Technology Radar

An opinionated guide to today's technology landscape

Volume 31
Oct. 2024

/thoughtworks

Strategy. Design. Engineering.

About the Radar

Thoughtworkers are passionate about technology. We build it, research it, test it, open source it, write about it and constantly aim to improve it — for everyone. Our mission is to champion software excellence and revolutionize IT. We create and share the Thoughtworks Technology Radar in support of that mission. The Thoughtworks Technology Advisory Board, a group of senior technology leaders at Thoughtworks, creates the Radar. They meet regularly to discuss the global technology strategy for Thoughtworks and the technology trends that significantly impact our industry.

The Radar captures the output of the Technology Advisory Board's discussions in a format that provides value to a wide range of stakeholders, from developers to CTOs. The content is intended as a concise summary.

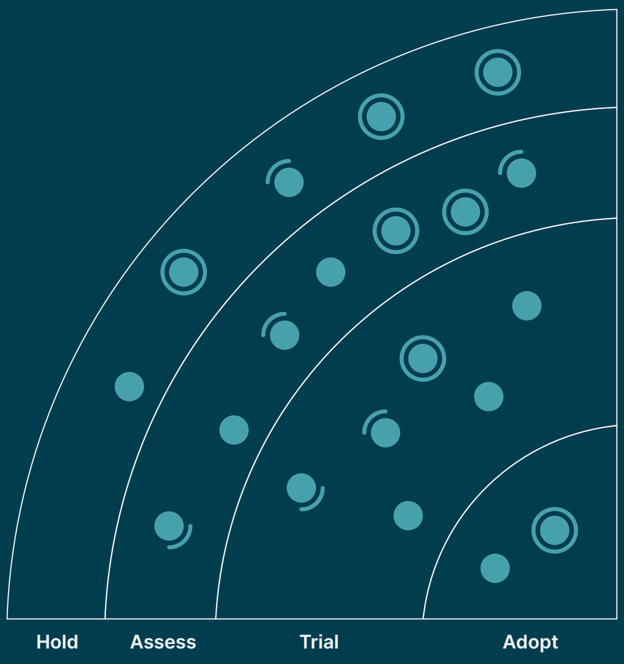
We encourage you to explore these technologies. The Radar is graphical in nature, grouping items into techniques, tools, platforms and languages and frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings to reflect our current position on them.

For more background on the Radar, see thoughtworks.com/radar/faq.

Radar at a glance

The Radar is all about tracking interesting things, which we refer to as blips. We organize the blips in the Radar using two categorizing elements: quadrants and rings. The quadrants represent different kinds of blips. The rings indicate our recommendation for using that technology.

A blip is a technology or technique that plays a role in software development. Blips are "in motion" — their position in the Radar often changes — usually indicating our increasing confidence in recommending them as they move through the rings.



Adopt: We feel strongly that the industry should be adopting these items. We use them when appropriate in our projects.

Trial: Worth pursuing. It's important to understand how to build up this capability. Enterprises can try this technology on a project that can handle the risk.

Assess: Worth exploring with the goal of understanding how it will affect your enterprise.

Hold: Proceed with caution.

◉ **New** ◔ **Moved in/out** ● **No change**

Our Radar is forward-looking. To make room for new items, we fade items that haven't moved recently, which isn't a reflection on their value but rather on our limited Radar real estate.

Contributors

The Technology Advisory Board (TAB) is a group of 20 senior technologists at Thoughtworks. The TAB meets twice a year face-to-face and biweekly virtually. Its primary role is to be an advisory group for Thoughtworks CTO Rachel Laycock.

The TAB acts as a broad body that can look at topics that affect technology and technologists at Thoughtworks. This edition of the Thoughtworks Technology Radar is based on a virtual meeting of the TAB in September 2024.

Rachel Laycock (CTO)

Martin Fowler (Chief Scientist)

Rebecca Parsons (CTO Emerita)

Bharani Subramaniam

Birgitta Böckeler

Camilla Falconi Crispim

Erik Dörnenburg

James Lewis

Ken Mugrage

Maya Ormaza

Mike Mason

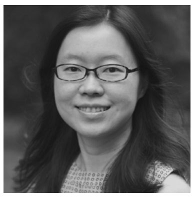
Neal Ford

Pawan Shah

Scott Shaw

Selvakumar Natesan

Shangqi Liu


Sofia Tania

Thomas Squeo

Vanya Seth

Will Amaral

Themes

Coding assistance antipatterns

To the surprise of no one, generative AI and LLMs dominated our conversations for this edition of the Radar, including emerging patterns around their use by developers. Patterns inevitably lead to antipatterns — contextualized situations developers should avoid. We see some antipatterns starting to appear in the hyperactive AI space, including the mistaken notion that humans can fully replace pair programming with AI as the companion, overreliance on coding assistance suggestions, code quality issues with generated code and faster growth rates of codebases. AI tends to solve problems via brute force rather than use abstractions, such as using dozens of stacked conditionals rather than the Strategy design pattern. The code quality issues in particular highlight an area of continued diligence by developers and architects to make sure they don't drown in "working-but-terrible" code. Thus, team members should double down on good engineering practices — such as unit testing, architectural fitness functions and other proven governance and validation techniques — to make sure that AI is helping your endeavors rather than encrypting your codebase with complexity.

Rust is anything but rusty

Rust has gradually become the systems programming language of choice. In every Radar session, Rust comes up in the subtext of our conversations over and over; many of the tools we discuss are written in Rust. It's the language of choice when replacing older system-level utilities as well as when re-writing part of an ecosystem for improved performance — indeed, the most common epithet for Rust-based tools seems to be "blazingly fast." For example, we see several tools in the Python ecosystem that have Rust-based alternatives to support noticeably better performance. The language designers and the community managed to create a well-liked ecosystem of core SDKs, libraries and dev tools, while providing stellar execution speed with fewer pitfalls than many of its predecessors. Many on our team are fans of Rust, and it seems that most developers who use it hold it in high regard.

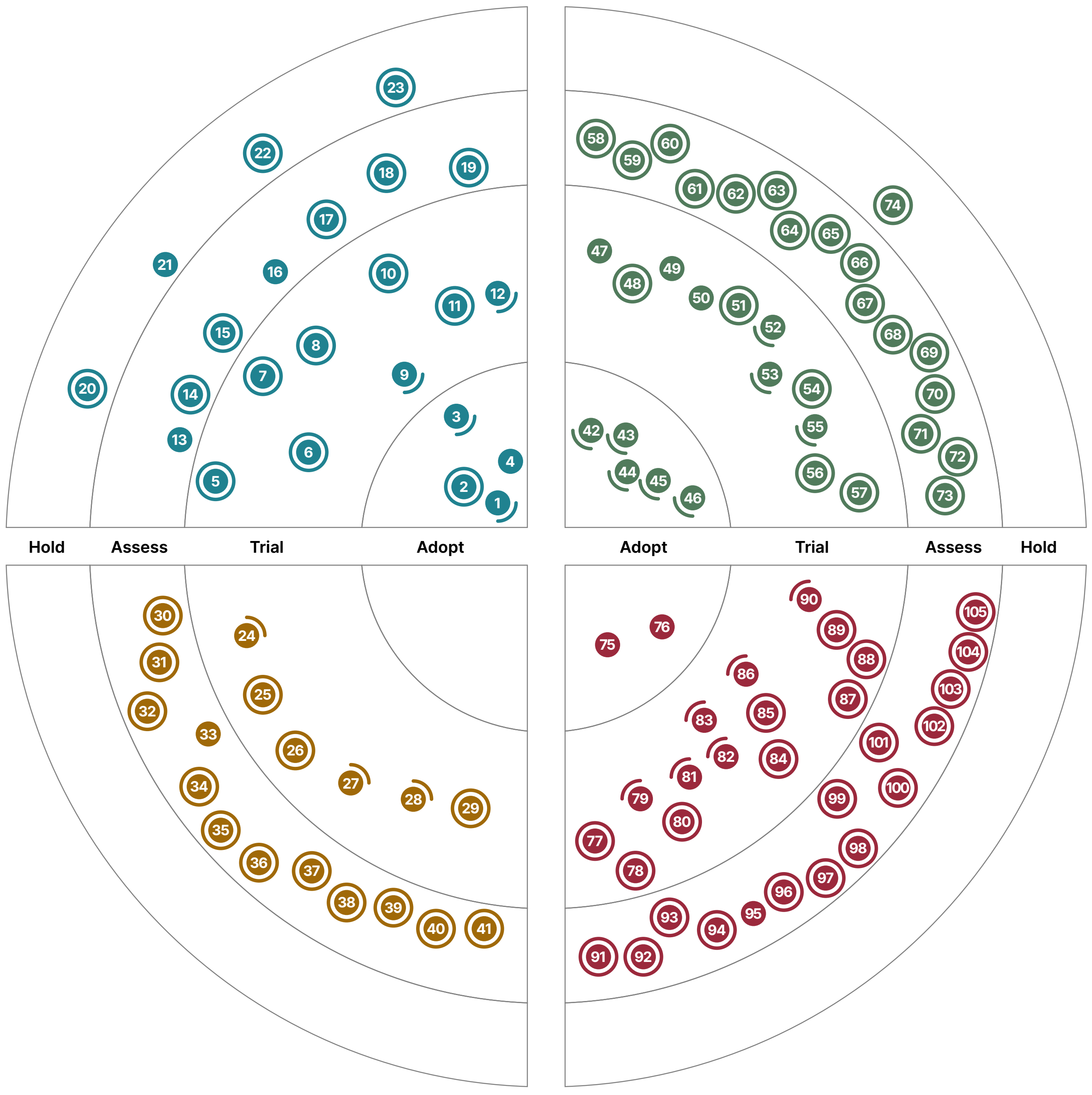
The gradual rise of WASM

WASM (WebAssembly) is a binary instruction format for a stack-based virtual machine, which sounds esoteric and too low level for most developer interests until people see the implications: the ability to run complex applications within a browser sandbox. WASM can run within existing JavaScript virtual machines, allowing applications that developers could formerly only implement in native frameworks and extensions to be embeddable within browsers. The four major browsers now support WASM 1.0 (Chrome, Firefox, Safari and Edge), opening exciting possibilities for sophisticated portable and cross-platform development. We've watched this standard over the last few years with great interest, and we're happy to see it start to flex its capabilities as a legitimate deployment target.

The Cambrian explosion of generative AI tools

Following the trajectory set out in the last few volumes of the Radar, we expected generative AI to feature prominently in our discussions. And, yet, we were still surprised by the explosion in the ecosystem of technology supporting language models: guardrails, evals, tools to build agents, frameworks to work with structured output, vector databases, cloud services and observability tools. In many ways, this rapid and varied growth makes perfect sense, though: The initial experience, the simplicity of a plain text prompt to a language model, has given way to engineering of software products. These may not live up to the dreams and wild claims that were made after people sent their first prompts to ChatGPT, but we see sensible and productive use of generative AI at many of our clients, and all these tools, platforms and frameworks play a role in getting LLM-based solutions into production. As was the case with the explosion of the JavaScript ecosystem around 2015, we expect this chaotic growth to continue for a while.

