技术雷达

针对当今科技领域发展的前沿指南

第32期
2025年4月

/thoughtworks
Design. Engineering. AI.

关于技术雷达

Thoughtworkers 始终对技术怀有炽热激情。我们致力于构建技术、开展研究、实施测试、推动开源、撰写洞见，并不懈追求技术的持续优化——这一切努力皆以普惠大众为宗旨。我们的使命是倡导软件卓越理念，引领信息技术革命。为此，我们创建并持续分享 Thoughtworks 技术雷达，将其作为实现这一使命的重要载体。

Thoughtworks 技术顾问委员会——由公司资深技术领袖组成的核心团队——负责技术雷达的编撰工作。该委员会定期召开会议，深入探讨 Thoughtworks 全球技术战略，以及对本行业产生深远影响的技术趋势。技术雷达以高度凝练的形式，系统呈现技术顾问委员会研讨成果，旨在为从开发者至首席技术官等广泛利益相关方提供价值参考。

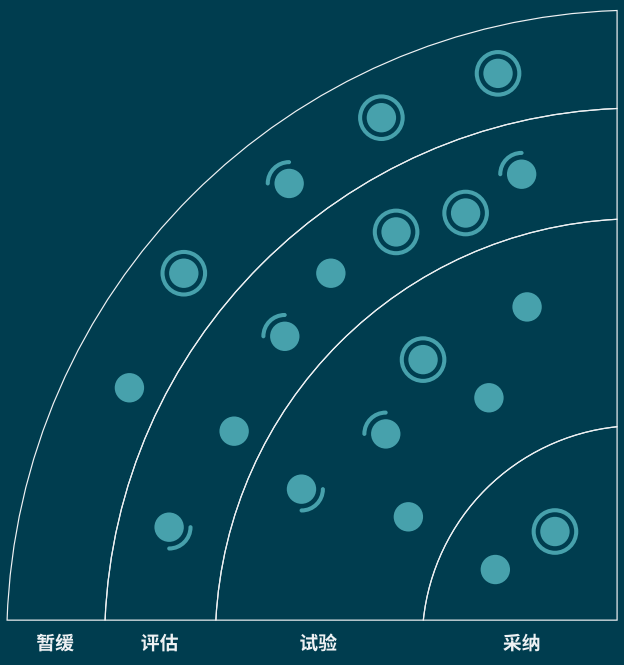
我们诚挚邀请您共同探索这些前沿技术。技术雷达采用可视化设计，将技术要素划分为四大象限：技术、工具、平台以及语言与框架。对于可能跨象限分布的技术条目，我们依据其核心属性进行精准归类。同时，通过四个同心环的布局设计，清晰展现我们对各项技术当前所处发展阶段的专业判断。

如需获取技术雷达的更多背景信息，请访问 thoughtworks.com/cn/radar/faq

雷达一览

技术雷达的核心使命是追踪具有价值的技术动向，我们将其定义为技术脉冲。本雷达通过两大核心维度对技术脉冲进行系统化组织：象限与评估环。技术象限区分技术脉冲的领域属性，评估环则体现我们对各项技术的应用建议层级。

技术脉冲指在软件开发领域产生影响力的技术或方法。这些脉冲始终处于动态演进之中——其在雷达图谱中的位置会持续变化——通常随着评估环的递进，标志着我们对其推荐力度的逐步增强。



采纳：我们坚定认为行业应当广泛采用此类技术。在项目条件适配时，我们会优先应用这些技术。

试验 具备显著探索价值。建议重点构建相关技术能力，企业可在风险可控的项目中开展实践验证。

评估：值得进行深度调研，重点评估其对组织产生的潜在影响。

暂缓：建议采取审慎态度对待技术应用。

新的　挪进 / 挪出　没有变化

技术雷达具有前瞻导向性。为保持内容时效性，我们对近期未发生位移的技术条目进行视觉淡化处理，此举并非否定其技术价值，而是受限于雷达版面的空间约束。

贡献者

技术顾问委员会（Technology Advisory Board，简称 TAB）由 21 位资深技术专家组成，是 Thoughtworks 技术治理体系的核心智库。该委员会采用 " 线下深度研讨 + 线上持续协同 " 的运作机制：每年举行两次线下全体会议，双周召开线上例会。其核心职能是为 Thoughtworks 首席技术官 Rachel Laycock 提供战略决策支持，驱动企业技术愿景的落地实施。

作为跨领域技术治理机构，TAB 聚焦影响企业技术演进与技术人才发展的关键议题，构建覆盖技术战略、创新实践与组织能力的全景洞察。本年度技术雷达的内容体系，源于该委员会 2025 年 2 月在曼谷举行的专项研讨成果，汇集了全球顶尖技术专家对行业趋势的前瞻判断。


Rachel Laycock
首席技术官

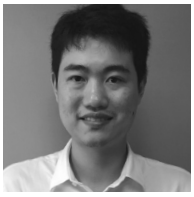

Martin Fowler
首席科学家


Bharani Subramaniam


Birgitta Böckeler


Bryan Oliver


Camilla Falconi Crispim


Chris Chakrit Riddhagni


Effy Elden


Erik Dörnenburg


James Lewis


Ken Mugrage


Maya Ormaza


Mike Mason


Neal Ford


Ni Wang
王妮


Nimisha Asthagiri


Pawan Shah


Selvakumar Natesan


Shangqi Liu
刘尚奇


Vanya Seth


Will Amaral

制作团队人员

编审委员会

- **William Amaral** — 产品负责人
- **Preeti Mishra** — 项目及活动经理
- **Richard Gall** — 内容编辑
- **Michael Koch** — 文字编辑
- **Gareth Morgan** — 内容与思想领导力总监

设计与多媒体部

- **Leticia Nunes** — 首席设计师
- **Sruba Deb** — 视觉设计师
- **Kevin Barry** — 多媒体专家
- **Ryan Cambage** — 多媒体专家
- **Anish Thomas** — 多媒体设计师
- **唐蓉** — 中文版设计师

数字与网络体验部

- **Rashmi Naganur** — 业务分析师
- **Brigitte Britten-Kelly** — 数字内容策略师
- **Vandita Kamboj** — 用户体验设计师
- **Anisha Thampy** — 视觉设计师
- **Lohith Amruthappa** — 数据分析专家
- **Neeti Thakur** — 营销自动化专员

传播与公共关系

- **Shalini Jagadish** — 内部传播专员
- **Hiral Shah** — 社交媒体运营专家
- **Abhishek Kasegaonkar** — 社交媒体运营专家
- **Linda Horiuchi** — 公共关系专员
- **Kathryn Jansing** — 公共关系专员
- **Soumyajit Dey** — 活动与广告策划专家
- **Anushree Tapuriah** — 活动与广告策划专家

中文翻译

- **廖燊** — 软件开发工程师
- **张霄翀** — 专家级软件开发工程师
- **余琦** — 专家级软件开发工程师
- **李天舒** — 软件开发工程师
- **程显通** — 软件开发工程师

本期主题

监督式智能编码代理

我们的多个主题都强调了生成式 AI 领域的快速创新，其中之一尤其聚焦于编码助手能力的快速提升。越来越多工具开始支持开发者直接在 IDE 内通过与 AI 聊天来驱动代码实现，这种方式也被称为 Agentic、“Prompt-to-code” 或 “基于对话的编程（Chat-oriented Programming, CHOP）”。在此模式下，AI 辅助工具不再局限于回答问题或生成代码片段，而是可以主动导航并修改代码、更新测试、执行命令，甚至主动修复 lint 错误和编译问题。尽管我们对声称能完全自主完成大型开发任务的代码 agent 持谨慎态度，但这种由开发者指导并由 agnet 完成行动的 “受监督的 Agents” 模式已有了一些值得肯定的实践效果。Cursor、Cline 和 Windsurf 正在 IDE 集成工具领域引领这一趋势，GitHub Copilot 也在持续进步。此外，aider 、goose 和 Claude Code 等终端型工具同样具备类似的 agent 助手的能力。然而，即使取得了显著进步，我们仍对这一趋势可能引发开发者对 AI 生成代码的 过度信赖与自满 保持谨慎。尽管目前在辅助代码便携上已有了不错的成果，但在代码审查过程中仍需要保持高度的警惕与审慎。毕竟，能力越强，责任越大……

可观测性的演进

随着分布式架构复杂性的不断增加，可观测性领域正经历深刻的变革。尽管可观测性一直是软件开发的基础能力，但它正在与整个软件开发生态系统一起不断演进。当前的一个新兴重点是 LLM observability（大语言模型可观测性），这是将 AI 应用于生产环境的关键环节。我们见证了众多用于监控和评估 LLM 性能的工具的涌现，包括 Weights & Biases Weave、Arize Phoenix、Helicone 和 HumanLoop 。此外，AI 辅助可观测性 也成为一大趋势，这类工具借助 AI 提升分析和洞察能力。同时，OpenTelemetry 的广泛采用正在推动可观测性走向标准化，帮助团队摆脱对特定供应商的依赖，并在工具选择上具备更大的灵活性。许多领先的可观测性工具（如 Alloy、Tempo 和 Loki）都已支持 OpenTelemetry。可观测性领域的快速创新反映了行业对其重要性的日益重视。这种演进不仅推动了技术和实践的更迭，也形成了一个良性循环，使得可观测性作为现代软件开发的核心能力不断得到巩固和提升。

RAG 中的“R”：检索的进化

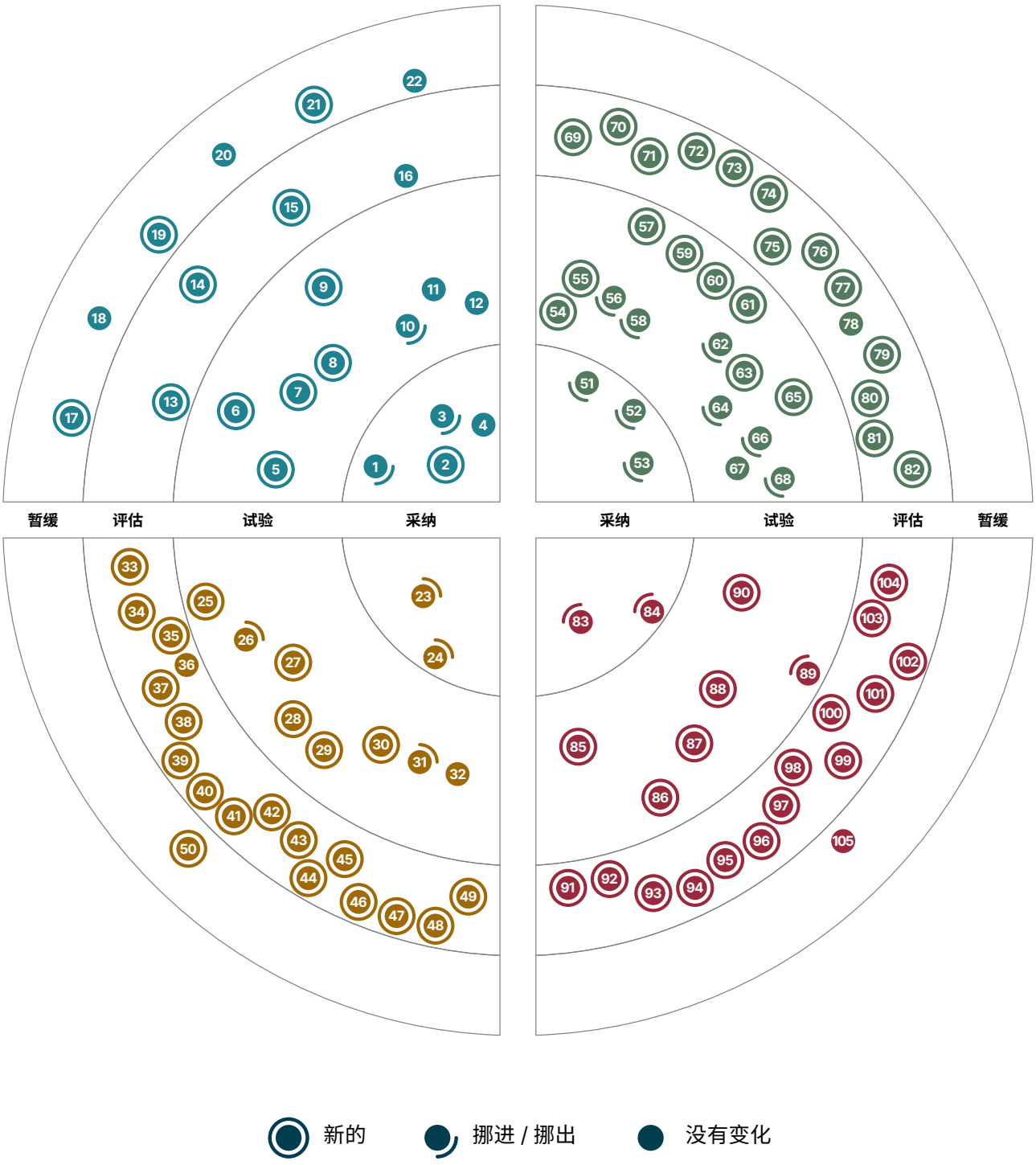
我们预计生成式 AI 生态系统的不同部分将以不同的速度演进，而在本期技术雷达中，我们观察到 RAG 中的“R”（检索增强生成中的检索）正在快速发展。与 LLM 这一“黑箱”的关键交互之一是定制提示词的输入，以生成相关且有用的响应。而 RAG 中对高效检索的需求日益增长，也催生了许多新工具和新技术，这些内容在本期雷达中得以重点呈现。例如，我们探讨了纠正型 RAG，它可以根据反馈或启发式规则动态调整响应；融合型 RAG，通过结合多种数据源和检索策略，提供更全面且稳健的响应；以及自助型 RAG，它完全避开传统的检索步骤，按需获取数据。此外，我们还强调了 FastGraphRAG，通过生成可供人类浏览的图表，提升了数据的可理解性。基于团队的讨论和提名，RAG 中的“R”已成为一个备受关注且快速演进的话题，展现出其在生成式 AI 领域中的关键作用。

