# CS 4530: Fundamentals of Software Engineering
# Lesson 6.3: Teams

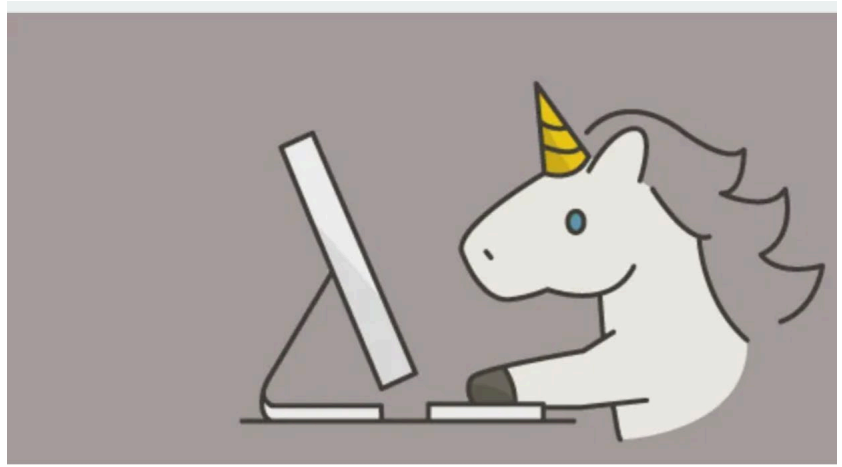
Adeel Bhutta, Jan Vitek and Mitch Wand

Khoury College of Computer Sciences

# Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Explain key advantages of working in a team and sharing information with your team
  - Describe the HRT pillars of social interaction
  - Understand why agile processes favor small teams
  - Apply root-cause analysis to construct a blameless post-mortem of a team project

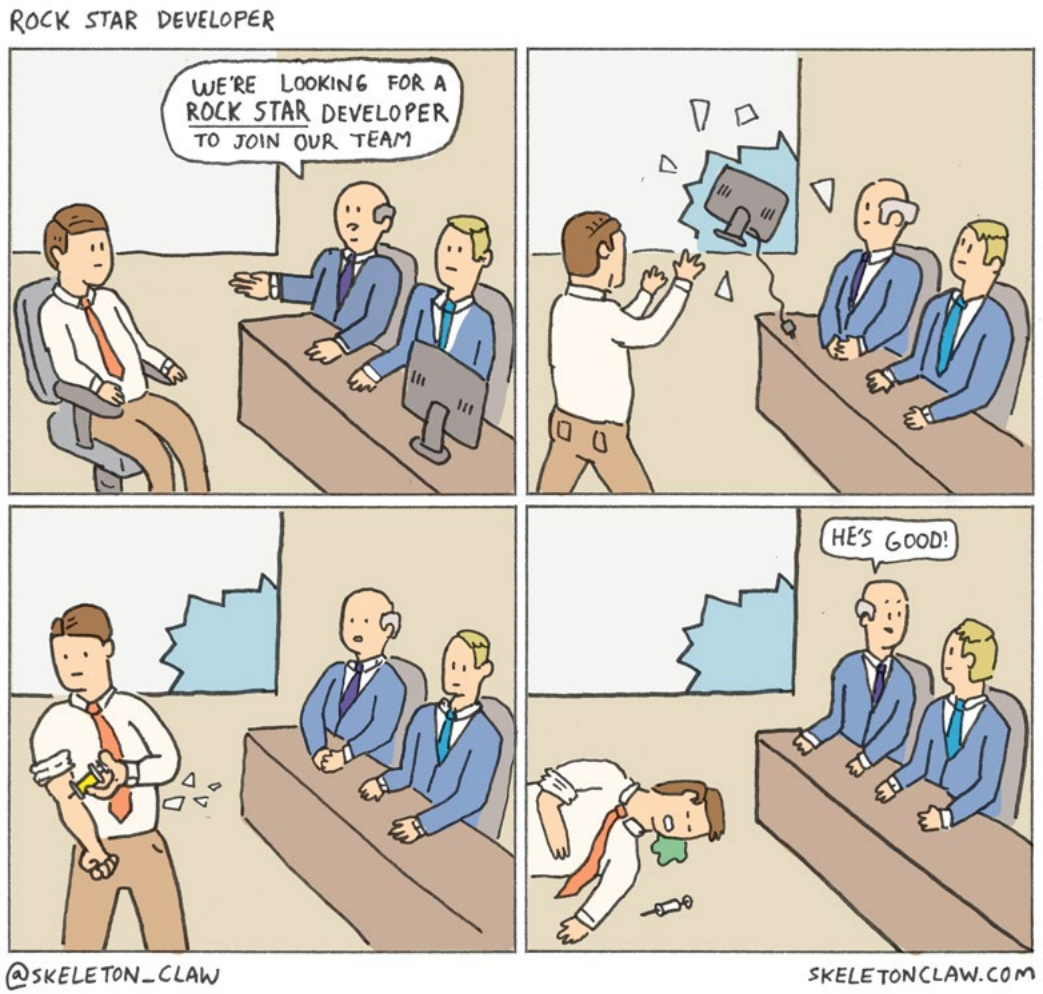
# Why Teams? “The 10x Engineer”



