/thoughtworks

Design. Engenharia. IA.



Technology Radar

Um guia de opinião sobre o universo de tecnologia atual

Volume 35
Novembro 2026

Sobre o Radar 3
Radar em um relance 4
Contribuidoras 5
Créditos de produção 6
Temas 8
O Radar 11
Técnicas 14
Plataformas 32
Ferramentas 42
Linguagens e Frameworks 56

Sobre o Radar

Thoughtworkers são pessoas apaixonadas por tecnologia. Nós desenvolvemos, pesquisamos, testamos, contribuímos com código livre, escrevemos sobre e visamos a sua constante melhoria — para todas as pessoas. Nossa missão é liderar e promover a excelência de software e revolucionar a indústria de TI. Nós criamos e compartilhamos o Technology Radar da Thoughtworks para apoiar essa missão. O Conselho Consultivo de Tecnologia (Technology Advisory Board, ou TAB), um grupo de lideranças experientes em tecnologia da Thoughtworks, é responsável por criar o Radar. O grupo se reúne regularmente para discutir a estratégia global de tecnologia da empresa e as tendências tecnológicas que impactam significativamente a nossa indústria.

O Radar capta o resultado das discussões do TAB em um formato que procura oferecer valor a uma ampla gama de pessoas interessadas, de pessoas que desenvolvem software a CTOs. O conteúdo é concebido para ser um resumo conciso.

Nós encorajamos você a explorar essas tecnologias. O Radar é gráfico por natureza, agrupando os itens em técnicas, ferramentas, plataformas, linguagens e frameworks. No caso de itens que podem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado.

Além disso, agrupamos esses itens em quatro anéis para refletir nossas opiniões atuais em relação a cada um. Para mais informações sobre o Radar, acesse: thoughtworks.com/pt/radar/faq.



© Thoughtworks, Inc. All Rights Reserved.

Radar em um relance

A ideia por trás do Radar é rastrear tópicos interessantes, que chamamos de blips. Organizamos blips no Radar usando duas categorias: quadrantes e anéis. Os quadrantes representam as diferentes naturezas dos blips. Os anéis indicam nossa recomendação para utilizar a tecnologia.

Um blip é uma tecnologia ou técnica que desempenha um papel no desenvolvimento de software. Os blips estão "em movimento" — suas posições no Radar estão constantemente mudando — geralmente indicando que nossa confiança em recomendá-los tem crescido à medida que se movimentam entre os anéis.



Adote: Acreditamos firmemente que a indústria deveria adotar esses itens. Quando apropriado, nós os usamos em nossos projetos.

Experimente: Vale a pena ir atrás. É importante entender como desenvolver essa capacidade. As empresas devem testar a tecnologia em um projeto que possa lidar com o risco.

Avalie: Vale explorar com o objetivo de compreender como afetará sua empresa.

Cautela: Proceda com cuidado. Esses items envolvem receios significativos que exigem uma avaliação criteriosa alinhada ao seu contexto.

Novo **Mudança de anel**

Sem alterações

Nosso Radar é um olhar para o futuro. Para abrir espaço para novos itens, apagamos itens que não foram modificados recentemente, o que não é um reflexo de seus valores, mas uma solução para nossa limitação de espaço.

© Thoughtworks, Inc. All Rights Reserved.

Contribuidoras

O Conselho Consultivo de Tecnologia (TAB) é um grupo formado por 18 tecnologistas experientes da Thoughtworks. O TAB se reúne duas vezes por ano pessoalmente e quinzenalmente por vídeochamada. Sua principal atribuição é ser um grupo consultivo para nossa CTO Rachel Laycock.

O TAB atua examinando tópicos que afetam soluções de tecnologia e tecnologistas da Thoughtworks. Esta edição do Thoughtworks Technology Radar é baseada em uma reunião do TAB realizada no Recife, em setembro de 2026.



Rachel Laycock (CTO)



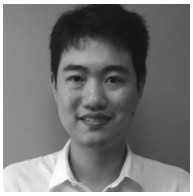
Alessio Ferri



Brandon Cook



Cecilia Geraldo



Chris Chakrit Riddhagni



Chris Ford



Effy Elden



Kief Morris



Muhilvarnan V



Ken Mugrage



Nati Rivera



Ni Wang



Renan Martins



Selvakumar Natesan



Shangqi Liu



Thiago Gregorio



Vanitha Kumar



Vanya Seth

© Thoughtworks, Inc. All Rights Reserved.