驾驭数据疆界

大数据一直是行业关注的核心，但在本期技术雷达的讨论中，焦点不再仅仅是数据的规模，而是如何处理丰富而复杂的数据。随着非结构化数据在企业中的日益普及和重要性，如何有效管理和包装这些数据，使其能够成功应用于 AI 技术、客户分析等领域，已成为当今企业发展的关键。

这一趋势在多个雷达条目中得到了体现：从矢量数据库工具到分析产品（如 Metabase），令人惊叹的是，软件生态系统的驱动力越来越多地来自于我们对数据的需求和期望。但这不仅仅是关于工具的讨论——在本期中，我们还关注到 数据产品思维，这一框架鼓励团队将产品思维的原则应用于数据生态系统中的分析部分。数据产品思维的兴起在某种程度上反映了长期以来如何正确利用数据这一持续挑战（早在 AI 崛起之前，这个议题就已被广泛讨论）。如今它走到聚光灯下（并进入我们的雷达讨论），表明数据管理的重要性正前所未有地凸显。如果没有足够的数据纪律，组织可能会在创新中遇阻，甚至在中长期内面临商业竞争劣势。

本期雷达



本期雷达

技术

采纳

1. 数据产品思维
2. Fuzz 测试
3. 软件物料清单（SBOM）
4. 威胁建模

试验

5. API 请求集合做为 API 产品的制品
6. 架构建议流程
7. GraphRAG
8. 按需特权访问管理（Just-in-time Privileged Access Management）
9. 模型蒸馏
10. 提示工程（Prompt Engineering）
11. 小语言模型（SLMs）
12. 利用生成式 AI 理解遗留代码库

评估

13. AI 友好的代码设计
14. AI 驱动的 UI 测试
15. 能力边界作为理解系统故障的模型
16. 从 LLMs 获取结构化输出

暂缓

17. AI 加速影子 IT（AI-accelerated Shadow IT）
18. 自满于 AI 生成的代码
19. 本地编码助手
20. 使用 AI 代替结对编程
21. 逆向 ETL（Reverse ETL）
22. SAFe™

平台

采纳

23. GitLab CI/CD
24. Trino

试验

25. ABsmartly
26. Dapr
27. Grafana Alloy
28. Grafana Loki
29. Grafana Tempo
30. Railway
31. Unblocked
32. Weights & Biases

评估

33. Arize Phoenix
34. Chainloop
35. DeepSeek R1
36. Deno
37. Graphiti
38. Helicone
39. Humanloop
40. 模型上下文协议（MCP）
41. Open WebUI
42. pg_mooncake
43. 推理模型（Reasoning Models）
44. Restate
45. Supabase
46. Synthesized
47. Tonic.ai
48. turbopuffer
49. VectorChord

暂缓

50. Tyk hybrid API management

工具

采纳

51. Renovate
52. uv
53. Vite

试验

54. Claude Sonnet
55. Cline
56. Cursor
57. D2
58. Databricks Delta Live Tables
59. JSON Crack
60. MailSlurp
61. Metabase
62. NeMo Guardrails
63. Nyx
64. OpenRewrite
65. Plerion
66. 软件工程代理（software engineering agents）
67. Tuple
68. Turborepo

评估

69. AnythingLLM
70. Gemma Scope
71. Hurl
72. Jujutsu
73. kubenetmon
74. Mergiraf
75. ModernBERT
76. OpenRouter
77. Redactive
78. System Initiative
79. TabPFN
80. v0
81. Windsurf
82. YOLO

暂缓

—

语言和框架

采纳

83. OpenTelemetry
84. React Hook Form

试验

85. Effect
86. Hasura GraphQL engine
87. LangGraph
88. MarkItDown
89. Module Federation
90. Prisma ORM

评估

91. .NET Aspire
92. Android XR SDK
93. Browser Use
94. CrewAI
95. ElysiaJs
96. FastGraphRAG
97. Gleam
98. GoFr
99. Java 后量子密码学（Java post-quantum cryptography）
100. Presidio
101. PydanticAI
102. 资源受限应用中使用 Swift
103. Tamagui
104. torchtune

暂缓

105. Node 超载

技术

采纳

1. 数据产品思维
2. Fuzz 测试
3. 软件物料清单（SBOM）
4. 威胁建模

试验

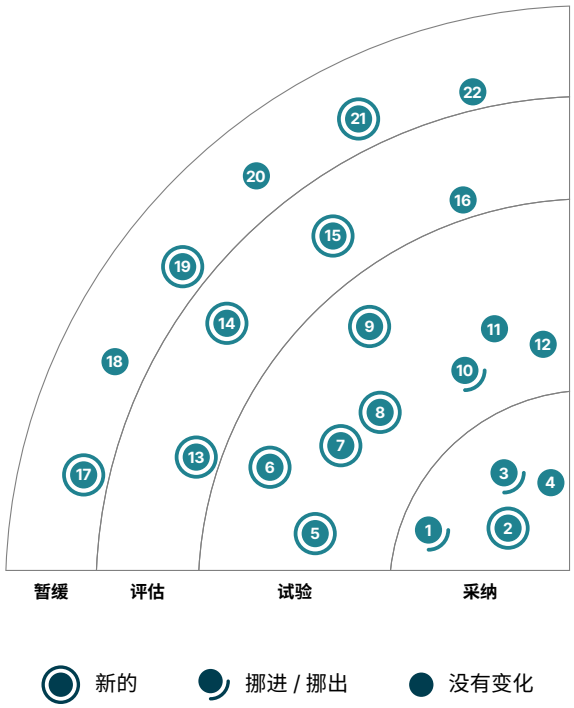
5. API 请求集合做为 API 产品的制品
6. 架构建议流程
7. GraphRAG
8. 按需特权访问管理（Just-in-time Privileged Access Management）
9. 模型蒸馏
10. 提示工程（Prompt Engineering）
11. 小语言模型（SLMs）
12. 利用生成式 AI 理解遗留代码库

评估

13. AI 友好的代码设计
14. AI 驱动的 UI 测试
15. 能力边界作为理解系统故障的模型
16. 从 LLMs 获取结构化输出

暂缓

17. AI 加速影子 IT（AI-accelerated Shadow IT）
18. 自满于 AI 生成的代码
19. 本地编码助手
20. 使用 AI 代替结对编程
21. 逆向 ETL（Reverse ETL）
22. SAFe™



1. 数据产品思维

采纳

企业正在积极采用 数据产品思维 作为管理数据资产的标准实践。这一方法将数据视为具有自身生命周期、质量标准，并专注于满足消费者需求的“产品”。无论组织选择 数据网格 还是 Lakehouse 架构，我们现在将其推荐为数据管理的默认建议。

我们强调数据产品思维中的以消费者为中心的原则，以推动更大的采用率和价值实现。这意味着通过 设计数据产品，从用例出发反向工作。同时，我们专注于使用现代数据目录（如 DataHub、Collibra、Atlan 和 Informatica）捕获和管理业务相关元数据与技术元数据。这些实践提升了数据的可发现性和可用性。此外，我们将数据产品思维应用于扩展 AI 项目，创建 AI 就绪数据。这一方法涵盖了全面的生命周期管理，确保数据不仅得到良好治理并具备高质量，同时在不再需要时能够符合法律和监管要求进行退役。

2. Fuzz测试

采纳

Fuzz 测试，或称模糊测试，是一种已经存在很长时间但仍然较少被广泛了解的测试技术。其目标是向软件系统输入各种无效数据并观察其行为。例如，对于一个 HTTP 端点，错误的请求通常应返回 4xx_ 错误，但 fuzz 测试往往会引发 5xx_ 错误甚至更严重的问题。随着工具的完善以及文档支持的增强，fuzz 测试在如今显得尤为重要，尤其是在更多 AI 生成代码和 自满于 AI 生成的代码 的背景下。因此，现在是采用 fuzz 测试的好时机，以确保代码的健壮性和安全性。

3. 软件物料清单（SBOM）

采纳

自我们在 2021 年首次提到 软件物料清单（SBOM）以来，其生成已经从新兴实践转变为我们项目中的默认选项。SBOM 生态系统显著成熟，提供了强大的工具支持，并实现了与 CI/CD 系统的无缝集成。工具如 Syft、Trivy 和 Snyk 能够从源代码到容器镜像生成全面的 SBOM，同时支持漏洞扫描。FOSSA 和 Chainloop 等平台通过与开发工作流集成以及实施安全策略，进一步提升了安全风险管理能力。尽管统一的 SBOM 标准仍在演化，但对 SPDX 和 CycloneDX 的广泛支持已显著降低了采用门槛。同时，AI 系统也需要 SBOM 的支持。英国政府的 AI 网络安全行为准则 和 CISA 的 AI 网络安全协作手册 就证明了这一点。我们将继续关注该领域的发展动态。

4. 威胁建模

采纳

在快速发展的 AI 驱动软件开发领域，威胁建模 比以往任何时候都更为关键，它不仅能够帮助构建安全的软件，同时还能保持敏捷性并避免出现 “安全三明治” 的情况。威胁建模是一组用于识别和分类潜在威胁的技术，可广泛应用于各种场景，包括生成式 AI 应用 ，这些应用 带来了独特的安全风险 。要想取得成效，威胁建模必须贯穿软件生命周期的各个阶段并定期执行，同时与其他安全实践相结合才能发挥最佳效果。这些实践包括定义跨职能的安全需求，以应对项目技术中的常见风险，以及利用自动化安全扫描工具进行持续监控，从而保障系统安全。

5. API 请求集合做为 API 产品的制品

试验

将 API 视为产品 意味着优先考虑开发者体验，不仅需要在 API 本身中融入合理和标准化的设计，还需要提供全面的文档以及流畅的入门体验。虽然 OpenAPI（如 Swagger）规范可以有效记录 API 接口，但入门仍然是一个挑战。开发者需要快速获取可用的示例，包括预配置的身份验证和真实的测试数据。随着 API 客户端工具（例如 Postman、Bruno 和 Insomnia）的逐步成熟，我们建议将 API 请求集合做为 API 产品的制品。API 请求集合应经过精心设计，以引导开发者完成关键工作流程，帮助他们轻松理解 API 的领域语言和功能。为了保持请求集合的最新状态，我们建议将其存储在代码库中，并将其与 API 的发布流水线集成在一起。

6. 架构建议流程

试验

在大型软件团队中，一个持久的挑战是确定由谁来做出塑造系统演进的架构决策。State of DevOps 报告 显示，传统的架构评审委员会（Architecture Review Boards）方式往往适得其反，不仅阻碍了工作流，还与低组织绩效相关。一种引人注目的替代方案是 架构建议流程 —— 这是一种去中心化的方法，任何人都可以做出架构决策，前提是他们向受影响的人和具有相关专业知识的人寻求建议。这种方法使团队能够在不牺牲架构质量的前提下优化工作流，无论是在小规模还是大规模环境中。乍看之下，这种方法似乎具有争议性，但像 架构决策记录（ADR）和建议论坛这样的实践能够确保决策是经过充分信息支持的，同时赋予那些最接近实际工作的人员决策权。我们看到这一模型在越来越多的组织中取得了成功，包括那些处于高度监管行业的组织。

7. GraphRAG

试验

在上次关于 检索增强生成（RAG）的更新中，我们已经介绍了 GraphRAG 。它最初在 微软的文章 中被描述为一个两步的流程：（1）对文档进行分块，并使用基于大语言模型的分析构建知识图谱；（2）通过嵌入检索的方式在查询时检索相关块，沿着知识图谱的边缘发现更多相关的分块，这些分块后续会被添加到增强提示中。在许多情况下，这种方法提高了大语言模型生成的响应数据的质量。我们在 使用生成式 AI 理解遗留代码库 的过程中也观察到了类似的好处——通过像抽象语法树和代码依赖这样的结构化信息去构建知识图谱。GraphRAG 模式正在获得更多的关注，像 Neo4j 的 GraphRAG Python 库这样的工具与架构正在不断出现以支持该模式。同时，我们认为 Graphiti 也符合广义上的 GraphRAG 模式。

8. 按需特权访问管理（Just-in-time Privileged Access Management）

试验

最小权限原则 确保用户和系统仅拥有执行任务所需的最低权限。特权凭证滥用是 安全漏洞 的主要原因之一，其中权限提升是常见的攻击向量。攻击者通常从低级访问权限开始，利用软件漏洞或配置错误获取管理员权限，尤其是在账号拥有过多或不必要权限时。另一个常被忽视的风险是静态特权（standing privileges）——即持续可用的特权访问，这大大增加了攻击面。按需特权访问管理（Just-in-time Privileged Access Management）有效缓解了这一问题，通过仅在需要时授予访问权限，并在任务完成后立即撤销权限，从而最大限度地降低暴露风险。真正的最小权限安全模型确保用户、应用程序和系统仅在最短时间内拥有完成任务所需的必要权限，这

是合规性和监管安全的关键要求。我们的团队通过自动化工作流程实现了这一模型：触发轻量化的审批流程，为用户分配临时角色并限制访问权限，同时为每个角色强制设置生存时间（TTL），确保权限在任务完成后自动过期，从而进一步减少特权滥用的风险。

9. 模型蒸馏

试验

Scaling laws 是推动 AI 快速发展的关键原则之一，即更大的模型、更大的数据集和更多的计算资源能够带来更强大的 AI 系统。然而，消费级硬件和边缘设备往往缺乏运行大尺寸模型的能力，因此产生了对模型蒸馏的需求。

模型蒸馏 将知识从一个更大、更强的模型（教师模型）转移到一个更小、更高效的模型（学生模型）。这一过程通常包括从教师模型生成一个样本数据集，并对学生模型进行微调，以捕获其统计特性。与通过移除参数来压缩模型的 剪枝 技术或 量化 不同，蒸馏旨在保留领域特定的知识，同时将精度损失降到最低。此外，蒸馏还可以与量化结合使用，以进一步优化模型。

这种技术最早由 Geoffrey Hinton 等人提出，现已被广泛应用。一个显著的例子是 Qwen/Llama 的 DeepSeek R1 蒸馏版本，它们在小模型中保留了强大的推理能力。随着蒸馏技术的日益成熟，它已不再局限于研究实验室，而是被广泛应用于从工业项目到个人项目的各类场景中。像 OpenAI 和 Amazon Bedrock 这样的供应商也提供了详细的指南，帮助开发者蒸馏自己的 小语言模型（SLMs）。我们认为，采用模型蒸馏技术能够帮助组织更好地管理 LLM 部署成本，同时释放 本地设备上 LLM 推理 的潜力。

