O'REILLY®

Technology Trends for 2024

What O'Reilly Learning Platform Usage Tells Us About Where the Industry Is Headed

Mike Loukides

REPORT

Technology Trends for 2024

What O'Reilly Learning Platform Usage Tells Us About Where the Industry Is Headed

Mike Loukides

Beijing · Boston · Farnham · Sebastopol · Tokyo O'REILLY®

Technology Trends for 2024
by Mike Loukides

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Mike Loukides
Copyeditor: Peyton Joyce
Interior Designer: David Futato
Cover Designer: Ellie Volckhausen
Illustrator: Ellie Volckhausen

January 2024: First Edition

Revision History for the First Edition
2024-01-23: First Release

978-1-098-16795-0

[LSI]

Table of Contents

Technology Trends for 2024

This has been a strange year. While we like to talk about how fast technology moves, internet time, and all that, in reality the last major new idea in software architecture was microservices, which dates to roughly 2015. Before that, cloud computing itself took off in roughly 2010 (AWS was founded in 2006); and Agile goes back to 2000 (the *Agile Manifesto* dates back to 2001, Extreme Programming to 1999). The web is over 30 years old; the Netscape browser appeared in 1994, and it wasn't the first. We think the industry has been in constant upheaval, but there have been relatively few disruptions: one every five years, if that.

2023 was one of those rare disruptive years. ChatGPT changed the industry, if not the world. We're skeptical about things like job displacement, at least in technology. But AI is going to bring changes to almost every aspect of the software industry. What will those changes be? We don't know yet; we're still at the beginning of the story. In this report about how people are using O'Reilly's learning platform, we'll see how patterns are beginning to shift.

Just a few notes on methodology: This report is based on O'Reilly's internal "Units Viewed" metric. Units Viewed measures the actual usage of content on our platform. The data used in this report covers January through November in 2022 and 2023. Each graph is scaled so that the topic with the greatest usage is 1. Therefore, the graphs can't be compared directly to each other.

Remember that these "units" are "viewed" by our users, who are largely professional software developers and programmers. They aren't necessarily following the latest trends. They're solving real-world problems for their employers. And they're picking up the skills they need to advance in their current positions or to get new ones. We don't want to discount those who use our platform to get up to speed on the latest hot technology: that's how the industry moves forward. But to understand usage patterns, it's important to realize that every company has its own technology stacks, and that those stacks change slowly. Companies aren't going to throw out 20 years' investment in PHP so they can adopt the latest popular React framework, which will probably be displaced by another popular framework next year.

Software Development

Most of the topics that fall under software development declined in 2023. What does this mean? Programmers are still writing software; our lives are increasingly mediated by software, and that isn't going to change.

Software developers are responsible for designing and building bigger and more complex projects than ever. That's one trend that won't change: complexity is always "up and to the right." Generative AI is the wild card: Will it help developers to manage complexity? Or will it add complexity all its own? It's tempting to look at AI as a quick fix. Who wants to learn about coding practices when you're letting GitHub Copilot write your code for you? Who wants to learn about design patterns or software architecture when some AI application may eventually do your high-level design? AI is writing low-level code now; as many as 92% of software developers are using it. Whether it will be able to do high-level design is an open question—but as always, that question has two sides: "Will AI do our design work?" is less interesting than "How will AI change the things we want to design?" And the real question that will change our industry is "How do we design systems in which generative AI and humans collaborate effectively?"

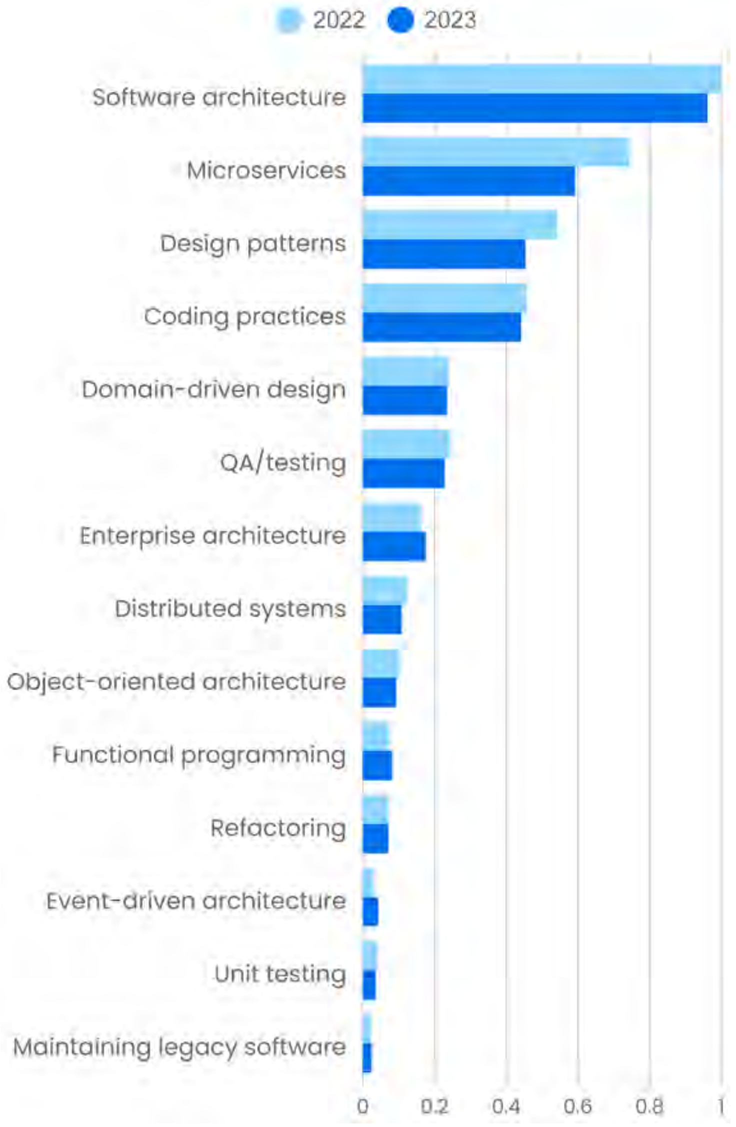

Figure 1. Software architecture

Regardless of the answers to these questions, humans will need to understand and specify what needs to be designed. Our data shows that most topics in software architecture and design are down year-over-year. But there are exceptions. While software architecture is down 3.9% (a relatively small decline), enterprise architecture is up

8.9%. Domain-driven design is particularly useful for understanding the behavior of complex enterprise systems; it's down, but only 2.0%. Use of content about event-driven architecture is relatively small, but it's up 40%. That change is important because event-driven architecture is a tool for designing large systems that have to ingest data from many different streams in real time. Functional programming, which many developers see as a design paradigm that will help solve the problems of distributed systems, is up 9.8%. So the software development world is changing. It's shifting toward distributed systems that manage large flows of data in real time. Use of content on topics relevant to that shift is holding its own or growing.

Microservices saw a 20% drop. Many developers expressed frustration with microservices during the year and argued for a return to monoliths. That accounts for the sharp decline—and it's fair to say that many organizations are paying the price for moving to microservices because it was "the thing to do," not because they needed the scale or flexibility that microservices can offer. From the start, microservice proponents have argued that the best way to develop microservices is to start with a monolith, then break the monolith into services as it becomes necessary. If implemented poorly, microservices deliver neither scale nor flexibility. Microservices aren't ideal for new greenfield projects, unless you're absolutely sure that you need them from the start—and even then, you should think twice. It's definitely not a technology to implement just to follow the latest fad.

Software developers run hot and cold on design patterns, which declined 16%. Why? It probably depends on the wind or the phase of the moon. Content usage about design patterns increased 13% from 2021 to 2022, so this year's decline just undoes last year's gain. It's possible that understanding patterns seems less important when AI is writing a lot of the code for you. It's also possible that design patterns seem less relevant when code is already largely written; most programmers maintain existing applications rather than develop new greenfield apps, and few texts about design patterns discuss the patterns that are embedded in legacy applications. But both ways of thinking miss the point. Design patterns are common solutions to common problems that have been observed in practice. Understanding design patterns keeps you from reinventing wheels. Frameworks like React and Spring are important because they implement design patterns. Legacy applications won't be improved by refactoring existing code just to use some pattern, but design patterns are useful for extending existing software and making it more flexible. And, of course, design patterns are used in legacy code—even code that was written before the term was coined! Patterns are discovered, not "invented"; again, they're common solutions to problems programmers have been solving since the beginning of programming.

At the same time, whenever there's a surge of interest in design patterns, there's a corresponding surge in pattern abuse: managers asking developers how many patterns they used (as if pattern count were a metric for good code), developers implementing FactoryFactoryFactory Factories, and the like. What goes around comes around, and the abuse of design patterns is part of a feedback loop that regulates the use of design patterns.

Programming and Programming Languages

Most of the programming languages we track showed declines in content usage. Before discussing specifics, though, we need to look at general trends. If 92% of programmers are using generative AI to write code and answer questions, then we'd certainly expect a drop in content use. That may or may not be advisable for career development, but it's a reality that businesses built on training and learning have to acknowledge. But that isn't the whole story either—and the bigger story leaves us with more questions than answers.

