Software Engineering
The Job

COMPSCI 308 – 2024-04-09
Luke Moffett, Duke CS
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
What is the Job?

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

<table>
<thead>
<tr>
<th>Start-Ups</th>
<th>Tech Company</th>
<th>Big Corp/Gov’t/Non-Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High Freedom</td>
<td>• Sophisticated Platforms</td>
<td>• Stability</td>
</tr>
<tr>
<td>• Small Tech Footprints</td>
<td>• Internal Standards and Bespoke Tools</td>
<td>• Technology <em>Supports</em> Mission (“Cost Center”)</td>
</tr>
<tr>
<td>• Uncertainty with Upside</td>
<td>• High Specialization</td>
<td>– Best Engineers Understand Both</td>
</tr>
<tr>
<td>• “Many Hats”</td>
<td>• Maintaining/Expanding Existing Products/Services</td>
<td>• Integrating or Customizing Tools from Tech Companies and Start-Ups</td>
</tr>
<tr>
<td>• Management Tends to be Technical</td>
<td>• Excellent Compensation</td>
<td>• Every Industry</td>
</tr>
<tr>
<td>• Developing New Products/Services</td>
<td>• High Skill Peers/Hard to Stand-Out</td>
<td>• Peer Abilities Vary</td>
</tr>
<tr>
<td>• Peer Abilities Vary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software Delivery Models

Software as a Product

1. Download
2. Install
3. Use

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.
Example Day

- Catch-Up on Emails/Chats
- Plan Day/Prep Stand-up Report
- Daily Stand-up
- Follow-up with Peers from Stand-up
- Code Your Stories
  (maybe Pair Programming)
  (Lots of interrupts to answer chats)
- Lunch

- Backlog Refinement or Sprint Planning
- Answer Questions for Peers
- Finish Daily Coding
  (Grab a peer for help)
- Meet with Manager
- Attend Training

(Duke)
Work Style features that Vary Significantly Between Places

- 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”

https://xkcd.com/2562/
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/
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/
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.
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.
AMA
• 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?
Appendix
The Software Delivery Pipeline

**Continuous Integration (CI)**
- Pre-Commit Checks: Code Formatting, Commit Message
- Pre-Merge Checks: Code Compilation, Can Merge?
- Post-Merge Checks: Unit Testing & Coverage

**Continuous Delivery (CD)**
- Release Automation: Install Package, Analyze Telemetry
- Integration Testing: Tests on Fresh Install
- Quality & Security Analysis

**Examples**
- Code Formatting
- Code Compilation
- Unit Testing & Coverage
- Can Merge?
- Quality & Security Analysis
- Dynamic Security Tests

**Flow**
- Developer Merge
- Continuous Integration (CI) Trigger
- Continuous Delivery (CD) Trigger
- Notify on Failure
- Rollback on Failure
- to Users