Infrastructure in Recession

Economic impacts of tech layoffs on FOSS communities

By Mike Nolan and Kripa Kundaliya

DIGITAL
/NFRASTRUCTURE
/NSIGHTS
FUND

Table of Contents

Introduction

The rise of open source software contribution and use has been accompanied by an increasingly direct linkage with corporate software development economics and life cycles. The 2021 report by O'Niel et al. shows that for many critical projects, the vast majority of contributions are now made by firm employees (O'Neil et al. 2021). This increasing reliance on corporate contribution and support has coincided with a massive explosion in the amount of critical open source software that underpins a wide variety of sectors, such as healthcare, transportation, and defense. As the US tech sector enters its third year of layoffs and budget cuts, the implications extend far beyond these companies' products (Sayegh, 2024) and into infrastructure that underpins other critical sectors.

The massive layoffs and rebudgeting within tech firms has the potential to disproportionately affect labor and resources aimed at sustaining open source software. This, in turn, could lead to more fragile digital infrastructure and increase pressure on already under-resourced maintainers to prop up a system whose maintenance may already be far beyond their capacity. Existing literature has reported on the individual experience and expectations of some maintainers, but little is known about the macroeconomics of open source software financing and maintenance.

This research builds upon earlier work of O'Niel with the aim of better understanding the systemic risk that major financial shifts may cause on vast technical systems such as the open source software commons. After reviewing the body of research on open source funding and maintenance, this paper builds a theoretical framework of engineering risk based on a case study of the Boeing 737 Max crashes, analyzes the findings of our empirical research, and provides guidance for policymakers regarding open source. The case study of the 737 Max may initially seem like an unusual comparison to make towards open source software but this was chosen as it provides evidence towards the impact of how long running technology systems (aircraft) are impacted over decades long financial trends. Open source software shares this quality in that many critical infrastructures are used and maintained over extremely long periods of time and also often play a key role in safety critical applications.

Literature Review

Open Source Software (OSS) serves as the foundational infrastructure for nearly all connected software systems today. This software allows for open access to anyone online and is licensed under terms that allow users to use, redistribute, and modify it without liability (OSI 2006). OSS is made up of millions of different software packages, each facing a wide variety of usage and maintenance. This massive web of software makes up a complex supply chain of software tools that underpin critical software systems across sectors such as transport, healthcare, communications, and defense. A study by Hoffmann et al. suggest that the cost for private sector firms to recreate their utilized open source software totals to $8.8 trillion while the existing cost of creation for that same sector only totals to $4.15 billion (Hoffmann, Nagle, and Zhou 2024).

This massive cost savings experienced by firms through utilizing open source software has encouraged total reliance on shared infrastructure in order to maintain firm profitability. However, as OSS has become more pervasive within proprietary business software ecosystems, allowing for infrastructure cost sharing between firms and “volunteer” independent contributors, so too has the proportion of software contributors and maintainers who are doing so on behalf of firms (O’Neil et al. 2021).

This increasingly centralized source of labor power within OSS creates a structural economic risk related to the reliability, security, and stability of this software. The systematic literature review conducted by Li et al. found that while firm involvement in open source software communities do bring additional labor to the project, it also fundamentally alters the project’s openness and goals in favor of those of the firm. Furthermore, while firms do bring additional labor to projects, existing literature has also shown that firm involvement tends to “bring challenges for volunteers’ long-term contribution” such as lack of transparency, domination of firm goals, and a general decrease of participation of volunteers over time (Li et al. 2025).

TideLift’s 2024 survey on the state of open source maintenance shows that only 12% of maintainers receive all of their income from OSS maintenance, with an additional 24% paid on a part-time basis to do so (Tidelift 2024). JetBrains Developer Ecosystem survey shows that in 2024, nearly 3/4ths of respondents experience burnout (JetBrains 2023). Intel’s 2023 Open Source Survey showed respondents most often ranking Maintainer Burnout as the most critical challenge to OSS that year (Intel 2023).

Reports of maintainer burnout show a unique change in perspective amongst long-standing OSS maintainers. Still, quantitatively, it is difficult to determine the impact this burnout has had on the amount of improvement and innovation within OSS globally. The GitHub Octoverse reports a nearly linear increase in total contributions year over year (YoY) from ~600M in 2021 to nearly 1B in 2024. These contributions coincide with massive growth in global south countries such as Nigeria, Columbia, Philippines, Morocco, and Kenya, each achieving 20-30% YoY growth in OSS developers (GitHub Staff 2024). What is absent, however, are growth statistics related to the US developer growth and statistics specifically related to maintainership. Still, this data reflects an increasing trend of periphery countries having greater amounts of software workers within their labor force.

New funding methods popularized by open source “tip jars” platforms such as Github Sponsors or fiscal sponsorship platforms such as Open Collective have seen increasing levels of adoption (Open Collective 2024) but still face structural issues of increasing the precarity of open source workers. Chris Aniszczyk, CTO at the Cloud Native Computing Foundation, argues this method of open source funding is effectively turning those maintaining infrastructure into gig workers and ends up “having developers run in a hamster wheel of no benefits and protections offered via normal employment schemes” (Chris Aniszczyk 2019).

Data on funding for OSS is sparse, even within companies; the 2024 Open Source Funding Survey has noted that “Owing largely to its scale and potential for underreporting, OSS contribution ***labor is a critical blind spot*** for organizations.” (Boysel et al. 2024). Despite this, the report estimates that roughly $7.7 billion USD is spent every year on funding open source software development, with 86% of that being through direct labor contributions. It is important to note that estimates from other researchers vary from €1 billion to $37.8 billion (Bureau of Labor Statistics, n.d.; Korkmaz et al. 2024; “The Value of Open Source Software - Working Paper - Faculty & Research - Harvard Business School,” n.d.; European Commission. Directorate General for Communications Networks, Content and Technology. 2021). This large variety in estimates exemplifies how open source labor is a largely unmeasured cost center in private business. This is particularly relevant as businesses re-assess what cost centers are worth investing in and begin cost-cutting efforts. In open source software, maintainers already face pressure to only deliver on certain types of work, such as tasks related explicitly to programming, and ignore other tasks, such as community management even when maintainers perceive them as valuable (Mel Chua and Stephen Jacobs 2020).

Jacobs and Chua show how labor regarding the management of an open source project’s community is seen as valuable yet still deprioritized in the PyPI community. This cultural deprioritization of non-core tasks in open source can lead to communities being unable to effectively continue development as collaboration becomes increasingly difficult due to a lack of non-technical capacity within the project. Many critical open source projects rely upon a combination of maintainer free time and external assistance to develop this non-technical capacity.

As budget cuts begin to affect open source projects, the technical risk borne by a lessening of non-technical capacity and auxiliary resources will likely not be immediately recognized in catastrophic failure or detrimental side effects. Instead, this risk will remain and develop as individual capacities are strained by the expectations of key stakeholders.