The Radar



● New ● Moved in/out ● No change

The Radar

Techniques

Adopt

1. 1% canary
2. Component testing
3. Continuous deployment
4. Retrieval-augmented generation (RAG)

Trial

5. Domain storytelling
6. Fine-tuning embedding models
7. Function calling with LLMs
8. LLM as a judge
9. Passkeys
10. Small language models
11. Synthetic data for testing and training models
12. Using GenAI to understand legacy codebases

Assess

13. AI team assistants
14. Dynamic few-shot prompting
15. GraphQL for data products
16. LLM-powered autonomous agents
17. Observability 2.0
18. On-device LLM inference
19. Structured output from LLMs

Hold

20. Complacency with AI-generated code
21. Enterprise-wide integration test environments
22. LLM bans
23. Replacing pair programming with AI

Platforms

Adopt

—

Trial

24. Databricks Unity Catalog
25. FastChat
26. GCP Vertex AI Agent Builder
27. Langfuse
28. Qdrant
29. Vespa

Assess

30. Azure AI Search
31. Databricks Delta Live Tables
32. Elastisys Compliant Kubernetes
33. FoundationDB
34. Golem
35. Iggy
36. Iroh
37. Large vision model (LVM) platforms
38. OpenBCI Galea
39. PGLite
40. SpinKube
41. Unblocked

Hold

—

Tools

Adopt

42. Bruno
43. K9s
44. SOPS
45. Visual regression testing tools
46. Wiz

Trial

47. AWS Control Tower
48. CCMenu
49. ClickHouse
50. Devbox
51. Difftastic
52. LinearB
53. pgvector
54. Snapcraft build tool
55. Spinnaker
56. TypeScript OpenAPI
57. Unleash

Assess

58. Astronomer Cosmos
59. ColPali
60. Cursor
61. Data Mesh Manager
62. GitButler
63. JetBrains AI Assistant
64. Mise
65. Mockoon
66. Raycast
67. ReadySet
68. Rspack
69. Semantic Router
70. Software engineering agents
71. uv
72. Warp
73. Zed

Hold

74. CocoaPods

Languages and Frameworks

Adopt

75. dbt
76. Testcontainers

Trial

77. CAP
78. CARLA
79. Databricks Asset Bundles
80. Instructor
81. Kedro
82. LiteLLM
83. LlamaIndex
84. LLM Guardrails
85. Medusa
86. Pkl
87. ROS 2
88. seL4
89. SetFit
90. vLLM

Assess

91. Apache XTable™
92. dbldatagen
93. DeepEval
94. DSPy
95. Flutter for Web
96. kotaemon
97. Lenis
98. LLMLingua
99. Microsoft Autogen
100. Pingora
101. Ragas
102. Score
103. shadcn
104. Slint
105. SST

Hold

—

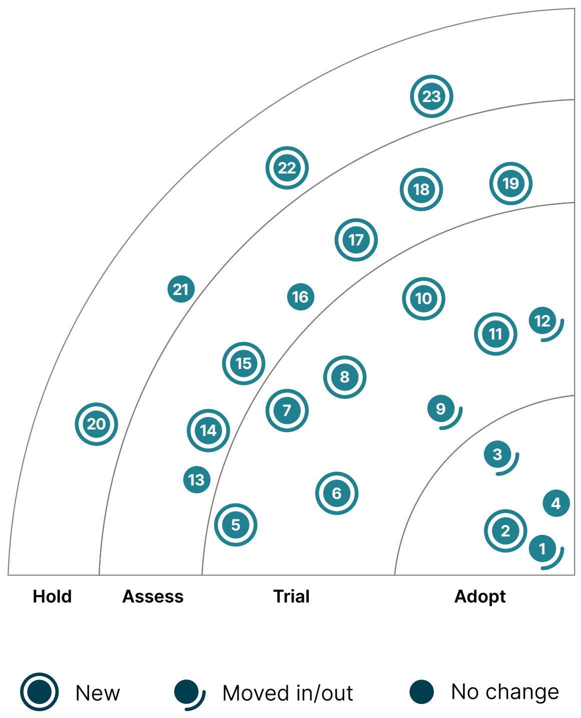
Techniques

Adopt
1. 1% canary
2. Component testing
3. Continuous deployment
4. Retrieval-augmented generation (RAG)

Trial
5. Domain storytelling
6. Fine-tuning embedding models
7. Function calling with LLMs
8. LLM as a judge
9. Passkeys
10. Small language models
11. Synthetic data for testing and training models
12. Using GenAI to understand legacy codebases

Assess
13. AI team assistants
14. Dynamic few-shot prompting
15. GraphQL for data products
16. LLM-powered autonomous agents
17. Observability 2.0
18. On-device LLM inference
19. Structured output from LLMs

Hold
20. Complacency with AI-generated code
21. Enterprise-wide integration test environments
22. LLM bans
23. Replacing pair programming with AI



1. 1% canary

Adopt

For many years, we've used the canary release approach to encourage early feedback on new software versions, while reducing risk through incremental rollout to selected users. The 1% canary is a useful technique where we roll out new features to a very small segment (say 1%) of users carefully chosen across various user categories. This enables teams to capture fast user feedback, observe the impact of new releases on performance and stability and respond as necessary. This technique becomes especially crucial when teams are rolling out software updates to mobile applications or a fleet of devices like edge computing devices or software-defined vehicles. With proper observability and early feedback, it gives the opportunity to contain the blast radius in the event of unexpected scenarios in production. While canary releases can be useful to get faster user feedback, we believe starting with a small percentage of users is mandatory to reduce and contain the risk of large-scale feature rollouts.

2. Component testing

Adopt

Automated testing remains a cornerstone of effective software development. For front-end tests we can argue whether the distribution of different test types should be the classic test pyramid or whether it should be a trophy shape. In either case, though, teams should focus on component testing because test suites should be stable and run quickly. Instead, what we're seeing is that teams forgo mastering component testing in favor of end-to-end browser-based testing as well as very narrowly defined unit tests. Unit tests have a tendency to force components to expose what should be purely internal functionality, while browser-based tests are slow, more flaky and harder to debug. Our recommendation is to have a significant amount of component tests and use a library like jsdom to run the component tests *in memory*. Browser tools like Playwright of course still have a place in end-to-end tests, but they shouldn't be used for component testing.

3. Continuous deployment

Adopt

We believe organizations should adopt continuous deployment practices whenever possible. Continuous deployment is the practice of automatically deploying every change that passes automated tests to production. This practice is a key enabler of fast feedback loops and allows organizations to deliver value to customers more quickly and efficiently. Continuous deployment differs from continuous delivery in that it only requires that code can be deployed at any time; it doesn't require that every change actually is deployed to production. We've hesitated to move continuous deployment into the Adopt ring in the past, as it's a practice that requires a high level of maturity in other areas of software delivery and is therefore not appropriate for all teams. However, Thoughtworker Valentina Servile's recent book *Continuous Deployment* provides a comprehensive guide to implementing the practice in an organization. It offers a roadmap for organizations to follow in order to achieve the level of maturity required to adopt continuous deployment practices.

4. Retrieval-augmented generation (RAG)

Adopt

Retrieval-augmented generation (RAG) is the preferred pattern for our teams to improve the quality of responses generated by a large language model (LLM). We've successfully used it in many projects, including the Jugalbandi AI platform. With RAG, information about relevant and trustworthy documents is stored in a database. For a given prompt, the database is queried, relevant documents are retrieved and the prompt augmented with the content of the documents, thus providing richer context to the LLM. This results in higher quality output and greatly reduced hallucinations. The context window — which determines the maximum size of the LLM input — has grown significantly with newer models, but selecting the most relevant documents is still a crucial step. Our experience indicates that a carefully constructed smaller context can yield better results than a broad and large context. Using a large context is also slower and more expensive. We used to rely solely on embeddings stored in a vector database to identify additional context. Now, we're seeing reranking and hybrid search: search tools such as Elasticsearch Relevance Engine as well as approaches like GraphRAG that utilize knowledge graphs created with the help of an LLM. A graph-based approach has worked particularly well in our work on understanding legacy codebases with GenAI.

5. Domain storytelling

Trial

Domain-driven design (DDD) has become a foundational approach to the way we develop software. We use it to model events, to guide software designs, to establish context boundaries around microservices and to elaborate nuanced business requirements. DDD establishes a ubiquitous language that both nontechnical stakeholders and software developers can use to communicate effectively about the business. Once established, domain models evolve, but many teams find it hard to get started with DDD. There's no one-size-fits-all approach to building an initial domain model. One promising technique we've encountered recently is domain storytelling. Domain storytelling is a facilitation technique where business experts are prompted to describe activities in the business. As the experts are guided through their narration, a facilitator uses a pictographic language to capture the relationships and actions between entities and actors. The process of making these stories visible helps to clarify and develop a shared understanding among participants. Since there is no single best approach to developing a domain model, domain storytelling offers a noteworthy alternative or, for a more comprehensive approach to DDD, companion to Event Storming, another technique we often use to get started with DDD.

6. Fine-tuning embedding models

Trial

When building LLM applications based on retrieval-augmented generation (RAG), the quality of embeddings directly impacts both retrieval of the relevant documents and response quality. Fine-tuning embedding models can enhance the accuracy and relevance of embeddings for specific tasks or domains. Our teams fine-tuned embeddings when developing domain-specific LLM applications for which precise information extraction was crucial. However, consider the trade-offs of this approach before you rush to fine-tune your embedding model.

7. Function calling with LLMs

Trial

Function calling with LLMs refers to the ability to integrate LLMs with external functions, APIs or tools by determining and invoking the appropriate function based on a given query and associated documentation. This extends the utility of LLMs beyond text generation, allowing them to perform specific tasks such as information retrieval, code execution and API interaction. By triggering external functions or APIs, LLMs can perform actions that were previously outside their standalone capabilities. This technique enables LLMs to act on their outputs, effectively bridging the gap between thought and action — similar to how humans use tools to accomplish various tasks. By introducing function calling, LLMs add determinism and factuality to the generation process, striking a balance between creativity and logic. This method allows LLMs to connect to internal systems and databases or even perform internet searches via connected browsers. Models like OpenAI's GPT series support function calling and fine-tuned models like Gorilla are specifically designed to enhance the accuracy and consistency of generating executable API calls from natural language instructions. As a technique, function calling fits within retrieval-augmented generation (RAG) and agent architectures. It should be viewed as an abstract pattern of use, emphasizing its potential as a foundational tool in diverse implementations rather than a specific solution.

8. LLM as a judge

Trial

Many systems we build have two key characteristics: being able to provide an answer based on questions about a large data set, and being next to impossible to follow how it arrived at that answer. Despite this opacity we still want to assess and improve the quality of the responses. With the LLM as a judge pattern we use an LLM to evaluate the responses of another system, which in turn might be based on an LLM. We've seen this pattern used to evaluate the relevance of search results in a product catalog and to assess whether an LLM-based chatbot was guiding its users in a sensible direction. Naturally, the evaluator system must be set up and calibrated carefully. It can drive significant efficiency gains, which, in turn, translates to lower costs. This is an ongoing area of research, with the current state summarized in this article.

9. Passkeys

Trial

Shepherded by the FIDO alliance and backed by Apple, Google and Microsoft, passkeys are nearing mainstream usability. Setting up a new login with passkeys generates a key pair: the website receives the public key and the user keeps the private key. Handling login uses asymmetric cryptography. The user proves they're in possession of the private key, which is stored on the user's device and never sent to the website. Access to passkeys is protected using biometrics or a PIN. Passkeys can be stored and synced within the big tech ecosystems, using Apple's iCloud Keychain, Google Password Manager or Windows Hello. For multiplatform users, the Client to Authenticator Protocol (CTAP) makes it possible for passkeys to be kept on a different device other than the one that creates the key or needs it for login. The most common objection to using passkeys claims that they are a challenge for less tech-savvy users, which is, we believe, self-defeating. These are often the same users who have poor password discipline and would therefore benefit the most from alternative methods. In practice, systems that use passkeys can fall back to more traditional authentication methods if required.

