# Software Engineering
# The Job

COMPSCI 308 – 2024-04-09

Luke Moffett, Duke CS

Duke

# Big Picture
# Every Job will be Unique

Everything in these slides is meant to give you a flavor of what to consider about places of employment.

Turquoise – Likely to have encountered in your projects

Purple – Would likely see in an internship

Orange – Would likely only see when employed

Duke

# What is the Job?

To build software that helps a business/government/non-profit achieve its goals.

Duke

# “The Best Job in America”

- Work is intellectually stimulating
- Colleagues are largely competent
- You build things that actually run at the end of the day
- Each day and task is different
- Culture of trying new things
- Constant Learning
- **Excellent** Pay
- Flexible work schedules and situations
- Many sources of job security
- Meritocratic culture
- Space for creativity
- Geeks are cool now, kind of


https://xkcd.com/303/

Duke

# Work Environments

## Start-Ups

- High Freedom
- Small Tech Footprints
- Uncertainty with Upside
- “Many Hats”
- Management Tends to be Technical
- Developing New Products/Services
- Peer Abilities Vary

## Tech Company

- Sophisticated Platforms
- Internal Standards and Bespoke Tools
- High Specialization
- Maintaining/Expanding Existing Products/Services
- Excellent Compensation
- High Skill Peers/Hard to Stand-Out

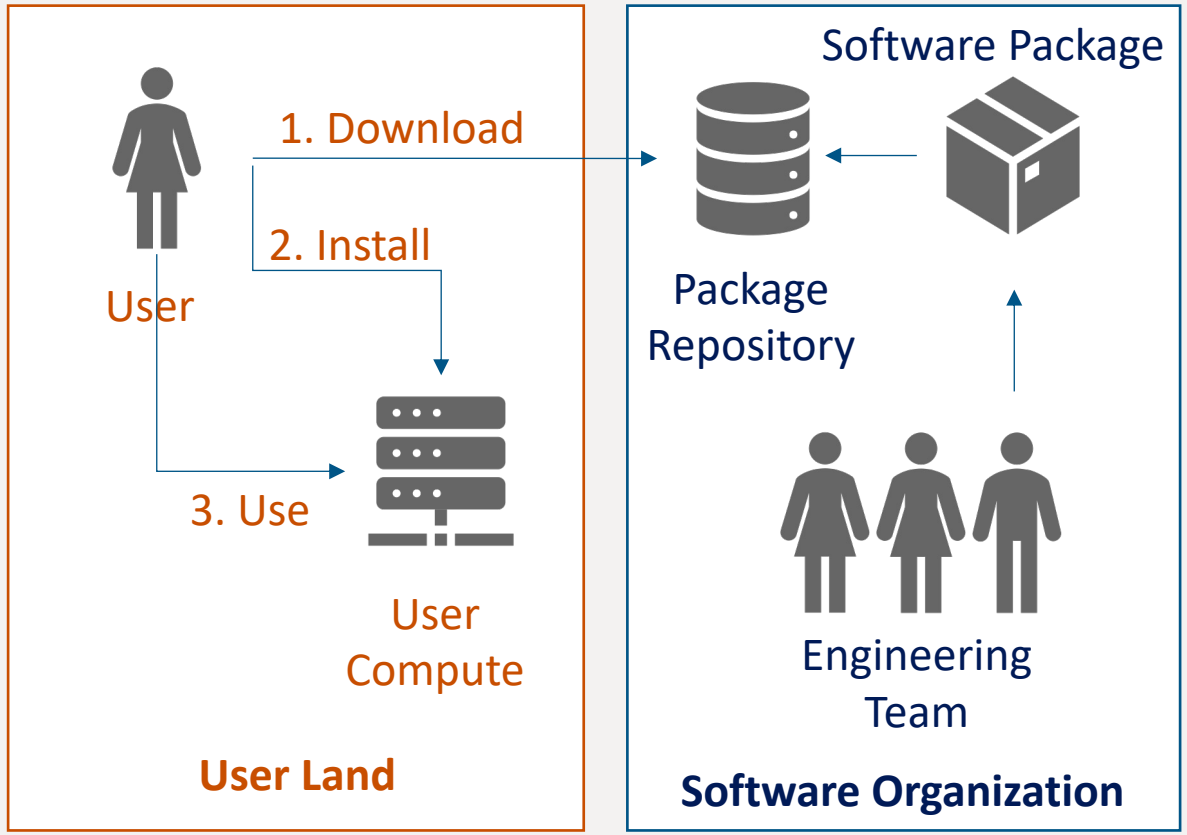
## Big Corp/Gov’t/Non-Profit

- Stability
- Technology S***upports*** Mission (“Cost Center”) – Best Engineers Understand Both
- Integrating or Customizing Tools from Tech Companies and Start-Ups
- Every Industry
- Peer Abilities Vary