To better understand how this risk develops within technical systems, this paper will explore a case study on the development of the Boeing 737 Max aircraft and the corresponding crashes in 2019. The development and engineering of aircraft include many particularities, regulations, & considerations that do not map directly onto open source digital infrastructure.

The fundamental forces that link these two situations together is the shared relationship between capital and labor. All for profit enterprises have an incentive to maximize profits for owners. This can be done by either expanding markets or, when markets saturate, reduce operational costs. The reduction of operational costs will ultimately impact how production happens, creating additional pressures on the labor force, to produce more with less. This cost-cutting may find efficiencies in production but also incentivizes the production of lower quality commodities. The case study of Boeing Company and the 737 Max provide us a window into these two forces and how they influence technology production over time.

A Case Study for Systemic Engineering Risk - Boeing

On August 01, 1997, Boeing announced that it had completed its merger with one of its two primary competitors, McDonnell Douglas, the third largest commercial aircraft manufacturer after Boeing and Airbus. This merger promised the ability to combine two American aircraft giants and utilize shared efficiencies to cut costs and pass those savings on to the consumer. Some economists have developed models that suggest that despite this merger resulting in a duopoly between Boeing and Airbus, it saved consumers as much as $5.14 billion (An and Zhao 2019).

Twenty years later, Lion Air flight JT 610 departed from Jakarta, flying a new Boeing 737 Max 8 with roughly only 800 hours of flight time. At 6:20 am, 44 seconds into the flight, sensors noteed two different airspeeds and altitude readings; ATC advised the captain to climb higher. The pilots attempted to climb higher despite contradictory readings of altitude, speed, and Angle of Attack (AoA). At 6:23 am, an automatic system (MCAS) within the plane begins to force the plane nose down, interpreting the contradictory AoA data and wresting control away from the pilots. The pilots frantically flipped through the manual, trying to understand what was happening and how to regain control of the aircraft. During this time, MCAS activated 20 separate times, pushing the nose of the plane down. Each time, the pilot pulled the nose back up to regain altitude. At 6:30 am, the captain asked ATC for permission to land and was diverted. The two pilots were navigating a plane with no functioning instruments while trying to diagnose what was going wrong and how to fix it. During this time, the plane continueed to intermittently try to push the nose down into the ground.

After little more than a minute and three more MCAS interruptions, the plane's black box data showed a descent of more than 10,000 feet/minute. Flight JT 610, with 189 souls on board, crashes into the Java Sea, killing all on board. Nearly five months later, Ethiopian Airlines flight ET 302 experienced a similar story, with the plane's MCAS system causing it to crash and killing all 157 souls on board (Soerjanto Tjahjono 2018).

The story of these two 737 Max crashes is often centered around the MCAS system, developed to counteract the new larger CFM International LEAP engines whose new placement created instability, causing the nose of the aircraft to pitch upward and cause potentially dangerous stalls. The MCAS software would use sensor data to digitally modify aircraft controls and pitch the plane downward to counteract this inherent instability. On the surface, the story of the Boeing 737 Max may seem like a faulty system, shipped too early without being caught by Quality Assurance (QA) processes, a bug accidentally shipped to production. This story belays the reality of aircraft engineering; aircraft are full of redundant systems, built to expect failure, and still safely operate until the aircraft can land and ensure whatever broken system is fixed. Engineers understand that within highly redundant systems, critical failure must be attributed to a systemic failure, unwinding overlapping systems of redundancy until none are left.

During the twenty years between the McDonnell Douglas merger and the 737 Max crashes, a slow unwinding of redundant systems, safety checks, and engineering methodology designed to

prevent catastrophic failure happened. Since merger between these two companies, a shift, not just in the mindset of engineering teams, but also in economic incentives and requirements for Boeing to maintain its rate of profit in an increasingly saturated market began.

Changing Times

Despite the Boeing company being significantly more profitable, larger, and the company acquiring McDonnell Douglas, it was McDonnell Douglas that was considered the “winner” of the acquisition. Even former Boeing Executives such as Ron Woodard said that it was really “McDonnell Douglas who bought Boeing with Boeing’s money” (Frost 2020). After the acquisition, McDonnell Douglas executives took over many positions of executive leadership, such as COO, and now held the largest portion of shares in Boeing. New executives such as Harry Stonecipher quickly implemented new cultural regimes, shedding away the family-like union atmosphere and threatening “underperforming” employees with termination (Frost 2020). The term underperforming, in this case, refers to a lack of productive output and not necessarily reduced quality of work. This change in strategy was necessary, according to management,to maximize shared efficiencies between the newly merged companies and pass that value onto the customer while also growing Boeing’s market share (An and Zhao 2019).

Being more “efficient” and passing savings onto the consumer allowed Boeing to grow further and maintain its position as one of the predominant commercial aircraft manufacturers in the world. This philosophy of efficient iteration can be seen in the changing design philosophy of the 737. The 737 was an airplane designed at a time when regional air travel was a nascent market in commercial aviation. The average amount of seats needed for a regional jet was comparatively low. Cargo was still largely loaded manually by hand, requiring the airframe to be near to the ground. The commercial jet market had not yet developed to a point where seat-mile cost required maximal fuel efficiency and, therefore, large turbo jet engines to utilize the principle of Carnot efficiency (Travis 2019).

A universal experience of commercial software engineers is sitting in a meeting with a manager discussing the pros and cons of incurring technical debt to ship the next iteration of their product or investing time in re-architecting their existing system to allow for more stable improvement. The amount of existing research on technical debt provides little consensus on its definition and how to measure it (Lenarduzzi et al. 2020). The value of shipping the next iteration is easily measured, and the cost of the systemic risk incurred by technical debt is nearly impossible to measure meaningfully. To a data-driven manager, sponsor, funder, or executive, the decision is an easy one.

For years, employees struggled with management sending dual messages of desiring feedback yet also being punished for providing feedback that isn’t wanted. After this culture became ubiquitous, came the announcement of Airbus’ new A320 New Engine Option (NEO) (Sterman and Quinn 2023). The new larger engines fitted allowed a massive 15-20% increase in fuel cost savings on average, a cost savings needed to compete in an increasingly saturated regional

commercial aviation sector. Boeing had to scramble with long-time clients threatening to adopt the Airbus aircraft if Boeing could not provide similar offerings.

The 737, initially designed in 1964, as opposed to the A320, designed in 1987, was designed for a different market, at a different time, with different technology. It has been iterated to meet changing customer needs while still maintaining the same general airframe and flight experience. Now, Boeing executives were faced with the same familiar question: Do they ship a new version of the 737 with larger engines on a shortened timeline to compete with the new A320? Or do they mitigate the systemic risk of building onto an outdated airframe by totally redesigning the next 737 with a new airframe but potentially lose important short-term business to their competitor? The decision was an easy one.