10. 提示工程（Prompt Engineering）

试验

提示工程（Prompt Engineering）是指为生成式 AI 模型设计与优化提示词（Prompt）的过程，其目标是生成高质量、上下文相关（Context-aware）的响应。这一过程通常包括针对特定任务或应用场景，精心构建清晰、具体且上下文相关的提示，以实现模型输出效果的最优化。随着大语言模型能力的不断提升，尤其是 推理模型 的出现，提示工程的实践也必须随之调整。根据我们在 AI 代码生成方面的经验，少样本提示（few-shot prompting）在与推理模型协作时，可能不如简单的零样本提示（zero-shot prompting）表现出色。此外，被广泛使用的 思维链（Chain-of-Thought，CoT）提示 技术也可能 降低 推理模型的表现——原因可能在于当前推理模型通过强化学习已内置了 微调过的 CoT 机制 。

我们的实际经验也得到了学术研究的印证，即“高级模型可能 消除 软件工程领域对提示工程的依赖”。但在实际场景中，传统提示 工程技术仍然是减少模型幻觉（Hallucinations）并提升输出质量的重要手段，特别是在考虑推理模型与普通 LLM 在响应时间和 Token 成本等因素存在显著差异的前提下。在构建 自主代理应用（Agentic Applications）时，我们建议根据实际需求策略性地选择模型，并持续迭代与优化提示模板及相应的工程方法。如何在性能、响应速度与 Token 成本之间找到最佳平衡，依然是充分发挥 LLM 效能的关键所在。

11. 小语言模型（SLMs）

试验

最近发布的 DeepSeek R1 充分展示了 小语言模型（SLMs）为何仍然备受关注。满血版 R1 拥有 6710 亿个参数，并且需要约 1342GB 的 VRAM 才能运行，这通常只能通过八块最先进的 NVIDIA GPU 组成的“迷你集群”来实现。然而，DeepSeek 也提供了“ 蒸馏版 ”，即 Qwen 和 Llama 等更小的开放权重模型，使其能力得以迁移，并能够在更普通的硬件上运行。尽管这些小型版本在性能上有所折损，但相较于以往的小语言模型，依然实现了巨大的性能飞跃。小语言模型领域仍在不断创新。自上次技术雷达以来，Meta 推出了 Llama 3.2，涵盖 10 亿和 30 亿参数规模；微软发布了 Phi-4，其 140 亿参数模型在质量上表现出色；谷歌则推出了 PaliGemma 2，一个支持视觉 - 语言任务的模型，提供 30 亿、100 亿和 280 亿参数版本。这些只是近期发布的小型模型中的一部分，但无疑表明了这一趋势仍值得持续关注。

12. 利用生成式AI理解遗留代码库

试验

过去几个月，利用生成式 AI 理解遗留代码库 这一领域取得了显著进展。主流工具如 GitHub Copilot 已被广泛宣传能够帮助现代化改造遗留代码库。类似 Sourcegraph Cody 等工具，也正在让开发者更轻松地导航和理解整个代码库。这些工具综合运用多种生成式 AI 技术提供上下文感知（Context-aware）的帮助，极大地简化了对复杂遗留系统的分析与处理。此外，S3LLM 等专业框架则展示了大语言模型（LLMs）如何有效处理大规模科学计算软件（例如 Fortran 或 Pascal），将 GenAI 强化的代码理解能力推广到传统企业 IT 以外的场景。我们认为，鉴于全球范围内大量的遗留代码，这种技术未来将持续获得更多关注并加速普及。

13. AI友好的代码设计

评估

监督式 软件工程代理 的能力正在不断提升，它们现在能够识别所需的更新，并对代码库进行更大范围的修改。然而，我们也注意到，开发者对 AI 生成代码的自满情绪 正在增加，很多人不愿意审查由 AI 创建的大型变更集。一个常见的理由是: 既然未来的修改可以交给 AI 来完成，那么面向人类的代码质量就没那么重要了。然而，事实恰恰相反——AI 编程助手在结构良好的代码库上表现得更好，因此 AI 友好的代码设计 对于代码的可维护性至关重要。

值得庆幸的是，面向人类的优秀软件设计同样能够为 AI 提供助力。比如，明确的命名可以为代码提供领域上下文和功能信息；模块化和抽象设计能够限制代码改动范围，使 AI 的工作上下文更易于处理；而 DRY（don’t repeat yourself）原则则能减少重复代码，让 AI 更容易确保行为一致性。到目前为止，最适合 AI 的设计模式依然与传统的软件设计最佳实践密切相关。随着 AI 的不断发展，我们预计会有更多专门针对 AI 的设计模式出现。因此，从现在开始以 AI 友好的视角来思考代码设计，将会对未来大有裨益。

14. AI驱动的UI测试

评估

AI 在软件团队中的应用正逐步超越单纯的代码生成，新的技术正在涌现。其中，AI 驱动的 UI 测试 正受到越来越多的关注，它利用 LLM 的能力来理解图形用户界面（GUI）。目前，该领域主要有几种不同的实现方式。一种方法是使用针对 UI 快照处理进行微调的多模态 LLM，这类工具允许测试脚本以自然语言编写，并能自主导航应用程序。例如，QA.tech 和 LambdaTest 的 KaneAI 就属于这一类别。另一种方法，则是像 Browser Use 这样，结合多模态基础模型与 Playwright，通过对网页结构的深入理解进行测试，而不是依赖于特定微调的模型。

在测试策略中引入 AI 驱动的 UI 测试时，需要考虑其价值所在。这些方法可以补充人工探索性测试。尽管 LLM 的非确定性特性可能会导致测试结果的不稳定性，但它的模糊匹配能力也可能成为优势，尤其适用于缺少选择器的遗留应用程序或经常变更标签和点击路径的应用。

15. 能力边界作为理解系统故障的模型

评估

优雅扩展性理论 定义了适应性系统（包括构建和操作软件的社会技术系统）的基本规则。这一理论中的一个关键概念是能力边界（competence envelope）—— 系统在面对失败时能够稳健运作的边界。当系统被推到能力边界之外时，它会变得脆弱，更容易崩溃。该模型为理解系统故障提供了一个有价值的视角，例如在 2024 Canva 故障 中观察到的复杂故障。

残余性理论 是软件架构思维中最近的发展，它提供了一种方法，可以通过故意引入压力源并分析系统随时间对历史压力源的适应情况，来测试系统的能力边界。这些方法与社会技术系统中的反脆弱性、弹性和稳健性概念相一致。我们期待这些理论在实践中的应用，为提升系统的适应性与可靠性提供新的思路和工具。

16. 从LLMs获取结构化输出

评估

从 LLMs 获取结构化输出 是指通过定义的结构模式来约束语言模型的响应。这可以通过指示通用模型以特定格式响应，或者通过微调模型使其“原生”输出例如 JSON 的结构化数据来实现。OpenAI 现在支持结构化输出，允许开发人员提供 JSON Schema、pydantic 或 Zod 对象来约束模型响应。这种能力在函数调用、API 交互和外部集成中尤其有价值，因为这些场景中格式的准确性和一致性至关重要。结构化输出不仅提升了 LLMs 与代码交互的方式，还支持更广泛的使用场景，例如生成用于呈现图表的标记语言。此外，结构化输出已被证明可以减少模型输出中的幻觉现象。

17. AI 加速影子 IT（AI-accelerated Shadow IT）

暂缓

AI 正在降低非专业开发人员自行构建和集成软件的门槛，使他们无需等待 IT 部门响应自己的需求。尽管我们对这种技术带来的潜力感到兴奋，但同时也开始关注到 AI 加速影子 IT（AI-accelerated Shadow IT）的初步迹象。一些无代码（No-code）工作流自动化平台已支持对 AI API（如 OpenAI 或 Anthropic）的集成，这使得用户可能倾向于将 AI 用作“胶带”，将此前难以实现的系统集成临时拼凑起来，例如通过 AI 将聊天消息转换为

ERP 系统的 API 调用。同时，越来越多具有自主 Agent 能力的 AI 编码助手，甚至允许仅经过基础培训的非技术人员创建内部工具应用。

这些迹象呈现出类似于电子表格（Spreadsheets）当年迅速扩散的特征：虽然为企业关键流程提供了快速解决方案，但在长期运行后往往会造成规模更大的技术债（Tech Debt）。如果不加管控，这种新型影子 IT 将导致未经治理的应用程序激增，安全隐患加剧，数据分散在不同系统内。我们建议企业对此风险保持警觉，并谨慎评估快速问题解决与长期技术稳定性之间的平衡与取舍。

18. 自满于 AI 生成的代码

暂缓

随着 AI 编码助手的普及，越来越多的数据和研究也揭示了关于 自满于 AI 生成的代码 所带来的问题。GitClear 最新的 代码质量研究 显示，到 2024 年，重复代码和代码频繁变更的现象比预测的还要严重，而提交历史中的重构活动却在减少。同样反映出对 AI 的自满，微软的研究 显示，AI 驱动的信心往往以牺牲批判性思维为代价——这种模式在长期使用编码助手时表现得尤为明显。随着监督式 软件工程代理 的兴起，这种风险进一步放大，因为当 AI 生成的变更集越来越大时，开发者在审查这些结果时面临的挑战也随之增加。而 vibe coding 的出现——即开发者在审查极少的情况下让 AI 生成代码——更是说明了人们对 AI 生成输出的信任正在增长。这种方法可能适用于原型或其他一次性代码，但我们强烈建议不要将其用于生产环境的代码。

19. 本地编码助手

暂缓

由于对代码机密性的担忧，许多组织对第三方 AI 编码助手保持谨慎态度。因此，许多开发者开始考虑使用 本地编码助手 ，即完全在本地机器上运行的 AI 工具，无需将代码发送到外部服务器。然而，本地助手仍然落后于依赖更大型、更强大模型的云端助手。即使是在高端开发者设备上，较小的模型仍然存在能力上的局限性。我们发现它们难以处理复杂的提示词，缺乏解决更大问题所需的上下文窗口，并且通常无法触发工具集成或函数调用。这些能力对于当前编码辅助领域的前沿技术——代理式工作流——尤为重要。

因此，我们建议在使用本地助手时保持较低的期望值，但也有一些功能在本地环境中是可行的。目前一些流行的 IDE 已将较小的模型嵌入其核心功能中，例如 Xcode 的预测代码补全和 JetBrains 的整行代码补全。此外，可在本地运行的大语言模型，如 Qwen Coder，为本地的行内建议和处理简单编码查询迈出了重要一步。您还可以使用 Continue 测试这些功能，该工具支持通过 Ollama 等运行时集成本地模型。

20. 使用AI代替结对编程

暂缓

当人们谈论编码助手的时候，关于 结对编程 的话题就会不可避免地被提及。我们所处的行业对于它爱恨交织：有的同行信任它，有的同行讨厌它。现在大家都会对编码助手们提出同样的问题：一个程序员可以选择与人工智能，而不是另外一个程序员，进行结对编程，从而达到同样的团队产出吗？ Github Copilot 甚至自称为“你的 AI 结对程序员”。然而当大家都认为编程助手可以在结对编程方面带来好处时，我们还是不建议完全 使用 AI 代替结对编程 。把编码助手当做结对编程者忽略了结对编程的一个关键收益：它可以让团队而不只是个人变得

更好。在帮助解决难题，学习新技术，引导新人，或者提高技术任务的效率从而让团队更关注战略性设计等这些方面，使用编程助手确实大有裨益。但在诸如将正在进行的工作的数量控制在低水平，减少团队交接与重复学习，让持续集成成为可能，或者改善集体代码所有权等等这些团队合作相关的层面，它没有带来什么好处。

21. 逆向ETL（Reverse ETL）

暂缓

我们注意到，所谓的 逆向 ETL（Reverse ETL）正在迅速扩散。常规 ETL 在传统数据架构中是很常见的，它将数据从事务处理系统传输到集中式分析系统（如数据仓库或数据湖）。尽管这种架构存在许多已被广泛记录的缺点，其中一些问题通过 数据网格 方法得到了缓解，但它在企业中仍然很常见。在这种架构中，将数据从集中分析系统逆向回流到事务处理系统，在某些特定场景下有其合理性，例如，集中系统可以汇总来自多个来源的数据，或者在向数据网格迁移的过程中，作为一种 过渡架构 的一部分。然而，我们观察到一个令人担忧的趋势，产品供应商正利用 Reverse ETL 的概念，将越来越多的业务逻辑转移到一个集中式的平台（即它们的产品）中。这种方法加剧了集中式数据架构所导致的许多问题。例如，过度依赖于将业务逻辑嵌入到一个庞大的中央数据平台中，会导致数据管理复杂性增加，并削弱领域团队对其数据的控制能力。因此，我们建议在从庞大的集中数据平台向事务处理系统引入数据流时，务必保持高度谨慎。

22. SAFe™

暂缓

我们持续观察到 SAFe™（Scaled Agile Framework®）（规模化敏捷框架）正被广泛采用。同时我们也注意到，SAFe 过度标准化和阶段性门限的流程会造成阻碍，这可能助长信息孤岛，其自上而下的管控模式会在价值流中产生浪费，并抑制工程人才的创造力，还会限制团队的自主性和实验空间。企业之所以采用 SAFe，一个关键原因是组织敏捷化过程非常复杂，他们希望像 SAFe 这样的框架能够提供一个基于流程的简单捷径，从而实现敏捷转型。鉴于 SAFe 的广泛应用——包括在我们的客户中——我们已经培训了 100 多名 Thoughtworks 咨询顾问，以便更好地为他们提供支持。尽管我们拥有深入的知识并且做了诸多尝试，但我们仍然认为，面对复杂问题，有时确实没有简单的解决方案。因此，我们一直建议采用与全面变革计划相结合的 更加精简、价值驱动的方法和治理模式 。

