ECE444: Software Engineering

Motivation, Teams

Shurui Zhou

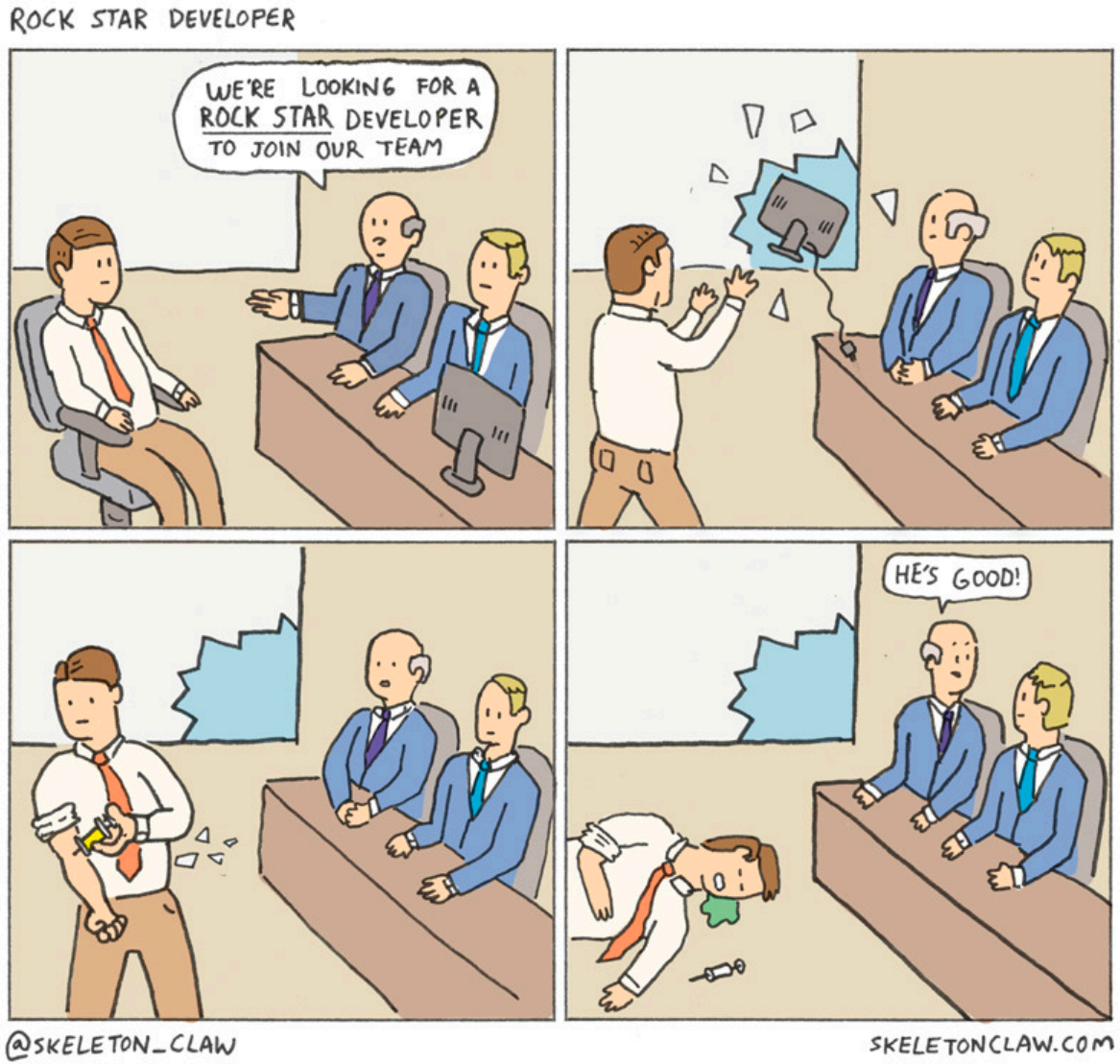
The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Learning Goals

- Understand the differences among developers and implications for hiring and teamwork.
- Describe various models of motivation and their relationship to productive work environments.
- Design conditions that motivate developers.

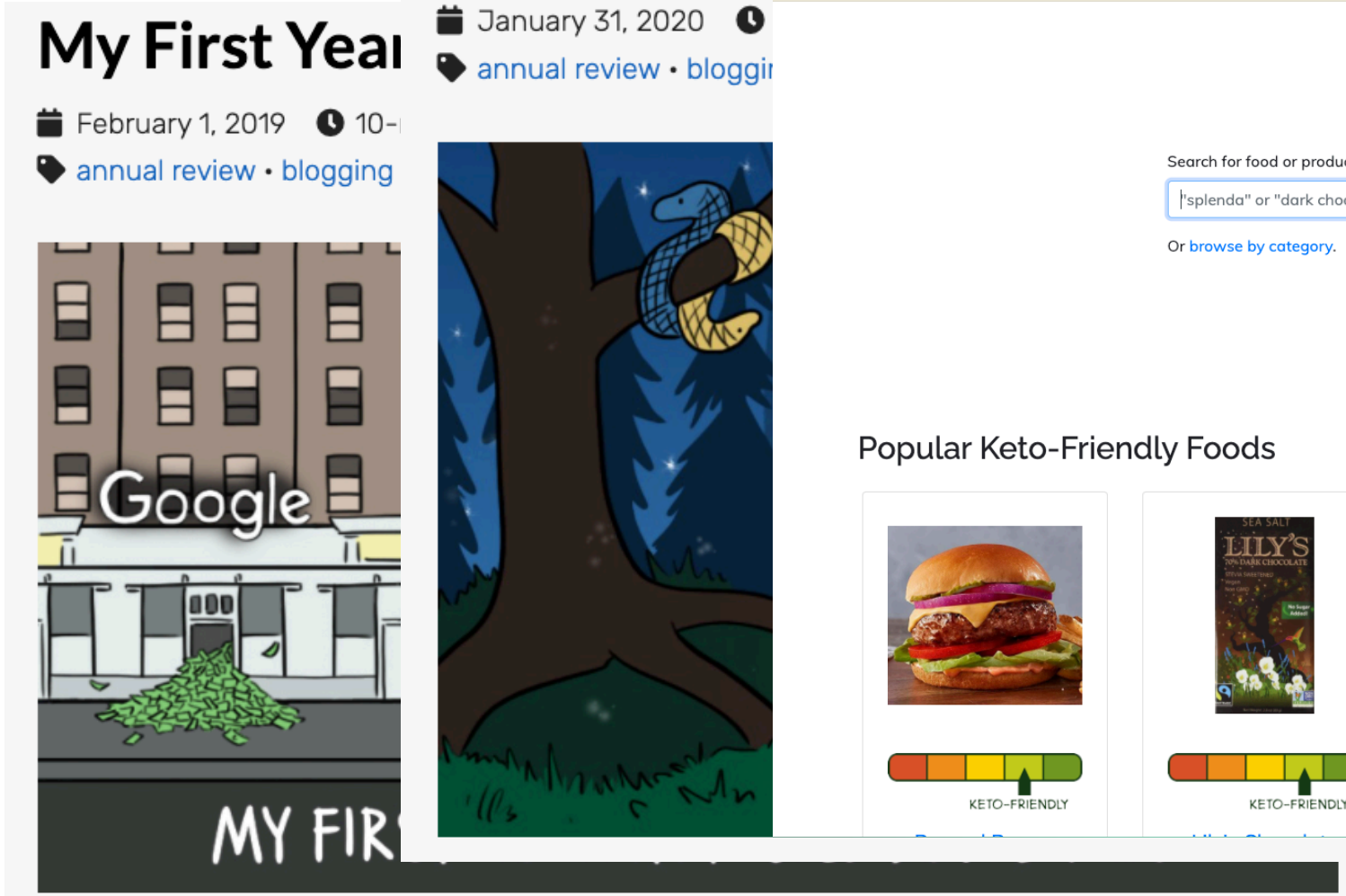
10X Engineers

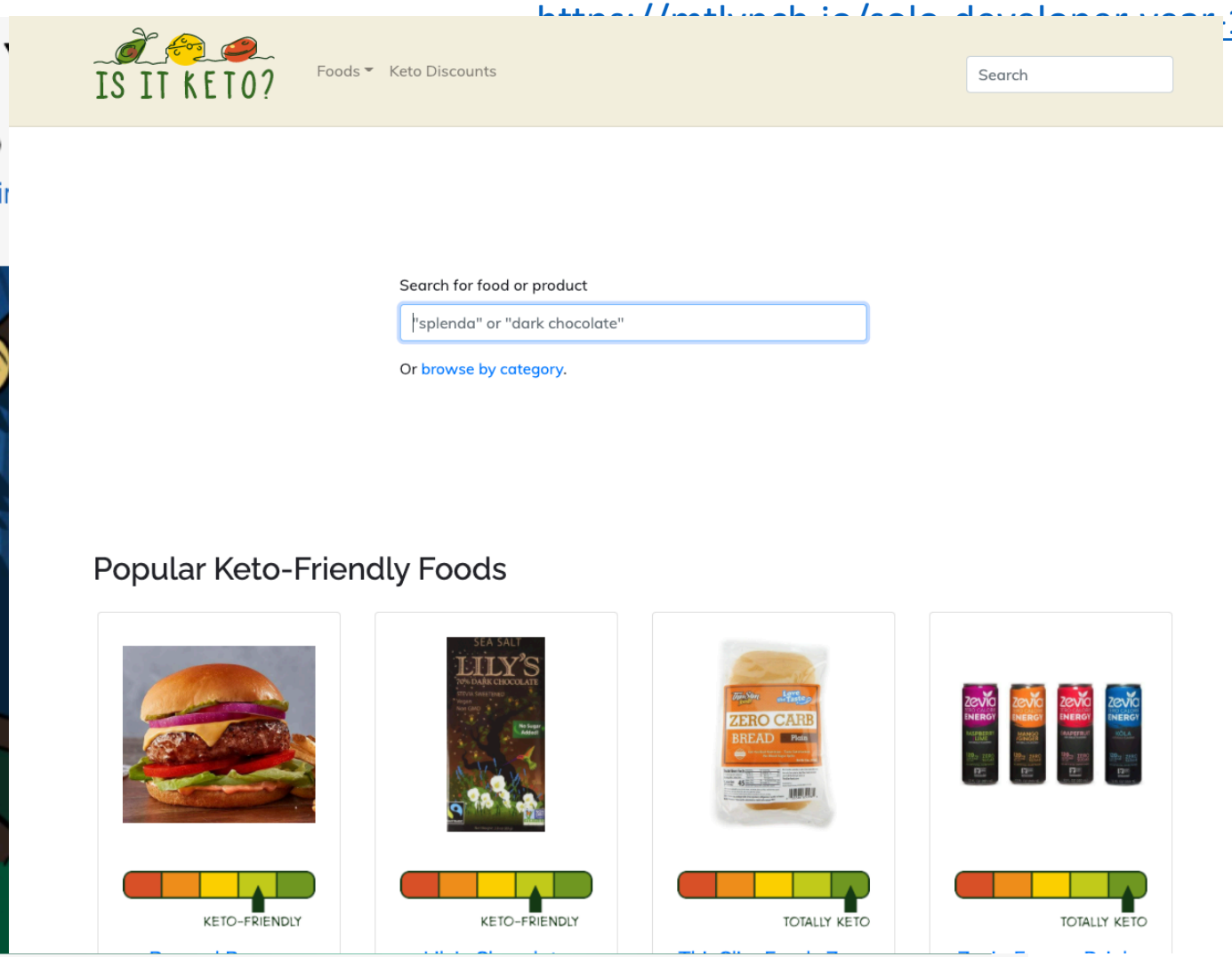
- Aka “rock-star”, “ninja”



Solo prog

https://mtlynch.io/solo-developer-year-1/





Operating Systems

B. RANDELL, Editor

Exploratory Experimental Studies Comparing Online and Offline Programming Performance

H. SACKMAN, W. J. ERIKSON, AND E. E. GRANT
System Development Corporation
Santa Monica, California

Two exploratory experiments were conducted at System Development Corporation to compare debugging performance of programmers working under conditions of online and offline access to a computer. These are the first known studies that measure programmers' performance under controlled conditions for standard tasks.

Statistically significant results of both experiments indicated faster debugging under online conditions, but perhaps the most important practical finding involves the striking individual differences in programmer performance. Methodological problems encountered in designing and conducting these experiments are described; limitations of the findings are pointed out; hypotheses are presented to account for results; and suggestions are made for further research.

Amid all these portents of the dominating role that computer programming will play in the emerging computer scene, one would expect that computer programming would be the object of intensive applied scientific study. This is not the case. There is, in fact, an applied scientific *lag* in the study of computer programmers and computer programming—a widening and critical lag that threatens the industry and the profession with the great waste that inevitably accompanies the absence of systematic and established methods and findings and their substitution by anecdotal opinion, vested interests, and provincialism.