Boeing managed to deliver their answer to the A320 NEO, the 737 Max just a year after the A320 NEO. Design and engineering teams worked at a pace never before seen, delivering, on average, double the amount of technical designs on tight budgets (Sterman and Quinn 2023). Boeing built a plane that claimed to fly virtually the same as every 737 built since 1967 yet be radically different in order to use massive new engines. Two years after its launch, critical flaws in the design of the 737 Max killed 346 people.

The 737 Max showcases the realities of how structural engineering risk is realized as technical debt. The form of technical debt will depend on the realities of the system being constructed but ultimately results in additional and unnecessary complexity being added rather than making the structural changes necessary to ensure a simpler more controlled system. It is this complexity that creates risk. This risk can be managed through additional QA, testing, oversight, and training. However, when complexity is increased due to cost-cutting, it is unlikely that the costly external QA systems will be upgraded to account for it.

Incurring technical debt was not an explicit decision in the case of the 737 Max. It was a result of continual prioritization of core production over all other auxiliary processes on a complex time frame at a reduced cost. This process did not begin at the development of the Max but rather began with the economic and cultural shift in 1997. Risk began accumulating with changing engineering priorities, a plateauing of regional aircraft markets, and a race to the bottom in order to ensure maximal profits in commercial aircraft development.

The case study of the 737 Max can show how risk is developed into a technical system, and the role economics play in creating that risk. It does not necessarily provide the answers or a formula to determine the exact outcome of that risk, nor does it provide an example on how to mitigate that risk. This framework can still serve as an early warning system in other sectors that develop critical technical infrastructure to spur further research on how to prevent catastrophe.

Methods

Research Design

This study seeks to develop a theory on the relationship between long-term financial shifts and their impact on structural risk within a long-lived technical system. It will utilize a limited scope empirical study to provide evidence towards tech sector layoffs and rebudgeting and the creation of technical risk within open source digital infrastructure. Due to the largely opaque nature of private sector labor investment and finances, the authors decided to take a mixed method approach, utilizing an analysis of available funding data combined with maintainer interviews to provide more in-depth perspectives of the realities maintainers are facing. This study also utilized an existing case study to better develop a theory of engineering failure related to firm economics.

Quantitative Methods

Data Collection

Quantitative data were collected through a variety of APIs provided by open source fiscal sponsors such as Open Collective and NumFocus. Other sponsors, such as the GitHub Sponsors program, provided some basic metadata but would not reveal data specifically related to funding, such as amount or sponsors. For data collection, the authors modified existing open source tools initially developed by the Georgetown CSET lab. Our team added additional capability to mass query time series data in order to analyze per-project funding over the last 5 years. Additionally, we developed a web front-end to visualize funding amounts and show projects with downward and upward trends.

Data Analysis

Once data was collected, linear regression analysis was performed on individual projects to determine trends over a five year period. High level statistics were then calculated looking at aggregate trends of all projects, total funding amount per year, and platform wide funding trends. Projects with negligible funding were excluded in order to eliminate outliers.

Qualitative Methods

Data Collection

Qualitative data was collected through semi-structured interviews with 12 participants that lasted an average of 44 minutes. Participants were recruited through outreach in popular open source maintainer groups such as the GitHub Maintainers forums, popular mailing lists related to open source foundations, and direct outreach. Our direct outreach strategy focused on existing contributors to previously heavily sponsored projects. Our sampling strategy prioritized maintainers who feel they had faced some change in their relationship with firms, funding, or stability in the last 3 years. This bias was deliberate in order to uncover relevant stories of maintainers, which means that our interviews are not necessarily representative of the experiences of all maintainers. Further participants were recruited by means of snowball sampling, allowing us to network among open source organizations and communities.

Title	Project Type	Project Sector	Organization Type
P1	Documentation	Web	Sponsored Project
P2	Funding	Corporate	Corporate Maintenance
P3	Software	Infrastructure	Independent
P4	Funding, Software	Corporate	Corporate Maintenance
P5	Community, Software	FOSS Community	Sponsored Project
P6	Software	Infrastructure, Corporate	Fiscal Sponsor
P7	Software	Corporate, Web	Independent
P8	Funding, Software, Community, Documentation	Web, Infrastructure, Hardware	Foundation
P9	Funding, Software	Corporate, Infrastructure	Foundation
P10	Software	Corporate, FOSS Community	Corporation
P11	Research	FOSS Community	Independent
P12	Software	Corporate, Infrastructure	Corporation

Table I: Interview participants

The interview protocol began with an introduction, informed consent, and better understanding participants' background and relation to open source. Many participants had a variety of relationships with different projects, organizations, and companies. Interview questions encouraged participants to speak broadly about their own experiences, allowing them to incorporate themes of experience across many different projects. The themes of interview questions began with introduction & context setting, their relationship to open source funding, changes in contribution & culture, and planning for future changes in their projects & organizations. Interviews were entirely conducted remotely via Zoom.

Data Analysis

Interview transcripts and recordings were first analyzed using thematic analysis to identify specific themes regarding struggles, strategies, and economic realities faced by participants. Insights were then developed taking a grounded theory approach, comparing experiences and themes of participants to the realities of the Boeing case study to determine similarities and differences. All interview coding and analysis was conducted using the Dovetail software suite. While Dovetail does provide "AI" analysis features, these were not used at any point during the analysis of data.

Consent & Privacy

All participants provided informed consent prior to participation via a signed form as well as verbally as their rights as a participant were restated prior to the interview. Participants were afforded to review their quotes used within the context of the study prior to the publication of the study. Furthermore, all names and organizational affiliations of participants were kept confidential. Interview transcripts were only accessible to the two authors.

Limitations

Although the mixed methods approach of this study provided a comprehensive analysis of the current economics of open source, significant limitations regarding the availability of public funding data and large samples of open source maintainers willing to speak candidly regarding economic struggles. Even when having access to suitable funding sources, maintainers face particularly precarious employment/funding conditions. Significant amounts of literature regarding firm sponsorship and involvement within open source requires self-reporting and reports are often released by the firms themselves (JetBrains 2023; Tidelift, n.d.; Nikki McDonald 2024; Staff 2024).

Results

Three primary topics were consistently mentioned amongst interview participants, covering struggles related to finance, contribution, and logistics of running an open source project. For each of these topics participants discussed existing struggles and occasionally inadequate solutions often implemented by firms that worsen pressures on maintainers.

Financial Struggles

Diversifying Income

More community-oriented projects are seeking to diversify their revenue sources to reduce reliance on a few major sponsors and mitigate risks from events like mass layoffs.