Scaled Agile Framework® 和 SAFe™ 是 Scaled Agile, Inc. 的商标。

平台

采纳

23. GitLab CI/CD
24. Trino

试验

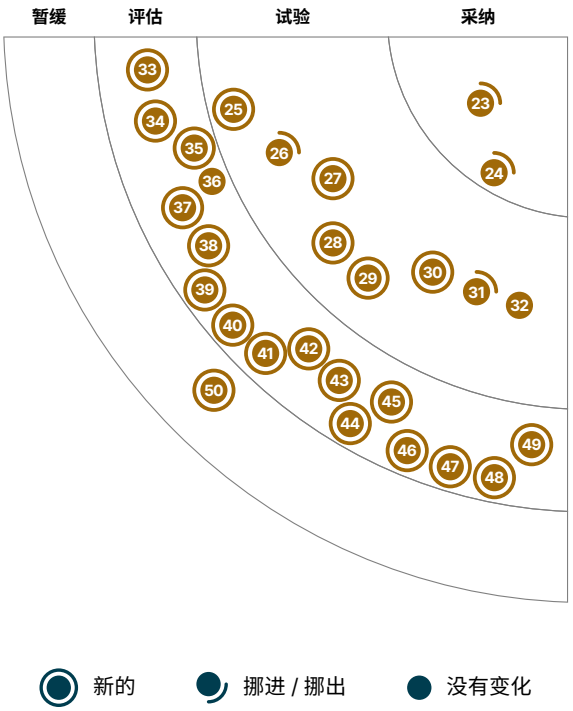
25. ABsmartly
26. Dapr
27. Grafana Alloy
28. Grafana Loki
29. Grafana Tempo
30. Railway
31. Unblocked
32. Weights & Biases

评估

33. Arize Phoenix
34. Chainloop
35. DeepSeek R1
36. Deno
37. Graphiti
38. Helicone
39. Humanloop
40. 模型上下文协议（MCP）
41. Open WebUI
42. pg_mooncake
43. 推理模型（Reasoning Models）
44. Restate
45. Supabase
46. Synthesized
47. Tonic.ai
48. turbopuffer
49. VectorChord

暂缓

50. Tyk hybrid API management



23. GitLab CI/CD

采纳

GitLab CI/CD 已发展为 GitLab 内部一个高度集成的系统，涵盖从代码集成、测试到部署和监控的所有环节。它支持复杂的工作流，包括多阶段流水线、缓存、并行执行和自动扩展运行器，非常适合大型项目和复杂流水线需求。我们特别想强调其内置的安全和合规工具（如 SAST 和 DAST 分析），使其非常适合具有高合规性要求的场景。此外，它还与 Kubernetes 无缝集成，支持云原生工作流，并提供实时日志、测试报告和可追溯性，以增强可观察性。

24. Trino

采纳

Trino 是一个开源的分布式 SQL 查询引擎，专为大数据的交互式分析查询而设计。它针对本地和云端环境进行了优化，支持直接在数据驻留的位置进行查询，包括关系型数据库和各种专有数据存储（通过连接器）。Trino 还能够查询存储为 Parquet 等文件格式的数据，以及像 Apache Iceberg 这样的开放表格格式。其内置的查询联邦功能允许将多个数据源的数据作为一个逻辑表进行查询，非常适合需要聚合多种来源数据的分析工作负载。Trino 是许多流行技术栈（如 AWS Athena、Starburst 和其他专有数据平台）的重要组成部分。我们的团队在各种用例中成功使用了 Trino，对于跨多个数据源进行分析的数据集查询，Trino 一直是一个可靠的选择。

25. ABsmartly

试验

ABsmartly 是一款先进的 A/B 测试与实验平台，专为快速且可信的决策制定而设计。其核心亮点是 Group Sequential Testing（GST）引擎，与传统 A/B 测试工具相比，可将测试结果的速度提升高达 80%。平台提供实时报告、深度数据分割以及通过 API 优先的方式实现的无缝全栈集成，支持在网页、移动端、微服务和机器学习模型中运行实验。

ABsmartly 专注于解决可扩展、数据驱动实验中的关键挑战，使得更快的迭代和更敏捷的产品开发成为可能。通过零延迟执行、强大的深度分割能力以及对多平台实验的支持，ABsmartly 对希望扩大实验文化并优先推动数据驱动创新的组织尤为有价值。借助其显著缩短的测试周期和自动化结果分析能力，ABsmartly 帮助我们比传统 A/B 测试平台更高效地优化功能和用户体验。

26. Dapr

试验

自从我们上次在技术雷达中介绍 Dapr 以来，它已经有了显著的发展。它的许多新特性包括任务调度、虚拟角色以及更为复杂的重试策略和可观察性组件。它的构建模块列表不断扩展，新增了任务、加密等功能。我们的团队还注意到 Dapr 在安全默认设置方面的日益关注，支持 mTLS 和无发行版镜像。总体而言，我们对 Dapr 的表现感到满意，并期待其未来演进。

27. Grafana Alloy

试验

前身为 Grafana Agent，Grafana Alloy 是一个开源的 OpenTelemetry Collector。Alloy 被设计为一个一体化的遥测数据收集器，用于收集包括日志、指标和跟踪在内的所有遥测数据。它支持常用的遥测数据格式，如 OpenTelemetry、Prometheus 和 Datadog。随着 Promtail 最近被弃用，Alloy 正逐渐成为遥测数据收集的首选工具——特别是在使用 Grafana 可观察性技术栈时，用于日志数据的收集。

28. Grafana Loki

试验

Grafana Loki 是一个受 Prometheus 启发的横向可扩展，高可用的多租户日志聚合系统。Loki 只对日志的元数据进行索引，并把它当做日志流的标签集，而日志数据本身则储存在像 S3, GCS 或 Azure Blob Storage 这样的块存储方案中。这样做的好处是 Loki 比竞争对手的运维复杂度更低，同时也降低了存储成本。正如你所期待的那样，它与 Grafana 和 Grafana Alloy 深度集成，即使其他的日志采集机制也被支持。

Loki 3.0 引入了对原生 OpenTelemetry 的支持，这使得与 OpenTelemetry 系统的数据摄入与集成如配置一个端点一样简单。此外，它还提供了高级的多租户功能，例如通过 shuffle-sharding 的方式实现各租户间的隔离，避免异常的租户（比如正在执行高负载查询或者出现故障）影响到集群中的其他租户。如果你还没有关注 Grafana 生态系统的最新发展，现在正是个好时机——它正在快速地演进。

29. Grafana Tempo

试验

Grafana Tempo 是一个高可扩展的分布式追踪后端，支持诸如 OpenTelemetry 等开放标准。Tempo 专为成本效率设计，依赖对象存储进行长期追踪数据的保存，并支持追踪查询、基于 Span 的指标生成以及与日志和指标的关联。Tempo 默认使用 Apache Parquet 为基础的列式块格式，提高了查询性能，并使下游工具能够访问追踪数据。查询通过 TraceQL 和 Tempo CLI 执行。Grafana Alloy 也可以配置以收集并转发追踪数据到 Tempo。我们的团队在 GKE 上自托管了 Tempo，使用 MinIO 作为对象存储，结合 OpenTelemetry 收集器以及 Grafana 用于追踪数据的可视化。

30. Railway

试验

Heroku 曾是许多开发者快速发布和部署应用程序的优秀选择。近年来，我们也看到了像 Vercel 这样更现代、轻量且易用的平台的兴起，虽然它们主要面向前端应用的部署。而在全栈部署领域的一个替代选择是 Railway，这是一个云端 PaaS 平台，简化了从 GitHub/Docker 部署到生产环境可观测性的整个流程。

Railway 支持大多数主流编程框架、数据库以及容器化部署。作为应用程序的长期托管平台，您可能需要仔细比较不同平台的成本。目前，我们的团队对 Railway 的部署和可观测性体验良好。其操作流畅，并且能够很好地与我们倡导的 持续部署 实践相结合。

31. Unblocked

试验

Unblocked 是一款现成的 AI 团队助手。通过与代码库、企业文档平台、项目管理工具以及沟通工具的集成，Unblocked 能帮助解答关于复杂业务和技术概念、架构设计与实现以及操作流程的问题。这在处理大型或遗留系统时尤为有用。在使用 Unblocked 的过程中，我们观察到团队更重视快速获取与代码和用户故事相关的上下文信息，而非生成代码或用户故事。对于这些生成任务，特别是编码场景，软件工程代理 更为适合。

32. Weights & Biases

试验

Weights & Biases 持续发展，自上次在技术雷达中提及以来，增加了更多面向 LLM 的功能。他们扩展了 Traces 并推出了 Weave，一个超越 LLM 系统跟踪的完整平台。Weave 支持创建系统评估、定义自定义指标、使用 LLM 作为任务（如摘要）的评判工具，并保存数据集以捕捉不同行为进行分析。这有助于优化 LLM 组件，并在本地和全局层面跟踪性能。该平台还支持迭代开发和高效调试，这对错误难以检测的代理系统尤为重要。此外，它还允许收集宝贵的人类反馈，这些反馈可以用于后续模型微调，从而进一步提升模型的表现和可靠性。

33. Arize Phoenix

评估

随着大语言模型和 AI agent 应用 的日益普及，LLM 可观测性的重要性不断凸显。此前，我们曾推荐过 Langfuse 和 Weights & Biases（W&B）等平台。 Arize Phoenix 是该领域另一个值得关注的新兴平台，我们团队在实际使用中也取得了积极的体验。Phoenix 提供了诸如 LLM 链路追踪、评估和提示词管理等核心功能，并可与主流 LLM 提供商和开发框架无缝集成，以在低成本、低配置的情况下实现对 LLM 输出内容、延迟和 Token 消耗等指标的深度洞察。目前我们仅限于对其开源工具的使用经验，而更全面的商业版 Arize 平台拥有更多高级能力，我们也期待未来对此进行进一步探索。

34. Chainloop

评估

Chainloop 是一个开源的软件供应链安全平台，帮助安全团队强制执行合规性要求，同时允许开发团队将安全合规无缝集成到 CI/CD 流水线中。它包括一个控制平面（Control Plane），作为安全策略的单一事实来源，以及一个 CLI，用于在 CI/CD 工作流 中运行声明（attestations）以确保合规性。安全团队可以定义 工作流契约（Workflow Contracts），明确需要收集哪些工件（如 SBOM 和漏洞报告）、存储位置以及如何评估合规性。Chainloop 使用 OPA 的策略语言 Rego 验证声明，例如确保 CycloneDX 格式的 SBOM 符合版本要求。在工作流执行过程中，安全工件（如 SBOM ）会附加到声明中，并推送到控制平面进行强制执行和审计。此方法确保可以一致且大规模地实施合规性，同时最大限度地减少开发工作流中的摩擦。最终，它实现了一个符合 SLSA 三级标准的单一事实来源，用于元数据、工件和声明的管理。

35. DeepSeek R1

评估

DeepSeek-R1 是 DeepSeek 推出的第一代 推理模型 。在一系列非推理模型的基础上，DeepSeek 的工程师设计并应用了多种方法来最大化硬件使用率。这些方法包括多头潜在注意力（Multi-Head Latent Attention, MLA）、专家混合（Mixture of Experts, MoE）门控、8 位浮点训练（FP8）以及底层 PTX 编程。这些方法结合其 高性能计算协同设计 方法使 DeepSeek-R1 在显著降低训练和推理成本的同时，达到与最先进模型（state-of-the-art）相媲美的表现。

DeepSeek-R1-Zero 另一个显著创新在于：工程师们可以通过简单的强化学习（RL），无需监督微调（SFT）即可让非推理模型展现出推理能力。此外，所有的 DeepSeek 模型都为开放权重，即它们可以被自由获取，但训练代码和训练数据仍然为专有。该代码库还包括六个从 DeepSeek-R1 蒸馏而来的稠密模型，基于 Llama 和 Qwen 构建，其中的 DeepSeek-R1-Distill-Qwen-32B 在多个基准测试中超越了 OpenAI-o1-mini。

36. Deno

评估

由 Node.js 的发明者 Ryan Dahl 创建的 Deno ，旨在修正他认为 Node.js 存在的错误。Deno 具有更严格的沙盒机制、内置的依赖管理以及原生的 TypeScript 支持——这也是对其用户群体的重要吸引力。许多开发者在 TypeScript 项目中更偏爱 Deno，因为它更像是一个真正的 TypeScript 运行时和工具链，而不仅仅是 Node.js 的一个附加组件。

自从被列入 2019 年技术雷达 以来，Deno 取得了显著进展。Deno 2 版本 引入了对 Node.js 和 npm 库的兼容性支持，并推出了长期支持（LTS）版本及其他改进。此前，阻碍 Deno 采用的主要障碍之一是需要重写 Node.js 应用程序，而这些更新降低了迁移的难度，同时扩展了对相关工具和系统的依赖选项。鉴于 Node.js 和 npm 庞大的生态系统，这些变化有望进一步推动 Deno 的普及。

此外，Deno 的 标准库 已趋于稳定，有助于遏制 npm 生态中过多低价值软件包的泛滥。Deno 提供的工具链和标准库，使 TypeScript 或 JavaScript 在服务器端开发中更具吸引力。然而，我们也提醒开发者，不应仅仅为了避免多语言编程而选择某个平台。

37. Graphiti

评估

Graphiti 构建了动态且时间感知的知识图谱，能够捕捉不断演化的事实和关系。我们的团队使用 GraphRAG 来挖掘数据之间的关系，从而提升检索与响应的准确性。在数据集不断变化的情况下，Graphiti 在图的边上维护了时间元数据，用以记录关系的生命周期。它能够将结构化和非结构化数据以离散的 Episodes 形式进行摄取，并支持基于时间、全文检索、语义和图算法的查询融合。 对于基于 LLM 的应用程序——无论是 RAG 模式还是 Agentic 方法——Graphiti 都能够支持长期记忆和基于状态的推理。

38. Helicone

评估

类似于 Langfuse、Weights & Biases 和 Arize Phoenix，Helicone 是一个面向企业需求的托管 LLMOps 平台，专注于管理 LLM 成本、评估 ROI 和降低风险。作为一个开源且以开发者为中心的平台，Helicone 支持生产级 AI 应用，覆盖整个 LLM 生命周期的提示词实验、监控、调试和优化。它能够实时分析来自不同 LLM 提供商的成本、利用率、性能以及代理堆栈跟踪。虽然 Helicone 大大简化了 LLM 运维管理，但作为一个正在发展的平台，它可能需要一定的专业知识才能充分利用其先进功能。我们的团队目前正在使用该平台，并获得了良好的体验。

