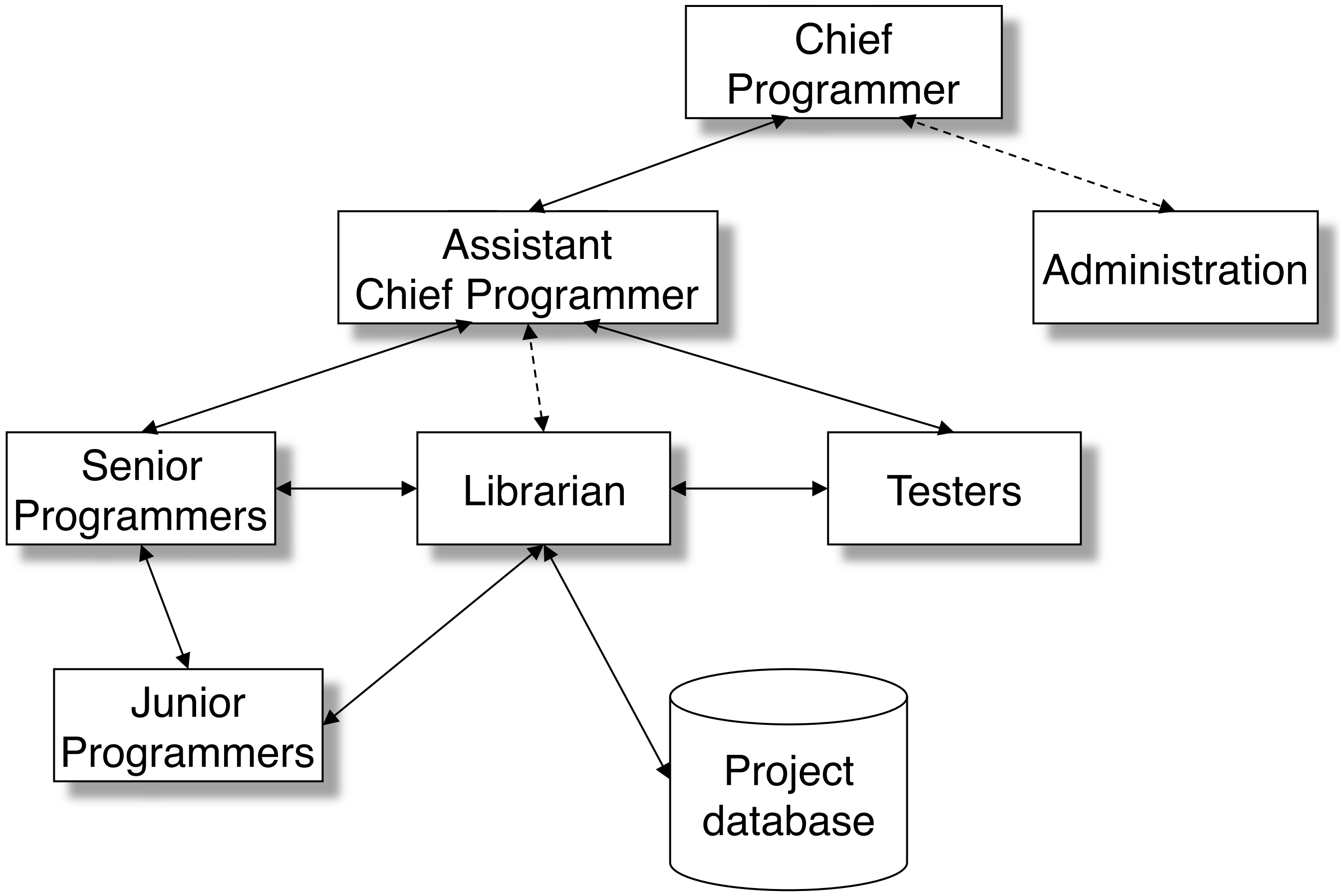
Disadvantage: vertical communication is ineffective



