Open Source Numbers Everybody Should Know

A Data-Driven Portrait of New Trends in How We Build Software, Open Source, & What Even is "Entry-Level" Now

Heather Miller
@heathercmiller

BOBKonf, February 28th 2020, Berlin, Germany

institute for SOFTWARE RESEARCH
Carnegie Mellon University
School of Computer Science

OH HAI

A bit about me

Assistant Professor in the
School of Computer Science
@ CMU

OH HAI

A bit about me

Founded the Scala Center, 2016

scalacenter
For open source. For education.

Assistant Professor in the
School of Computer Science
@ CMU

OH HAI

A bit about me

Founded the Scala Center, 2016

PhD in Computer Science, 2015
EPFL under Martin Odersky

Assistant Professor in the
School of Computer Science
@ CMU

OH HAI

A bit about me

Founded the Scala Center, 2016

PhD in Computer Science, 2015
EPFL under Martin Odersky

Assistant Professor in the
School of Computer Science
@ CMU

scalacenter
For open source. For education.

Worked a lot on Scala

- Scala Futures
- Scala's concurrency libraries
- Typeclass derivation
- Lightweight type system extensions
- Programming models for distributed programming
- Coursera MOOCs

OH HAI

A bit about me

Founded the Scala Center, 2016

PhD in Computer Science, 2015
EPFL under Martin Odersky

Assistant Professor in the School of Computer Science @ CMU

Worked a lot on Scala

- Scala Futures
- Scala's concurrency libraries
- Typeclass derivation
- Lightweight type system extensions
- Programming models for distributed programming
- Coursera MOOCs

Joined CMU as an assistant prof in 2018

My research:

- bringing programming language techniques to dist. systems
- making microservice architectures more reliable
- distributed actor runtimes

Building a new lab at CMU

Doing stuff like making building microservice-based apps feel like you're programming in one language rather than 20. Building and formalizing language-level distributed and concurrent programming abstractions.

Assistant Professor in the
School of Computer Science

with some fine folks!

Chris Meiklejohn
@cmeik

Matthew Weidner

+ you?
We're always looking for new students!

Building a new lab at CMU

PLEASE COME CHAT WITH ME ABOUT THIS WORK!

Doing stuff like making building microservice-based apps feel like you're programming in one language rather than 20. Building and formalizing language-level distributed and concurrent programming abstractions.

Assistant Professor in the
School of Computer Science

with some fine folks!

Chris Meiklejohn
@cmeik

Matthew Weidner

+ you?
We're always looking for new students!

Just a bit more detail about some of our current projects...

Rethinking the mathematical formulation of CRDTs

Matthew Weidner

We present a new construction: semidirect product of op-based CRDTs, which combines the operations of two CRDTs into one while handling conflicts between their concurrent operations in a uniform way. **Composability for CRDTs!**

What if fault-injection was a sort of testing done at CI time?

Chris Meiklejohn
@cmeik

We present a novel testing methodology for distributed applications, called Resilience Driven Development (RDD). With RDD, developers first specify application behavior as integration tests. Then, a novel fine-grained fault injection approach that uses exhaustive search is used to find bugs in the application.

Just a bit more detail about some of our current projects...

Can we check global configurations of microservices before they're deployed?

How do ideas in open-source developer communities spread?

Data structures for federated machine learning

Just a bit more detail about current projects...

PLEASE COME CHAT WITH ME ABOUT THIS WORK!

Can we check global configurations of microservices before they're deployed?

How do ideas in open-source developer communities spread?

Data structures for federated machine learning

UM, SO WAIT

Then why are you talking about open source stuff?

The two hardest problems in computer science are:

– **Jeff Bigham**
@jeffbigham

The two hardest problems in computer science are: *(i)* people,

– **Jeff Bigham**
@jeffbigham

The two hardest problems in computer science are: *(i)* people, *(ii)*, convincing computer scientists that the hardest problem in computer science is people,

– **Jeff Bigham**
@jeffbigham

The two hardest problems in computer science are: *(i)* people, *(ii)*, convincing computer scientists that the hardest problem in computer science is people, and, *(iii)* off by one errors.

– **Jeff Bigham**
@jeffbigham

SCALA CENTER

I'm the founding director.

And suddenly my focus is 200% what is happening in open source Scala, and how we can keep growing our ecosystem, tools, and improve developer experience for anyone.

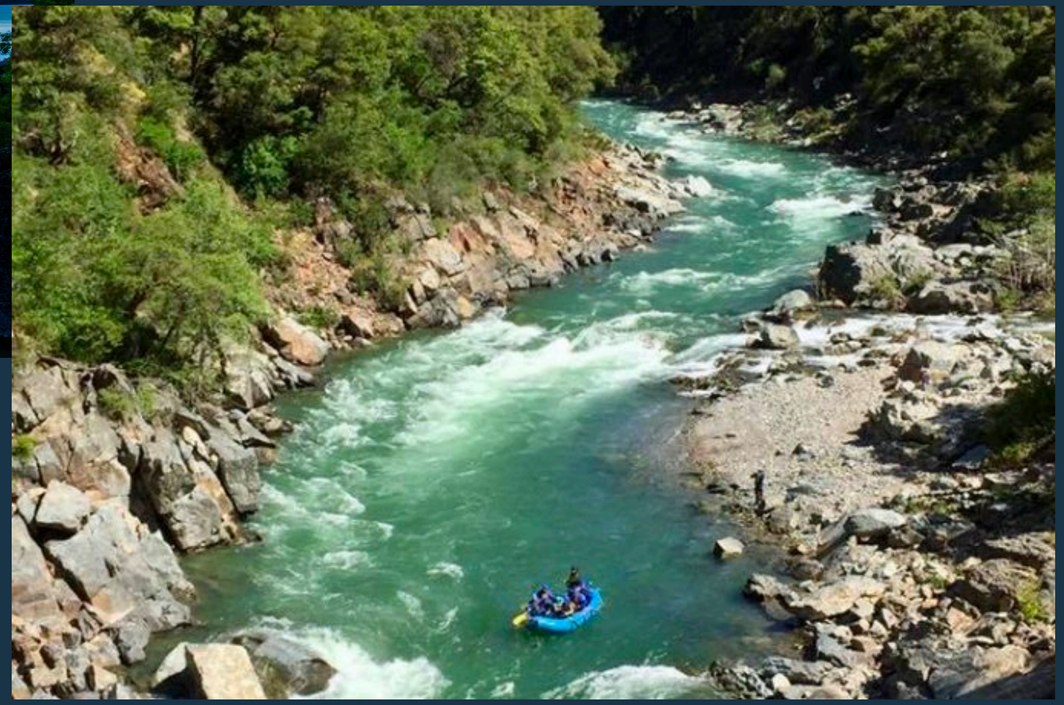
Not only people paying into the Scala Center. But anyone with an internet connection. A good developer experience should be free.

This shift in focus was eye-opening.

I quickly observed problems with the health of some of our core projects in our own ecosystem that was cause for concern.

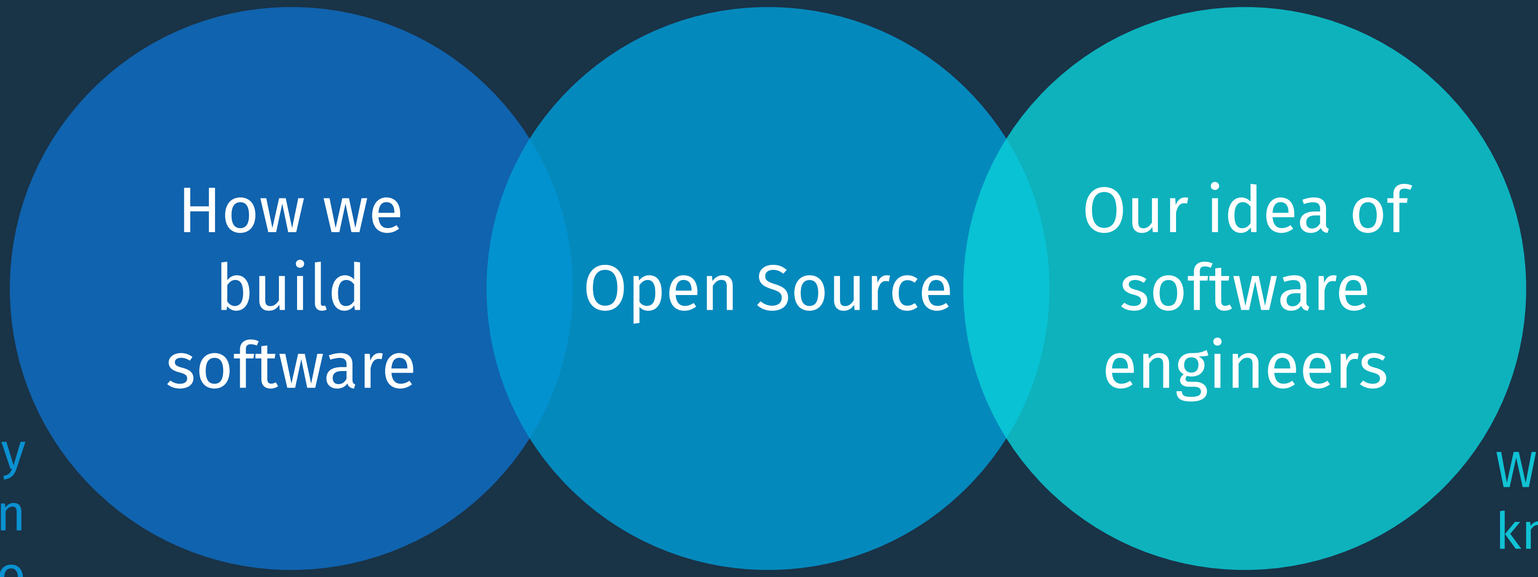
And what's worse, this trend is common throughout the open source community.

So I started collecting data on these fast-changing trends



Things that are changing *fast:*

& that more people should be aware of



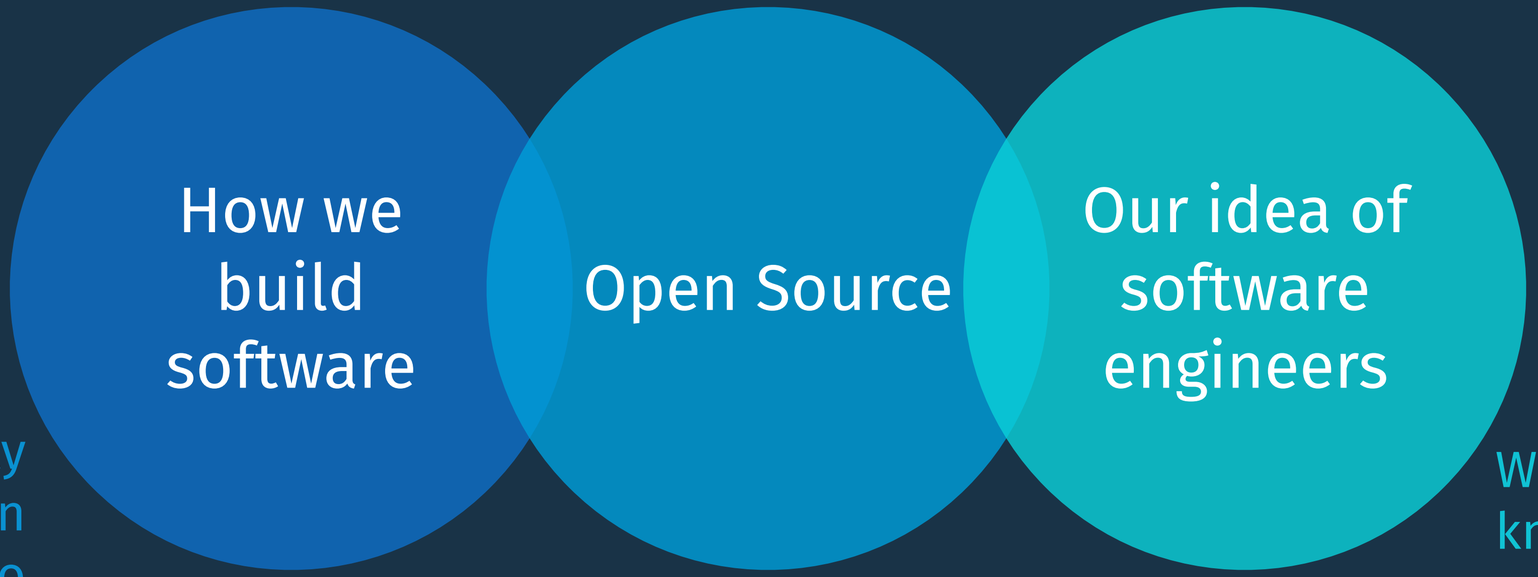
What we actually do nowadays when we sit down to build an app.

The common infrastructure and tools that we all depend on

What SWEs should know, how much experience they have, and who they are.

Things that are changing *fast:*

& that more people should be aware of



What we actually do nowadays when we sit down to build an app.

The common infrastructure and tools that we all depend on

What SWEs should know, how much experience they have, and who they are.

This talk will cover fast-changing trends in these 3 areas

FIRST,

How people are getting into tech is changing

We all already know that hiring is difficult

HOW PEOPLE ARE GETTING INTO TECH IS CHANGING

We all already know that hiring is difficult

Bloomberg

Technology

Demand for Programmers Hits Full Boil as U.S. Job Market Simmers

By Craig Torres
March 8, 2018, 12:00 AM EST

- Companies turn to homegrown apprentices to meet demand
- Mapbox finds success with women coders who refer friends

HOW PEOPLE ARE GETTING INTO TECH IS CHANGING

We all already know that hiring is difficult

Bloomberg

Technology

Demand for Programmers Hits Full Boil as U.S. Job Market Simmers

By Craig Torres
March 8, 2018, 12:00 AM EST

- Companies turn to homegrown apprentices to meet demand
- Mapbox finds success with women coders who refer friends

LIVE
Watch
Listen

HACKERNOON

Subscribe

Crypto / Blockchain | Coding | A.I. | Futurism | Startups | About | Podcast | Community

2018's Software Engineering Talent Shortage— It's quality, not just quantity

November 12th 2017

TWEET THIS

Forrester projects that firms will pay 20% above market for quality engineering talent in 2018

HOW PEOPLE ARE GETTING INTO TECH IS CHANGING

We all already know that hiring is difficult

Bloomberg

Technology

Demand for Programmers Hits Full Boil as U.S. Job Market Simmers

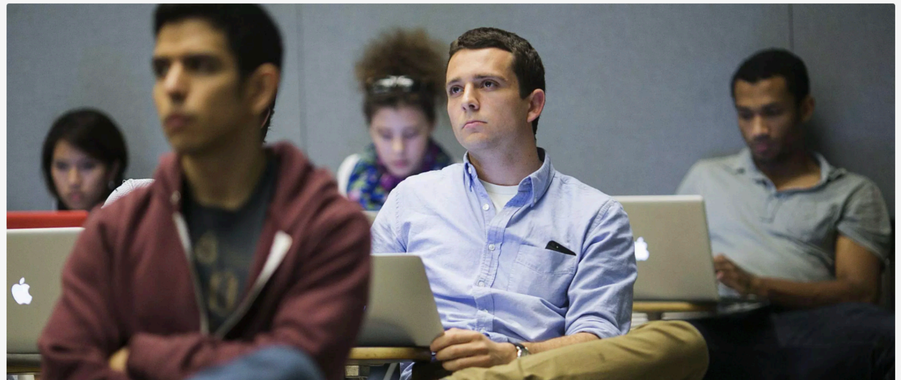
By Craig Torres
March 8, 2018, 12:00 AM EST

QUARTZ

GOOD JOB ODDS

You probably should have majored in computer science

By Sarah Kessler • March 10, 2017



LIVE
Watch
Listen

HACKERNOON

Subscribe

Crypto / Blockchain | Coding | A.I. | Futurism | Startups | About | Podcast | Community

2018's Software Engineering Talent Shortage— It's quality, not just quantity

November 12th 2017

TWEET THIS

Forrester projects that firms will pay 20% above market for quality engineering talent in 2018

There’s a shortage of tech workers

Subsequent slides focus on the US, but I’m going to start by saying that this is a problem in Germany too:

HOW PEOPLE ARE GETTING INTO TECH IS CHANGING

There's a shortage of tech workers

Data from May 2019

2019

8%

of jobs in Germany destined to developers

17% of all available jobs in Dresden are for developers

Data from Taledo https://www.taledo.com/blog/job-search/developer-job-germany-infographic

HOW PEOPLE ARE GETTING INTO TECH IS CHANGING

There's a shortage of tech workers

Data from May 2019

2019

8%

of jobs in Germany destined to developers

17% of all available jobs in Dresden are for developers

Open developer jobs by major city	# open positions
Berlin	4,754
Munich	4,601
Hamburg	2,749
Frankfurt	2,202
Stuttgart	1,989
Cologne	1,728
Nuremburg	1,418

Data from Taledo https://www.taledo.com/blog/job-search/developer-job-germany-infographic

HOW PEOPLE ARE GETTING INTO TECH IS CHANGING

There's a shortage of tech workers

Data from May 2019

2019

8%

of jobs in Germany destined to developers

17% of all available jobs in Dresden are for developers

Open developer jobs by major city	# open positions
Berlin	4,754
Munich	4,601
Hamburg	2,749
Frankfurt	2,202
Stuttgart	1,989
Cologne	1,728
Nuremburg	1,418

Open developer jobs by state	# open positions
Bavaria	10,018
Baden-Württemberg	7,820
North Rhine-Westphalia	6,915
Berlin	4,789
Hesse	3,836
Hamburg	2,750
Lower Saxony	2,130