Rachel Stephens provides two fascinating pieces of the puzzle in a recent article on the RedMonk blog, but those pieces don't fit together exactly. First, she notes the decline in questions asked on Stack Overflow and states (reasonably) that asking a nonjudgmental AI assistant might be a preferable way for beginners to get their questions answered. We agree; we at O'Reilly have built O'Reilly Answers to provide that kind of assistance (and are in the process of a major upgrade that will make it even more useful). But Stack Overflow shows a broad peak in questions from 2014 to 2017, with a sharp decline afterward; the number of questions in 2023 is barely 50% of the peak, and the 20% decline from the January 2023 report to the July report is only somewhat sharper than the previous drops. And there was no generative AI, no ChatGPT, back in 2017 when the decline began. Did generative AI play a role? It would be foolish to say that it didn't, but it can't be the whole story.

Stephens points to another anomaly: GitHub pull requests declined roughly 25% from the second half of 2022 to the first half of 2023. Why? Stephens guesses that there was increased GitHub activity during the pandemic and that activity has returned to normal now that we've (incorrectly) decided the pandemic is over. Our own theory is that it's a reaction to GPT models leaking proprietary code and abusing open source licenses; that could cause programmers to be wary of public code repositories. But those are only guesses. This change is apparently not an error in the data. It might be a one-time anomaly, but no one really knows the cause. *Something* drove down programmer activity on GitHub, and that's inevitably a part of the background to this year's data.

So, what does O'Reilly's data say? As it has been for many years, Python is the most widely used programming language on our platform. This year, we didn't see an increase; we saw a very small (0.14%) decline. That's noise; we won't insult your intelligence by claiming that "flat in a down market" is really a gain. It's certainly fair to ask whether a language as popular as Python has gathered all the market share that it will get. When you're at the top of the adoption curve, it's difficult to go any higher and much easier to drop back. There are always new languages ready to take some of Python's market share. The most significant change in the Python ecosystem is Microsoft's integration of Python into Excel spreadsheets, but it's too early to expect that to have had an effect.

Use of content about Java declined 14%, a significant drop but not out of line with the drop in GitHub activity. Like Python, Java is a mature language and may have nowhere to go but down. It has never been "well loved"; when Java was first announced, people walked out of the doors of the conference room claiming that Java was dead before you could even download the beta. (I was there.) Is it time to dance on Java's grave? That dance has been going on since 1995, and it hasn't been right yet.

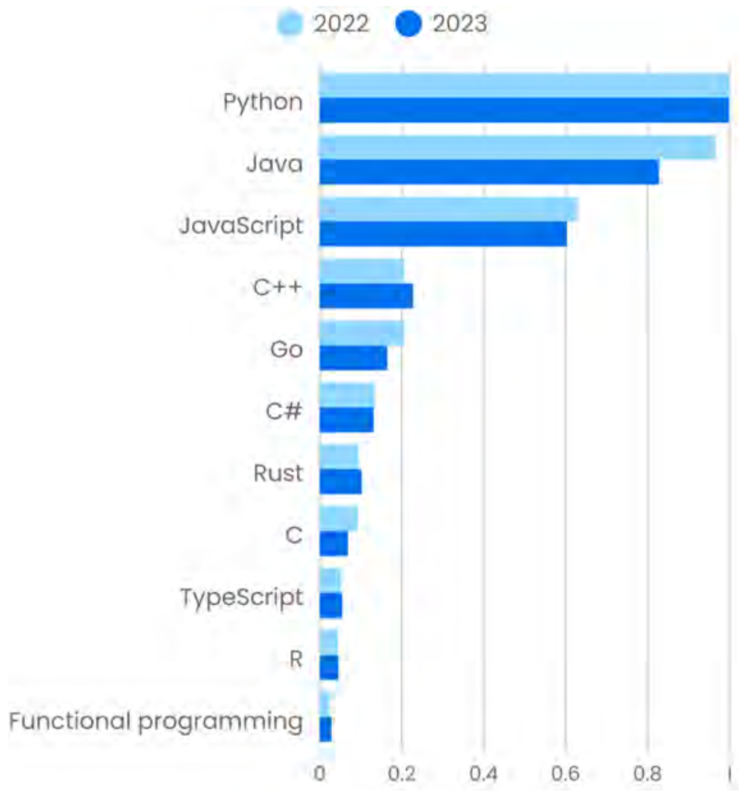

Figure 2. Programming languages

JavaScript also declined by 3.9%. It's a small decline and probably not meaningful. TypeScript, a version of JavaScript that adds static typing and type annotations, gained 5.6%. It's tempting to say that these cancel each other out, but that's not correct. Usage of TypeScript content is roughly one-tenth the usage of JavaScript content. But it is correct to say that interest in type systems is growing among web developers. It's also true that an increasing number of junior developers use JavaScript only through a framework like React or Vue. Boot camps and other crash programs often train students in "React," with little attention on the bigger picture. Developers trained in programs like these may be aware of JavaScript but may not think of themselves as JavaScript developers, and may not be looking to learn more about the language outside of a narrow, framework-defined context.

We see growth in C++ (10%), which is surprising for an old, well-established language. (C++ first appeared in 1985.) At this point in C++'s history, we'd expect it to be a headache for people maintaining legacy code, not a language for starting new projects. Why is it growing? While C++ has long been an important language for game development, there are signs that it's breaking out into other areas. C++ is an ideal language for embedded systems, which often require software that runs directly on the processor (for example, the software that runs in a smart lightbulb or in the braking system of any modern car). You aren't going to use Python, Java, or JavaScript for those applications. C++ is also an excellent language for number crunching (Python's numeric libraries are written in C++), which is increasingly important as artificial intelligence goes mainstream. It has also become the new "must have" language on résumés: knowing C++ proves that you're tough, that you're a "serious" programmer. Job anxiety exists—whether or not it's merited is a different question—and in an environment where programmers are nervous about keeping their current jobs or looking forward to finding a new one, knowing a difficult but widely used language can only be an asset.

Use of content about Rust also increased from 2022 to 2023 (7.8%). Rust is a relatively young language that stresses memory safety and performance. While Rust is considered difficult to learn, the idea that memory safety is baked in makes it an important alternative to languages like C++. Bugs in memory management are a significant source of vulnerabilities, as noted in NIST's page on "Safer Languages," and Rust does a good job of enforcing safe memory usage. It's now used in operating systems (Linux kernel components), tool development, and even enterprise software.

We also saw 9.8% growth in content about functional programming. We didn't see gains for any of the historical functional programming languages (Haskell, Erlang, Lisp, and Elixir) though; most saw steep declines. In the past decade, most programming languages have added functional features. Newer languages like Rust and Go have had them from the start. And Java has gradually added features like closures in a series of updates. Now programmers can be as functional as they want to be without switching to a new language.

Finally, there are some programming languages that we don't yet track but that we're watching with interest. Zig is a simple imperative language that's designed to be memory safe, like Rust, but relatively easy to learn. Mojo is a superset of Python that's compiled, not interpreted. It's designed for high performance, especially for numerical operations. Mojo's goal is to facilitate AI programming in a single language rather than a combination of Python and some other language (typically C++) that's used for performance-critical numerical code. Where are these languages going? It will be some years before they reach the level of Rust or Go, but they're off to a good start.

So what does all this tell us about training and skill development? It's easy to think that, with Copilot and other tools to answer all your questions, you don't need to put as much effort into learning new technologies. We all ask questions on Google or Stack Overflow, and now we have other places to get answers. Necessary as that is, the idea that asking questions can replace training is naive. Unlike many who are observing the influence of generative AI on programming, we believe that it will increase the gap between entry-level skills and senior developer skills. Being a senior developer—being a senior anything—requires a kind of fluency that you can't get just from asking questions. I may never be a fluent user of Python's pandas library (which I used extensively to write this report); I asked lots of questions, and that has undoubtedly saved me time. But what happens when I need to solve the next problem? The kind of fluency that you need to look at a problem and understand how to solve it doesn't come from asking simple "How do I do this?" questions. Nor does it preclude asking lots of "I forgot how this function works" questions. That's why we've built O'Reilly Answers, an AI-driven service that finds solutions to questions using content from our platform. But expertise does require developing the intellectual muscle that comes from grappling with problems and solving them yourself rather than letting something else solve them for you. (And that includes forcing yourself to remember all the messy syntax details.) People who think generative AI is a shortcut to expertise (and the job title and salary that expertise merits) are shortchanging themselves.

Artificial Intelligence

In AI, there's one story and only one story, and that's the GPT family of models. Usage of content on these models exploded 3,600% in the past year. That explosion is tied to the appearance of ChatGPT in November 2022. But don't make the mistake of thinking that ChatGPT came out of nowhere. GPT-3 created a big splash when it was released in 2020 (complete with a clumsy web-based interface). GPT-2 appeared in 2019, and the original unnumbered GPT was even earlier. The real innovation in ChatGPT wasn't the technology itself (though the models behind it represent a significant breakthrough in AI performance); it was packaging the model as a chatbot. That doesn't mean that the GPT explosion wasn't real. While our analysis of search trends shows that interest in ChatGPT has peaked among our platform's users, interest in natural language processing (NLP) showed a 195% increase—and from a much higher starting point.[1] That makes sense, given the more technical nature of our audience. Software developers will be building on top of the APIs for GPT and other language models and are likely less interested in ChatGPT, the web-based chat service. Related topics generative models (900%) and Transformers (325%) also showed huge gains. Prompt engineering, which didn't exist in 2022, became a significant topic, with roughly the same usage as Transformers. As far as total use, NLP is almost twice GPT. However you want to read the data, this is AI's big year, largely due to the GPT models and the idea of generative AI.