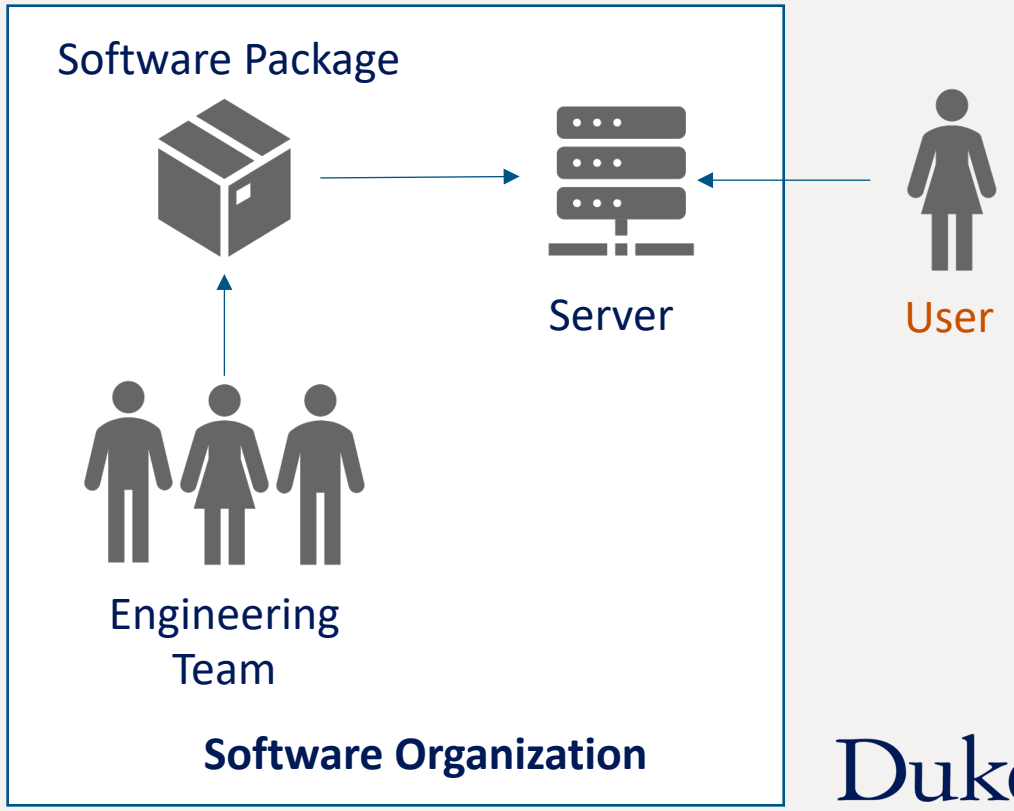
Duke

# Software Delivery Models

## Software as a Product



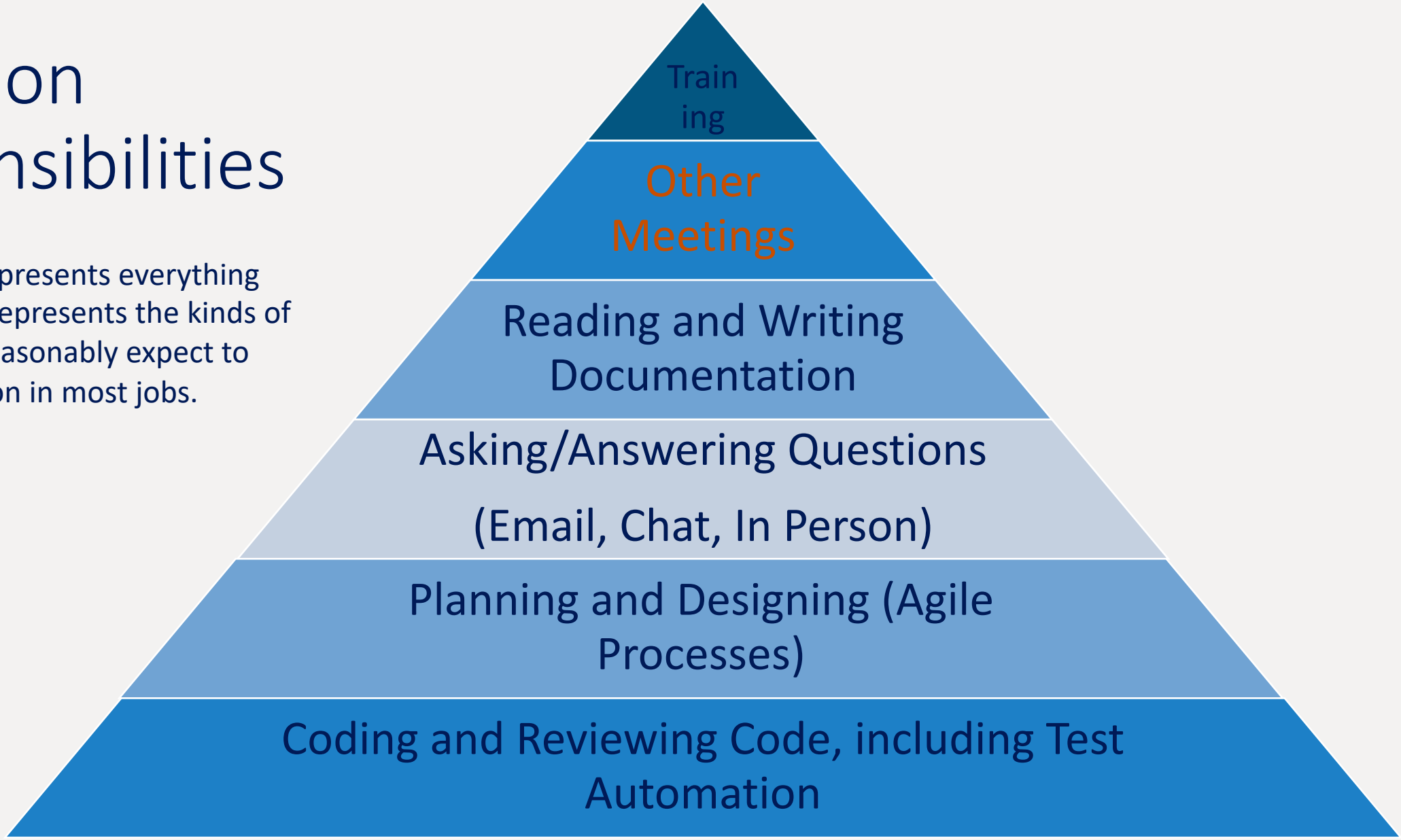

## Software as a Service




Duke

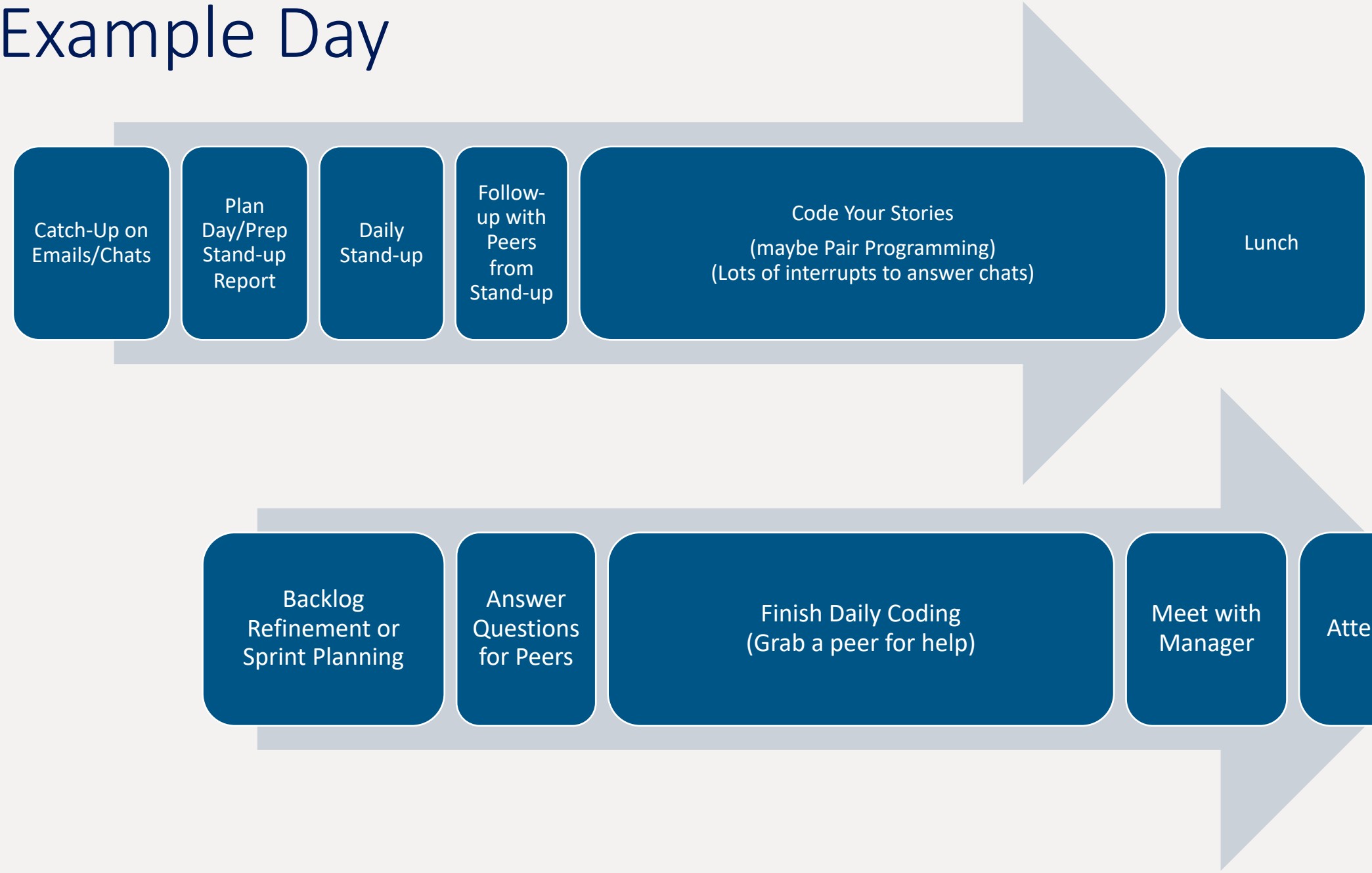
# Common Responsibilities

This in no way represents everything you will do, but represents the kinds of things you can reasonably expect to spend you time on in most jobs.




Duke

# Example Day




Duke

# Work Style features that Vary Significantly Between Places


https://xkcd.com/2562/

- On-Call Responsibilities
- Degree of Independence/Explicitness of Rules
- Frequency of Releases and Degree of Automation
- Technical Sophistication of “Platforms”
- Technical sophistication of management and peers
- Remote vs. In-Person vs. Hybrid
- Pair vs. Independent Programming
- Synchronous vs. Asynchronous Communication
- “Culture”

Duke

# Daily Challenges of Software Work

- Estimation is extremely difficult, which makes effective planning a constant challenge
- Very difficult to decide what's "good enough" – scope is always in flux
- More features than you could ever get done
- Getting access to the knowledge you need (made worse with poor documentation)
- Trade-off between investing for the future and delivering now
  - Shortcuts lead to tech debt, which makes everything slower and harder
- Many ways to do things and standardization takes effort
- Software verification (testing) is an art and not a science


https://xkcd.com/619/

Duke

# Daily Challenges of Software Work

- Rarely working on a project from scratch and need to account for “current state”
- Major decisions are made by higher-ups who don’t understand technical details
- Continuous Learning
  - Need facility in dozens of tools
  - Tools change too quickly to master of all of them
- Subject matter experts are frequently not technically knowledgeable
- Many decisions are made for non-technical reasons (ie, cost of contract)
- Engineering teams have to choose implementations within their ability
- Complexity builds up over time and has to be continually pruned
- Every piece of code has to be maintained by *someone*


