

# Software Best Practices in Practice

Anjali Pal  
anjali@cs.washington.edu

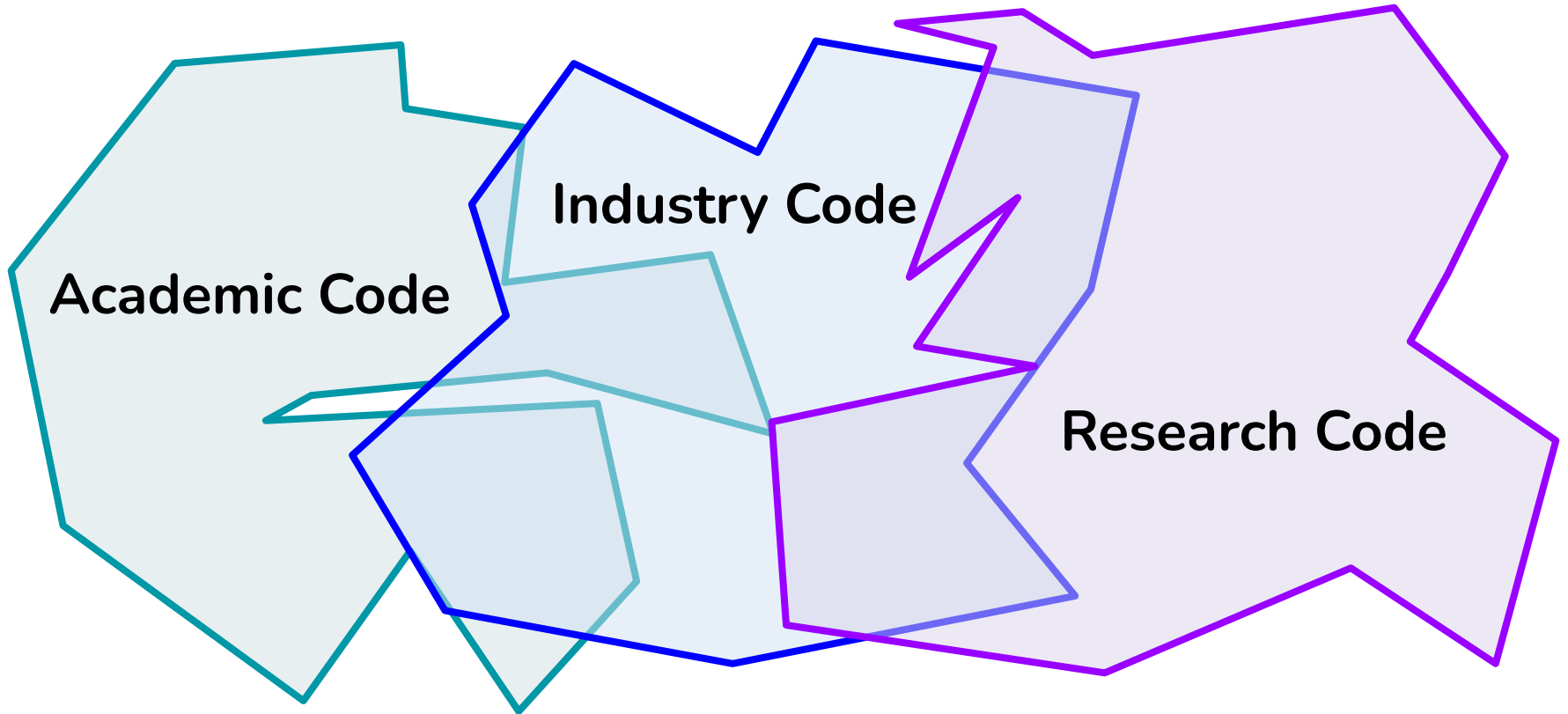
The talk I thought I was going to give:

**Academic Code**

**Industry Code**

**Research Code**

The talk I'm actually going to give:




# Guiding questions

- Who writes the code?
- Who maintains the code?
- Who uses the code?
- Who decides what the code should do?
- Who is impacted by bugs or incorrect code?

