JUXT AI Radar

An engineer's guide to the AI landscape
from JUXT's CTO & AI chapter members

Q3 2026

Introduction

Frontier models and AI coding assistants have continued to improve at a relentless pace since our last update, and many software engineering teams are now working almost entirely within an agentic coding harness. In our work on agentic AI platforms for large enterprises, we see first-hand the same transition being rolled out to other knowledge workers.

What limits adoption now is the engineering and security work needed to turn their capabilities into reliable human-AI "centaur" systems. Most of this quarter's updates are in this area.

In the Adopt ring, MCP and RAG picked up substantial new security guidance covering authentication, least-privilege tool grants, audit logging and defences against indirect prompt injection. CaMeL enters our radar in Assess as a research-level answer to the same problem. There have been some very public failures this quarter, from Claude Code's own source code being accidentally exposed to agents with broad permissions destroying production data.

The fastest-growing category is also the one we're most cautious about. OpenClaw, NVIDIA's NemoClaw and Moonshot's KimiClaw are persistent agent runtimes built around the same idea: always-on autonomous agents that can control computers. The security models do not hold up to the breadth of access these systems require, and we put all three in Hold. Where they're already in use, sandboxing and human oversight are the only reasonable mitigations.

On the other hand, practices for agentic engineering are beginning to catch up. Spec-driven development enters Adopt in Techniques. In our experience, a structured specification, with guidance on trade-offs, architecture and goals, shapes and constrains an AI implementation more reliably than instructions written in prose. Formal specification languages bring a rigour often lacking from natural-language markdown specifications, and offer opportunities to give fast feedback on ideas and features before any code is written.

Two new Trial entries in Tools stand out, partly because we are seeing strong client demand for both: legacy modernisation and quality hardening. AI-assisted code migration takes the rote work out of porting codebases between languages, frameworks or major versions. AI red teaming tools probe deployed systems for prompt injection, data leakage and toxicity, adversarial testing that used to require a specialist consultancy on retainer. Both are cases of AI helping us produce more robust, better-tested software than most teams could justify writing by hand.

Ownership of models is moving back inside the organisation. The case for running open weight models on your own infrastructure has hardened: EU AI Act compliance requirements and the shift from flat-rate to usage-based AI billing both reward moving inference in-house. Small language models enter Trial in Platforms for the same reasons. Fine-tuned on a few thousand of an organisation's own labelled examples, an SLM can handle classification and extraction work at a fraction of the API cost, on hardware already in place, with data that never leaves the network. For regulated or sensitive workloads, hosted APIs are no longer the only option.

For the latest updates and interactive exploration, visit juxt.pro/ai-radar.

— Henry Garner (CTO, JUXT), June 2026

Radar overview

Our radar visualises the AI landscape across two dimensions: quadrants divide the space by category, while radiants indicate our confidence in each technology.

Quadrants

The radar is divided into four quadrants, each representing a category of AI technology:

Languages & frameworks
The frameworks, libraries, and protocols that underpin AI development. These are the software foundations your applications are built with.

Techniques
Methodologies and practices for building AI systems: approaches such as RAG, prompt engineering, agent design patterns and evaluation methodologies. The "how" of AI development.

Tools
Software that enhances AI development workflows without being embedded in your application code: IDE extensions, CLI utilities, testing frameworks and observability solutions.

Platforms
Infrastructure and managed services that host and run AI workloads: cloud AI services, vector databases, model serving platforms and MLOps infrastructure.

Radiants

Each item is placed on one of four concentric rings representing our recommended adoption stance:

Adopt
Technologies we have high confidence in. Proven in production with strong community support; we'd recommend them for appropriate use cases without hesitation.

Trial
Worth pursuing. These show strong promise and are mature enough to invest time in learning and piloting. Consider them for new projects where you can absorb some risk.

Assess
Worth exploring to understand how they might affect your organisation. Research is warranted, but they're not yet ready for serious investment.

Hold
Proceed with caution. Technologies that are immature, overhyped, or being superseded. Not recommended for new work, though existing usage may be fine.

Contributors

The AI Radar is produced by JUXT's AI Chapter, drawing on our experience following industry developments and working with clients across sectors. The radar represents our current viewpoint, subject to change as the technology landscape evolves. We welcome feedback via LinkedIn, BlueSky, or email.

Henry Garner

Henry is JUXT's CTO and leader of the AI Chapter. He's implemented AI systems in domains as diverse as education, financial services, and local government in roles spanning data scientist, software engineer and CTO. He's author of the book Clojure for Data Science and maintainer of the open source statistics library kixi.stats.

Ben Halton

Ben is an account manager at JUXT with extensive experience engineering and architecting complex systems. His career spans domains from risk systems in Tier 1 banks to retail recommendation engines and wine trading platforms. His interest in AI is especially in how it can enhance developer experience and productivity.

Denis Lobanov

Denis is a software engineer at JUXT whose technical experience spans developing Linux kernel modules for Satellite communications to distributed graph databases to web backends. He is currently focused on providing a platform that integrates, controls and secures LLM interaction.

Oliver Marshall

Oliver is a software engineer at JUXT who specialises in building data processing systems and backend infrastructure. He's recently worked on integrating cutting-edge database technology for a client and approaches AI technologies with healthy skepticism: hopeful about what's possible but focused on what actually works in practice.

Neale Swinnerton

Neale is a principal engineer at JUXT. He's spent his career in software development across many domains. He sees himself as an engineer more than a scientist, advising teams how to use pragmatic workflows to improve developer productivity and joy.

Chris Williams

Chris is a software engineer at JUXT with broad experience across industries and technologies. He has long seen automated testing as a superpower for building reliable systems, and now views large language models as the next tool for boosting productivity while supporting real learning.

We're grateful to **Rhi Hanger**, whose insights informed our ontology coverage, **Gio Kiladze**, who authored the process mining section, and **Lucio D'Alessandro**, who built the tooling that generates this publication.

Radar at a glance

Languages & frameworks

ADOPT

1. PyTorch
2. dbt
3. MCP

TRIAL

4. Microsoft Agent Framework
5. A2A
6. LLM testing frameworks
7. LlamaIndex
8. LangChain & LangGraph
9. Formal specification languages

ASSESS

10. Prolog
11. JAX
12. OpenAI AgentKit
13. PydanticAI
14. Smolagents
15. CrewAI
16. DSPy
17. LinkML

HOLD

18. TensorFlow

Techniques

ADOPT

19. Classical ML
20. RAG
21. LLM-as-a-judge
22. BERT variants
23. Few-shot prompting
24. Agentic tool use
25. Spec-driven development

TRIAL

26. Cross-encoder reranking
27. Ontologies for AI grounding
28. Model distillation & synthetic data
29. UMAP
30. Claude Skills
31. Structured RAG

ASSESS

32. Neurosymbolic AI
33. World models
34. LLM reproducibility
35. Hypothetical document embeddings (HyDE)
36. Fine-tuning with LoRA
37. Physical AI and robotics foundation models
38. CaMeL

HOLD

39. Chain of thought (CoT)
40. AI pull request review

Radar at a glance

Tools

ADOPT

41. Software engineering copilots
42. Provider-agnostic LLM facades
43. Notebooks

TRIAL

44. MLflow
45. Vector databases
46. Local model execution environments
47. LLM observability tools
48. LLM API gateways
49. AI red teaming tools
50. AI-assisted code migration

ASSESS

51. AI application bootstrappers
52. Visual computer use agents
53. Lakera
54. Structured output libraries

HOLD

55. OpenClaw
56. Conversational data analysis

Platforms

ADOPT

57. Foundation models
58. Weights & Biases
59. Temporal
60. Data pipeline orchestration tools
61. Cloud model hosting platforms

TRIAL

62. Production AI monitoring platforms
63. Open weight LLMs
64. Small language models
65. AI-powered workflow automation platforms
66. Digital twin platforms

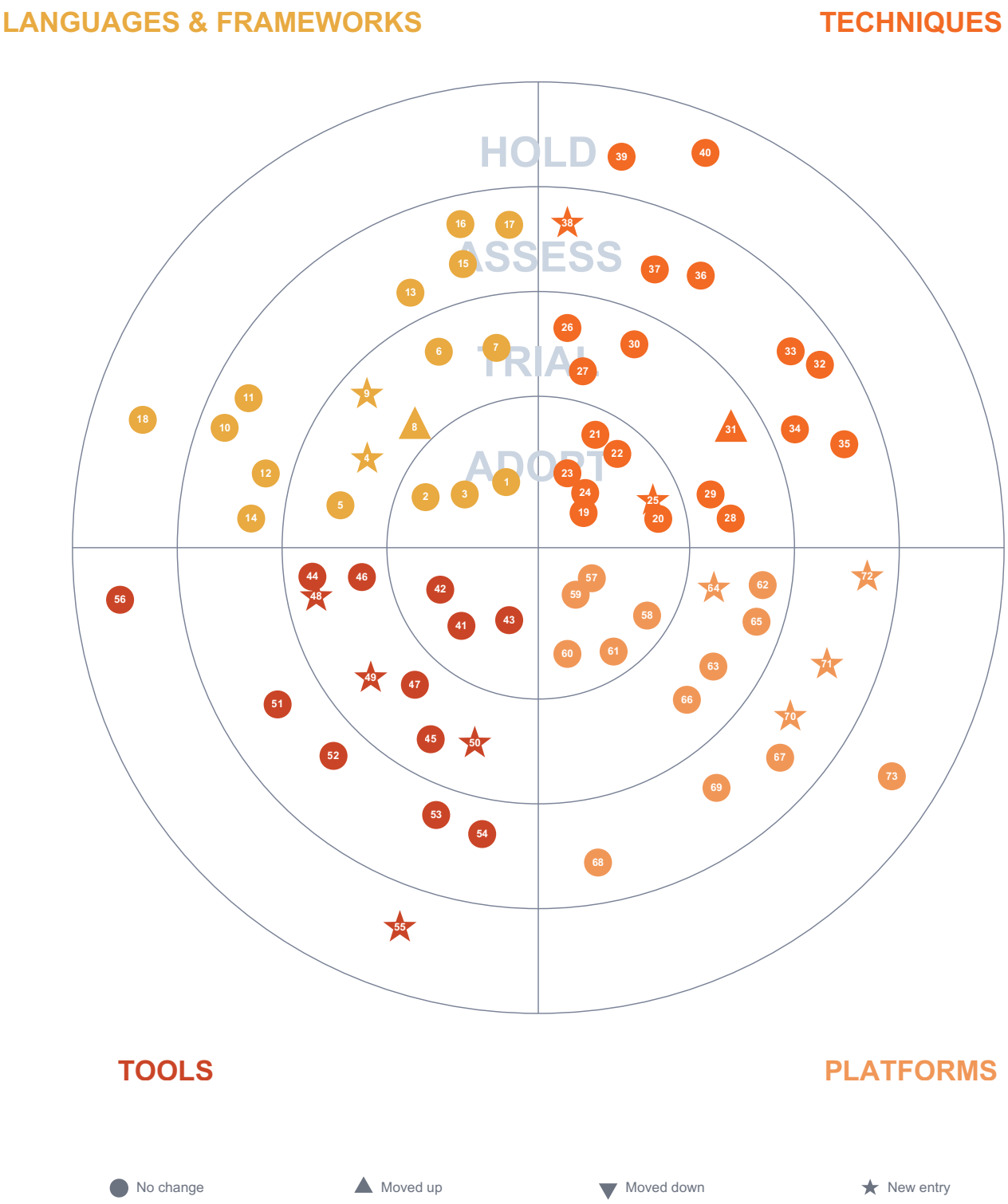
ASSESS

67. Galileo
68. Kubeflow
69. Process mining platforms
70. Taalas
71. AI governance platforms
72. Agent memory architectures

HOLD

73. Building against vendor-specific APIs

The Radar


LANGUAGES & FRAMEWORKS
TECHNIQUES
HOLD
ASSESS
TRIAL
ADOPT
TOOLS
PLATFORMS
No change
Moved up
Moved down
New entry

Languages & frameworks

The frameworks, libraries, and protocols that underpin AI development. These are the software foundations your applications are built with.