The problem of the applied scientific lag in computer programming is strikingly highlighted in the field of online versus offline programming. The spectacular increase in the number of time-shared computing systems over the last few years has raised a critical issue for many, if not most, managers of computing facilities. Should they or should they not convert from a batch-processing operation, or from some other form of noninteractive information processing, to time-shared operations? Spirited controversy has been generated at professional meetings, in the literature, and at grass roots, but virtually no experimental comparisons have been made to test and evaluate these competing alternatives objectively under controlled conditions. Except for related studies by Gold 1967 [4], and by

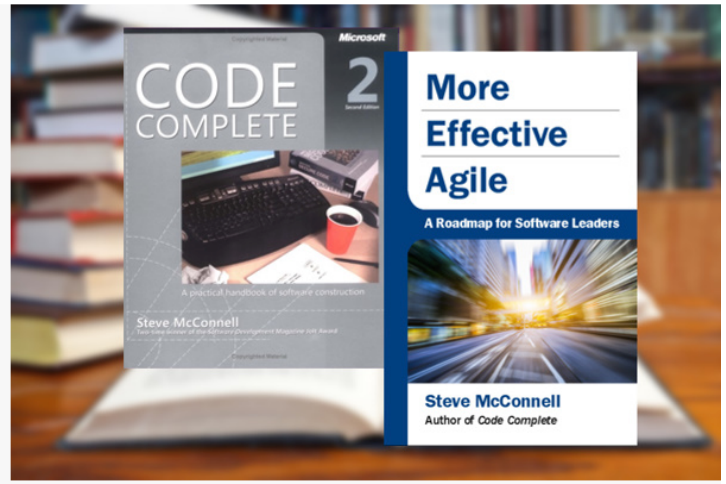
10x

- Reported as early as 1968 (Sackman, Erickson, and Grant)
  - Coding time 20:1
  - Debugging time 25:1
  - Program size 5:1
  - Execution speed 10:1
  - No correlation to amount of experience
- "order-of-magnitude differences among programmers" repeatedly reported
- Differences not explained by
  - programming language
  - years of experience

https://www.construx.com/blog/the-origins-of-10x-how-valid-is-the-underlying-research/

“During the time I was at Boeing in the mid 1980s, there was a project that had about 80 programmers working on it that was at risk of missing a critical deadline. The project was critical to Boeing, and so they moved most of the 80 people off that project and brought in one guy who finished all the coding and delivered the software on time.”

– Steve McConnell

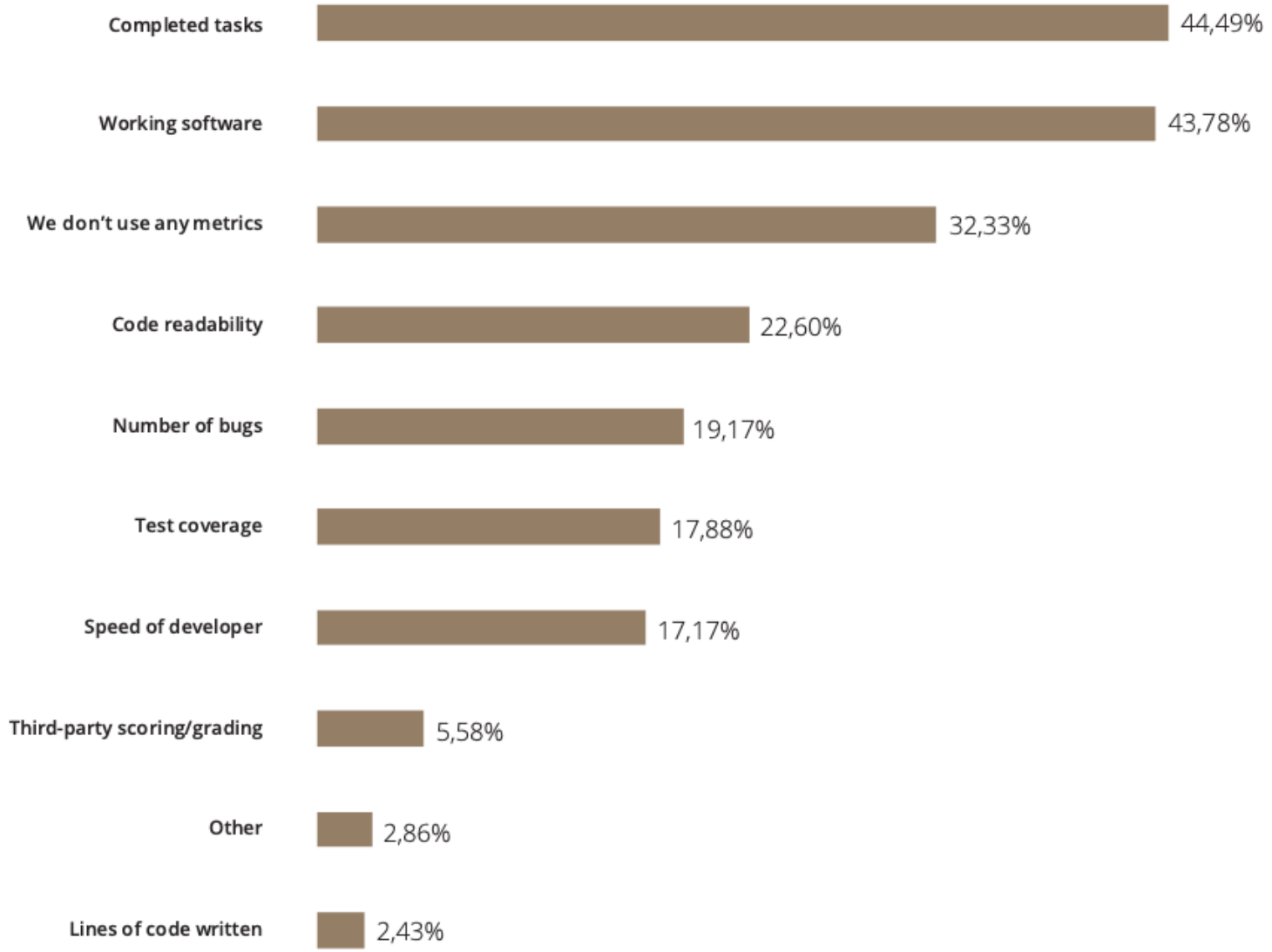


10x of Teams

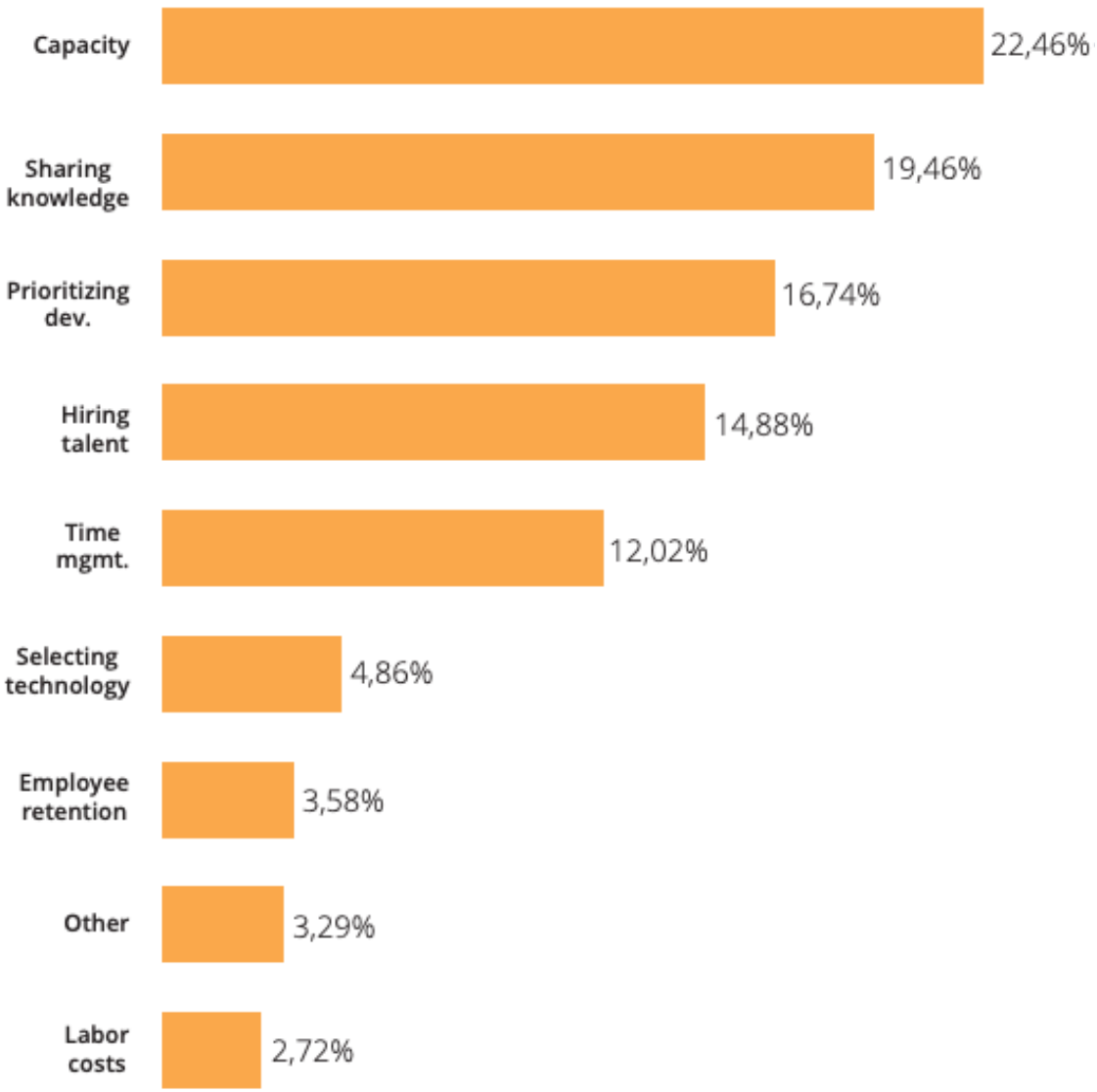
- Lotus 123 version 3
- 260 staff years
- 400,000 lines of code.

- Microsoft Excel 3.0
- 50 staff years
- 649,000 lines of code

What metrics do you use to measure developer performance? (2020)

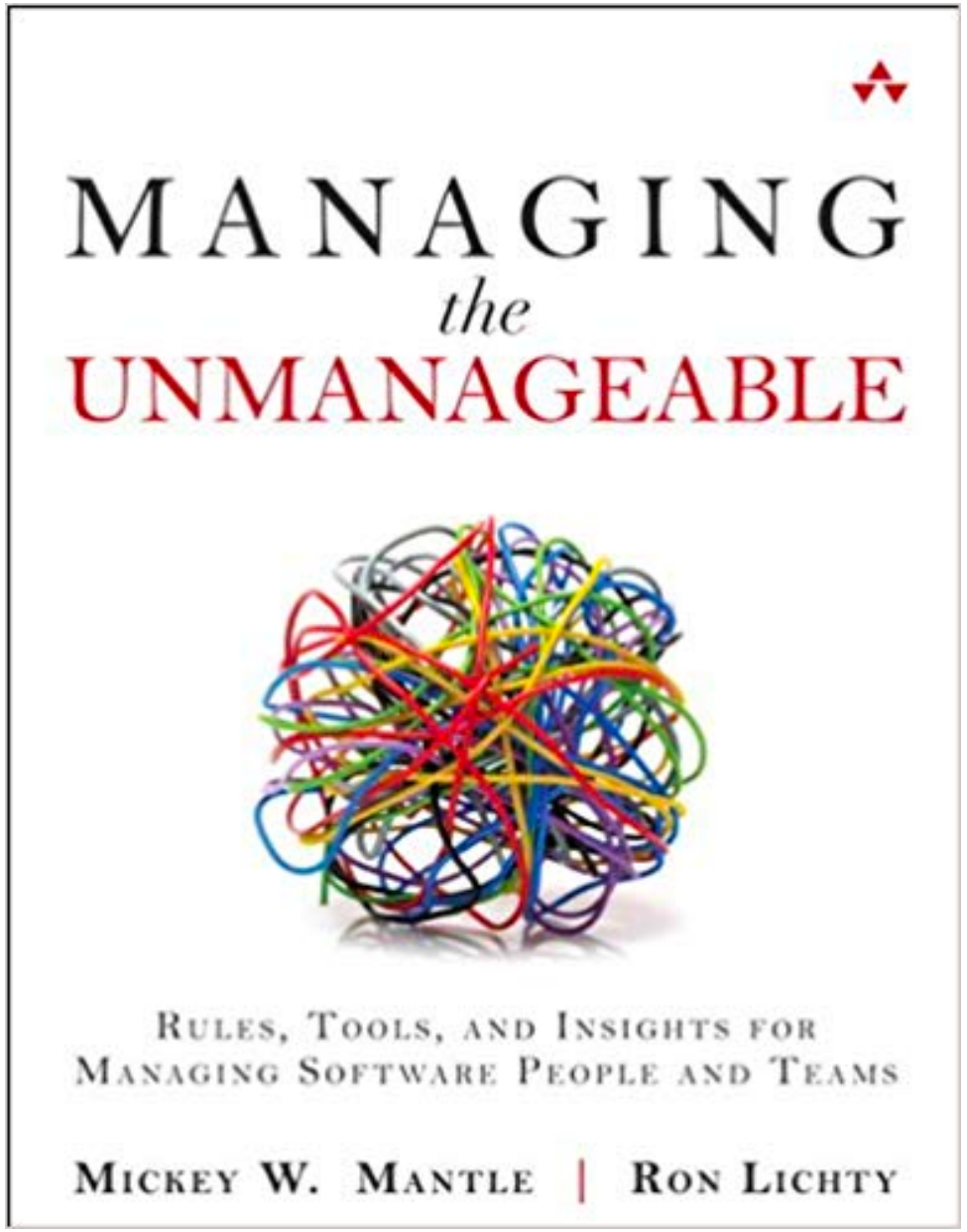


What is your biggest challenge in software development?



https://codingsans.com/blog/recruiting-engineers

RECRUITING ENGINEERS: ASANA'S SECRETS TO HIRING TALENT (INTERVIEW WITH GREG SABO, ENGINEERING MANAGER AT ASANA)



"I just wish that I had this book when I started as a first-time manager five years ago!"