Software development is  
fundamentally about people,  
not about code

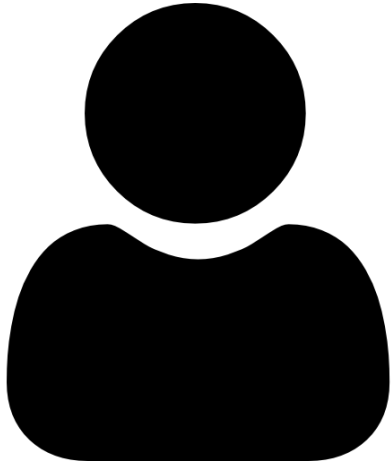
Software development is  
fundamentally about people,  
not about code



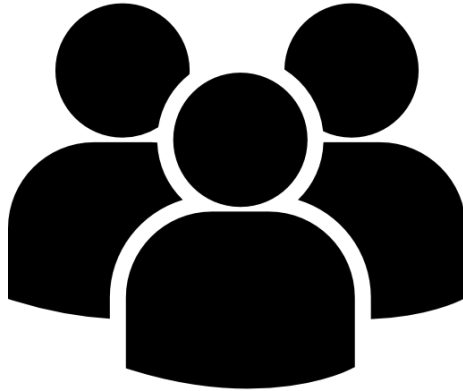
So our development  
practices should reflect this!

Who writes the code?

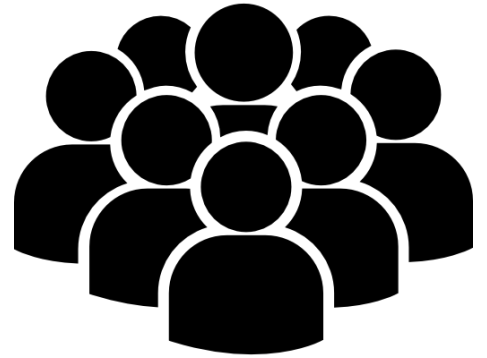
Who writes the code?



Solo Developer



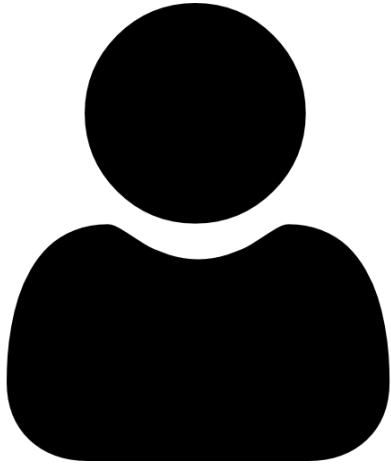
Small Team



Large Team



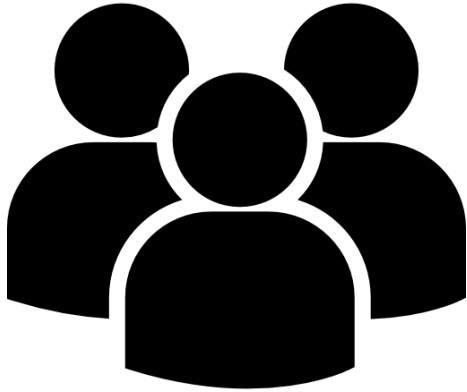
# Who writes the code?



Solo Developer

- Probably no real version control
- No code review
- Minimal tests
- Minimal comments
- Fairly small codebase

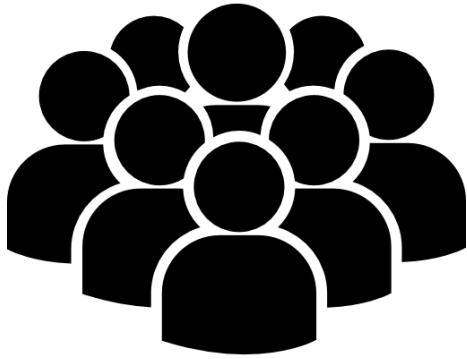
# Who writes the code?