Adopt

1. PyTorch
2. dbt
3. MCP

Trial

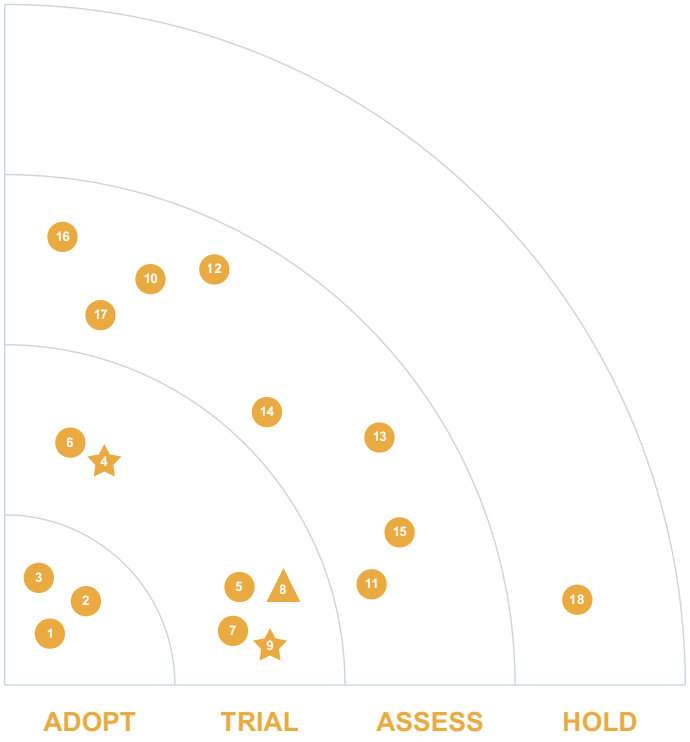
4. Microsoft Agent Framework
5. A2A
6. LLM testing frameworks
7. LlamaIndex
8. LangChain & LangGraph
9. Formal specification languages

Assess

10. Prolog
11. JAX
12. OpenAI AgentKit
13. PydanticAI
14. Smolagents
15. CrewAI
16. DSPy
17. LinkML

Hold

18. TensorFlow



Adopt

Mature, well-supported technologies ready for production use.

PyTorch

PyTorch has demonstrated consistent maturity and widespread adoption across both research and production environments, which is why we place it in our Adopt ring. We're seeing it emerge as the default choice for many machine learning teams, particularly those working on deep learning projects, thanks to its intuitive Python-first approach and dynamic computational graphs that make debugging and prototyping significantly easier.

The framework's robust ecosystem, exceptional documentation and strong community support make it a reliable choice for teams at any scale. While TensorFlow remains relevant, particularly in production deployments, PyTorch's seamless integration with popular machine learning tools, extensive pre-trained model repository and growing deployment options through TorchServe have addressed previous concerns about production readiness. The framework's adoption by major technology organisations and research institutions, coupled with its regular release cycle and stability, gives us confidence in recommending it as a default choice for new machine learning projects.

dbt

We've placed dbt (data build tool) in the Adopt ring because it has proven to be an essential framework for organising and managing the data transformations that feed AI systems. dbt brings software engineering best practices such as version control and testing to data transformation workflows, which is crucial when preparing data for AI model training and inference.

The reliability and maintainability of AI systems heavily depend on the quality of their input data, and dbt helps teams achieve this by making data transformations more transparent and trustworthy. We've seen teams successfully use dbt to create clean, well-documented data pipelines that connect data warehouses to AI applications, while maintaining the agility to quickly adapt to changing requirements. Its integration with modern data platforms and strong community support make it a solid choice for organisations building out their AI infrastructure.

MCP

Anthropic's Model Context Protocol (MCP) has rapidly gained adoption, addressing the need for standardised integration between language models and external tools. MCP solves the common problem of connecting AI models to organisational data without requiring custom integration work for each connection. MCP servers are straightforward to create, and the growing ecosystem of community-created servers reduces development overhead further.

Since our last radar, we've seen rapid uptake within organisations. Some are pursuing ambitious goals of making all internal APIs AI-accessible via MCP servers, creating a unified interface through which AI assistants interact with enterprise systems. The implementation investment for that level of coverage is easy to underestimate.

Security needs to be a first-class concern. Every MCP server must enforce authentication and authorisation independently of the calling model, with tool grants following the principle of least privilege. Audit logging of all tool calls is essential for traceability. MCP servers returning untrusted data can become an indirect prompt injection vector, so outputs from external sources need careful sanitisation before being fed back into model context.

A broader architectural concern surfaced in April 2026 when OX Security demonstrated that MCP's STDIO transport executes arbitrary commands without validation, leading to multiple CVEs across downstream projects. More concerning was their proof of concept showing malicious entries accepted by most MCP registries. Treat MCP servers sourced from public marketplaces with the same caution as any third-party dependency: review the code, pin versions and run servers with minimal permissions.

For simpler workflows that operate on local files and code rather than external services, Claude Skills offer a lighter-weight alternative worth considering before committing to MCP server development.

See also: Claude Skills, Agentic tool use.

Trial

Promising technologies with growing adoption that are worth exploring in production-adjacent settings.

Microsoft Agent Framework

Microsoft Agent Framework (MAF) launched in October 2025, merging AutoGen's agent abstractions with Semantic Kernel's enterprise features into a single open-source platform. MAF supports Python and .NET, offering graph-based workflows for multi-agent orchestration alongside session-based state management and telemetry.

We've placed it in Trial rather than Adopt because the framework is still reaching stability. That said, it consolidates Microsoft's previously fragmented agent story, and organisations building multi-agent systems on Microsoft's stack should evaluate it as their primary framework.

As with any agent framework granting tool access, apply least-privilege scoping and build in prompt injection awareness from the start.

See also: Agentic tool use, MCP, A2A.

A2A

Google's Agent2Agent (A2A) protocol addresses the need for standardised communication between AI agents. Launched in April 2025 and now governed by the Linux Foundation, A2A enables agents from different providers to discover capabilities and collaborate without custom integration.

The protocol complements MCP rather than competing with it. MCP connects models to tools and data sources; A2A handles agent-to-agent communication. The design centres on "Agent Cards" that advertise capabilities in JSON format, enabling dynamic task delegation. The protocol supports text and video streaming with built-in security features for enterprise deployment.

We've placed A2A in Trial because it remains relatively new with limited production deployment patterns. The protocol's complexity is justified when agents need to collaborate across runtime boundaries: different vendors, different stacks or different organisations.

Inside a single runtime, simpler options usually suffice. A single orchestrating agent can call tools directly through MCP. A workflow engine such as Temporal or an AI-powered workflow automation platform coordinates LLM steps deterministically. A multi-agent framework such as LangGraph or Microsoft Agent Framework handles orchestration in-process. Reach for A2A when none of these fit your topology.

LLM testing frameworks

DeepEval provides a systematic framework for evaluating LLM outputs, with built-in metrics for relevance, factual accuracy, hallucination detection and toxicity. It integrates with pytest, making it accessible to teams familiar with Python testing workflows.

Promptfoo takes a CLI-first approach with strong CI/CD integration. Where DeepEval is a Python library you embed in your test suite, Promptfoo runs as a standalone tool comparing outputs across models and prompt variants. OpenAI acquired Promptfoo in March 2026, though it remains open-source under the MIT licence. Teams running multi-model strategies should consider whether the acquisition affects their comfort with Promptfoo as a neutral evaluation tool.

DeepEval suits teams that want evaluation embedded in their Python test suite. Promptfoo suits teams wanting standalone prompt regression testing, particularly those with CI/CD pipelines that gate deployments on evaluation results.

See also: LLM-as-a-judge, AI red teaming tools.

LlamaIndex

LlamaIndex, formerly GPT Index, started as a framework for indexing data so that LLMs could retrieve it efficiently. It has matured considerably since then, with production deployments across Fortune 500 organisations and a Workflows 1.0 release that adds a lightweight agentic orchestration layer alongside the retrieval core.

We keep it in Trial because LlamaIndex's strongest case is narrower than its scope suggests: retrieval-heavy systems that benefit from its 100+ data connectors and fine-grained indexing controls. For RAG over large corpora or enterprise knowledge bases, it is the framework we would recommend looking at first. For general LLM orchestration, LangChain & LangGraph or Microsoft Agent Framework tend to be a better fit.

LangChain & LangGraph

LangChain and its companion LangGraph move up to Trial this quarter. LangGraph 1.0 reached stable release, addressing earlier concerns about abstraction churn and giving teams a more reliable foundation for building multi-step LLM workflows.

LangChain handles general-purpose LLM interactions while LangGraph extends this to stateful, graph-based agent workflows. The rapid pace of change in the underlying AI platforms means that some of LangChain's abstractions may still become less relevant as the ecosystem evolves, so we recommend focused experiments that test whether these tools simplify your specific use case.

See also: Microsoft Agent Framework, AutoGen.

Formal specification languages

Formal specification languages allow teams to describe system behaviour with enough precision that properties can be verified before code is written. They sit on a spectrum: lightweight languages that structure intent, model checkers that explore state spaces and full theorem provers that deliver mathematical proof. AI assistants have lowered the barrier to entry, making formal specification viable for a broader range of software than the safety-critical systems that historically justified the investment.

TLA+, created by Leslie Lamport, is the most widely adopted formal specification language for distributed systems. Amazon used it to find subtle bugs in AWS infrastructure that testing alone could not surface. Alloy, developed by Daniel Jackson at MIT, takes a lighter approach with automatic analysis, well suited for exploring design spaces and finding counterexamples early. FizzBee offers a more accessible alternative to TLA+ designed for practitioners. On the verification-aware end of the spectrum, Dafny and Lean 4 have both seen substantial AI-assisted activity over the past year, with dedicated POPL workshops and research projects targeting them as LLM proof-synthesis targets.