1 Google Trends suggests that we may be seeing a resurgence in ChatGPT searches. Meanwhile, searches for ChatGPT on our platform appear to have bottomed out in October, with a very slight increase in November. This discrepancy aligns well with the difference between our platform and Google's. If you want to use ChatGPT to write a term paper, are you going to search Google or O'Reilly?

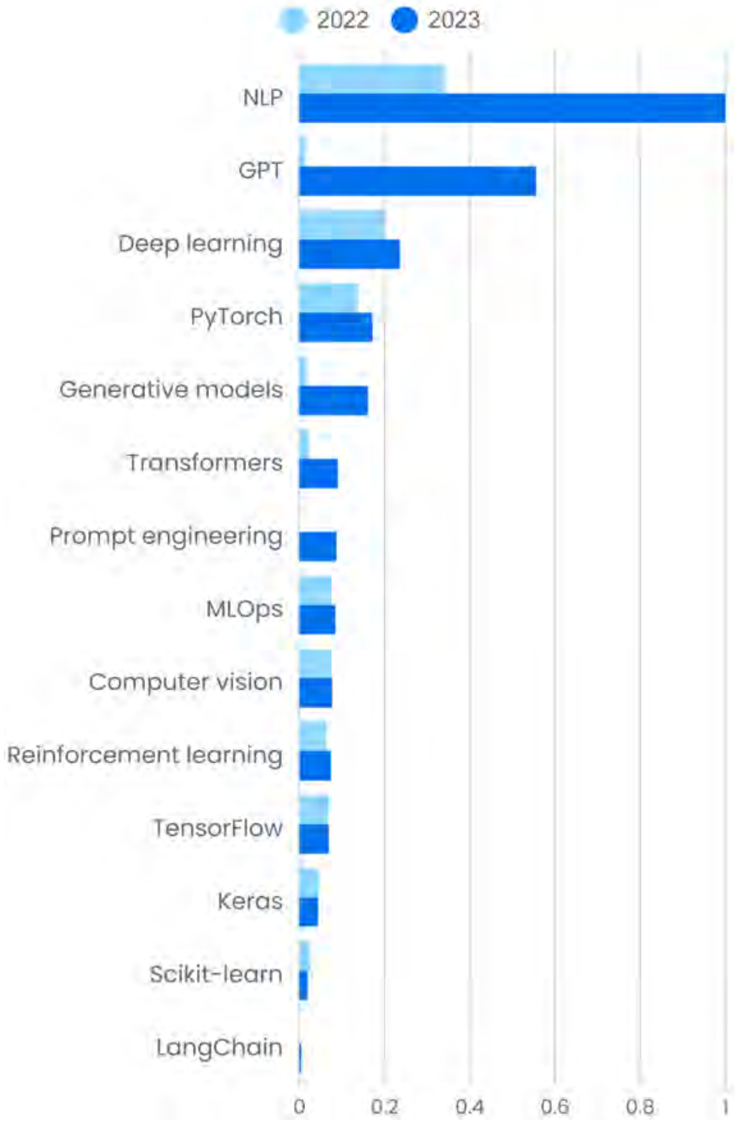

Figure 3. Artificial intelligence

But don't assume that the explosion of interest in generative AI meant that other aspects of AI were standing still. Deep learning, the creation and application of neural networks with many layers, is fundamental to every aspect of modern AI. Usage in deep learning content grew 19% in the past year. Reinforcement learning, in which models are trained by giving “rewards” for solving problems, grew 15%. Those gains only look small in comparison to the triple- and quadruple-digit gains we’re seeing in natural language processing. PyTorch, the Python library that has come to dominate programming in machine learning and AI, grew 25%. In recent years, interest in PyTorch has been growing at the expense of TensorFlow, but TensorFlow showed a small gain (1.4%), reversing (or at least pausing) its decline. Interest in two older libraries, scikit-learn and Keras, declined: 25% for scikit-learn and 4.8% for Keras. Keras has largely been subsumed by TensorFlow, while scikit-learn hasn’t yet incorporated the capabilities that would make it a good platform for building generative AI. (An attempt to implement Transformers in scikit-learn appears to be underway at Hugging Face.)

We’ve long said that operations is the elephant in the room for machine learning and artificial intelligence. Building models and developing applications is challenging and fun, but no technology can mature if IT teams can’t deploy, monitor, and manage it. Interest in operations for machine learning (MLOps) grew 14% over the past year. This is solid, substantial growth that only looks small in comparison with topics like generative AI. Again, we’re still in the early stages—generative AI and large language models are only starting to reach production. If anything, this increase probably reflects older applications of AI. There’s a growing ecosystem of startups building tools for deploying and monitoring language models, which are fundamentally different from traditional applications. As companies deploy the applications they’ve been building, MLOps will continue to see solid growth. (More on MLOps when we discuss operations below.)

LangChain is a framework for building generative AI applications around groups of models and databases. It’s often used to implement the retrieval-augmented generation (RAG) pattern, where a user’s prompt is used to look up relevant items in a vector database; those items are then combined with the prompt, generating a new prompt that is sent to the language model. There isn’t much content about LangChain available yet, and it didn’t exist in 2022, but it’s clearly going to become a foundational technology. Likewise, vector databases aren’t yet in our data. We expect that to change next year. They are rather specialized, so we expect usage to be relatively small, unlike products like MySQL—but they will be very important.

AI wasn't dominated entirely by the work of OpenAI; Meta's LLaMA and Llama 2 also attracted a lot of attention. The source code for LLaMA was open source, and its weights (parameters) were easily available to researchers. Those weights quickly leaked from "researchers" to the general public, where they jump-started the creation of smaller open source models. These models are much smaller than behemoths like GPT-4. Many of them can run on laptops, and they're proving ideal for smaller companies that don't want to rely on Microsoft, OpenAI, or Google to provide AI services. (If you want to run an open source language model on your laptop, try llamafile.) While huge "foundation models" like the GPT family won't disappear, in the long run open source models like Alpaca and Mistral may prove to be more important to software developers.

It's easy to think that generative AI is just about software development. It isn't; its influence extends to just about every field. Our ChatGPT: Possibilities and Pitfalls Superstream was the most widely attended event we've ever run. There were over 28,000 registrations, with attendees and sponsors from industries as diverse as pharmaceuticals, logistics, and manufacturing. Attendees included small business owners, sales and marketing personnel, and C-suite executives, along with many programmers and engineers from different disciplines. We've also been running courses focused on specific industries: Generative AI for Finance had over 2,000 registrations, and Generative AI for Government over 1,000. And more than 1,000 people signed up for our Generative AI for Healthcare event.

Data

In previous years, we would have told the story of AI as part of the story of data. That's still correct; with its heavy emphasis on mathematics and statistics, AI is a natural outgrowth of data science. But this year, AI has become the superstar that gets top billing, while data is a supporting actor.

That doesn't mean that data is unimportant. Far from it. Every company uses data: for planning, for making projections, for analyzing what's happening within the business and the markets they serve. So it's not surprising that the second biggest topic in data is Microsoft Power BI, with a 36% increase since 2022. SQL Server also showed a 5.3% increase, and statistics toolbox R increased by 4.8%.

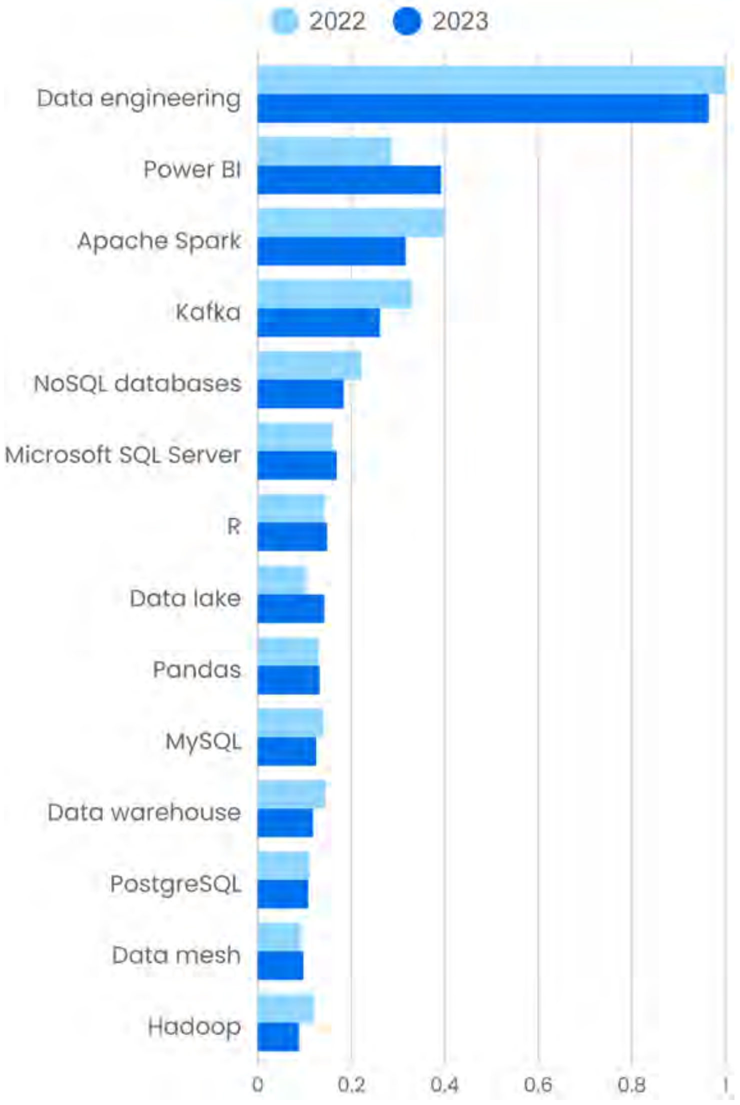

Figure 4. Data analysis and databases

Data engineering was by far the most heavily used topic in this category; it showed a 3.6% decline, stabilizing after a huge gain from 2021 to 2022. Data engineering deals with the problem of storing data at scale and delivering that data to applications. It includes moving data to the cloud, building pipelines for acquiring data and getting data to application software (often in near real time), resolving the issues that are caused by data siloed in different organizations, and more. Two of the most important platforms for data engineering, Kafka and Spark, showed significant declines in 2023 (21% and 20%, respectively). Kafka and Spark have been workhorses for many years, but they are starting to show their age as they become "legacy technology." (Hadoop, down 26%, is clearly legacy software in 2023.) Interest in Kafka is likely to rise as AI teams start implementing real-time models that have up-to-the-minute knowledge of external data. But we also have to point out that there are newer streaming platforms (like Pulsar) and newer data platforms (like Ray).