Small Team

- Version control
- Small commits (or painful merge conflicts)
- Code review
- Style guide
- Tests

# Who writes the code?

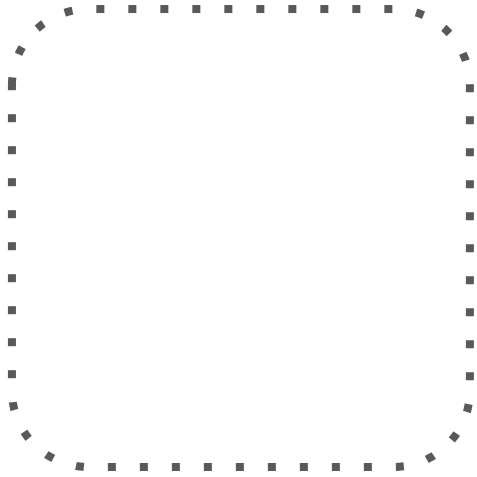


Large Team

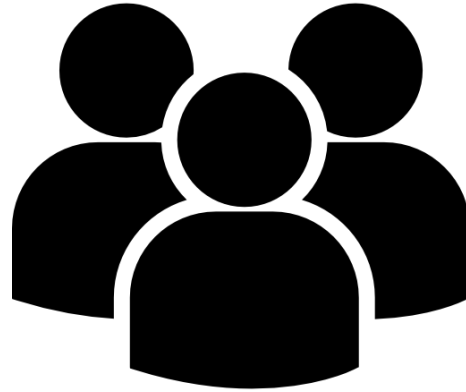
- Clear lines of ownership between teams
- Large, potentially old codebase
- No one knows everything about the codebase
- Constant turnover

Who maintains the code?

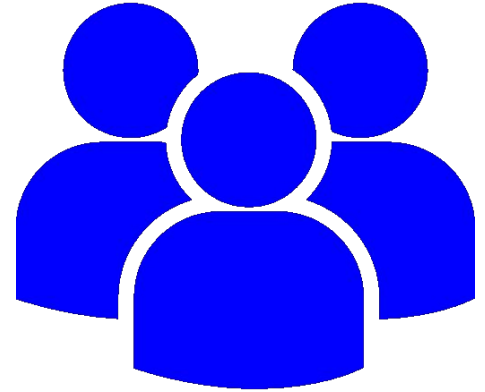
# Who maintains the code?



No One

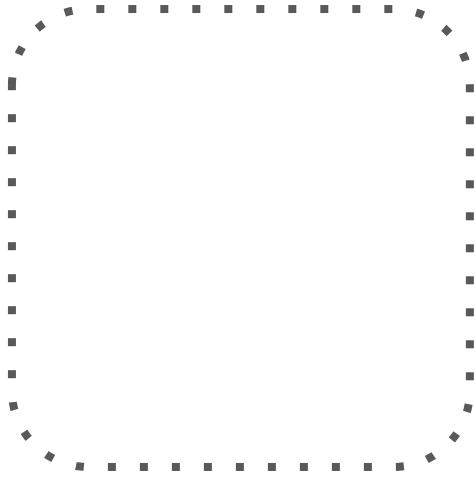


Same Team



Different Team

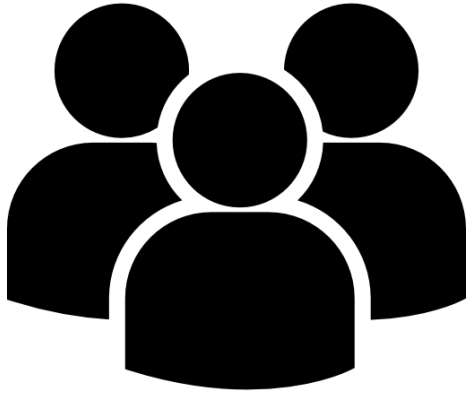
# Who maintains the code?