We've been developing Allium, our own specification language at the practical end of this spectrum. Allium captures system behaviour in a structured, machine-readable format that AI agents can use to guide implementation and generate tests. It cannot prove properties hold across all possible states the way TLA+ or Alloy can, but it captures intent precisely enough to serve as contracts between humans and AI. The right level of formality depends on what is at stake.

AI has shifted the economics. Specifying behaviour rigorously used to be reserved for systems where the cost of failure was high enough to justify the upfront effort. With AI agents now both authoring specifications and producing implementations against them, behavioural specification is more cost-effective than it was, and arguably more necessary, given how readily models resolve ambiguity in directions you did not intend.

See also: Spec-driven development, Neurosymbolic AI, Prolog.

Assess

Emerging or specialised technologies that merit evaluation for specific use cases.

Prolog

Prolog sits in Assess due to its renewed relevance for neurosymbolic AI architectures. This decades-old logic programming language offers something LLMs fundamentally lack: guaranteed logical inference with explainable reasoning chains.

LLMs excel at understanding natural language but cannot reliably follow complex rules or explain why they reached a conclusion. Prolog does exactly this. By coupling an LLM with a Prolog reasoning engine, teams can build systems where the LLM handles ambiguous input and Prolog enforces business logic, validates conclusions or traverses knowledge graphs. Implementations typically use Prolog to represent domain rules that validate LLM outputs before they reach users. This pattern is particularly valuable in regulated industries where decisions must be auditable.

We've kept Prolog in Assess because the tooling ecosystem for LLM integration remains immature and performance can be challenging at scale. Teams should also consider whether semantic web technologies (RDF, OWL, SPARQL) might serve similar purposes with better tooling support.

See also: Neurosymbolic AI, Ontologies for AI grounding.

JAX

JAX sits in our Assess ring as we observe increasing interest in this ML framework that combines NumPy's familiar API with hardware acceleration and automatic differentiation. While TensorFlow and PyTorch remain dominant in the ML ecosystem, we're seeing JAX gain traction particularly in research settings and among teams working on custom ML architectures.

JAX's functional approach to ML computation and its ability to compile to multiple hardware targets through XLA (Accelerated Linear Algebra) set it apart from more established frameworks. It shows promise for projects requiring high-performance numerical computing, though teams should weigh its relative immaturity in deployment tooling and a smaller ecosystem of pre-built components. We recommend teams experimenting with JAX do so on research projects or contained proofs-of-concept before considering broader adoption.

OpenAI AgentKit

OpenAI launched AgentKit at DevDay in October 2025, comprising Agent Builder for visual workflow design, ChatKit for embeddable interfaces, integrated evals and a Connector Registry for tool integration. ChatKit and the evals capabilities are generally available; Agent Builder and the Connector Registry are still in beta.

AgentKit sits in Assess because adopting it represents a substantial commitment to the OpenAI ecosystem. Unlike framework-agnostic alternatives such as LangChain or Microsoft Agent Framework, teams using AgentKit tie their agent infrastructure to a single provider's roadmap. There is no separate AgentKit fee, since usage rolls into standard API pricing, but agentic workloads consume tokens unpredictably and the cost trajectory needs modelling up front.

For organisations already invested in OpenAI's platform, AgentKit offers a streamlined path from prototype to production. Teams that need vendor flexibility will want to evaluate open alternatives first.

PydanticAI

PydanticAI brings the developer experience of FastAPI to generative AI application development. It is built by the team behind Pydantic, the validation layer that underpins the OpenAI, Anthropic and Google ADK SDKs alongside LangChain, LlamaIndex and others. It offers model-agnostic LLM support, structured responses through Pydantic validation and a dependency injection system for testing.

PydanticAI reached v1 in September 2025 and has since added durable execution, streaming structured outputs and graph-based control flow. It uses existing Python patterns rather than introducing new paradigms, which makes it immediately approachable for teams already familiar with the ecosystem. Production deployments are appearing, and for Python-first stacks it is a strong default to evaluate.

Smolagents

Smolagents is Hugging Face's minimalist agent framework. The core fits inside a thousand lines of code. Its distinguishing feature is the code agent pattern: rather than calling tools through structured tool-call protocols, agents write Python snippets that run against the tool catalogue. This typically reduces LLM calls per task by around a third and produces more transparent reasoning traces.

The natural concern is that letting an agent run arbitrary Python is a security risk. Smolagents addresses this through sandboxed execution backends, including Blaxel, E2B, Modal and Docker, which the framework expects to be enabled outside local development. We keep it in Assess because adoption outside Hugging Face's own examples is still limited, and the code-agent pattern is more opinionated than standard tool-calling. For teams in the Hugging Face stack who want a lighter alternative to LangGraph or Microsoft Agent Framework, it is worth a look.

CrewAI

CrewAI is a framework for orchestrating teams of specialised agents that collaborate through defined roles and task delegation. It supports human-in-the-loop integration and has become one of the most-adopted multi-agent frameworks, with substantial enterprise use and a dedicated production architecture called CrewAI Flows.

CrewAI is now the de facto choice for many organisations building agent collaboration. Best practices for multi-agent design are still emerging across the industry, and managing several agents adds operational overhead that a single orchestrating agent often avoids. Worth evaluating where the work benefits from multiple specialised agents rather than one capable one.

DSPy

DSPy treats prompts as optimisable programs rather than handcrafted text. Developed at Stanford, developers define signatures (input-output specifications) and modules (composable building blocks), and DSPy's optimisers automatically generate effective prompts based on example data. The optimisation process can discover strategies that humans might not have considered.

The framework shows particular promise for complex pipelines involving multiple LLM calls or retrieval steps. DSPy remains in Assess because it introduces a programming paradigm unfamiliar to most teams, and the investment pays off only for systems that benefit from automated prompt optimisation. For simpler single-prompt applications, traditional approaches may remain more practical.

LinkML

LinkML allows teams to define data models in YAML and generates multiple outputs: JSON Schema for validation, Python dataclasses for code, RDF/OWL for semantic web compatibility and documentation. This makes it valuable for phased ontology development where teams want to start practically but preserve the option for formalisation later.

The framework emerged from biomedical informatics but applies broadly. For AI applications, LinkML models can define entities and relationships for knowledge graphs and structured output schemas for LLMs. It remains in Assess because adoption is relatively niche. Organisations already committed to JSON Schema may find less incremental value, but for teams starting fresh on knowledge representation, LinkML offers a middle path between ad-hoc schemas and full OWL modelling.

Hold

Not recommended for new projects; better alternatives exist.

TensorFlow

We have placed TensorFlow in the Hold ring for several reasons. While TensorFlow remains a capable deep learning framework that helped popularise machine learning at scale, we're seeing teams struggle with its larger API surface and fragmented deployment story compared to more modern alternatives. The framework's syntax and intricate architecture could act as headwinds for teams new to machine learning.

PyTorch has emerged as the clear community favourite for both research and production deployments, with arguably a more intuitive programming model and better debugging capabilities. For new projects we recommend exploring higher-level tools or PyTorch unless there are compelling reasons to use TensorFlow, such as maintaining existing deployments or specific requirements around TensorFlow Extended (TFX) for ML pipelines.

Techniques

Methodologies and practices for building AI systems: approaches such as RAG, prompt engineering, agent design patterns and evaluation. The "how" of AI development.

Adopt

19. Classical ML
20. RAG
21. LLM-as-a-judge
22. BERT variants
23. Few-shot prompting
24. Agentic tool use
25. Spec-driven development

Trial

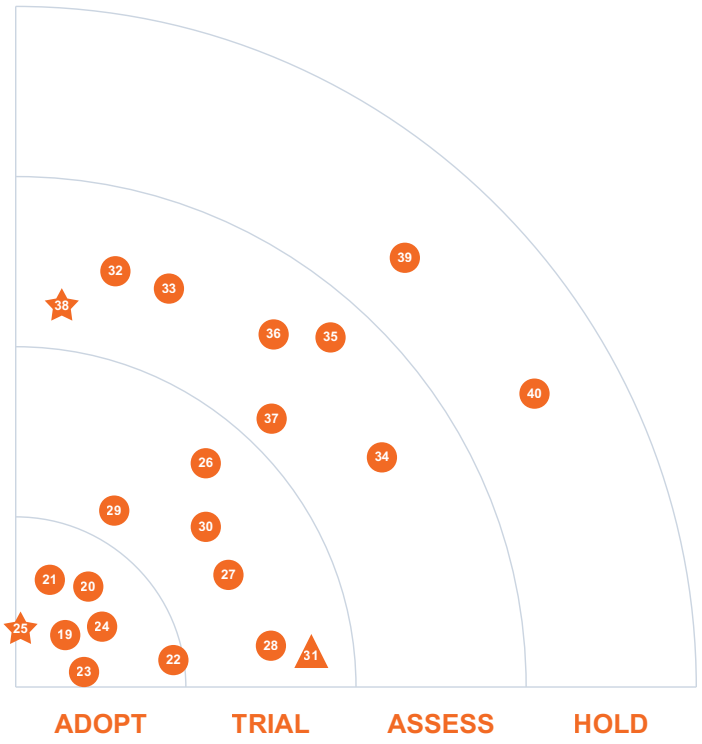
26. Cross-encoder reranking
27. Ontologies for AI grounding
28. Model distillation & synthetic data
29. UMAP
30. Claude Skills
31. Structured RAG

Assess

32. Neurosymbolic AI
33. World models
34. LLM reproducibility
35. Hypothetical document embeddings (HyDE)
36. Fine-tuning with LoRA
37. Physical AI and robotics foundation models
38. CaMeL

Hold

39. Chain of thought (CoT)
40. AI pull request review



Adopt

Mature, well-supported approaches ready for production use.

Classical ML

Classical machine learning approaches such as random forests, gradient boosting (XGBoost, LightGBM), linear/logistic regression and support vector machines remain the best balance of explainability and efficiency for structured data problems. These techniques routinely outperform more complex approaches on tabular data while training faster and costing less to run.

Realising these benefits requires quality training data and staff with appropriate expertise. Unlike LLM-based solutions that have democratised AI for organisations without data science teams, classical ML demands specialised knowledge in feature engineering and model selection. For organisations with the necessary capabilities, these methods work well even with smaller enterprise datasets, matching or exceeding the performance of more complex approaches while remaining more interpretable and easier to maintain.

RAG

Retrieval-Augmented Generation (RAG) combines search and text generation to produce more accurate responses, grounding them in real data and reducing confabulation. It is valuable where accuracy and traceability matter, such as customer service or compliance. Implementation requires attention to document processing and embedding strategies, but tooling has lowered the barriers.

We're watching techniques such as Self-RAG, which prompts the model to gather more evidence or refine its responses; early results suggest it reduces confabulations further.

RAG introduces an indirect prompt injection attack surface. Retrieved documents are injected into model context, so adversarial content in any document reaches the prompt via the retrieval step. Retrieval access controls and provenance tracking for ingested documents help mitigate this risk.

See also: Cross-encoder reranking, Structured RAG, Hypothetical document embeddings (HyDE).