"Becoming a great engineering leader requires more than technical know-how; Ron and Mickey's book provides a practical cookbook for the important softer side of engineering leadership, which can be applied to any software development organization."

Why Programmers Seem Unmanageable?

- *Writing a new program from scratch is akin to writing a novel.*
- Anyone can be a programmer
- The practices of SE have had minimal impact

"If having fun is what most programmers do, you may begin to understand why managing programmers is so challenging. If you are being paid to have fun, why would you want to be managed? Being managed takes part of the fun out of the work!"

"Managing programmers is a lot like herding cats"

Understand Programmers

- Programming disciplines (client, server, database, web dev,..)
- Types of programmers (system engineers/architects, system programmer, application programmer, not really programmer)
- Domain expertise
- Programmer job requirements and abilities
- Proximity and relationship
- Generational styles
- Personality styles

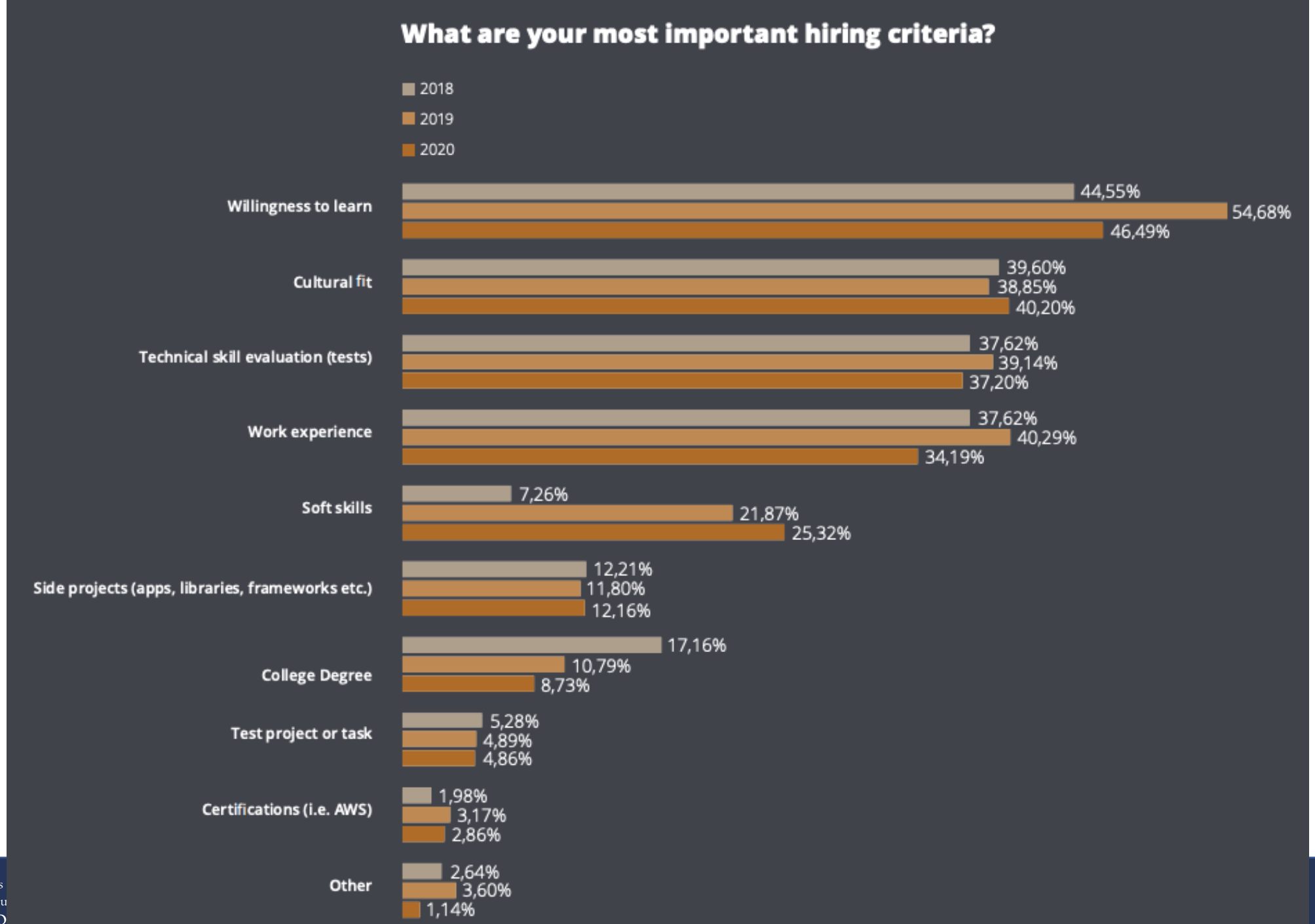

What are your most important hiring criteria?
2018
2019
2020
Willingness to learn
44,55%
54,68%
46,49%
Cultural fit
39,60%
38,85%
40,20%
Technical skill evaluation (tests)
37,62%
39,14%
37,20%
Work experience
37,62%
40,29%
34,19%
Soft skills
7,26%
21,87%
25,32%
Side projects (apps, libraries, frameworks etc.)
12,21%
11,80%
12,16%
College Degree
17,16%
10,79%
8,73%
Test project or task
5,28%
4,89%
4,86%
Certifications (i.e. AWS)
1,98%
3,17%
2,86%
Other
2,64%
3,60%
1,14%

Table 2.3. Client Programmers: Level Criteria

Programmer 3 (Entry Level)

- Knowledge of Windows, Mac, or Linux
- Basic knowledge of good coding practices
- Aware of/interested in Internet technologies
- Aware of/interested in database technologies
- Knowledge of C/C++
- Ability to work in a team and take direction
- Can work with supervisor to plan tasks

Programmer 2 (Some Experience)

- Produced one or more commercial apps
- Proficiency on Windows, Mac, or Linux
- Experience with good coding practices
- Conversant with Internet technologies
- Conversant with database technologies
- Solid understanding of C/C++
- Self-motivated and can take direction
- Can independently plan tasks

Programmer 1 (Experienced)

- Produced two or more commercial apps
- Proficiency on two platforms
- Familiar with Internet technologies
- Familiar with database technologies
- Well versed in C/C++
- Good communication skills
- Self-motivated, minimal direction
- Good project-planning and schedule-estimating skills
- Recognizes problems and helps group adapt

Senior Programmer 2

- Produced two or more commercial apps
- Proficiency on two platforms
- Understands cross-platform issues
- Knowledgeable about Internet technologies
- Knowledgeable about database technologies
- In-depth knowledge of C/C++
- Strong communication skills
- Self-motivated, minimal direction
- Excellent analysis, project-planning, and schedule-estimating skills
- Watches for changing conditions and plans adaptations

Senior Programmer 1

- Produced two or more complex commercial apps or technologies
- Thorough knowledge of two platforms
- Understands cross-platform issues
- Excellent knowledge of Internet technologies
- Knowledgeable about database technologies
- Expert knowledge of C/C++
- Expert in software design practices
- Strong communication skills, industry relationships
- Self-motivated, works independently
- Excellent analysis, project-planning, and schedule-estimating skills
- Generates, enhances, and promotes new ideas
- Watches for changing conditions and plans adaptations

Understand Programmers

- Programming disciplines (client, server, database, web dev,..)
- Types of programmers (system engineers/architects, system programmer, application programmer, not really programmer)
- Domain expertise
- Programmer job requirements and abilities
- Proximity and relationship
- Generational styles (In-house, Geographically distant, Contractors, Contracted managed teams, Outsourcing companies)
- Personality styles

Personality Style

- Left-Brain versus Right-Brain People
- Night versus Morning People
- Cowboys versus Farmers
- Heroes
- Introverts
- Cynics
- …

Interview Advice

Look for people who are:

1. Smart, and
2. Get things done.

https://www.joelonsoftware.com/2006/10/25/the-guerrilla-guide-to-interviewing-version-30/

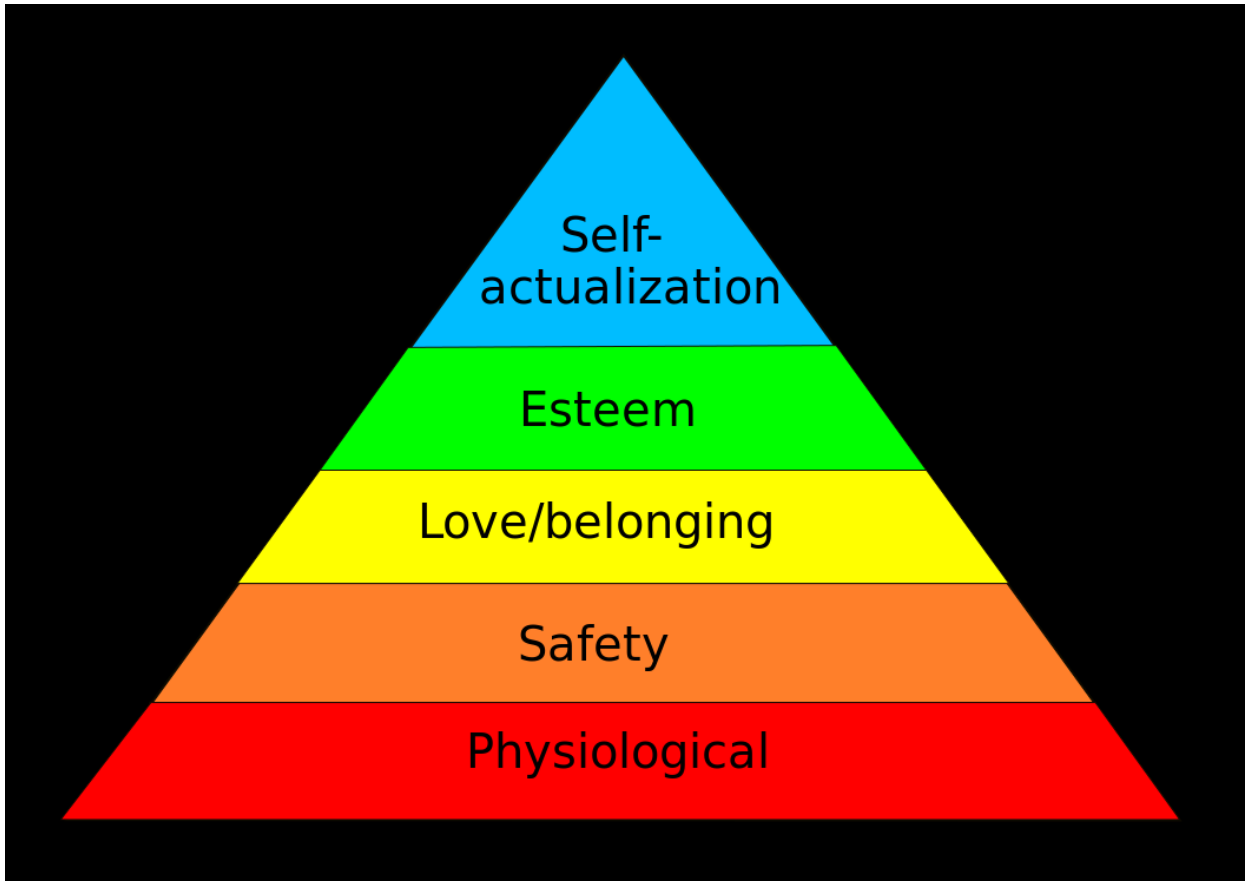
Motivating Programmers

ENGINEERING PRODUCTIVITY: HOW TWO SIGMA KEEPS DEVELOPERS ENGAGED (INTERVIEW WITH CAMILLE FOURNIER, MANAGING DIRECTOR AT TWO SIGMA)

Theories

- Maslow’s Hierarchy of Needs
- Herzberg’s Motivation and Hygiene Factors
- Daniel Pink, Drive: The Surprising Truth About What Motivates Us.