> *[Our first priority is to] diversify funding sources because if the funding is just like coming from one aspect or one location, if anything happens to that location, it will affect the community or the project,*
>
> - *P5*

- *We do not have a very diverse income. I wish it would be more diversified. … I'd rather move us from having one or two major sponsors that we now really depend on ... to maybe having four or five smaller sponsors which we can rely on long-term..*
  - *P1*

Strategies for many foundations include pursuing government grants, as well as expanding and integrating with services related to open source maintenance & training.

… these sort of nontraditional revenue share, like Hero Devs … I haven't seen that anywhere else, and I feel like that's the type of collaboration that really benefits everybody.

- *P8*

A lot of our expenditure … goes to developer, collaboration, IT and infrastructure for projects ... We fund marketing, training and certification work.

- *P9*

Working to Survive

As full-time maintainers struggle to find ways to sustain their own position as someone responsible for software, they also increasingly struggle to develop communities of volunteers to assist in both the development and maintenance of their software. Job duties are increasing lending to an environment where volunteering is more difficult for individuals and even when this labor is potentially beneficial, individuals do not necessarily have the capacity to make that justification and carry it out.

I don't pay [my contributors] anything and there's no stipend … [Initially,] people were just like, "OK, let's do it" but as time goes on … those that are honest enough to tell me, "I don't feel like I can do this. I got a second job because it's remote so I need to concentrate on my job."

- *P5*

When your performance doesn't necessarily tie to [open source contribution], along with constant shifting priorities, it makes it pretty much impossible to work on open source well.

- *P6*

Non-Essential Expenses

Due to the availability of free open source software, there is increased scrutiny on justifying payments and demonstrating the return on investment (ROI) for specific companies. Projects and members struggle to connect their work to business value, making it difficult to quantify the impact on the organization's bottom line. As a result, during economic downturns, open source program offices (OSPOs) within companies like Facebook, Google, and Microsoft faced deep cuts, especially if their contributions were not directly tied to revenue or business outcomes. The Linux Foundation also experienced challenges due to economic conditions and membership churn, leading to a need for greater fiscal conservatism and efficiency.

Paying money to open source that you can otherwise get for free is seen as an inefficiency. And so at all layers of the stack, there's a lot of pressure for efficiency.

- *P4*

[Open source maintainers & contributors] don't do a good job of connecting work to business value … And when you are now in a, in a place where you have to show … how do 'you' specifically help impact the business to hold on to your job? That really forces the person, the manager, like everything, to focus more internally because it's a lot easier to quantify that kind of impact.

- *P6*

Here's what I've seen. The first thing was many of the OSPOS were cut deeply. Facebook, there were like two people left in the OSPO … Google, let go a whole bunch of people in the OSPO … The companies where the OSPO was set up as not being tied to revenue and not really tied to outcomes really struggled … The way that companies need to think about open source is … hard to quantify. It's not the same as [spending] million dollars here and we get $1.5 million out … it's a different type of calculation.

- *P8*

If you kind of look at our challenges, economic conditions and membership churn are impacting the budget a little bit. So we have to be a little bit more efficient and a little bit fiscally conservative than we maybe have in previous years.

- *P9*

Meaningless Tips

Some of the most popular platforms (Github Sponsors and Open Collective) for attempting to alleviate the financial burden placed upon open source maintainers center maintainers as volunteers to be sustained through donations[1] rather than employment. Some interviewees believe these crowdfunding approaches are insufficient and do not adequately compensate for the value created by open source projects. They argue that companies should contribute full-time employees or significant funding to foundations supporting open source development.

GitHub has its tip jar for open source maintainers … All of those things that I've seen don't … seem to scale big enough. We really need companies with full time employees contributing to this stuff … It's really hard to scale up with lots of $5 tips here and there … [Firms] contributing full time employees or contributing large amounts of money to a foundation seemed like they were on the right track and I hope that we stay on that track.

[1] Platforms such as Open Collective are also frequently used as fiscal sponsors for grant based funding and contracting work.

- *P12*

Regardless of how much github sponsors is paying, regardless of how much open collective is paying for projects, it's insignificant compared to the value that they're creating … [Maintainers] should not be begging for sponsorship, which is what the github sponsors program represents to me. It's like, why do I have to beg for money?

- *P2*

Donations sort of inherently ebb and flow with economic tides. They tend to be sort of trend driven as well. [For example,] people are not doing a lot of charity in the Java space even though it's huge.

- *P4*

Contribution Struggles

Less volunteerism

Participants reported that layoffs and the general economic environment of firm involvement in open source communities have significantly impacted volunteer contributors' morale and general involvement. Not all types of projects were impacted equally, with some reporting that single maintainer projects and core maintainers on community-oriented projects saw a dropoff in contributions.

I think in terms of people who are active, regular core contributors to our project who aren't [employed by our firm]. I think there's definitely some uncertainty there. … We had just some people who were very involved and active with the project but are not active and involved with the project now. There was definitely an emotional experience that happened there.

- *P10*

The percentage, if you ask me if the percentage of contributors in open source were like 80% since [before the] layoffs, has reduced to like 60%.

- *P5*

We just did like a heartbeat survey to try and reach out to [single maintainer projects] … A whole bunch of the small projects, we haven't heard back from and I really don't know why yet. My suspicion at this point is that we'll see a decrease in activity on single-maintainer projects.

- *P8*

I've certainly seen a decline from core contributors that used to give content updates or write new content every week or every month … Those are the kind of people that we would really like to be part of the community and be there often. In our project only 5% of the contributors contribute more than 10 pull requests in a year. If any one of them leaves, you notice.

- *P1*

The Next Generation

Many participants highlighted that many current maintainers are older individuals who had more flexibility earlier in their careers to contribute to open source projects than what is typical for younger people. Additionally, some noted there is a lack of organized efforts for generational handoff of critical open source software that will be needed for decades to come and some say that there used to be a pipeline of early career individuals contributing to open source to get noticed for job opportunities, but this has decreased due to hiring freezes at major tech companies.

Most of the maintainers of top-end open source tend to be older, white, male. It's sort of a generational thing where they benefited from having free time and flexibility earlier in their careers. So they were able to do open source … Some of these pieces of code are going to be with our society forever … We have no theory as a society of how we're gonna maintain [long-lived pieces of software].

- *P4*

These days, it's not really like all that before [where employment seemed more stable] because people are just … trying to get a job and where you can work, and they pay you. … Generally, people are not really that motivated to just go into open source and contribute because they feel like, why not [dedicate effort] to my work and do something that will pay me for.

- *P5*

There's often this pipeline of early career people … who will start to contribute to open source in order to be known by people at the big companies who will then come to them to be hired … It's a way to get yourself known in the network of people who are hiring … I have definitely seen a decrease in that kind of thing over the past couple of years … It's well known that Microsoft and Google and Meta are not hiring, they've been in hiring freezes for a couple of years.

- *P12*

Burnout Exit

Burnout amongst maintainers was a ubiquitous problem among nearly all participants regardless of organization type. Volunteers in particular facing burnout would often lead to negative outcomes for open source projects such as decreased participation from stakeholders, fragile ownership by firms, and lack of maintenance/security.