LLM-as-a-judge

LLM-as-a-judge has proven one of the most practical techniques for evaluating AI system outputs. Today's strongest models provide nuanced, multidimensional critique that simpler evaluation methods cannot match, except for very constrained metrics such as exact match or BLEU scores.

The technique is widely adopted in both offline and online evaluation. Offline, it scales far better than human assessment, allowing teams to test thousands of outputs quickly. Online, an LLM judge can evaluate another LLM's output in real-time, enabling dynamic workflow adjustments based on quality assessments.

Research demonstrates that frontier models provide judgements correlating strongly with human preferences across many evaluation dimensions. We recommend using a different LLM as the judge than the one being evaluated, and viewing this as an augmentation to human evaluation rather than a replacement. The strongest LLMs can identify nuanced issues in reasoning and factuality that would otherwise require substantial human review time.

See also: DeepEval.

BERT variants

Bidirectional Encoder Representations from Transformers (BERT) revolutionised NLP by processing words in relation to their entire context rather than sequentially. The family has continued to evolve, with ModernBERT the latest iteration, improving training times and accuracy through architectural updates.

BERT-style models serve a different purpose from generative models. Where GPT generates text, BERT models are optimised for understanding tasks such as classification and sentiment analysis. They are also the basis for the semantic vector embeddings that RAG systems use to retrieve relevant context for generative models.

We recommend DeBERTa for new NLP projects, as it handles word relationships more effectively using a disentangled attention mechanism. DistilBERT is smaller and faster whilst retaining most performance, valuable for production deployments with strict latency requirements. Domain-specific variants exist for biomedical (BioBERT) and financial text (FinBERT), though these require expertise to use effectively.

Few-shot prompting

Providing examples to guide model responses has proven consistently effective across Large Language Models.

This is shifting. As models become more capable, interactive multi-turn approaches are gaining favour: rather than providing examples upfront, practitioners prompt models to ask clarifying questions and iterate toward a solution. The pattern often produces better results, particularly in agentic workflows where the model can refine its approach.

Few-shot prompting retains an important role in non-interactive contexts. System prompts and automated pipelines do not afford clarifying dialogue, and well-chosen examples remain the most effective way to establish output format and domain conventions. We typically see diminishing returns beyond 3-5 examples, with token consumption as the main trade-off.

Agentic tool use

We've moved agentic tool use to the Adopt ring for local, sandboxed environments. AI coding assistants that can edit files, run tests, execute shell commands and perform web searches deliver considerably more value than those limited to conversation.

The ecosystem has matured to support this. Standards such as MCP and OpenAI's Function Calling provide reliable integration patterns, while improved observability tooling lets teams monitor what agents are doing. The Development Containers specification makes it straightforward to isolate agent execution.

The risks magnify for applications accepting external user input. Prompt injection attacks remain an unsolved problem. An agent that safely edits files for a developer becomes a liability when processing untrusted input. Our recommendation: adopt for local developer tooling and internal workflows, but proceed with caution for customer-facing systems, treating each tool permission as a potential attack vector.

See also: Visual computer use agents, Model Context Protocol, Temporal.

Spec-driven development

Software engineers working agentically are finding that natural-language prompts are not enough to constrain the behaviour of an AI system. Prompts carry ambiguities that the model resolves silently, and the implementation that emerges may differ from the one the engineer thought they had asked for. Spec-driven development has gained popularity as the response: an agentic engineering practice where a written specification, rather than a conversation, becomes the contract the AI works against.

The practice has moved into the mainstream of agentic engineering over the past year. GitHub Spec Kit and Amazon's Kiro are the most visible tools: Spec Kit is agent-agnostic, while Kiro ships a dedicated specs mode that generates EARS notation by default. Both treat structured markdown as the source of truth, with code generated or verified against it. A parallel revival of Gherkin and behaviour-driven development sits in the same middle ground between prose and formal language.

The direction we find most exciting goes further. A specification written in natural-language markdown, even one structured in EARS or Gherkin form, only moves the ambiguity problem one step back. The prose may be richer than a prompt, but it still relies on the model to disambiguate it. We are following formal specification languages instead, including our own Allium, an open-source language that captures system behaviour in a structured, machine-readable form. Allium is early in its life and external uptake remains nascent, but in our own work it gives AI agents an unambiguous reference and lets engineers detect contradictions before any code is written.

See also: Formal specification languages, Claude Skills.

Trial

Promising approaches with growing adoption, worth exploring for teams ready to invest in emerging patterns.

Cross-encoder reranking

Cross-encoder reranking enhances AI search and chat systems by examining initial search results more carefully. While embedding search is fast and good at finding broadly relevant content, cross-encoder reranking excels at understanding subtle relevance signals by examining the query and potential results together.

Most teams use a two-step process: embedding search finds 50-100 potentially relevant items, then cross-encoder reranking sorts these candidates to surface the most relevant. The technique often reduces confabulations in downstream LLM responses by ensuring higher quality context selection. Implementation has become straightforward with libraries such as sentence-transformers providing ready-to-use models. Teams should be mindful of the additional latency and may need to tune the number of candidates based on performance requirements.

Ontologies for AI grounding

As AI systems scale beyond isolated experiments, shared meaning becomes critical infrastructure. Ontologies provide what LLMs lack: authoritative definitions of entities and relationships that don't shift with statistical probability. They ground responses in agreed definitions, enable knowledge graph traversal that pure RAG cannot achieve and support the structured outputs that agentic systems require.

Traditional ontology development tends toward two failure modes: academic approaches aiming for formal completeness using OWL, and pragmatic approaches creating spreadsheets that grow unmaintainable. The key is to start lightweight and formalise selectively. Mark Burgess argues that traditional ontologies impose rigid hierarchies that don't match how language models represent meaning, proposing alternative graph structures designed to work with vector embeddings. For organisations needing to ground AI in domain knowledge today, ontologies offer a practical path with mature tooling.

Graph databases such as Neo4j provide accessible implementation options, while LinkML offers YAML-based modelling without deep ontology expertise. Start with a painful, high-value domain rather than attempting to model the entire organisation.

See also: LinkML, Neurosymbolic AI, Prolog.

Model distillation & synthetic data

Model distillation involves training a smaller, more efficient model to mimic a larger one. A common pattern uses LLMs to generate synthetic training data for the smaller model: the large LLM acts as a "teacher", creating diverse examples that help the "student" learn desired behaviour. This makes AI deployment more practical for edge devices or resource-constrained environments.

We're keeping it in Trial because the process requires considerable expertise. Teams need to validate the quality of generated training data and ensure the distilled model maintains acceptable performance. There is ongoing debate about amplification of biases through this approach.

Check the licence of models used for distillation. Llama forbids using its output to train other models.

UMAP

UMAP (Uniform Manifold Approximation and Projection) is a dimensionality reduction technique that has gained substantial traction in the AI community. While t-SNE has been the go-to choice for visualising high-dimensional data, UMAP offers better preservation of global structure and runs significantly faster, making it valuable for large-scale AI applications such as exploring embedding spaces and analysing neural network activations.

UMAP's parameters need careful tuning to avoid misleading visualisations.

The Python UMAP library provides extensive documentation and explanation, with implementations also available for Rust, Java and R.

Claude Skills

Claude Skills are reusable prompt templates that codify workflows and domain expertise into repeatable patterns for AI coding assistants. Our teams have found them valuable for drafting proposals, structured debugging, generating commit messages and writing PR descriptions. The common thread is tasks that benefit from consistent approach and structured output.

Skills provide a simpler solution than MCP servers for many problems. Where MCP requires implementing a server and managing the protocol lifecycle, Skills are markdown files that encode expertise directly. Skills work well with data that exists as files in your project, since the AI assistant can already read those. MCP extends reach to running services and systems beyond filesystem access.

Structured RAG

Structured RAG extends basic RAG by organising retrieved knowledge as graphs, schemas or typed records rather than flat text chunks. Microsoft's GraphRAG uses an LLM to build a knowledge graph from source documents during indexing, then queries that graph at retrieval time. This addresses a weakness in standard RAG: questions requiring synthesis across many documents rather than finding a single relevant passage.

GraphRAG has matured since our last radar. LazyGraphRAG reduced indexing costs to a fraction of the original, removing the biggest barrier to adoption. Neo4j provides dedicated examples combining graph-based retrieval with LLM generation.

The trade-off remains upfront investment. Graph-based indexing requires more compute and design than vector-based RAG, and the knowledge graph must be maintained as source documents change. If your queries are primarily about finding relevant passages, standard RAG with cross-encoder reranking may suffice. If they require reasoning across documents, structured approaches justify their cost.

See also: RAG, Ontologies for AI grounding, Cross-encoder reranking.

Assess

Emerging or specialised approaches that warrant investigation for specific use cases, but require careful evaluation before adoption.

Neurosymbolic AI

Neurosymbolic AI combines neural networks with symbolic reasoning to address fundamental limitations of pure LLM approaches. Neural networks excel at pattern recognition and handling ambiguity, while symbolic AI provides logical reasoning and explainable inference. LLMs understand natural language well but cannot guarantee rule compliance or explain their reasoning in auditable ways.

Fundamentally, this is an architectural problem. LLMs operate through probabilistic pattern matching over language, not causal modelling. As Mark Burgess argues in his work on semantic spacetime, language models "paraphrase intentional knowledge" rather than tracing actual causal chains. Precise answers to precise questions require systems that explicitly encode what causes what.

This matters most in regulated sectors. Regulatory rules are non-negotiable constraints, not suggestions a model can approximate. Risk models need to know what entities are and how they relate, and compliance requires explainable decision trails. Similar pressures apply across financial services, healthcare and insurance.

Practical implementations range from lightweight to sophisticated. On the simpler end, teams constrain LLM outputs to valid ontology terms or use knowledge graphs to ground RAG retrieval. More advanced implementations use symbolic reasoning engines to validate LLM-generated conclusions. Renewed interest in Prolog reflects exploration of logic programming alongside LLMs.

We've placed this in Assess because production patterns are still emerging, but organisations in regulated sectors should be experimenting now.

See also: Prolog, Ontologies for AI grounding, Agentic tool use, World models.

World models

World models sit in the Assess ring as an emerging alternative to pure language model architectures for tasks requiring causal reasoning and planning. Where LLMs predict the next token based on statistical patterns in text, world models build internal representations of how environments behave, enabling systems to simulate outcomes before acting.

The field is developing along several paths. Yann LeCun's Joint Embedding Predictive Architecture (JEPA) learns by predicting missing information in an abstract embedding space rather than reconstructing raw pixels or tokens. Meta's V-JEPA and VL-JEPA extend this to video

and vision-language tasks with significantly fewer parameters than autoregressive alternatives. Karl Friston's active inference framework, implemented by Verses AI in their AXIOM system, takes a different approach rooted in how biological systems model their environments. Rather than chasing reward signals, active inference agents build generative models and act to minimise prediction error, with Verses reporting 60% performance improvement using only 3% of comparable deep learning compute. Generative world models form a third strand, with NVIDIA Cosmos and Google DeepMind's Genie 3 creating physically plausible simulated environments for training robots and autonomous systems.