**What makes a 10x Developer?**

#10xdeveloper #productivity #beginners #career

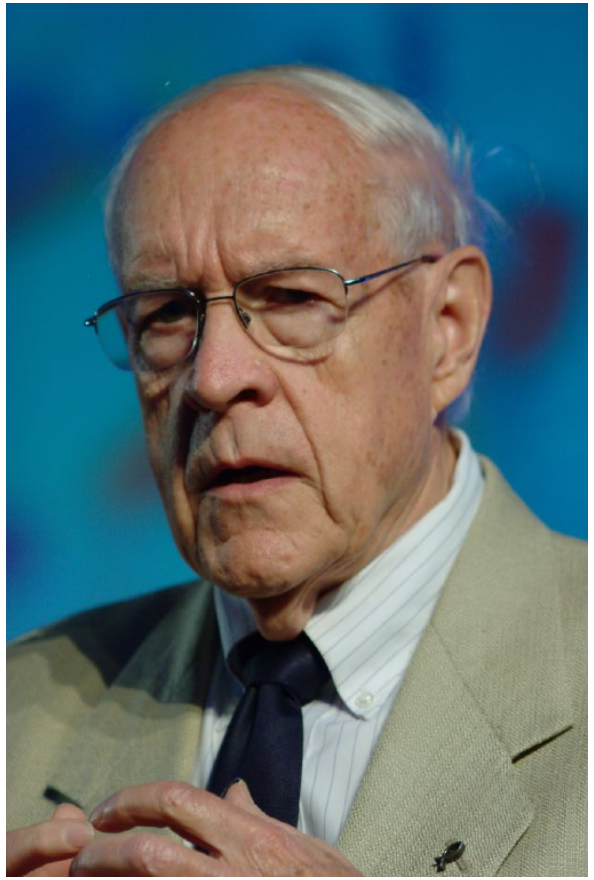
Davide de Paolis Mar 11, 2019 · 6 min read



# Teams are hard: Brooks' Law

"Adding manpower to a late software project makes it later"

Fred Brooks, 1975



# What goes wrong with teams in software development?

- How do you structure teams effectively?
- How do you encourage team-members to treat each other well?
- How do you encourage teams to share knowledge and collaborate?
- How do you respond to failures?

# How do we structure teams efficiently?

- Examining Brooks' Law: "Adding manpower to a late software project makes it later"
- How many communication links are needed to finish a task?

# Agile Favors “Two-Pizza” Teams

Q: How many people on a team?

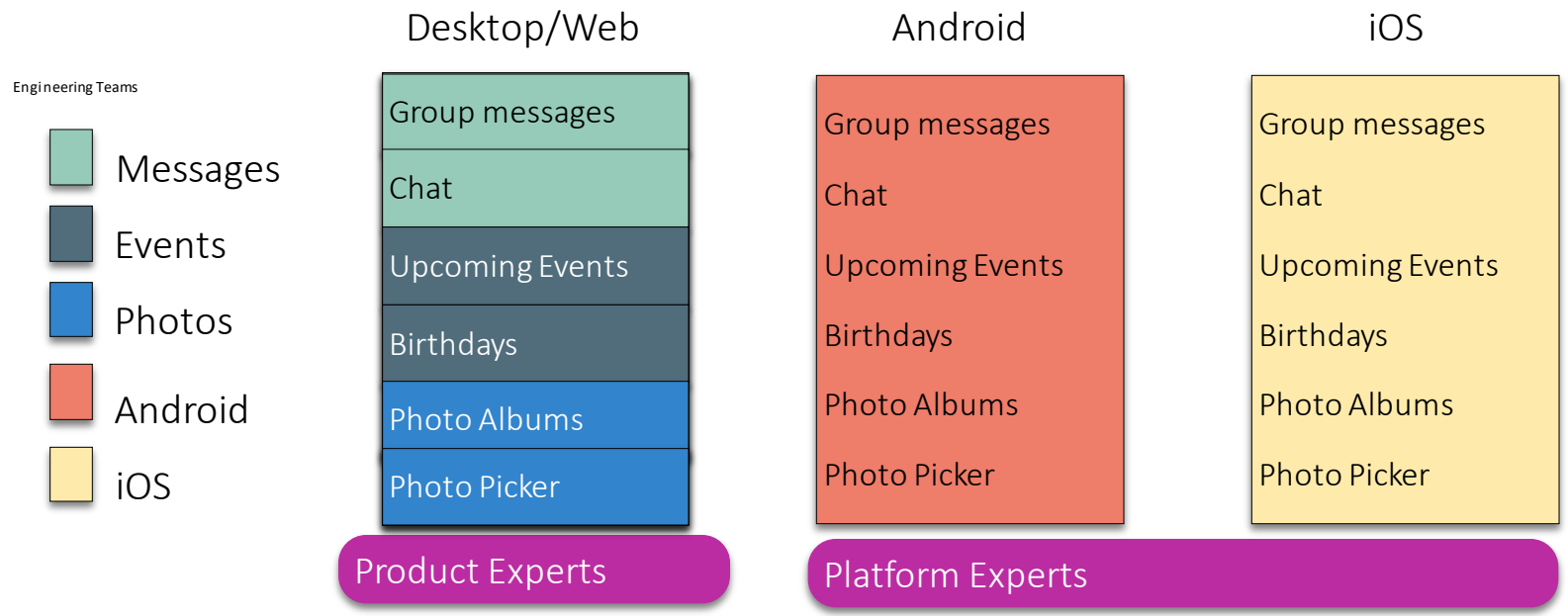
A: “No more than you could feed with two pizzas”