Designing enterprise-scale data storage systems is a core part of data engineering. Interest in data warehouses saw an 18% drop from 2022 to 2023. That's not surprising; data warehouses also qualify as legacy technology. Two other patterns for enterprise-scale storage show significant increases: Usage of content about data lakes is up 37% and, in absolute terms, significantly higher than that of data warehouses. Usage for data mesh content is up 5.6%. Both lakes and meshes solve a basic problem: How do you store data so that it's easy to access across an organization without building silos that are only relevant to specific groups? Data lakes can include data in many different formats, and it's up to users to supply structure when data is utilized. A data mesh is a truly distributed solution: each group is responsible for its own data but makes that data available throughout the enterprise through an interoperability layer. Those newer technologies are where we see growth.

The two open source data analysis platforms were virtually unchanged in 2023. Usage of content about R increased by 3.6%; we've already seen that Python was unchanged, and pandas grew by 1.4%. Neither of these is going anywhere, but alternatives, particularly to pandas, are appearing.

Operations

Whether you call it operations, DevOps, or something else, this field has seen some important changes in the past year. We've witnessed the rise of developer platforms, along with the related topic, platform engineering. Both of those are too new to be reflected in our data: you can't report content use before content exists. But they are influencing other topics.

We've said in the past that Linux is table stakes for a job in IT. That's still true. But the more the deployment process is automated—and platform engineering is just the next step in "Automate All the Things"—the less developers and IT staff need to know about Linux. Software is packaged in containers, and the containers themselves run as virtual Linux instances, but developers don't need to know how to find and kill out-of-control processes, do a backup, install device drivers, or perform any of the other tasks that are the core of system administration. Usage of content about Linux is down 6.9%: not a major change but possibly a reflection of the fact that the latest steps forward in deploying and managing software shield people from direct contact with the operating system.

Similar trends reduce what developers and IT staff need to know about Kubernetes, the near-ubiquitous container orchestrator (down 6.9%). Anyone who uses Kubernetes knows that it's complex. We've long expected "something simpler" to come along and replace it. It hasn't—but again, developer platforms put users a step further away from engaging with Kubernetes itself. Knowledge of the details is encapsulated either in a developer platform or, perhaps more often, in a Kubernetes service administered by a cloud provider. Kubernetes can't be ignored, but it's more important to understand high-level principles than low-level commands.

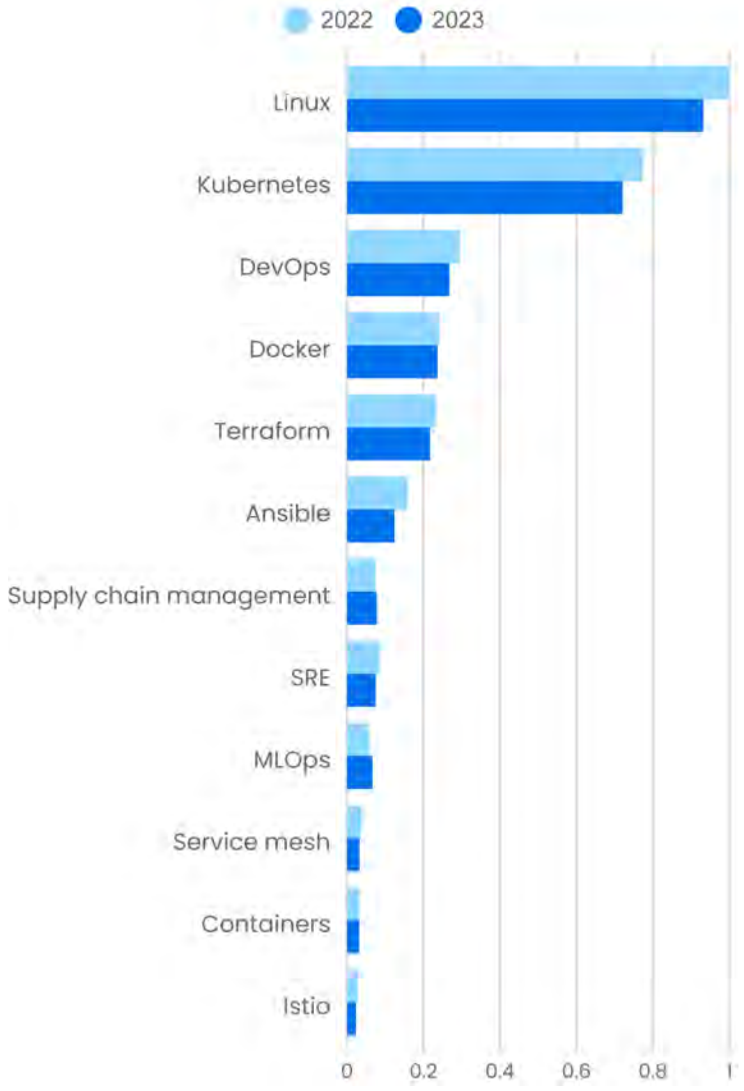

Figure 5. Infrastructure and operations

DevOps (9.0%) and SRE (13%) are also down, though we don't think that's significant. Terms come and go, and these are going. While operations is constantly evolving, we don't believe we'll ever get to the mythical state of "NoOps," nor should we. Instead, we'll see constant evolution as the ratio of systems managed to operations staff grows ever higher. But we *do* believe that sooner rather than later, someone will put a new name on the disciplines of DevOps and its close relative, SRE. That new name might be "platform engineering," though that term says more about designing deployment pipelines than about carrying the pager and keeping the systems running; platform engineering is about treating developers as customers and designing internal developer platforms that make it easy to test and deploy software systems with minimal ceremony. We don't believe that platform engineering subsumes or replaces DevOps. Both are partners in improving experience for developers and operations staff (and ratcheting up the ratio of systems managed to staff even higher).

That's a lot of red ink. What's in the black? Supply chain management is up 5.9%. That's not a huge increase, but in the past few years we've been forced to think about how we manage the software supply chain. Any significant application easily has dozens of dependencies, and each of those dependencies has its own dependencies. The total number of dependencies, including both direct and inherited dependencies, can easily be hundreds or even thousands. Malicious operators have discovered that they can corrupt software archives, getting programmers to inadvertently incorporate malware into their software. Unfortunately, security problems never really go away; we expect software supply chain security to remain an important issue for the foreseeable (and unforeseeable) future.

We've already mentioned that MLOps, the discipline of deploying and managing models for machine learning and artificial intelligence, is up 14%. Machine learning and AI represent a new kind of software that doesn't follow traditional rules, so traditional approaches to operations don't work. The list of differences is long:

- While most approaches to deployment are based on the idea that an application can be reproduced from a source archive, that isn't true for AI. An AI system depends as much on the training data as it does on the source code, and we don't yet have good tools for archiving training data.
- While we've said that open source models such as Alpaca are much smaller than models like GPT-4 or Google's Gemini, even the smallest of those models is very large by any reasonable standard.

- While we've gotten used to automated testing as part of a deployment pipeline, AI models aren't deterministic. A test doesn't necessarily give the same result every time it runs. Testing is no less important for AI than it is for traditional software (arguably it's more important), and we're starting to see startups built around AI testing, but we're still at the beginning.

That's just a start. MLOps is a badly needed specialty. It's good to see growing interest.

Security

Almost all branches of security showed growth from 2022 to 2023. That's a welcome change: in the recent past, many companies talked about security but never made the investment needed to secure their systems. That's changing, for reasons that are obvious to anyone who reads the news. Nobody wants to be a victim of data theft or ransomware, particularly now that ransomware has evolved into blackmail.

The challenges are really very simple. Network security, keeping intruders off of your network, was the most widely used topic and grew 5%. Firewalls, which are an important component of network security, grew 16%. Hardening, a much smaller topic that addresses making systems less vulnerable to attack, grew 110%. Penetration testing remained one of the most widely used topics. Usage dropped 5%, although a 10% increase for Kali Linux (an important tool for penetration testers) largely offsets that decline.

The 22% growth in security governance is another indicator of changed attitudes: security is no longer an ad hoc exercise that waits for something to happen and then fights fires. Security requires planning, training, testing, and auditing to ensure that policies are effective.