10. Small language models

Trial

Large language models (LLMs) have proven useful in many areas of applications, but the fact that they are large can be a source of problems: responding to a prompt requires a lot of compute resources, making queries slow and expensive; the models are proprietary and so large that they must be hosted in a cloud by a third party, which can be problematic for sensitive data; and training a model is prohibitively expensive in most cases. The last issue can be addressed with the RAG pattern, which side-steps the need to train and fine-tune foundational models, but cost and privacy concerns often remain. In response, we're now seeing growing interest in small language models (SLMs). In comparison to their more popular siblings, they have fewer weights and less precision, usually between 3.5 billion and 10 billion parameters. Recent research suggests that, in the right context, when set up correctly, SLMs can perform as well as or even outperform LLMs. And their size makes it possible to run them on edge devices. We've previously mentioned Google's Gemini Nano, but the landscape is evolving quickly, with Microsoft introducing its Phi-3 series, for example.

11. Synthetic data for testing and training models

Trial

Synthetic data set creation involves generating artificial data that can mimic real-world scenarios without relying on sensitive or limited-access data sources. While synthetic data for structured data sets has been explored extensively (e.g., for performance testing or privacy-safe environments), we're seeing renewed use of synthetic data for unstructured data. Enterprises often struggle with a lack of labeled domain-specific data, especially for use in training or fine-tuning LLMs. Tools like Bonito and Microsoft's AgentInstruct can generate synthetic instruction-tuning data from raw sources such as text documents and code files. This helps accelerate model training while reducing costs and dependency on manual data curation. Another important use case is generating synthetic data to address imbalanced or sparse data, which is common in tasks like fraud detection or customer segmentation. Techniques such as SMOTE help balance data sets by artificially creating minority class instances. Similarly, in industries like finance, generative adversarial networks (GANs) are used to simulate rare transactions, allowing models to be robust in detecting edge cases and improving overall performance.

12. Using GenAI to understand legacy codebases

Trial

Generative AI (GenAI) and large language models (LLMs) can help developers write and understand code. Help with understanding code is especially useful in the case of legacy codebases with poor, out-of-date or misleading documentation. Since we last wrote about this, techniques and products for using GenAI to understand legacy codebases have further evolved, and we've successfully used some of them in practice, notably to assist reverse engineering efforts for mainframe modernization. A particularly promising technique we've used is a retrieval-augmented generation (RAG) approach where the information retrieval is done on a knowledge graph of the codebase. The knowledge graph can preserve structural information about the codebase beyond what an LLM could derive from the textual code alone. This is particularly helpful in legacy codebases that are less self-descriptive and cohesive. An additional opportunity to improve code understanding is that the graph can be further enriched with existing and AI-generated documentation, external dependencies, business domain knowledge or whatever else is available that can make the AI's job easier.

13. AI team assistants

Assess

AI coding assistance tools are mostly talked about in the context of assisting and enhancing an individual contributor's work. However, software delivery is and will remain team work, so you should be looking for ways to create AI team assistants that help create the 10x team, as opposed to a bunch of siloed AI-assisted 10x engineers. Fortunately, recent developments in the tools market are moving us closer to making this a reality. Unblocked is a platform that pulls together all of a team's knowledge sources and integrates them intelligently into team members' tools. And Atlassian's Rovo brings AI into the most widely used team collaboration platform, giving teams new types of search and access to their documentation, in addition to unlocking new ways of automation and software practice support with Rovo agents. While we wait for the market to further evolve in this space, we've been exploring the potential of AI for knowledge amplification and team practice support ourselves: We open-sourced our Haiven team assistant and started gathering learnings with AI assistance for noncoding tasks like requirements analysis.

14. Dynamic few-shot prompting

Assess

Dynamic few-shot prompting builds upon few-shot prompting by dynamically including specific examples in the prompt to guide the model's responses. Adjusting the number and relevance of these examples optimizes context length and relevancy, thereby improving model efficiency and performance. Libraries like scikit-llm implement this technique using nearest neighbor search to fetch the most relevant examples aligned with the user query. This technique lets you make better use of the model's limited context window and reduce token consumption. The open-source SQL generator vanna leverages dynamic few-shot prompting to enhance response accuracy.

15. GraphQL for data products

Assess

GraphQL for data products is the technique of using GraphQL as an output port for data products for clients to consume the product. We've talked about GraphQL as an API protocol and how it enables developers to create a unified API layer that abstracts away the underlying data complexity, providing a more cohesive and manageable interface for clients. GraphQL for data products makes it seamless for consumers to discover the data format and relationships with GraphQL schema and use familiar client tools. Our teams are exploring this technique in specific use cases like talk-to-data to explore and discover big data insights with the help of large language models, where the GraphQL queries are constructed by LLMs based on the user prompt and the GraphQL schema is used in the LLM prompts for reference.

16. LLM-powered autonomous agents

Assess

LLM-powered autonomous agents are evolving beyond single agents and static multi-agent systems with the emergence of frameworks like Autogen and CrewAI. This technique allows developers to break down a complex activity into several smaller tasks performed by agents where each agent is assigned a specific role. Developers can use preconfigured tools for performing the task, and the agents converse among themselves and orchestrate the flow. The technique is still in its early stages of development. In our experiments, our teams have encountered issues like agents going into continuous loops and uncontrolled behavior. Libraries like LangGraph offer greater control of agent interactions with the ability to define the flow as a graph. If you use this technique, we suggest implementing fail-safe mechanisms, including timeouts and human oversight.

17. Observability 2.0

Assess

Observability 2.0 represents a shift from traditional, disparate monitoring tools to a unified approach that leverages structured, high-cardinality event data in a single data store. This model captures rich, raw events with detailed metadata to provide a single source of truth for comprehensive analysis. By storing events in their raw form, it simplifies correlation and supports real-time and forensic analysis and enables deeper insights into complex, distributed systems. This approach allows for high-resolution monitoring and dynamic investigation capabilities. Observability 2.0 prioritizes capturing high-cardinality and high-dimensional data, allowing detailed examination without performance bottlenecks. The unified data store reduces complexity, offering a coherent view of system behavior, and aligning observability practices more closely with the software development lifecycle.

18. On-device LLM inference

Assess

Large language models (LLMs) can now run in web browsers and on edge devices like smartphones and laptops, enabling on-device AI applications. This allows for secure handling of sensitive data without cloud transfer, extremely low latency for tasks like edge computing and real-time image or video processing, reduced costs by performing computations locally and functionality even when internet connectivity is unreliable or unavailable. This is an active area of research and development. Previously, we highlighted MLX, an open-source framework for efficient machine learning on Apple silicon. Other emerging tools include Transformers.js and Chatty. Transformers.js lets you run transformers in the browser using ONNX Runtime, supporting models converted from PyTorch, TensorFlow and JAX. Chatty leverages WebGPU to run LLMs natively and privately in the browser, offering a feature-rich in-browser AI experience.

19. Structured output from LLMs

Assess

Structured output from LLMs refers to the practice of constraining a language model's response into a defined schema. This can be achieved either through instructing a generalized model to respond in a particular format or by fine-tuning a model so it "natively" outputs, for example, JSON. OpenAI now supports structured output, allowing developers to supply a JSON Schema, pydantic or Zod object to constrain model responses. This capability is particularly valuable for enabling function calling, API interactions and external integrations, where accuracy and adherence to a format are critical. Structured output not only enhances the way LLMs can interface with code but also supports broader use cases like generating markup for rendering charts. Additionally, structured output has been shown to reduce the chance of hallucinations within model output.

20. Complacency with AI-generated code

Hold

AI coding assistants like GitHub Copilot and Tabnine have become very popular. According to StackOverflow's 2024 developer survey, "72% of all respondents are favorable or very favorable of AI tools for development". While we also see their benefits, we're wary about the medium- to long-term impact this will have on code quality and caution developers about complacency with AI-generated code. It's all too tempting to be less vigilant when reviewing AI suggestions after a few positive experiences with an assistant. Studies like this one by GitClear show a trend of faster growing codebases, which we suspect coincide with larger pull requests. And this study by GitHub has us wondering whether the mentioned 15% increase of the pull request merge rate is actually a good thing or whether people are merging larger pull requests faster because they trust the AI results too much. We're still using the basic "getting started" advice we gave over a year ago, which is to beware of automation bias, sunk cost fallacy, anchoring bias and review fatigue. We also recommend that programmers develop a good mental framework about where and when not to use and trust AI.

21. Enterprise-wide integration test environments

Hold

Creating enterprise-wide integration test environments is a common, wasteful practice that slows everything down. These environments invariably become a precious resource that's hard to replicate and a bottleneck to development. They also provide a false sense of security due to inevitable discrepancies in data and configuration overhead between environments. Ironically, a common objection to the alternatives — either ephemeral environments or multiple on-prem test environments — is cost. However, this fails to take into account the cost of the delays caused by enterprise-wide integration test environments as development teams wait for other teams to finish or for new versions of dependent systems to be deployed. Instead, teams should use ephemeral environments and, preferably, a suite of tests owned by the development team that can be spun up and discarded cheaply, using fake stubs for their systems rather than actual replicas. For other techniques that support this alternative take a look at contract testing, decoupling deployment from release, focus on mean time to recovery and testing in production.

22. LLM bans

Hold

Rather than instituting blanket LLM bans in the workplace, organizations should focus on providing access to an approved set of AI tools. A ban only pushes employees to find unapproved and potentially unsafe workarounds, creating unnecessary risks. Much like the early days of personal computing, people will use whatever tools they feel are effective to get their work done, regardless of the barriers in place. By not providing a safe and endorsed alternative, companies risk employees using unapproved LLMs which come with intellectual property, data leakage and liability risks. Instead, offering secure, enterprise-approved LLMs or AI tools ensures both safety and productivity. A well-governed approach allows organizations to manage data privacy, security, compliance and cost concerns while still empowering employees with the capabilities that LLMs offer. In the best case, well-managed access to AI tools can accelerate organizational learning around the best ways to use AI in the workplace.

23. Replacing pair programming with AI

Hold

When people talk about coding assistants, the topic of pair programming inevitably comes up. Our profession has a love-hate relationship with it: some swear by it, others can't stand it. Coding assistants now beg the question, can a human pair with the AI, instead of another human, and get the same results for the team? GitHub Copilot even calls itself "your AI pair programmer." While we do think a coding assistant can bring some of the benefits of pair programming, we advise against fully replacing pair programming with AI. Framing coding assistants as pair programmers ignores one of the key benefits of pairing: to make the team, not just the individual contributors, better. Coding assistants can offer benefits for getting unstuck, learning about a new technology, onboarding or making tactical work faster so that we can focus on the strategic design. But they don't help with any of the team collaboration benefits, like keeping the work-in-progress low, reducing handoffs and relearning, making continuous integration possible or improving collective code ownership.