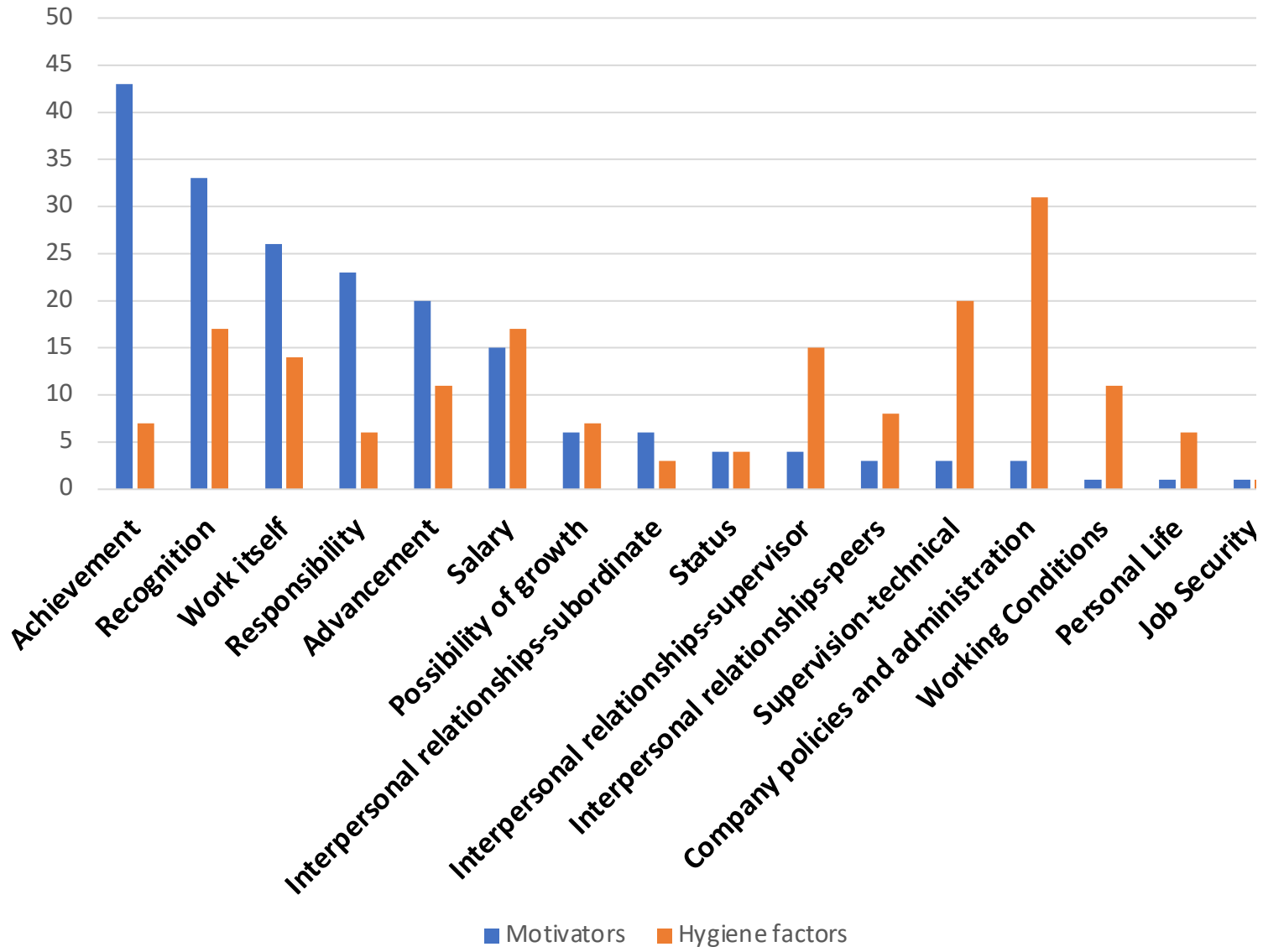
Maslow's hierarchy of needs (1943)



Herzberg's Motivation and Hygiene Factors
(1960s)

- (aka two-factor theory)
- Different factors for satisfaction and dissatisfaction
  - Addressing dissatisfaction does not lead to satisfaction
- Step 1: Eliminate dissatisfaction
- Step 2: Create condition for satisfaction

(Observation by Mantle and Lichty, not empirical data)



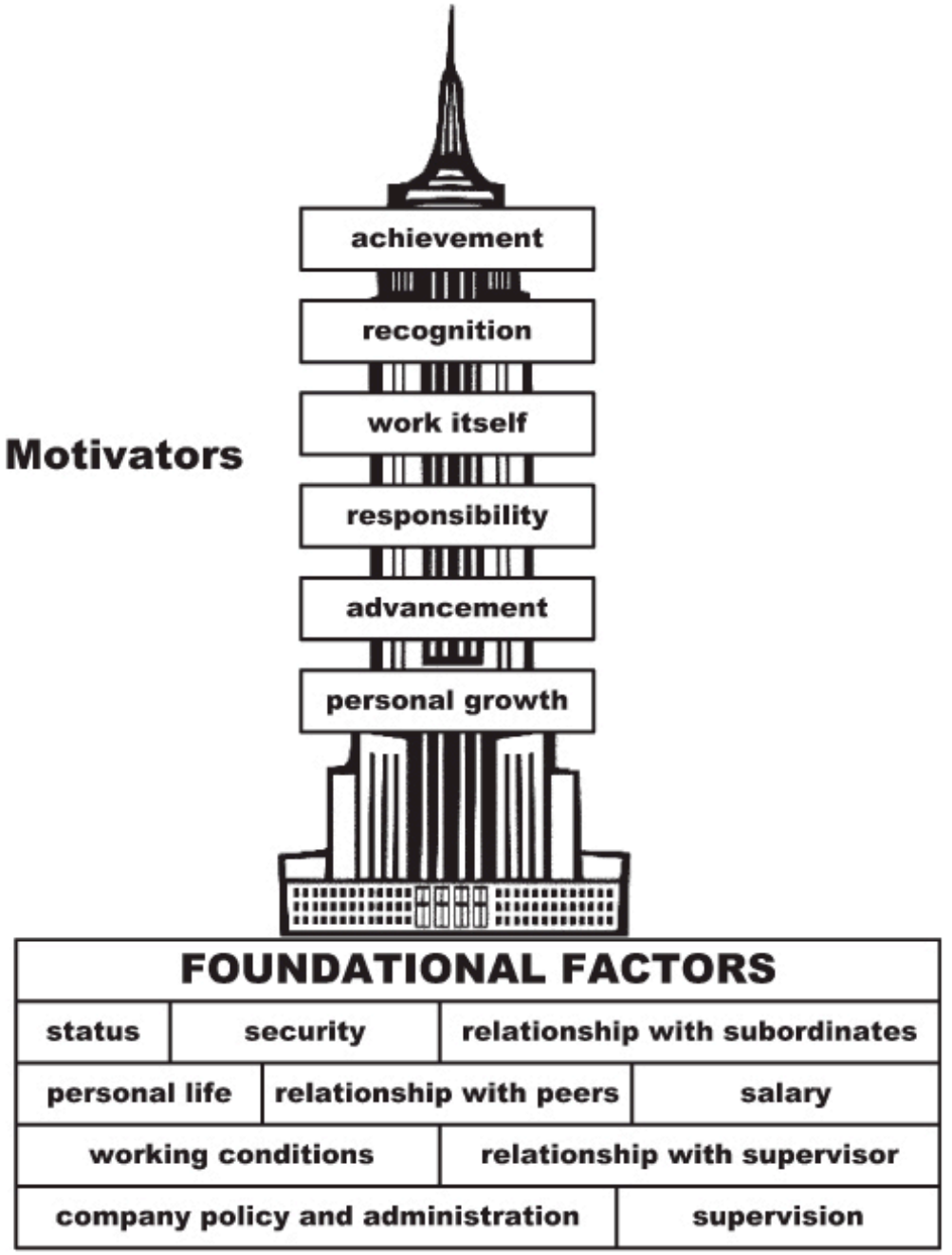
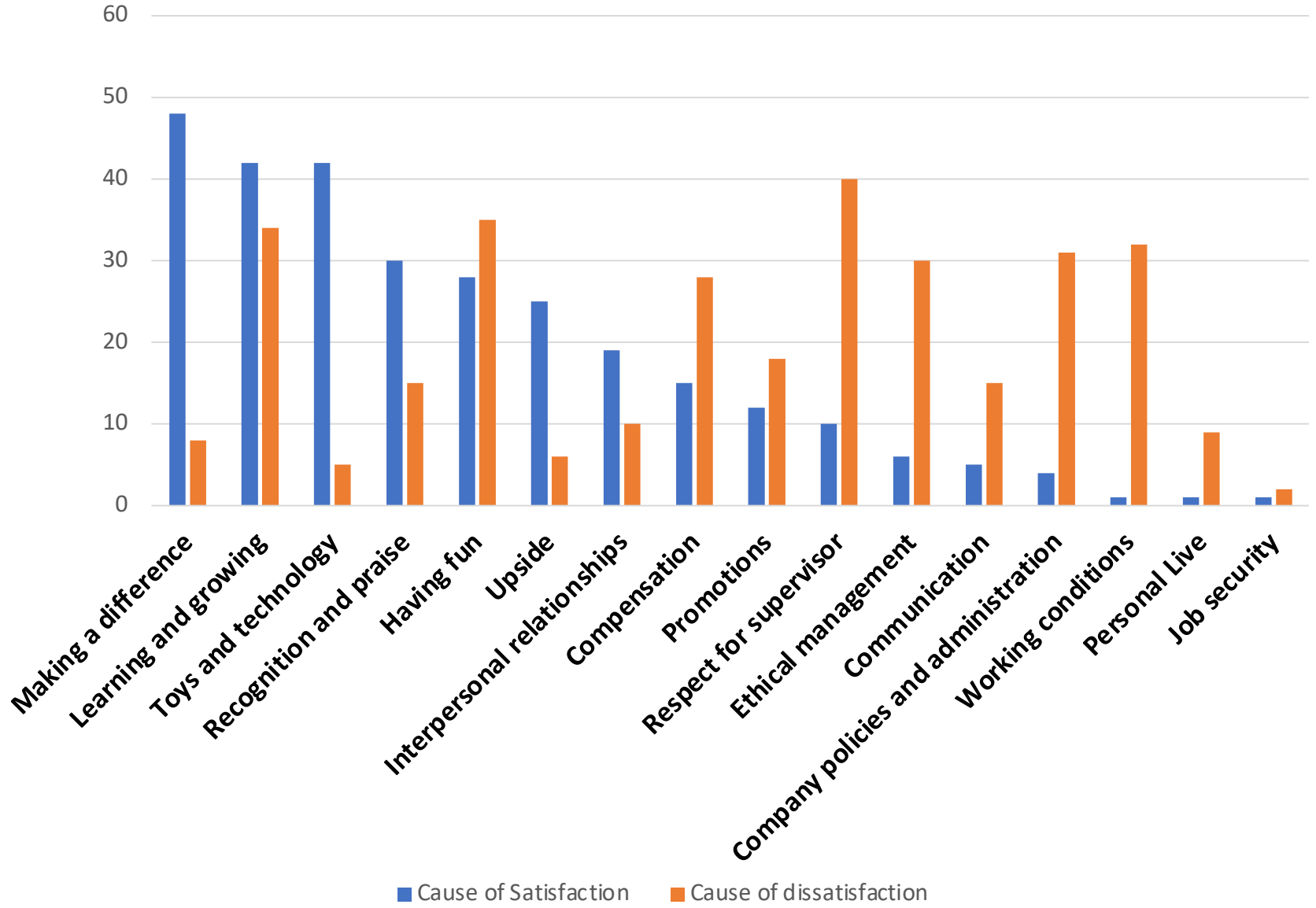


Figure 7.4. Herzberg's Motivators

(Observation by Mantle and Lichty, not empirical data)



Addressing Causes of Dissatisfaction

- Respect for supervisor
- Having fun
- Learning and growing
- Good working conditions
- Sane company policies and administration
- Ethical management
- Fair compensation

- (often within control)

Addressing Causes of Dissatisfaction (selective)

- Respect as supervisor
  - gain technical credit
  - respect others
  - lead by example
  - help solve technical problems
  - manage and coach
- Having fun
  - out of office play
  - celebrations of accomplishments and occasions

Addressing Causes of Dissatisfaction (selective)

- Learning and growing
  - protect time for learning
  - explore new technologies; prototype
  - budget for attending conferences, seminars, inhouse training
  - invite guest speakers
- Good working conditions
  - plenty of whiteboards
  - room for discussions
  - Quiet space, Limit interruptions, avoid meeting culture
  - cubicles vs separate offices
  - fire “jerks”
  - free food
  - flexible hours, flexible dress, flexible space

Addressing Causes of Dissatisfaction (selective)