39. Humanloop

评估

Humanloop 是一个新兴的平台，致力于通过在关键决策点引入人类反馈，使 AI 系统更加可靠、适应性更强并且更符合用户需求。平台提供了用于人工标注、主动学习和人工参与的微调工具，同时支持根据业务需求对大语言模型进行评估。此外，它还帮助优化生成式 AI 解决方案的生命周期，以更高的成本效益实现更大的控制和效率。Humanloop 支持通过共享工作区、版本控制的提示管理以及 CI/CD 集成进行协作，从而有效防止回归。它还提供了可观测性功能，例如追踪、日志记录、警报和安全防护，用于监控和优化 AI 系统性能。这些功能使其在部署 AI 于高风险或受监管领域的组织中尤为重要，在这些领域中，人类监督是关键。凭借对负责任 AI 实践的关注，Humanloop 对于希望构建可扩展且符合伦理要求的 AI 系统的团队来说，值得认真评估。

40. 模型上下文协议（MCP）

评估

在提示生成中，最大的挑战之一是确保 AI 工具能够访问与任务相关的所有上下文信息。这些上下文信息通常已经存在于我们每天使用的系统中，如维基、问题追踪器、数据库或可观察性系统。AI 工具与这些信息源的无缝集成可以显著提高 AI 生成输出的质量。

模型上下文协议（MCP）是由 Anthropic 发布的开放标准，提供了一个标准化的框架，用于将 LLM 应用与外部数据源和工具集成。它定义了 MCP 服务器和客户端，服务器访问数据源，客户端则集成并使用这些数据来增强提示。许多编码助手已经实现了 MCP 集成，使其能够作为 MCP 客户端运行。MCP 服务器可以通过两种方式运行：本地运行，作为在用户机器上运行的 Python 或 Node 进程；或者远程运行，作为通过 SSE 连接的服务器（尽管我们尚未看到远程服务器的使用）。目前，MCP 主要以第一种方式使用，开发者通过克隆开源的 MCP 服务器 实现 来使用它。虽然本地运行的服务器提供了一种避免第三方依赖的简洁方式，但它们对于非技术用户仍然不够友好，并且带来了治理和更新管理等挑战。尽管如此，可以预见这一标准未来可能会发展成一个更成熟、更易于用户使用的生态系统。

41. Open WebUI

评估

Open WebUI 是一个开源的自托管 AI 平台，功能多样且强大。它支持兼容 OpenAI 的 API，并能够与 OpenRouter 和 GroqCloud 等提供商集成。通过 Ollama，它可以完全离线运行，连接到本地或 自托管 的模型。Open WebUI 内置了 RAG（检索增强生成）功能，让用户可以在聊天驱动的体验中与本地或基于网络的文档互动。平台提供了细粒度的 RBAC（基于角色的访问控制）功能，可以为不同用户组配置不同的模型和平台能力。此外，Open WebUI 支持通过 Functions 进行扩展，这些基于 Python 的构建模块能够自定义和增强平台功能。一个重要功能是其 模型评估 系统，其中包含用于在特定任务上对比 LLM 的模型竞技场。Open WebUI 可根据不同需求部署于各类场景——无论是个人 AI 助手、团队协作助手，还是企业级 AI 平台，都能灵活适配。

42. pg_mooncake

评估

pg_mooncake 是一个 PostgreSQL 扩展，用于添加列式存储和向量化执行功能。其列存表可存储为 Iceberg 或 Delta Lake 表格，数据可以存放在本地文件系统或兼容 S3 的云存储中。pg_mooncake 支持从 Parquet、CSV 文件甚至 Hugging Face 数据集加载数据，非常适合需要列式存储的重度数据分析场景。它消除了在技术栈中添加专用列式存储技术的需求，为数据密集型分析提供了高效解决方案。

43. 推理模型（Reasoning Models）

评估

自上次雷达发布以来，推理模型（Reasoning Models）的突破和普及是人工智能领域最重要的进展之一。这些模型，也被称为“思考模型”，在诸如前沿数学和编码等 基准测试 中，它们已达到人类顶级水平的表现。

推理模型通常通过强化学习（RL）或监督式微调（SFT）进行训练，增强了诸如逐步思考（思维链）、探索替代方案（思维树）和 自我修正 等能力。典型代表包括 OpenAI 的 o1 / o3 、DeepSeek R1 和 Gemini 2.0 Flash Thinking 。然而，这些模型应被视为与通用大型语言模型（LLM）不同的类别，而非简单的高级版本。

这种能力提升伴随着代价。推理模型需要更长的响应时间和更高的 token 消耗，因此我们戏称它们为“ 更慢的 AI ”（如果当前的 AI 还不够慢的话）。并非所有任务都值得采用这类模型。对于文本摘要、内容生成或快速响应聊天机器人等简单任务，通用 LLM 仍然是更好的选择。我们建议在 STEM 领域、复杂问题解决和决策制定中使用推理模型——例如，将 LLM 用作评判者或通过推理模型显式的 CoT 输出来提高最终结果的可解释性。截至撰写本文时，混合推理模型 Claude 3.7 Sonnet 已发布，暗示了传统 LLM 和推理模型之间融合的可能性。

44. Restate

评估

Restate 是一个持久化执行平台，类似于 Temporal ，由 Apache Flink 的原始创始人开发。功能方面，它提供了将工作流作为代码、状态事件处理、Saga 模式和持久化状态机等特性。Restate 使用 Rust 编写，并作为单个二进制文件部署，利用分布式日志记录事件，并通过基于 Flexible Paxos 的虚拟共识算法来实现，这保证了在节点故障时的持久性。平台提供了 Java、Go、Rust 和 TypeScript 等常见语言的 SDK。我们仍然认为，在分布式系统中最好避免使用分布式事务，因为这会带来额外的复杂性和不可避免的运维开销。然而，如果在你的环境中无法避免分布式事务，这个平台是值得评估的。

45. Supabase

评估

Supabase 是一个开源的 Firebase 替代方案，用于构建可扩展且安全的后端。它提供了一整套集成服务，包括 PostgreSQL 数据库、认证、即时 API、Edge Functions、实时订阅、存储以及向量嵌入。 Supabase 的目标是简化后端开发，使开发者能够专注于构建前端体验，同时利用开源技术的强大功能和灵活性。与 Firebase 不同，Supabase 基于 PostgreSQL 构建。如果您正在进行原型设计或开发 MVP（最小可行产品），Supabase 值得考虑，因为在原型阶段之后，它更容易迁移到其他 SQL 解决方案。

46. Synthesized

评估

在软件开发中，一个常见的挑战是为开发和测试环境生成测试数据。理想情况下，测试数据应尽可能接近生产环境，同时确保不暴露任何个人身份信息或敏感信息。虽然这看似简单，但测试数据的生成却远非易事。这也是我们对 Synthesized 感兴趣的原因——一个可以屏蔽和子集化现有生产数据，或生成具有统计相关性的合成数据的平台。Synthesized 可直接集成到构建流水线中，并提供隐私屏蔽功能，通过不可逆的数据混淆技术（如哈希、随机化和分组）实现逐属性匿名化。此外，它还可以生成大量合成数据用于性能测试。尽管该平台包含了当下流行的生成式 AI 功能，但其核心功能针对开发团队长期存在的一个真实挑战，值得进一步探索。

47. Tonic.ai

评估

Tonic.ai 属于一类日益增长的平台，专注于为开发、测试和 QA 环境生成真实且去标识化的合成数据。与 Synthesized 类似，Tonic.ai 提供一套全面的工具，满足各种数据合成需求，这与 Synthetic Data Vault 更偏向于库的方式形成对比。Tonic.ai 能生成结构化和非结构化数据，在保持生产数据统计属性的同时，通过差分隐私技术确保隐私和合规。其关键功能包括自动检测、分类和去除非结构化数据中的敏感信息，以及通过 Tonic Ephemeral 按需提供数据库环境。此外，Tonic Textual 是一个安全的数据湖解决方案，帮助 AI 开发人员利用非结构化数据支持 检索增强生成（RAG）系统和 LLM 微调。对于希望在严格的数据隐私要求下加速工程开发并生成可扩展且真实数据的团队，Tonic.ai 是一个值得评估的平台。

48. turbopuffer

评估

turbopuffer 是一个无服务器的多租户搜索引擎，能够在对象存储上无缝集成向量搜索和全文搜索。我们非常欣赏其架构和 设计选择 ，尤其是在耐久性、可扩展性和成本效率方面的专注。通过将对象存储用作预写日志并保持查询节点的无状态化，它非常适合高规模的搜索工作负载。

turbopuffer 专为性能和准确性而设计，开箱即可提供 高召回率，即使是复杂的基于过滤条件的查询也不例外。它将冷查询结果缓存到 NVMe SSD，并将经常访问的命名空间保存在内存中，从而实现对数十亿文档的低延迟搜索。这使其非常适合 大规模 文档检索、向量搜索以及 检索增强生成（RAG）等 AI 应用。然而，其对对象存储的依赖也带来了查询延迟的权衡，使其在需要无状态分布式计算的工作负载中最为高效。turbopuffer 驱动了诸如 Cursor 这样的大规模生产系统，但目前仅通过 推荐或邀请 方式获得访问权限。

49. VectorChord

评估

VectorChord 是一个用于向量相似性搜索的 PostgreSQL 扩展，由 pgvecto.rs 的创建者开发，作为其继任者。它是开源的，与 pgvector 数据类型兼容，并且专为磁盘高效和高性能的向量搜索而设计。它采用 IVF（倒排文件索引）以及 RaBitQ 量化技术，能够实现快速、可扩展且准确的向量搜索，同时显著降低计算需求。与该领域其他 PostgreSQL 扩展一样，它利用了 PostgreSQL 的生态系统，使向量搜索能够与标准事务操作并行执行。尽管 VectorChord 仍处于早期阶段，但对于向量搜索的工作负载，值得您进行评估。

50. Tyk hybrid API management

暂缓

我们观察到多个团队在使用 Tyk hybrid API management 解决方案时遇到了问题。虽然托管控制平面和自管数据平面的设计理念为复杂的基础设施（如多云和混合云）提供了灵活性，但团队反馈在 Tyk 的 AWS 托管环境中，控制平面发生的故障通常是通过内部发现的，而非由 Tyk 主动检测到，这暴露了 Tyk 在可观测性方面的潜在不足。此外，事故支持的响应速度似乎偏慢，仅通过工单和邮件进行沟通在这些紧急情况下并不理想。同时，团队还反映 Tyk 文档的成熟度不足，尤其在处理复杂场景和问题时，文档往往不能提供足够的指导。此外，Tyk 生态系统中的其他产品似乎也不够成熟。例如，企业开发者门户被报告为不向后兼容，且定制能力有限。特别是在 Tyk 的混合部署场景中，我们建议谨慎使用，并将继续关注其成熟度的发展。

工具

采纳

51. Renovate
52. uv
53. Vite

试验

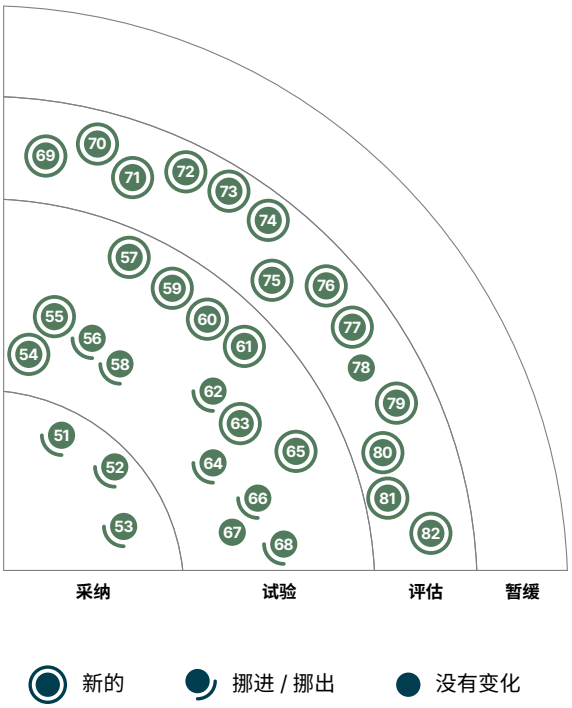
54. Claude Sonnet
55. Cline
56. Cursor
57. D2
58. Databricks Delta Live Tables
59. JSON Crack
60. MailSlurp
61. Metabase
62. NeMo Guardrails
63. Nyx
64. OpenRewrite
65. Plerion
66. 软件工程代理(software engineering agents)
67. Tuple
68. Turborepo

评估

69. AnythingLLM
70. Gemma Scope
71. Hurl
72. Jujutsu
73. kubenetmon
74. Mergiraf
75. ModernBERT
76. OpenRouter
77. Redactive
78. System Initiative
79. TabPFN
80. v0
81. Windsurf
82. YOLO

暂缓

—



51. Renovate

采纳

Renovate 已经变成了我们很多团队在依赖项版本管理工具的首选。虽然 Dependabot 仍然是 GitHub 仓库的一个安全默认选项，我们依然推荐评估 Renovate，因为它提供了更全面且可定制的方案。为了最大程度发挥 Renovate 的优势，应配置它来监控并更新所有依赖项，包括工具、基础设施以及私有或内部托管的依赖项。同时为了减少开发者的工作量，可以考虑 自动合并依赖更新的 PR.