Créditos de produção

Equipe Editorial

- **Ken Mugrage**
  Technical Editor
- **Nati Rivera**
  Product Manager
- **Debashri Dutta**
  Project and Campaign Manager
- **Osama Shaikh**
  Project and Campaign Manager
- **Richard Gall**
  Content Editor
- **Kriti Anand**
  Content Editor
- **Michael Koch**
  Copy Editor
- **Gareth Morgan**
  Head of Content and Thought Leadership

Experiência digital

- **Rashmi Naganur**
  Business Analyst
- **Brigitte Britten-Kelly**
  Digital Content Strategist
- **Vandita Kamboj**
  UX Designer
- **Lohith Amruthappa**
  Analytics Specialist
- **Mukilan Kesavan**
  Developer Lead

Tradução — Português

- **Isabella Velleda**
- **Ayrton Araujo**
- **Carol Soares**
- **Catarina Bressan**
- **Geraldo Castro**
- **Gustavo de Paula Andrade**
- **Maísa Mendes**
- **Mayck Xavier**
- **Pedro Cavalcante**
- **Rafael Alves**
- **Renata Silva**
- **Vinícius Félix**

Design e multimídia

- **Leticia Nunes**
  Visual Designer
- **Sruba Deb**
  Visual Designer
- **Kevin Barry**
  Multimedia Specialist
- **Ryan Cambage**
  Multimedia Specialist
- **Anish Thomas**
  Multimedia Designer

Comunicação

- **Shalini Jagadish**
  Internal Communications Specialist
- **Yasmin Brussulo**
  Internal Communications Specialist
- **Abhishek Kasegaonkar**
  Social Media Specialist
- **Marsh Abraham**
  Public Relations Specialist
- **Anushree Tapuriah**
  Campaigns and Advertisement Specialist

Time de desenvolvimento

- **Nati Rivera**
  Product Manager
- **Alejandro Batanero**
  Technical Lead
- **Raluca Iordache**
  Developer
- **Nacho Martínez**
  Developer

Infraestrutura de marketing

- **Sahil Gupta**
  Marketing Automation Specialist

© Thoughtworks, Inc. All Rights Reserved.

Créditos de produção

Contribuições de Thoughtworkers

Índia

- Abdul Jeelani
- Akash Dixit
- Akshar Srivastava
- Anil EM
- Anitha Lakshmi V
- Ankit Adlakha
- Ankita Verma
- Anusha Panyala
- Balaji G Sunil Kumar
- Birlla P
- Chetan
- Daniel Euchar
- Dhananjay Shelke
- Deepthi R
- Jonathan Cole
- Karthikeyan Vijayan
- Krishnaswamy Subramanian
- Kumar Sankara Iyer
- Lucky Bajaj
- Maheshkumar Kharade
- Malini S
- Mayank Jain
- Mayank Kumar
- Nikhil Mane
- Nithya Sivakumar
- Pallavi Sinha
- Partha Dash
- Piyush Bhargava
- Pranesh Pokale
- Pritam Kumar
- Raghav Jain
- Rahul Garg
- Rajasekaran Gopal
- Rajat Lakhina
- Saiprabhanj Turaga
- Sakthe Karthika
- Sarath Kumar
- Saurabh Machave
- Shadayani Varshney
- Shanmugapriya Srinivasan
- Stevin Sony
- Sudhir Yelikar
- Suman Maity
- Sunit Parekh
- Sureshkumar V
- Vinoth Kumar

Europa

- Abel Guillen
- Alejandro Batanero
- Alejandro Torres Hernández
- Andrew Harmel-Law
- Arif Wider
- Ayhan Tasyurt
- David Rubio
- Gaurav Patole
- Ivan Vidal
- Jose (Pepe) S. Gomez de Olea
- Kiran Prakash
- Luca Giuberti
- Marco Baroetto Parisi
- Martin Kramer
- Matteo Vaccari
- Moritz Wilke
- Otto Tasnadi
- Raquel Devesa
- Ricardo Piccoli
- Rieke Heinze
- Scott Davies
- Sergio Jimenez
- Shweta Justice
- Tamara Carpintero
- Yuvaraj Birari