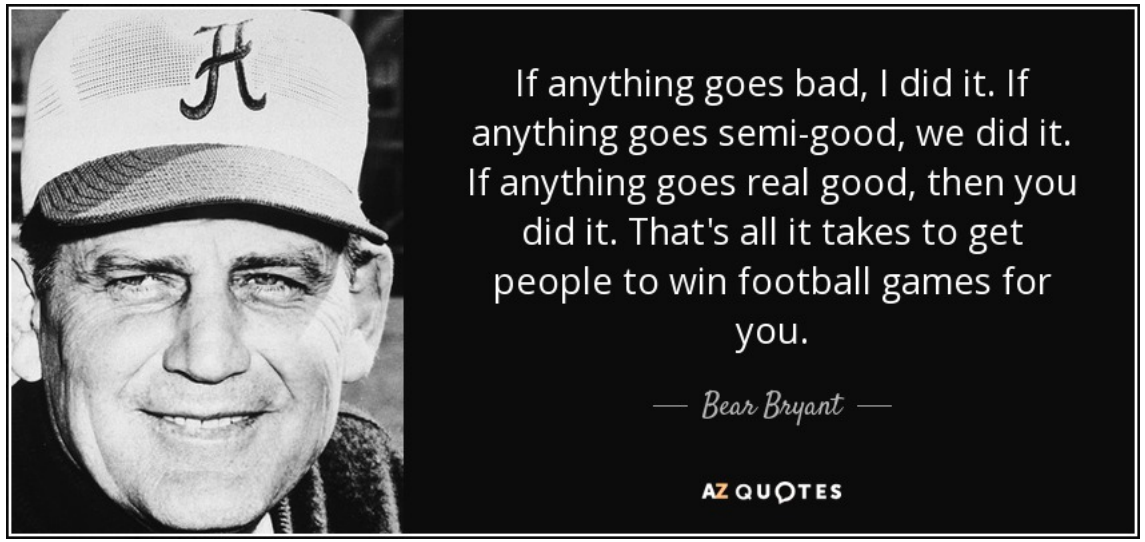
- Sane company policies and administration
  - communicate frequently (vision, intentions, requirements, schedules, …)
  - protect staff from organizational distractions
  - protect staff from bad communication practices (establish culture)

Addressing Motivating Factors (selective)

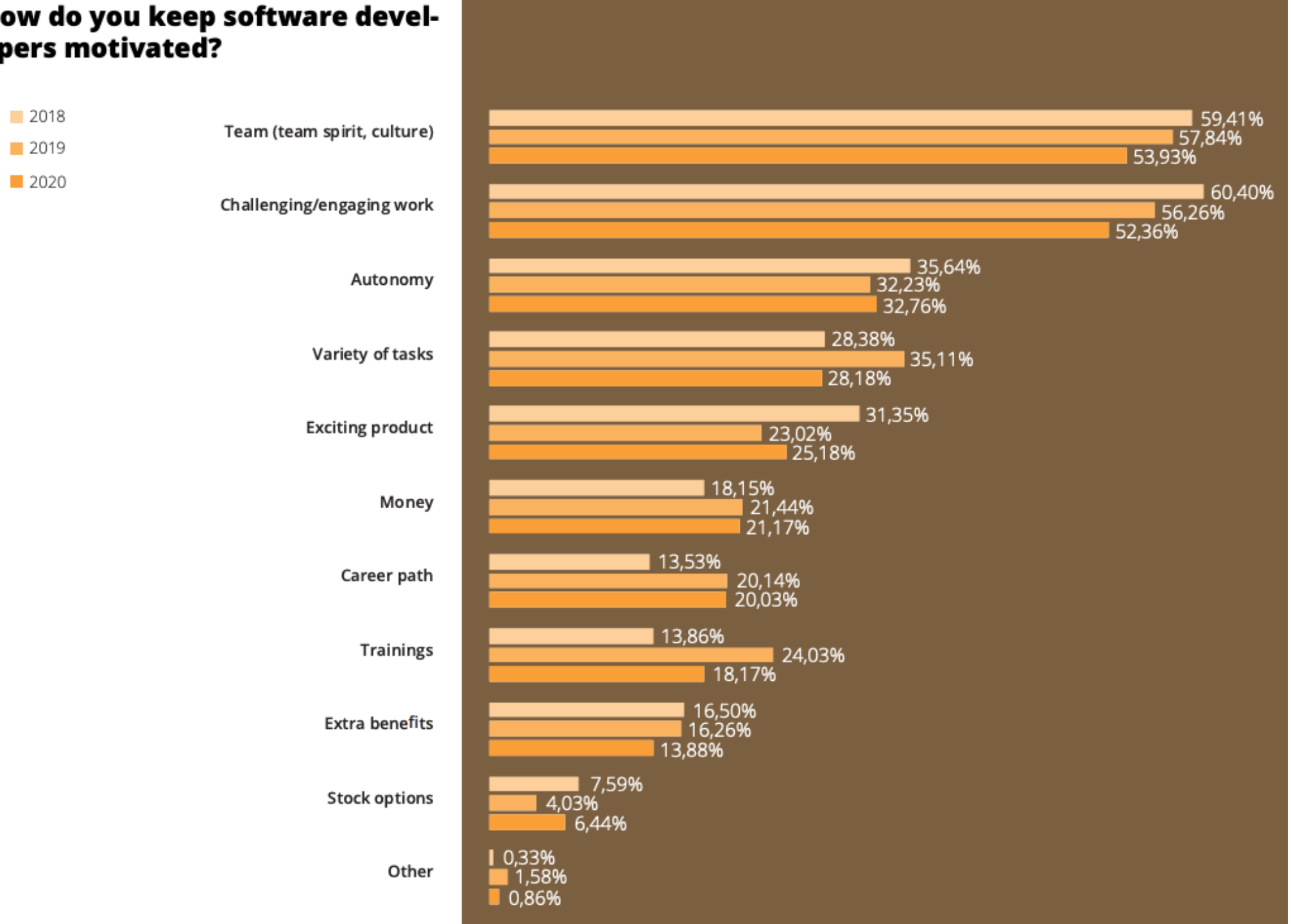
- Making a difference
  - worthy goals, longterm vision
  - Steve Jobs when recruiting John Scully from Pepsi: “Do you want to sell sugar water or change to world”
- Toys and technology
  - modern hardware, large screens, phones, …

Addressing Motivating Factors (selective)

- Recognition and praise
  - praise loudly and specifically, blame softly/privately
  - celebrate success



How do you keep software developers motivated?



Why do engineers choose TO JOIN particular teams?

Reasons grouped by clustering analysis	Percent
Liked new team and/or technology (exciting, manager)	**85.8%**
Coworker asked me to join (new team, old team)	**37.8%**
Joined for better opportunities (location, domain, lack of other options)	**24.5%**
Followed my manager (former or current)	**14.6%**

Why do engineers want to leave their teams?

Reasons grouped by clustering analysis	Percent
Change is coming (technology, charter, re-org, turnover)	**52.6%**
Seeking new challenges or location (role, location, challenges)	**39.0%**
Dissatisfaction with manager (priorities, goals, person, actions)	**31.6%**
The grass is always greener on the other side (novelty, escape)	**12.3%**
Not a good fit (bored, no need for my skills)	**5.3%**
Poor team dynamics (dysfunctional, no career growth)	**4.4%**

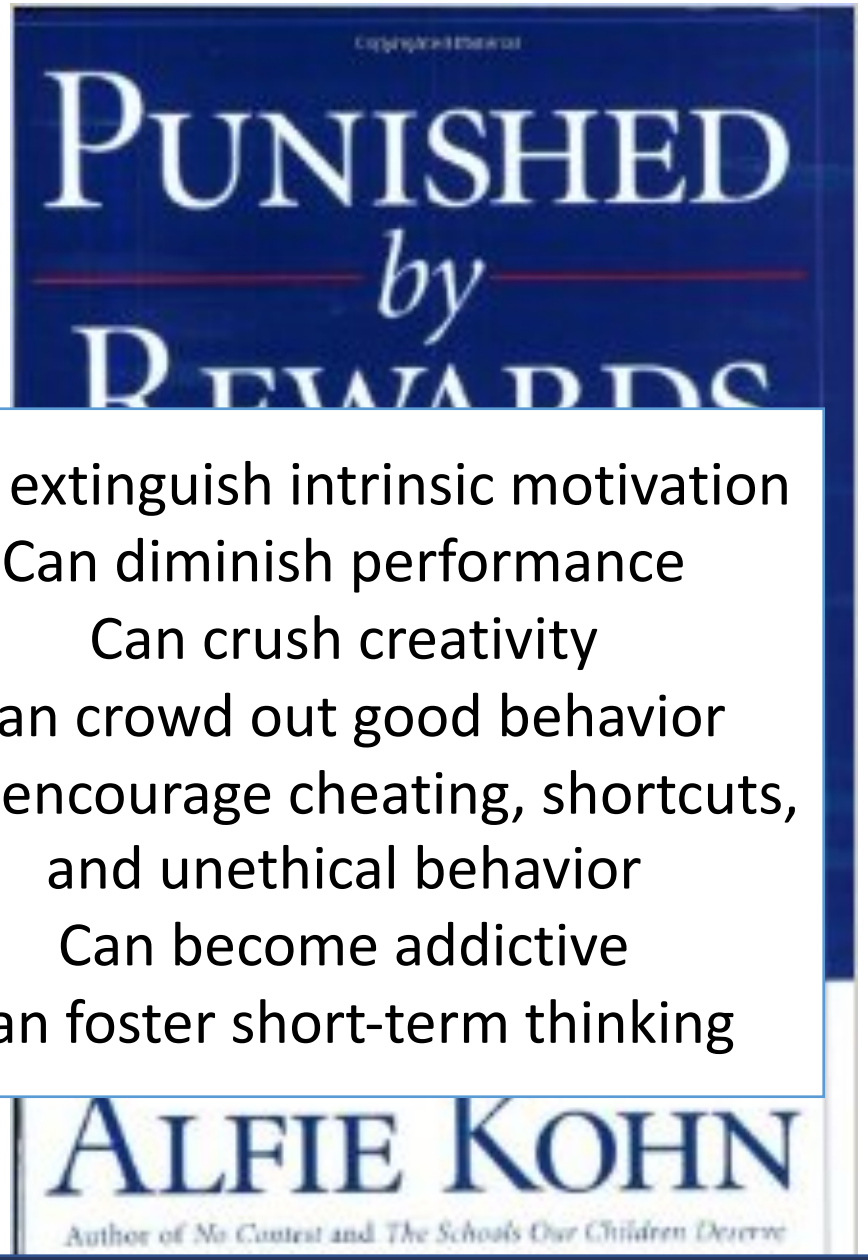
JULIA ROZOVSKY, ANALYST, GOOGLE PEOPLE OPERATIONS | NOVEMBER 17, 2015

The five keys to a successful Google team

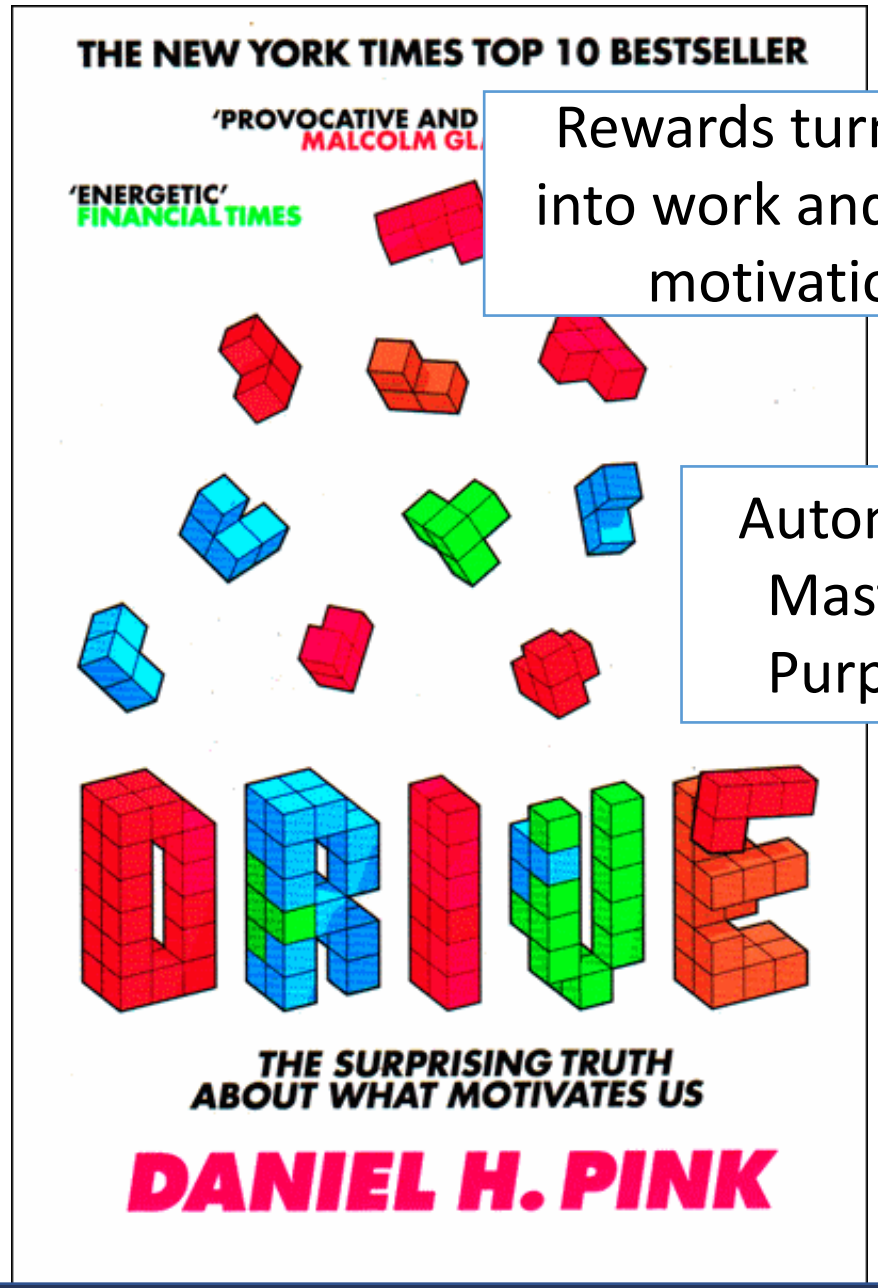
Pod. Work group. Committee. Autonomous collective. Whatever you call it, you're part of one at Google and probably wherever you work: a team. So if we know what makes managers great, why don't we know what makes a team great?