52. uv

采纳

自上一次技术雷达以来，我们积累了更多关于 uv 的实践经验，并收到了团队的极大好评。uv 是一个由 Rust 编写的下一代 Python 包和项目管理工具，其核心价值主张是“极快的速度”。在基准测试中，uv 的性能远超其他 Python 包管理工具，加速了构建和测试周期，显著提升了开发者体验。除了性能，uv 还提供了统一的工具集，有效取代了像 Poetry、pyenv 和 pipx 等工具。然而，我们对包管理工具的担忧依然存在：一个强大的生态系统、成熟的社区和长期支持至关重要。由于 uv 相对较新，将其移至 Adopt 阶段是一个大胆的决定。然而，许多数据团队都渴望 摆脱 Python 的传统包管理系统。而我们的前线开发者也一致推荐 uv，认为这是目前最好的工具。

53. Vite

采纳

自从 Vite 上次出现在 radar 以来，它的影响力在进一步提升。作为一个高性能的前端构建工具，Vite 提供了快速热重载的特性。它正在被众多前端框架采用并推荐为默认选择，比如 Vue, SvelteKit，以及最近 废弃了 create-react-app 的 React。此外 Vite 最近获得了重大投资，这促成了 VoidZero 的成立。VoidZero 是一个专注于 Vite 发展的组织，这笔投资预计将加速 Vite 的开发，并提升其项目的长期可持续性。

54. Claude Sonnet

试验

Claude Sonnet 是一款擅长编码，写作，分析和视觉处理的先进语言模型。它可在浏览器，终端和大多数主流 IDE 中使用，并支持与 GitHub Copilot 集成。截至目前的基准测试显示，该模型在 3.5 和 3.7 版本中的表现显著优于早前推出的模型。它还擅长解析图表并从图片中提取文本，同时提供以开发者体验为中心的特性，譬如它在浏览器 UI 中的“Artifacts”功能，用于生成和交互动态内容（如代码片段和 HTML 设计）。

我们在软件开发中较为广泛地使用了 Claude Sonnet 的 3.5 版本，并发现它在多个项目中显著地提高了生产力。它在从零开始的项目（greenfield projects）中表现尤为出色，特别是在协同软件设计和架构讨论方面。尽管目前还难言有 AI 模型是“稳定”的编码助手，但 Claude Sonnet 已经是我们使用过的模型中最可靠的一个。截至撰写本文时，Claude 3.7 也已发布，并展现出很大潜力，但我们尚未在生产环境中进行全面测试。

55. Cline

试验

Cline 是一个开源的 VSCode 扩展程序，目前在监督型 软件工程代理 领域中是最强有力的竞争者之一。它让开发者能够完全通过 Cline 聊天来驱动实现，并与他们已经使用的 IDE 无缝集成。Cline 的核心功能包括计划与执行模式（Plan & Act mode）、透明的 token 使用跟踪以及 MCP 集成，帮助开发者高效地与 LLM 交互。Cline 展现了在处理复杂开发任务方面的高级能力，尤其是在结合 Claude 3.5 Sonnet 时表现突出。它支持处理大型代码库、自动化无头浏览器测试，并能够主动修复错误。与基于云的解决方案不同，Cline 通过 本地存储数据 增强了隐私保护。其开源特性不仅确保了更高的透明度，还支持社区驱动的改进。需要注意的是，由于 Cline 的代码上下文编排虽然高效，但资源消耗较高，因此开发者需关注 token 使用成本。另外，Cline 可能面临 速率限制 的潜在瓶颈，这可能会减慢工作流。在此问题解决之前，建议使用像 OpenRouter 这样提供更优速率限制的 API 提供商。

56. Cursor

试验

我们依然对 Cursor 这款以 AI 为核心的代码编辑器印象深刻，它在竞争激烈的 AI 代码辅助领域依然保持领先地位。其代码上下文编排功能十分高效，并支持多种模型，包括使用自定义 API 密钥的选项。Cursor 团队经常在其他厂商之前推出创新的用户体验功能，并在聊天功能中集成了丰富的上下文提供者，例如 Git 差异对比（git diffs）、先前的 AI 对话、网络搜索、库文档以及 MCP 集成等。与 Cline 和 Windsurf 等工具类似，Cursor 也因其强大的智能代理编码模式而脱颖而出。该模式允许开发者直接通过 AI 聊天界面指导实现过程，工具可以自主读取和修改文件，并执行命令。此外，我们还欣赏 Cursor 在检测生成代码的 lint 及编译错误方面的能力，它能够主动识别并进行修正。

57. D2

试验

D2 是一个开源的 图表即代码 工具，帮助用户通过文本创建和定制图表。它引入了 D2 图表脚本语言，以简单的声明式语法优先保证可读性而非紧凑性。D2 自带默认 主题，并使用与 Mermaid 相同的 布局引擎 。我们的团队非常欣赏其轻量化的语法，尤其适用于软件文档和架构图的场景。

58. Databricks Delta Live Tables

试验

Delta Live Tables（DLT）在简化和优化数据管道管理方面持续展现出其价值，通过声明式方法支持实时流处理和批量处理。通过自动化复杂的数据工程任务（如手动检查点管理），DLT 减少了运营开销，同时确保了端到端系统的稳健性。其简化管道编排的能力，几乎无需人工干预，大大提升了可靠性和灵活性。此外，像物化视图这样的功能为特定用例提供了增量更新和性能优化。

然而，团队需要深入理解 DLT 的细节，才能充分利用其优势并规避潜在的陷阱。作为一种有主见的抽象层，DLT 自行管理表数据，并限制每次仅允许单个管道插入数据。流式表仅支持追加操作，这需要在设计中仔细考量。此外，删除 DLT 管道时会同时删除底层表和数据，可能带来一定的操作性问题。

59. JSON Crack

试验

JSON Crack 是一个用于在 Visual Studio Code 中将文本数据渲染为交互式图表的扩展工具。尽管其名称中包含“JSON”，但它实际上支持包括 YAML、TOML 和 XML 在内的多种格式。与 Mermaid 和 D2 不同，这些工具通过文本形式生成特定的可视化图表，而 JSON Crack 则是一个用于直观查看以文本格式存储数据的工具。其布局算法表现良好，并且支持选择性隐藏分支和节点，是探索数据集的绝佳选择。此外，它还提供了一个基于 Web 的配套工具，但我们对在线代码格式化或解析服务的依赖仍需保持谨慎。需要注意的是，JSON Crack 有节点数量的限制，对于超过几百个节点的文件会引导用户使用其商业版工具。

60. MailSlurp

试验

测试涉及电子邮件的工作流通常复杂且耗时。开发团队需要为自动化构建自定义的电子邮件 API 客户端，同时还需要设置临时收件箱以满足手动测试场景的需求，例如在主要发布之前进行用户测试或内部产品培训。当开发客户入职产品时，这些挑战会变得更加明显。我们对 MailSlurp 的使用体验非常积极。它是一个邮件服务器和 SMS API 服务，提供用于创建收件箱和电话号码的 REST API，同时还支持直接在代码中验证电子邮件和消息。其无代码的仪表板对于手动测试准备也非常有用。此外，像自定义域名、webhook、自动回复和转发等功能在更复杂的场景中也值得一试。

61. Metabase

试验

Metabase 是一款开源的分析和商业智能工具，允许用户从各种数据源（包括关系型数据库和 NoSQL 数据库）中可视化和分析数据。该工具帮助用户创建可视化和报告，将其组织到仪表板中，并轻松分享数据洞察。此外，它还提供了一个 SDK，用于在 Web 应用程序中嵌入交互式仪表板，并能够匹配应用程序的主题和样式——这使其对开发者非常友好。通过官方支持和社区支持的数据连接器，Metabase 在不同的数据环境中表现出极大的灵活性。作为一款轻量级的 BI 工具，我们的团队发现它在管理应用程序中的交互式仪表板和报告方面非常实用。

62. NeMo Guardrails

试验

NeMo Guardrails 是 NVIDIA 提供的一个易于使用的开源工具包，可帮助开发者为用于对话式应用的大型语言模型实施“护栏”。自我们上一次在技术雷达中提到它以来，NeMo 在团队中的应用显著增加，并且不断改进。最近对 NeMo Guardrails 的更新主要集中在扩展集成能力和加强安全性、数据管理及控制方面，与该项目的核心目标保持一致。

NeMo 的 文档 进行了重大改进，提高了可用性，并新增了多个集成，包括 AutoAlign 和 Patronus Lynx ，同时支持 Colang 2.0。 关键升级包括增强了 内容安全性和安全功能 ，以及最近发布的支持通过输出轨道流式处理 LLM 内容的功能，从而提高性能。我们还看到新增了对 Prompt Security 的支持。此外，NVIDIA 还 发布 了三种新的微服务： 内容安全微服务、主题控制微服务 和 越狱检测微服务，这些微服务都已集成至 NeMo Guardrails。

基于其不断扩展的功能集和在生产中的日益广泛使用，我们将 NeMo Guardrails 的状态提升至试验（Trial）。建议查看最新的 发布说明 ，以全面了解自我们上次提到以来的所有更新内容。

63. Nyx

试验

Nyx 是一个多功能的语义化版本发布工具，支持各种软件工程项目。它对编程语言无依赖，并兼容所有主流的持续集成和源代码管理平台，具备极高的适配性。尽管许多团队在 主干开发 中使用语义化版本管理，Nyx 还支持 Gitflow、OneFlow 和 GitHub Flow 等工作流。在生产环境中，Nyx 的一大优势是其自动生成变更日志的能力，并且内置支持 Conventional Commits 规范。

如前几期技术雷达中所提到的，我们对依赖 长期分支 的开发模式（如 Gitflow 和 GitOps）持谨慎态度，因为这些模式引入了许多挑战，即使是像 Nyx 这样强大的工具也难以完全解决这些问题。我们强烈推荐在 CI/CD 工作流中尝试 Nyx，尤其是在主干开发中，我们已经多次见证其成功应用。

64. OpenRewrite

试验

OpenRewrite 一直是我们进行大规模代码重构的得力工具，尤其适用于基于规则的重构场景，例如迁移到广泛使用的库的新 API 版本，或对从相同模板创建的多个服务进行更新。除了对 Java 的强大支持外，OpenRewrite 还引入了对 JavaScript 等语言的支持。在框架（如 Angular）采用短期 LTS 发布周期的背景下，保持项目及时升级变得越来越重要，而 OpenRewrite 在这一过程中表现出色。虽然使用 AI 编程助手是另一种选择，但对于基于规则的更改，AI 通常运行较慢、成本更高且可靠性较低。我们特别欣赏 OpenRewrite 内置的丰富规则集（recipes），这些规则明确描述了需要执行的更改。其重构引擎、内置规则集以及构建工具插件均为开源软件，这为团队在需要进行大规模代码更新时提供了更大的便利和灵活性。

65. Plerion

试验

Plerion 是一个专注于 AWS 的云安全平台，通过与托管服务提供商集成，帮助发现云基础设施、服务器和应用程序中的风险、错误配置和漏洞。与 Wiz 类似，Plerion 使用基于风险的优先级策略对检测到的问题进行排序，旨在帮助用户“专注于最重要的 1% 问题”。 我们的团队对 Plerion 的使用体验非常积极，认为它为客户提供了重要的洞察力，并进一步强调了对组织实施主动安全监控的重要性。

66. 软件工程代理（Software engineering agents）

试验

自我们六个月前首次讨论 软件工程代理（software engineering agents）以来，行业内仍然缺乏对“代理（Agent）”这一术语的统一定义。然而，一个重要的进展已经浮现——并非完全自主的编码代理（其能力仍然令人怀疑）——而是在 IDE 内的监督代理模式（supervised agentic modes）。这些模式允许开发者通过聊天驱动实现，工具不仅可以修改多个文件中的代码，还能执行命令、运行测试并响应 IDE 反馈（如 linting 或编译错误）。

这种方式有时被称为“面向聊天的编程（chat-oriented programming，CHOP）”或“从提示到代码（prompt-to-code）”，它让开发者保持控制的同时，将更多责任转移给 AI，这跟传统的自动补全类辅助工具有很大不同。该领域的领先工具包括 Cursor、Cline 和 Windsurf，而 GitHub Copilot 稍显落后，但正在快速追赶。这些代理模式的有效性取决于所使用的模型（以 Claude's Sonnet 系列为当前业界领先）以及工具与 IDE 集成的深度，为开发者提供良好的体验。

我们发现这些工作流具有吸引力且潜力巨大，并显著提高了编码速度。然而，保持问题范围小有助于开发者更好地审查 AI 生成的更改。这种方法在低抽象提示以及 AI 友好的代码库 中效果最佳——这些代码库结构良好且经过充分测试。随着这些模式的改进，它们也会加剧开发者 自满于 AI 生成的代码 。为了缓解这一问题，我们建议采用结对编程和其他严格的审查实践，尤其是在生产代码中。

67. Tuple

试验

Tuple 是一款专为远程结对编程优化的工具，最初设计是为了填补 Slack 的 Screenhero 停止服务后留下的空白。自我们上次在技术雷达中提到它以来，Tuple 已获得更广泛的应用，并解决了之前的诸多问题和限制，现在还支持 Windows 平台。一个关键的改进是增强了桌面共享功能，新增的隐私功能允许用户在共享屏幕时隐藏私人应用窗口（如短信），同时专注于共享工具（例如浏览器窗口）。此前，UI 限制让 Tuple 更像是一个专用的结对编程工具，而不是通用的协作工具。随着这些更新，用户现在可以在 IDE 之外的内容上进行协作。

但需要注意的是，远程结对的伙伴可以访问整个桌面。如果没有正确配置，这可能会成为安全隐患，尤其是在结对伙伴不够值得信任的情况下。我们强烈建议在使用 Tuple 之前，教育团队了解其隐私设置、最佳实践和使用礼仪。

我们鼓励团队将最新版 Tuple 纳入开发工作流中进行尝试。它与我们的 务实的远程结对 建议一致，提供了低延迟的结对体验、直观的用户体验（UX）和显著的易用性改进。

68. Turborepo

试验

Turborepo 通过分析、缓存、并行化和优化构建任务，帮助管理大型 JavaScript 或 TypeScript monorepo，从而加速构建过程。在大型 monorepo 中，项目之间通常存在相互依赖关系；每次更改时重新构建所有依赖项既低效又耗时，而 Turborepo 简化了这一过程。与 Nx 不同，Turborepo 的默认配置使用多个 package.json 文件（每个项目一个）。这种方式允许在单个 monorepo 中包含不同版本的依赖项（如不同版本的 React），而 Nx 并不提倡这种做法。尽管这可能被视为一种反模式，但它确实能够解决某些特定用例，例如从多仓库迁移到 monorepo 的过程中，团队可能暂时需要多个版本的依赖项。根据我们的经验，Turborepo 设置相对简单且性能表现优秀。