One key to security is knowing who your users are and which parts of the system each user can access. Identity and access management (IAM) has often been identified as a weakness, particularly for cloud security. As systems grow more complex, and as our concept of "identity" evolves from individuals to roles assigned to software services, IAM becomes much more than usernames and passwords. It requires a thorough understanding of who the actors are on your systems and what they're allowed to do. This extends the old idea of "least privilege": each actor needs the ability to do exactly what they need, no more and no less. The use of content about IAM grew 8.0% in the past year. It's a smaller gain than we would have liked to see but not insignificant.

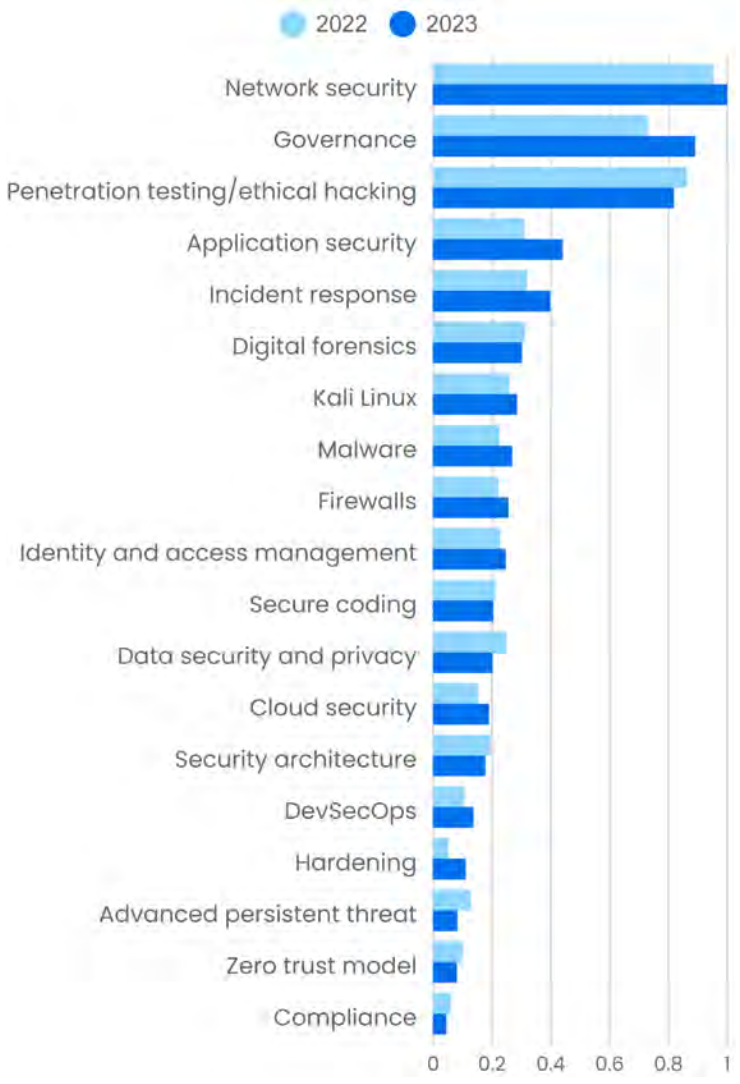

Figure 6. Security

Application security grew 42%, showing that software developers and operations staff are getting the message. The DevSecOps "shift left" movement, which focuses on software security early in the development process, appears to be winning; use of content about DevSecOps was up 30%. Similarly, those who deploy and maintain applications have become even more aware of their responsibilities. Developers may design identity and access management into the code, but operations is responsible for configuring these correctly and ensuring that access to applications is only granted appropriately. Security can't be added after the fact; it has to be part of the software process from beginning to the end.

Advanced persistent threats (APTs) were all over the news a few years ago. We don't see the term APT anywhere near as much as we used to, so we're not surprised that usage has dropped by 35%. Nevertheless, nation-states with sophisticated offensive capabilities are very real, and cyber warfare is an important component of several international conflicts, including the war in Ukraine.

It's disappointing to see that usage of content about zero trust has declined by 20%. That decrease is more than offset by the increase in IAM, which is an essential tool for zero trust. But don't forget that IAM is just a tool and that the goal is to build systems that don't rely on trust, that always verify that every actor is appropriately identified and authorized. How can you defend your IT infrastructure if you assume that attackers already have access? That's the question zero trust answers. Trust nothing; verify everything.

Finally, compliance is down 27%. That's more than offset by the substantial increase of interest in governance. Auditing for compliance is certainly a part of governance. Focusing on compliance itself, without taking into account the larger picture, is a problem rather than a solution. We've seen many companies that focus on compliance with existing standards and regulations while avoiding the hard work of analyzing risk and developing effective policies for security. "It isn't our fault that something bad happened; we followed all the rules" is, at best, a poor way to explain systemic failure. If that compliance-oriented mindset is fading, good riddance. Compliance, understood properly, is an important component of IT governance. Understood badly, compliance is an unacceptable excuse.

Finally, a word about a topic that doesn't yet appear in our data. There has, of course, been a lot of chatter about the use of AI in security applications. AI will be a great asset for log file analysis, intrusion detection, incident response, digital forensics, and other aspects of cybersecurity. But, as we've already said, there are always two sides to AI. How does AI change security itself? Any organization with AI applications will have to protect them from exploitation. What vulnerabilities does AI introduce that didn't exist a few years ago? There are many articles about prompt injection, sneaky prompts designed to "jailbreak" AI systems, data leakage, and other vulnerabilities—and we believe that's only the beginning. Securing AI systems will be a critical topic in the coming years.

Cloud Computing

Looking at platform usage for cloud-related topics, one thing stands out: cloud native. Not only is it the most widely used topic in 2023, but it grew 175% from 2022 to 2023. This marks a real transition. In the past, companies built software to run on-premises and then moved it to the cloud as necessary. Despite reports (including ours) that showed 90% or more "cloud adoption," we always felt that was very optimistic. Sure, 90% of all companies may have one or two experiments *in* the cloud—but are they really building *for* the cloud? This huge surge in cloud native development shows that we've now crossed that chasm and that companies have stopped kicking the tires. They're building for the cloud as their primary deployment platform.

You could, of course, draw the opposite conclusion by looking at cloud deployment, which is down 27%. If companies are developing for the cloud, how are those applications being deployed? That's a fair question. However, as cloud usage grows, so does organizational knowledge of cloud-related topics, particularly deployment. Once an IT group has deployed its first application, the second isn't necessarily "easy" or "the same," but it is familiar. At this point in the history of cloud computing, we're seeing few complete newcomers. Instead we're seeing existing cloud users deploying more and more applications. We're also seeing a rise in tools that streamline cloud deployment. Indeed, any provider worth thinking about has a tremendous interest in making deployment as simple as possible.

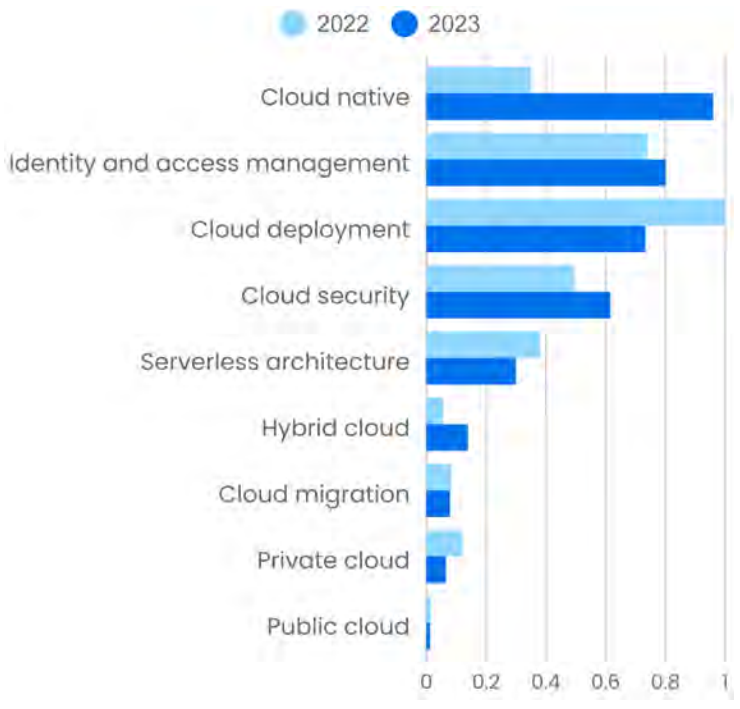

Figure 7. Cloud architecture

Use of content about cloud security grew 25%, and identity and access management (IAM) grew 8%. An epidemic of data theft and ransomware that continues to this day put security on the corporate map as a priority, not just an expense with annual budget requests that sounded like an extortion scam: "Nothing bad happened this year; give us more money and maybe nothing bad will happen next year." And while the foundation of any security policy is good local security hygiene, it's also true that the cloud presents its own issues. Identity and access management: locally, that means passwords, key cards, and (probably) two-factor authentication. In the cloud, that means IAM, along with zero trust. Same idea, but it would be irresponsible to think that these aren't more difficult in the cloud.

Hybrid cloud is a smaller topic area that has grown significantly in the past year (145%). This growth points partly to the cloud becoming the de facto deployment platform for enterprise applications. It also acknowledges the reality of how cloud computing is adopted. Years ago, when "the cloud" was getting started, it was easy for a few developers in R&D to expense a few hours of time on AWS rather than requisitioning new hardware. The same was true for data-aware marketers who wanted to analyze what was happening with their potential customers—and they might choose Azure. When senior management finally awoke to the need for a "cloud strategy," they were already in a hybrid situation, with multiple wildcat projects in multiple clouds. Mergers and buyouts complicated the situation more. If company A is primarily using AWS and company B has invested heavily in Google Cloud, what happens when they merge? Unifying behind a single cloud provider isn't going to be worth it, even though cloud providers are providing tools to simplify migration (at the same time as they make their own clouds difficult to leave). The cloud is naturally hybrid. "Private cloud" and "public cloud," when positioned as alternatives to each other and to a hybrid cloud, smell like "last year's news." It's not surprising that usage has dropped 46% and 10%, respectively.