Rationale:

- Decrease communication burdens
- Focus conversations to relevant topics

# Agile Favors “Product” teams, not ”Platform” teams

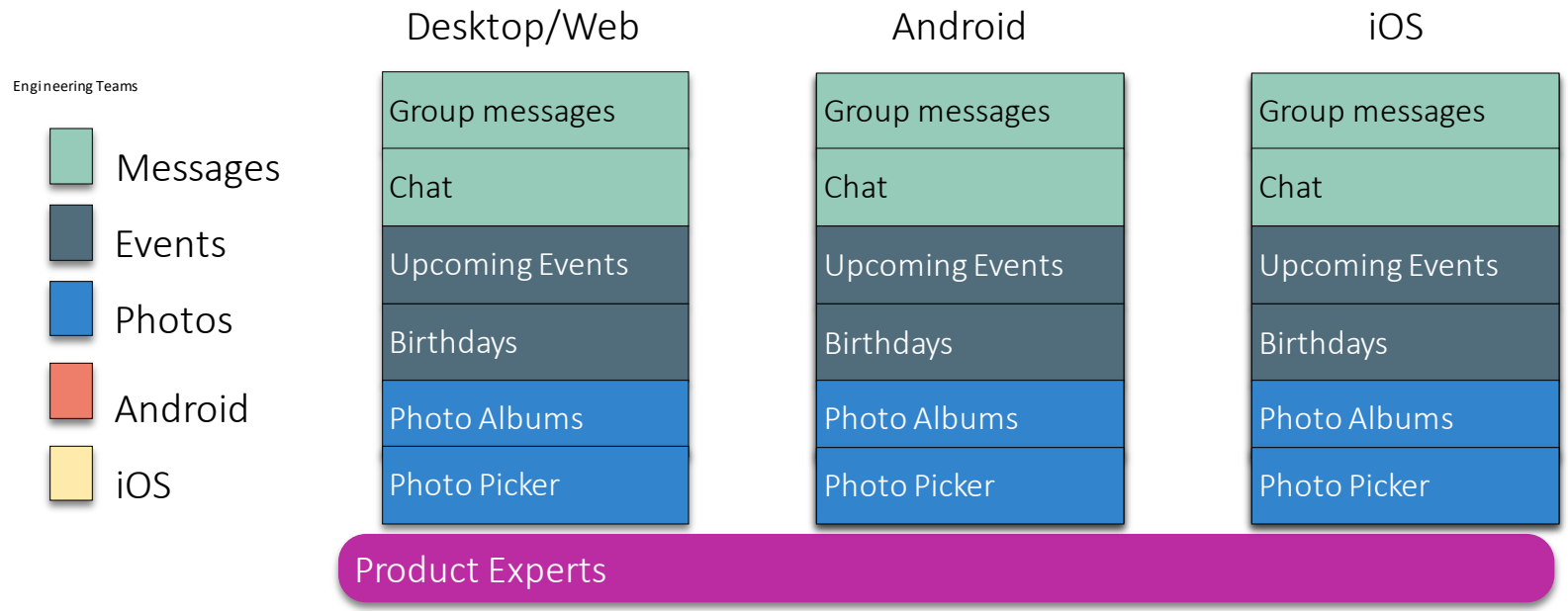
Example: Facebook mobile teams (with platform organization)




https://www.youtube.com/watch?v=Nffzkkdq7GM

# Agile Favors “Product” teams, not ”Platform” teams

Example: Facebook mobile teams (with platform organization)