For financial services, MarS from Microsoft Research demonstrates the pattern applied to market simulation, generating realistic interactive market scenarios for forecasting and anomaly detection without real capital at risk. The paper was accepted at ICLR 2025.

The enterprise value: these approaches offer causal modelling rather than statistical pattern matching. An LLM asked "what happens if I do X?" can only paraphrase similar scenarios from its training data. A world model can simulate the consequences. For teams wanting to experiment, Meta's V-JEPA 2 and NVIDIA Cosmos models are available on HuggingFace under permissive licences.

See also: Neurosymbolic AI, Physical AI and robotics foundation models.

LLM reproducibility

Large language models are non-deterministic even at temperature zero. This presents a fundamental challenge for regulated industries where Model Risk Management frameworks require reproducible, auditable decision-making. Banking regulations such as OCC/SR 11-7 assume a level of model stability that generative AI does not provide.

The underlying cause extends beyond floating-point arithmetic. Research demonstrates that batch-dependent kernel operations cause outputs to vary with server load rather than input alone. Smaller open weight models on controlled infrastructure tend to achieve more reproducible outputs than larger models served via shared APIs. Where stochastic behaviour is acceptable, the variation must be well-characterised so it can be explained to regulators as a designed property rather than an infrastructure artefact. Prompts and model versions should be treated as versioned code with change control and rollback procedures.

For teams requiring determinism from larger models, SGLang now offers deterministic inference building on batch-invariant operators, with the underlying research selected for oral presentation at NeurIPS 2025. Teams subject to MRM requirements should be actively evaluating their options now.

See also: Neurosymbolic AI, LLM-as-a-judge.

Hypothetical document embeddings (HyDE)

HyDE (Hypothetical Document Embeddings) addresses a common problem in search systems: poor performance when searching content that differs from training data. HyDE asks a large language model to imagine what an ideal document answering the query might look like, bridging the gap between how users ask questions and how information is written.

The system creates several hypothetical documents, converts them into embeddings and blends them together. This averaged representation finds real documents that are mathematically similar, often leading to more relevant results than traditional methods. The approach is particularly effective within RAG systems where accurate retrieval is crucial. Teams should evaluate HyDE for cases where high-precision retrieval is needed and the additional latency is acceptable.

See also: RAG, BERT variants, Cross-encoder reranking.

Fine-tuning with LoRA

Low-Rank Adaptation (LoRA) makes model customisation more practical by adding a small set of trainable parameters while keeping the original model unchanged, reducing computing requirements by 3-4 orders of magnitude while maintaining most of the performance of full fine-tuning.

Tools such as Lightning AI's lit-gpt and axolotl support implementation. We place it in Assess rather than Trial because successfully applying LoRA still requires significant ML expertise and careful attention to training data quality. Fine-tuning ties you to a specific model architecture, and given the pace of AI advancement, tomorrow's general-purpose models may outperform your carefully tuned older models. Migrating fine-tuned weights between architectures is particularly challenging. LoRA should only be deployed when the immediate business value clearly outweighs the technical and opportunity costs.

Physical AI and robotics foundation models

Physical AI represents the convergence of foundation model capabilities with robotics. Where traditional robotics relied on brittle, task-specific programming, robotics foundation models enable machines to generalise across tasks and adapt to novel situations.

The technical breakthrough is Vision-Language-Action (VLA) models, which extend vision-language models to include physical action outputs. NVIDIA's Isaac GR00T N1 represents the first open humanoid robot foundation model, using a dual-system architecture that separates deliberate planning from rapid reactive control. Google's Gemini Robotics is advancing similar capabilities. World Foundation Models complement these by enabling simulation-based training: NVIDIA Cosmos generates physically plausible synthetic environments that can train robots on scenarios too dangerous or rare to capture in the real world.

Production deployments remain concentrated in well-resourced organisations. The gap between research demonstrations and reliable industrial deployment is substantial. Hardware costs have fallen substantially over the past few years, but perception and control challenges in unstructured environments remain formidable. Organisations with physical AI ambitions should be experimenting, while approaching production timelines with caution.

See also: Digital twin platforms, World models.

CaMeL

CaMeL (CApabilities for MachinE Learning) is a defence architecture from Google DeepMind for mitigating prompt injection in agentic systems. The paper, Defeating Prompt Injections by Design, treats prompt injection as a problem to be solved structurally rather than through prompt cleverness or red-teaming alone. The architecture splits responsibilities across two models: a privileged P-model processes only user instructions and emits a program defining execution steps, while a quarantined Q-model handles external data but cannot call tools directly. A custom interpreter tracks data provenance, enforcing capability-based security so untrusted data cannot escalate privileges. The implementation is open source.

Prompt injection is the dominant unsolved problem for agentic systems, and it appears throughout this radar: in MCP, in agentic tool use, in AI red teaming tools and in our Hold placement for OpenClaw. Most current defences are statistical: filters, classifiers, evaluation harnesses. CaMeL is one of the first credible attempts to make the architecture itself prove that untrusted data cannot reach privileged operations. On the AgentDojo benchmark it solves 77% of tasks with provable security guarantees, often reducing successful attacks to zero. Simon Willison has a useful walkthrough for readers wanting an accessible introduction.

We've placed CaMeL in Assess because it addresses prompt injection more rigorously than anything else we have seen, but production patterns have not yet emerged. The limitations are real: users must define security policies, which carries fatigue risk; running two models adds latency and cost; and the approach has not been battle-tested at scale. For teams building agentic systems for regulated industries, this paper is required reading.

See also: Agentic tool use, Visual computer use agents, OpenClaw.

Hold

Not recommended for new projects; better alternatives exist.

Chain of thought (CoT)

Chain of Thought (CoT) has moved to Hold. While useful when it emerged, research from Wharton's Generative AI Labs demonstrates diminishing returns: gains are rarely worth the time cost, and for reasoning models such as o3 and GPT-5.2, CoT prompting can decrease performance since step-by-step reasoning is already internalised at the architecture level.

For non-reasoning models, CoT still shows modest benefits on mathematical and symbolic reasoning tasks, but these are precisely the domains where better alternatives are emerging. Dedicated reasoning models handle them natively, while neurosymbolic architectures offer more reliable solutions by coupling LLMs with explicit reasoning engines.

The frontier of prompt engineering has moved to structuring problems effectively. Frameworks such as the 5 Whys and inversion now offer more value than CoT prompting. Step-by-step reasoning is now handled by the models and architectures rather than the prompts.

See also: Neurosymbolic AI.

AI pull request review

AI's code review capabilities have improved substantially. Developers who work effectively with multi-turn AI conversations can now get useful feedback at every level: syntax issues, architectural patterns and subtle runtime concerns such as race conditions.

Yet we've kept AI Pull Request Review in Hold, for organisational rather than technical reasons. PR review isn't just about finding errors; it's a knowledge-sharing mechanism where senior developers mentor juniors and the team maintains awareness of how the codebase evolves. Teams who delegate review to AI often see a decline in collective code ownership.

We recommend using AI as a first-pass reviewer to catch issues before human review, but preserving the human step as deliberate practice for team alignment and knowledge transfer.

Tools

Software that enhances AI development workflows without being embedded in your application code: IDE extensions, CLI utilities, testing frameworks and observability.

Adopt

41. Software engineering copilots
42. Provider-agnostic LLM facades
43. Notebooks

Trial

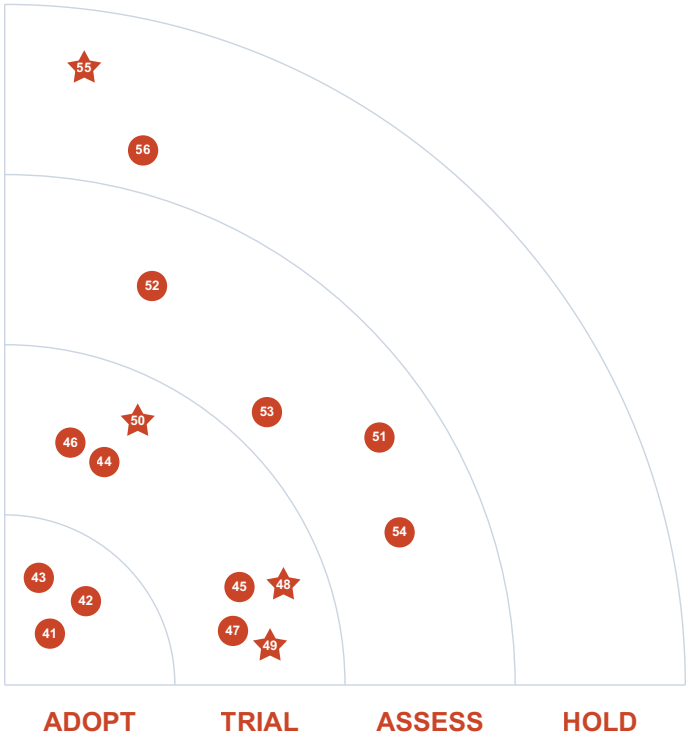
44. MLflow
45. Vector databases
46. Local model execution environments
47. LLM observability tools
48. LLM API gateways
49. AI red teaming tools
50. AI-assisted code migration

Assess

51. AI application bootstrappers
52. Visual computer use agents
53. Lakera
54. Structured output libraries

Hold

55. OpenClaw
56. Conversational data analysis



Adopt

Mature, well-supported tools with proven track records in production development workflows.

Software engineering copilots

AI-augmented development represents a permanent shift in software engineering. Teams not actively building capability here are falling behind.

The tooling falls into two categories. Model-agnostic interfaces let teams switch between providers: OpenCode stands out for its terminal experience and breadth of integration, Cursor, Windsurf and Zed are standalone editors, and CLI tools such as Aider and Cline work across providers. Provider-specific tools such as Claude Code, Gemini CLI and OpenAI Codex are optimised for their respective models. GitHub Copilot and Tabnine offer traditional IDE integrations.

Two approaches have emerged: free-form "vibe coding" and structured methodologies. Kiro offers both, with a conversational mode and a dedicated specs mode for drafting requirements before code generation. Cursor enables teams to codify standards through .cursorrules.

Senior engineers derive the greatest value, using AI for routine tasks whilst maintaining quality oversight. Junior developers often struggle to evaluate AI suggestions. Success correlates with intentional training around effective AI collaboration and a "trust but verify" mindset.

Provider-agnostic LLM facades

The LLM market moves quickly enough that today's best choice may be displaced within months. A facade between your application and the underlying provider keeps that switching cost manageable. Options range from the lightweight AISuite and Simon Willison's LLM library to heavier alternatives such as LangChain and LlamaIndex. A thin in-house wrapper is also a reasonable choice for teams that prefer to minimise dependencies. We have seen enough projects hampered by tight coupling to a single provider to consider some form of abstraction a default.

Notebooks

Notebooks remain the de facto standard for data science and ML experimentation. Code, prose and visualisations live alongside each other, which suits the iterative work of model development and gives technical and non-technical collaborators something concrete to discuss. Jupyter is the most widely used, with cloud-hosted equivalents from Google Colab, AWS Sagemaker, Azure and Databricks. Language-specific alternatives include Pluto.jl for Julia, Clerk for Clojure and Polynote for Scala.