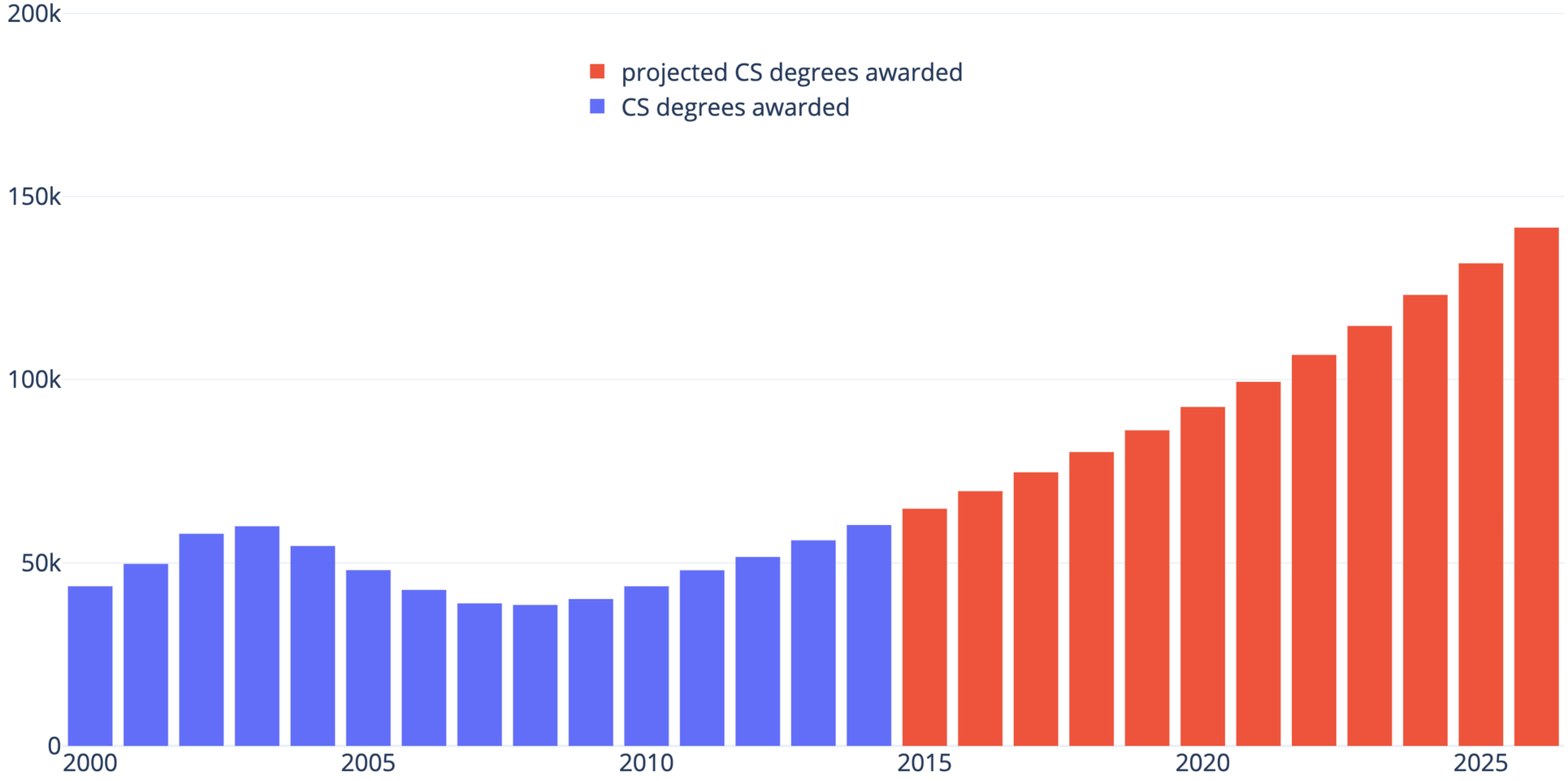
Data from Taledo https://www.taledo.com/blog/job-search/developer-job-germany-infographic

There’s a shortage of tech workers

*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

Students graduating with CS/IT-related bachelor’s degrees



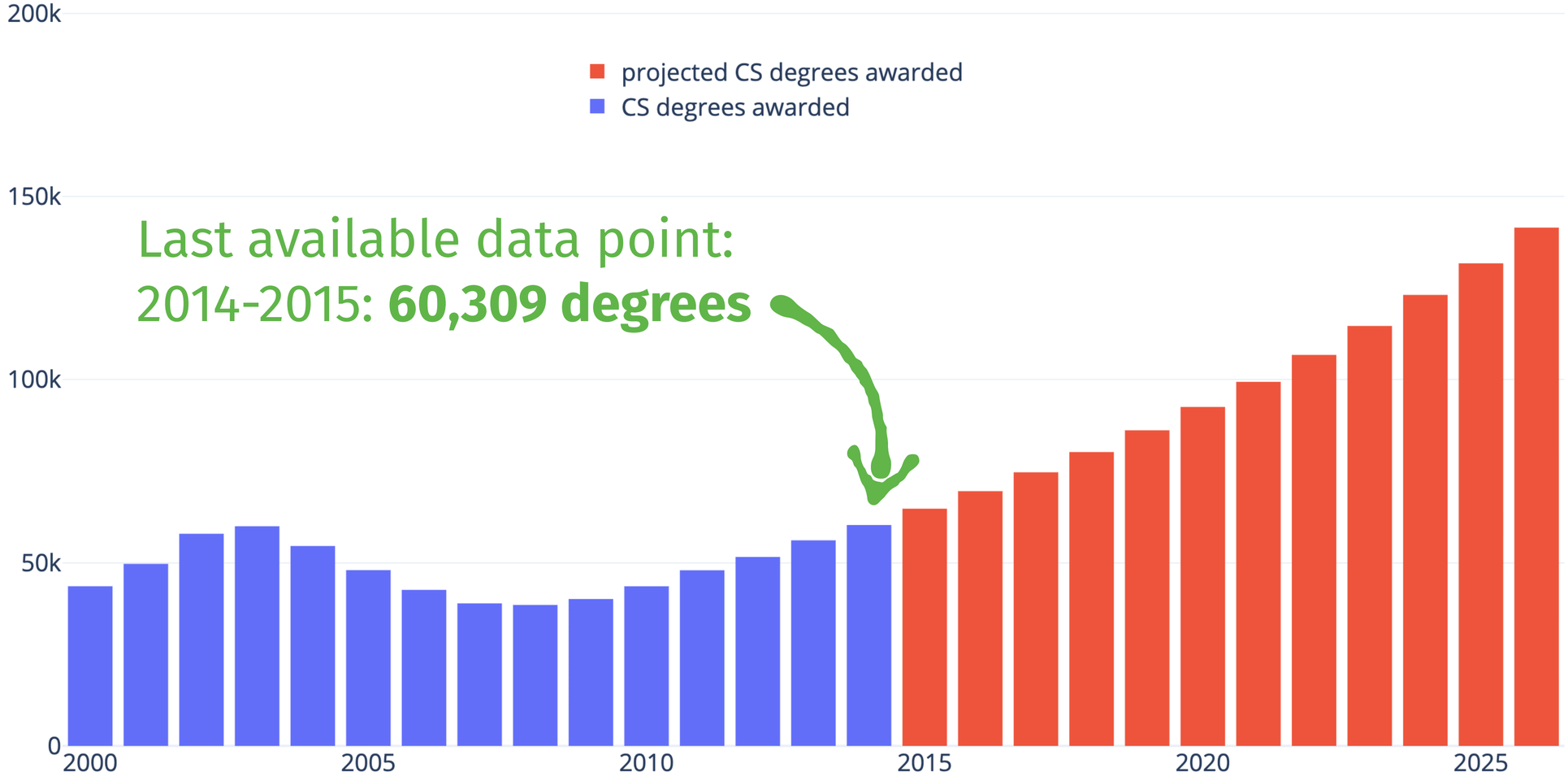
*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

There's a shortage of tech workers

*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

Students graduating with CS/IT-related bachelor's degrees



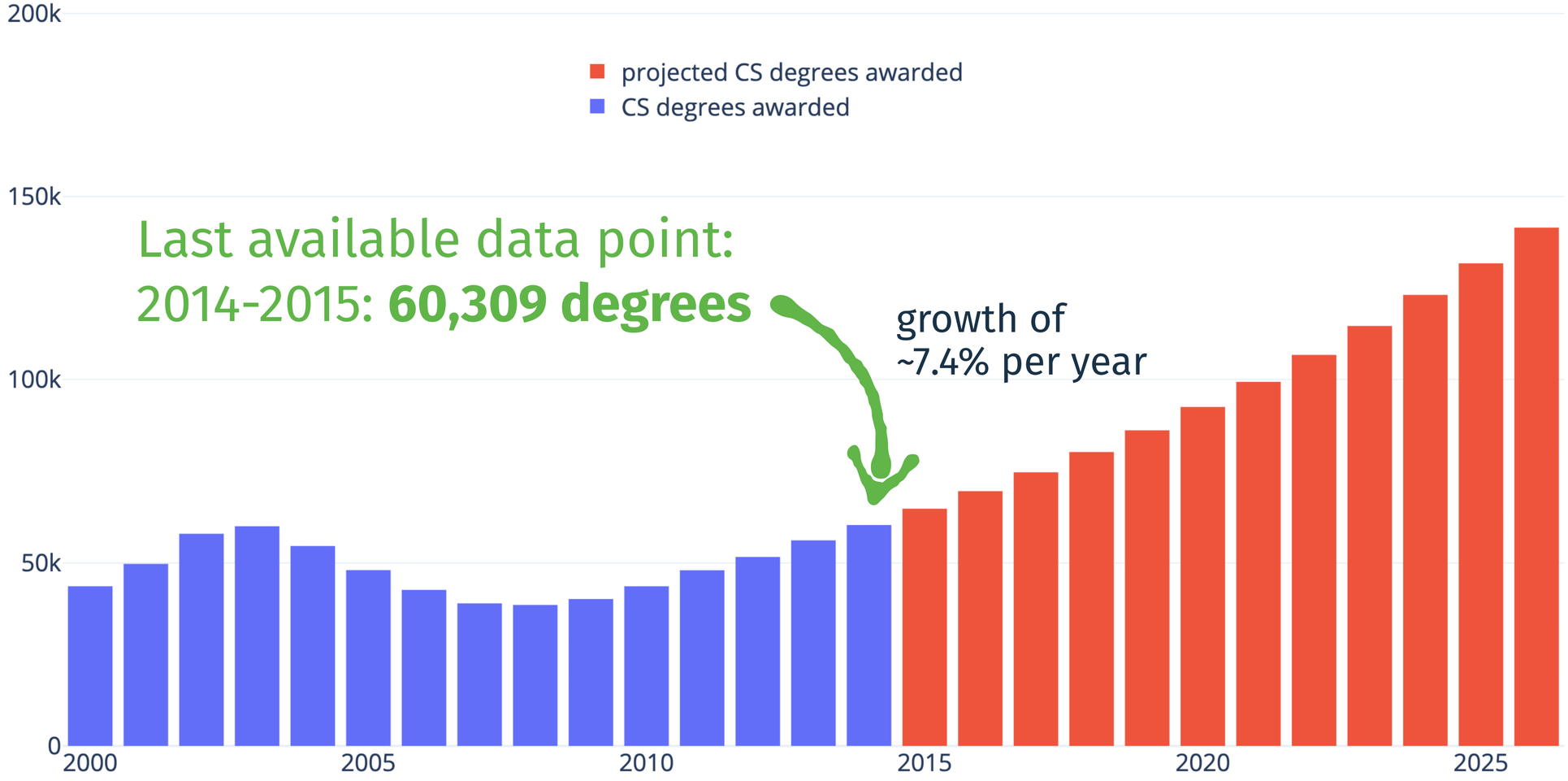
*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

There's a shortage of tech workers

*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

Students graduating with CS/IT-related bachelor's degrees



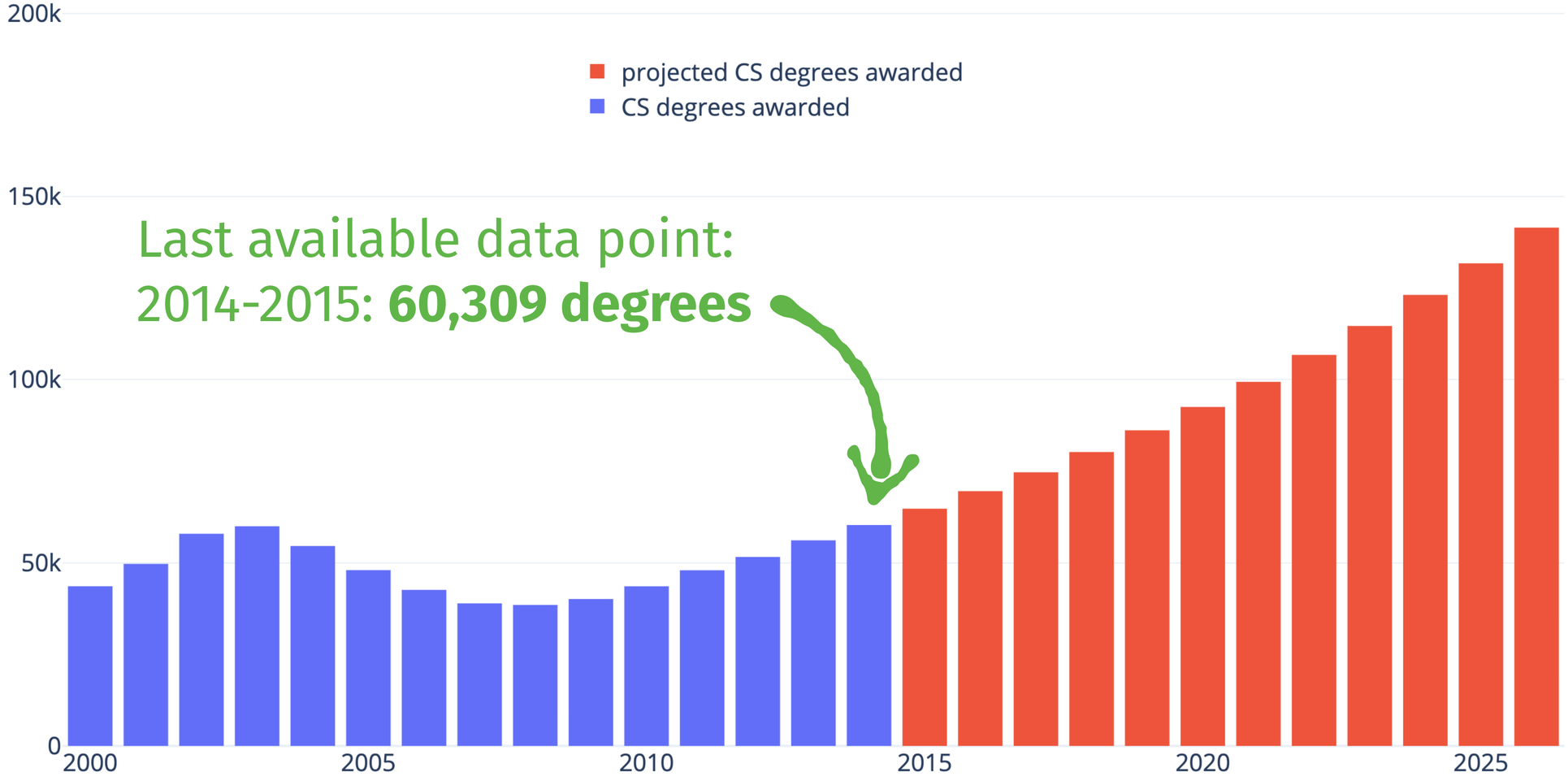
*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

There's a shortage of tech workers

*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

Students graduating with CS/IT-related bachelor's degrees



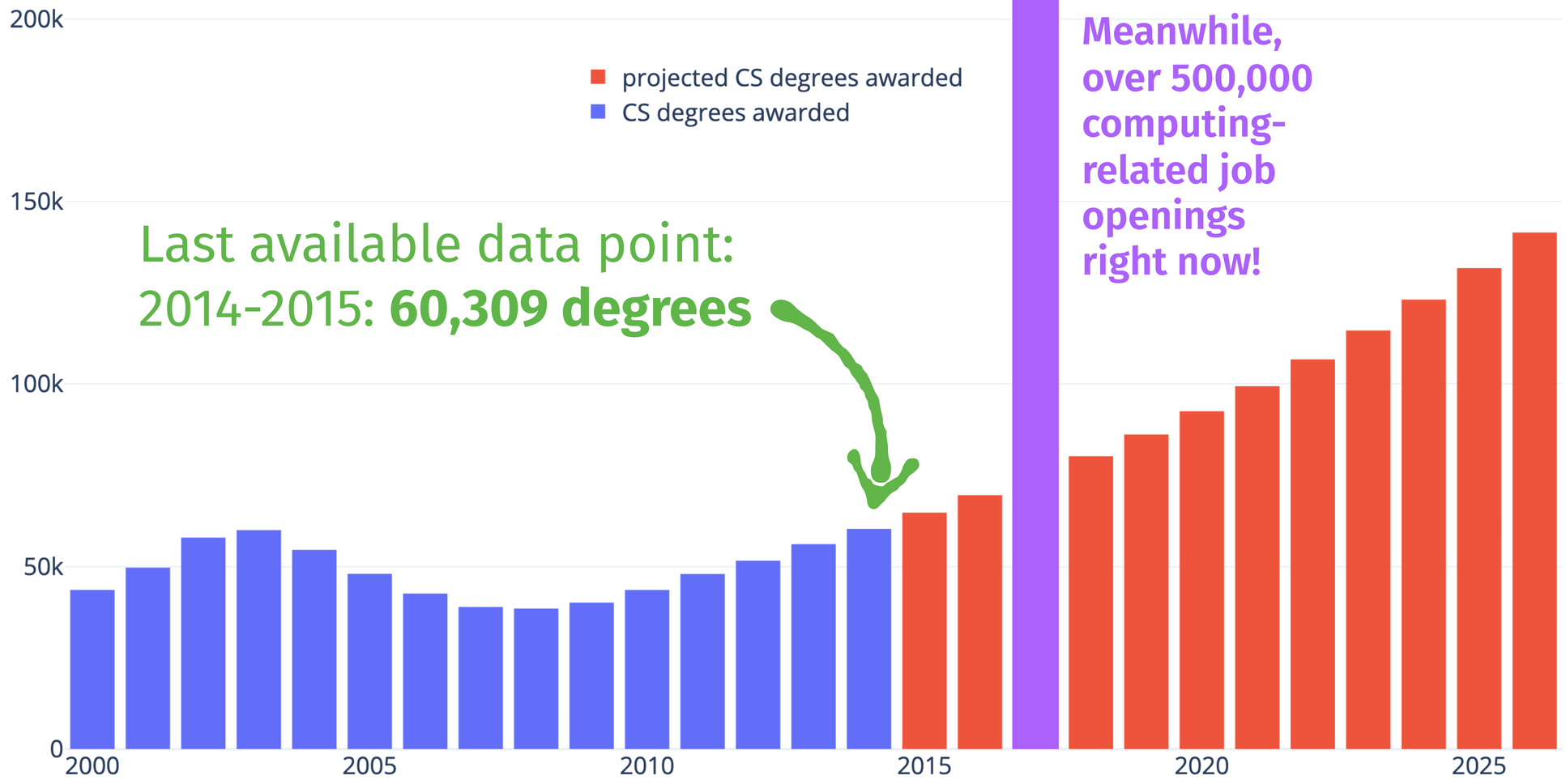
*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