Platforms

Adopt

—

Trial

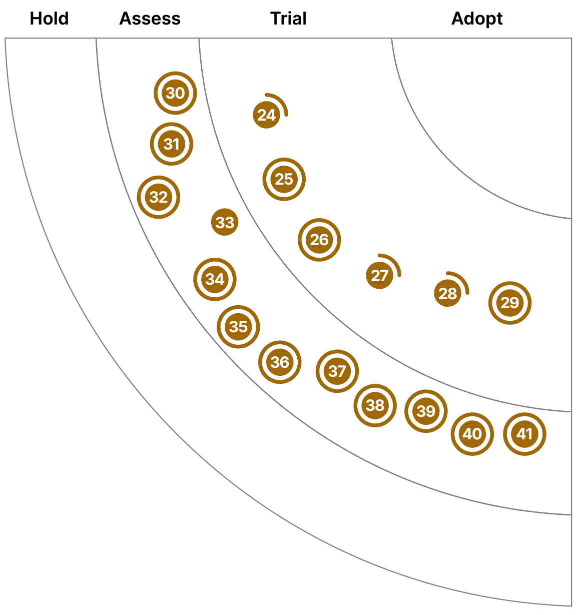
24. Databricks Unity Catalog
25. FastChat
26. GCP Vertex AI Agent Builder
27. Langfuse
28. Qdrant
29. Vespa

Assess

30. Azure AI Search
31. Databricks Delta Live Tables
32. Elastisys Compliant Kubernetes
33. FoundationDB
34. Golem
35. Iggy
36. Iroh
37. Large vision model (LVM) platforms
38. OpenBCI Galea
39. PGLite
40. SpinKube
41. Unblocked

Hold

—



◉ New   ◔ Moved in/out   ● No change

24. Databricks Unity Catalog

Trial

Databricks Unity Catalog is a data governance solution for assets such as files, tables or machine learning models in a lakehouse. It's a managed version of the open-source Unity Catalog that can be used to govern and query data kept in external stores or under Databricks management. In the past our teams have worked with a variety of data management solutions such as Hive metastore or Microsoft Purview. However, Unity Catalog's combined support for governance, metastore management and data discovery makes it attractive because it reduces the need to manage multiple tools. One complication our team discovered is the lack of automatic disaster recovery in the Databricks-managed Unity Catalog. They were able to configure their own backup and restore functionality but a Databricks-provided solution would have been more convenient. Note that even though these governance platforms usually implement a centralized solution to ensure consistency across workspaces and workloads, the responsibility to govern can still be federated by enabling individual teams to govern their own assets.

25. FastChat

Trial

FastChat is an open platform for training, serving and evaluating large language models. Our teams use its model-serving capabilities to host multiple models — Llama 3.1 (8B and 70B), Mistral 7B and Llama-SQL — for different purposes, all in a consistent OpenAI API format. FastChat operates on a controller-worker architecture, allowing multiple workers to host different models. It supports worker types such as vLLM, LiteLLM and MLX. We use vLLM model workers for their high throughput capabilities. Depending on the use case — latency or throughput — different types of FastChat model workers can be created and scaled. For example, the model used for code suggestions in developer IDEs requires low latency and can be scaled with multiple FastChat workers to handle concurrent requests efficiently. In contrast, the model used for Text-to-SQL doesn't need multiple workers due to lower demand or different performance requirements. Our teams leverage FastChat's scaling capabilities for A/B testing. We configure FastChat workers with the same model but different hyperparameter values and pose identical questions to each, identifying optimal hyperparameter values. When transitioning models in live services, we conduct A/B tests to ensure seamless migration. For example, we recently migrated from CodeLlama 70B to Llama 3.1 70B for code suggestions. By running both models concurrently and comparing outputs, we verified the new model met or exceeded the previous model's performance without disrupting the developer experience.

26. GCP Vertex AI Agent Builder

Trial

GCP Vertex AI Agent Builder provides a flexible platform for creating AI agents using natural language or a code-first approach. The tool seamlessly integrates with enterprise data through third-party connectors and has all the necessary tools to build, prototype and deploy AI agents. As the need for AI agents grows, many teams struggle with understanding their benefits and implementation. GCP Vertex AI Agent Builder makes it easier for developers to prototype agents quickly and handle complex data tasks with minimal setup. Our developers have found it particularly useful for building agent-based systems — such as knowledge bases or automated support systems — that efficiently manage both structured and unstructured data. That makes it a valuable tool for developing AI-driven solutions.

27. Langfuse

Trial

LLMs function as black boxes, making it difficult to determine their behavior. Observability is crucial for opening this black box and understanding how LLM applications operate in production. Our teams have had positive experiences with Langfuse for observing, monitoring and evaluating LLM-based applications. Its tracing, analytics and evaluation capabilities allow us to analyze completion performance and accuracy, manage costs and latency and understand production usage patterns, thus facilitating continuous, data-driven improvements. Instrumentation data provides complete traceability of the request-response flow and intermediate steps, which can be used as test data to validate the application before deploying new changes. We've utilized Langfuse with RAG (retrieval-augmented generation), among other LLM architectures, and LLM-powered autonomous agents. In a RAG-based application, for example, analyzing low-scoring conversation traces helps identify which parts of the architecture — pre-retrieval, retrieval or generation — need refinement. Another option worth considering in this space is Langsmith.

28. Qdrant

Trial

Qdrant is an open-source vector similarity search engine and database written in Rust. It supports a wide range of text and multimodal dense vector embedding models. Our teams have used open-source embeddings like MiniLM-v6 and BGE for multiple product knowledge bases. We use Qdrant as an enterprise vector store with multi-tenancy to store vector embeddings as separate collections, isolating each product's knowledge base in storage. User access policies are managed in the application layer.

29. Vespa

Trial

Vespa is an open-source search engine and big data processing platform. It's particularly well-suited for applications that require low latency and high throughput. Our teams like Vespa's ability to implement hybrid search using multiple retrieval techniques, to efficiently filter and sort many types of metadata, to implement multi-phased ranking, to index multiple vectors (e.g., for each chunk) per document without duplicating all the metadata into separately indexed documents and to retrieve data from multiple indexed fields at once.

30. Azure AI Search

Assess

Azure AI Search, formerly known as Cognitive Search, is a cloud-based search service designed to handle structured and unstructured data for applications like knowledge bases, particularly in retrieval-augmented generation (RAG) setups. It supports various types of search, including keyword, vector and hybrid search, which we believe will become increasingly important. The service automatically ingests common unstructured data formats including PDF, DOC and PPT, streamlining the process of creating searchable content. Additionally, it integrates with other Azure services, such as Azure OpenAI, allowing users to build applications with minimal manual integration effort. From our experience, Azure AI Search performs reliably and is well-suited for projects hosted in the Azure environment. Through its custom skills, users can also define specific data processing steps. Overall, if you're working within the Azure ecosystem and need a robust search solution for a RAG application, Azure AI Search is worth considering.

31. Databricks Delta Live Tables

Assess

Databricks Delta Live Tables is a declarative framework designed for building reliable, maintainable and testable data processing pipelines. It allows data engineers to define data transformations using a declarative approach and automatically manages the underlying infrastructure and data flow. One of the standout features of Delta Live Tables is its robust monitoring capabilities. It provides a directed acyclic graph (DAG) of your entire data pipeline, visually representing data movement from source to final tables. This visibility is crucial for complex pipelines, helping data engineers and data scientists track data lineage and dependencies. Delta Live Tables is deeply integrated into the Databricks ecosystem, which also brings some challenges to customizing interfaces. We recommend teams carefully evaluate the compatibility of input and output interfaces before using Delta Live Tables.

32. Elastisys Compliant Kubernetes

Assess

Elastisys Compliant Kubernetes is a specialized Kubernetes distribution designed to meet stringent regulatory and compliance requirements, particularly for organizations operating in highly regulated industries such as healthcare, finance and government. It has automated security processes, provides multicloud and on-premises support and is built on top of a zero-trust security architecture. The emphasis on built-in compliance with laws such as GDPR and HIPAA and controls like ISO27001 makes it an attractive option for companies that need a secure, compliant and reliable Kubernetes environment.

33. FoundationDB

Assess

FoundationDB is a multi-model database, acquired by Apple in 2015 and then open-sourced in April 2018. The core of FoundationDB is a distributed key-value store, which provides strict serializability transactions. Since we first mentioned it in the Radar, it has seen significant improvements — including smart data distributions to avoid write hotspots, a new storage engine, performance optimizations and multi-region replication support. We're using FoundationDB in one of our ongoing projects and are very impressed by its unbundled architecture. This architecture allows us to scale different parts of the cluster independently. For example, we can adjust the number of transaction logs, storage servers and proxies based on our specific workload and hardware. Despite its extensive features, FoundationDB remains remarkably easy to run and operate large clusters.

34. Golem

Assess

Durable computing, a recent movement in distributed computing, uses an architecture style of explicit state machine to persist the memory of serverless servers for better fault tolerance and recovery. Golem is one of the promoters of this movement. The concept can work in some scenarios, such as long-running microservices sagas or long-running workflows in AI agent orchestration. We've evaluated Temporal previously for similar purposes and Golem is another choice. With Golem you can write WebAssembly components in any supported language besides Golem being deterministic and supporting fast startup times. We think Golem is an exciting platform worth evaluating.

35. Iggy

Assess

Iggy, a persistent message streaming platform written in Rust, is a relatively new project with impressive features. It already supports multiple streams, topics and partitions, at-most-once delivery, message expiry and TLS support over QUIC, TCP and HTTP protocols. Running as a single server, Iggy currently achieves high throughput for both read and write operations. With upcoming clustering and io_uring support, Iggy can be a potential alternative to Kafka.

36. Iroh

Assess

Iroh is a relatively new distributed file storage and content delivery system that's designed as an evolution of existing decentralized systems like IPFS (InterPlanetary File System). Both Iroh and IPFS can be used to create decentralized networks for storing, sharing and accessing content addressed using opaque content identifiers. However, Iroh removes some of the limitations of IPFS implementations such as having no maximum block size and providing a syncing mechanism for data via range-based set reconciliation over documents. The project's roadmap includes bringing the technology to the browser via WASM, which raises some intriguing possibilities for building decentralization into web applications. If you don't want to host your own Iroh nodes, you can use its cloud service, iroh.network. There are already several SDKs available in a variety of languages, and one goal is to be more user-friendly and easier to use than alternative IPFS systems. Even though Iroh is still in its very early days, it's worth keeping an eye on it, as it could become a significant player in the decentralized storage space.

37. Large vision model (LVM) platforms

Assess

Large language models (LLMs) grab so much of our attention these days, we tend to overlook ongoing developments in large vision models (LVMs). These models can be used to segment, synthesize, reconstruct and analyze video streams and images, sometimes in combination with diffusion models or standard convolutional neural networks. Despite the potential for LVMs to revolutionize the way we work with visual data, we still face significant challenges in adapting and applying them in production environments. Video data, for instance, presents unique engineering challenges for collecting training data, segmenting and labeling objects, fine-tuning models and then deploying the resulting models and monitoring them in production. So, while LLMs lend themselves to simple chat interfaces or plain text APIs, a computer vision engineer or data scientist must manage, version, annotate and analyze large quantities of streaming video data; this work requires a visual interface. LVM platforms are a new category of tools and services — including V7, Nvidia Deepstream SDK and Roboflow — that have emerged to address these challenges. Deepstream and Roboflow are particularly interesting to us because they combine an integrated GUI development environment for managing and annotating video streams with a set of Python, C++ or REST APIs to invoke the models from application code.

38. OpenBCI Galea

Assess

There is growing interest in the use of brain-computer interfaces (BCIs) and their potential application to assistive technologies. Non-invasive technologies using electroencephalography (EEG) and other electrophysical signals offer a lower risk alternative to brain implants for those recovering from injuries. Platforms are now emerging on which researchers and entrepreneurs can build innovative applications without having to worry about the low-level signal processing and integration challenges. Examples of such platforms are Emotive and OpenBCI which offer open-source hardware and software for building BCI applications. OpenBCI's latest product, the OpenBCI Galea, combines BCI with the capabilities of a VR headset. It gives developers access to an array of time-locked physiological data streams along with spatial positioning sensors and eye tracking. This wide range of sensor data can then be used to control a variety of physical and digital devices. The SDK supports a range of languages and makes the sensor data available in Unity or Unreal. We're excited to see this capability offered in an open-source platform so researchers have access to the tools and data they need to innovate in this space.

39. PGLite

Assess

PGLite is a WASM build of a PostgreSQL database. Unlike previous attempts that required a Linux virtual machine, PGLite directly builds PostgreSQL to WASM, allowing you to run it entirely in the web browser. You can either create an ephemeral database in memory or persist it to disk via indexedDB. Since we last mentioned local-first applications in the Radar, the tooling has evolved considerably. With Electric and PGlite, you can now build reactive local-first applications on PostgreSQL.