1. **Psychological safety:** Can we take risks on this team without feeling insecure or embarrassed?
2. **Dependability:** Can we count on each other to do high quality work on time?
3. **Structure & clarity:** Are goals, roles, and execution plans on our team clear?
4. **Meaning of work:** Are we working on something that is personally important for each of us?
5. **Impact of work:** Do we fundamentally believe that the work we're doing matters?

https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/



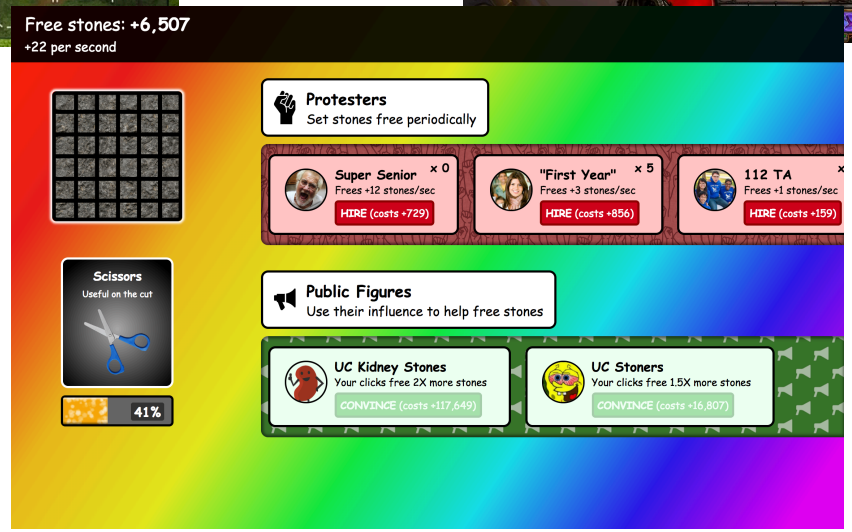
Can extinguish intrinsic motivation
Can diminish performance
Can crush creativity
Can crowd out good behavior
Can encourage cheating, shortcuts, and unethical behavior
Can become addictive
Can foster short-term thinking



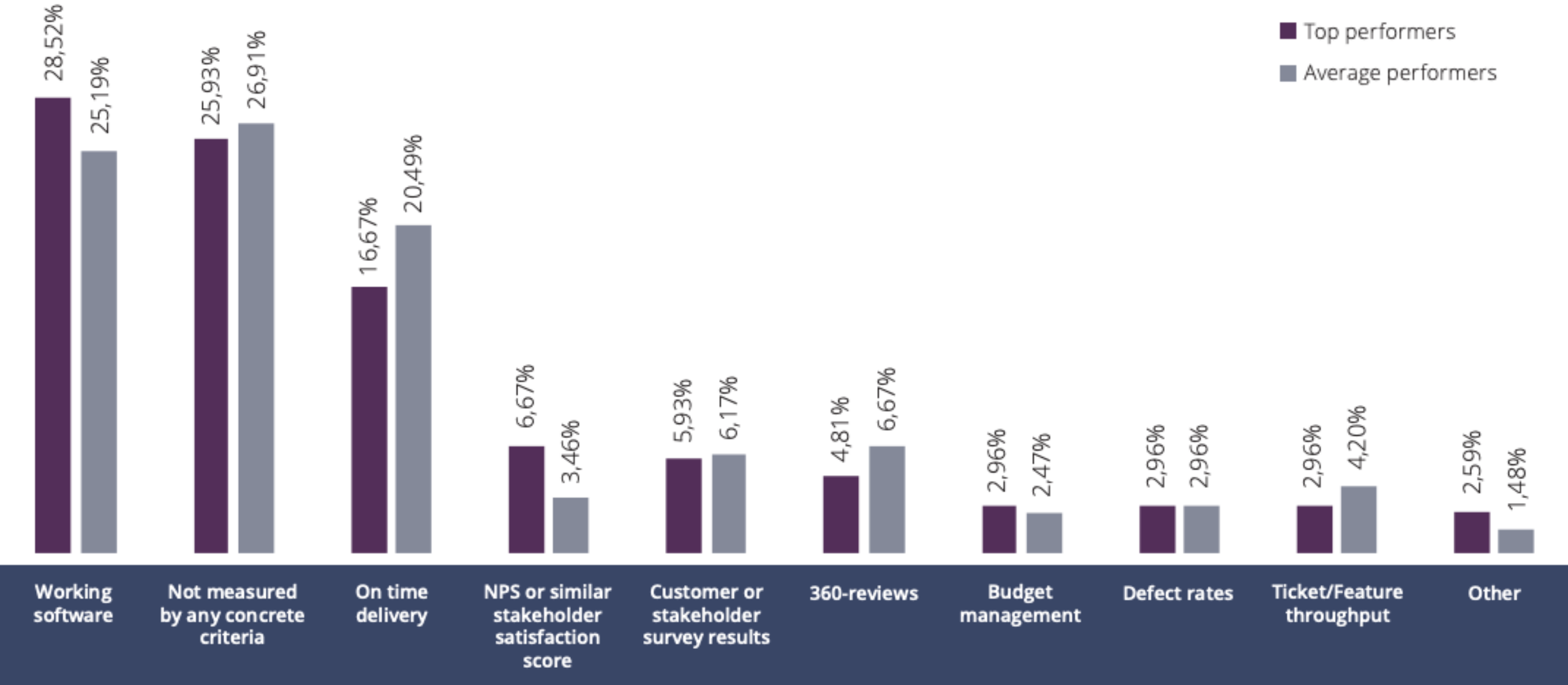
Rewards turn play into work and drain motivation

Autonomy
Mastery
Purpose

Rewards (aka grinding)



What is the #1 measured criteria for success of software development managers? (2020, top vs average)



Further Reading

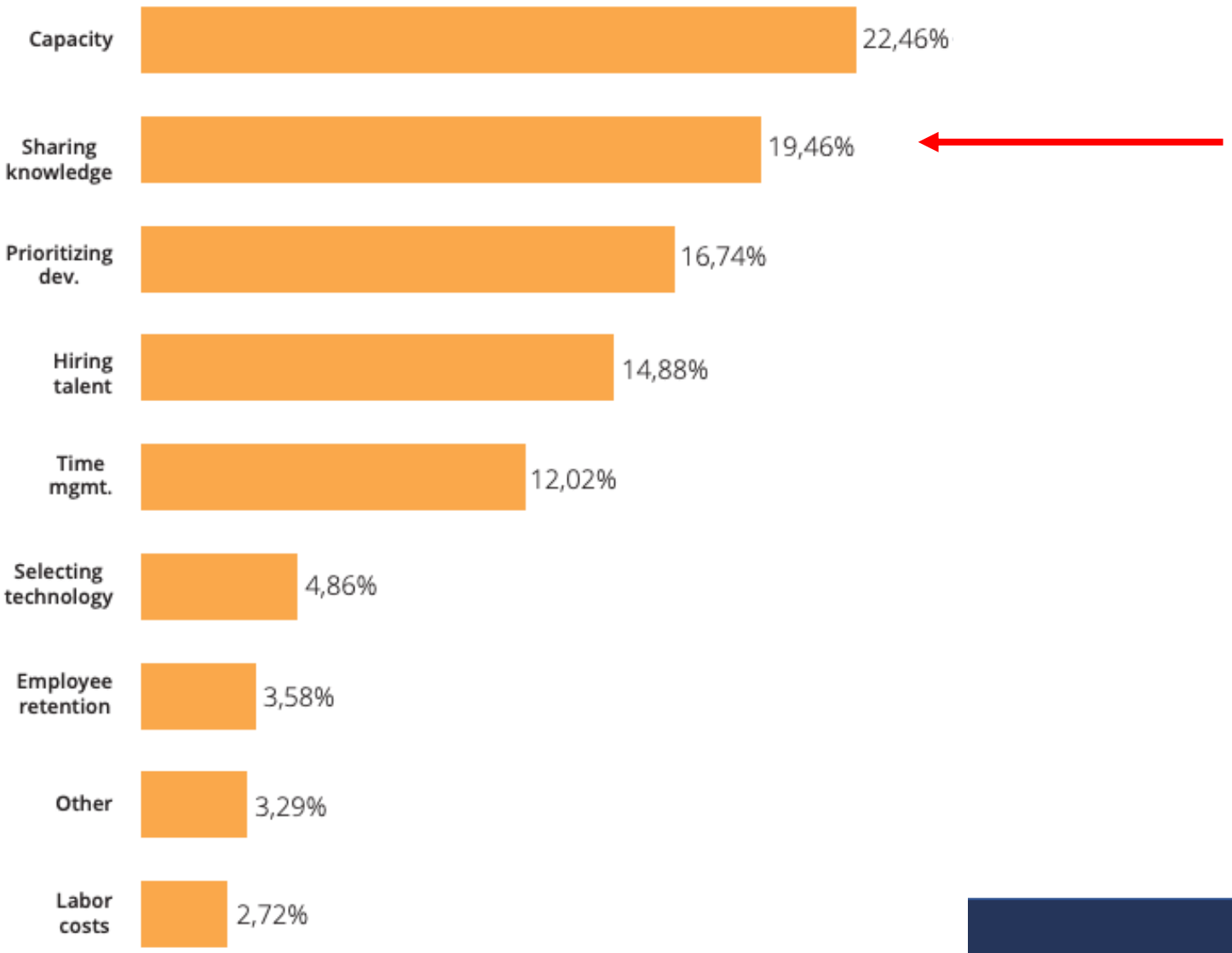
- Mantle and Lichty. Managing the Unmanageable. Addison-Wesley, 2013
  - Very accessible and practical tips at recruiting and managment
- DeMarco and Lister. Peopleware. 3rd Edition. Addison Wesley, 2013
  - Anecdotes, stories, and tips on facilitating teams, projects, and environments
- Pink. Drive: The Surprising Truth About What Motivates Us. Riverhead 2011
  - Detailed discussion of motivating factors for creative people
- Sommerville. Software Engineering. 8th Edition. Chapter 25