There's a shortage of tech workers

*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

Students graduating with CS/IT-related achelor's degrees

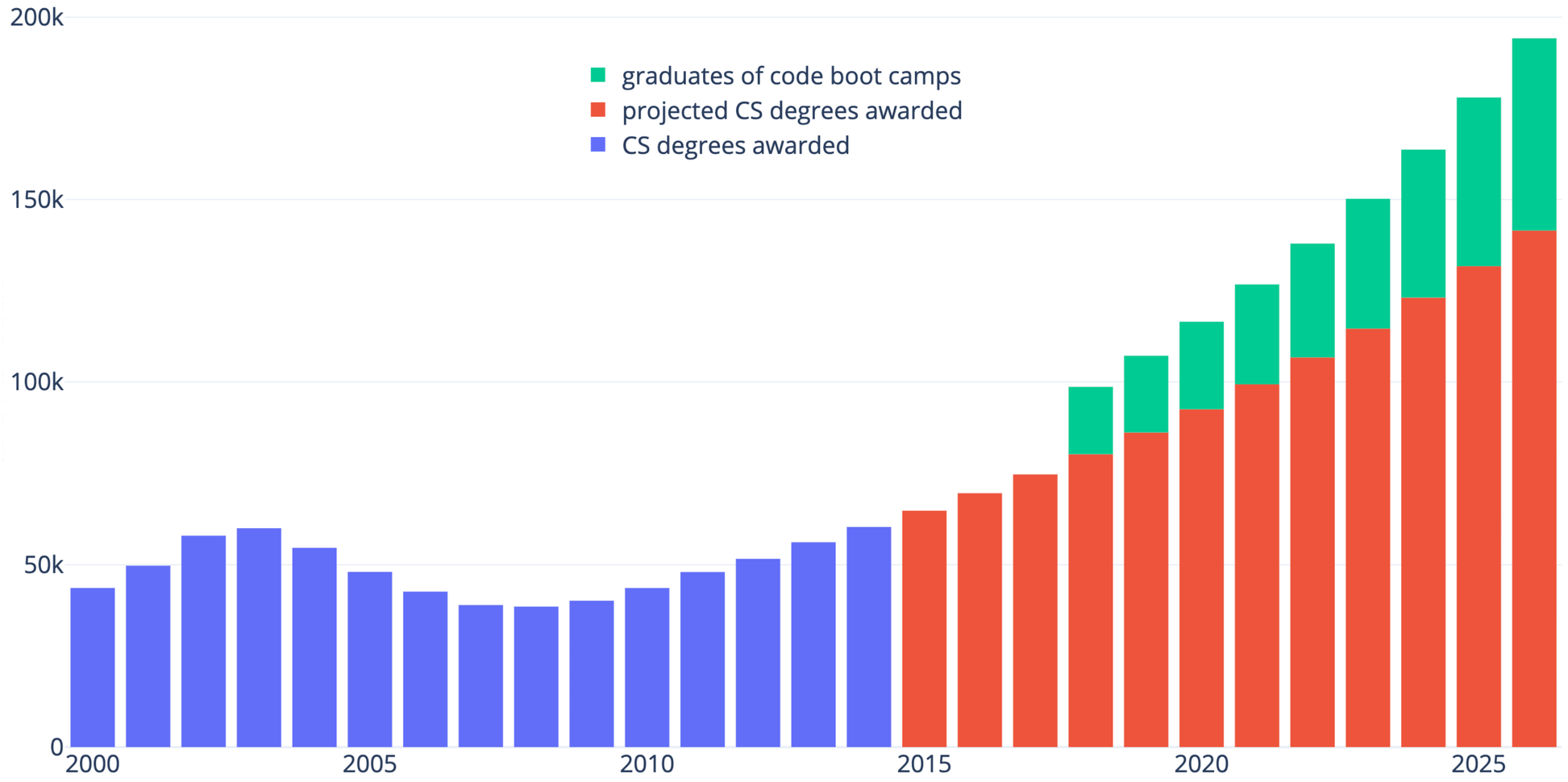


*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

There's still a shortage of tech workers, when you include code bootcamps too

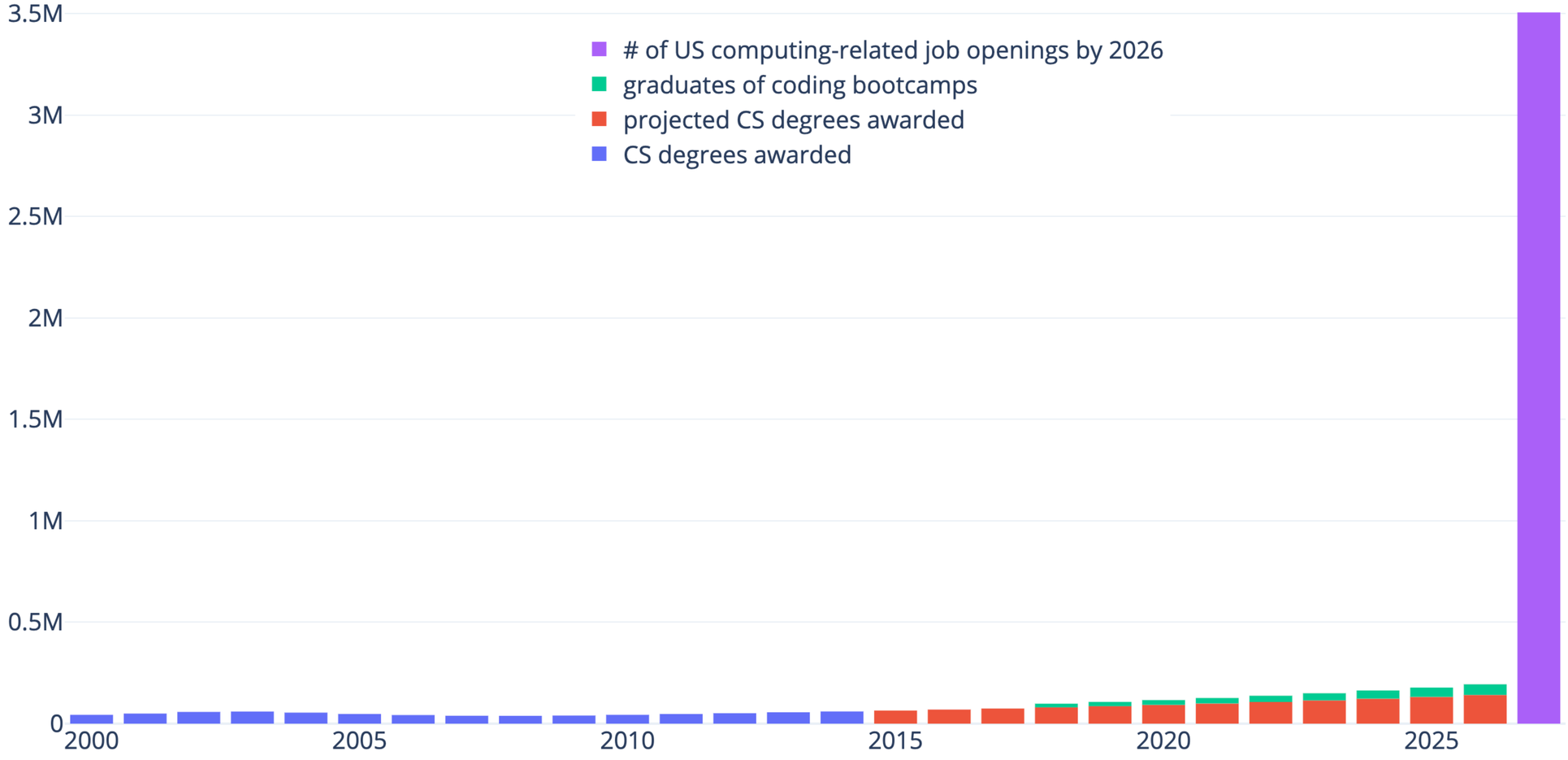
*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf



*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

By 2026, there will be 3.5 million computing-related job openings*

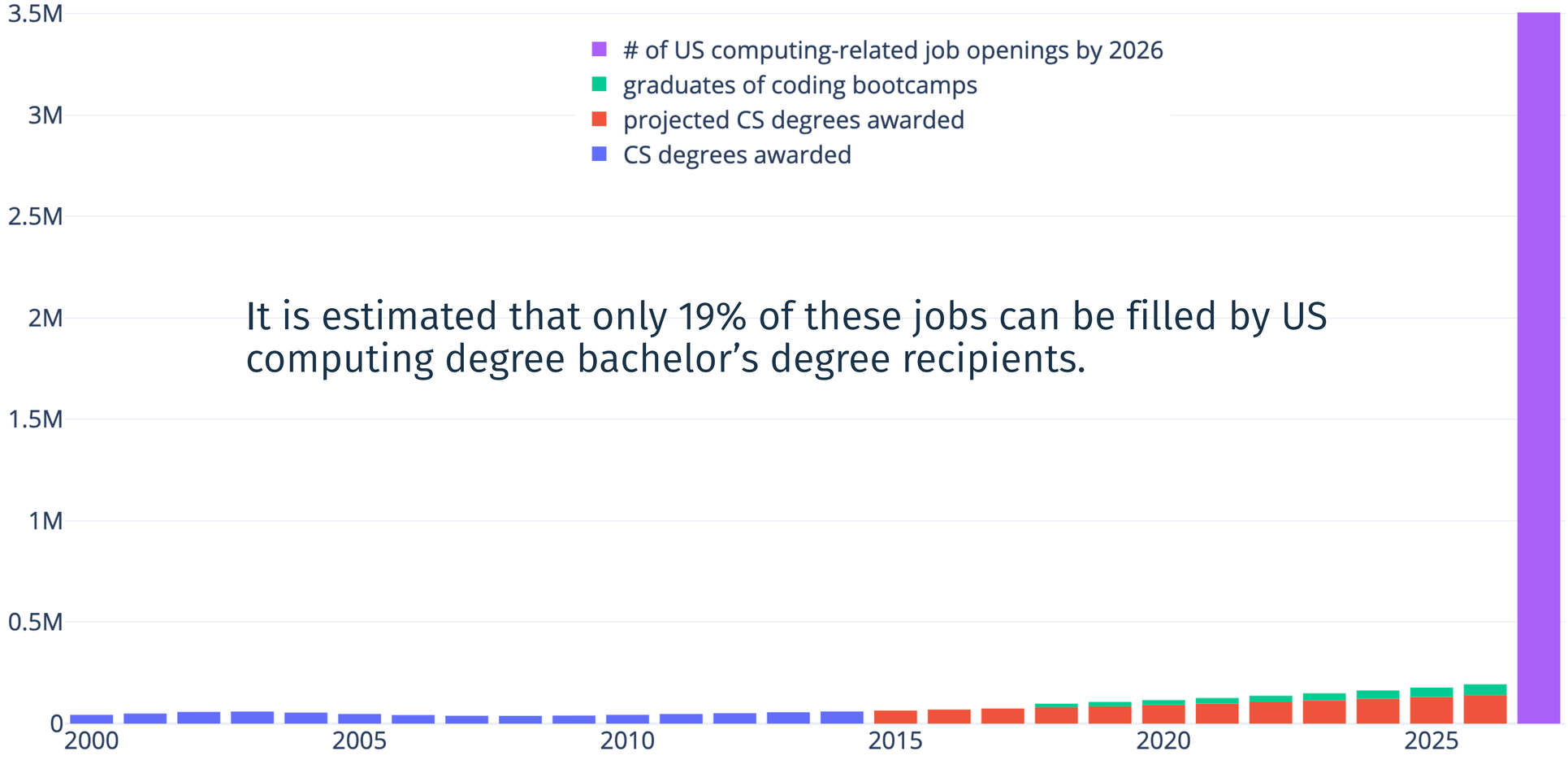


*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

By 2026, there will be 3.5 million computing-related job openings*



It is estimated that only 19% of these jobs can be filled by US computing degree bachelor's degree recipients.

*note, this is a US-centric view!

https://www.ncwit.org/sites/default/files/resources/btn_05092019_web.pdf

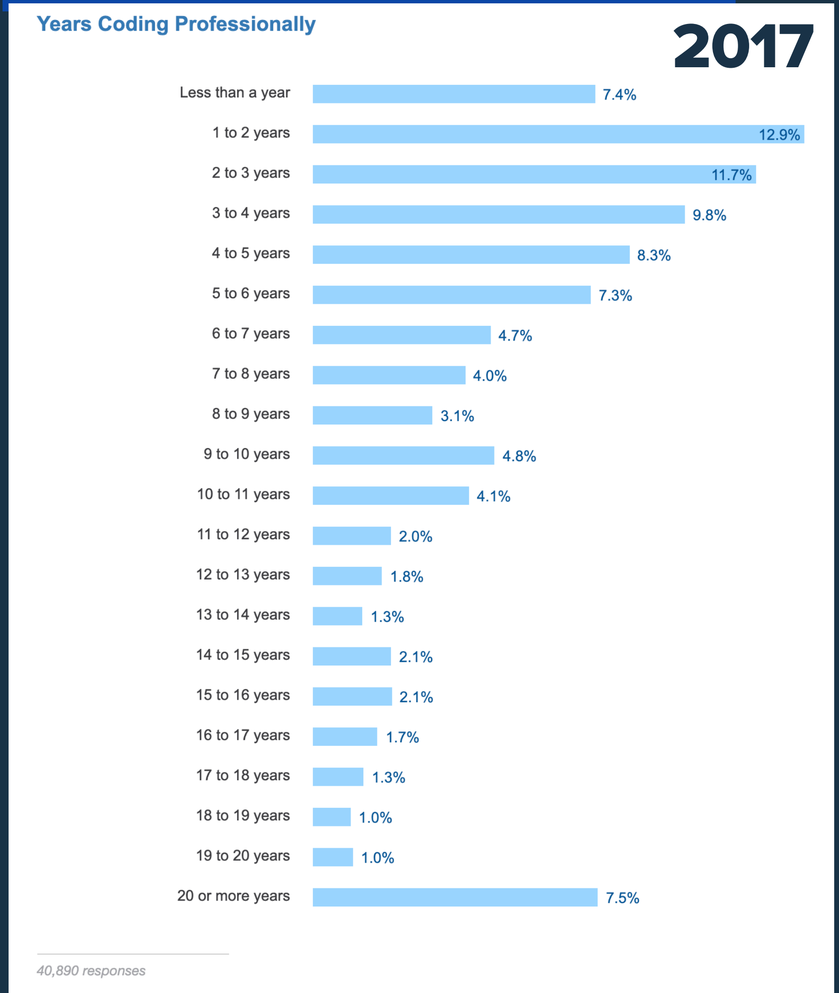
*Department of Labor Statistics, Employment Projections (Occupational Category: 15-1100) Includes new and replacement jobs and assumes current undergraduate degree (CIP 11) production levels persist

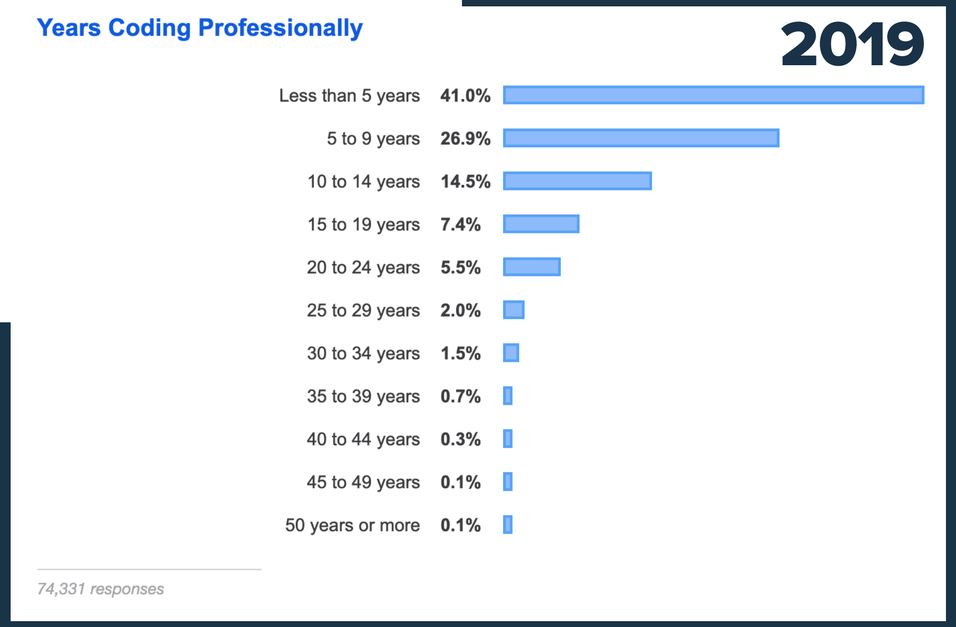
THE TECH WORKERS WE HAVE...

A large portion of professional developers are new

stackoverflow

Developer Survey Results



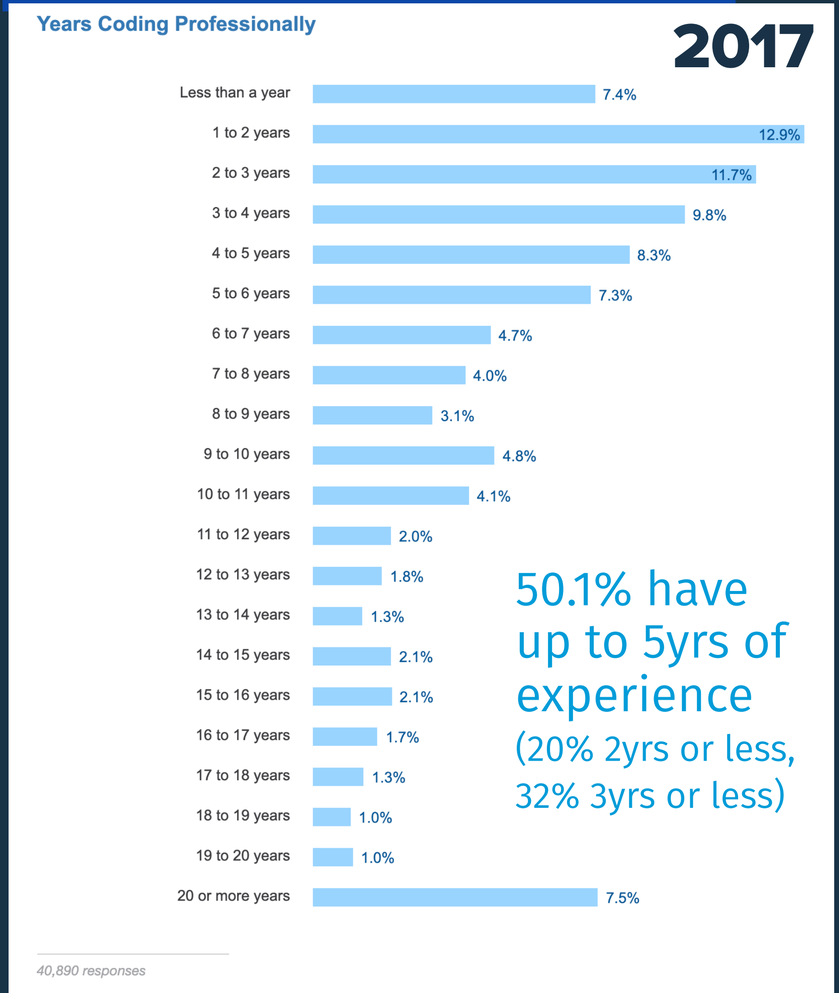


THE TECH WORKERS WE HAVE...