40. SpinKube

Assess

SpinKube is an open-source serverless run time for WebAssembly on Kubernetes. While Kubernetes offers robust auto-scaling capabilities, the cold start time of containers can still necessitate pre-provisioning for peak loads. We believe WebAssembly's millisecond startup time provides a more dynamic and flexible serverless solution for on-demand workloads. Since our previous discussion of Spin, the WebAssembly ecosystem has made significant advancements. We're excited to highlight SpinKube, a platform that simplifies the development and deployment of WebAssembly-based workloads on Kubernetes.

41. Unblocked

Assess

Unblocked provides software development lifecycle (SDLC) asset and artifact discovery. It integrates with common application lifecycle management (ALM) and collaboration tools to help teams understand codebases and related resources. It improves code comprehension by delivering immediate, relevant context about the code, making it easier to navigate and understand complex systems. Engineering teams can securely and compliantly access discussions, assets and documents related to their work. Unblocked also captures and shares local knowledge that often resides with experienced team members, making valuable insights accessible to everyone, regardless of experience level.

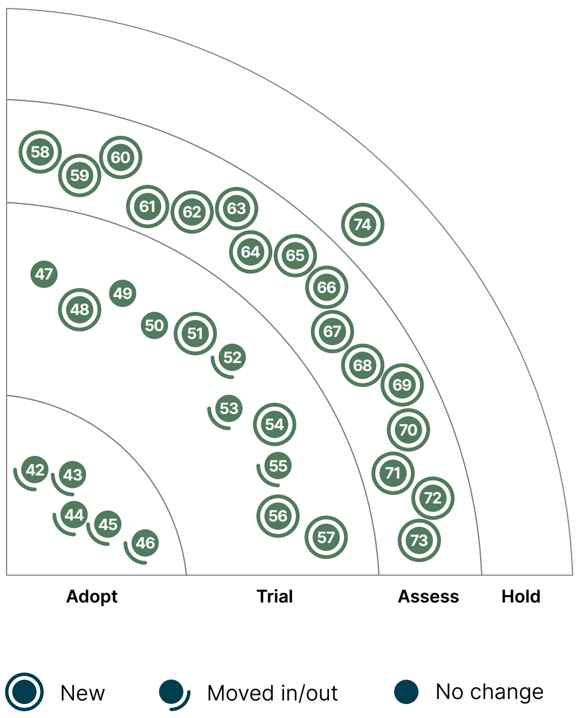
Tools

Adopt
42. Bruno
43. K9s
44. SOPS
45. Visual regression testing tools
46. Wiz

Trial
47. AWS Control Tower
48. CCMenu
49. ClickHouse
50. Devbox
51. Difftastic
52. LinearB
53. pgvector
54. Snapcraft build tool
55. Spinnaker
56. TypeScript OpenAPI
57. Unleash

Assess
58. Astronomer Cosmos
59. ColPali
60. Cursor
61. Data Mesh Manager
62. GitButler
63. JetBrains AI Assistant
64. Mise
65. Mockoon
66. Raycast
67. ReadySet
68. Rspack
69. Semantic Router
70. Software engineering agents
71. uv
72. Warp
73. Zed

Hold
74. CocoaPods



42. Bruno

Adopt

Bruno is an open-source desktop alternative to Postman and Insomnia for API testing, development and debugging. It aims to provide superior collaboration, privacy and security with its simple offline-only design. Collections are stored directly in your filesystem — written in a custom plain text markup language, Bru Lang, and can be shared with Git or a version control tool of your choice to collaborate. Bruno is available both as a desktop app and a CLI tool. It also offers an official VS Code extension, with plans for additional IDE support. Bruno has become the default choice for several Thoughtworks teams, but we also advise teams to be on guard when working under VPN and proxy environments, since requests made in such conditions have been reported to fail unexpectedly.

43. K9s

Adopt

K9s has improved its visualization capabilities by integrating more detailed graphs and views. It now offers better representation of logs and metrics and is more flexible in how it displays custom resources (CRDs). The operations on pods have been expanded and include greater integration with debugging tools (e.g., kubectl debug) and enhanced support for multi-cluster environments. Support for CRDs has significantly improved and now provides better navigation and management of these resources as well as smoother interaction with custom resources. The shortcuts panel has also been enhanced to make it more accessible for developers who are less experienced with kubectl. This is a significant improvement, as K9s initially focused primarily on DevOps teams.

44. SOPS

Adopt

SOPS is an editor of encrypted files that supports various file formats of encrypts with KMS. Our advice when it comes to secrets management has always been to decouple it from source code. However, when faced with a choice between full automation (in the spirit of infrastructure as code) and a few manual steps (using tools like vaults) for managing, seeding and rotating seed secrets, teams often face a tradeoff. For example, our teams use SOPS to manage seed credentials for bootstrapping infrastructure. In some situations, however, it's impossible to remove secrets from legacy code repositories. In those instances, we use SOPS to encrypt secrets in text files. SOPS integrates with cloud-managed keystores such as AWS and GCP Key Management Service (KMS) or Azure Key Vault as sources of encryption keys. It also works cross-platform and supports PGP keys. Several of our teams use SOPS by default when they have to manage secrets in the code repository.

45. Visual regression testing tools

Adopt

We've highlighted visual regression testing tools before and have observed their algorithms evolve from primitive pixel-level comparison to sophisticated pattern-matching and optical character recognition (OCR). Early visual regression tools generated many false positives and were only useful in later stages of development when the interface became stable. BackstopJS avoids this problem by configuring selectors and viewports to pinpoint visual tests to specific elements on the page. But machine learning has made it easier to detect and compare visual elements more accurately, even if they happen to have moved or contain dynamic content. These tools have become steadily more useful and are well-positioned to take advantage of the latest developments in AI and machine learning. Several commercial tools such as Applitools and Percy now claim to use AI in their visual regression tests. One of our teams has been using Applitools Eyes extensively and have been happy with the results. Although visual regression tests are no substitute for well-written end-to-end functional tests, they're a valuable addition to the testing toolbox. We're moving them to adopt because they have become a safe default option as one element in a comprehensive UI test strategy.

46. Wiz

Adopt

Wiz has emerged as the cloud security platform of choice on many of our projects. Our teams appreciate that it enables them to detect risks and threats sooner than similar tools as it continuously scans for changes. Wiz can detect and alert on misconfigurations, vulnerabilities and leaked secrets both in artifacts that have yet to be deployed to live environments (container images, infrastructure code) as well as live workloads (containers, VMs and cloud services). We also appreciate the powerful reporting capability for both development teams and leadership. This analysis helps us understand how a vulnerability can affect a given service so that we can resolve issues in that context.

47. AWS Control Tower

Trial

AWS Control Tower continues to be our go-to choice for managing AWS accounts in a multi-team environment. It provides a convenient mechanism to preconfigure security and compliance controls that will be automatically applied to new landing zones. This is an example of "compliance at the point of change" because the controls are applied and verified whenever new infrastructure is created, eliminating the need for manual compliance checks later on. AWS Control Tower Account Factory for Terraform (AFT) has continued to evolve since our last volume and is now available in more AWS regions. AFT allows Control Tower accounts to be provisioned by an infrastructure-as-code pipeline. We like that AFT can be customized to send webhooks or take specific actions to integrate safely and securely with external tools like GitHub Actions. Our teams have reported great results using AWS Control Tower to manage accounts, but we do wish AWS would accept community contributions to the project when there are opportunities for enhancement.

48. CCMenu

Trial

For teams practicing continuous integration it's important to be aware of the state of the central build on the continuous integration (CI) system. Before the pandemic, dashboards on large TV screens in the team rooms provided this information at a glance. With remote working here to stay, a solution is needed that works on individual developer workstations. For the Mac that niche is covered by CCMenu, a small app written by a Thoughtworker. Originally part of CruiseControl, it works with all servers that can provide information in cctray format, including Jenkins and TeamCity. A recent rewrite has added support for GitHub Actions and paved the way for deeper integration with more CI servers and authentication styles.

49. ClickHouse

Trial

ClickHouse is an open-source, columnar online analytical processing (OLAP) database for real-time analytics. It started as an experimental project in 2009 and has since matured into a highly performant and linearly scalable analytical database. Its efficient query processing engine together with data compression makes it suitable to run interactive queries without pre-aggregation. ClickHouse is also a great storage choice for OpenTelemetry data. Its integration with Jaeger allows you to store massive volumes of traces and analyze them efficiently.

50. Devbox

Trial

Despite advances in development tooling, maintaining consistent local development environments remains a challenge for many teams. Onboarding new engineers often entails running commands or custom scripts that can fail unpredictably across different machines and result in inconsistencies. To solve this challenge, our teams have increasingly relied on Devbox. Devbox is a command-line tool that provides an approachable interface for creating reproducible, per-project local development environments, leveraging the Nix package manager without using virtual machines or containers. It has notably streamlined their onboarding workflow because once it has been configured for a codebase, it takes one CLI command (devbox shell) to reproduce the defined environment on a new device. Devbox supports shell hooks, custom scripts and devcontainer.json generation for integration with VSCode.

51. Difftastic

Trial

Difftastic is a tool for highlighting differences between code files in a syntax-aware way. This is quite different from textual diffing tools, like the venerable Unix diff command. For example, Difftastic will ignore newlines inserted to break up long statements in languages like Java or TypeScript that are semicolon delimited. The tool only highlights changes that impact the syntax of the program. It does this by first parsing the files into abstract syntax trees and then computing the distance between them using Dijkstra's algorithm. We've found Difftastic to be particularly useful for understanding changes when reviewing large codebases. Difftastic can be used on any programming language for which a parser is available and out of the box supports more than 50 programming languages as well as structured text formats like CSS and HTML. This isn't a new tool, but we thought it was worth calling attention to in the age of LLM coding assistants where human-in-the-loop reviews of ever larger codebases are increasingly critical.

52. LinearB

Trial

LinearB, a software engineering intelligence platform, has empowered our engineering leaders with data-driven insights to support continuous improvement. It aligns key areas such as benchmarking, workflow automation and targeted investments in enhancing developer experience and productivity. Our experience with LinearB highlights its ability to foster a culture of improvement and efficiency within engineering teams. Our teams have used the platform to track key engineering metrics, identify areas for enhancement and implement evidence-based actions. These capabilities align well with LinearB's core value proposition: benchmarking, automating metric collection and enabling data-driven improvements. LinearB integrates with source code, application lifecycle, CI/CD and communication tools and uses both preconfigured and custom engineering metrics to provide comprehensive quantitative insights into developer experience, productivity and team performance. As advocates of DORA, we appreciate LinearB's strong emphasis on these specific metrics and its ability to measure key aspects of software delivery performance, which are essential for improving efficiency. Historically, teams have faced challenges in gathering DORA-specific metrics, often relying on complex custom dashboards or manual processes. LinearB continues to offer a compelling solution that automates the tracking of these metrics and delivers real-time data that supports proactive decision-making around developer experience, productivity and predictability.

53. pgvector

Trial

pgvector is an open-source vector similarity search extension for PostgreSQL, allowing the storage of vectors alongside structured data in a single, well-established database. While it lacks some advanced features of specialized vector databases, it benefits from ACID compliance, point-in-time recovery and other robust features of PostgreSQL. With the rise of generative AI-powered applications, we see a growing pattern of storing and efficiently searching embedding vectors for similarities, which pgvector addresses effectively. With pgvector's growing use in production environments, especially where teams are already using a cloud provider that offers managed PostgreSQL, and its proven ability to meet common vector search needs without requiring a separate vector store, we're confident in its potential. Our teams have found it valuable in projects comparing structured and unstructured data, demonstrating its potential for broader adoption, and we're therefore moving it to the Trial ring.