No One

- Lots of code is forgotten shortly after its written
- “Good design” less important
- It’s okay to optimize for speed/ease over long-term maintenance sometimes

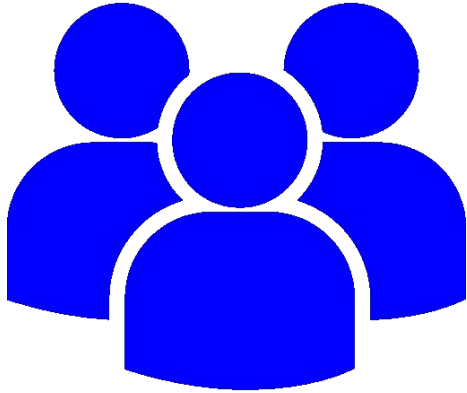
# Who maintains the code?



Same Team

- Testing pipeline
- Code review spreads expertise
- Easier to maintain code you know
- Consistent work keeps code fresh in mind
- Older code is harder to maintain

# Who maintains the code?




Different Team

- Code doesn't capture **why** decisions were made
- If decisions weren't documented and the people who made them are no longer involved, you are likely to repeat some mistakes
- Or worse, be afraid to make changes



Who uses the code?

# Who uses the code?




Software Package  
Command Line Tool  
API

Developers



Website  
App

“Normal Users”



Specialized  
Programs

“Expert Users”

# Who uses the code?

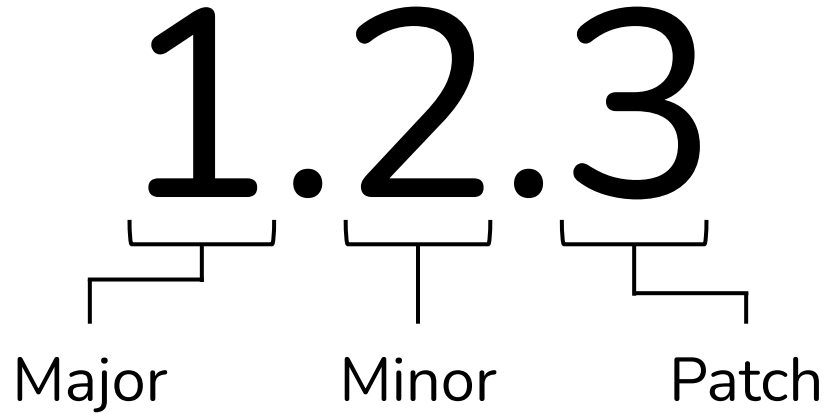
Software Package  
Command Line Tool  
API