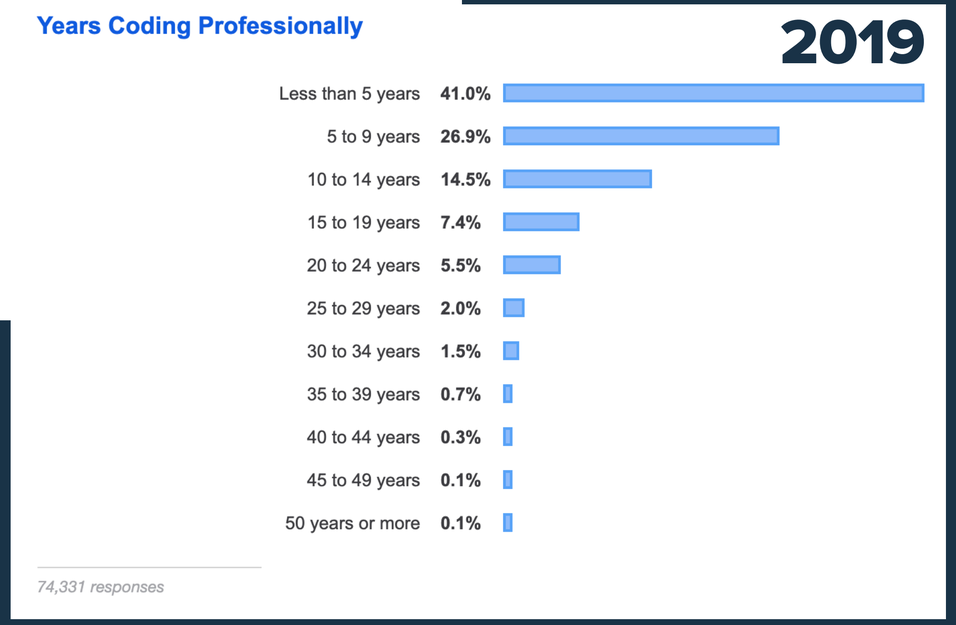
A large portion of professional developers are new

stackoverflow

Developer Survey Results



50.1% have up to 5yrs of experience
(20% 2yrs or less, 32% 3yrs or less)

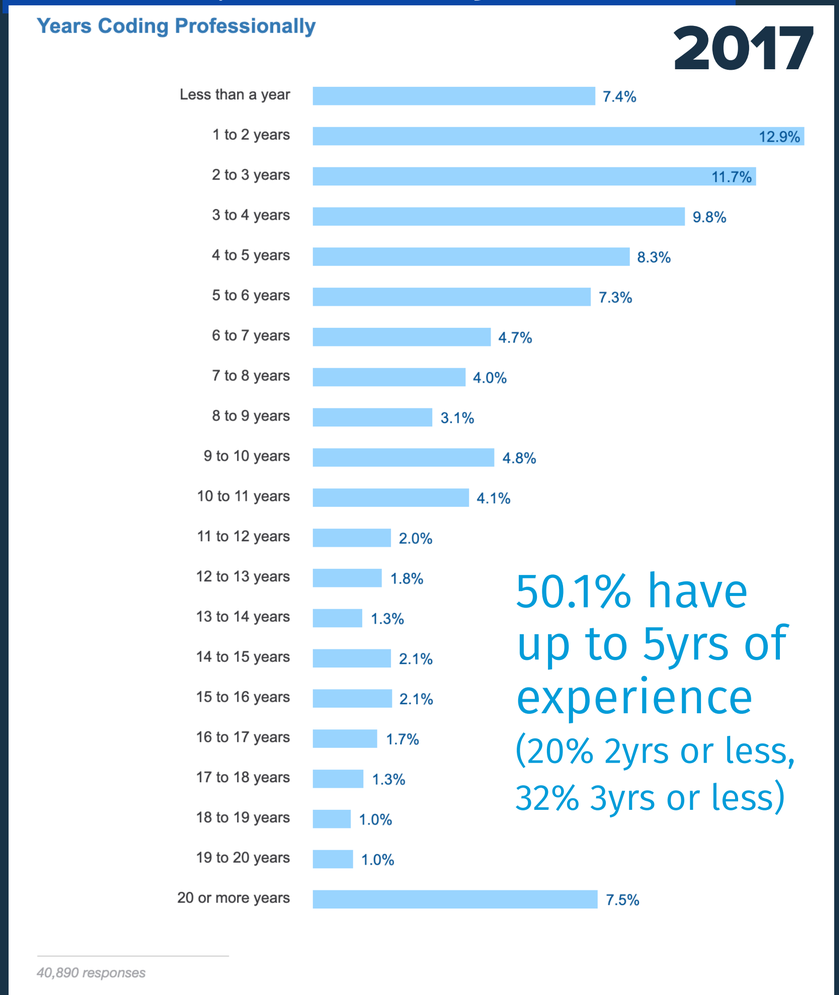


THE TECH WORKERS WE HAVE...

A large portion of professional developers are new

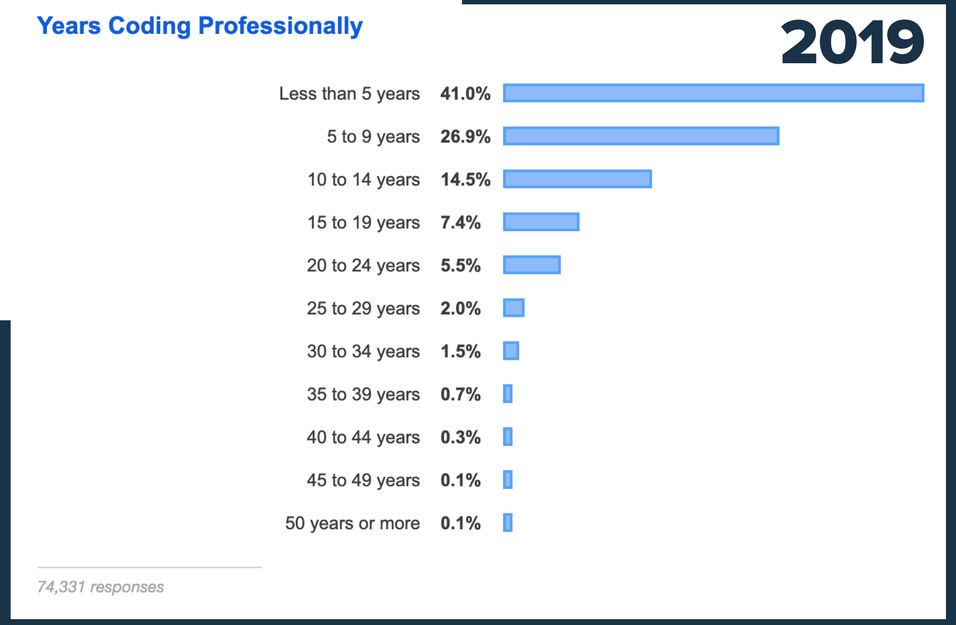
stackoverflow

Developer Survey Results



50.1% have up to 5yrs of experience
(20% 2yrs or less, 32% 3yrs or less)

57.5% have up to 5yrs of experience



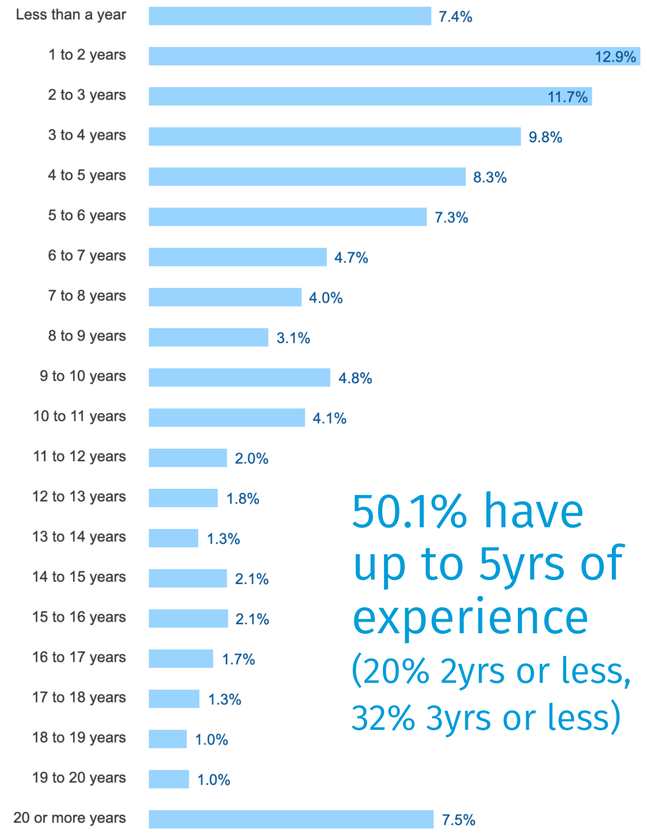
THE TECH WORKERS WE HAVE...

A large portion of professional developers are new

stackoverflow

Developer Survey Results

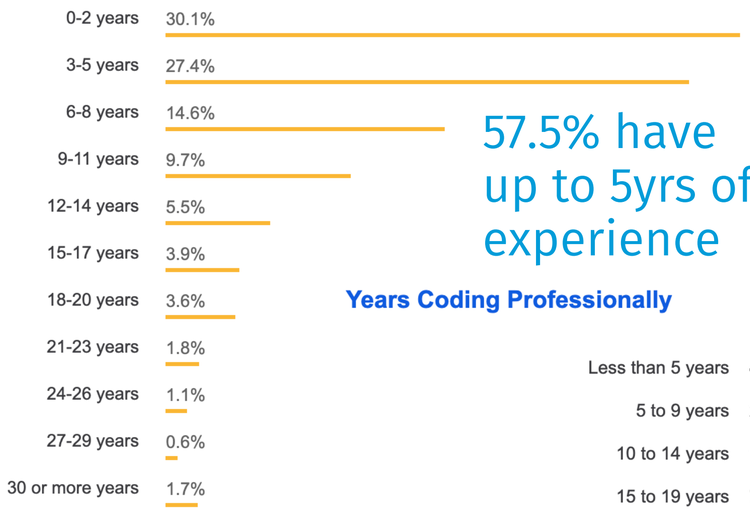
Years Coding Professionally — **2017**



50.1% have up to 5yrs of experience
(20% 2yrs or less, 32% 3yrs or less)

40,890 responses

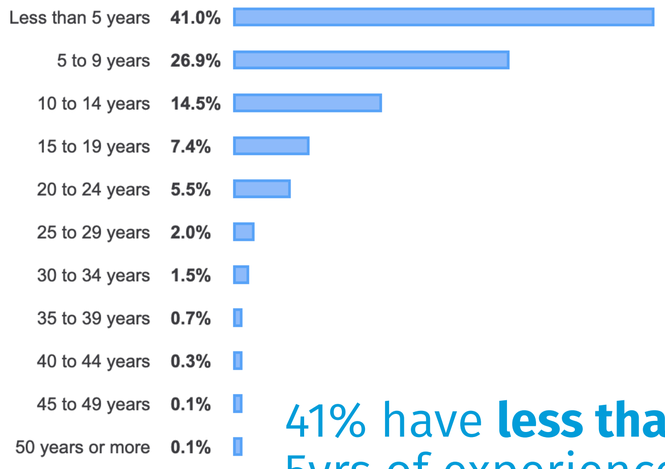
Years Coding Professionally — **2018**



57.5% have up to 5yrs of experience

77,903 responses

Years Coding Professionally — **2019**



41% have **less than** 5yrs of experience

74,331 responses

TAKEAWAY:

We need to adapt, culturally, to make room for lots more newcomers

The demand for developers is just going to get more and more ridiculous.

The years of experience of practicing SWEs is dropping overall.

There's a tidal wave of newcomers entering our profession, and it's not going to slow down.

It's going to pick up speed.

TAKEAWAY:

We need to adapt, culturally, to make room for lots more newcomers

The demand for developers is just going to get more and more ridiculous.

The years of experience of practicing SWEs is dropping overall.

There's a tidal wave of newcomers entering our profession, and it's not going to slow down.

It's going to pick up speed.

What do we do?

"New frameworks are lowering the barrier to entry," Caleb Fristoe (founder of CodeTN) says; that's a far cry from the days when you had to learn the syntax of several programming languages to build useful software. "Rather than typing these seven lines of code to get a menu to pop down, you just download the framework from a code base that allows you to do that in a simpler way," he explains. "Frameworks are taking the hard work that developers prided themselves on out of the equation."

The New Jobs
By Marina Krakovsky
Communications of the ACM, January 2018, Vol. 61 No. 1, Pages 21-23
10.1145/3157077

https://cacm.acm.org/magazines/2018/1/223883-the-new-jobs/fulltext

TAKEAWAY:

Existing devs are burning out

"Unable to fill tech vacancies, employers shuffle off additional duties to current employees, which leads to burnout and has a negative impact on local business development. Over 30% of respondents surveyed by Indeed admit that this issue accelerates staff turnover."

US Tech Talent Shortage in Numbers
March 26, 2019
https://www.daxx.com/blog/development-trends/software-engineer-shortage-us-2019

"With companies unable to fill open positions, current employees are expected to fill the gaps. In many cases this results in employee turnover. Over a third of respondents we surveyed (36%) said the lack of timely hiring has caused burnout in existing employees and affected their businesses."

Is the Tech Talent War Hurting Innovation? Hiring Managers and Tech Recruiters Respond
December 5, 2016
http://blog.indeed.com/2016/12/05/impact-of-tech-talent-shortage/

TAKEAWAY:

Obviously, increased diversity would help

We know that the people who develop software are not a representative of sample of society. Making more underrepresented minorities at home in tech is an obvious solution to increasing our numbers.

But also...

Immigration = good, more tech workers

Remote workers = good, more tech workers

*note, this is a US-centric view!

TAKEAWAY:

WE ACTUALLY HAVE TO DO SOMETHING

WE PROBABLY NEED TO BE BETTER HUMANS TO THOSE AROUND US.

WE PROBABLY ACTUALLY ALL NEED TO GET GOOD AT MENTORING ONE ANOTHER.

*again, this is a US-centric view!

TAKEAWAY:

WE ACTUALLY HAVE TO DO SOMETHING

WE PROBABLY NEED TO BE BETTER HUMANS TO THOSE AROUND US.

WE PROBABLY ACTUALLY ALL NEED TO GET GOOD AT MENTORING ONE ANOTHER.

But also...

Diversity = great, more of the population can be tech workers

Immigration = good, more tech workers

Remote workers = good, more tech workers

*again, this is a US-centric view!

A QUICK ASIDE...

We need to care about diversity for more than economics alone.

PAPER: Gender and tenure diversity in GitHub teams.

Increased diversity = increased productivity

Research from one of my colleagues:

There is evidence that software teams that are more diverse are more productive.

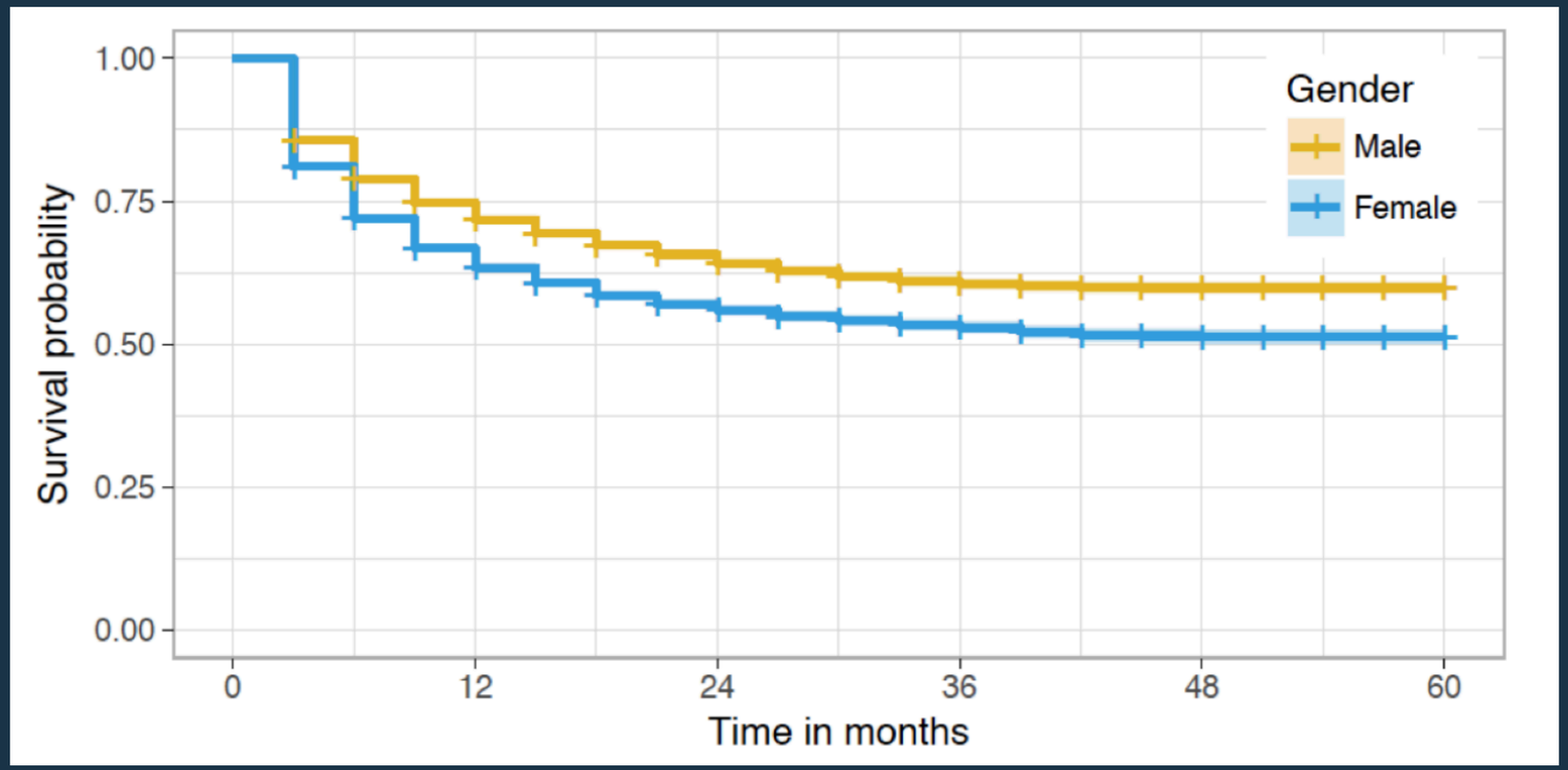
Holding other confounds fixed, teams that are more diverse with respect to gender and/or tenure/ experience tend to write code faster than teams that are less diverse.

Gender and tenure diversity in GitHub teams. Vasilescu, B., Posnett, D., Ray, B., Brand, M.G.J. van den, Serebrenik, A., Devanbu, P., and Filkov, V. CHI 2015

PAPER: The Impact of Social Capital on Sustained Participation in Open Source

How do we stop people from disengaging?

Women disengage earlier than men:



Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source. *Qiu, H.S., Nolte, A., Brown, A., Serebrenik, A., and Vasilescu, B. ICSE 2019*

Carnegie Mellon University
School of Computer Science

STRUDEL

PAPER: The Impact of Social Capital on Sustained Participation in Open Source

How do we stop people from disengaging?