You can review PRs and see the enthusiasm drop off … And soon enough … you'll see the tone of the maintainer change [where they say], "feel free to fork the project or raise the PR yourself." That's essentially killing the project because either [a company like] AT&T or Adobe forks the project and starts maintaining a copy of it themselves, whereas it would have been significantly cheaper for them [to pay the maintainer].

- *P2*

The zeal [from people I know] that wanted to join open source contribution … If I look at it now, it's reducing.

- *P5*

Our thesis was that many maintainers were becoming frustrated and burnt out. In order to keep the system viable, sustainable … there would need to be investment.

- *P4*

AI Enshittification

AI products are being proposed across the software industry, particularly towards open source maintenance as a solution to the increasing amount of pressures maintainers face. Many of the participants expressed largely negative views and experiences related to AI systems, citing how short term benefits of AI systems are largely outweighed by a number of negative side effects.

Some noted that AI products, even useful ones, are creating spam, increasing pressure on maintainers in many ways through deliberately raising expectations while simultaneously lowering the amount of resources directed towards human maintenance. Others simply could not see major benefits from existing systems while seeing their own projects being used unwittingly to power them.

People [still] want to be open source maintainers. But as you've brought about all these bots, all these security dashboards, and you've got these companies pushing automatic bots and automatic issue creation and raising PRs about security vulnerabilities and blah, blah, blah. Well, maintainers on successful projects are just sick of it. They're like, oh, I'm getting 1000 PRs, and I'm not getting paid for this. Why do I need to work on all these PRs?

- *P2*

[Regarding] this trend toward AI and this expectation among some that these jobs aren't coming back, that they'll be replaced with AI, I'm not personally of that opinion. I think that actually what we're seeing is how much human experience is required for this kind of work.

- *P12*

We're content creators, and we believe that human-created content can very well be consumed by these large language models and whatnot. But our job isn't necessarily going to be replaced by these things because we're actually feeding these machines. … I know that Reddit sold its content for $200 million or something? Well, our content is CC BY-SA, so we don't own any licenses because it's open source, and we believe in open source, and we want to give it out broadly. But, with AI, it's like, you just take, and this content was created for free, but it's not really for free because some people spend time on it, some people maintain it, and those people need backing.

- *P1*

Techno-Colonialism

Many interview participants discussed the topic of increased involvement of firms trying to utilize workers from Global South (GS) countries in order to both bolster the amount of labor being dedicated to open source software maintenance and utilize the economic precarity of GS tech workers to increase cost savings from firms on open source maintenance. Many of these programs are branded as DEI but rarely provide the same economic securities of full-time employment that are found in the Global North.

There was a recent open position that they decided to not fill in the United States but fill in Eastern Europe instead because it's a lower cost location … Those positions tend to be sort of one or two year contracts. So they're always a little bit tenuous. And we always wonder, are we gonna have the same number of slots … every year?

- *P12*

I'm seeing way more people from West Africa and South Asia and parts of [Latin America] that 10 years ago would be really unusual to see someone from this part of the world … in the project.

- *P10*

I am part of the [Github] All In Africa program … I found out some, some of them, they have courses to [guide newcomers to open source contribution] … The whole thing is that generally people are not really that motivated to just go into open source and just contribute because they feel like, OK, why not [seek paid work opportunities]?

- *P5*

Logistical Struggle

No travel, events, etc.

By far the biggest areas of direct budget cuts experienced by most participants were related to travel, events, and in-person coordination activities related to open source projects and communities. Across the board, there has been a noticeable decrease or ceasing of firms

spending on employee travel and sponsorships for open source related events. Firms have effectively shifted this cost onto their employees or non-profit organizations (who are also seeing decreased sponsorships from firms).

> *[My employer has] made it much harder for [company] employees to attend [the leading open source conference in our field]. They weren't funding nearly as much travel; people had to use their own funds to travel. Even people whose job it is to speak at conferences, like that's their job title … I did have a conversation with one of the people who organized this [conference] this year. They were worried that they may have to reduce the space in a couple of years … They have [also] seen this decrease in sponsorship overall.*
>
> - *P12*

> *Events are really key motivational tools for communities and problem-solving spaces for communities … and those have been very heavily impacted by cutting back on sponsorships … A lot of places have just stopped doing that altogether, though.*
>
> - *P4*

> *[We] used to but we didn't run an event this year because of budgets being cut. A lot of folks weren't able to travel. … [We] put aside from our budget about $60,000, $75,000 to help people in the community go to events. We ask that people ask their companies first. All companies have cut back on that spending.*
>
> - *P8*

> *What we have noticed is a lot of companies have … minimized travel for a lot of their maintainers and people. We [for] this year already … we probably paid three times the amount of maintainer scholarships in general.*
>
> - *P9*

> *I'm just noticing that there's a little bit more of a scrutiny happening now towards what I would call community events. So events … more focused on like contributor communities … These events don't necessarily turn revenue because we're not going there to sign deals and meet with customers, although sometimes there are customers in these spaces.*
>
> - *P10*

Lack of Long Term Planning

Particularly due to a combination of less opportunities for in-person coordination events as well as increased pressures on maintainers to fulfill multiple roles, long term planning of project roadmaps and sustainability has become increasingly rare.

> *A lot of this strategic big picture [stuff] … we had to drop a lot of that stuff and just focus on trying to just keep things going in the short term … doing the releases every six*

months because that's still gotta happen. … I'd say a lot of our bigger picture, strategic stuff suffered as a result of that.

- *P10*

What's become harder is doing the bigger things, doing the bigger changes because those do kind of require that in person meeting when things get really complicated in my own experience … For the really big multiyear projects, it's really helpful to get in a room with a white board and all be in one place. And that's the thing that's that I've seen that's really been cut back in recent years.

- *P12*

Coordinating Offices & Non-Technical Roles

Coordinating offices such as OSPOs, external foundations & coordinating bodies have become significantly less staffed and effective. Those doing coordination work also have struggled more to find work after layoffs. This showcases a general tendency of firms to now be devaluing coordination work outside of their organization, one of the primary benefits of open source software development.

OSPOs have been very difficult to work with. Meta for example, the person that still works there is a fantastic person. Like he's, he's awesome, but he just doesn't have the time. I also noticed this with Uber. Uber basically, like dropped off the face of the planet when it came to open source … So, yeah, I would say that companies have deprioritized a lot of the work related to supporting the projects that they've donated to.

- *P8*

As a result [of layoffs], we lost a couple of very essential roles to our project. One was someone who … had won the highest um award that an employee can earn in the company and was working on code of conduct and kind of community engagement … We had a program manager who managed the release schedule of [our project] ... also laid off.