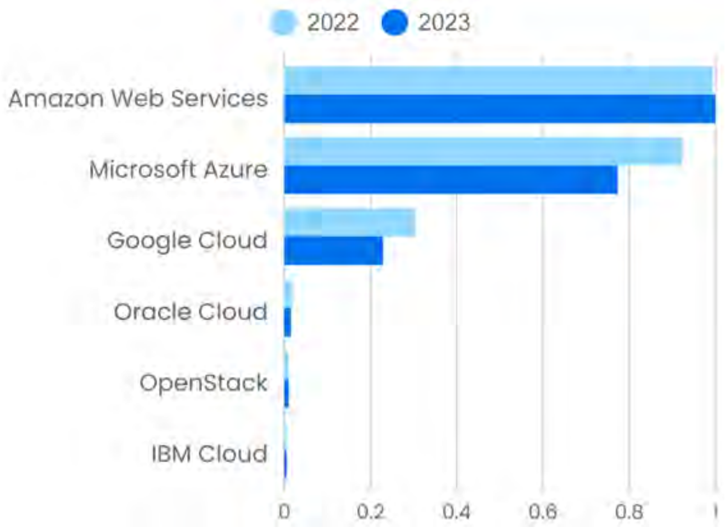

Figure 8. Cloud providers

What about the perennial horse race between Amazon Web Services, Microsoft Azure, and Google Cloud? Is anyone still interested, except perhaps investors and analysts? AWS showed a very, very small gain (0.65%), but Azure and Google Cloud showed significant losses (16% and 22%, respectively). We expected to see Azure catch up to AWS because of its lead in AI as a service, but it didn't. As far as our platform is concerned, that's still in the future.

Web Development

React and Angular continue to dominate web development. JavaScript is still the lingua franca of web development, and that isn't likely to change any time soon.

But the usage pattern has changed slightly. Last year, React was up, and Angular was sharply down. This year, usage of React content hasn't changed substantially (down 0.33%). Angular is down 12%, a smaller decline than last year but still significant. When a platform is as dominant as React, it may have nowhere to go but down. Is momentum shifting?

We see some interesting changes among the less popular frameworks, both old and new. First, Vue isn't a large part of the overall picture, and it isn't new—it's been around since 2014—but if its 28% annual growth continues, it will soon become a dominant framework. That increase represents a solid turnaround after losing 17% from 2021 to 2022. Django is even older (created in 2005), but it's still widely used—and with an 8% increase this year, it's not going away. FastAPI is the newest of this group (2018). Even though it accounts for a very small percentage of platform use, it's easy for a small change in usage to have a big effect. An 80% increase is hard to ignore.

It's worth looking at these frameworks in a little more detail. Django and FastAPI are both Python-based, and FastAPI takes full advantage of Python's type hinting feature. Python has long been an also-ran in web development, which has been dominated by JavaScript, React, and Angular. Could that be changing? It's hard to say, and it's worth noting that Flask, another Python framework, showed a 12% decrease. As a whole, Python frameworks probably declined from 2022 to 2023, but that may not be the end of the story. Given the number of boot camps training new web programmers in React, the JavaScript hegemony will be hard to overcome.

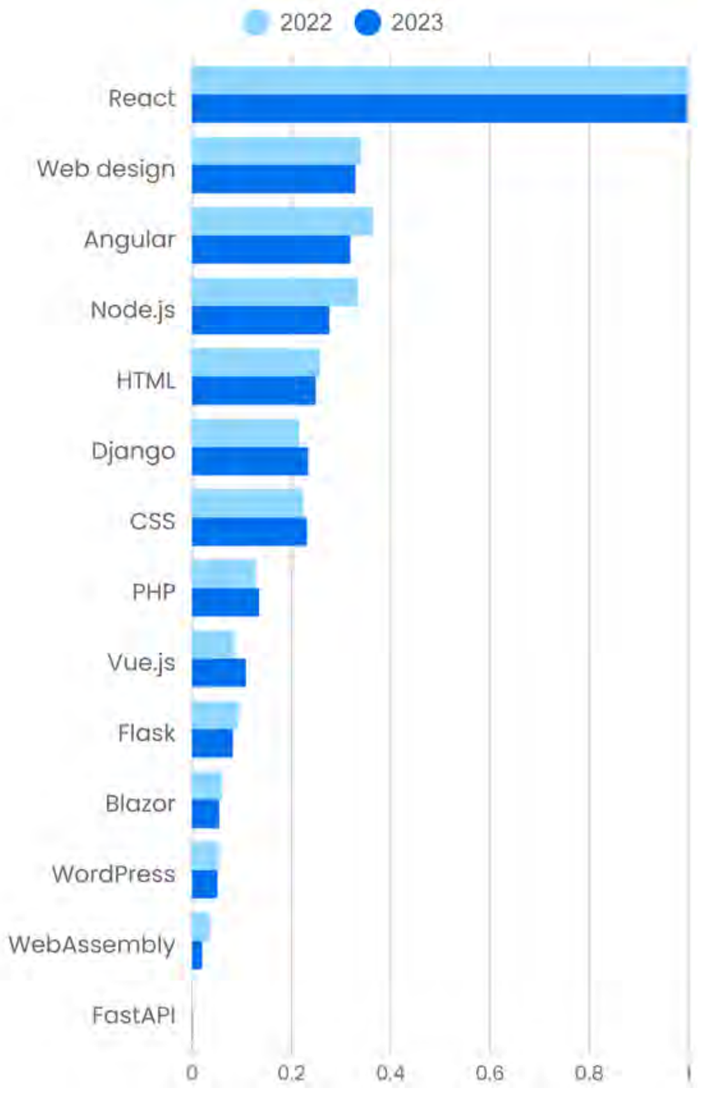

Figure 9. Web development

What about PHP, another long-standing framework that dates back to 1995, when the web was indeed young? PHP grew 5.9% in the past year. The use of content about PHP is small compared to frameworks like React and Angular or even Django. PHP certainly doesn't inspire the excitement that it did in the 1990s. But remember that over 80% of the web is built on PHP. It's certainly not trendy, it's not capable of building the feature-rich sites that many users expect—but it's everywhere. WordPress (down 4.8%), a content management system used for millions of websites, is based on PHP. But regardless of the number of sites that are built on PHP or WordPress, Indeed shows roughly three times as many job openings for React developers as for PHP and WordPress combined. PHP certainly isn't going away, and it may even be growing slightly. But we suspect that PHP programmers spend most of their time maintaining older sites. They already know what they need to do that, and neither of those factors drives content usage.

What about some other highly buzzworthy technologies? After showing 74% growth from 2021 to 2022, WebAssembly (Wasm) declined by 41% in 2023. Blazor, a web framework for C# that generates code for Wasm, declined by 11%. Does that mean that Wasm is dying? We still believe Wasm is a very important technology, and we frequently read about amazing projects that are built with it. It isn't yet a mature technology—and there are plenty of developers willing to argue that there's no need for it. We may disagree, but that misses the point. Usage of Wasm content will probably decline gradually...until someone creates a killer application with it. Will that happen? Probably, but we can't guess when.

What does this mean for someone who's trying to develop their skills as a web developer? First, you still can't go wrong with React, or even with Angular. The other JavaScript frameworks, such as Next.js, are also good options. Many of these are metaframeworks built on React, so knowing them makes you more versatile while leveraging knowledge you already have. If you're looking to broaden your skills, Django would be a worthwhile addition. It's a very capable framework, and knowing Python will open up other possibilities in software development that may be helpful in the future, even if not now.

Certification

This year, we took a different approach to certification. Rather than discussing certification for different subject areas separately (that is, cloud certification, security certification, etc.), we used data from the platform to build a list of the top 20 certifications and grouped them together. That process gives a slightly different picture of which certifications are important and why. We also took a brief look at

O'Reilly's new badges program, which gives another perspective on what our customers want to learn.