"I have used a fake GitHub handle [...] so that people would assume I was male"

Article

new media & society

'Patches don't have gender' What is not open in open source software

Dawn Nafus
Intel Labs, USA

Abstract
While open source software development promise
software production often compared to a gift econo
than other forms of software production. The speci
openness in everyday practice exacerbates the exclus
construct that affects more than intellectual prope
ideas about authorship, agency, and the circumstan
can and cannot be exchanged. While open source deve
to the social, notions of openness tie the social to th
one another and relieving them of obligations that m
forms of gift exchange. In doing so, men monopoliz
de-legitimize the kinds of social ties necessary to bu

Perceptions of Diversity on G

Bogdan Vasilescu
University of California, Davis
vasilescu@ucdavis.edu

Vladimir Filkov
University of California, Da
filkov@cs.ucdavis.edu

***Abstract*—Understanding one's work environment is important for one's success, especially when working in teams. In virtual collaborative environments this amounts to being aware of the technical and social attributes of one's team members. Focusing on Open Source Software teams, naturally very diverse both socially and technically, we report the results of a user survey that tries to resolve how teamwork and individual attributes are perceived by developers collaborating on GitHub, and how those perceptions influence their work. Our findings can be used as complementary data to quantitative studies of developers' behavior on GitHub.**

I. INTRODUCTION

Software development is technical and knowledge-intensive, but also human-centric and collaborative, benefiting from the social attributes of the people involved. Open Source Software (OSS) communities, in particular, tend to be quite diverse, with contributors ranging from professional developers to volunteers, all with varied personalities, educational and cul-

attribute
work en
our know
on produ
the very
online p
In this
in softw
tries to
perceive
those pe
of resear
OSS t
their off
organica
while int
channels
pull-based development model [19] enables anyone to submit

Developers are aware of each other's gender

Which of the following characteristics of your team members are you aware of?

- 74% • Programming skills
- 48% • Gender
- 45% • Real name
- 42% • Social skills
- 40% • Country of residence
- 39% • Personality
- 31% • Reputation as programmer
- 30% • Ethnicity
- 30% • Employment
- 28% • GitHub experience
- 26% • Educational level
- 23% • Age
- 11% • Hobbies
- 4% • Political views

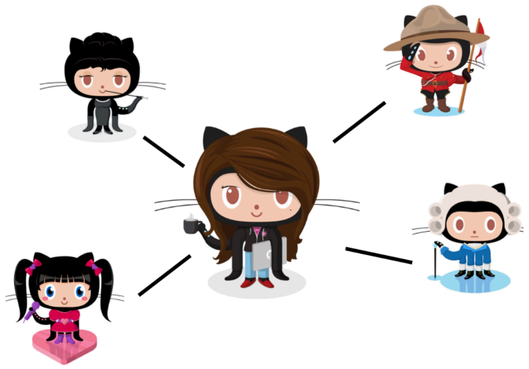
PAPER: The Impact of Social Capital on Sustained Participation in Open Source

How do we stop people from disengaging?

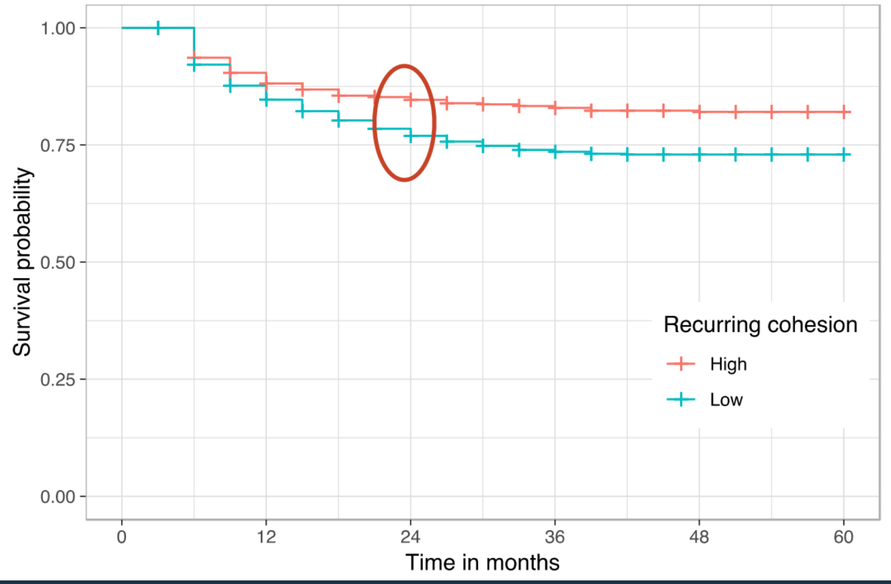
People on informationally diverse teams engage longer:

Being part of teams with more diverse information ~ more prolonged engagement, esp. for women

Information diversity should reduce the risk of demographic-based echo chambers.



More social capital ~ more prolonged engagement



Take away: Invest in building social capital & Foster informationally diverse teams

Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source. *Qiu, H.S., Nolte, A., Brown, A., Serebrenik, A., and Vasilescu, B. ICSE 2019*

Carnegie Mellon University School of Computer Science

STRIDEL

TAKEAWAY:

SCIENCE ACTUALLY SAYS THAT DIVERSITY + PEOPLE MENTORING EACH OTHER MAKES YOU BUILD BETTER SOFTWARE. LIKE REALLY.

NEXT,

A bit about how we build software

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

Black Duck
(now Synopsys)
runs an annual survey asking companies about their open source use.

They survey >1,000 companies about their open source usage.

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

How People Build Software Changed Dramatically Since 2010

2010

39%

of companies said they "ran on open source"

Interviewed 1,240 companies (2015)

Interviewed 1,313 companies (2016)

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

How People Build Software Changed Dramatically Since 2010

2010

39%

of companies said they "ran on open source"

2015

78%

of companies said they "ran on open source"

Interviewed 1,240 companies (2015)

Interviewed 1,313 companies (2016)

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

How People Build Software Changed Dramatically Since 2010

2010

39%

of companies said they "ran on open source"

2015

78%

of companies said they "ran on open source"

2x

This is up 2x over 2010!

Interviewed 1,240 companies (2015)

Interviewed 1,313 companies (2016)

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

How People Build Software Changed Dramatically Since 2010

2010

39%

of companies said they "ran on open source"

2015

78%

of companies said they "ran on open source"

2x

This is up 2x over 2010!

COMPANIES ARE DEPENDING MORE AND MORE ON OSS!

Interviewed 1,240 companies (2015)

Interviewed 1,313 companies (2016)

https://www.blackducksoftware.com/2016-future-of-open-source

Why did companies suddenly decide to shift building atop OSS?

https://www.blackducksoftware.com/2016-future-of-open-source

Why did companies suddenly decide to shift building atop OSS?

2016

Top 3 reasons to use OSS:

- **#1** quality of solutions
- **#2** competitive features & technical capabilities
- **#3** ability to customize & fix

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

Why did companies suddenly decide to shift building atop OSS?

2016

Top 3 reasons to use OSS:

- **#1** quality of solutions
- **#2** competitive features & technical capabilities
- **#3** ability to customize & fix

2015

OSS vs proprietary:

- **66%** of companies consider OSS options before proprietary alternatives

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: 2015 & 2016 Black Duck "Future of Open Source" Survey

Why did companies suddenly decide to shift building atop OSS?

2016

Top 3 reasons to use OSS:

- **#1** quality of solutions
- **#2** competitive features & technical capabilities
- **#3** ability to customize & fix

2015

OSS vs proprietary:

- **66%** of companies consider OSS options before proprietary alternatives

OPEN SOURCE BECAME THE DEFAULT CHOICE

https://www.blackducksoftware.com/2016-future-of-open-source

OPEN SOURCE SURVEYS: Black Duck 2017 Survey

The rapid increase in reliance on open source continues

2017

60%

of companies surveyed increased open source usage.

Main attributed reason:

- low cost with no vendor lock-in

https://www.blackducksoftware.com/open-source-360deg-survey

OPEN SOURCE SURVEYS: Synopsys 2018 Survey (was Black Duck)

Everything is OSS now

Scanned/analyzed (anonymized) data of over **1,100** commercial code bases.

https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2018-ossra.pdf

OPEN SOURCE SURVEYS: Synopsys 2018 Survey (was Black Duck)

Everything is OSS now

Scanned/analyzed (anonymized) data of over **1,100** commercial code bases.

Black Duck audits found open source components in **96%** of the applications scanned, with an average **257** components per application.

- Open source components in 96% of applications scanned!
- Average of **257** open source components per application!

https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2018-ossra.pdf

OPEN SOURCE SURVEYS: Synopsys 2018 Survey (was Black Duck)

Everything is OSS now

Scanned/analyzed (anonymized) data of over **1,100** commercial code bases.

Black Duck audits found open source components in **96%** of the applications scanned, with an average **257** components per application.

- Open source components in 96% of applications scanned!
- Average of **257** open source components per application!

96% USED OSS IN 2018!

https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2018-ossra.pdf

OPEN SOURCE SURVEYS: Synopsys 2018 Survey (was Black Duck)

Everything is OSS now

Scanned/analyzed (anonymized) data of over **1,100** commercial code bases.

Black Duck audits found open source components in **96%** of the applications scanned, with an average **257** components per application.

- Open source components in 96% of applications scanned!
- Average of **257** open source components per application!

96% USED OSS IN 2018!

In 2017, 36% of code base was open source components. In 2018, that number is 57%.

The average percentage of codebase that was open source was **57%** vs. **36%** last year. Many applications now contain more open source than proprietary code.

https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2018-ossra.pdf

OPEN SOURCE SURVEYS: Synopsys 2018 Survey (was Black Duck)

Everything is OSS now

Scanned/analyzed (anonymized) data of over **1,100** commercial code bases.

Black Duck audits found open source components in **96%** of the applications scanned, with an average **257** components per application.

- Open source components in 96% of applications scanned!
- Average of **257** open source components per application!

96% USED OSS IN 2018!

In 2017, 36% of code base was open source components. In 2018, that number is 57%.

The average percentage of codebase that was open source was **57%** vs. **36%** last year. Many applications now contain more open source than proprietary code.

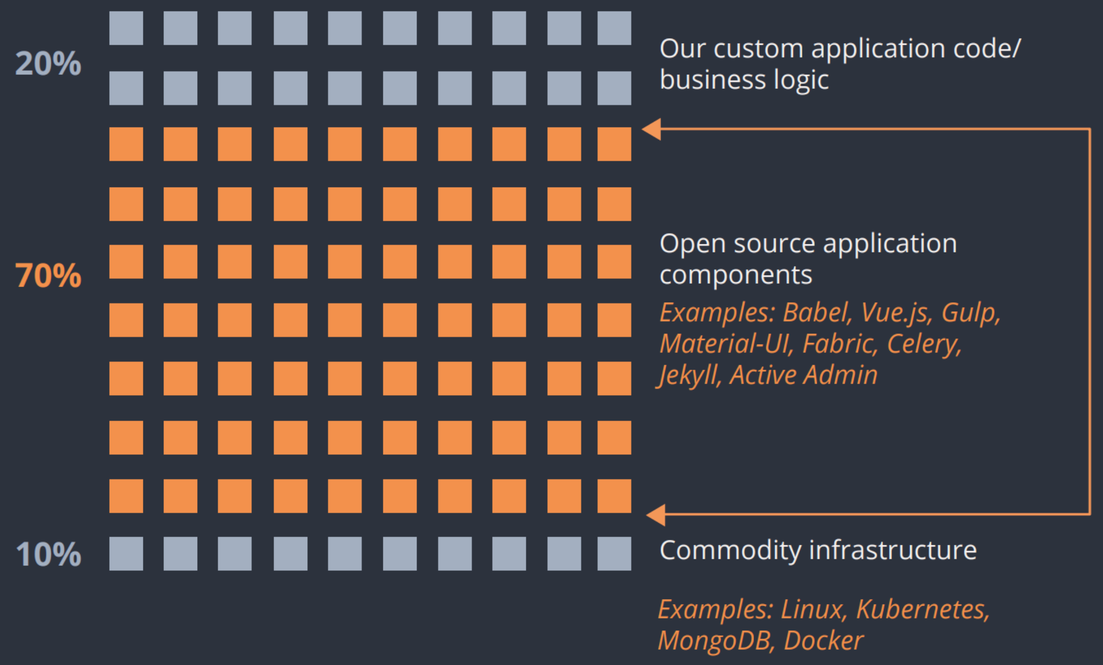
MANY APPS ARE NOW MORE OPEN SOURCE CODE THAN PROPRIETARY

https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2018-ossra.pdf

OPEN SOURCE SURVEYS: 2018 Tidelift Professional Open Source Survey

Software now is mostly made out of OSS components

Most applications are built on top of a foundation of 70% or more open source code



Top layer of our app, our job to support

Most of our app, but no one's job to support

Supported by big vendors or cloud providers

TIDELIFT

https://cdn2.hubspot.net/hubfs/4008838/Introduction_to_Managed_Open_Source.pdf

TAKE A MINUTE TO INTERNALIZE THAT.

TAKE A MINUTE TO INTERNALIZE THAT.

Synopsys:

- **in 2017:** 36% of code bases are open source components.
- **in 2018:** 57% of code bases are open source components.

TAKE A MINUTE TO INTERNALIZE THAT.

Synopsys:

in 2017: 36% of code bases are open source components.
in 2018: 57% of code bases are open source components.

Tidelift:

in 2018: 70% of code bases are open source components,
only 20% is custom application/business logic.

TAKE A MINUTE TO INTERNALIZE THAT.

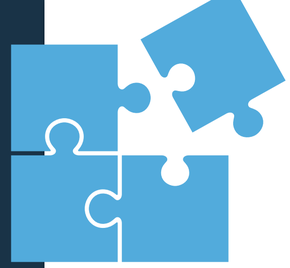
Synopsys:

in 2017: 36% of code bases are open source components.
in 2018: 57% of code bases are open source components.

Tidelift:

in 2018: 70% of code bases are open source components,
only 20% is custom application/business logic.

Software now constructed from OSS puzzle pieces



Instagram

Mike Krieger
Instagram co-founder

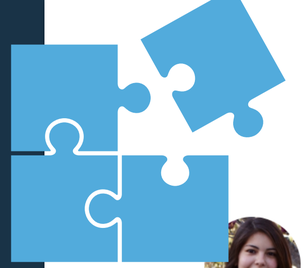
Blog article:
Advice on picking tech for your startup

> Borrow instead of building whenever possible

> There are hundreds of fantastic open-source projects that have been built through the hard experience of creating and scaling companies; especially around infrastructure and monitoring...that can save you time and let you focus on actually building out your product.

https://opbeat.com/blog/posts/picking-tech-for-your-startup/

Software now constructed from OSS puzzle pieces



Instagram

Mike Krieger
Instagram co-founder

Blog article:
Advice on picking tech for your startup

> Borrow instead of building whenever possible

> There are hundreds of fantastic open-source projects that have been built through the hard experience of creating and scaling companies; especially around infrastructure and monitoring...that can save you time and let you focus on actually building out your product.

https://opbeat.com/blog/posts/picking-tech-for-your-startup/

Nadia Eghbal Follow
subtle + overt = subvert
Jan 25 · 4 min read

Open source was worth at least $143M of Instagram's $1B acquisition

Every tech company built after 2000 has benefitted from open source infrastructure—that is, free, public code that anybody can use to build software.

It's saved companies countless dollars, developer hours, and headaches to be able to use someone else's code to get up and running instead of having to build everything from scratch.

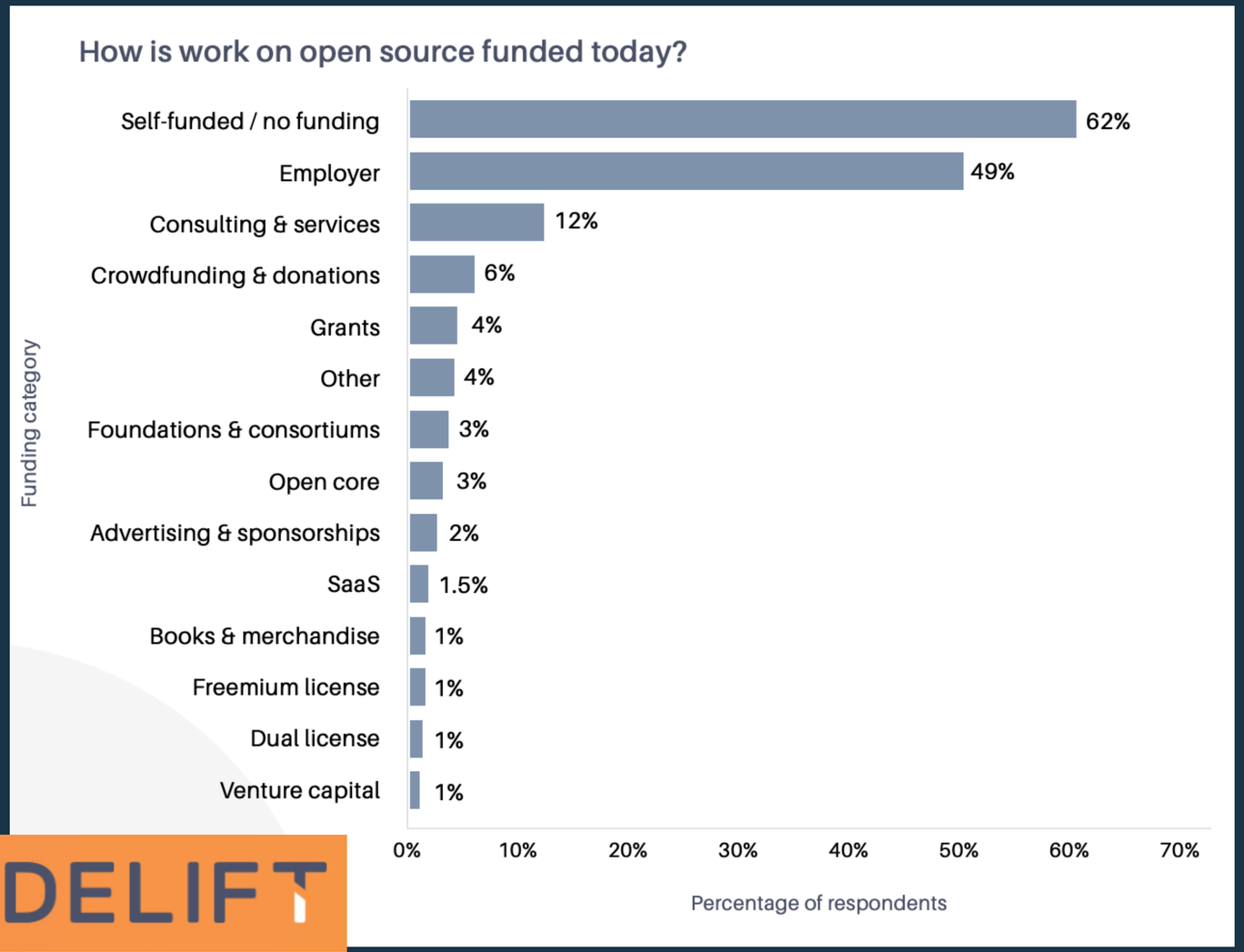
I decided to take a stab at calculating how much that infrastructure is actually worth to a company.

Instagram is a great example to look at, not just because of its acquisition price, but how quickly it was able to scale and exit.

https://medium.com/@nayafia/open-source-was-worth-at-least-143m-of-instagram-s-1b-acquisition-808bb85e4681#.d6gzzr9nk

OPEN SOURCE SURVEYS: 2018 Tidelift Professional Open Source Survey

Yet, most have to self-support their OSS work



62%

of respondents said that they are required to financially support their open source work with their own funds, or that they receive no external funding at all.