Trial

Promising tools with growing adoption that are worth exploring for teams building AI systems.

MLflow

MLflow is an open-source platform for managing the machine learning lifecycle. The 3.0 release expanded it to cover generative AI: alongside the classical experiment tracking, model registry and deployment tooling, MLflow now offers OpenTelemetry-based tracing for LLMs and agents, capturing prompt construction, tool calls, memory retrievals, latency and token costs. This puts it in direct overlap with LLM observability tools such as Phoenix, Langfuse and LangSmith.

MLflow runs self-hosted or as a managed service on Databricks, SageMaker, Azure ML, Red Hat OpenShift AI and Nebius. That breadth is part of the appeal: teams avoid the lock-in of monolithic MLOps platforms while still being able to pick a managed offering where one fits.

Realising the benefits self-hosted requires technical expertise to configure and integrate. Unlike a turnkey platform such as Vertex AI, MLflow does not provide a plug-and-play experience; its modular components must be tailored to specific use cases. We recommend it for organisations that value flexibility and have the proficiency to manage integrations.

Vector databases

Vector databases have emerged as specialised tools for managing the high-dimensional embeddings required by AI models. Prominent solutions include Pinecone, Qdrant, Milvus and Weaviate.

Traditional databases may suffice for simpler operations, and alternative approaches such as Timescale's PGAI vectorizer bring vector search directly into Postgres, avoiding the data consistency challenges of keeping embeddings synchronised across databases. If a dedicated vector database is required, Pinecone leads in production readiness but comes with managed service costs, while open-source alternatives such as Qdrant and Milvus offer greater control but demand more operational expertise.

For prototyping, Chroma offers a Python-first approach with minimal configuration. A 2025 Rust rewrite improved performance, though it remains best suited for small-to-medium scale applications. LanceDB takes a different approach as an embedded vector database, similar in philosophy to SQLite. It operates as a library within your application using Apache Arrow's columnar format, making it compelling for local AI assistants and edge deployments where data must remain on-device. The trade-off is limited high-concurrency support.

Local model execution environments

Tools such as Ollama, LM Studio and AnythingLLM provide accessible ways to run open weight models on local hardware. These enable experimentation with models from Meta, Mistral, DeepSeek, Alibaba and OpenAI without API costs or sending data to external services. Many now support tool calling via MCP and connections to commercial APIs for hybrid workflows.

These tools serve developers testing AI features, teams comparing model responses and organisations exploring capabilities with sensitive data that cannot leave their infrastructure.

LLM observability tools

Modern agentic builds involve multi-step reasoning, tool orchestration, RAG retrieval and chains of LLM calls where a single user request might trigger dozens of internal operations. Debugging why an agent produced an unexpected result requires visibility into every step of that chain. This is distinct from production AI monitoring, which focuses on drift detection in deployed systems.

Phoenix, from Arize AI, has emerged as a leading open-source option. Built on OpenTelemetry, it provides tracing and evaluation with auto-instrumentation for LangChain, LlamaIndex, DSPy and direct integrations with OpenAI, Anthropic and AWS Bedrock. Langfuse is the most popular fully open-source alternative (MIT licence), combining tracing and evaluation with strong multi-turn conversation support.

For LangChain-committed teams, LangSmith provides native integration that surfaces framework internals in debugging views. Helicone takes a lightweight proxy approach: route API calls through its endpoint for observability without SDK changes. MLflow 3 also offers OpenTelemetry-based tracing for LLMs and agents, which is a natural choice where a single platform across classical ML and GenAI matters. Since these tools capture prompt and response data, data sovereignty matters too: Phoenix and Langfuse both offer self-hosting for teams with data residency requirements.

LLM API gateways

As organisations adopt multiple model providers, an infrastructure layer emerges between applications and providers. LLM API gateways handle routing, caching, failover, rate limiting and cost tracking at the proxy level, complementing code-level abstraction from libraries such as AISuite. A facade is a developer choice about how to call models; a gateway is a platform decision about how to manage model traffic across the organisation.

LiteLLM is the most widely adopted open-source option, providing an OpenAI-compatible proxy with spend tracking, budget controls and key management. Portkey offers a managed alternative with semantic caching and conditional routing. Kong brings enterprise API management experience to LLM traffic.

The operational value for multi-provider deployments is clear: centralised audit logging, per-team budget enforcement and automatic failover. Worth evaluating whether a gateway simplifies the operational story before building equivalent functionality in-house.

See also: Provider-agnostic LLM facades, LLM observability tools.

AI red teaming tools

Our security coverage has grown this quarter, with guidance on prompt injection in MCP and RAG and architectural defences such as CaMeL. Missing has been offensive testing: tools that systematically probe AI systems for vulnerabilities before attackers do.

Promptfoo is the standout, evolving from a prompt evaluation CLI into a comprehensive red teaming platform. OpenAI acquired it in March 2026 but the MIT licence remains. It ships financial services plugins for PCI DSS and banking regulation testing. Microsoft's PyRIT focuses on multi-turn attack orchestration with Azure AI Foundry integration. NVIDIA's Garak takes an agentic approach, autonomously probing for prompt injection, data leakage and toxicity.

The EU AI Act will require adversarial testing for high-risk systems by August 2026. We recommend starting with Promptfoo for breadth and CI/CD integration.

See also: CaMeL, Agentic tool use, LLM observability tools.

AI-assisted code migration

Large-scale code migration sits in a gap between what copilots do (assist line-by-line) and what bootstrappers do (generate new projects). Migration tools operate at codebase scale, applying thousands of coordinated changes to upgrade language versions, swap frameworks or modernise APIs. The pattern that works best combines deterministic code transformations with AI for edge cases that rules alone cannot handle.

Moderne is the leading platform, built on the open-source OpenRewrite engine. OpenRewrite provides deterministic "recipes" for common transformations such as Java version upgrades, Spring Boot migrations and Jakarta EE transitions, while Moderne adds AI-assisted handling of non-standard patterns and enterprise-scale orchestration.

These migrations are well-understood but labour-intensive, exactly where deterministic transformation augmented by AI pays off. We recommend trialling OpenRewrite on a representative repository before committing to the Moderne platform.

See also: Software engineering copilots, Spec-driven development.

Assess

Emerging tools that require careful evaluation before adoption.

AI application bootstrappers

AI application bootstrappers generate complete applications from prompts or designs. Lovable (formerly GPT Engineer) has emerged as a leader alongside V0, Bolt.new and Replit Agent. Google entered the space with Firebase Studio. These tools can take projects from concept to working application in hours.

Capabilities are improving rapidly. Lovable's visual editor allows Figma-like manipulation with automatic code updates. V0 excels at production-ready React components. Bolt.new runs full-stack development in the browser.

However, success still correlates strongly with existing engineering expertise. Senior developers use them as accelerators, understanding how to refactor generated code. Teams without this expertise risk shipping code they cannot maintain or debug. The gap between "working demo" and "production-ready system" remains substantial.

We recommend these primarily for prototyping and proof-of-concept work, with clear separation from production codebases unless your team has the engineering depth to take ownership of generated code.

Visual computer use agents

AI agents that interact with computers through visual understanding have matured but remain risky. Claude Computer Use controls desktops and browsers by seeing the screen and reasoning about interface elements. OpenAI Operator focuses on web browser automation through a managed environment. Browser Use offers an open-source alternative across multiple providers.

Reliability for bounded tasks has improved, with standard office workflows seeing success rates in the high 80s. However, prompt injection attacks, where malicious instructions hidden on web pages hijack agent behaviour, represent a systemic vulnerability. OpenAI has acknowledged this problem "may never be fully solved".

For many automation needs, programmatic approaches via APIs and workflow automation platforms remain more reliable and secure. Visual computer use is best suited to isolated environments where the agent cannot access sensitive data. Teams should grant minimal permissions and maintain human oversight for high-stakes actions.

See also: Agentic tool use, CaMeL.

This section was previously titled "Agentic computer use".

Lakera

Lakera was acquired by Check Point Software for approximately $300M in November 2025. Lakera Guard, its core AI safety scanning product, is being integrated into Check Point's CloudGuard WAF as part of a broader application security offering. The underlying capability of scanning LLM inputs and outputs for prompt injection, toxic content and data leakage remains relevant, but the product context has changed substantially.

Technical limitations from our earlier evaluation still apply: scanning is text-only with no multimodal support, custom rules rely on regex patterns rather than context-aware analysis, and scanning is non-stateful with no awareness of conversation history. Teams evaluating Lakera should now assess it as part of the Check Point ecosystem rather than as a standalone product.

Structured output libraries

Libraries such as Instructor, Outlines and Marvin address a common challenge: LLMs naturally produce freeform text, but applications need structured data. These libraries constrain outputs to match specified structures through prompting, logit manipulation or grammar-based generation. Instead of hoping an LLM produces valid JSON, developers specify Pydantic models and receive guaranteed-valid objects. For agentic systems this is essential, as agents need to produce function calls and decision objects that downstream code can reliably process.

The space is evolving quickly. Instructor has gained traction for its simplicity and Pydantic integration, while Outlines offers more sophisticated constrained generation. Native structured output features from model providers (OpenAI's JSON mode, Anthropic's tool use) may reduce the need for external libraries in some scenarios.

A broader category of runtime guardrails has grown up alongside these libraries. NVIDIA's NeMo Guardrails and Guardrails AI go beyond schema conformance to include prompt injection scanning, content filtering and hallucination checks. Teams building production LLM applications should evaluate both levels: structured outputs for type safety and guardrails for content safety.

See also: PydanticAI, Lakera, AI red teaming tools.

Hold

Not recommended for new projects due to better alternatives or limited long-term viability.

OpenClaw

OpenClaw is an open-source agent runtime created by Peter Steinberger, who later joined OpenAI. It runs persistent, always-on AI agents that execute multi-step tasks by controlling computers: clicking, typing, navigating applications, browsing the web. OpenClaw supports Claude, DeepSeek and OpenAI as backends, and has spawned a wave of imitators. NVIDIA's NemoClaw wraps OpenClaw in the NVIDIA Agent Toolkit with sandboxed execution against Nemotron 3 Super models. Moonshot's KimiClaw runs natively on kimi.com with a community skill marketplace and persistent cloud memory.

The security model has not been figured out across any of these variants. Persistent agent runtimes grant broad computer access to AI agents processing potentially untrusted instructions. The same prompt injection vulnerabilities that affect all visual computer use agents apply here, amplified by the breadth of access and the always-on character of the deployment. An agent with permission to control your browser, email and file system has an enormous blast radius if compromised.

We do not recommend OpenClaw or its variants for new projects until the security model matures. Teams already using one should enforce strict sandboxing, limit accessible applications and maintain human oversight for actions involving sensitive data.

See also: Visual computer use agents, Agentic tool use, Software engineering copilots, CaMeL.