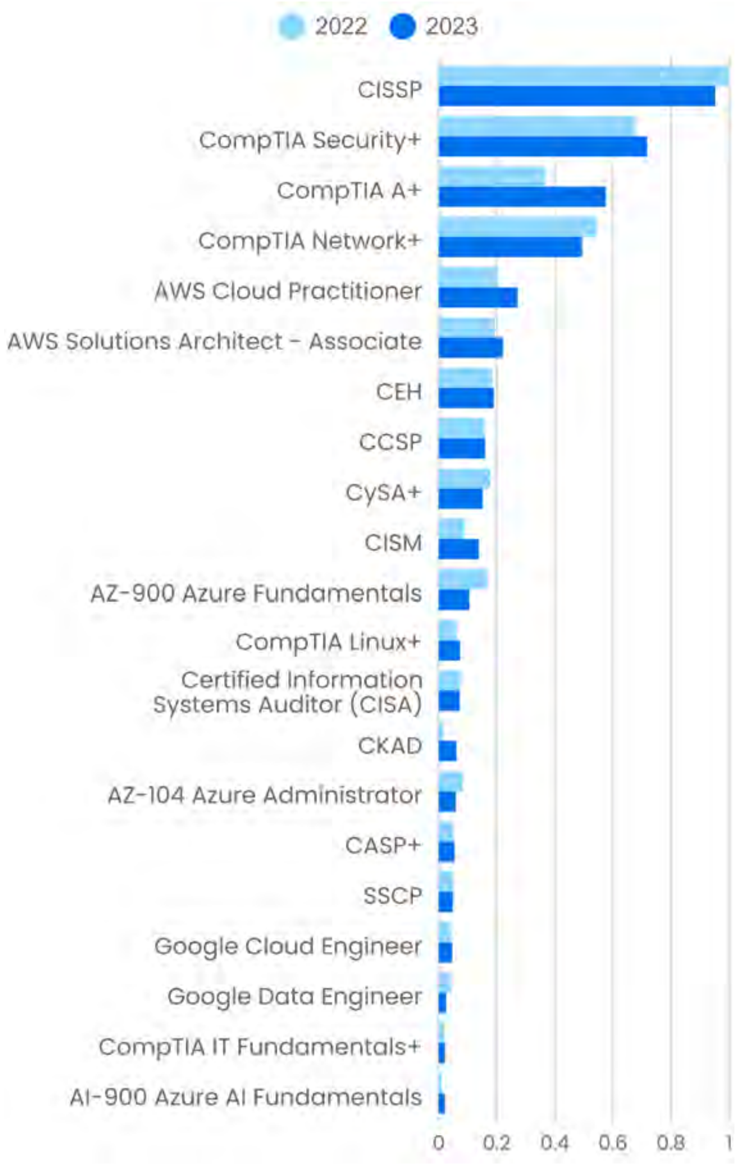

Figure 10. Certification

Based on the usage of content in our platform (including practice tests), the most popular certifications are security certifications: CISSP (which declined 4.8%) and CompTIA Security+ (which grew 6.0%). CISSP is an in-depth exam for security professionals, requiring at least five years' experience before taking the exam. Security+ is more of an entry-level exam, and its growth shows that security staff are still in demand. ISACA's Certified Information Security Manager (CISM) exam, which focuses on risk assessment, governance, and incident response, isn't as popular but showed a 54% increase. CompTIA's Certified Advanced Security Practitioner (CASP+) showed a 10% increase—not as large but part of the same trend. The Certified Ethical Hacker (CEH) exam, which focuses on techniques useful for penetration testing or red-teaming, is up 4.1%, after a decline last year. Those increases reflect where management is investing. Hoping that there won't be an incident has been replaced by understanding exposure, putting in place governance mechanisms to minimize risk, and being able to respond to incidents when they occur.

What really stands out, however, isn't security: it's the increased use of content about CompTIA A+, which is up 58%. A+ isn't a security exam; it's advertised as an entry-level exam for IT support, stressing topics like operating systems, managing SaaS for remote work, troubleshooting software, hardware, and networking problems, and the like. It's testimony to the large number of people who want to get into IT. Usage of content about the CompTIA Linux+ exam was much lower but also grew sharply (23%)—and, as we've said in the past, Linux is "table stakes" for almost any job in computing. It's more likely that you'll encounter Linux indirectly via containers or cloud providers rather than managing racks of computers running Linux; but you will be expected to know it. The Certified Kubernetes Administrator (CKAD) exam also showed significant growth (32%). Since it was first released in 2014, Kubernetes has become an inescapable part of IT operations. The biggest trend in IT, going back 70 years or so, has been the increase in the ratio of machines to operators: from multiple operators per machine in the '60s to one operator per machine in the era of minicomputers to dozens and now, in the cloud, to hundreds and thousands. Complex as Kubernetes is—and we admit, we keep looking for a simpler alternative—it's what lets IT groups manage large applications that are implemented as dozens of microservices and that run in thousands of containers on an uncountable number of virtual machines.

Kubernetes has become an essential skill for IT. And certification is becoming increasingly attractive to people working in the field; there's no other area in which we see so much growth.

Cloud certifications also show prominently. Although "the cloud" has been around for almost 20 years, and almost every company will say that they are "in the cloud," in reality many companies are still making that transition. Furthermore, cloud providers are constantly adding new services; it's a field where keeping up with change is difficult. Content about Amazon Web Services was most widely used. AWS Cloud Practitioner increased by 35%, followed by AWS Solutions Architect (Associate), which increased 15%. Microsoft Azure certification content followed, though the two most prominent exams showed a decline: Azure Fundamentals (AZ-900) was down 37%, and Azure Administration (AZ-104) was down 28%. Google Cloud certifications trailed the rest: Google's Cloud Engineer showed solid growth (14%), while its Data Engineer showed a significant decline (40%).

Content about Microsoft's AI-900 exam (Azure AI Fundamentals) was the least-used among the certifications that we tracked. However, it gained 121%—it more than doubled—from 2022 to 2023. While we can't predict next year, this is the sort of change that trends are made of. Why did this exam suddenly get so hot? It's easy, really: Microsoft's investment in OpenAI, its integration of the GPT models into Bing and other products, and its AI-as-a-service offerings through Azure have suddenly made the company a leader in cloud-based AI. While we normally hedge our bets on smaller topics with big annual growth—it's easy for a single new course or book to cause a large swing—AI isn't going away, nor is Microsoft's leadership in cloud services for AI developers.

Late in 2023, O'Reilly began to offer badges tied to course completion on the O'Reilly learning platform. Badges aren't certifications, but looking at the top badges gives another take on what our customers are interested in learning. The results aren't surprising: Python, GPT (not just ChatGPT), Kubernetes, software architecture, and Java are the most popular badges.

However, it's interesting to look at the difference between our B2C customers (customers who have bought platform subscriptions as individuals) and B2B customers (who use the platform via a corporate subscription). For most topics, including those listed above, the ratio of B2B to B2C customers is in the range of 2:1 or 3:1 (two or three times as many corporate customers as individuals). The outliers are for topics like communications skills, Agile, Scrum, personal productivity, Excel, and presentation skills: users from B2B accounts obtained these badges four (or more) times as often as users with personal accounts. This makes sense: these topics are about teamwork and other skills that are valuable in a corporate environment.

There are few (if any) badge topics for which individual (B2C) users outnumbered corporate customers; that's just a reflection of our customer base. However, there were some topics where the ratio of B2B to B2C customers was closer to one. The most interesting of these concerned artificial intelligence: large language models (LLMs), TensorFlow, natural language processing, LangChain, and MLOps. Why is there more interest among individuals than among corporate customers? Perhaps by next year we'll know.

Design

The important story in design is about tools. Topics like user experience and web design are stable or slightly down (down 0.62% and 3.5%, respectively). But usage about design tools is up 105%, and the VC unicorn Figma is up 145%. Triple-digit growth probably won't continue, but it's certainly worth noticing. It highlights two important trends that go beyond typical design topics, like UX.

First, low-code and no-code tools aren't new, but many new ones have appeared in the past year. Their success has been aided by artificial intelligence. We already have AI tools that can generate text, whether for a production site or for a mockup. Soon we'll have no-code tools that don't just spit out a wireframe but will be able to implement the design itself. They will be smart about what the user wants them to do. But to understand the importance of low-code to design, you have to look beyond the use designers will make of these tools. Designers will also be designing these tools, along with other AI-powered applications. Tools for designers have to be well-designed, of course: that's trivial. But what many discussions about AI ignore is that designing applications that use AI well is far from trivial. We've all been blindsided by the success of ChatGPT, which made the GPT models instantly accessible to everyone. But once you start thinking about the possibilities, you realize that a chat is hardly an ideal interface for an AI system.[2] What will the users of these systems really need? We've only just started down that path. It will be an exciting journey—particularly for designers.

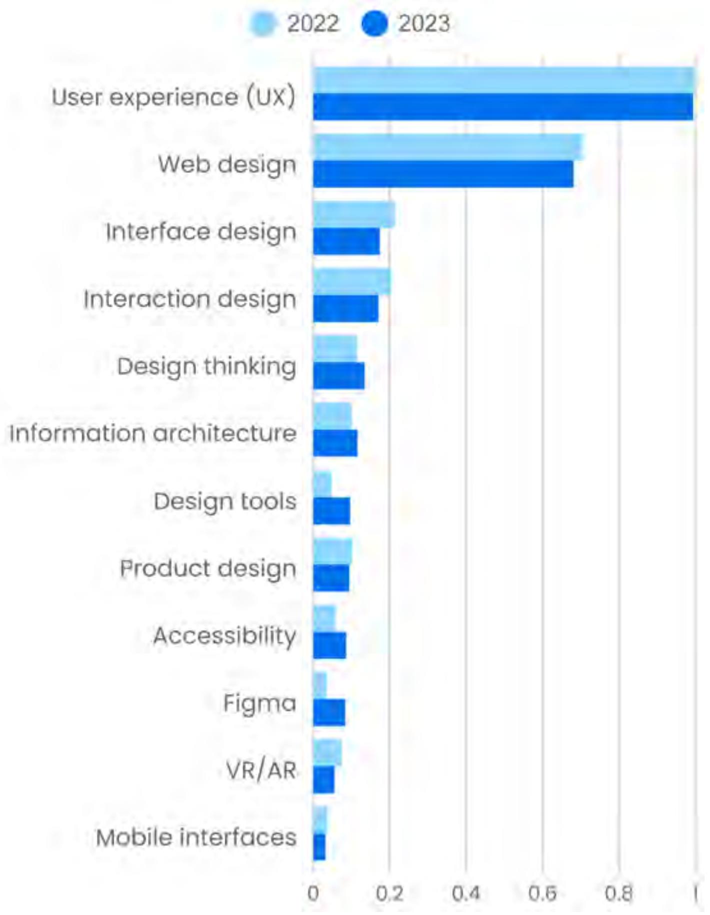

Figure 11. Design

Second, Figma is important because it's a breakthrough in tools for collaboration. Tools that allow remote employees to collaborate productively are crucial when coworkers can be anywhere: in an office,

2 Phillip Carter's article, "All the Hard Stuff Nobody Talks About when Building Products with LLMs," is worth reading. While it isn't specifically about design, almost everything he discusses is something designers should think about.

at home, or on another continent. The last year and a half has been full of talk about virtual reality, metaverses, and the like. But what few have realized is that the metaverse isn't about wearing goggles—it's about seamless collaboration with friends and coworkers. Use of content about AR and VR dropped 25% because people have missed the real story: we don't need 3D goggles; we need tools for collaboration. And, as with low-code, collaboration tools are both something to design with and something that needs to be designed. We're on the edge of a new way to look at the world.