APAC

- Bojian Chen
- Guangyi Li
- Siyuan Li
- Xiaoming Gao
- Yijie
- Amit Singh
- Bank Thanayut Tiratatri
- Kaiwanyawit Chawankul
- Kelvin Nicholson
- Lilly Ryan
- Manish Joshi
- Michael Joyce
- Naaman Campbell
- Natthapong Intharak
- Neil Price
- Nguyen Viet Son
- Noor Yeaser Khan
- Rose Aunaetitrakul
- Salah Chalermthai
- Santi Liwsakphaiboon
- Thomas Hoang

Américas

- Aravind Kunnath
- Daniel Marinho
- Daniel Romero
- Daniel Santibañez
- Danilo Ferreira
- Enderson Cândido
- Gustavo Rivas
- Jipson Medardo Saad
- Loreto Sanchez
- Luiz Miccieli
- Marco Vela
- Maria Augusta Mateus
- Maria Rosa
- Milton Oliveira
- Nicolas Baquero
- Pablo Sosa
- Rocio Mera
- Sebastián Santiago
- Sebastián Stuardo
- Sergio Jara
- Varun V M
- Victoria Souza
- Yani Ovando

© Thoughtworks, Inc. All Rights Reserved.

Temas



Engenharia de harness em escala

Discutimos se a engenharia de harness deveria aparecer como um blip individual no Radar, mas dado que é um tema que conecta muitas técnicas e tecnologias, decidimos destacá-lo como tal neste volume.

Embora o termo não seja novo, esta edição destaca um ecossistema crescente de práticas e ferramentas que mostra como a engenharia de harness (também chamada de engenharia de estruturas de agentes) está se consolidando como uma disciplina. Ela abrange o conjunto de componentes e práticas que estruturam o ambiente de um agente de IA, da memória e do contexto aos ciclos de feedback e guardrails. Em particular, um ecossistema emergente de plugins está tomando forma, ilustrado por Agent Plugins, pelo marketplace de plugins para agentes e ferramentas de suporte como o Agent Package Manager.

Junto a esses desenvolvimentos, está se tornando claro que o harness é um ativo vivo que exige responsabilidade e manutenção contínuas. Discutimos, por exemplo, a avaliação da qualidade do harness, a construção de um harness que se autoaperfeiçoa e o uso de um flywheel de feedback para garantir que os agentes melhorem ao longo do tempo. O desafio mais amplo é dimensionar o harness para atender às necessidades empresariais. Nos próximos meses, poderemos ver enterprise harness sendo encarado como um produto, não muito diferente do que aconteceu com plataformas e a engenharia de plataformas há alguns anos.

Além dos modelos de ponta

Estamos entrando em uma nova fase no uso de modelos de IA, na qual optar automaticamente por um modelo proprietário de ponta (frontier model) não é mais a escolha óbvia. Esta edição destaca inovações significativas nesse espaço, incentivando as organizações a pensar de forma mais criativa sobre quais modelos usar, como e quando.

Os modelos open-weight são particularmente notáveis aqui. Há doze meses, a diferença em relação aos modelos proprietários de ponta era gritante, mas hoje ela é muito menor para muitas tarefas. O custo é um grande impulsionador, mas não o único: os modelos open-weight podem rodar em seus ambientes de nuvem pública, em neoclouds ou em seu próprio hardware. Essa flexibilidade pode ajudar a atender a requisitos estritos de privacidade e reduzir a dependência de um único fornecedor. Isso é especialmente visível na engenharia de software, onde a família GLM tornou os modelos open-weight alternativas plausíveis aos modelos proprietários de ponta para algumas cargas de trabalho de agentes de programação. A auto-hospedagem (self-hosting) também pode ajudar as organizações a manter o código-fonte e os prompts dentro de sua própria infraestrutura. Mas essa mudança não se limita aos modelos open-weight. Os modelos de programação locais estão ganhando popularidade,

© Thoughtworks, Inc. All Rights Reserved.

Temas



enquanto desenvolvimentos como modelos fundacionais para dados estruturados e modelos de decisão como o Jev refletem o reconhecimento crescente de que os LLMs de ponta não são a única abordagem para a IA.