54. Snapcraft build tool

Trial

Snapcraft is an open-source command-line tool for building and packaging self-contained applications called snaps on Ubuntu, other Linux distributions and macOS. Snaps are easy to deploy and maintain across hardware platforms, including Linux machines, virtual environments and vehicle on-board computer systems. While Snapcraft offers a public app store for publishing snaps, our teams use the build tool to package the autonomous driving system as a snap without publishing it to the public app store. This allows us to build, test and debug the embedded software system locally while publishing it to an internal artifact repository.

55. Spinnaker

Trial

Spinnaker is an open-source continuous delivery platform created by Netflix. It implements cluster management and deployment of baked images to the cloud as first-class features. We like Spinnaker's opinionated approach for deploying microservices. In previous editions, we noted its inability to configure pipelines as code, but that has been addressed with the addition of the spin CLI. Even though we don't recommend Spinnaker for simple CD scenarios, it has become a tool of choice for many in complex situations with equally complex deployment pipelines.

56. TypeScript OpenAPI

Trial

TypeScript OpenAPI (or tsoa) is an alternative to Swagger for generating OpenAPI specs from your code. It's code-first, with TypeScript controllers and models as the single source of truth and uses TypeScript annotations or decorators rather than requiring more complex files and configurations when using OpenAPI tooling for TypeScript. It generates both 2.0 and 3.0 API specifications and routes can be generated for Express, Hapi and Koa. If you're writing APIs in TypeScript, this project is worth taking a look at.

57. Unleash

Trial

Although using the simplest feature toggle possible remains our recommended approach, scaling teams and faster development make managing hand-crafted toggles more complex. Unleash is an option widely used by our teams to address this complexity and enable CI/CD. It can be used either as a service or self-hosted. It provides SDKs in several languages with a good developer experience and friendly UI for administration. Although there's no official support for the OpenFeature specification yet, you can find community-maintained providers for Go and Java. Unleash can be used for simple feature toggles as well as segmentation and gradual rollouts, making it a suitable option for feature management at scale.

58. Astronomer Cosmos

Assess

Astronomer Cosmos is an Airflow plugin designed to provide more native support for dbt core workflows in Airflow. With the plugin installed, when DbtDag is used to wrap a dbt workflow, it turns dbt nodes into Airflow tasks/task groups, allowing engineers to visualize dbt dependency graphs and their execution progress directly in the Airflow UI. It also supports using Airflow connections instead of dbt profiles, potentially reducing configuration sprawl. We're experimenting with the tool for its potential to make working with dbt in Airflow more seamless.

59. ColPali

Assess

ColPali is an emerging tool for PDF document retrieval using vision language models, addressing the challenges of building a strong retrieval-augmented generation (RAG) application that can extract data from multimedia documents containing images, diagrams and tables. Unlike traditional methods that rely on text-based embedding or optical character recognition (OCR) techniques, ColPali processes entire PDF pages, leveraging a visual transformer to create embeddings that account for both text and visual content. This holistic approach enables better retrieval as well as reasoning for why certain documents are retrieved, and significantly enhances RAG performance against data-rich PDFs. We've tested ColPali with several clients where it has shown promising results, but the technology is still in the early stages. It's worth assessing, particularly for organizations with complex visual document data.

60. Cursor

Assess

The arms race for AI-assisted programming tools is ongoing, and the most eye-catching one is Cursor. Cursor is an AI-first code editor designed to enhance developer productivity by deeply integrating AI into the coding workflow. We've paid attention to it in previous Radar assessments, but it's clear that the recent continuous improvement of Cursor has ushered in a qualitative change. In our use, Cursor has demonstrated strong contextual reasoning capabilities based on the existing codebase. While other AI code tools like GitHub Copilot tend to generate and collaborate around code snippets, Cursor's multi-line and multi-file editing operations make it stand out. Cursor is forked from VSCode and developed based on it, providing a fast and intuitive interaction method that conforms to the developer's intuition. Powerful operations can be completed with `ctrl/cmd+K` and `ctrl/cmd+L`. Cursor is leading a new round of competition in AI programming tools, especially regarding developer interaction and understanding of codebases.

61. Data Mesh Manager

Assess

Data Mesh Manager provides the metadata layer of a typical data mesh platform. In particular, it focuses on the definition of data products and the specification of data contracts using the OpenContract initiative and can be integrated into build pipelines using the associated DataContract CLI. The application also provides a data catalog to discover and explore data products and their metadata and allows for federated governance, including the definition of data quality metrics and the management of data quality rules. It's one of the first native tools in this space, which means it isn't just trying to retrofit existing platforms to the data mesh paradigm.

62. GitButler

Assess

Despite its power and utility Git's command line interface is notoriously complex when it comes to managing multiple branches and staging commits within them. GitButler is a Git client that provides a graphical interface that aims to simplify this process. It does this by tracking uncommitted file changes independently of Git and then staging those changes into virtual branches. One might argue that this is a solution to a problem that shouldn't exist in the first place; if you make small changes and push to trunk frequently, there's no need for multiple branches. However, when your workflow involves pull requests, the branching structure can become complex, particularly if there is a long review cycle before a PR can be merged. To address this, GitButler also integrates with GitHub so you can selectively group changes into pull requests and issue them directly from the tool. GitButler is another entry into the growing category of blips aimed at managing the complexity inherent in the PR process.

63. JetBrains AI Assistant

Assess

JetBrains AI Assistant is a coding assistant designed to integrate smoothly with all JetBrains IDEs to support code completion, test generation and style guide adherence. Built on models like OpenAI and Google Gemini, it stands out for its ability to ensure consistent output by remembering coding styles for future sessions. Our developers found its test generation capabilities particularly useful and noted its ability to handle longer outputs without stability issues. However, unlike some competitors, JetBrains does not host its own models, which may not work for clients concerned about third-party data handling. Still, the tool's integration with JetBrains IDE makes it a promising choice for teams exploring AI-driven coding assistants.

64. Mise

Assess

Developers working in a polyglot environment often find themselves having to manage multiple versions of different languages and tools. mise aims to solve that problem by providing one tool to replace nvm, pyenv, rbenv and rustup, among others, and is a drop-in replacement for asdf. Mise is written in Rust for shell interaction speed, and unlike asdf which uses shell-based shims, mise modifies the PATH environment variable ahead of time, so the tool run times are called directly. This is partly why mise is faster than asdf. For those developers already familiar with asdf, it provides the same functionality but with a few key differences. Being written in Rust, it's faster and has a few features that asdf doesn't, such as the ability to install multiple versions of the same tool at the same time and more forgiving commands, including fuzzy matching. It also provides an integrated task runner, useful for things like running linters, tests, builders, servers and other tasks that are specific to a project.
If you're a bit fed up with having to use multiple tools to manage your development environment as well as the at times clunky syntax of other tools, mise is definitely worth a look.

65. Mockoon

Assess

Mockoon is an open-source API mocking tool. It has an intuitive interface, customizable routes and dynamic responses as well as the ability to automate the creation of mock data sets. Mockoon is compatible with OpenAPI and lets you generate different scenarios that can be tested locally and integrated with a development pipeline. You can also create "partial mocks" by intercepting the requests and only faking the calls that are defined in Mockoon. The partial mocks help simulate specific API routes or endpoints and forward other requests to actual servers. While partial mocks can be useful in certain scenarios, there is a risk of overuse, which may lead to unnecessary complexity. Besides that, Mockoon remains a valuable tool for quickly setting up mock APIs as well as improving and automating development workflows.

66. Raycast

Assess

Raycast is a macOS freemium launcher that enables you to quickly launch applications, run commands, search for files and automate tasks from your keyboard. Our teams value its out-of-the-box features for developers and its easy extensibility, which allows you to interact with third-party apps and services like VSCode, Slack, Jira and Google, among many others. Raycast is tailored for productivity and minimizes context switching, making it a useful tool for anyone looking to streamline their daily tasks. Pro users have access to Raycast AI, a specialized AI-powered search assistant.

67. ReadySet

Assess

ReadySet is a query cache for MySQL and PostgreSQL. Unlike traditional caching solutions that rely on manual invalidation, ReadySet leverages database replication streams to incrementally update its cache. Through partial view materialization, ReadySet achieves lower tail latencies than a traditional read replica. ReadySet is wire compatible with MySQL and PostgreSQL, so you can deploy it in front of your database to horizontally scale read workloads without requiring application changes.

68. Rspack

Assess

Many of our teams working on web-based frontends have switched from older bundling tools — Webpack comes to mind — to Vite. A new entrant in this field is Rspack, which after 18 months in development has just seen its 1.0 release. Designed as a drop-in replacement for Webpack, it's compatible with plug-ins and loaders in the Webpack ecosystem. This can be an advantage over Vite when migrating complex Webpack setups. The main reason why our teams are migrating to newer tools like Vite and Rspack is developer experience and, in particular, speed. Nothing breaks the flow of development more than having to wait a minute or two before getting feedback on the last code change. Written in Rust, Rspack delivers significantly faster performance than Webpack, and in many cases, it's even faster than Vite.

69. Semantic Router

Assess

When building LLM-based applications, determining a user's intent before routing a request to a specialized agent or invoking a specific flow is critical. Semantic Router acts as a superfast decision-making layer for LLMs and agents, enabling efficient and reliable routing of requests based on semantic meaning. By using vector embeddings to infer intent, Semantic Router reduces unnecessary LLM calls, offering a leaner, cost-effective approach to understanding user intent. Its potential extends beyond intent inference, serving as a versatile building block for various semantic tasks. The speed and flexibility it offers position it as a strong contender in environments that require fast, real-time decision-making without the overhead of LLMs.

70. Software engineering agents

Assess

One of the hottest topics right now in the GenAI space is the concept of software engineering agents. These coding assistance tools do more than just help the engineer with code snippets here and there; they broaden the size of the problem they can solve, ideally autonomously and with minimum interference from a human. The idea is that these tools can take a GitHub issue or a Jira ticket and propose a plan and code changes to implement it, or even create a pull request for a human to review. While this is the next logical step to increase the impact of AI coding assistance, the often advertised goal of generic agents that can cover a broad range of coding tasks is very ambitious, and the current state of tooling is not showing that convincingly yet. However, we can see this working sooner rather than later for a more limited scope of straightforward tasks, freeing up developer time to work on more complex problems. Tools that have been released with beta versions of agents include GitHub Copilot Workspace, qodo flow, Tabnine's agents for JIRA, or Amazon Q Developer. The SWE Bench benchmark lists more tools in that space, but we caution you to take benchmarks in the AI space with a grain of salt.

71. uv

Assess

Rust is well suited for writing command-line tools due to its fast startup performance, and we see people rewriting some toolchains in it. We mentioned Ruff, a Python linter written in Rust in the previous Radar. For this volume, we evaluated uv, a Python package management tool written in Rust. The value proposition of uv is "blazing fast" and it beats other Python package management tools by a large margin in their benchmarks. However, during our Radar evaluation, we discussed whether optimizing within seconds for build tools is a marginal improvement. Compared with performance, what's more important for a package management system is the ecosystem, mature community and long-term support. That being said, feedback from the project team has shown us that this marginal speed improvement could be a big plus for improving feedback cycles and overall developer experience — we tend to manually make CI/CD caching very complex to achieve this little performance improvement; uv simplifies our Python environment management. Considering there is still much room for improvement in package and env management for Python development, we think uv is an option worth assessing.