Over 1,200 respondents

https://tidelift.com/about/2018-tidelift-professional-open-source-survey-results

NOW,

A bit about open source

Of course, things could get terrifying from here

HYPOTHETICAL WORST CASE

OpenSSL
Cryptography and SSL/TLS Toolkit

In 2014,

66% of all web servers were using OpenSSL

Meanwhile, OpenSSL was maintained by only a few volunteers

https://fordfoundcontent.blob.core.windows.net/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

HYPOTHETICAL WORST CASE

OpenSSL
Cryptography and SSL/TLS Toolkit

In 2014,

66% of all web servers were using OpenSSL

Meanwhile, OpenSSL was maintained by only a few volunteers

> Steve Marquess, noticed that one contributor, Stephen Henson, was working full time on OpenSSL. Curious, Marquess asked him what he did for income, and was shocked to learn that Henson made one-fifth of Marquess's salary.

https://fordfoundcontent.blob.core.windows.net/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

HYPOTHETICAL WORST CASE

OpenSSL
Cryptography and SSL/TLS Toolkit

In 2014,

66% of all web servers were using OpenSSL

Meanwhile, OpenSSL was maintained by only a few volunteers

> Steve Marquess, noticed that one contributor, Stephen Henson, was working full time on OpenSSL. Curious, Marquess asked him what he did for income, and was shocked to learn that Henson made one-fifth of Marquess's salary.

> Marquess had always considered himself to be a strong programmer, but his skills paled in comparison to Henson's. ... Henson had been working on OpenSSL since 1998.

https://fordfoundcontent.blob.core.windows.net/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

HYPOTHETICAL WORST CASE

OpenSSL
Cryptography and SSL/TLS Toolkit

In 2014,

66% of all web servers were using OpenSSL

Meanwhile, OpenSSL was maintained by only a few volunteers

> Steve Marquess, noticed that one contributor, Stephen Henson, was working full time on OpenSSL. Curious, Marquess asked him what he did for income, and was shocked to learn that Henson made one-fifth of Marquess's salary.

> Marquess had always considered himself to be a strong programmer, but his skills paled in comparison to Henson's. ... Henson had been working on OpenSSL since 1998.

"I had always assumed, (as had the rest of the world) that the OpenSSL team was large, active, and well resourced."

– Steve Marquess

https://fordfoundcontent.blob.core.windows.net/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

HYPOTHETICAL WORST CASE

OpenSSL
Cryptography and SSL/TLS Toolkit

In 2014,

66% of all web servers were using OpenSSL

Meanwhile, OpenSSL was maintained by only a few volunteers

> Steve Marquess, noticed that one contributor, Stephen Henson, was working full time on OpenSSL. Curious, Marquess asked him what he did for income, and was shocked to learn that Henson made one-fifth of Marquess's salary.

> Marquess had always considered himself to be a strong programmer, but his skills paled in comparison to Henson's. ... Henson had been working on OpenSSL since 1998.

"I had always assumed, (as had the rest of the world) that the OpenSSL team was large, active, and well resourced."

– Steve Marquess

In reality, OpenSSL wasn't even able to support one person's work.

https://fordfoundcontent.blob.core.windows.net/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

HYPOTHETICAL WORST CASE

OpenSSL
Cryptography and SSL/TLS Toolkit

In 2014,

66% of all web servers were using OpenSSL

Meanwhile, OpenSSL was maintained by only a few volunteers

> Steve Marquess, noticed that one contributor, Stephen Henson, was working full time on OpenSSL. Curious, Marquess asked him what he did for income, and was shocked to learn that Henson made one-fifth of Marquess's salary.

> Marquess had always considered himself to be a strong programmer, but his skills paled in comparison to Henson's. ... Henson had been working on OpenSSL since 1998.

"I had always assumed, (as had the rest of the world) that the OpenSSL team was large, active, and well resourced."

– Steve Marquess

In reality, OpenSSL wasn't even able to support one person's work.

INDUSTRY, GOVERNMENT, ETC ARE OFTEN UNAWARE OF INFRASTRUCTURE'S FUNDING ISSUES

https://fordfoundcontent.blob.core.windows.net/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

ANOTHER WORST CASE

Ever heard of the truck factor?

Truck Factor:

the minimal # of developers that have to be hit by a truck (or quit) before a project is incapacitated

- Look at the 133 most active projects on GitHub
- Determine the amount of information concentrated in individual team members from commits.

https://peerj.com/preprints/1233.pdf

ANOTHER WORST CASE

Ever heard of the truck factor?

Truck Factor:

the minimal # of developers that have to be hit by a truck (or quit) before a project is incapacitated

- Look at the 133 most active projects on GitHub
- Determine the amount of information concentrated in individual team members from commits.

64% OF PROJECTS RELIED ON 1-2 DEVS TO SURVIVE

https://peerj.com/preprints/1233.pdf

ANOTHER WORST CASE

Ever heard of the truck factor?

Truck Factor:

the minimal # of developers that have to be hit by a truck (or quit) before a project is incapacitated

- Look at the 133 most active projects on GitHub
- Determine the amount of information concentrated in individual team members from commits.

64% OF PROJECTS RELIED ON 1-2 DEVS TO SURVIVE

https://peerj.com/preprints/1233.pdf

Table 2: Truck Factor results

TF	Repositories
1	alexreisner/geocoder, atom/atom-shell, bjorn/tiled, bumptech/glide, celery/celery, celluloid/celluloid, dropwizard/dropwizard, dropwizard/metrics, erikhuda/thor, Eugeny/ajenti, getsentry/sentry, github/android, gruntjs/grunt, janl/mustache.js, jrburke/requirejs, justinfrench/formtastic, kivy/kivy, koush/ion, kriswallsmith/assetic, Leaflet/Leaflet, less/less.js, mailpile/Mailpile, mbostock/d3, mitchellh/vagrant, mitsuhiko/flask, mongoid/mongoid, nate-parrott/Flashlight, nicolasgramlich/AndEngine, paulasmuth/fnordmetric, phacility/phabricator, powerline/powerline, puphpet/puphpet, ratchetphp/Ratchet, ReactiveX/RxJava, sandstorm-io/capnproto, sass/sass, sebastianbergmann/phpunit, sferik/twitter, silexphp/Silex, sstephenson/sprockets, substack/node-browserify, thoughtbot/factory_girl, thoughtbot/paperclip, wp-cli/wp-cli
2	**activeadmin/activeadmin**, ajaxorg/ace, ansible/ansible, apache/cassandra, bup/bup, **clojure/clojure**, composer/composer, cucumber/cucumber, driftyco/ionic, drupal/drupal, elasticsearch/elasticsearch, **elasticsearch/logstash**, excilys/androidannotations, facebook/osquery, facebook/presto, FriendsOfPHP/PHP-CS-Fixer, github/linguist, Itseez/opencv, **jadejs/jade**, jashkenas/backbone, JohnLangford/vowpal_wabbit, jquery/jquery-ui, libgdx/libgdx, **meskyanichi/backup**, **netty/netty**, **omab/django-social-auth**, **openframeworks/openFrameworks**, **plataformatec/devise**, prawnpdf/prawn, **pydata/pandas**, Respect/Validation, **sampsyo/beets**, SFTtech/openage, **sparklemotion/nokogiri**, **strongloop/express**, thinkaurelius/titan, **ThinkUpLLC/ThinkUp**, thumbor/thumbor, xetorthio/jedis
3	**bbatsov/rubocop**, bitcoin/bitcoin, **bundler/bundler**, **divio/django-cms**, **haml/haml**, **jnicklas/capybara**, **mozilla/pdf.js**, **rg3/youtube-dl**, **mrdoob/three.js**, **spring-projects/spring-framework**, **yiisoft/yii2**
4	**boto/boto**, **BVLC/caffe**, **codemirror/CodeMirror**, **gradle/gradle**, **ipython/ipython**, **jekyll/jekyll**, **jquery/jquery**
5	**iojs/io.js**, **meteor/meteor**, **ruby/ruby**, **WordPress/WordPress**
6	**chef/chef**, **cocos2d/cocos2d-x**, **diaspora/diaspora**, **emberjs/ember.js**, **resque/resque**, **Shopify/active_merchant**, **spotify/luigi**, **TryGhost/Ghost**
7	**django/django**, **joomla/joomla-cms**, **scikit-learn/scikit-learn**
9	**JetBrains/intellij-community**, **puppetlabs/puppet**, **rails/rails**
11	**saltstack/salt**, **Seldaek/monolog**, **v8/v8**
12	**git/git**, **webscalesql/webscalesql-5.6**
13	**fog/fog**
14	**odoo/odoo**
18	**php/php-src**
19	**android/platform_frameworks_base**, **moment/moment**
23	**fzaninotto/Faker**
56	**caskroom/homebrew-cask**
130	**torvalds/linux**
250	**Homebrew/homebrew**

ANOTHER WORST CASE

Ever heard of the truck factor?

Truck Factor:

the minimal # of developers that have to be hit by a truck (or quit) before a project is incapacitated

- Look at the 133 most active projects on GitHub
- Determine the amount of information concentrated in individual team members from commits.

64% OF PROJECTS RELIED ON 1-2 DEVS TO SURVIVE

https://peerj.com/preprints/1233.pdf

TRUCK FACTOR RESULTS:

The higher the TF the better!

Only a handful of projects with a high TF...

Table 2: Truck Factor results

TF	Repositories
1	alexreisner/geocoder, atom/atom-shell, bjorn/tiled, bumptech/glide, celery/celery, celluloid/celluloid, dropwizard/dropwizard, dropwizard/metrics, erikhuda/thor, Eugeny/ajenti, getsentry/sentry, github/android, gruntjs/grunt, janl/mustache.js, jrburke/requirejs, justinfrench/formtastic, kivy/kivy, koush/ion, kriswallsmith/assetic, Leaflet/Leaflet, less/less.js, mailpile/Mailpile, mbostock/d3, mitchellh/vagrant, mitsuhiko/flask, mongoid/mongoid, nate-parrott/Flashlight, nicolasgramlich/AndEngine, paulasmuth/fnordmetric, phacility/phabricator, powerline/powerline, puphpet/puphpet, ratchetphp/Ratchet, ReactiveX/RxJava, sandstorm-o/capnproto, sass/sass, sebastianbergmann/phpunit, sferik/twitter, silexphp/Silex, sstephenson/sprockets, substack/node-browserify, thoughtbot/factory_girl, thoughtbot/paperclip, wp-cli/wp-cli
2	activeadmin/activeadmin, ajaxorg/ace, ansible/ansible, apache/cassandra, bup/bup, clojure/clojure, composer/composer, cucumber/cucumber, driftyco/ionic, drupal/drupal, elasticsearch/elasticsearch, elasticsearch/logstash, excilys/androidannotations, facebook/osquery, facebook/presto, FriendsOfPHP/PHP-CS-Fixer, github/linguist, Itseez/opencv, jadejs/jade, jashkenas/backbone, JohnLangford/vowpal_wabbit, jquery/jquery-ui, libgdx/libgdx, meskyanichi/backup, netty/netty, omab/django-social-auth, openframeworks/openFrameworks, plataformatec/devise, prawnpdf/prawn, pydata/pandas, Respect/Validation, sampsyo/beets, SFTtech/openage, sparklemotion/nokogiri, strongloop/express, thinkaurelius/titan, ThinkUpLLC/ThinkUp, thumbor/thumbor, xetorthio/jedis
3	bbatsov/rubocop, bitcoin/bitcoin, bundler/bundler, divio/django-cms, haml/haml, jnicklas/capybara, mozilla/pdf.js, rg3/youtube-dl, mrdoob/three.js, spring-projects/spring-framework, yiisoft/yii2
4	boto/boto, BVLC/caffe, codemirror/CodeMirror, gradle/gradle, ipython/ipython, jekyll/jekyll, jquery/jquery
5	iojs/io.js, meteor/meteor, ruby/ruby, WordPress/WordPress
6	chef/chef, cocos2d/cocos2d-x, diaspora/diaspora, emberjs/ember.js, resque/resque, Shopify/active_merchant, spotify/luigi, TryGhost/Ghost
7	django/django, joomla/joomla-cms, scikit-learn/scikit-learn
9	JetBrains/intellij-community, puppetlabs/puppet, rails/rails
11	saltstack/salt, Seldaek/monolog, v8/v8
12	git/git, webscalesql/webscalesql-5.6
13	fog/fog
14	odoo/odoo
18	php/php-src
19	android/platform_frameworks_base, moment/moment
23	fzaninotto/Faker
56	caskroom/homebrew-cask
130	torvalds/linux
250	Homebrew/homebrew

TRUCK FACTOR RESULTS

A sampling of some low truck factors...

Truck Factor 1:

gruntjs/grunt

wp-cli/wp-cli

sass/sass

mbostock/d3

ReactiveX/RxJava

Truck Factor 2:

apache/cassandra

clojure/clojure

pydata/pandas

netty/netty

drupal/drupal

LESSONS FROM Scala

Ecosystem & Community is Everything

Empirical Analysis of Programming Language Adoption

Looked at lots of data.

- 10 years of repository meta data, tracking up to 590,000 open source projects
- Survey data of developers over multiple surveys, ranging from 1,000 to 13,000 respondents

OOPSLA'13

Empirical Analysis of Programming Language Adoption

Leo A. Meyerovich
UC Berkeley *
lmeyerov@eecs.berkeley.edu

Ariel Rabkin
Princeton University
asrabkin@cs.princeton.edu

Abstract

Some programming languages become widely popular while others fail to grow beyond their niche or disappear altogether. This paper uses survey methodology to identify the factors that lead to language adoption. We analyze large datasets, including over 200,000 SourceForge projects, 590,000 projects tracked by Ohloh, and multiple surveys of 1,000-13,000 programmers.

We report several prominent findings. First, language adoption follows a power law; a small number of languages account for most language use, but the programming market supports many languages with niche user bases. Second, intrinsic features have only secondary importance in adoption. Open source libraries, existing code, and experience strongly influence developers when selecting a language for a project. Language features such as performance, reliability, and simple semantics do not. Third, developers will steadily learn and forget languages, and the overall number of languages developers are familiar with is independent of age. Developers select more varied languages if their education exposed them to different language families. Finally, when considering intrinsic aspects of languages, developers prioritize expressivity over correctness. They perceive static types as more valuable for properties such as the former rather than for correctness checking.

1. Introduction

Some programming languages succeed and others fail. Understanding this process is a foundational step towards enabling advocates to influence its out-

aid developers in determining when and whether to bet on a new, experimental language. To date, the language adoption process has not been quantitatively studied in a large scale. This paper addresses that gap. We use a combination of survey research and software repository mining to investigate the factors that influence developer language choice.

Since little is quantified about the programming language adoption process, we focus on broad research questions:

What statistical properties describe language popularity? We begin (Section 3) with an empirical analysis of language use across many open source projects. Such a macro-scale analysis reveals what trajectories languages tend to follow. Our analysis includes the overall distribution of language use, and how it varies based on the kind of project and developer experience.

We found that popularity follows a power law, which means that most usage is concentrated in a small number of languages, but many unpopular languages will still find a user base. The popular languages are used across a variety of application domains while less popular ones tend to be used for niche domains. Even in niche domains, popular languages are still more typically used.

Which factors most influence developer decision-making for language selection? Section 4 examines the subjective motivations of developers when picking languages for specific projects. Knowing what matters to developers helps language designers and advocates address their perceived needs.

Through multiple surveys, we saw that developers value open source libraries as the dominant factor in choosing programming languages. Social factors not tied to intrinsic lan-

such as existing personal or team experience,

Empirical Analysis of Programming Language Adoption

Looked at lots of data.

- 10 years of repository meta data, tracking up to 590,000 open source projects
- Survey data of developers over multiple surveys, ranging from 1,000 to 13,000 respondents

BIG QUESTION:

Which factors most influence developer decision-making for language selection?

OOPSLA'13

Empirical Analysis of Programming Language Adoption

Leo A. Meyerovich
UC Berkeley *
lmeyerov@eecs.berkeley.edu

Ariel Rabkin
Princeton University
asrabkin@cs.princeton.edu

Abstract

Some programming languages become widely popular while others fail to grow beyond their niche or disappear altogether. This paper uses survey methodology to identify the factors that lead to language adoption. We analyze large datasets, including over 200,000 SourceForge projects, 590,000 projects tracked by Ohloh, and multiple surveys of 1,000-13,000 programmers.

We report several prominent findings. First, language adoption follows a power law; a small number of languages account for most language use, but the programming market supports many languages with niche user bases. Second, intrinsic features have only secondary importance in adoption. Open source libraries, existing code, and experience strongly influence developers when selecting a language for a project. Language features such as performance, reliability, and simple semantics do not. Third, developers will steadily learn and forget languages, and the overall number of languages developers are familiar with is independent of age. Developers select more varied languages if their education exposed them to different language families. Finally, when considering intrinsic aspects of languages, developers prioritize expressivity over correctness. They perceive static types as more valuable for properties such as the former rather than for correctness checking.

1. Introduction

Some programming languages succeed and others fail. Understanding this process is a foundational step towards enabling advocates to influence its out-

aid developers in determining when and whether to bet on a new, experimental language. To date, the language adoption process has not been quantitatively studied in a large scale. This paper addresses that gap. We use a combination of survey research and software repository mining to investigate the factors that influence developer language choice.

Since little is quantified about the programming language adoption process, we focus on broad research questions:

What statistical properties describe language popularity? We begin (Section 3) with an empirical analysis of language use across many open source projects. Such a macro-scale analysis reveals what trajectories languages tend to follow. Our analysis includes the overall distribution of language use, and how it varies based on the kind of project and developer experience.

We found that popularity follows a power law, which means that most usage is concentrated in a small number of languages, but many unpopular languages will still find a user base. The popular languages are used across a variety of application domains while less popular ones tend to be used for niche domains. Even in niche domains, popular languages are still more typically used.