- *P10*

The only pattern I could identify in that case is like most folks that were the technical work, maybe for a better way to put this. … They were able to find work a little bit more quickly than folks that were on the community management [side].

- *P9*

Discussion

It is clear participants perceive that layoffs within the tech sector can have much more far-reaching implications than simply an adjustment to the amount of labor dedicated to open source software production. Private sector tech firm budgets and business strategy has been shown to deeply impact the priorities of open source maintainers, technical quality, and the pipeline of workers who will maintain software systems over several generations.

The impact of firms on open source software extends beyond simply the software's effect on firm operations and infrastructure. Due to the shared nature of open source infrastructure, this software often underpins critical sectors such as healthcare, state operations, military, and beyond. Firms in many ways both direct and subsidize infrastructure that powers the entirety of the shared internet. Because of this, shifting priorities within the consumer digital sector may have incredible knock-on effects to sectors that otherwise would have little to do with this business.

Optimizing for "Core" Development

Across many different participants and types of organizations, a theme of many experiences of maintainers has been finding new ways to sustain the software development lifecycle as firms seek to find expenditures that cannot be directly tied to ROI. This re-direction of efforts to core operations bears similarities to those seen in the Boeing case study. Boeing's strategy of optimizing production resulted in less investment in long term engineering projects such as developing a new regional air frame that could more easily adopt larger engines.

Participants noticed that activities and organizations related to coordination, planning, and external collaboration were significantly affected by budget cuts and layoffs. Many in-person events and related travel have experienced significant financial difficulties, leading to a ceasing of coordination activities or shifting the financial burden onto workers to continue carrying out these duties. Staff positions related to coordination (particularly in OSPOs) have also seen significant cuts, leading to decreased efficiency regarding collaboration between organizations.

This trend hints at the possibility of an increasingly fractured open source ecosystem. While short-term gains may be realized through cutting budgets related to travel and external coordination, longer-term costs may be incurred upon all actors as open source project development fractures and focus inward upon core software utilized by firms. Coordinating offices and events serve as connecting bodies between workers at different organizations to come together and better understand the engineering needs of disparate organizations and sectors. Without these it is likely that workers will more likely seek to opt to develop new software particular to firms needs when an exact solution is not available.

Cost Cutting and Quality

The notion of software quality is not easily measured but many participants alluded to a potential significant long term decrease in quality of OSS due to various cost cutting strategies being implemented by firms targeting OSS. The introduction of tools and automation is normal in software development strategies. However, many participants expressed hesitation at the fervor related to utilizing generative AI technologies to massively increase the productivity of individual maintainers. These worries parallel similar worries of engineers in the Boeing case study who struggled to maintain quality assurances given engineering expectations (Kitroeff,

Gelles, and Nicas 2019) and eventually have been wholly replaced by automated systems (Dominic Gates 2019).

There is no doubt that generative AI will significantly increase the “output” of new code, documentation, issues, and Pull Requests. However, many expressed concerns related to supporting these processes to ensure quality and architectural considerations are taken into account. The forcing of these technologies and expectations on maintainers to do more with less has concerned many that they will simultaneously have to produce more with less dedicated resources to ensure that which they produce is of acceptable quality.

This climate of increased production and decreased community has led to many workers feeling a growing sense of alienation. Burnout and issues regarding morale have been common topics amongst interview participants and amongst the greater OSS community. This combined with little liability amongst maintainers creates an environment where structural risk related to software security and reliability can grow significantly before being realized as a catastrophic incident. Many have perceived that software security amongst OSS projects has already decreased due to maintainer burnout.

Generational Support and Long Term Sustainability

The nature of open source digital infrastructure being a shared component between many different technologies lends itself to a technology lifecycle that is longer than other types of software. With little centralized control over the software support and use, the process of ending a software lifecycle or continuing its support over many decades is unclear for many maintainers. The continual support by firms through direct employee contribution, supporting various community initiatives, and supporting training and early career positions within OSS served as an initial bulwark to prevent the continual aging of the OSS maintenance population.

Now, with increased responsibilities upon early-career employees, decreasing career opportunities for young software developers, and less capacity from maintainers to support newcomers to OSS communities, the generational handoff is becoming increasingly difficult. The aging of open source maintainers is a risk that will not be immediately felt. Late-career and half-retired maintainers have both the capacity and financial resources to continue maintenance as is desirable. The diffuse nature of OSS maintenance and development means there are few to no bodies to govern the labor that powers and sustains our collective digital infrastructure. Instead, many of these critical infrastructures are coasting on a golden age where developers had significantly more labor power and freedom to act against the whims of their employer’s shareholders.

Firms' preference towards outsourcing technology development to developing economies where they can take advantage of global pay disparities provides new opportunities for finding able workers to maintain the next generation of open source software. However, these outsourced workers are just as vulnerable (if not more vulnerable) to the types of economic exploitation faced by open source maintainers. Furthermore, creating increasingly global supply chains with the aim of cost cutting can have knock-on effects towards the quality of product. Engineers at

Boeing note that outsourcing oftentimes came with the side effect of lower levels of quality assurance and oversight (Cheong 2025).

All three of these impacts showcase that the shifting priorities of western technology firms are creating knock-on effects on open source infrastructure that are indirect and will be realized as structural risk rather than immediate qualitative change in that infrastructure. It takes years, even decades for infrastructure to degrade from lack of investment.

Recommendations

Risk cannot be eliminated when developing new systems. It is an inherent quality of innovation and a required part of developing new and useful technologies. Yet a mindset that ignores risk, particularly in favor of profit seeking, can and has led to horrible disasters. There is no silver bullet for managing the risk in a technical system but we can learn from previous disasters and current conditions to develop technology in responsible ways.

Formalize OSS Development

The initial permissive marketized vision of open source software development where the production and use happened in an entirely individualistic method made sense when the software was developed as simple discrete applications, and users had the technical ability to gain autonomy from these relations. Now, open source software spans across types and sectors and impacts users around the world daily. This software forms our critical digital infrastructure, and its scope extends beyond singular technical users.

This critical digital infrastructure (a subset of all open source software) must be maintained differently as its impact extends far beyond individual users. It is one thing to expect someone to pave their own driveway and another to maintain the road on which they commute to work[2]. Critical digital infrastructure maintenance must be formalized through organizations dedicated directly to this work. Private sector companies such as TideLift and HeroDevs, among others, have begun this work but still make maintaining infrastructure an optional task for private sector firms. Maintaining our roads and bridges is not an option but a requirement.