Com essa gama crescente de opções, escolher o modelo certo para cada tarefa está se tornando uma responsabilidade importante para as equipes de engenharia. O roteamento de modelos está emergindo como uma consideração fundamental. Alertamos contra o uso indiscriminado de modelos de ponta e incentivamos as equipes a fazer escolhas deliberadas de modelos, mantendo-se abertas a alternativas.

Colocando mais agentes em ação

Em nossa última edição, os agentes de programação foram o foco da maioria dos blips e temas. Desta vez, muitas propostas se concentraram em colocar mais agentes em ação, descrevendo principalmente pessoas desenvolvedoras executando várias sessões de agentes em paralelo e guiando cada uma delas. Outras descreveram a entrega de blocos maiores de trabalho aos agentes, que são executados por períodos mais longos, como execuções agênticas autônomas noturnas, onde pessoas desenvolvedoras definem as tarefas e revisam os resultados. Essa é uma abordagem mais leve do que as abordagens multiagente coordenadas que cobrimos em edições anteriores. Um time de agentes de programação atribui papéis aos agentes em um fluxo de trabalho predefinido. Os enxames de agentes de programação vão além, com agentes orquestradores dividindo e atribuindo o trabalho por conta própria. Em ambas as abordagens, as pessoas desenvolvedoras delegam mais tomadas de decisão. As Thoughtworkers estão em grande parte executando múltiplas sessões sob um controle mais direto, o que consideramos um ponto de partida sensato para as pessoas engenheiras.

Entre as muitas ferramentas que discutimos para gerenciar sessões paralelas, incluindo Claude Squad, T3 Code, Orca e Paseo, o Herdr despertou o maior interesse. A maioria inclui uma caixa de entrada de atenção (attention inbox) que organiza as sessões por status e prioriza aquelas que estão aguardando uma resposta ou aprovação. O Omnigent possui uma, e o Paseo permite que pessoas desenvolvedoras verifiquem as sessões pelo celular. Os Git worktrees dão a cada uma dessas sessões sua própria cópia de trabalho, embora possam deixar para trás um emaranhado de branches parcialmente integradas. Dentro de uma sessão, os subagentes assumem tarefas específicas e mantêm a exploração e retries fora do contexto principal. O AgentsView permite que pessoas desenvolvedoras revisem sessões passadas, enquanto o Hunk ajuda a revisar as alterações geradas por agentes.

© Thoughtworks, Inc. All Rights Reserved.

Temas



À medida que as pessoas desenvolvedoras executam mais agentes, sua capacidade de atenção e revisão tornam-se gargalos. Quando a revisão não consegue acompanhar, a supervisão corre o risco de se tornar teatro de supervisão humana. A pressão se estende para as etapas seguintes. Quando várias pessoas desenvolvedoras executam, cada uma, múltiplos agentes, as alterações podem chegar mais rápido do que a esteira de entrega (delivery pipeline) consegue absorver. O mapeamento do caminho para produção ajuda a identificar onde o trabalho se acumula. Antes de adicionar sessões, as equipes devem considerar quanto seus processos de revisão e entrega conseguem suportar. A curto prazo, esperamos ver emergir mais práticas e ferramentas para orquestrar o trabalho baseado em agentes entre múltiplas pessoas engenheiras. Ferramentas como o Entire CLI, que compartilha sessões de agentes entre uma equipe, são um sinal inicial do interesse crescente em tornar o trabalho com agentes entre várias pessoas engenheiras mais eficaz.