State of Software Development

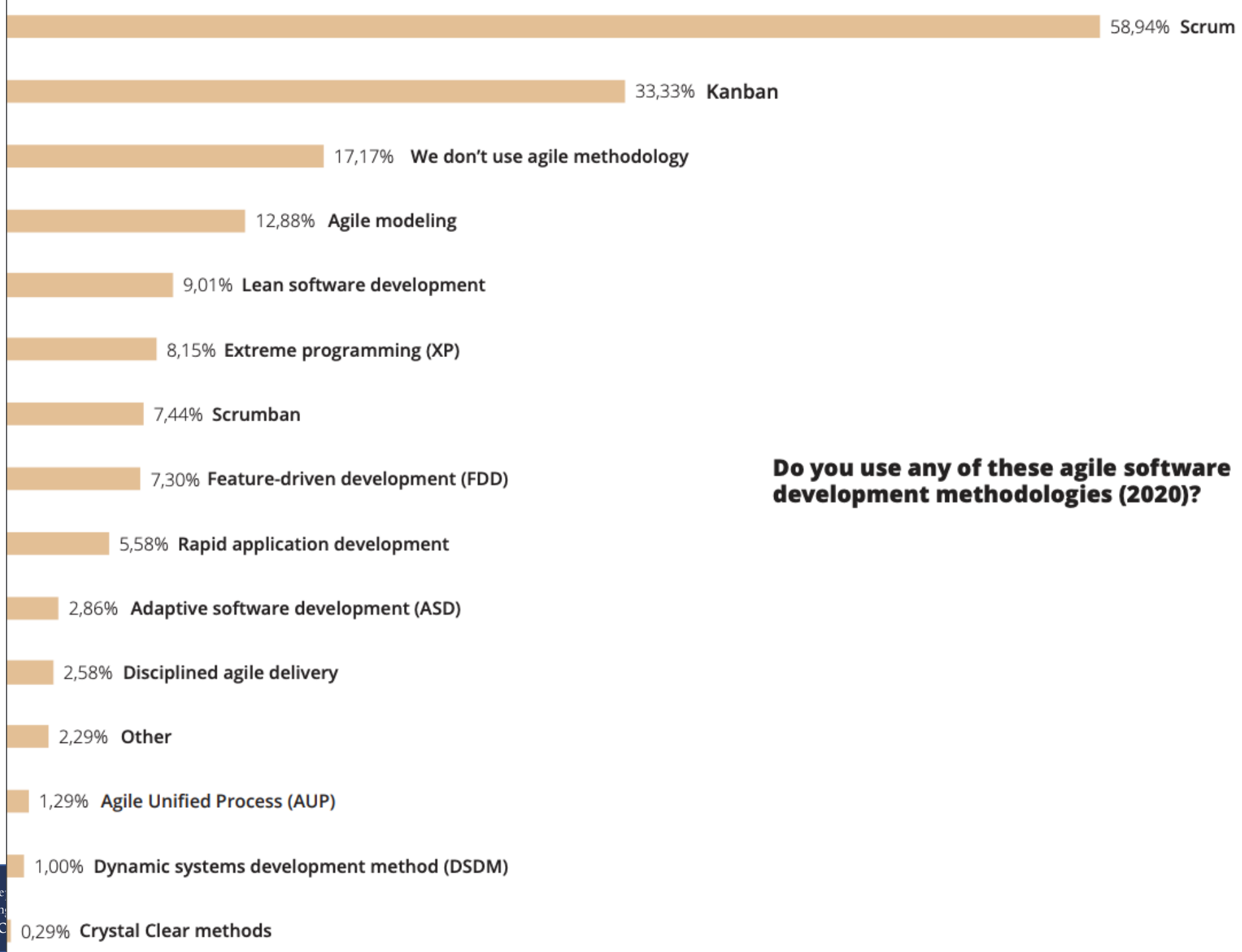
https://codingsans.com/uploads/landing/State-of-Software-Development-2020.pdf

What industry are you in?

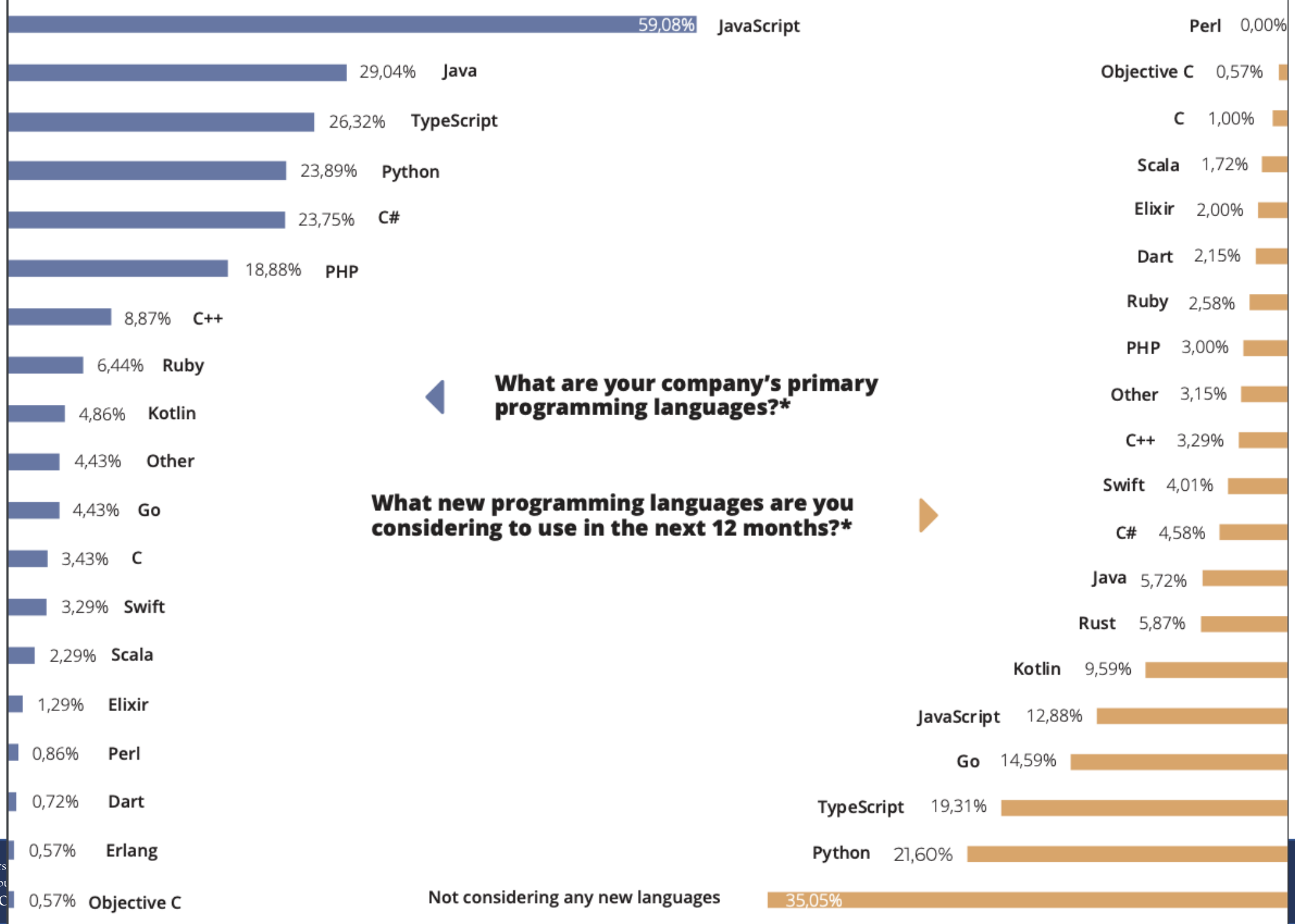
What is your biggest challenge in software development?

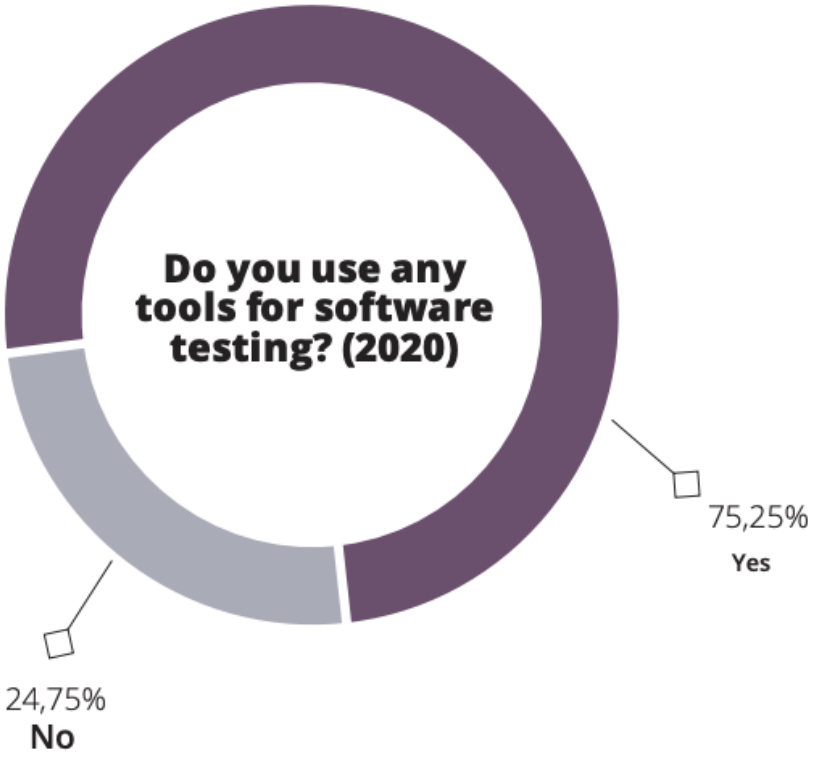


What is your biggest challenge in software development?

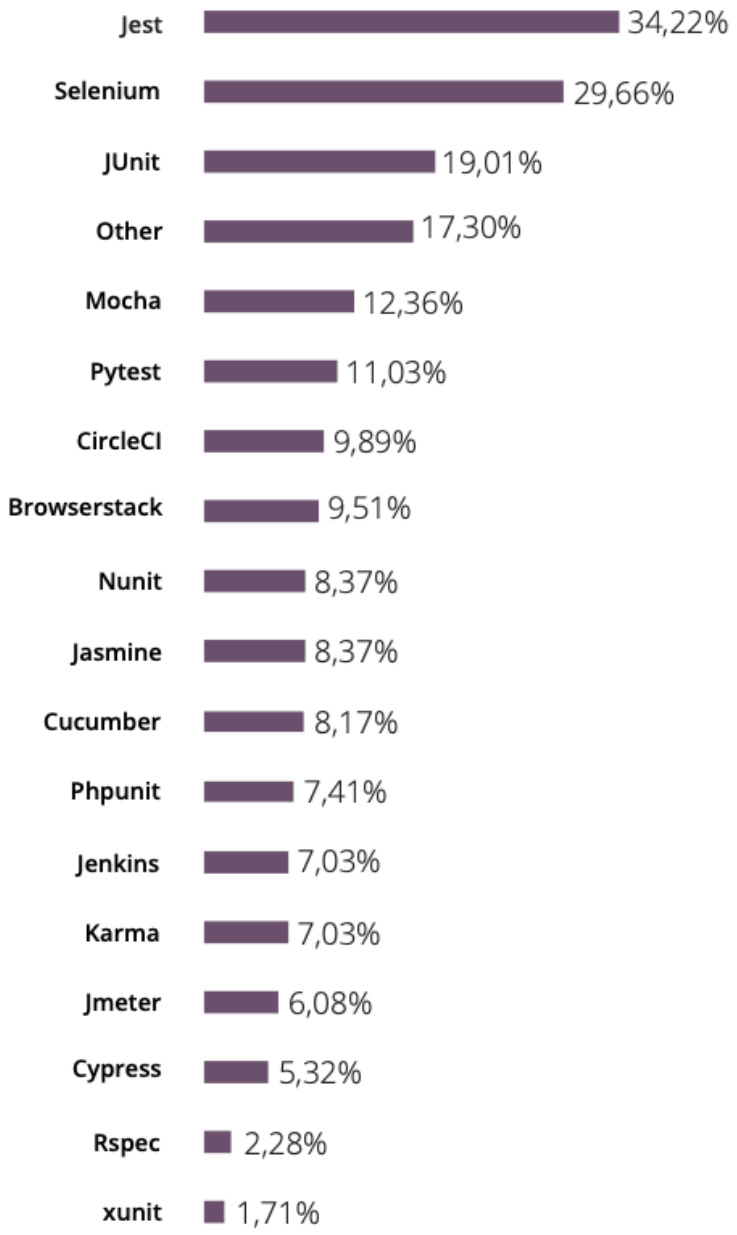


Do you use any of these agile software development methodologies (2020)?