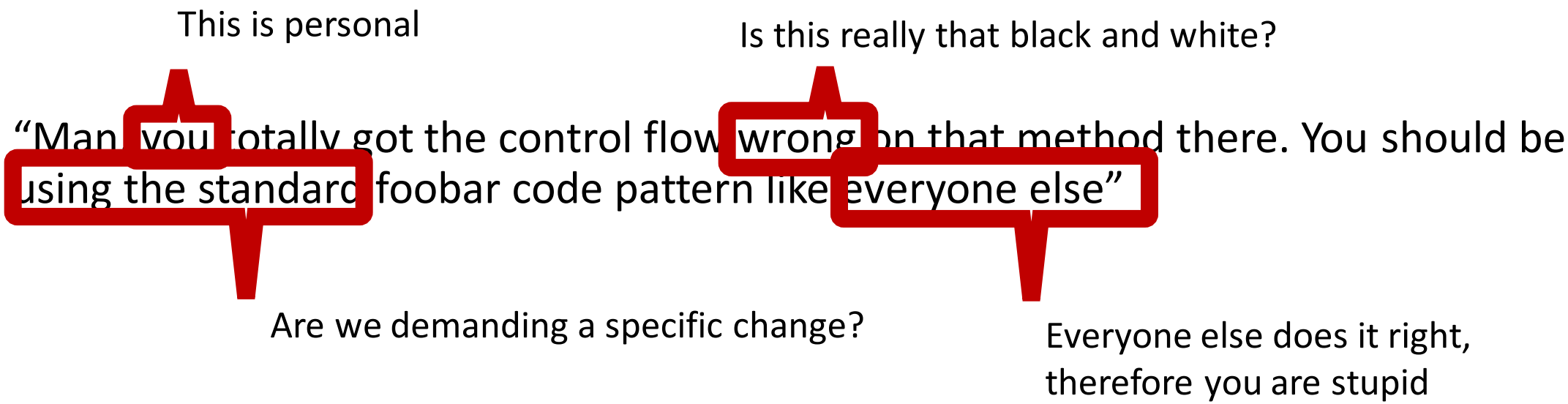

https://www.youtube.com/watch?v=Nffzkkdq7GM

# Encourage team members to treat others well? Three Pillars of Social Skills

- Pillar 1: **Humility**: You are not the center of the universe (nor is your code!). You're neither omniscient nor infallible. You're open to self-improvement.
- Pillar 2: **Respect**: You genuinely care about others you work with. You treat them kindly and appreciate their abilities and accomplishments.
- Pillar 3: **Trust**: You believe others are competent and will do the right thing, and you're OK with letting them drive when appropriate.

From "Debugging Teams" by Ben Collins-Sussman and Brian Fitzpatrick

# HRT Example: Code Review




From “Debugging Teams” by Ben Collins-Sussman and Brian Fitzpatrick

# HRT Example: Code Review

“Man, you totally got the control flow wrong on that method there. You should be using the standard foobar code pattern like everyone else”

‘Hmm, I’m confused by the control flow in this section here. I wonder if the foobar code pattern might make this clearer and easier to maintain?

Humility! This is about *me,* not you

From “Debugging Teams” by Ben Collins-Sussman and Brian Fitzpatrick

# Scaling Communication

- Knowledge sharing needs to scale linearly (or sub linearly) with org growth:
  - Mentorship
  - Q&A
  - Mailing lists
  - Tech talks
  - Documentation

# Bus Factor & Importance of Information Sharing


Ensure the future of your work!

Consider inviting another GitHub user to be your successor.

Inviting a successor helps ensure the continuity of your work in case you are unable to access your account. Learn more

Invite a successor

# Responding to Failures

In software, in humans, and in processes.

How do we learn:

- What went well?
- What went wrong?
- Where we got lucky?
- How do we prevent it from happening again?

# Blameless Post-Mortems

- What actions did you take at the time?
- What effects did you observe at the time?
- What were the expectations that you had?
- What assumptions did you make?
- What is your understanding of the timeline of events as they occurred?

# How Not to Respond to Failures

1. Some engineer contributes to failure or incident
2. Engineer is punished/shamed/blamed/retrained
3. Engineers as a whole become silent on details to management to avoid being scapegoated
4. Management becomes less informed about what actually is happening, do not actually find/fix root causes of incidents
5. Process repeats, amplifying every time

# Conducting Postmortems

- Apply this technique after any event you would like to avoid in the future
- Apply this to technical and non-technical events
- Focus on improvement, resilience, and collaboration: what could any of the actors have done better?
- Google's generic postmortem template

# Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Explain key advantages of working in a team and sharing information with your team
  - Describe the HRT pillars of social interaction
  - Understand why agile processes favor small teams
  - Apply root-cause analysis to construct a blameless post-mortem of a team project