Construindo AI factories

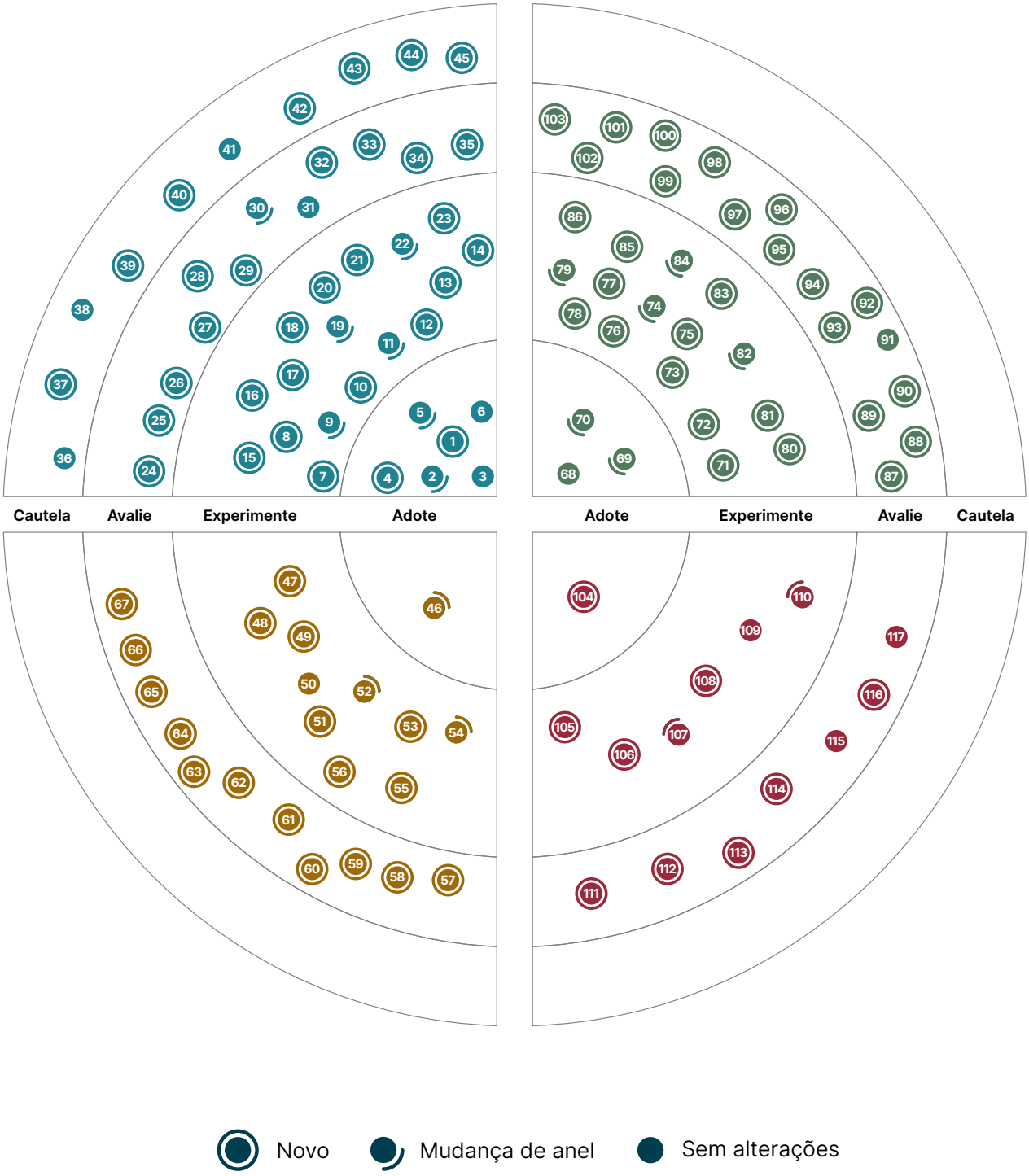
Embora as exigências de infraestrutura de IA tenham aparecido em vários volumes recentes do Radar, esta edição as traz para um foco muito mais preciso. À medida que o uso de IA generativa e de agentes se intensifica, as empresas estão descobrindo que a economia dos tokens se resume, em última análise, à economia da infraestrutura. O custo da IA depende não apenas do preço do modelo, mas também da seleção de modelos, roteamento, agrupamento (batching), movimentação de dados e da eficiência da plataforma subjacente. Grande parte da inovação necessária para tornar a IA economicamente sustentável está, portanto, localizada em camadas mais profundas da stack — e muitas vezes passa despercebida no radar de muitas pessoas.

Ao preparar esta edição, encontramos um ecossistema vibrante emergindo em torno da infraestrutura de IA. O Kubernetes, em particular, está se tornando uma base fundamental para o que às vezes é descrito como AI factories: plataformas estruturadas para executar cargas de trabalho de IA em escala. Vários blips ilustram essa mudança. O KServe e o KAITO oferecem suporte à disponibilização (serving) e implantação de modelos, bem como ao gerenciamento de cargas de trabalho otimizado para aceleradores. Mais acima na stack, o llm-d aborda a inferência distribuída e o escalonamento, enquanto o LiteLLM fornece roteamento e abstração entre modelos e provedores de inferência. A stack das AI factories também se estende além da computação para o armazenamento e redes. Tecnologias como o VAST DataStore atendem às demandas de armazenamento de alto rendimento de cargas de trabalho com uso intensivo de aceleradores, enquanto o Netris traz automação de rede definida por software para a infraestrutura subjacente, ajudando as equipes a provisionar e operar a conectividade de alta largura de banda da qual os clusters de IA dependem.