Conversational data analysis

Tools such as pandas-ai, tablegpt, promptql and Julius enable natural language querying of databases. Modern MCP servers can provide substantial context to models, including schema understanding and data contents. Our experience with JUXT's XTDB revealed remarkable moments where models traversed complex table structures with apparent ease.

For experienced analysts, these tools represent a meaningful productivity boost, converting natural language into draft queries that can be refined. However, generated queries can be inefficient or incorrect despite appearing plausible. Uber's QueryGPT demonstrates both the potential and complexity, highlighting the guardrails required for reliable results.

We've placed this in Hold because successful deployment requires users capable of understanding and validating generated queries. These tools offer substantial benefits for data teams with appropriate expertise, but should be approached cautiously by those unable to review AI-generated database queries.

Platforms

Infrastructure and managed services that host and run AI workloads: cloud AI services, vector databases, model serving platforms and MLOps infrastructure.

Adopt

57. Foundation models
58. Weights & Biases
59. Temporal
60. Data pipeline orchestration tools
61. Cloud model hosting platforms

Trial

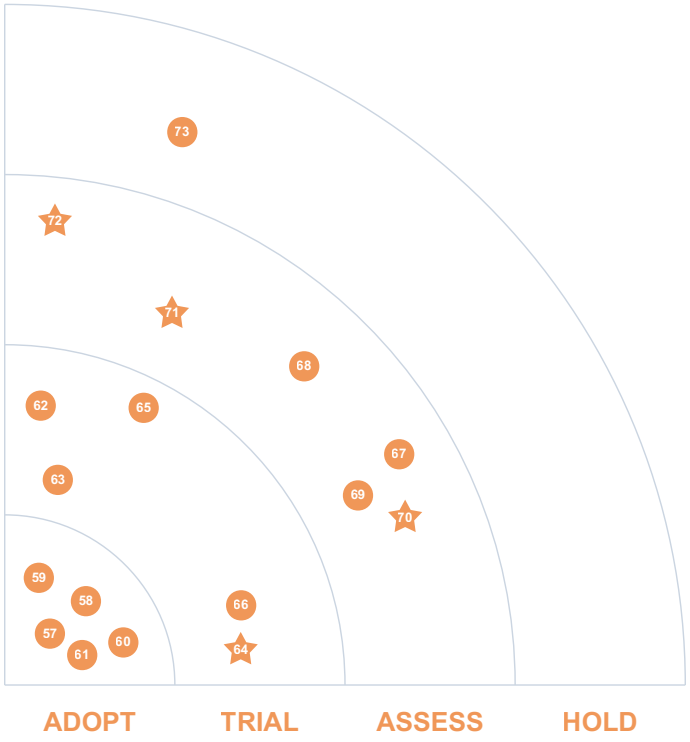
62. Production AI monitoring platforms
63. Open weight LLMs
64. Small language models
65. AI-powered workflow automation platforms
66. Digital twin platforms

Assess

67. Galileo
68. Kubeflow
69. Process mining platforms
70. Taalas
71. AI governance platforms
72. Agent memory architectures

Hold

73. Building against vendor-specific APIs



Adopt

Established, well-supported services ready for production AI deployments.

Foundation models

Foundation model providers continue to evolve rapidly. Major players such as OpenAI, Anthropic, Google and Meta compete alongside emerging organisations including DeepSeek, Alibaba and IBM.

Providers differentiate across three tiers: smaller, faster models optimised for speed and cost; larger, more capable models balancing capabilities with response times; and specialised reasoning models for complex problem-solving. The distinction between general and reasoning models is blurring, with GPT-5.4 and Claude Opus 4.6 integrating extended reasoning natively rather than through separate variants.

Foundation models warrant adoption for many business applications when paired with appropriate infrastructure (few-shot prompting, guardrails, RAG and evaluation frameworks). There is no universal "best model". We recommend benchmarking against your specific use cases, considering factors beyond raw performance: pricing, reliability, data privacy requirements and deployment options. High-quality open weight models with permissive licensing provide additional options for organisations with specific security or deployment requirements.

Key considerations

- **Performance & capabilities** (accuracy, speed, and domain-specific strengths)
- **Total cost of ownership** (API costs, compute resources, and integration)
- **Deployment options & technical requirements** (cloud, self-hosted, edge)
- **Data privacy & compliance** (regulatory, legal, and security implications)
- **Integration & lifecycle management** (context limitations, version control, updates)
- **Vendor stability & support** (roadmap alignment, documentation, community)

Foundation model providers feature comparison (April 2026)

Provider	Open Weights	Enterprise Focus	Reasoning Models	Edge Deployment	Long Context	Embedding API	Agentic Workflows	Model Selection Link
Alibaba	✓			✓	✓	✓		Models
Anthropic		✓	✓		✓		✓	Models
AWS		✓	✓		✓			Models
Cohere	✓	✓	✓		✓	✓		Models
DeepSeek	✓		✓	✓				Models
Google		✓	✓		✓	✓	✓	Models
IBM	✓	✓	✓	✓	✓			Models
Meta	✓			✓	✓			Models
MiniMax	✓			✓	✓		✓	Models
Mistral AI	✓	✓	✓	✓		✓		Models
OpenAI	✓	✓	✓	✓	✓	✓	✓	Models
Stability AI	✓			✓				Models
X	✓		✓		✓		✓	Models
Zhipu AI	✓	✓	✓		✓			Models

Feature definitions

- **Open Weights**: Models whose weights are publicly available for download and customisation
- **Enterprise Focus**: Strong emphasis on governance, security, and enterprise integration
- **Reasoning Models**: Specialised models for complex reasoning tasks such as mathematics or step-by-step problem solving
- **Edge Deployment**: Optimised for deployment on edge devices or resource-constrained environments
- **Long Context**: Support for context windows of 250K tokens or more
- **Embedding API**: Dedicated text embedding models and APIs for generating vector representations of text for semantic search and similarity tasks
- **Agentic Workflows**: Ability to autonomously plan and execute multi-step tasks using tools and external services. Goes beyond basic function calling to include complex workflow orchestration, error handling, dynamic planning based on intermediate results, and completing entire business processes without human intervention at each step

Weights & Biases

CoreWeave acquired Weights & Biases in May 2025. The platform continues under new ownership and our recommendation stands, though teams should monitor how the product evolves.

Weights & Biases tracks and visualises machine learning experiments. It provides a robust solution for managing ML workflows, particularly with complex models and large datasets. By making experiment tracking light touch, requiring just a few lines of code, it removes the friction that prevents teams from maintaining good measurement practices.

Collaboration features such as shared dashboards and reports make results visible to the whole team. Rather than knowledge being siloed in individual notebooks, experiments become shared assets. This visibility leads to faster knowledge sharing and quicker iteration cycles. However, tool adoption alone isn't enough; teams need to actively foster a culture that values measurement and experimentation for these benefits to materialise.

Temporal

Temporal is a workflow orchestration platform that provides durable execution for long-running processes. Although not AI-specific, it has become increasingly relevant as organisations build production agentic systems that must survive failures and run reliably over extended periods.

The core value is durability. If a multi-step process fails halfway through, it resumes from exactly where it left off. This differs from Kubernetes, which restarts crashed containers but knows nothing about application state. Temporal remembers that your workflow was on step 5 of 10, waiting for human approval, with specific context variables intact. The two are complementary, each handling failures at its respective layer.

Temporal's programming model treats workflows as ordinary code in familiar languages (Go, Java, Python, TypeScript, .NET) rather than configuration or visual diagrams. For agentic systems specifically, it addresses failure modes that many agent frameworks ignore: LLM calls timing out, tool invocations needing retry with backoff, human approvals taking days. Built-in retry policies and the ability to pause workflows indefinitely handle these cleanly.

We recommend Temporal for organisations moving beyond prototype agents toward production deployments where reliability matters. For simpler use cases, lighter-weight alternatives may suffice.

Data pipeline orchestration tools

Data pipeline orchestration has become essential infrastructure for managing complex data workflows, particularly those supporting AI initiatives. Whilst transformation tools such as dbt handle the "what" of data processing, orchestration platforms manage the scheduling, execution and monitoring of entire pipelines.

Apache Airflow is the established standard, with broad integration support across cloud platforms, though operating it well takes deliberate investment in DAG conventions and tooling. Prefect emphasises developer experience and dynamic workflow adaptation, with faster development cycles but fewer third-party integrations. Dagster takes an asset-centric approach where data assets become first-class citizens, providing built-in lineage tracking and data quality monitoring.

The choice depends on organisational context. Established enterprises with diverse toolchains often gravitate towards Airflow's ecosystem breadth, teams prioritising developer velocity prefer Prefect, and organisations with complex lineage requirements consider Dagster's asset-aware approach.

Cloud model hosting platforms

Model hosting has evolved beyond simple API access, with distinct platforms serving different needs from prototyping to enterprise production. Cloud-based hosting has become the default for most AI deployments.

Enterprise production environments typically use AWS Bedrock, Google Vertex AI or Azure OpenAI Service, which provide fine-tuning capabilities with enterprise security and integration with existing cloud infrastructure. For performance-critical applications, specialised providers such as Fireworks AI and Together AI focus on inference optimisation and custom model deployment, though teams must weigh simplified deployment against reduced ecosystem integration.

The inference hardware market is shifting beneath these platforms. Groq's custom LPU chips and Cerebras's wafer-scale processors have attracted major investment. For most organisations, these developments improve existing hosting providers' offerings rather than requiring a direct relationship with the chip makers.

Development teams and startups often prefer Replicate, Modal or Hugging Face Inference Endpoints, which offer direct paths from trained model to production API with flexible pricing. Hugging Face supports deployment of 60,000+ models with minimal configuration. The trade-off is more limited enterprise governance.

The choice reflects organisational priorities: enterprises with compliance requirements gravitate towards major cloud providers, performance-focused teams benefit from specialised inference platforms, and development teams prioritising rapid iteration prefer simplified deployment.

Trial

Platforms with growing adoption that offer innovative approaches worth exploring for specific use cases.

Production AI monitoring platforms

Whilst experiment tracking tools such as Weights & Biases and MLflow excel at managing the development lifecycle, a distinct category has emerged to monitor AI systems in production. These tools detect drift and unexpected behaviour in deployed models that only surface when models encounter real-world data at scale.

Arize AI provides unified observability across traditional ML and LLM applications, continuously tracking feature and embedding drift. Evidently AI offers both an open-source library and cloud platform, with over 100 metrics covering data quality and drift monitoring.

The key benefit is proactive detection: organisations learn about performance degradation before customer impact rather than discovering issues through support tickets. For teams already practising observability, AI-specific monitoring represents a natural extension of existing practices.

Open weight LLMs

Open weight LLMs (sometimes incorrectly called "open source") reached maturity in 2025, with some surpassing frontier models on specific tasks. MiniMax M2.7, Moonshot's Kimi K2.5, Zhipu's GLM-5 and DeepSeek V3.2 compete directly with closed models on coding and reasoning benchmarks. Smaller models such as Microsoft's Phi-4 and Google's Gemma 4 run on consumer hardware, making self-hosted inference practical for local coding assistants, on-device processing and latency-sensitive applications.