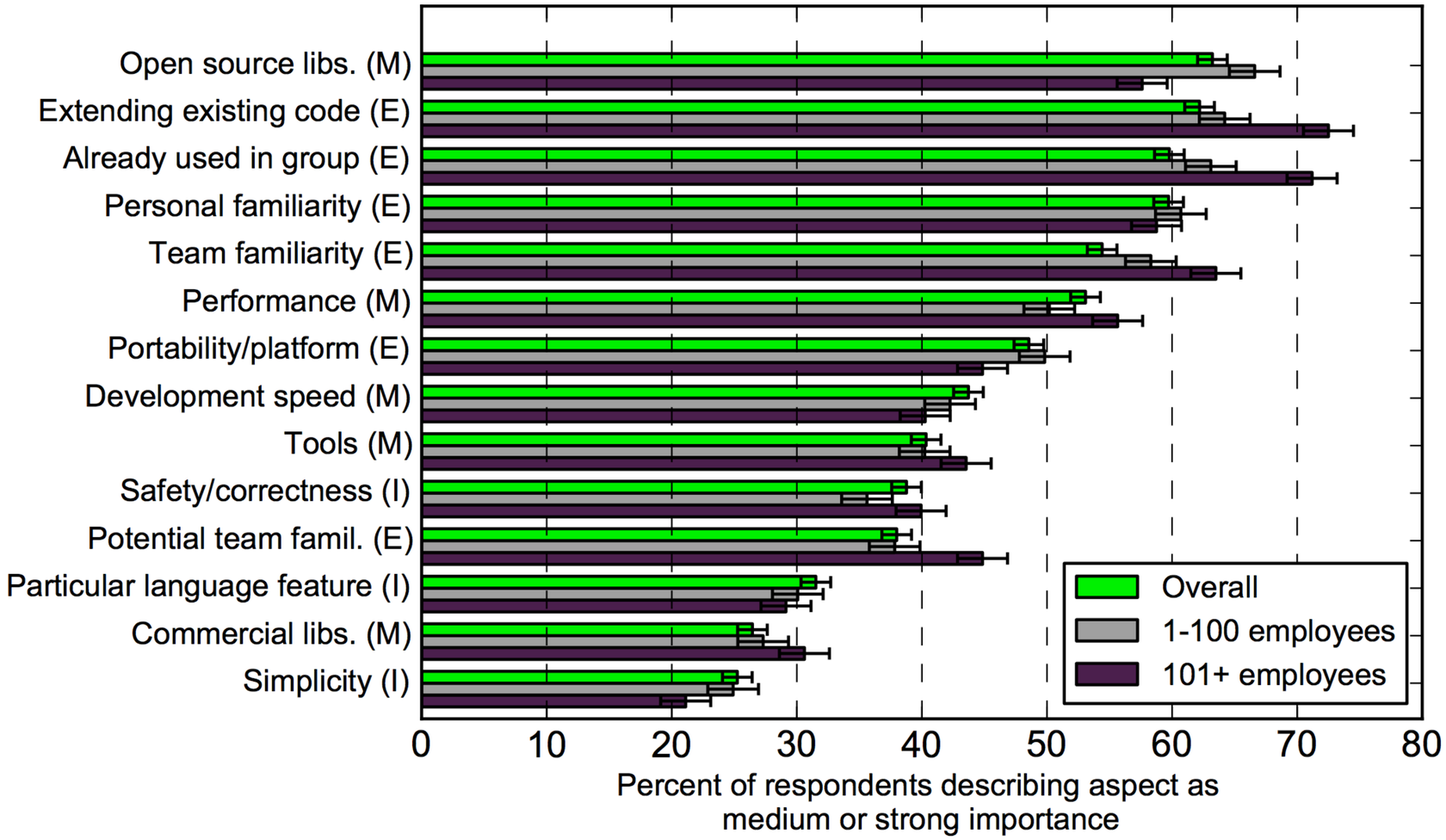
Which factors most influence developer decision-making for language selection? Section 4 examines the subjective motivations of developers when picking languages for specific projects. Knowing what matters to developers helps language designers and advocates address their perceived needs.

Through multiple surveys, we saw that developers value open source libraries as the dominant factor in choosing programming languages. Social factors not tied to intrinsic lan-

such as existing personal or team experience,

EMPIRICAL ANALYSIS OF PROGRAMMING LANGUAGE ADOPTION

Importance of different factors when picking a language:



Empirical Analysis of Programming Language Adoption

OOPSLA'13

BIG QUESTION:

Which factors most influence developer decision-making for language selection?

> Through multiple surveys, we saw that developers value open source libraries as the dominant factor in choosing programming languages. Social factors not tied to intrinsic language features, such as existing personal or team experience, also rate highly.

Empirical Analysis of Programming Language Adoption

Leo A. Meyerovich
UC Berkeley *
lmeyerov@eecs.berkeley.edu

Ariel Rabkin
Princeton University
asrabkin@cs.princeton.edu

Abstract

Some programming languages become widely popular while others fail to grow beyond their niche or disappear altogether. This paper uses survey methodology to identify the factors that lead to language adoption. We analyze large datasets, including over 200,000 SourceForge projects, 590,000 projects tracked by Ohloh, and multiple surveys of 1,000-13,000 programmers.

We report several prominent findings. First, language adoption follows a power law; a small number of languages account for most language use, but the programming market supports many languages with niche user bases. Second, intrinsic features have only secondary importance in adoption. Open source libraries, existing code, and experience strongly influence developers when selecting a language for a project. Language features such as performance, reliability, and simple semantics do not. Third, developers will steadily learn and forget languages, and the overall number of languages developers are familiar with is independent of age. Developers select more varied languages if their education exposed them to different language families. Finally, when considering intrinsic aspects of languages, developers prioritize expressivity over correctness. They perceive static types as more valuable for properties such as the former rather than for correctness checking.

1. Introduction

Some programming languages succeed and others fail. Understanding this process is a foundational step towards enabling advocates to influence its out-

aid developers in determining when and whether to bet on a new, experimental language. To date, the language adoption process has not been quantitatively studied in a large scale. This paper addresses that gap. We use a combination of survey research and software repository mining to investigate the factors that influence developer language choice.

Since little is quantified about the programming language adoption process, we focus on broad research questions:

What statistical properties describe language popularity? We begin (Section 3) with an empirical analysis of language use across many open source projects. Such a macro-scale analysis reveals what trajectories languages tend to follow. Our analysis includes the overall distribution of language use, and how it varies based on the kind of project and developer experience.

We found that popularity follows a power law, which means that most usage is concentrated in a small number of languages, but many unpopular languages will still find a user base. The popular languages are used across a variety of application domains while less popular ones tend to be used for niche domains. Even in niche domains, popular languages are still more typically used.

Which factors most influence developer decision-making for language selection? Section 4 examines the subjective motivations of developers when picking languages for specific projects. Knowing what matters to developers helps language designers and advocates address their perceived needs.

Through multiple surveys, we saw that developers value open source libraries as the dominant factor in choosing programming languages. Social factors not tied to intrinsic language features, such as existing personal or team experience,

Empirical Analysis of Programming Language Adoption

BIG QUESTION:

Which factors most influence developer decision-making for language selection?

> Through multiple surveys, we saw that developers value open source libraries as the dominant factor in choosing programming languages. Social factors not tied to intrinsic language features, such as existing personal or team experience, also rate highly.

We emphasize four results from the data:

1. **Open source libraries.** Open source libraries are the most influential factor for language choice overall and the most influential factor for commercial projects at small companies. They are an important factor, but not the most important factor, at large companies.
2. **Social Factors outweigh Intrinsics.** Existing code or expertise with the language are four of the top five factors for adoption. In contrast, intrinsic factors, such as a language's simplicity or safety, rank low. Implementation attributes, like performance and tool quality, have both intrinsic and extrinsic components. (Some languages lend themselves more easily than others to a high-performance implementation.) These mixed attributes vary in importance.
3. **Domain specialization.** Libraries, developer experience, and legacy code are all important in language selection. These factors are often associated with particular application domains. Thus, the developer emphasis on these attributes helps explain the result in Section 3.2 that less-popular languages are more niche-specific.
4. **Company size matters.** Employees at larger companies place significantly more value on legacy code and knowledge than do employees at small companies.

(From the paper)

So you might say...

A open source project

✨is✨

its Community.

So you might say...

A open source project 's success

✨is✨ attributed to

its Community.

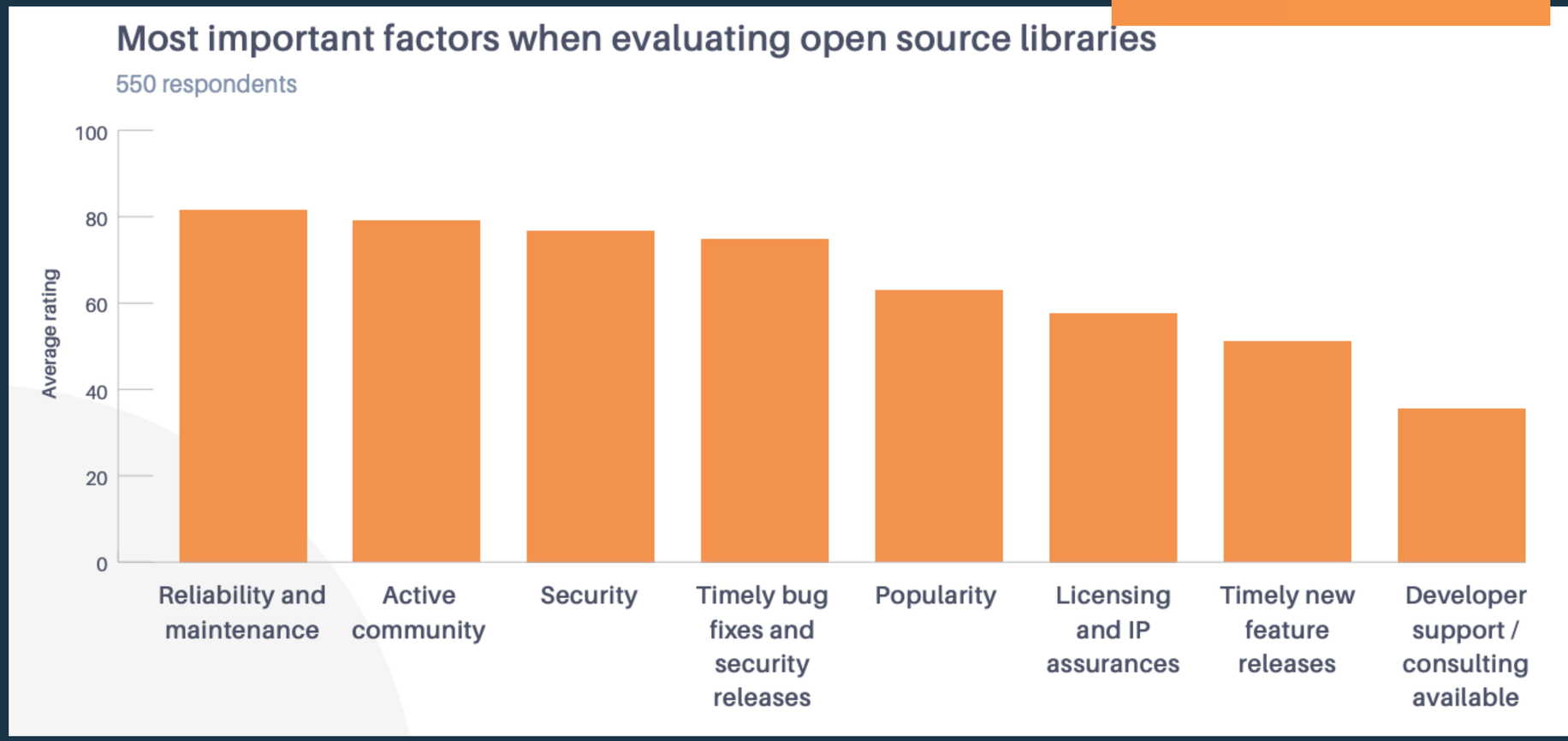
OPEN SOURCE SURVEYS: 2018 Tidelift Professional Open Source Survey

Professional users want an active community!

TIDELIFT

What do professional users care about most?

- Software that is reliable and well maintained
- Software that has an active community using and supporting it
- Software that is secure
- Software with maintainers who provide timely bug fixes and security releases



https://tidelift.com/about/2018-tidelift-professional-open-source-survey-results

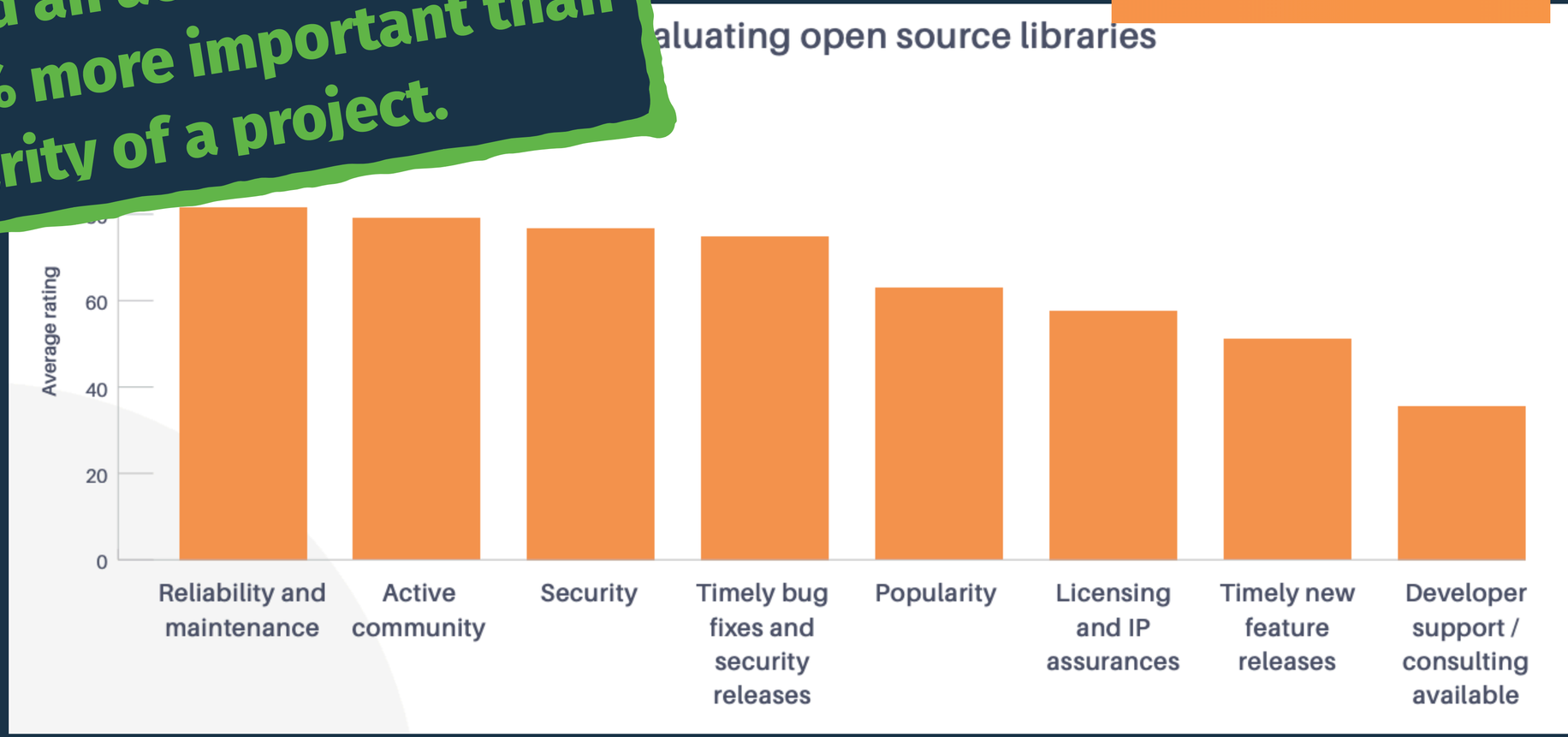
OPEN SOURCE SURVEYS: 2018 Tidelift Professional Open Source Survey

Professional users want an active community!

Respondents rated an active community as being over 20% more important than the popularity of a project.

What do

- Software that has an active community using and supporting it
- Software that is secure
- Software with maintainers who provide timely bug fixes and security releases



https://tidelift.com/about/2018-tidelift-professional-open-source-survey-results

What does this all mean?

TAKEAWAY:

Community & Ecosystem are among the most important factors to an open source project's success.

To stay alive, there should be a relentless focus on growing the community and ecosystem! And that is hard work.

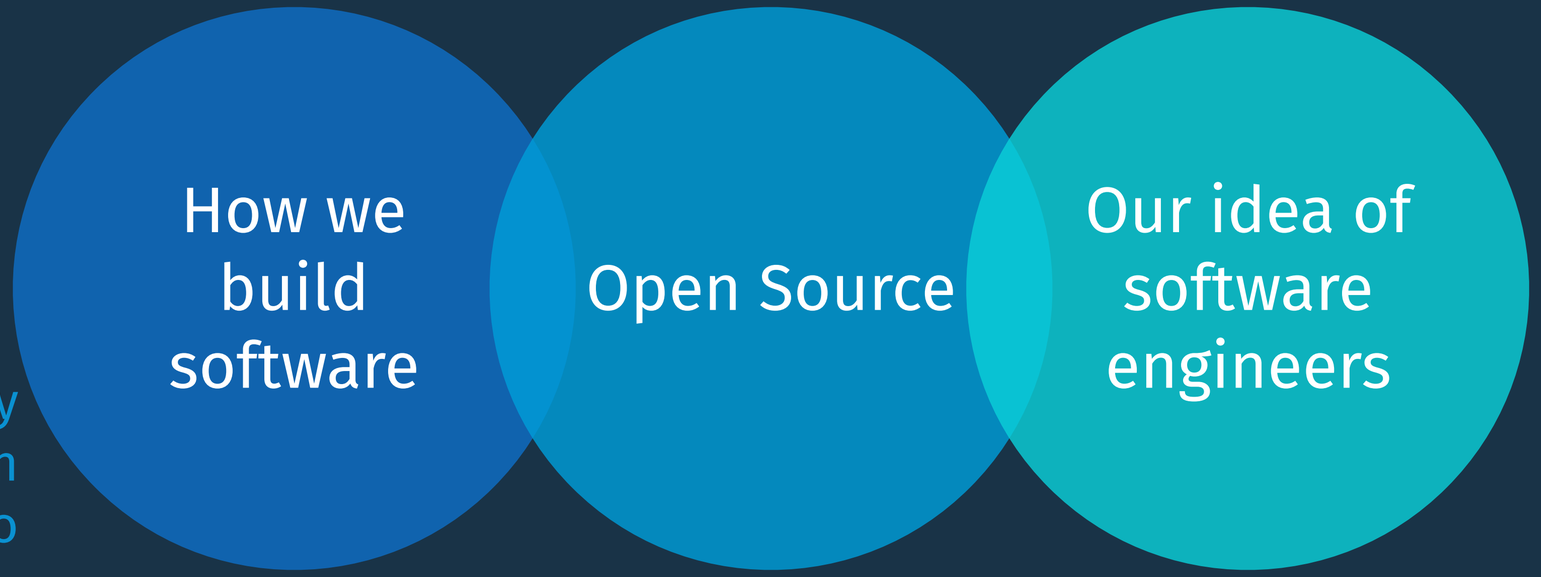
PHEW.

Let’s wrap it up

Things that are changing *fast:*

& that more people should be aware of

We saw 3 areas...