A mudança maior é que modelos, GPUs, motores de inferência, armazenamento e redes são cada vez mais gerenciados como recursos programáveis coordenados por uma camada de plataforma comum, tornando a infraestrutura de IA um fator central na economia de IA empresarial.

© Thoughtworks, Inc. All Rights Reserved.

O Radar



© Thoughtworks, Inc. All Rights Reserved.

Técnicas

Adote

1. Sandboxing de agentes
2. Testes de mutação
3. Mapeamento do caminho para produção
4. Criptografia pós-quântica
5. Revelação progressiva de contexto
6. Modelagem de ameaças

Experimente

7. Marketplace de plugins para agentes
8. Bases de conhecimento mantidas por agentes
9. Ancorar agentes de programação a uma aplicação de referência
10. Harness de programação para agentes de propósito geral
11. Flywheel de feedback
12. Raciocínio sobre base de código ancorado em grafos
13. Grilling
14. Planos visuais interativos com agentes
15. Managed agents runtime
16. Modelos open-weight
17. Supervisão humana baseada no nível de risco
18. Harness que se autoaperfeiçoa
19. Modelos de linguagem de pequeno porte
20. Guardrails com estado
21. Subagentes para melhor gestão de contexto
22. Análise de fluxo tóxico para IA
23. Execuções agênticas autônomas noturnas

Avalie

24. Responsabilidade de agentes no pipeline
25. Caixa de sugestões para agentes
26. Perfilamento de contexto
27. Enterprise harness
28. Modelos fundacionais para dados estruturados
29. Avaliação da qualidade do harness
30. Modelos de programação locais
31. MITRE ATLAS
32. Agentes de programação com contexto de produção
33. Chamada programática de ferramentas
34. Revisão semântica de textos editados por IA
35. Testes como dados em vez de testes como código

Cautela

36. Inchaço de instruções para agentes
37. Rendição cognitiva a skills de agentes
38. Shadow IT acelerada por IA
39. Proliferação de documentação gerada por IA
40. Uso indiscriminado de modelos de ponta
41. Volume de código produzido como medida de produtividade
42. Teatro de supervisão humana
43. Meat proxy
44. Compressão ingênua de contexto
45. Pull requests empilhados

Plataformas

Adote

46. VictoriaMetrics

Experimente

47. Kafka share groups
48. KServe
49. LibreChat
50. LiteLLM
51. Microsoft Copilot Retrieval API
52. Restate
53. Talos Linux
54. Temporal
55. vCluster
56. VictoriaLogs

Avalie

57. Agent Client Protocol (ACP)
58. Agent Plugins
59. agentgateway
60. AWS DevOps Agent
61. Claude Managed Agents
62. Modelos de decisão
63. KAITO
64. llm-d
65. Netris
66. Omnigent
67. VAST dataStore

Cautela

—

Ferramentas

Adote

68. mise
69. OpenAI Codex
70. OpenCode

Experimente

71. Act
72. Agent Package Manager
73. ast-grep
74. Beads
75. Chaos Mesh
76. Cube Core
77. Figma Code Connect
78. FluxMarkdown
79. Git AI
80. Linear
81. mattpocock/skills
82. OpenSpec
83. OpenWiki
84. Pi (agente de programação)
85. Sourcebot
86. Timefold Solver

Avalie

87. AgentsView
88. Apple container
89. Deepsec
90. DeepSeek Harness (DSH)
91. Entire CLI
92. FreeToken
93. GLM
94. Habit Hooks
95. Herdr
96. Hunk
97. nono.sh
98. OpenResearch CLI (orx)
99. Safe Chain
100. Swamp
101. Vale
102. Varlock
103. zizmor