Developers

- Clear APIs
- Good documentation
- Easy to install + get started
- Open source

# How do users get new versions of the code?

Versioned Releases



# Who uses the code?



Website

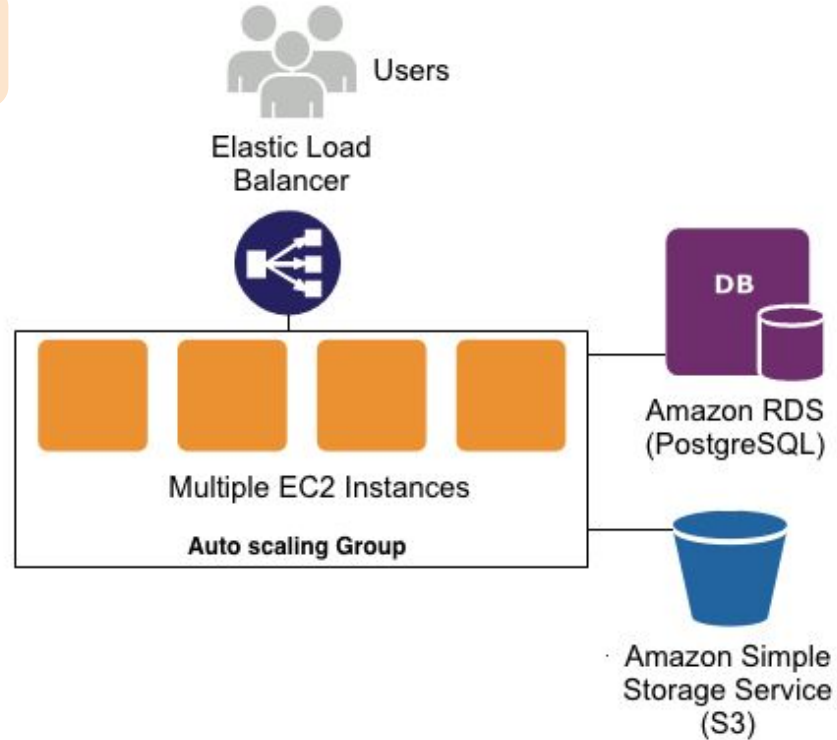
App

“Normal Users”

- UI/UX is extremely important, and often undervalued
- Accessibility should be built in from the beginning, not added later
- Lots of things to consider for a large, diverse user base: time zones, translation, device sharing, privacy, etc

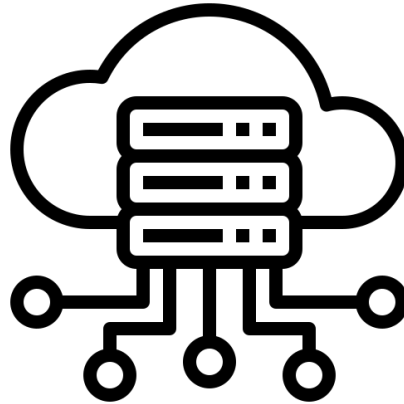
# How do users get new versions of the code?

Deployed Services



# How do users get new versions of the code?

Deployed Services



# Who uses the code?

Specialized  
Programs

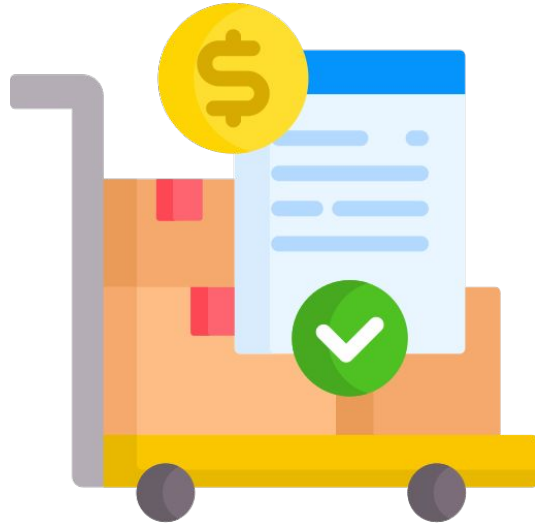
“Expert Users”

- Hard to make changes
  - Government contracts
  - In-depth user training
- Expert/power users have different UI/UX goals than average users



# How do users get new versions of the code?

Specialized Programs



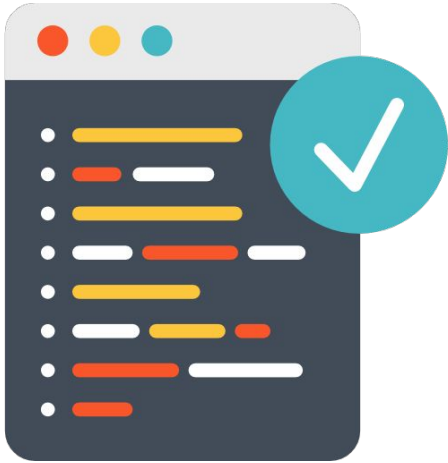
Who decides what the code should do?

Who decides what the code should do?



Who is impacted by  
incorrect code?