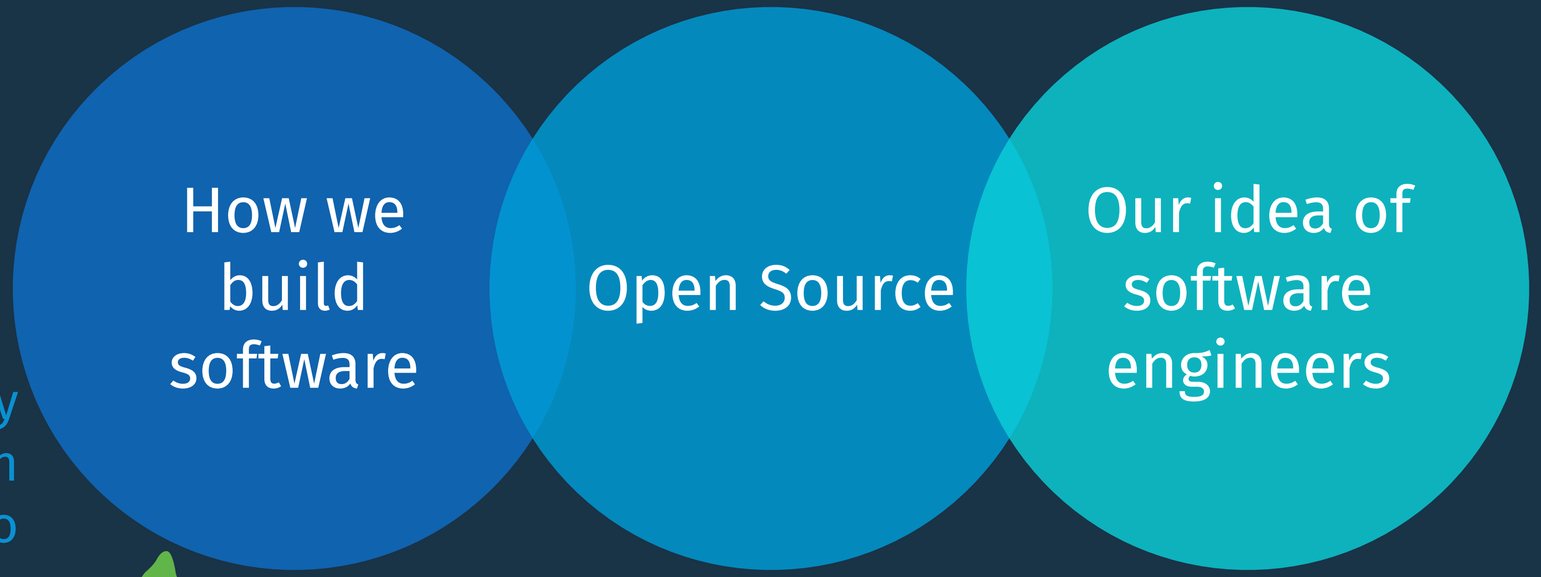
What we actually do nowadays when we sit down to build an app.

What SWEs should know, how much experience they have, and who they are.

Things that are changing *fast:*

& that more people should be aware of

We saw 3 areas...



What we actually do nowadays when we sit down to build an app.

What SWEs should know, how much experience they have, and who they are.

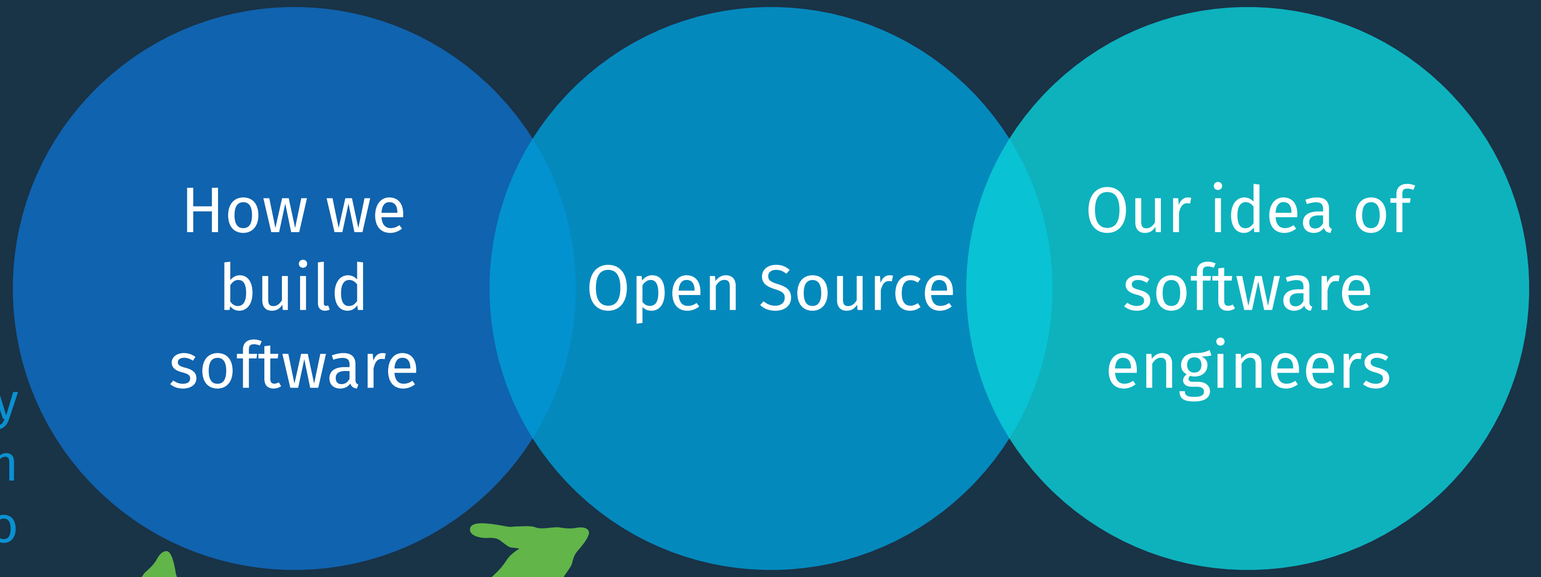
TAKEAWAY:

We largely glue together open source components, now (We didn't do this as much, not even like 3 years ago.)

Things that are changing *fast:*

& that more people should be aware of

We saw 3 areas...



What we actually do nowadays when we sit down to build an app.

What SWEs should know, how much experience they have, and who they are.

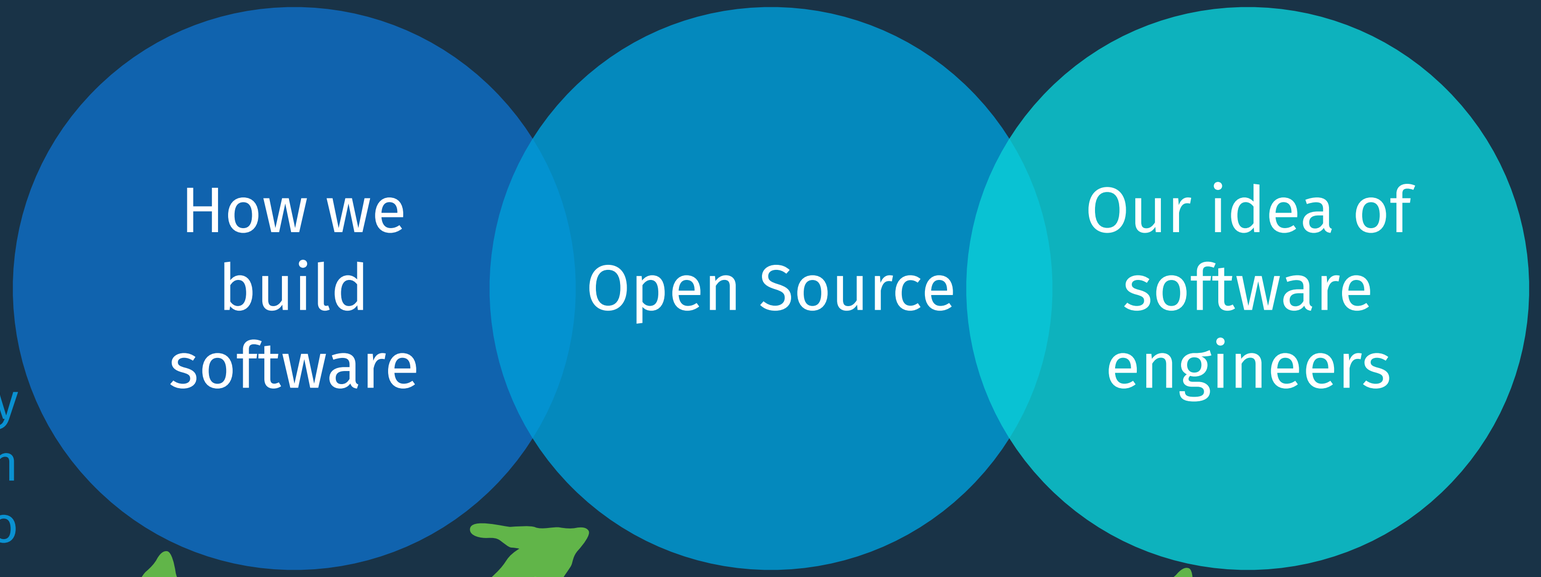
TAKEAWAY:
We largely glue together open source components, now (We didn't do this as much, not even like 3 years ago.)

TAKEAWAY:
There remain sustainability issues in OSS that we should be more cognizant of.

Things that are changing *fast:*

& that more people should be aware of

We saw 3 areas...



What we actually do nowadays when we sit down to build an app.

TAKEAWAY:
We largely glue together open source components, now (We didn't do this as much, not even like 3 years ago.)

TAKEAWAY:
There remain sustainability issues in OSS that we should be more cognizant of.

What SWEs should know, how much experience they have, and who they are.

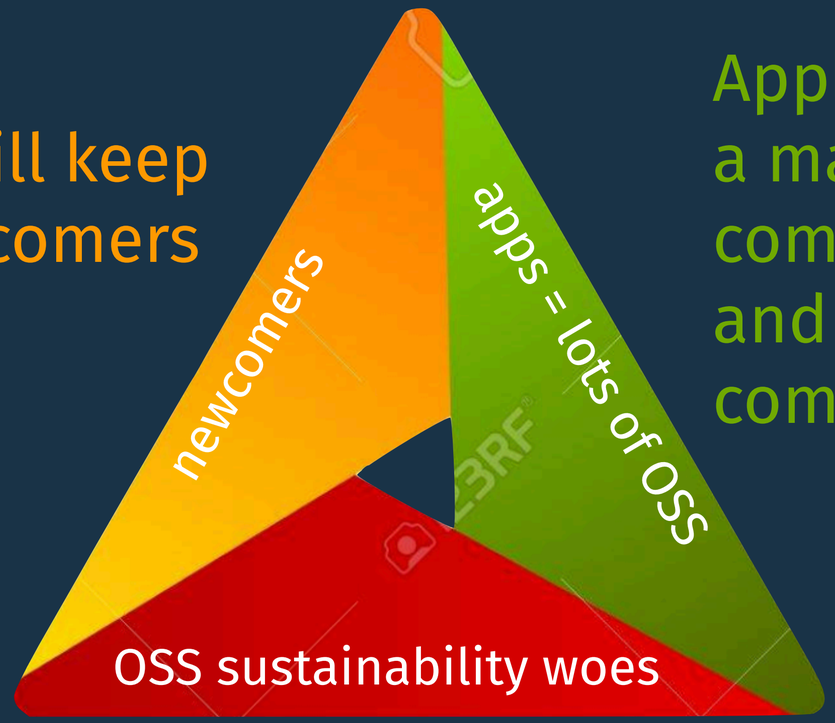
TAKEAWAY:
Most developers are extreme newcomers. We need to adapt to this.

What does this all mean?

- **We're getting a lot of newcomers.**
- **Those newcomers don't all have fancy CS degrees.**
- **But maybe they don't need one?**
  - Frameworks are king!
  - Applications now are only 20% business logic (in one estimate)
- **We're getting a lot of newcomers.**
- **We're actually getting good at reuse nowadays.**
- **The pieces that we (&newcomers) are reusing are largely open source**
- **Corporate subsidy is helping open source *a lot* (49%) but 62% is self-funded/not funded**
- **We're getting a lot of newcomers.**

It’s all related

We’re getting and will keep getting a lot of newcomers across our industry

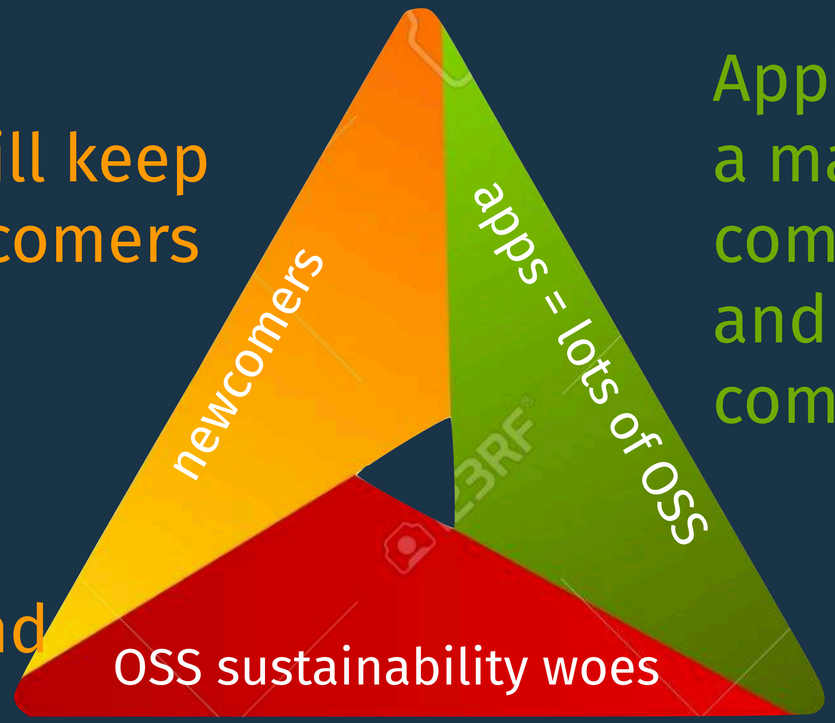


Applications are becoming a majority open source components. Frameworks, and gluing together OS components.

Open source software continues to struggle with sustainability.

It’s all related

We’re getting and will keep getting a lot of newcomers across our industry



Applications are becoming a majority open source components. Frameworks, and gluing together OS components.

This is making it easier for anyone to build applications!

Newcomers depend on OSS, and on it being easy to use

Open source software continues to struggle with sustainability.

What does this all mean?

As managers...

Your engineers are going to need to be good mentors. Put a greater emphasis on mentorship sooner than later. Find ways to allow your engineers to give back to the OSS you depend on.

As practicing engineers...

Mentorship is a thing you should drop everything and focus on getting better at. Now.

As educators...

Maybe everyone doesn't need a CS degree. Maybe we have to figure out ways for people more quickly and affordably learn enough to become an application developer. Maybe this is OK.

What does this all mean?

As managers...

Y tors. Put a
g r. Find ways to
allow your e d on.

As practicing en

Mentorship i cus on
getting bette

As educators...

Maybe every ve to
figure out wa earn
enough to become an application developer. Maybe this is OK.

ALSO, WE SHOULD NOT FORGET TO ACTUALLY TRY TO INVEST BACK IN OPEN SOURCE.

It's also not enough to throw money at a project. You actually need to invest in its community.

ON MENTORSHIP:

A good resource to start...

by **Gergely Orosz**

Home
About
My Reading List
Popular Articles
Talks
Newsletter
Work with Me
RSS Feed
twitter
linkedin

Subscribe via email
Subscribe in a reader

https://blog.pragmaticengineer.com/developers-mentoring-other-developers

Developers mentoring other developers: practices I've seen work well

15 AUGUST 2019

How does mentoring work? I asked this question ten years into my software engineering career when I joined Uber. Until then, I've never received or done mentoring, or at least never put this label on any activity I've done before.

Uber, however, had an official mentoring program. Almost every engineer I met had a mentor. Mentorship is an expectation for senior and above engineers, it being listed in our engineering competencies. Since working here, I've been mentored, been a mentor, and have observed engineers around me grow via mentorship.

Mentorship has been the best things that's sped up my growth and others engineers around me. **This post discusses mentorship practices that work well engineer-to-engineer**. The practices come from my own experience, observations I've made people mentoring each other

ON MENTORSHIP:

A good resource to start...

by Gergely Orosz

Developers mentoring other developers: practices I've seen work well

15 AUGUST 2019

Mentoring that we already do:

- Onboarding
- Informal mentorship (e.g., code reviews)

Mentoring that we should do more of:

Formal mentorship is more effort, but provides more opportunities for growth. This kind of mentorship is rare!

So far, only: structured at (typically) large tech companies, or within online communities (e.g., CodingCoach)

https://blog.pragmaticengineer.com/developers-mentoring-other-developers

ON MENTORSHIP:

A good resource to start...

by Gergely Orosz

Developers mentoring other developers: practices I've seen work well

15 AUGUST 2019

Mentoring that we already

- Onboarding
- Informal mentorship

Check the article for a more detailed dive into a structure for doing more formal mentorship

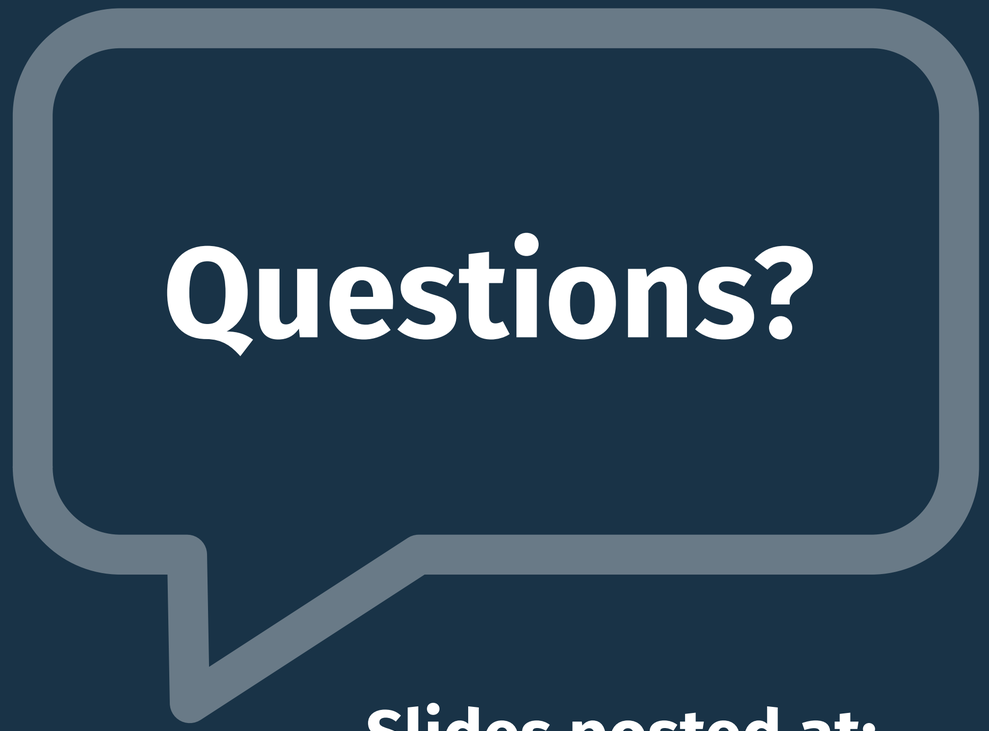
Mentoring that we should do more of:

Formal mentorship is more effort, but provides more opportunities for growth. This kind of mentorship is rare!

So far, only: structured at (typically) large tech companies, or within online communities (e.g., CodingCoach)

https://blog.pragmaticengineer.com/developers-mentoring-other-developers

THANK YOU!



Slides posted at:

https://speakerdeck.com/heathermiller/open-source-numbers-everybody-should-know-bobkonf