Cautela

—

Linguagens e Frameworks

Adote

104. Open Data Contract Standard (ODCS)

Experimente

105. BoTorch
106. Deep Agents
107. Kagent
108. OpenFGA
109. Presidio
110. Promptfoo

Avalie

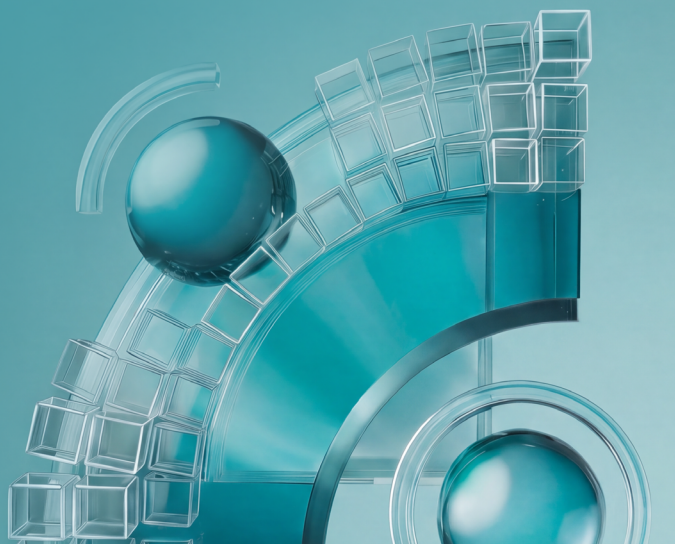
111. Apple Core AI
112. CALM
113. Hindsight
114. Quint
115. SolidJS
116. spartan/ui
117. TLA+

Cautela

—

© Thoughtworks, Inc. All Rights Reserved.

Técnicas



Adote

1. Sandboxing de agentes
2. Testes de mutação
3. Mapeamento do caminho para produção
4. Criptografia pós-quântica
5. Revelação progressiva de contexto
6. Modelagem de ameaças

Experimente

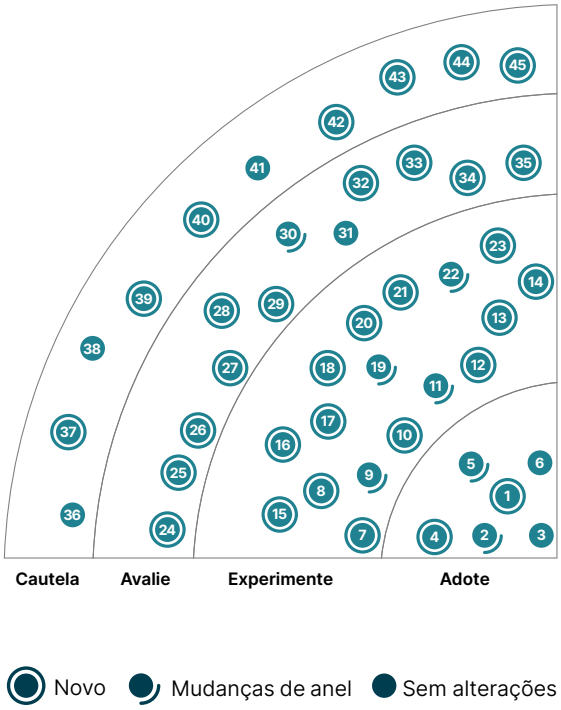
7. Marketplace de plugins para agentes
8. Bases de conhecimento mantidas por agentes
9. Ancorar agentes de programação a uma aplicação de referência
10. Harness de programação para agentes de propósito geral
11. Flywheel de feedback
12. Raciocínio sobre base de código ancorado em grafos
13. Grilling
14. Planos visuais interativos com agentes
15. Managed agents runtime
16. Modelos open-weight
17. Supervisão humana baseada no nível de risco
18. Harness que se autoaperfeiçoa
19. Modelos de linguagem de pequeno porte
20. Guardrails com estado
21. Subagentes para melhor gestão de contexto
22. Análise de fluxo tóxico para IA
23. Execuções agênticas autônomas noturnas

Avalie

24. Responsabilidade de agentes no pipeline
25. Caixa de sugestões para agentes
26. Perfilamento de contexto
27. Enterprise harness
28. Modelos fundacionais para dados estruturados
29. Avaliação da qualidade do harness
30. Modelos de programação locais
31. MITRE ATLAS
32. Agentes de programação com contexto de produção
33. Chamada programática de ferramentas
34. Revisão semântica de textos editados por IA
35. Testes como dados em vez de testes como código