# Who is impacted by incorrect code?



Class Assignment

- Bad grade

# Who is impacted by incorrect code?



User-facing  
Product

- Site down (company loses money)
  - Impact depends on what people rely on the software for
- User data loss
- Leak sensitive information about customers

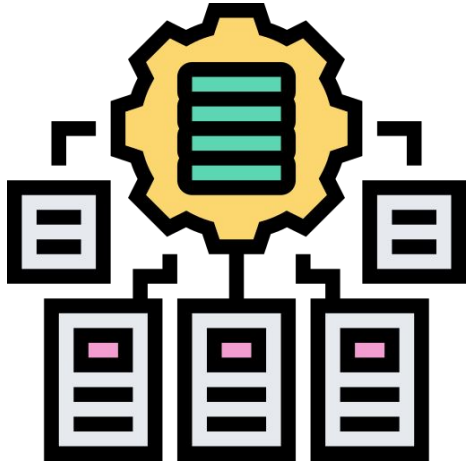
# Who is impacted by incorrect code?



Healthcare  
Software

- Miss important signal in patient heart rate, blood pressure, etc
- Too much or not enough medicine
- People could die

# Who is impacted by incorrect code?



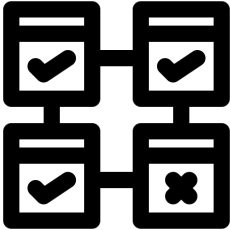
Low-level code

- Widespread implications because this code underlies pretty much everything
- Security vulnerabilities especially scary



# Who is impacted by incorrect code?

Testing



Verification

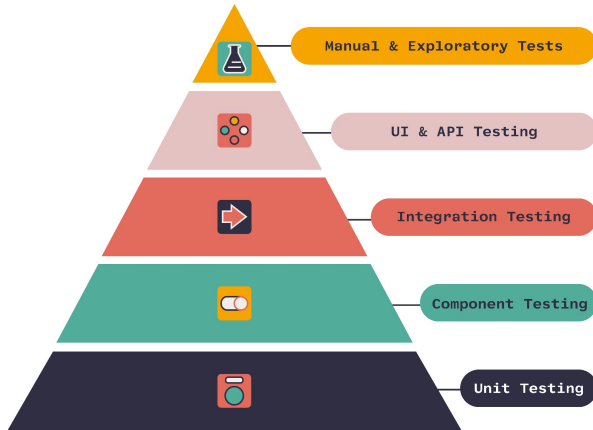


Lower

**Correctness  
Guarantees**

Higher

# Testing



Testing

- Ensures that the code does the right thing **on a specific input**
- Code coverage = Percentage of code executed while running tests
- Good testing can increase **confidence** in the system and **prevent regression**

# Testing

Program testing can be used very effectively to show the presence of bugs but never to show their absence

- Edsger W. Dijkstra

# Verification

How can we **prove** that our program has some property?

# Hoare Logic

$$\{ P \} C \{ R \}$$


Tony Hoare

If **P** is true **before** execution...

And **C** terminates...

Then **R** will be true **after** execution

# Hoare Logic

{ P } C { R }

{ x = 2 } skip

{ x = 2 }

{ x = 2 } y = x + 1

{ x = 2; y = 3 }

{ x = 1 } if x > 0 then y = 1 else y = 2

{ x = 1; y = 1 }

# Hoare Logic

$$\frac{}{\{P\} \text{ skip } \{P\}} \text{ skip}$$

$$\frac{\{P\} S_1 \{Q\} \quad \{Q\} S_2 \{R\}}{\{P\} S_1; S_2 \{R\}} \text{ Seq}$$

$$\frac{}{\{True\} \text{ abort } \{False\}} \text{ abort}$$

$$\frac{\{P \wedge e\} c_1 \{Q\} \quad \{P \wedge \neg e\} c_2 \{Q\}}{\{P\} \text{ if } e \text{ then } c_1 \text{ else } c_2 \{Q\}} \text{ If}$$

$$\frac{}{\{P[x \mapsto E]\} x := E \{P\}} \text{ assign}$$

$$\frac{\{I \wedge e \neq 0\} c \{I\}}{\{I\} \text{ while } e \text{ c } \{I \wedge e = 0\}} \text{ While}$$