69. AnythingLLM

评估

AnythingLLM 是一个开源桌面应用程序，可以与大型文档或内容交互，支持开箱即用的大语言模型（LLMs）和向量数据库集成。它具备可插拔的嵌入模型架构，可以与大多数商业化 LLM 以及由 Ollama 管理的开源模型一起使用。除了支持 检索增强生成（RAG）模式外，用户还可以创建和组织不同技能作为代理（agents）来执行自定义任务和工作流。AnythingLLM 允许用户将文档和交互分组到不同的工作空间中，这些工作空间类似于长生命周期的线程，每个线程都有独立的上下文。最近，它还新增了通过简单的 Docker 镜像部署为多用户 Web 应用的功能。一些团队已将其用作本地个人助手，发现它是一个强大且实用的工具。

70. Gemma Scope

评估

机械解释性（Mechanistic Interpretability）——理解大型语言模型的内部运行机制——正在成为一个日益重要的领域。像 Gemma Scope 和开源库 Mishax 这样的工具，为 Gemma2 系列开源模型提供了深入的洞察。这些解释性工具在调试模型的意外行为、识别导致幻觉、偏见或其他失败案例的组件方面发挥了关键作用，并通过提供更深入的可见性来建立对模型的信任。虽然这一领域对研究人员尤其具有吸引力，但需要注意的是，随着 DeepSeek-R1 的近期发布，模型训练正在成为超越传统大玩家的更多企业的可行选择。随着生成式 AI 的不断发展，解释性与安全性的重要性只会与日俱增。

71. Hurl

评估

Hurl 是处理一系列 HTTP 请求的利器，这些请求可通过使用 Hurl 特定语法的纯文本文件定义。除了发送请求以外，Hurl 还可以验证响应数据，确保请求返回特定的 HTTP 状态码，使用 XPATH, JSONPath 或者正则表达式断言响应头和内容，以及提取相应数据到变量之中，以便在链式请求之中调用。

凭借这些特性，Hurl 不仅可以胜任简单的 API 自动化工作，而且也能作为 API 的自动测试工具使用。它支持生成基于 HTML 或 JSON 格式的详细测试报告，这使它在测试流程中更具实用性。虽然像 Bruno 和 Postman 这样的专业工具提供了图形用户界面以及更丰富的功能，但 Hurl 更以简洁著称。和同样使用纯文本文件的 Bruno 类似，Hurl 的测试文件也可以存储在代码仓库中。

72. Jujutsu

评估

Git 是当前占据主导地位的分布式版本控制系统（VCS），拥有绝大多数的市场份额。然而，尽管 Git 已在过去十多年中占据主导地位，开发者仍然在其复杂的分支、合并、变基以及冲突解决工作流中苦苦挣扎。这种持续的挫败感催生了一系列旨在缓解痛点的工具——有些通过可视化方式简化复杂性，有些则提供图形界面以完全抽象操作过程。

Jujutsu 更进一步，提供了一个完整的 Git 替代方案，同时通过 使用 Git 仓库作为存储后端 保持兼容性。这使开发者能够继续使用现有的 Git 服务器和服务，同时受益于 Jujutsu 简化的工作流。Jujutsu 将自己定位为“既简单又强大”，强调为不同经验水平的开发者提供易用性。其一大亮点是 一流的冲突解决功能 ，这一特性有潜力显著改善开发者的使用体验。

73. kubenetmon

评估

监控和理解与 Kubernetes 相关的网络流量可能是一项挑战，尤其是当您的基础设施跨多个可用区、区域或云时。由 ClickHouse 构建并最近开源的 kubenetmon，旨在通过提供主要云提供商之间详尽的 Kubernetes 数据传输计量解决这一问题。如果您正在运行 Kubernetes 并对账单中不清晰的数据传输成本感到困扰，不妨探索一下 kubenetmon。

74. Mergiraf

评估

解决合并冲突可能是软件开发中最不受欢迎的活动之一。尽管有一些技术可以减少合并的复杂性，例如实践持续集成（按照其原意至少每天合并到共享主干），但我们仍看到在合并上花费了过多的精力。长期功能分支 是一个原因，而 AI 辅助编码也往往会增加变更集的规模。Mergiraf 可能是一个解决方案——这是一款新工具，通过查看语法树而不是将代码视为文本行来解决合并冲突。作为一个 git 合并驱动程序，它可以被配置为让 git 子命令（如 merge 和 cherry-pick）自动使用 Mergiraf，而不是默认的合并算法，从而显著提高合并效率和准确性。

75. ModernBERT

评估

BERT（Bidirectional Encoder Representations from Transformers）的继任者 ModernBERT 是一系列新一代的 encoder-only transformer 模型，专为广泛的自然语言处理（NLP）任务设计。作为一个可直接替代 BERT 的升级版本，ModernBERT 不仅提升了性能和准确性，还解决了 BERT 的一些局限——特别是通过引入“交替注意力”（Alternating Attention）实现了对极长上下文长度的支持。 对于有 NLP 需求的团队，在默认选择 通用生成式模型 之前，请优先考虑 ModernBERT。

76. OpenRouter

评估

OpenRouter 是一个统一的 API，可以用于访问多个大型语言模型（LLM）。它为 主流 LLM 提供商 提供了单一的集成点，简化了实验过程，降低了供应商锁定的风险，并通过将请求路由到最合适的模型来优化成本。像 Cline 和 Open WebUI 这样的流行工具都使用 OpenRouter 作为它们的端点。在我们的技术雷达讨论中，我们质疑大多数项目是否真的需要在模型之间切换，尤其考虑到 OpenRouter 为了盈利，在这一封装层之上需要增加价格加成。然而，我们也认识到 OpenRouter 提供了多种负载均衡策略，有助于优化成本。其一项特别有用的功能是绕过 API 速率限制。如果您的应用程序超出了单一 LLM 提供商的速率限制，OpenRouter 可以帮助您突破这一限制，实现更高的吞吐量。

77. Redactive

评估

Redactive 是一个企业级 AI 赋能平台，专为帮助受监管的组织安全地为 AI 应用（例如 AI 助手和协作工具）准备非结构化数据而设计。它可以与像 Confluence 这样的内容平台集成，创建用于 检索增强生成（RAG）搜索的安全文本索引。通过仅提供实时数据并从源系统强制执行实时用户权限，Redactive 确保 AI 模型访问的是准确且授权的信息，而不会影响安全性。此外，它还为工程团队提供工具，支持他们安全地使用任何大语言模型构建 AI 应用场景。对于正在探索 AI 驱动解决方案的组织，Redactive 提供了一种简化的数据准备和合规方法，在安全与可访问性之间取得平衡，为团队在受控环境中试验 AI 能力提供支持。

78. System Initiative

评估

我们对 System Initiative 依然感到非常兴奋。这款实验性工具为 DevOps 工作开辟了一条全新的激进方向。我们非常欣赏该工具背后富有创造性的思考，并希望它能够激励更多人突破基础设施即代码（Infrastructure-as-Code）的现状。System Initiative 目前已结束 Beta 阶段，并以 Apache 2.0 许可的形式免费开源。尽管其开发者已经在生产环境中使用该工具来管理基础设施，但它在满足大型企业需求的规模化能力方面仍有改进空间。然而，我们依然认为值得一试，以体验一种与众不同的 DevOps 工具方法。

79. TabPFN

评估

TabPFN 是一个基于 Transformer 的模型，专为在小规模表格数据集上实现快速而准确的分类而设计。它利用了上下文学习（In-Context Learning, ICL），直接从标注样本中进行预测，无需超参数调整或额外训练。TabPFN 在数百万个合成数据集上预训练，因而能够很好地泛化到不同的数据分布，同时对缺失值和异常值具有较强的处理能力。它的优势包括高效处理异构数据以及对无信息特征的鲁棒性。

TabPFN 尤其适用于对速度和准确性要求较高的小规模应用场景。然而，它在处理大规模数据集时面临扩展性挑战，并且在回归任务中能力有限。作为一项前沿解决方案，TabPFN 值得评估，尤其是在表格分类任务中，它有潜力超越传统模型，并为 Transformer 在表格数据中的应用开辟新可能性。

80. v0

评估

v0 是由 Vercel 开发的一款 AI 工具，可根据截图、Figma 设计或简单的提示生成前端代码。它支持包括 React、Vue、shadcn 和 Tailwind 在内的多种前端框架。除了 AI 生成代码的功能之外，v0 还提供了出色的用户体验，包括预览生成的代码并一步部署到 Vercel 的能力。尽管构建真实世界的应用程序通常需要集成多个功能，不仅仅局限于单个界面，但 v0 为原型设计提供了一个稳固的工具，并可作为开发复杂应用程序的起点初始化代码。

81. Windsurf

评估

Windsurf 是 Codeium 推出的 AI 编程助手，以其“代理型”（agentic）能力而闻名。类似于 Cursor 和 Cline，Windsurf 允许开发者通过 AI 聊天驱动实现代码的导航、修改以及命令的执行。它经常发布针对“代理模式”的全新功能和集成。例如，最近它推出了一个浏览器预览功能，使代理能够轻松访问 DOM 元素和浏览器控制台，还提供了一个网页研究功能，让 Windsurf 在适当情况下可以在互联网上查找文档和解决方案。Windsurf 支持多种主流 AI 模型，用户可以启用并引用网页搜索、库文档以及 MCP（Model Context Protocol）集成作为额外的上下文提供者。这些能力让 Windsurf 成为开发者高效工作的强大工具。

82. YOLO

评估

YOLO（You Only Look Once）系列由 Ultralytics 开发，并持续在推动计算机视觉模型领域的进步。其最新版本 YOLO11 在精度和效率方面比以前的版本有了显著的提升。YOLO11 可以在 极少资源消耗下高速 运行图像分类任务，这使其适用于边缘设备的实时应用。此外，我们发现使用同样的框架还可以执行姿势估计，物体检测，图像分割以及其他任务，这一特性十分强大。这一重大进展也提醒我们，“传统”的机器学习模型的表现可能比像大语言模型（LLM）等通用 AI 模型更加出色。

语言和框架

采纳

83. OpenTelemetry
84. React Hook Form

试验

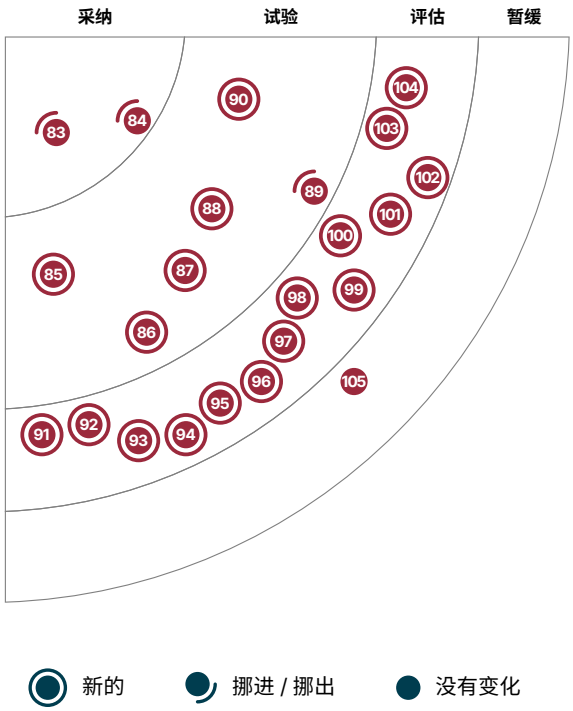
85. Effect
86. Hasura GraphQL engine
87. LangGraph
88. MarkItDown
89. Module Federation
90. Prisma ORM

评估

91. .NET Aspire
92. Android XR SDK
93. Browser Use
94. CrewAI
95. ElysiaJs
96. FastGraphRAG
97. Gleam
98. GoFr
99. Java 后量子密码学（Java post-quantum cryptography）
100. Presidio
101. PydanticAI
102. 资源受限应用中使用 Swift
103. Tamagui
104. torchtune

暂缓

105. Node 超载



83. OpenTelemetry

采纳

OpenTelemetry 正迅速成为可观察性领域的行业标准。随着 OpenTelemetry 协议（OTLP）规范的发布，行业内有了一个标准化的方式来处理追踪（traces）、指标（metrics）和日志（logs）。这减少了在监控分布式解决方案和满足互操作性需求时的多重集成或主要代码重构的需要。随着 OpenTelemetry 扩展对日志和性能分析的支持，OTLP 为所有遥测数据提供了一个一致的传输格式，简化了应用的仪表化过程，使全栈可观察性对于微服务架构更加易于实现且具有可扩展性。OTLP 已被诸如 Datadog 、New Relic 和 Grafana 等供应商采纳，帮助企业构建灵活的、与供应商无关的可观察性技术栈，避免被锁定在专有解决方案中。OTLP 支持 gzip 和 zstd 压缩，大幅减少遥测数据的大小并降低带宽使用——这对于处理高遥测数据量的环境尤为关键。为了支持长期发展，OTLP 确保 OpenTelemetry 保持强大且面向未来的标准，巩固了其作为遥测传输事实标准的地位。

84. React Hook Form

采纳

我们将 React Hook Form 视为 Formik 的替代方案。由于默认使用非受控组件，它提供了显著更好的开箱即用的表现，特别是在大型表单上。React Hook Form 很好地和各种基于 schema 的验证库（比如 Yup, Zod 等）进行了集成。此外 React Hook Form 提供了很大的灵活性，使其易于和现有代码库和其他库集成。你可以把 React Hook Form 和像 shadcn 或者 AntD 这样的外部受控组件库一起使用。凭借出色的性能、无缝集成和活跃的开发社区，它是构建大型表单或表单密集型应用的可靠选择。

85. Effect

试验

Effect 是一个强大的 TypeScript 库，用于构建复杂的同步和异步程序。在 Web 应用开发中，常常需要为异步处理、并发、状态管理和错误处理等任务编写大量样板代码。而 Effect-TS 通过采用函数式编程的方法简化了这些流程。借助 TypeScript 的类型系统，Effect 能够在编译时捕获难以检测的问题。我们的团队曾使用过 fp-ts 进行函数式编程，但发现 Effect-TS 提供的抽象更贴近日常任务的需求，同时使代码更易于组合和测试。尽管传统方法如 Promise/try-catch 或 async/await 也能处理类似场景，但在使用 Effect 之后，我们的团队发现没有理由回到传统方法。