Chief Programmer Teams

Based on hospital surgical teams

Chief programmer is not a manager - concentrates on technical issues



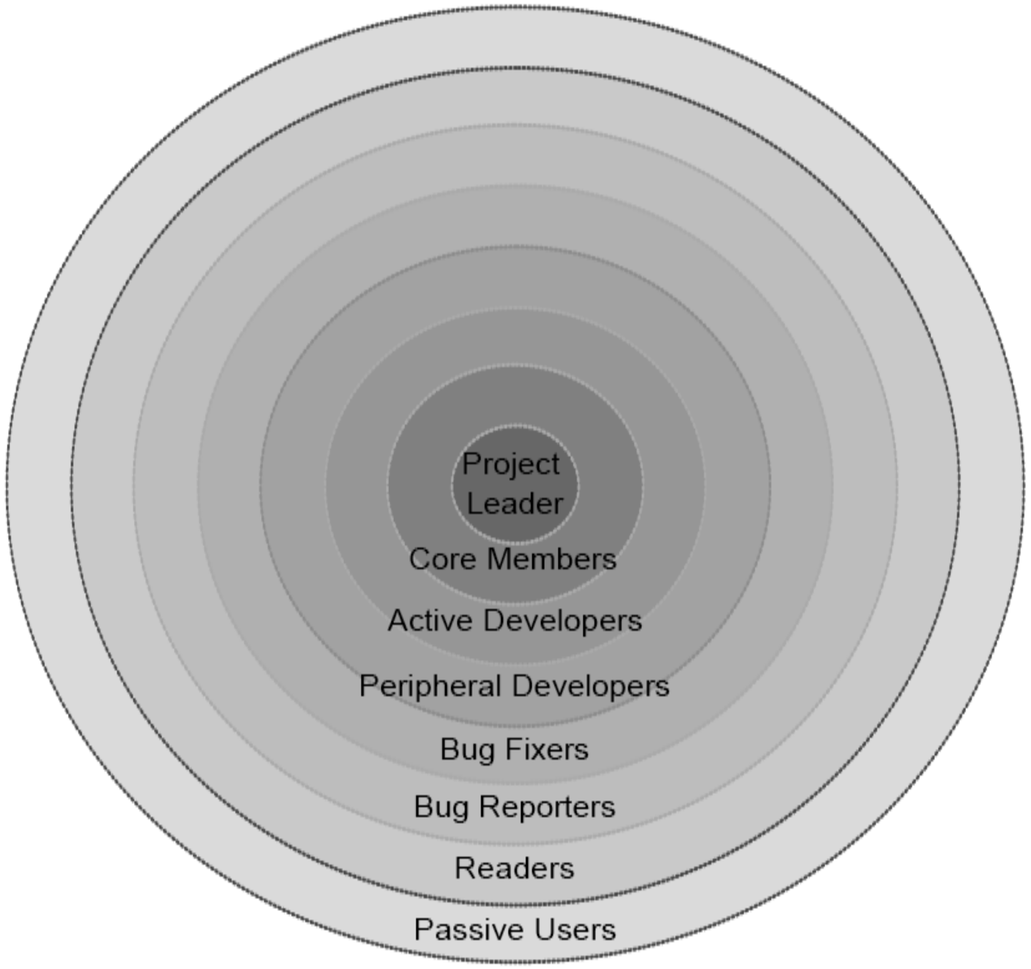
Matrix Organization

Identify specific skill sets

- Assign people to projects according to needed skills
- People work on multiple projects

	real-time programming	graphics	data-bases	QA	Testing
Project 1	X			X	X
Project 2	X		X	X	X
Project 3		X	X	X	X

Open Source - Onion Model



General Principles

Use fewer, better people

- Performance of best programmers better by an order of magnitude!

Fit tasks to capabilities and motivations of people

Help people to get the most out of themselves

- opportunity to accept new challenges and be rewarded

Balance the team

- E.g. team players vs. star performers
- Practice “egoless” programming

Remove people who do not fit the team

High Performance Teams

Nurture a team culture

- A team is not a family
- Team members help one another, but don't tolerate freeloaders

Instill the right values

- Discuss examples
- Reward people who uphold the team values

Build trust

- All feedback should be constructive
- Foster lively & healthy debate about issues and risks

Effective Communication

- Use face-to-face whenever possible
- Use phone or F2F to resolve email debates
- Get everyone using IM
- Encourage social events for the team
- Physical layout of office space is important

Organizational Clarity

Things every team member should know:

- What is the mission of the team?
- What is the vision for the system to be delivered?
- How will you measure team success?
- Who are the project stakeholders?
- How will you measure project success?
- Who is responsible for what?
- What procedures should you follow to do the work?

Who can change the code?

Collective Ownership

- Anyone can change any code or model
- Works well for small teams
- Promotes shared responsibility
- (Needs good version management tools)

Change Control

- Each sub-team can only change their subsystem
- Reduces unexpected problems when code changed by others
- Promotes development of expertise
- More important on larger projects