System for **proving**  
properties of programs

$$\frac{P \implies A \quad \{A\} c \{B\} \quad B \implies Q}{\{P\} c \{Q\}} \text{ Consequence}$$



**Why am I writing this code?**

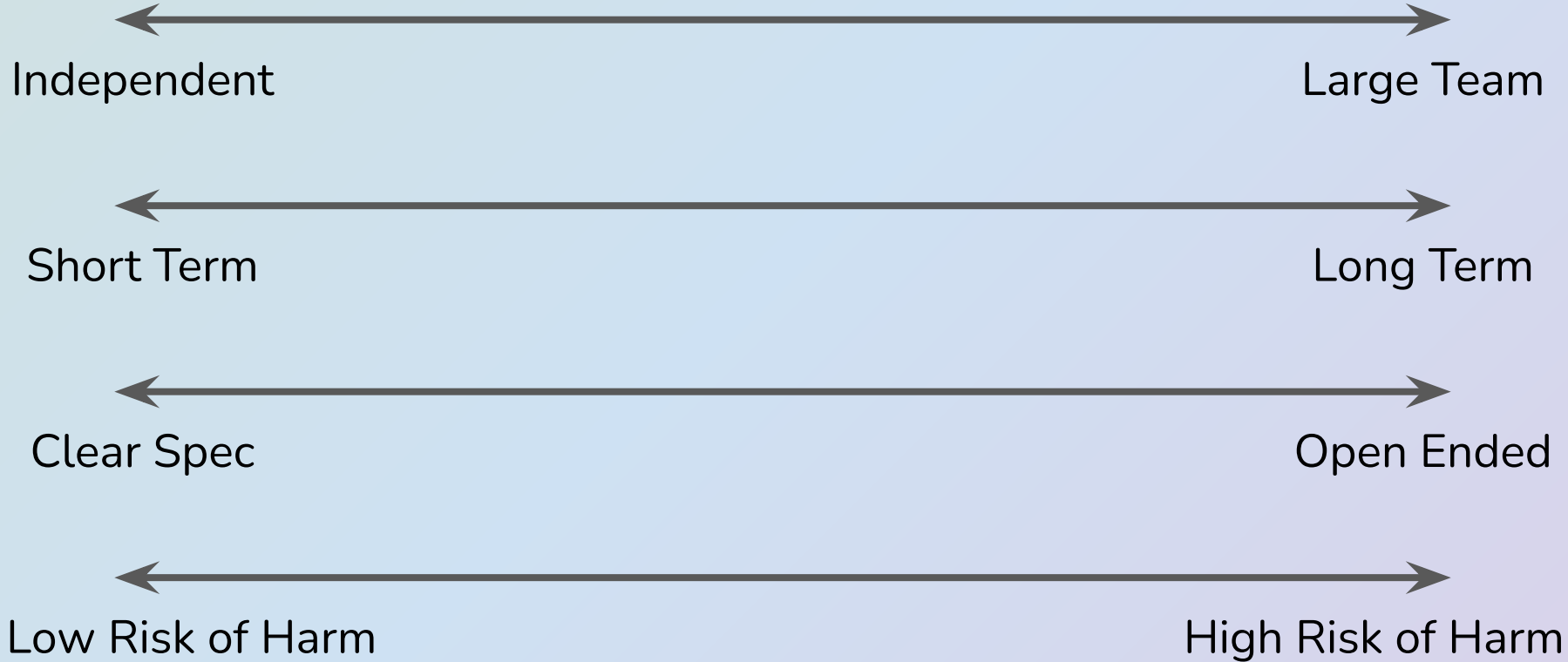
**Who am I writing this code with?**

**Who am I writing this code for?**

**What is the long-term plan for the code?**

**Who could get hurt if the code goes wrong?**





# Questions?

Anjali Pal

[anjali@cs.washington.edu](mailto:anjali@cs.washington.edu)