86. Hasura GraphQL engine

试验

Hasura GraphQL engine 是一个通用的数据访问层，可简化在不同数据源上构建、运行和管理高质量 API 的过程。它能够为各种数据库（包括 PostgreSQL、MongoDB 和 ClickHouse）及数据源即时生成 GraphQL API，使开发者能够快速且安全地获取所需数据。我们发现，Hasura 在实现 服务端资源聚合 的 GraphQL 应用场景中非常易用，并已将其应用于多个 数据产品项目 中。然而，对于其强大的联合查询和统一模式管理功能，我们仍然保持 谨慎态度 。值得一提的是，Hasura 最近推出了 PromptQL 功能，允许开发者利用大语言模型（LLM）实现更自然直观的数据交互。

87. LangGraph

试验

LangGraph 是一款面向基于 LLM 的 多 agent 应用构建的编排（Orchestration）框架。与抽象程度较高的 LangChain 相比，它提供了更底层的节点（Nodes）和边（Edges）等基本原语，允许开发者精细地控制 agent 工作流、记忆管理与状态持久化。这种基于图的设计使工作流更加可控且易于定制，使得在生产级应用中的调试、扩展和维护变得更加容易。尽管其学习曲线较陡，但 LangGraph 凭借其轻量化与模块化设计，在开发 agent 应用时展现出了强大的灵活性和扩展性。

88. MarkItDown

试验

MarkItDown 能将多种格式（PDF、HTML、PowerPoint、Word）转换为 Markdown，从而增强文本的可读性和上下文保留。由于 LLM 可以从 格式化提示（如标题和章节）中获取上下文，Markdown 能够很好地保留结构以提升理解能力。在基于 RAG 的应用中，我们的团队使用 MarkItDown 将文档预处理为 Markdown 格式，确保逻辑标记（如标题、子章节）保持完整。在生成嵌入之前，结构感知的分块方法帮助保留了完整的章节上下文，从而提升了查询响应的清晰度，尤其是对于复杂文档而言。Markdown 被广泛用于文档编写，而 MarkItDown 的 CLI 工具也因此成为一个极具价值的开发者生产力工具。

89. Module Federation

试验

Module Federation 允许在微前端之间指定共享模块并实现依赖的去重。随着 2.0 版本的发布，模块联邦已经发展到可以独立于 webpack 工作。此更新引入了一些关键功能，包括联邦运行时（Federation Runtime）、全新的插件 API，以及对流行框架（如 React 和 Angular）及热门构建工具（如 Rspack 和 Vite）的支持。通过采用模块联邦，大型 Web 应用可以拆分为更小、更易管理的微前端，使不同的团队能够独立开发、部署和扩展，同时高效地共享依赖项和组件。

90. Prisma ORM

试验

Prisma ORM 是一个开源的数据库工具包，帮助 Node.js 和 Typescript 应用简化应用数据库操作。它提供了一种现代化的、类型安全的数据库访问方式，能够自动化数据库模式迁移，并提供直观的查询 API。与传统 ORM 不同，Prisma ORM 使用纯 JavaScript 对象定义数据库类型，而无需依赖装饰器或类。我们对 Prisma ORM 的使用体验非常积极，它不仅更符合 TypeScript 的开发生态，还能很好地融入函数式编程范式中。

91. .NET Aspire

评估

.NET Aspire 旨在简化开发者本地机器上分布式应用的编排工作。Aspire 允许您在本地开发环境中编排多个服务，包括多个 .NET 项目、依赖的数据库和 Docker 容器——所有这些都可以通过一条命令完成。此外，Aspire 为本地开发提供了观察工具，包括日志记录、跟踪和指标仪表板，这些工具与用于预生产或生产环境的工具解耦。这大大改善了开发者在构建、调整和调试任何系统的可观察性方面的开发体验。

92. Android XR SDK

评估

Google 和三星与高通合作推出了 Android XR, 这是一款专门为扩展现实（Extended Reality, XR）设计的操作系统。它在未来计划支持智能眼镜以及其他智能设备。大多数的安卓应用无需修改或者只需少量修改即可运行在该系统上，但它的核心理念是从零构建全新的空间应用，或者将现有的应用“空间化”。目前，全新的 Android XR SDK 被指定为此类项目的首选开发工具。Google 还提供了 开发指南 帮助开发者选择 SDK 内集成的工具与技术。目前，该 SDK 已经推出开发预览版本。

93. Browser Use

评估

Browser Use 是一个开源的 Python 库，使基于 LLM 的 AI 代理能够使用网页浏览器并访问 Web 应用。它可以控制浏览器并执行包括页面导航、输入操作和文本提取在内的各种步骤。该库支持多标签页管理，能够在多个 Web 应用之间协调执行操作。在需要 LLM 代理访问网页内容、执行操作并获取结果的场景中，该库非常有用。它支持多种 LLM，并利用 Playwright 来控制浏览器，结合视觉理解与 HTML 结构提取，以优化 Web 交互体验。该库在多代理场景中正逐渐受到关注，使代理能够协作完成涉及 Web 交互的复杂工作流。

94. CrewAI

评估

CrewAI 是一个专为构建和管理 AI 代理而设计的平台，它能让多个 AI 代理协同工作，共同完成复杂任务。我们可以将其理解为一群 AI 工作者组成的团队，每个成员都有自己的专长，并能齐心协力以达成共同目标。我们曾经在雷达中的 LLM 驱动自主代理 中提及过它。除了现有的 Python 开源库以外，CrewAI 现在还推出了企业级的解决方案，使组织可以创建基于代理的应用程序并应用于真实业务场景，在云基础设施上运行，并连接到现有的数据源（如 SharePoint 或者 JIRA）。我们已经多次使用 CrewAI 去应对生产环境中出现的问题，例如自动验证促销码，调查交易失败的原因以及处理客户支持相关的请求。在代理技术快速发展的背景下，我们对 CrewAI 的能力充满信心，因此将其归入“评估”类别。

95. ElysiaJs

评估

ElysiaJS 是一个端到端类型安全的 TypeScript Web 框架，最初主要为 Bun 设计，但也兼容其他的 JavaScript 运行环境。与 tRPC 等替代方案不同（这类方案会强制要求特定的 API 接口结构），ElysiaJS 不强制任何特定的 API 接口结构。这使得开发者可以创建符合行业标准的 API，如 RESTful、JSON: API 或 OpenAPI，同时仍能提供端到端的类型安全。ElysiaJS 在 Bun 运行环境上运行表现极好，在某些基准测试中甚至可与 Java 或 Go Web 框架相媲美。如果你在构建 backend-for-frontend（BFF），ElysiaJS 是一个尤其值得考虑的选择。

96. FastGraphRAG

评估

FastGraphRAG 是 GraphRAG 的一个开源实现，专为高检索准确性和性能而设计。它利用 个性化 PageRank 限制图导航范围，仅关注图中与查询最相关的节点，从而提升检索准确性并改善大语言模型（LLM）的响应质量。FastGraphRAG 还提供图形的可视化表示，帮助用户理解节点关系及其检索过程。该工具支持增量更新，非常适合处理动态和不断演变的数据集。针对大规模 GraphRAG 用例进行了优化，FastGraphRAG 在提升性能的同时有效降低了资源消耗。

97. Gleam

评估

Erlang/OTP 是一个强大的平台，用于构建高并发、可扩展且具备容错能力的分布式系统。传统上，其语言是动态类型的，而 Gleam 在语言层面引入了类型安全。Gleam 构建于 BEAM 之上，将函数式编程的表达能力与编译时的类型安全相结合，从而减少运行时错误并提高可维护性。Gleam 拥有现代化的语法，与 OTP 生态系统高度集成，充分利用了 Erlang 和 Elixir 的优势，同时确保了强大的互操作性。Gleam 社区活跃且友好，我们期待它的持续发展。

98. GoFr

评估

GoFr 是一个专为构建 Golang 微服务而设计的框架，通过抽象常见的微服务功能（如日志记录、追踪、指标、配置管理和 Swagger API 文档生成）来简化开发工作。它支持多种数据库，处理数据库迁移，并且能够与 Kafka 和 NATs 等消息代理进行 pub/sub 操作。此外，GoFr 还包括支持定时任务的 cron 作业功能。该框架旨在降低构建和维护微服务的复杂性，让开发者能够将更多精力集中于业务逻辑的编写，而非基础设施的管理。尽管市面上已有许多流行的 Go 库用于构建 Web API，但 GoFr 正在逐步获得关注，非常值得基于 Golang 的微服务开发团队探索和使用。

99. Java后量子密码学（Java post-quantum cryptography）

评估

非对称加密——这一保障了多数现代通信安全的核心概念——依赖于数学上难以求解的问题。然而，今天使用的算法中的问题在量子计算机面前将变得容易解决，这推动了替代方案的研究。基于格的密码学 是其中最被看好的候选技术。尽管与密码学相关的量子计算机仍需多年以后才能面世，对于那些需要长期保证安全的应用来说，后量子密码学已成为一个值得关注的领域。此外还存在一种风险，攻击者可能会记录当前的加密数据并等待量子计算机推出后破解。

Java 后量子密码学在 JDK 24 中迈出了第一步，该版本预计在 3 月底正式发布。此次更新引入了 JEP 496 和 JEP 497 ，分别实现了一种密钥封装机制以及一套数字签名算法。二者均基于标准并设计为可以抵挡未来可能的量子计算攻击。虽然来自 Open Quantum Safe 项目的 liboqs 已经提供了基于 C 语言的实现并提供了 JNI 封装，但原生 Java 版本实现的出现依旧鼓舞人心。

100. Presidio

评估

Presidio 是一个数据保护 SDK，用于在结构化和非结构化文本中 识别 和 匿名化 敏感数据。它可以通过命名实体识别（NER）、正则表达式和基于规则的逻辑检测个人身份信息（PII），如信用卡号、姓名和位置等。Presidio 支持 自定义 PII 实体识别和去标识化，使企业能够根据其特定的隐私要求进行调整。尽管 Presidio 能够自动识别敏感信息，但它并非万无一失，可能会遗漏或误识别数据。在依赖其结果时，务必保持谨慎。

101. PydanticAI

评估

随着构建基于 LLM 的应用程序以及代理的技术的快速发展，构建和编排这种应用的框架往往难以跟上步伐，或者找到合适的长期适用的抽象。PydanticAI 是该领域的最新成员，旨在简化实现过程的同时，避免不必要的复杂度。它由著名的 Pydantic 的开发者们打造，并吸收了早期框架的经验——其中的很多框架已经依赖 Pydantic。PydanticAI 没有尝试成为一个多功能工具，而是提供了一个轻量级但是功能强大的方案。它不仅兼容主流的模型 API，而且内置了 结构化输出处理 的特性，并且引入了基于图形的抽象层来管理复杂的代理工作流。

102. 资源受限应用中使用 Swift

评估

自从 Swift 6.0 发布以来，这门语言已经超越了 Apple 生态系统的限制，通过对主要操作系统的改进支持，使在 资源受限应用中使用 Swift 变得更加可行。传统上，这一领域主要由 C、C++ 以及近年来的 Rust 占据，因为它们具备低级别的控制、高性能以及符合诸如 MISRA、ISO 26262 和 ASIL 等标准的认证编译器和库。尽管 Rust 已逐步获得类似的认证，但 Swift 尚未开始这一认证过程，这限制了其在安全关键型应用中的使用。

Swift 的日益普及，得益于其在性能与安全特性之间的良好平衡，包括强类型安全和自动引用计数（ARC）内存管理功能。虽然 Rust 的所有权模型提供了更强的内存安全保证，但 Swift 提供了一种不同的权衡方式，这种方式对某些开发者来说更加易于接受。Swift 和 Rust 都基于 LLVM/Clang 编译器后端，这使得一方的技术进步也能惠及另一方。凭借其将代码编译为优化机器码的能力、开源的开发模式以及不断扩展的跨平台支持，Swift 正逐步成为应用范围更广的有力竞争者——远远超越了其最初的 iOS 根基。

103. Tamagui

评估

Tamagui 是一个用于高效共享 React Web 和 React Native 样式的库。它提供了一个 设计系统，包含可复用的已样式化和未样式化组件，可以在多个平台上无缝渲染。其可选的 优化编译器 通过将样式化组件转换为网页上的原子化 CSS 和原生视图上的提升样式对象，显著提升了性能。

104. torchtune

评估

torchtune 是一个专为 PyTorch 设计的库，用于编写、后训练以及实验性探索大语言模型。它支持单 GPU 和多 GPU 设置，并通过 FSDP2 实现分布式训练。该库提供基于 YAML 的 recipes（配方），用于微调、推理、评估以及量化感知训练等任务。每个配方都聚焦于特定功能，避免复杂的参数标志配置，注重代码清晰性而非过度抽象化。此外，torchtune 包含一个强大的 CLI，可用于高效地下载模型、管理配方和运行实验。

105. Node 超载

暂缓

几年前，我们观察到了一种现象，即 Node 超载：Node.js 经常被用在一些不合理的场景中，甚至在没有考虑其他替代方案的情况下就被选择了。尽管我们理解某些团队倾向于使用单一语言栈——即便要付出一些权衡的代价——但我们仍然倡导 多语言编程（Polyglot Programming）。当时，我们指出 Node.js 因在 IO 密集型工作负载中的高效性而享有应得的声誉，但也提到其他框架已经赶上，提供了更好的 API 和更优越的整体性能。同时，我们警告说，Node.js 从未适合计算密集型工作负载，这一局限性至今仍然是一个重大的挑战。如今，随着数据密集型工作负载的兴起，我们看到团队在应对这些问题时也面临着越来越大的困难。

想要了解技术雷达最新的新闻和洞见?

点击订阅，以接收每两个月一次、来自 Thoughtworks 的技术洞察和未来趋势探索邮件。

现在订阅

Thoughtworks 是一家全球性软件及技术咨询公司，集战略、设计和工程技术咨询服务于一体，致力于推动数字创新。我们在 19 个国家 / 地区的 48 个办公室拥有超过 10,500 名员工。 在过去的 30 年里，我们为全球各地的众多合作伙伴倾力服务，与客户一起创造了非凡的影响力，帮助他们以技术为优势解决复杂的业务问题。

/thoughtworks

Design. Engineering. AI.