72. Warp

Assess

Warp is a terminal for macOS and Linux. It splits command outputs into blocks to improve readability. Warp features AI-driven capabilities such as intelligent command suggestions and natural language processing. It also includes notebook features that allow users to organize commands and outputs and to add annotations and documentation. You can leverage these features to create README files or onboarding materials and provide a structured and interactive way to present and manage terminal workflows. Warp easily integrates with Starship, a flexible cross-shell prompt, allowing you to customize the terminal experience and retrieve information about running processes, the specific version of a tool you're using, Git details or the current Git user, among other details.

73. Zed

Assess

After the shutdown of the Atom text editor project, its creators built a new editor named Zed. Written in Rust and optimized to leverage modern hardware, Zed feels fast. It has all the features we expect from a modern editor: support for many programming languages, a built-in terminal and multibuffer editing to name a few. AI-assisted coding is available through integration with several LLM providers. As ardent pair programmers we're intrigued by the remote collaboration feature built into Zed. Developers find each other via their GitHub IDs and can then collaborate on the same workspace in real time. It's too early to tell whether development teams can and want to escape the pull of the Visual Studio Code ecosystem, but Zed is an alternative to explore.

74. CocoaPods

Hold

CocoaPods has been a popular dependency management tool for Swift and Objective-C Cocoa projects. However, the CocoaPods team announced that the project is in maintenance mode after more than a decade of being a key tool for iOS and macOS developers. While the tool and its resources will remain available, active development will cease. Developers are encouraged to transition to Swift Package Manager, which offers native integration with Xcode and better long-term support from Apple.

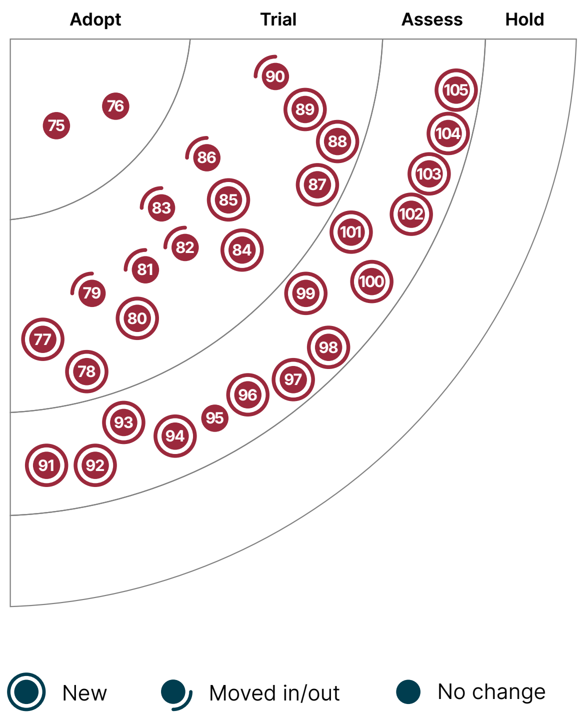
Languages and Frameworks

Adopt
75. dbt
76. Testcontainers

Trial
77. CAP
78. CARLA
79. Databricks Asset Bundles
80. Instructor
81. Kedro
82. LiteLLM
83. LlamaIndex
84. LLM Guardrails
85. Medusa
86. Pkl
87. ROS 2
88. seL4
89. SetFit
90. vLLM

Assess
91. Apache XTable™
92. dbldatagen
93. DeepEval
94. DSPy
95. Flutter for Web
96. kotaemon
97. Lenis
98. LLMLingua
99. Microsoft Autogen
100. Pingora
101. Ragas
102. Score
103. shadcn
104. Slint
105. SST

Hold
—



75. dbt

Adopt

We continue to see dbt as a strong, sensible option for implementing data transformations in ELT pipelines. We like that it lends itself to engineering rigor and enables practices like modularity, testability and reusability of SQL-based transformations. dbt integrates well with many cloud data warehouses, lakehouses and databases — including Snowflake, BigQuery, Redshift, Databricks and Postgres — and has a healthy ecosystem of community packages surrounding it. The native support it recently introduced (in dbt core 1.8+ and the recently introduced dbt Cloud "versionless" experience) for unit testing further strengthens its position in our toolbox. Our teams appreciate that the new unit testing feature allows them to easily define static test data, set up output expectations and test both incremental and full-refresh modes of their pipelines. In many cases, this has allowed them to retire homegrown scripts while maintaining the same level of quality.

76. Testcontainers

Adopt

In our experience, Testcontainers are a useful default option for creating a reliable environment for running tests. It's a library, ported to multiple languages, that Dockerizes common test dependencies — including various types of databases, queuing technologies, cloud services and UI testing dependencies like web browsers — with the ability to run custom Dockerfiles when needed. Recently, a desktop version was released that allows for the visual management of test sessions and the ability to manage more complex scenarios which our teams have found very useful.

77. CAP

Trial

CAP is a .NET library that implements the Outbox pattern. When working with distributed messaging systems like RabbitMQ or Kafka, we frequently face the challenge of ensuring that database updates and event publications are performed atomically. CAP addresses this challenge by recording the intent to publish the event in the same database transaction that caused the event. We find CAP to be quite useful as it supports several databases and messaging platforms while guaranteeing at-least-once delivery.

78. CARLA

Trial

CARLA is an open-source simulator for autonomous driving research used to test autonomous driving systems before production deployment. It offers flexibility in creating and reusing 3D models of vehicles, terrain, humans, animals and more, making it possible to simulate scenarios like a pedestrian stepping onto the street or encountering an oncoming vehicle at a specific speed. The autonomous driving system under test must recognize these dynamic actors and take appropriate action — such as braking. Our teams use CARLA for the ongoing development and testing of autonomous driving systems.

79. Databricks Asset Bundles

Trial

Databricks Asset Bundles (DABs), which reached general availability in April 2024, is becoming the go-to tool for packaging and deploying Databricks assets that facilitates the adoption of software engineering practices in our data teams. DABs supports packaging the configuration of workflows and tasks, as well as the code to be executed in those tasks, as a bundle that can be deployed to multiple environments through CI/CD pipelines. It comes with templates for common types of assets and supports custom templates, which allows for the creation of tailored service templates for data engineering and ML projects. Our teams have increasingly adopted it as a key part of their engineering workflows. Even though DABs includes templates for notebooks and supports deploying them to production, we don't recommend productionizing notebooks and instead encourage intentionally writing production code with the engineering practices that support the maintainability, resiliency and scalability needs of such workloads.

80. Instructor

Trial

When we use large language model (LLM) chatbots as end users, they usually come back to us with an unstructured natural language answer. When building GenAI applications that are more than chatbots, it can be useful to ask the LLM for a structured answer in JSON, YAML or other formats, and then parse and use that response in the application. However, as LLMs are nondeterministic, they might not always do what we ask them to do. Instructor is a library that can be used to help us request structured output from LLMs. You can define the intended output structure and configure retries if the LLM doesn't return the structure you asked for. As the best end user experience for working with LLMs is often to stream the results to them instead of waiting for the full response, Instructor also takes care of parsing partial structures from a stream.

81. Kedro

Trial

Kedro has significantly improved as a tool for MLOps and has maintained its focus on modularity and engineering practices, which we liked from the start. One step that highlights its modularity is the introduction of the standalone kedro-datasets package, which decouples code from data. Kedro has added enhancements in its CLI, starter project templates and telemetry capabilities. Additionally, the recent release of a VS Code extension is a good boost to the developer experience.

82. LiteLLM

Trial

LiteLLM is a library for seamless integration with various large language model (LLM) providers' APIs that standardizes interactions through an OpenAI API format. It supports an extensive array of providers and models and offers a unified interface for completion, embedding and image generation. LiteLLM simplifies integration by translating inputs to match each provider's specific endpoint requirements. It also provides a framework needed to implement many of the operational features needed in a production application such as caching, logging, rate limiting and load balancing. This ensures uniform operation across different LLMs. Our teams are using LiteLLM to make it easier to swap various models in and out — a necessary feature in today's landscape where models are evolving quickly. It's crucial to acknowledge when doing this that model responses to identical prompts vary, indicating that a consistent invocation method alone may not fully optimize completion performance. Also, each model implements add-on features uniquely and a single interface may not suffice for all. For example, one of our teams had difficulty taking advantage of function calling in an AWS Bedrock model while proxying through LiteLLM.

83. LlamaIndex

Trial

LLamaIndex includes engines that enable you to design domain-specific, context-augmented LLM applications and support tasks like data ingestion, vector indexing and natural language question-answering on documents, to name a few. Our teams used LlamaIndex to build a retrieval-augmented generation (RAG) pipeline that automates document ingestion, indexes document embeddings and queries these embeddings based on user input. Using LlamaHub, you can extend or customize LlamaIndex modules to suit your needs and build, for example, LLM applications with your preferred LLMs, embeddings and vector store providers.

84. LLM Guardrails

Trial

LLM Guardrails is a set of guidelines, policies or filters designed to prevent large language models (LLMs) from generating harmful, misleading or irrelevant content. The guardrails can also be used to safeguard LLM applications from malicious users attempting to misuse the system with techniques like input manipulation. They act as a safety net by setting boundaries for the model to process and generate content. There are some emerging frameworks in this space like NeMo Guardrails, Guardrails AI and Aporia Guardrails our teams have been finding useful. We recommend every LLM application have guardrails in place and that its rules and policies be continuously improved. Guardrails are crucial for building responsible and trustworthy LLM chat apps.

85. Medusa

Trial

In our experience, most e-commerce solutions for building shopping websites usually fall into the 80/20 trap — we can easily build 80% of what we want but can't do anything about the remaining 20%. Medusa offers a good balance. It's a highly customizable open-source commerce platform that allows developers to create unique and tailored shopping experiences that can be self-hosted or run on Medusa's platform. Built on Next.js and PostgreSQL, Medusa accelerates the development process with its comprehensive range of modules — from basic shopping cart and order management to advanced features like gift card modules and tax calculation for different regions. We've found Medusa to be a valuable framework and applied it to a few projects.

86. Pkl

Trial

Pkl is an open-source configuration language and tooling initially created for use internally at Apple. Its key feature is its type and validation system, allowing configuration errors to be caught prior to deployment. Pkl has enabled our teams to reduce code duplication (for cases such as environment overrides) and perform validation before configuration changes are applied to live environments. It generates JSON, PLIST, YAML and .properties files and has extensive IDE and language integration, including code generation.

87. ROS 2

Trial

ROS 2 is an open-source framework designed for the development of robotic systems. It provides a set of libraries and tools that enable the modular implementation of applications, covering functions like inter-process communication, multithreaded execution and quality of service. ROS 2 builds on its predecessor by providing improved real-time capabilities, better modularity, increased support for diverse platforms and sensible defaults. ROS 2 is gaining traction in the automotive industry; its node-based architecture and topic-based communication model are especially attractive for manufacturers with complex, evolving in-vehicle applications, such as autonomous driving functionality.

88. seL4

Trial

In software-defined vehicles (SDV) or other safety-critical scenarios, the real-time stability of the operating system is crucial. A few companies monopolize this field due to its high entry barriers, so open-source solutions like seL4 are precious. seL4 is a high-assurance, high-performance operating system microkernel. It uses formal verification methods to "mathematically" ensure the operating system's behavior complies with the specification. Its microkernel architecture also minimizes core responsibilities to ensure system stability. We've seen EV companies like NIO engage with the seL4 ecosystem, and there may be more development in this area in the future.