States must also directly fund our digital infrastructure and find ways to generate the necessary revenue to do so, whether through tax schemes targeting IT firms or fee-for-service schemes. Existing initiatives such as the NSF POSE program and the German Sovereign Tech Fund are a start but currently fund infrastructure in ways that are more similar to scientific research rather than infrastructure. More committed suggestions for funding to a comprehensive open source infrastructure can be found within proposals such as the “Eurostack” championed by Francesca Bria that proposes centralized funding of technological infrastructure within the EU and center “independent, open platforms to prevent corporate dominance and ensure transparency” (Francesca Bria 2024). Others still advocating for a state-led digital infrastructure program within the EU highlight the need for state control over key technologies because of the shifting

[2] Excuse the American centric example.

priorities of private sector corporations. Rikap et al. advocate for exactly this and explicitly note that “ States should also subsidize the development and maintenance of free software solutions where they do not currently exist that should run on the public-led cloud” (Cecilia Rikap et al. 2024).

Simplify OSS Infrastructure

Open source software is prized for its flexibility for contribution and use but with critical digital infrastructure, that flexibility leads to a massive proliferation of software packages, particularly as roles related to external collaboration within firms are eliminated. States, multilateral organizations, and business leagues such as the Linux Foundation must dedicate resources to minimizing this issue. This is no simple task, but if these groups can begin to formalize maintenance on a subset of digital infrastructure, it will also create clear leaders for technical solutions and provide a space for democratic control over shared infrastructure rather than an oligopoly of competing solutions maintained by competing tech firms.

Simplifying our shared digital infrastructure will also massively reduce the cost of maintenance by maximizing shared maintenance between users. This will also make explicit what is considered critical digital infrastructure and what isn’t, allowing for more effective economic planning and enabling greater amounts of innovation to be developed on top of reliable infrastructure. This simplification will finally allow for easier monitoring of infrastructure health. The CHAOSS community has made enormous strides in developing tooling to measure community health. Still, with a subset of critical infrastructure identified, third-party watchdogs will be able to monitor and report on infrastructure health.

Empower Engineers

If there is one clear cause of technical debt in the case study of the Boeing 737 Max it is the continual disempowerment of engineers and other technical staff. There are piles of evidence of engineers who foresaw problems but couldn’t speak up or were ignored. If firms, states, and multilateral organizations really care about high-quality infrastructure and avoiding harm, then the workers sustaining this infrastructure must have a seat at the table. Finding new ways to organize this labor and develop a collective voice for open source maintainers is critical in ensuring our collective safety. This will become more feasible as we formalize development, as with the formalization of funding and collaboration, so too will structures exist for maintainers to speak with one voice.

Create & Enforce Liability

OSS development works because of the elimination of liability when distributing software. Those can utilize open source software without worrying about liability to the original author and authors can distribute their software without warranty. This naturally creates an ecosystem

where development and contribution happens without bureaucracy. However, not all contributions are the same and not all use is the same.

Regulators must consider how open source is often being used as a scapegoat, allowing firms to offload the pressure of maintaining our critical digital infrastructure on individuals while benefiting greatly from it. Some amount of liability must be created to target those who benefit the most from our shared digital infrastructure. If maintenance of critical digital infrastructure is formalized, firms can “buy in” to that development through fees or contributions and limit liability as a form of insurance. The cost here would be significantly less than maintaining their own infrastructure and the economic benefits from avoiding catastrophic failures would also make this an ideal way to financially sustain critical digital infrastructure.

Conclusion

There’s no crystal ball to tell us what will come of the continual mass layoffs and fiscal austerity happening within the Western technology sector. However, what this research has shown is that the biggest impact being faced by maintainers of open source digital infrastructure is a lessening of capacity to explicitly collaborate, plan for the future, and consolidate efforts. Increasing pressure is being placed upon individuals to somehow fill in the growing gaps in these areas by dedicating more personal time to maintenance and paying for travel & associated coordination costs. These pressures imply a growing focus on increasing the core production of software while reducing costs.

The lessons learned from the Boeing 737 Max case study showcase how engineering culture and methods can be massively affected by corporate financing and demands. This change can happen slowly over years and decades but result in catastrophic failure that cannot be immediately unwound. Despite the increasing pressures on open source maintainers led by firm austerity measures, this does not necessarily mean we are doomed to repeat history. If firms, states, and multilateral organizations act now, massive cost savings could be achieved alongside high-quality shared digital infrastructure. To do so, we must formalize and simplify the development of our most critical digital infrastructure, allow engineers a seat at the table to advocate for this infrastructure, and have regulators develop liability on those developing and utilizing this infrastructure.

Future research is necessary to develop the methods needed to identify the most critical infrastructure, evaluate its quality, and recommend methods to rectify any issues. Infrastructure changes are important between sectors and geographies, and therefore, further research is needed to provide more in-depth insights into each relevant sector.

OSS is approaching a crossroads. The mass adoption of OSS signaled an era of a new form of software development, tearing down barriers and enabling mass innovation alongside effective cost-sharing. Now, we approach a new era with new challenges stemming from proliferation and decreased economic growth. With new policies and forward-thinking, this new era can drive us into the future, encouraging innovation with high-quality infrastructure. However, inaction may

drive us down a path of increasingly poor quality infrastructure that underpins the lives of millions of people. The growth of OSS has created a vibrant global community of intensely committed, passionate, and talented people who are more than capable of creating the infrastructure of the future. We have no excuses.

References

An, Yonghong, and Wei Zhao. 2019. “Dynamic Efficiencies of the 1997 Boeing-McDonnell Douglas Merger.” *The RAND Journal of Economics* 50 (3): 666–94. https://doi.org/10.1111/1756-2171.12291.

Boysel, Sam, Frank Nagle, Hilary Carter, Anna Hermansen, Kevin Crosby, Jeff Luszcz, Stephanie Lincoln, Daniel Yue, Manuel Hoffmann, and Alexander Staub. 2024. “2024 Open Source Software Funding Report.” November 19, 2024. https://opensourcefundingsurvey2024.com/.

Bureau of Labor Statistics. n.d. “U.S. Department of Labor, Occupational Outlook Handbook, Software Developers, Quality Assurance Analysts, and Testers.” Bureau of Labor Statistics. Accessed March 6, 2025. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm.

Cecilia Rikap, Cédric Durand, Edemilson Paraná, Paolo Gerbaudo, and Paris Marx. 2024. “Reclaiming Digital Sovereignty.” UCL. https://www.ucl.ac.uk/bartlett/public-purpose/publications/2024/dec/reclaiming-digital-sovereignty.

Cheong, Charissa. 2025. “I Worked at Boeing for over 30 Years. I Witnessed the Fallout of Outsourcing Firsthand.” Business Insider. January 6, 2025. https://www.businessinsider.com/worked-at-boeing-30-years-outscourcing-problems-787-2025-1.

Chris Aniszczyk. 2019. “Troubles with the Open Source Gig Economy and Sustainability Tip Jar.” March 25, 2019. https://www.aniszczyk.org/2019/03/25/troubles-with-the-open-source-gig-economy-and-sustainability-tip-jar/.

Cusumano, Michael A. 2021. “Boeing’s 737 MAX: A Failure of Management, Not Just Technology.” *Communications of the ACM* 64 (1): 22–25. https://doi.org/10.1145/3436231.