https://xkcd.com/979/

Duke

# Applying

- Big organizations have formal processes for interns/new grads with no flexibility, small organizations treat them as any other hire with a lot of flexibility. **Know which process you are in.**
- **Go to on-campus recruiting events with your resume**.
- As a student/new grad, **your resume should be short** (1 page, unless you have a lot of OSS contributions)
- Tech internships are long interviews. Your best chance to get a job at a company is to get an internship there first.
- There are no shortcuts for passing leetcode-like screenings. You must practice if you want a job that requires it.

Duke

# Interviewing

- **Know who you are talking to**. What is their role? Are they technical?
  - Screening is generally done by a non-technical person first (HR or recruiting) – they care about when you graduate and if you seem easy to work with.
- No one expects new grads to know everything (or very much, really). They expect you to be able to explain what you have learned. **Stick to what you know and admit what you don't** (these can be great prompts for you to ask questions back to your interviewer).
- Be excited about the tech. Looking like a "geek" is a good thing.
- Whatever the question, **explain your thought process**. People hiring engineers prioritize clear reasoning above almost everything else.
- **Don't obsess over syntax** unless you are writing code that is about to be run. Ask "can I use pseudo-your-favorite-language?".
- Pair programming is good practice for coding interviews.

Duke

# AMA

Duke

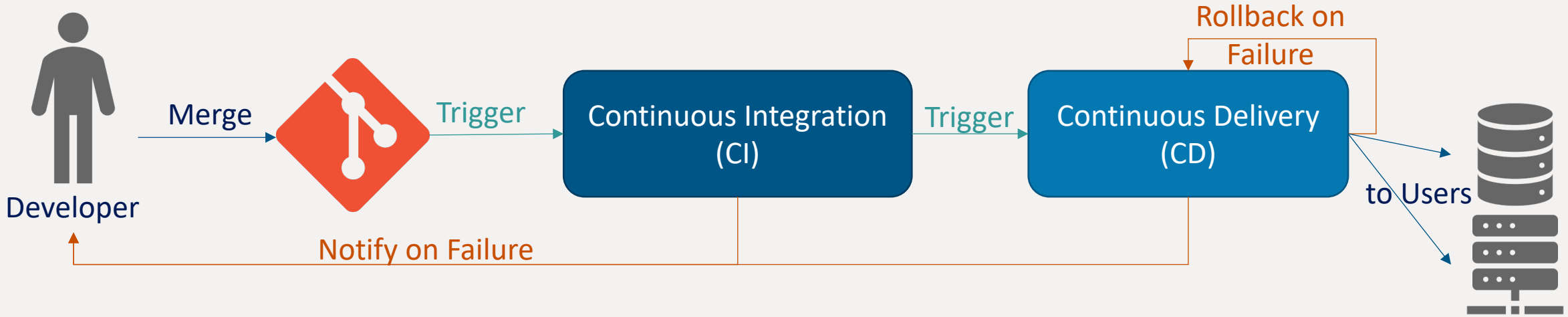
- Have you tried different project management strategies (Scrum, Kanban, etc.)? What are the differences between them and which one is your favorite/least favorite?
- How do you resolve conflicts (e.g., if two people have different design ideas)?
- What's your favorite part and least favorite part about industry?
- how do you form connections with your team/with your manager?
- how do you get the most out of each meeting?
- Do you ever struggle to be professional/formal? What are some ways you try to work around that? Is being informal more common?
- How did you grade interviewees when you were doing interviews?

Duke

# Appendix

Duke

# The Software Delivery Pipeline




## Continuous Integration (CI)

| Pre-Commit Checks | Pre-Merge Checks | Post-Merge Checks | Integration Testing |
|---|---|---|---|
| Examples: Code Formatting | Code Compilation | Unit Testing & Coverage | Tests on Fresh Install |
| Commit Message | Can Merge? | Quality & Security Analysis | Dynamic Security Tests |

## Continuous Delivery (CD)

| Release Automation | |
|---|---|
| Install Package | Analyze Telemetry |
| Publish Docs | Notify Engineers |