89. SetFit

Trial

Most of the current crop of AI-based tools are generative — they generate text and images and use generative pre-trained transformers (GPTs) to do so. For use cases that require working with existing text — to classify pieces of text or to determine intent — sentence transformers are the tool of choice. In this field, SetFit is a framework for fine-tuning sentence transformers. We like SetFit because it uses contrastive learning to separate different intent classes from each other, often achieving clear separation with a very small number of examples, even 25 or less. Sentence transformers can also play a role in a generative AI system. We've successfully used SetFit for intent detection in a customer-facing chatbot system that uses an LLM, and even though we're aware of OpenAI's moderation API, we chose a classifier based on SetFit to perform additional fine-tuning to achieve stricter filtering.

90. vLLM

Trial

vLLM is a high-throughput, memory-efficient inference engine for LLMs that can run in the cloud or on-premise. It seamlessly supports multiple model architectures and popular open-source models. Our teams deploy dockerized vLLM workers on GPU platforms like NVIDIA DGX and Intel HPC, hosting models such as Llama 3.1(8B and 70B), Mistral 7B and Llama-SQL for developer coding assistance, knowledge search and natural language database interactions. vLLM is compatible with the OpenAI SDK standard, facilitating consistent model serving. Azure's AI Model Catalog uses a custom inference container to enhance model serving performance, with vLLM as the default inference engine due to its high throughput and efficient memory management. The vLLM framework is emerging as a default for large-scale model deployments.

91. Apache XTable™

Assess

Among the available open table formats that enable lakehouses — such as Apache Iceberg, Delta and Hudi — no clear winner has emerged. Instead, we're seeing tooling to enable interoperability between these formats. Delta UniForm, for example, enables single-directional interoperability by allowing Hudi and Iceberg clients to read Delta tables. Another new entrant to this space is Apache XTable™, an Apache incubator project that facilitates omnidirectional interoperability across Hudi, Delta and Iceberg. Like UniForm, it converts metadata among these formats without creating a copy of the underlying data. XTable could be useful for teams experimenting with multiple table formats. However, for long-term use, given the difference in the features of these formats, relying heavily on omnidirectional interoperability could result in teams only being able to use the "least common denominator" of features.

92. dbldatagen

Assess

Preparing test data for data engineering is a significant challenge. Transferring data from production to test environments can be risky, so teams often rely on fake or synthetic data instead. In this Radar, we explored novel approaches like synthetic data for testing and training models. But most of the time, lower-cost procedural generation is enough. dbldatagen (Databricks Labs Data Generator) is such a tool; it's a Python library for generating synthetic data within the Databricks environment for testing, benchmarking, demoing and many other uses. dbldatagen can generate synthetic data at scale, up to billions of rows within minutes, supporting various scenarios such as multiple tables, change data capture and merge/join operations. It can handle Spark SQL primitive types well, generate ranges and discrete values and apply specified distributions. When creating synthetic data using the Databricks ecosystem, dbldatagen is an option worth evaluating.

93. DeepEval

Assess

DeepEval is an open-source python-based evaluation framework, for evaluating LLM performance. You can use it to evaluate retrieval-augmented generation (RAG) and other kinds of apps built with popular frameworks like LlamaIndex or LangChain, as well as to baseline and benchmark when you're comparing different models for your needs. DeepEval provides a comprehensive suite of metrics and features to assess LLM performance, including hallucination detection, answer relevancy and hyperparameter optimization. It offers integration with pytest and, along with its assertions, you can easily integrate the test suite in a continuous integration (CI) pipeline. If you're working with LLMs, consider trying DeepEval to improve your testing process and ensure the reliability of your applications.

94. DSPy

Assess

Most language model-based applications today rely on prompt templates hand-tuned for specific tasks. DSPy, a framework for developing such applications, takes a different approach that does away with direct prompt engineering. Instead, it introduces higher-level abstractions oriented around program flow (through `modules` that can be layered on top of each other), metrics to optimize towards and data to train or test with. It then optimizes the prompts or weights of the underlying language model based on those defined metrics. The resulting codebase looks much like the training of neural networks with PyTorch. We find the approach it takes refreshing for its different take and think it's worth experimenting with.

95. Flutter for Web

Assess

Flutter is known for its cross-platform support for iOS and Android applications. Now, it has expanded to more platforms. We've evaluated Flutter for Web previously — it allows us to build apps for iOS, Android and the browser from the same codebase. Not every web application makes sense in Flutter, but we think Flutter is particularly suited for cases like progressive web apps, single-page apps and converting existing Flutter mobile apps to the web. Flutter had already supported WebAssembly (WASM) as a compilation target in its experimental channel, which means it was under active development with potential bugs and performance issues. The most recent releases have made it stable. The performance of a Flutter web application compiled to its WASM target is far superior to its JavaScript compilation target. The near-native performance on different platforms is also why many developers initially choose Flutter.

96. kotaemon

Assess

kotaemon is an open-source RAG-based tool and framework for building Q&A apps for knowledge base documents. It can understand multiple document types, including PDF and DOC formats, and provides a web UI, based on Gradio, which allows users to organize and interact with a knowledge base via a chat interface. It has built-in RAG pipelines with a vector store and can be extended with SDKs. kotaemon also cites the source documents in its responses, along with web-based inline previews and a relevance score. For anyone wanting to do a RAG-based document Q&A application, this customizable framework is a very good starting point.

97. Lenis

Assess

Lenis is a lightweight and powerful smooth scrolling library designed for modern browsers. It enables smooth scrolling experiences, such as WebGL scroll syncing and parallax effects, which makes it ideal for teams building pages with fluid, seamless scroll interactions. Our developers found Lenis simple to use, offering a streamlined approach for creating smooth scrolls. However, the library can have issues with accessibility, particularly with vertical and horizontal scrolling interactions, which could confuse users with disabilities. While visually appealing, it needs careful implementation to maintain accessibility.

98. LLMLingua

Assess

LLMLingua enhances LLM efficiency by compressing prompts using a small language model to remove nonessential tokens with minimal performance loss. This approach allows LLMs to maintain reasoning and in-context learning while efficiently processing longer prompts, which addresses challenges like cost efficiency, inference latency and context handling. Compatible with various LLMs without additional training and supporting frameworks like LLamaIndex, LLMLingua is great for optimizing LLM inference performance.

99. Microsoft Autogen

Assess

Microsoft Autogen is an open-source framework that simplifies the creation and orchestration of AI agents, enabling multi-agent collaboration to solve complex tasks. It supports both autonomous and human-in-the-loop workflows, while offering compatibility with a range of large language models LLMs and tools for agent interaction. One of our teams used Autogen for a client to build an AI-powered platform where each agent represented a specific skill, such as code generation, code review or summarizing documentation. The framework enabled the team to create new agents seamlessly and consistently by defining the right model and workflow. They leveraged LlamaIndex to orchestrate workflows, allowing agents to manage tasks like product search and code suggestions efficiently. While Autogen has shown promise, particularly in production environments, concerns about scalability and managing complexity as more agents are added remain. Further assessment is needed to evaluate its long-term viability in scaling agent-based systems.

100. Pingora

Assess

Pingora is a Rust framework to build fast, reliable and programmable network services. Originally developed by Cloudflare to address Nginx's shortcomings, Pingora is already showing great potential, as newer proxies like River are being built on its foundation. While most of us don't face Cloudflare's level of scale, we do encounter scenarios where flexible application-layer routing is essential for our network services. Pingora's architecture allows us to leverage the full power of Rust in these situations without sacrificing security or performance.

101. Ragas

Assess

Ragas is a framework designed to evaluate the performance of retrieval-augmented generation (RAG) pipelines, addressing the challenge of assessing both retrieval and generation components in these systems. It provides structured metrics such as faithfulness, answer relevance and context utilization which help evaluate the effectiveness of RAG-based systems. Our developers found it useful for running periodic evaluations to fine-tune parameters like top-k retrievals and embedding models. Some teams have integrated Ragas into pipelines that run daily, whenever the prompt template or the model changes. While its metrics offer solid insights, we're concerned that the framework may not capture all the nuances and intricate interactions of complex RAG pipelines, and we recommend considering additional evaluation frameworks. Nevertheless, Ragas stands out for its ability to streamline RAG assessment in production environments, offering valuable data-driven improvements.

102. Score

Assess

Many organizations that implement their own internal development platforms tend to create their own platform orchestration systems to enforce organizational standards among developers and their platform hosting teams. However, the basic features of a paved-road deployment platform for hosting container workloads in a safe, consistent and compliant manner are similar from one organization to another. Wouldn't it be nice if we had a shared language for specifying those requirements? Score is showing some promise of becoming a standard in this space. It's a declarative language in the form of YAML that describes how a containerized workload should be deployed and which specific services and parameters it will need to run. Score was originally developed by Humanitec as the configuration language for their Platform Orchestrator product, but it is now under custodianship of the Cloud Native Computing Foundation (CNCF) as an open-source project. With the backing of the CNCF, Score has the potential to be more widely used beyond the Humanitec product. It has been released with two reference implementations: Kubernetes and Docker Compose. The extensibility of Score will hopefully lead to community contributions for other platforms. Score certainly bears a resemblance to the Open Application Model (OAM) specification for Kubevela, but it's more focused on the deployment of container workloads than the entire application. There is also some overlap with SST, but SSI is more concerned with deployment directly into a cloud infrastructure rather than onto an internal engineering platform. We're watching Score with interest as it evolves.

103. shadcn

Assess

shadcn challenges the traditional concept of component libraries by offering reusable, copy-and-paste components that become part of your codebase. This approach gives teams full ownership and control, enabling easier customization and extension — areas where more popular conventional libraries like MUI and Chakra UI often fall short. Built with Radix UI and Tailwind CSS, shadcn integrates seamlessly into any React-based application, which makes it a good fit for projects prioritizing control and extensibility. It includes a CLI to help in the process of copying and pasting the components into your project. Its benefits also include reducing hidden dependencies and avoiding tightly coupled implementations, which is why shadcn is gaining traction as a compelling alternative for teams seeking a more hands-on, adaptable approach to front-end development.

104. Slint

Assess

Slint is a declarative GUI framework for building native user interfaces for Rust, C++ or JavaScript applications. Although it's a multiplatform UI framework with important features such as live preview, responsive UI design, VS Code integration and a native user experience, we particularly want to highlight its usefulness for embedded systems. Teams developing embedded applications have traditionally faced a limited number of options for UI development, each with its own trade-offs. Slint offers the perfect balance between developer experience and performance, using an easy-to-use, HTML-like markup language and compiling directly to machine code. At run time, it also boasts a low-resources footprint, which is critical for embedded systems. In short, we like Slint because it brings proven practices from web and mobile development to the embedded ecosystem.

105. SST

Assess

SST is a framework for deploying applications into a cloud environment along with provisioning all the services that the application needs to run. SST is not just an IaC tool; it's a framework with a TypeScript API that enables you to define your application environment, a service that deploys your application when triggered on a Git push as well as a GUI console to manage the resulting application and invoke the SST management features. Although SST was originally based on AWS Cloud Formation and CDK, its latest version has been implemented on top of Terraform and Pulumi so that, in theory, it's cloud agnostic. SST has native support for deploying several standard web application frameworks, including Next.js and Remix, but also supports headless API applications. SST appears to be in a category of its own. While it bears some resemblance to platform orchestration tools like Kubevela, it also provides developer conveniences like a live mode that proxies AWS Lambda invocations back to a function running on the developer's local machine. Right now, SST remains a bit of a curiosity, but it is a project and part of a category of tools worth watching as it evolves.

Stay up to date with all Radar-related news and insights

Subscribe to the Technology Radar to receive emails every other month for tech insights from Thoughtworks and future Technology Radar releases.

Subscribe now

Thoughtworks is a global technology consultancy that integrates strategy, design and engineering to drive digital innovation. We are 10,500+ people strong across 48 offices in 19 countries. Over the last 30 years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology as the differentiator.

/thoughtworks

Strategy. Design. Engineering.