Dominic Gates. 2019. “Boeing Overhauls Quality Controls: More High-Tech Tracking but Fewer Inspectors.” The Seattle Times. January 20, 2019. https://www.seattletimes.com/business/boeing-aerospace/boeing-overhauls-its-quality-controls-more-high-tech-tracking-but-fewer-inspectors/.

El-Deeb, Ahmed. 2023. “The First Tech Layoff Wave After Years of Hypergrowth: How This Affects the Industry?” *ACM SIGSOFT Software Engineering Notes* 48 (1): 4–5. https://doi.org/10.1145/3573074.3573075.

European Commission. Directorate General for Communications Networks, Content and Technology. 2021. *The Impact of Open Source Software and Hardware on*

Technological Independence, Competitiveness and Innovation in the EU Economy: Final Study Report. LU: Publications Office. https://data.europa.eu/doi/10.2759/430161.

Francesca Bria. 2024. “X. European Digital Independence: Building the EuroStack.” *AI Now Institute* (blog). October 15, 2024. https://ainowinstitute.org/publication/x-european-digital-independence-building-the-eurostack.

Frost, Natasha. 2020. “The 1997 Merger That Paved the Way for the Boeing 737 Max Crisis.” Quartz. January 3, 2020. https://qz.com/1776080/how-the-mcdonnell-douglas-boeing-merger-led-to-the-737-max-crisis.

GitHub Staff. 2024. “Octoverse: AI Leads Python to Top Language as the Number of Global Developers Surges.” *The GitHub Blog* (blog). October 29, 2024. https://github.blog/news-insights/octoverse/octoverse-2024/.

Hoffmann, Manuel, Frank Nagle, and Yanuo Zhou. 2024. “The Value of Open Source Software.” SSRN Scholarly Paper. Rochester, NY: Social Science Research Network. https://doi.org/10.2139/ssrn.4693148.

JetBrains. 2023. “The State of Developer Ecosystem in 2023 Infographic.” JetBrains: Developer Tools for Professionals and Teams. 2023. https://www.jetbrains.com/lp/devecosystem-2023.

Kitroeff, Natalie, David Gelles, and Jack Nicas. 2019. “Boeing 737 Max Safety System Was Vetoed, Engineer Says.” *The New York Times*, October 2, 2019, sec. Business. https://www.nytimes.com/2019/10/02/business/boeing-737-max-crashes.html.

Korkmaz, Gizem, J. Bayoán Santiago Calderón, Brandon L. Kramer, Ledia Guci, and Carol A. Robbins. 2024. “From GitHub to GDP: A Framework for Measuring Open Source Software Innovation.” *Research Policy* 53 (3): 104954. https://doi.org/10.1016/j.respol.2024.104954.

Lenarduzzi, Valentina, Terese Besker, Davide Taibi, Antonio Martini, and Francesca Arcelli Fontana. 2020. “Technical Debt Prioritization: State of the Art. A Systematic Literature Review.” arXiv. https://doi.org/10.48550/arXiv.1904.12538.

Li, Xuetao, Yuxia Zhang, Cailean Osborne, Minghui Zhou, Zhi Jin, and Hui Liu. 2025. “Systematic Literature Review of Commercial Participation in Open Source Software.” *ACM Transactions on Software Engineering and Methodology* 34 (2): 1–31. https://doi.org/10.1145/3690632.

Marie Christine Duggan. 2024. “As Boeing Cracks, Is It Capitalism, or Kafka? | Dollars & Sense.” Dollars & Sense. October 2024. https://www.dollarsandsense.org/archives/2024/0924duggan.html.

Mel Chua and Stephen Jacobs. 2020. “Conceptual Mismatches.” Rochester Institute of Technology. https://github.com/FOSSRIT/mismatches.

Montandon, João Eduardo, Cristiano Politowski, Luciana Lourdes Silva, Marco Tulio Valente, Fabio Petrillo, and Yann-Gaël Guéhéneuc. 2021. “What Skills Do IT Companies Look for in New Developers? A Study with Stack Overflow Jobs.” *Information and Software Technology* 129 (January):106429. https://doi.org/10.1016/j.infsof.2020.106429.

Nikki McDonald. 2024. “Annual Intel Open Source Community Survey Results.” https://www.intel.com/content/www/us/en/developer/articles/community/annual-intel-open-source-community-survey-results.html.

O’Neil, Mathieu, Xiaolan Cai, Laure Muselli, Fred Pailler, and Stefano Zacchiroli. 2021. *The Coproduction of Open Source Software by Volunteers and Big Tech Firms*. Bruce, Australian Capital Territory: News & Media Research Centre.

Open Collective. 2024. “Open Collective Inc - Investor Update 2023.” Open Collective. February 7, 2024. https://blog.opencollective.com/open-collective-inc-investor-update/.

OSI. 2006. “The Open Source Definition.” Open Source Initiative. July 7, 2006. https://opensource.org/osd.

Sayegh, Emil. 2024. “The Great Tech Reset: Unpacking The Layoff Surge Of 2024.” Forbes. August 19, 2024. https://www.forbes.com/sites/emilsayegh/2024/08/19/the-great-tech-reset-unpacking-the-layoff-surge-of-2024/.

Soerjanto Tjahjono. 2018. “PRELIMINARY Aircraft Accident Investigation Report.” Aircraft Accident Investigation Report KNKT.18.10.35.04. Jakarta: Komite Nasional Keselamatan Transportasi. https://tailstrike.com/media/5abb0zcw/291018.pdf.

Sterman, John D., and James Quinn. 2023. “Boeing’s 737 MAX 8 Disasters | MIT Sloan.” MIT Sloan School. https://mitsloan.mit.edu/teaching-resources-library/boeings-737-max-8-disasters.

Szulik, Keenan. n.d. “Open Source Is Everywhere: Survey Results Part 1.” Accessed March 6, 2025. https://blog.tidelift.com/open-source-is-everywhere-survey-results-part-1.

“The Value of Open Source Software - Working Paper - Faculty & Research - Harvard Business School.” n.d. Accessed March 6, 2025. https://www.hbs.edu/faculty/Pages/item.aspx?num=65230.

Tidelift. n.d. “The 2024 Tidelift State of the Open Source Maintainer Report.” Tidelift. Accessed January 3, 2025. https://explore.tidelift.com/2024-tidelift-survey.

Travis, Gregory. 2019. “How the Boeing 737 Max Disaster Looks to a Software Developer - IEEE Spectrum.” IEEE Spectrum. April 18, 2019. https://spectrum.ieee.org/how-the-boeing-737-max-disaster-looks-to-a-software-developer.

Wang, Celine. 2023. “The Impact of Corporate Layoffs on Firm Performance in the US: Evidence from the Tech Industry.” Erasmus University Rotterdam.