This quarter, source code was accidentally exposed through AI provider tooling, and autonomous agents with broad permissions destroyed production data. A self-hosted deployment puts data flow and model versioning under the operator's control. Prompts, outputs and proprietary code stay within the organisation, and the model only changes when the operator changes it. The pull, in short, is towards organisations retaining ownership of their data and the models that act on it.

Regulation is heading the same way. The EU AI Act, financial services rules and healthcare frameworks all want organisations to show where data flowed and how models behaved, not take a vendor's word for it. GLM-5 was trained entirely on Huawei Ascend chips with no NVIDIA dependency, a reminder that supply chain sovereignty extends down to the silicon.

Self-hosting still needs considerable ML engineering expertise, and total cost of ownership isn't always lower than API alternatives once compute and engineering time are factored in. For routine work, pay-per-use APIs still win. For regulated or sensitive data, the calculation has changed.

Small language models

Small language models (SLMs) are the smaller cousins of the open weight LLMs above. The current crop sits in the 1B–7B parameter range and runs comfortably on a single GPU, a modern laptop or in some cases a phone. Microsoft's Phi-4-Mini (3.8B), Google's Gemma 4 E2B and E4B (2–4B effective parameters), Meta's Llama 3.2 1B and 3B and Mistral's Ministral variants now perform at a level that was out of reach at this size two years ago.

In our experience, the case for SLMs comes down to using your own data. A few thousand labelled examples and a fine-tuning run produce a model that has the organisation's terminology and edge cases built in. Hosted in-house, the data stays within the organisation, and the resulting model is small enough to deploy on the infrastructure that team already operates rather than something purpose-built.

The natural fit is classification work where the language is nuanced and the cost of getting it wrong is real: routing customer messages by intent, triaging support tickets, tagging documents against a controlled vocabulary or extracting structured fields from free text. These are jobs that traditional keyword and feature-engineered approaches have never handled well, and where reaching for a frontier API on every request is overkill on both cost and latency. A fine-tuned SLM responds in the low hundreds of milliseconds on commodity hardware, faster on a GPU, and improves as more of the organisation's examples are added.

The trade-off is scope and discipline. SLMs will not match frontier models on open-ended reasoning or breadth of world knowledge, and getting them to behave reliably means taking the task definition, the labelling pipeline and evaluation seriously rather than leaning on clever prompting. For teams prepared to do that work, the result is a model that is cheap to run, fits the infrastructure they already have and is trained on the data they already own.

See also: Open weight LLMs, Local model execution environments.

AI-powered workflow automation platforms

Visual workflow automation platforms allow teams to build AI-powered business processes through drag-and-drop interfaces rather than code. Zapier focuses on connecting SaaS applications with AI capabilities. n8n offers self-hosting, open-source licensing and extensive customisation for technical teams. Microsoft Power Automate provides native Office 365 integration with enterprise governance. Make.com emphasises sophisticated visual workflow design with AI agent functionality.

Common use cases include lead qualification using LLM analysis, automated content generation, customer support routing and data processing pipelines incorporating AI models.

When evaluating, consider technical capability, data sovereignty requirements and scalability. Self-hosted solutions such as n8n offer maximum control but require expertise, while SaaS offerings reduce overhead but may have cost implications at scale. Teams should also assess error recovery and debugging capabilities, as AI components can fail less predictably than traditional integrations.

Digital twin platforms

A digital twin is a virtual representation of a physical system that maintains bidirectional synchronisation with its real-world counterpart. Unlike traditional simulation, digital twins continuously ingest live sensor data, enabling organisations to test changes in simulation before deploying them, diagnose problems without physical inspection and simulate the impact of new equipment on existing workflows.

NVIDIA Omniverse has emerged as the dominant platform, providing a simulation environment built on OpenUSD that enables physically accurate rendering and real-time collaboration. Its Isaac Sim extension targets robotics simulation specifically.

Digital twin platforms remain in Trial because successful deployment requires significant organisational investment beyond the platform itself. The challenge lies in data integration, maintaining synchronisation between physical and virtual systems and building capability to act on simulation insights. These are most relevant for organisations operating complex physical infrastructure: manufacturing plants, logistics networks, energy systems or robotics deployments. Organisations earlier in their data maturity journey should ensure foundational sensor instrumentation and data pipelines are in place before investing.

Assess

Emerging or specialised services that require careful evaluation before adoption.

Galileo

Galileo takes a distinctive approach to AI evaluation: rather than relying on expensive frontier LLMs as judges, it uses purpose-built small language models called Luna-2 for low-latency evaluation of hallucination detection, context adherence and output quality. This matters in production, where evaluation metrics need to run at serving latency to act as quality gates.

For agent debugging, Galileo provides Timeline, Conversation and Graph views for tracing execution paths. Cloud and on-prem deployment options make it viable for regulated industries with data residency requirements.

The Luna-2 approach is a differentiator. Most evaluation platforms use frontier models as judges, which is slow, expensive and creates a circular dependency on the providers you're evaluating.

The trade-off is vendor dependency. Teams wanting open-source flexibility should consider Langfuse for tracing or Phoenix for observability. The evaluation tooling space remains competitive, and we want to see how it settles before moving Galileo beyond Assess.

See also: LLM observability tools, Production AI monitoring platforms.

Kubeflow

Kubeflow is an open-source ML platform built on Kubernetes, combining orchestration capabilities with ML-specific tools: Pipelines for workflow automation and KFServing for model deployment. This integrated approach helps bridge the gap between data scientists and operations teams.

Several factors keep Kubeflow in Assess. Implementing it demands expertise in both Kubernetes and ML engineering. Many organisations struggle with setup and ongoing maintenance, and report long timelines before seeing tangible benefits.

Organisations with established ML practices and Kubernetes expertise should consider it, particularly for challenges around model deployment, experiment reproducibility or resource utilisation. Smaller teams or those earlier in their ML journey may prefer managed options such as Vertex AI Pipelines.

Process mining platforms

You cannot reliably automate processes that have not yet been optimised. Most of a process flows as expected, but the exceptions and edge cases determine whether automation succeeds or fails. As agentic AI moves toward enterprise deployment, process mining is becoming

essential preparation: it uses event logs from enterprise systems to discover how processes actually execute versus the designed workflow. The related discipline of task mining records user activity at the desktop level, capturing the tacit knowledge of how workers handle exceptions and workarounds. Together, these capabilities map the reality that AI agents would need to replicate.

The major platforms serve different organisational profiles. Celonis, the market leader, delivers the deepest analytics for complex multi-system processes but requires significant investment. ABBYY Timeline offers integrated process and task mining in a more accessible package, suited to business users who want to identify bottlenecks without coding. QPR ProcessAnalyzer, available in the Snowflake Marketplace, is the natural choice for Snowflake-centric companies. UiPath Process Mining provides seamless integration between discovery and automation execution, though this creates ecosystem lock-in. Microsoft Power Automate Process Mining is limited compared to specialist platforms but lowers the barrier for organisations already on the Microsoft stack. Fluxicon Disco is a standalone desktop application best suited for consultants and rapid proof-of-concept projects.

We recommend starting with process discovery in a contained domain rather than attempting an enterprise-wide rollout.

Taalas

Taalas hard-wires AI models directly into custom silicon (ASICs). Rather than running models on general-purpose GPUs, Taalas "prints" a specific model onto a chip, unifying storage and compute at DRAM-level density. The result is a fixed-function chip that holds one model and cannot be rewritten. Their first product, the HC1, integrates Meta's Llama 3.1 8B, with claimed performance of 16,000 tokens per second per user at 10x GPU throughput and 20x lower production cost.

The trade-off is radical inflexibility. Each chip runs exactly one model. When that model is superseded, the chip becomes electronic waste. This makes Taalas most compelling for scenarios where a specific model will be deployed at scale for an extended period: edge inference, embedded systems or dedicated infrastructure for a known workload. For regulated industries, a model baked into silicon is inherently versioned and immutable, simplifying reproducibility concerns. Teams subject to Model Risk Management requirements may find this appealing.

We've placed Taalas in Assess because the technology is early and the economics only work at significant scale. The assumption that a single model will remain useful long enough to justify dedicated silicon runs counter to how fast the field is moving.

See also: Open weight LLMs, Local model execution environments.

AI governance platforms

The EU AI Act reaches full enforcement in August 2026. High-risk AI systems must demonstrate conformity assessments, risk documentation, human oversight and ongoing monitoring. Banks and insurers already operate under model risk management frameworks (SR 11-7 in the US, SS1/23 in the UK), but these were written for statistical models with stable inputs and deterministic outputs. Extending them to LLM-based systems requires tooling that existing GRC platforms were not designed to provide.

Credo AI has emerged as the leading dedicated platform, providing AI risk classification aligned to the EU AI Act, NIST AI RMF and ISO 42001. It powers the compliance accelerators in IBM watsonx.governance through an OEM partnership, and Microsoft is integrating it with Azure AI Foundry. Others compose governance workflows from existing tools: model cards in Hugging Face, risk registers in GRC platforms, audit trails from observability tooling. Dedicated platforms offer structured workflows out of the box, while bespoke approaches require more engineering but can match how your organisation already works.

The space is young and consolidating fast. Financial services organisations should be evaluating now, particularly those with EU AI Act obligations, but should expect the vendor landscape to shift.

See also: LLM reproducibility, Production AI monitoring platforms.

Agent memory architectures

Stateless agents hit a wall quickly. An AI coding assistant that forgets what it learned yesterday, or a customer service agent that can't recall a conversation from last week, forces users to re-establish context on every interaction. Agent memory architectures address this with tiered storage: working context for the current session, long-term memory for persistent knowledge and episodic recall for past interactions.

Mem0 has gained the most traction, serving as the exclusive memory provider for the AWS Agent SDK. It sits alongside your agent framework, automatically extracting and retrieving memories without changes to orchestration logic. Letta (formerly MemGPT) pioneered treating context management like virtual memory, paging information in and out as needed. Zep differentiates on temporal accuracy, maintaining a knowledge graph that tracks how facts change over time.

For regulated industries, agent memory introduces governance questions your existing data policies may not cover. Where does the memory reside? How do you audit what an agent "knows" about a customer? How do you comply with deletion requests? Involve compliance and data protection teams early.

See also: Agentic tool use, Microsoft Agent Framework.

Hold

Not recommended for new projects; better alternatives exist.

Building against vendor-specific APIs

Tightly coupling applications to vendor-specific LLM APIs poses significant risk in a market where capabilities and pricing shift monthly. Organisations that build directly against OpenAI, Anthropic or other proprietary APIs often find themselves locked in, facing painful migrations when a better or more cost-effective model emerges.

We recommend abstraction libraries that provide a common interface to multiple providers. AISuite or Simon Willison's LLM CLI let you switch between models with minimal code changes, handling the nuances of different APIs behind a consistent interface. These abstractions add some complexity and may limit access to vendor-specific features, but the protection against lock-in outweighs these drawbacks in most cases.

juxt.pro/ai-radar