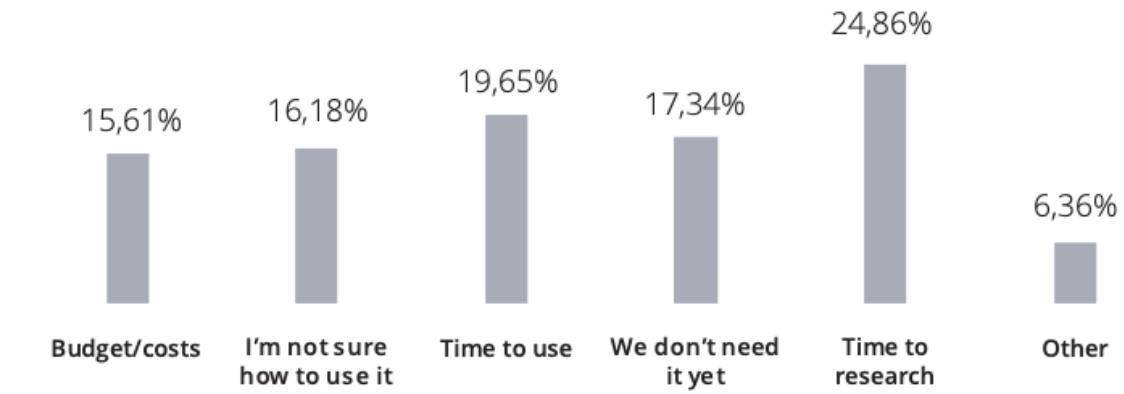

What are your company's primary programming languages?*
59,08% JavaScript
29,04% Java
26,32% TypeScript
23,89% Python
23,75% C#
18,88% PHP
8,87% C++
6,44% Ruby
4,86% Kotlin
4,43% Other
4,43% Go
3,43% C
3,29% Swift
2,29% Scala
1,29% Elixir
0,86% Perl
0,72% Dart
0,57% Erlang
0,57% Objective C
What new programming languages are you considering to use in the next 12 months?*
Perl 0,00%
Objective C 0,57%
C 1,00%
Scala 1,72%
Elixir 2,00%
Dart 2,15%
Ruby 2,58%
PHP 3,00%
Other 3,15%
C++ 3,29%
Swift 4,01%
C# 4,58%
Java 5,72%
Rust 5,87%
Kotlin 9,59%
JavaScript 12,88%
Go 14,59%
TypeScript 19,31%
Python 21,60%
Not considering any new languages
35,05%



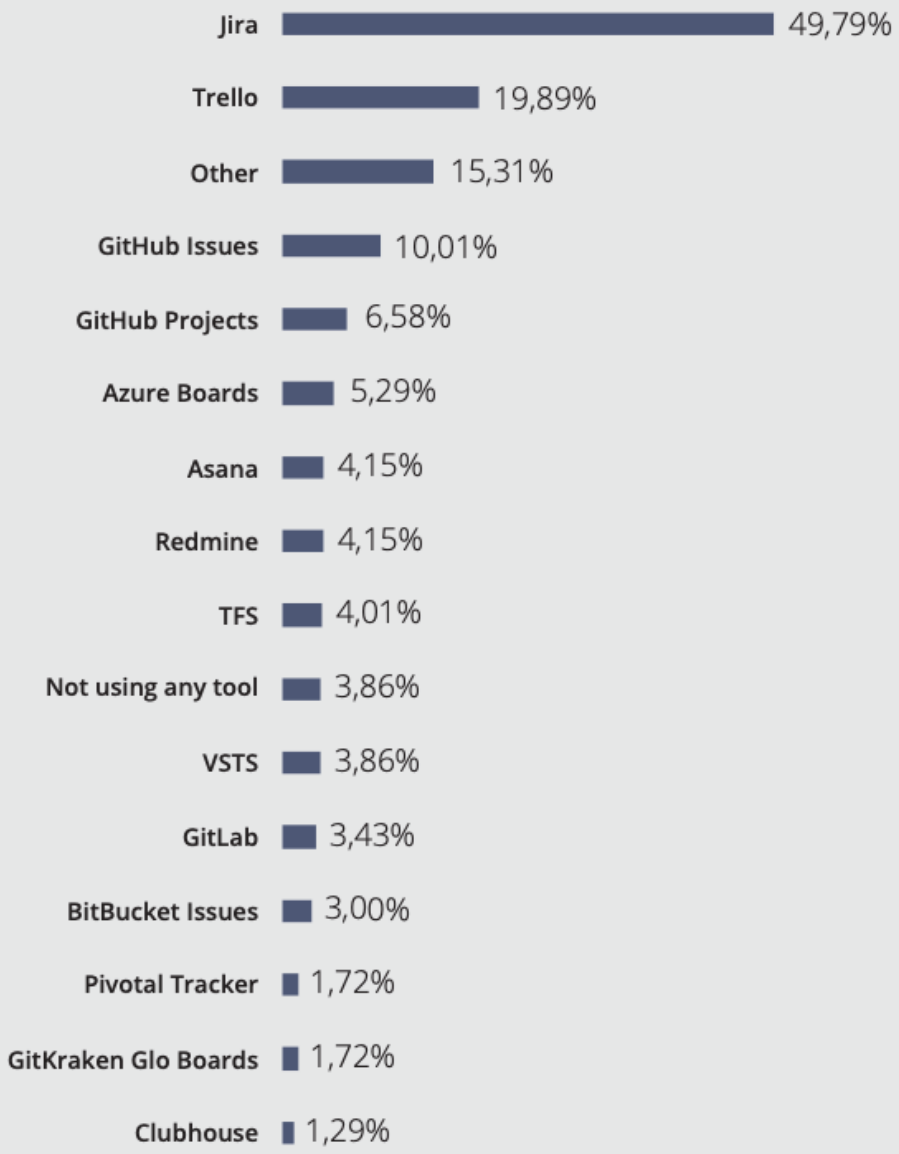
What tool(s) do you use for testing?



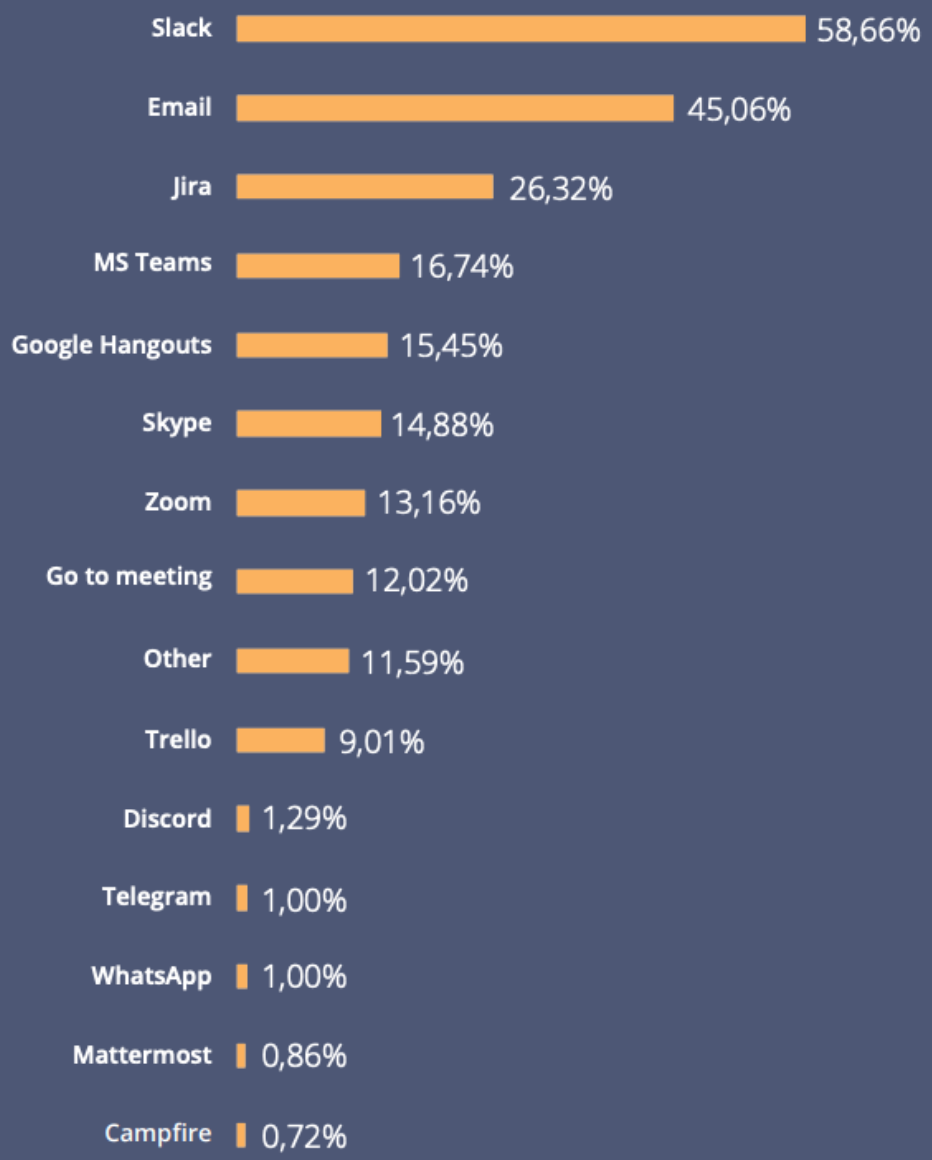
What are the factors that are limiting you from using a software testing tool?



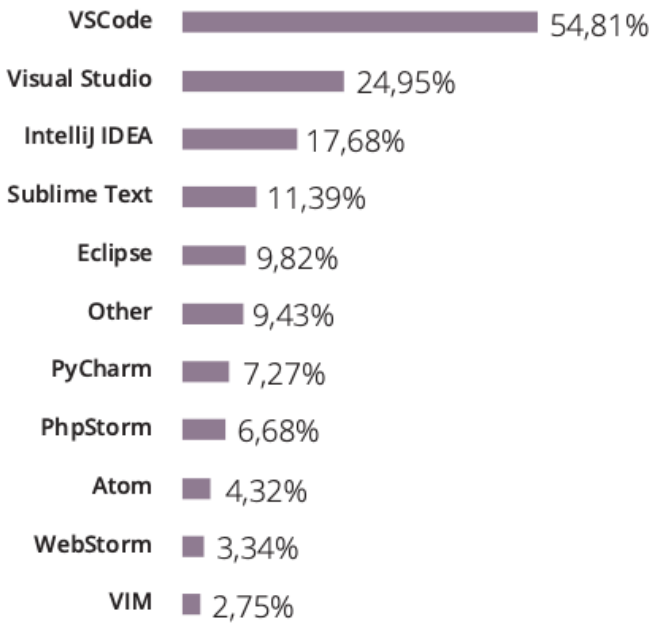
What tool do you use for project management?



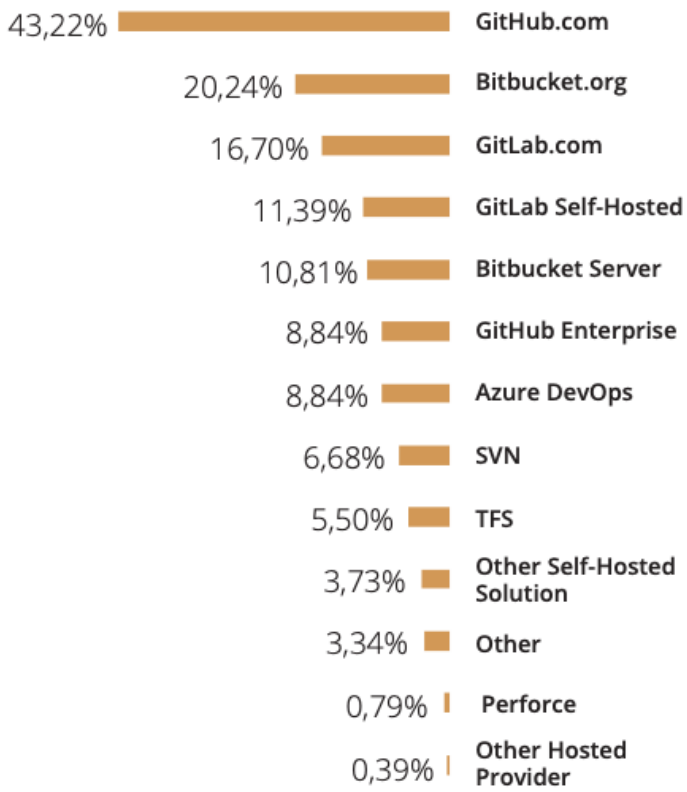
What tools do you use to communicate during a project?



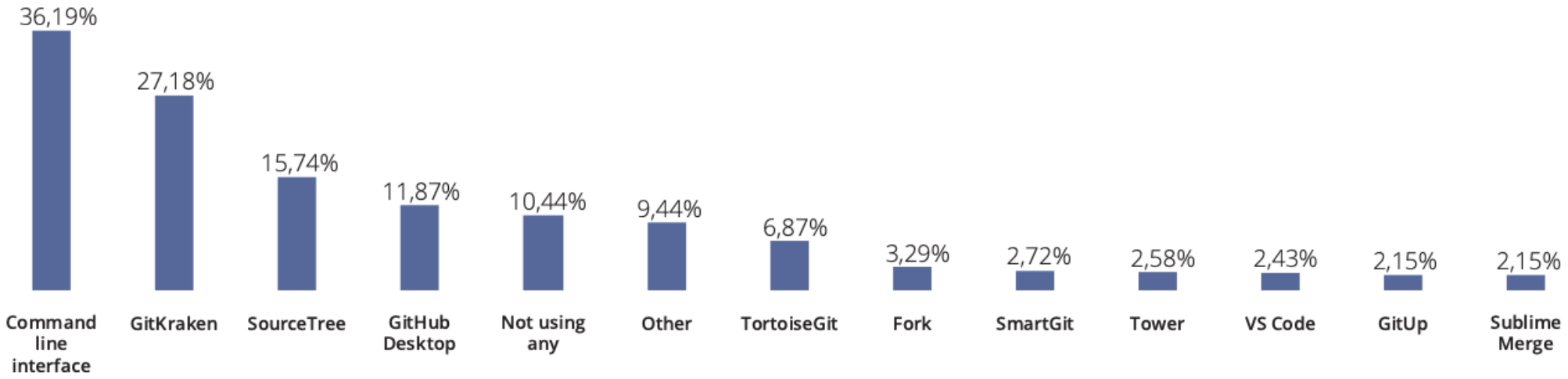
What IDE(s) do you use?



What version control system do you use?



What source control client(s) do you use?



What is the #1 cause of delivery problems for your team? (manager vs developer)