Use of content about information architecture was up 16%, recovering from its decline from 2021 to 2022. The need to present information well, to design the environments in which we consume information online, has never been more important. Every day, there's more information to absorb and to navigate—and while artificial intelligence will no doubt help with that navigation, AI is as much a design problem as a design solution. (Though it's a "good problem" to have.) Designing and building for accessibility is clearly related to information architecture, and it's good to see more engagement with that content (up 47%). It's been a long time coming, and while there's still a long way to go, accessibility is being taken more seriously now than in the past. Websites that are designed to be usable by people with impairments aren't yet the rule, but they're no longer exceptions.

Professional Development

Almost everyone involved with software starts as a programmer. But that's rarely where they end. At some point in their career, they are asked to write a specification, lead a team, manage a group, or maybe even found a company or serve as an executive in an existing company.

O'Reilly is the last company to believe that software developers are neck-bearded geeks who want nothing more than to live in a cave and type on their terminals. We've spent most of our history fighting against that stereotype. Nevertheless, going beyond software development is a frequent source of anxiety. That's no doubt true for anyone stepping outside their comfort zone in just about any field, whether it's accounting, law, medicine, or something else. But at some point in your career, you have to do something that you aren't prepared to do. And, honestly, the best leaders are usually the ones who have some anxiety, not the ones whose reaction is “I was born to be a leader.”

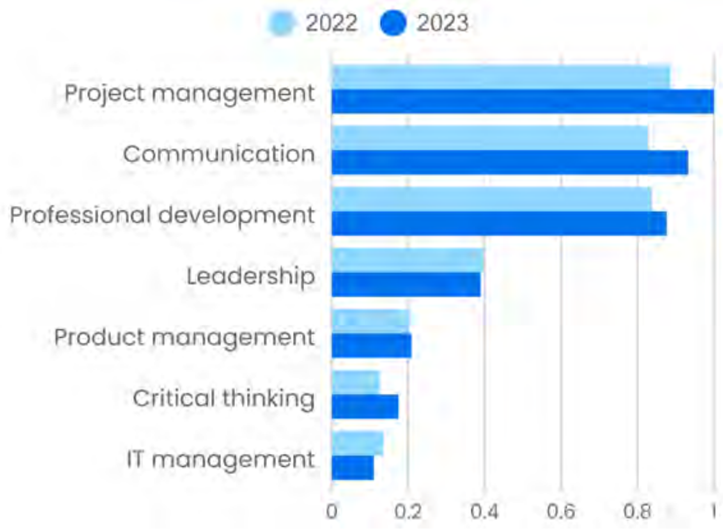

Figure 12. Professional development

For the past few years, our audience has been interested in professional growth that goes beyond just writing software or building models for AI and ML. Project management is up 13%; the ability to manage large projects is clearly seen as an asset for employees who are looking for their next promotion (or, in some cases, their next job). Whatever their goals might be, anyone looking for a promotion or a new job—or even just solidifying their hold on their current job—would be well served by improving their communications skills (up 23%). Professional development (up 22%) is a catch-all topic that appears to be responding to the same needs. What’s driving this? 2023 began and ended with a lot of news about layoffs. But despite well-publicized layoffs from huge companies that overhired during the pandemic, there’s little evidence that the industry as a whole has suffered. People who are laid off seem to be snapped up quickly by new employers. Nevertheless, anxiety is real, and the emphasis we’re seeing on professional development (and specifically, communications and project management skills) is partially a result of that anxiety. Another part of the story is no doubt the way AI is changing the workplace. If generative AI makes people more efficient, it frees up time for them to do other things, including strategic thinking about product development and leadership. It may finally be time to value "individuals and interactions over processes and tools," and "customer collaboration over contract negotiation," as the *Agile Manifesto* claims. Doing so will require a certain amount of reeducation, focusing on areas like communications, interpersonal skills, and strategic thinking.

Product management, the discipline of managing a product's lifecycle from the initial idea through development and release to the market, is also a desirable skill. So why is it only up 2.8% and not 20% like project management? Product management is a newer position in most companies; it has strong ties to marketing and sales, and as far as fear of layoffs is concerned (whether real or media driven), product management positions may be perceived as more vulnerable.

A look at the bottom of the chart shows that usage of content that teaches critical thinking grew 39%. That could be in part a consequence of ChatGPT and the explosion in artificial intelligence. Everyone knows that AI systems make mistakes, and almost every article that discusses these mistakes talks about the need for critical thinking to analyze AI's output and find errors. Is that the cause? Or is the desire for better critical thinking skills just another aspect of professional growth?

A Strange Year?

Back at the start, I said this was a strange year. As much as we like to talk about the speed at which technology moves, reality usually doesn't move that fast. When did we first start talking about data? Tim O'Reilly said "Data is the next Intel Inside" in 2005, almost 20 years ago. Kubernetes has been around for a decade, and that's not counting its prehistory as Google's Borg. Java was introduced in 1995, almost 30 years ago, and that's not counting its set-top box prehistory as Oak and Green. C++ first appeared in 1985. Artificial intelligence has a prehistory as long as computing itself. When did AI emerge from its wintry cave to dominate the data science landscape? 2016 or 2017, when we were amazed by programs that could sort images into dogs and cats? Sure, Java has changed a lot; so has what we do with data. Still, there's more continuity than disruption.

This year was one of the few years that could genuinely be called disruptive. Generative AI will change this industry in important ways. Programmers won't become obsolete, but programming as we know it might. Programming will have more to do with understanding problems and designing good solutions than specifying, step-by-step, what a computer needs to do. We're not there yet, but we can certainly imagine a day when a human language description leads reliably to working code, when "Do what I meant, not what I said" ceases to be the programmer's curse. That change has already begun, with tools like GitHub Copilot. But to thrive in that new industry, programmers will need to know more about architecture, more about design, more about human relations—and we're only starting to see that in our data, primarily for topics like product management and communications skills. And perhaps that's the definition of "disruptive": when our systems and our expectations change faster than our ability to keep up. I'm not worried about programmers "losing their jobs to an AI," and I really don't see that concern among the many programmers I talk to. But whatever profession you're in, you will lose out if you don't keep up. That isn't kind or humane; that's capitalism. And perhaps I should have used ChatGPT to write this report.[3]

Jerry Lee Lewis might have said "There's a whole lotta disruption goin' on." But despite all this disruption, much of the industry remains unchanged. People seem to have tired of the terms DevOps and SRE, but so it goes: the half-life of a buzzword is inevitably short, and these have been extraordinarily long-lived. The problems these buzzwords represent haven't gone away. Although we aren't yet collecting the data (and don't yet have enough content for which to collect data), developer platforms, self-service deployment, and platform engineering look like the next step in the evolution of IT operations. Will AI play a role in platform engineering? We'd be surprised if it didn't.

Movement to the cloud continues. While we've heard talk of cloud "repatriation," we see no evidence that it's happening. We do see evidence that organizations realize that the cloud is naturally hybrid and that focusing on a single cloud provider is short-sighted. There's also evidence that organizations are now paying more than lip service to security, particularly cloud security. That's a very good sign, especially after many years in which companies approached security

3 I didn't. Not even for data analysis.

by hoping nothing bad would happen. As many chess grandmasters have said, “Hope is never a good strategy.”

In the coming year, AI’s disruption will continue to play out. What consequences will it have for programming? How will jobs (and job prospects) change? How will IT adapt to the challenge of managing AI applications? Will they rely on AI-as-a-service providers like OpenAI, Azure, and Google, or will they build on open source models, which will probably run in the cloud? What new vulnerabilities will AI applications introduce into the security landscape? Will we see new architectural patterns and styles? Will AI tools for software architecture and design help developers grapple with the difficulties of microservices, or will it just create confusion?

In 2024, we’ll face all of these questions. Perhaps we’ll start to see answers. One thing is clear: it’s going to be an exciting year.

About the Author

Mike Loukides is vice president of content strategy for O'Reilly Media, Inc. He's edited many highly regarded books on technical subjects that don't involve Windows programming. He's particularly interested in programming languages, Unix and what passes for Unix these days, and system and network administration. Mike is the author of *System Performance Tuning* and a coauthor of *Unix Power Tools*. Most recently, he's been fooling around with data and data analysis, exploring languages like R, Mathematica, and Octave, and thinking about how to make books social. Mike can be reached on Twitter as @mikeloukides and on LinkedIn.