Cautela

36. Inchaço de instruções para agentes
37. Rendição cognitiva a skills de agentes
38. Shadow IT acelerada por IA
39. Proliferação de documentação gerada por IA
40. Uso indiscriminado de modelos de ponta
41. Volume de código produzido como medida de produtividade
42. Teatro de supervisão humana
43. Meat proxy
44. Compressão ingênua de contexto
45. Pull requests empilhados



Novo Mudanças de anel Sem alterações

1. Sandboxing de agentes

Adote

Anteriormente, recomendamos a execução em sandbox para agentes de programação. À medida que os agentes de IA se tornam cada vez mais autônomos e capazes de atuar não apenas em ambientes de código, mas também em sistemas de negócios, restringir seu escopo operacional tornou-se fundamental. Agora consideramos o sandboxing de agentes um padrão sensato para todos os agentes. Executar agentes em ambientes isolados, com privilégios restritos e acesso controlado a sistemas externos, reduz o risco e limita o impacto de comportamentos inesperados, ações não intencionais ou exploração maliciosa. Recomendamos que as equipes apliquem uma abordagem baseada em riscos ao sandboxing, fundamentada pela modelagem de ameaças de seus agentes e sistemas. Embora não exista uma solução única para todos os casos, manchetes recentes sobre segurança reforçam a importância de aplicar práticas de segurança consolidadas, como o princípio do menor privilégio, ao construir e executar agentes.

2. Testes de mutação

Adote

Os testes de mutação avaliam a qualidade da suíte de testes introduzindo automaticamente pequenos bugs artificiais no código da aplicação e verificando se pelo menos um teste falha. Se os testes passarem apesar da alteração no código, a mutação sobrevive, revelando lacunas na cobertura de testes ou na qualidade das asserções. À medida que as equipes adotam assistentes de programação por IA, o volume de código e testes gerados por IA aumenta rapidamente, tornando as métricas tradicionais de cobertura de código mais fáceis de serem infladas com testes superficiais que carecem de validação significativa.

Essa técnica atua como uma proteção (guardrail), ajudando a garantir que os testes gerados não passem simplesmente sem capturar alterações de lógica. Além de frameworks dedicados, os agentes de IA agora podem executar testes de mutação de forma autônoma durante a execução, modificando diretamente o código, executando testes e avaliando asserções em seu próprio ciclo. Como as execuções automatizadas de mutação consomem tempo, pessoas desenvolvedoras devem focar em pacotes internos específicos ou recorrer a verificações incrementais para executar testes apenas no código alterado recentemente.

3. Mapeamento do caminho para produção

Adote

O mapeamento do caminho para produção é uma prática padrão na Thoughtworks que ganhou nova relevância com o advento da programação agêntica. O aumento na vazão de código pode expor ou amplificar gargalos em outras partes do caminho para a produção, tornando essencial que as equipes otimizem o fluxo ponta a ponta do software em vez de focar apenas na velocidade de programação. A atividade é realizada com mais frequência em um workshop que inclui todas as pessoas envolvidas no design, desenvolvimento, liberação e operação do software. Primeiro, as participantes listam as etapas do processo em um quadro branco, desde a estação de trabalho da desenvolvedora até a produção. Em uma segunda passagem, elas registram os pontos de dor. A atividade é frequentemente reveladora, pois as participantes identificam atrasos, riscos e inconsistências dos quais não tinham conhecimento prévio. Enfrentar esses gargalos é fundamental para melhorar a produtividade geral da entrega.

© Thoughtworks, Inc. All Rights Reserved.

4. Criptografia pós-quântica

Adote

A criptografia pós-quântica (PQC) refere-se a algoritmos criptográficos projetados para proteger dados digitais contra ataques de futuros computadores quânticos. Com o avanço da computação quântica, a ameaça à criptografia convencional torna-se cada vez mais real: dados criptografados hoje podem ser coletados agora e descriptografados mais tarde, tornando a troca de chaves a área mais urgente para migração. Com os padrões do NIST agora finalizados (ML-KEM, ML-DSA e SLH-DSA), governos de todo o mundo estabelecem a migração entre 2030 e 2035.

Infraestruturas modernas, linguagens (por exemplo, o Java a partir do JDK 24) e frameworks estão oferecendo cada vez mais suporte nativo a PQC em suas versões mais recentes. Habilitá-lo, no entanto, exige a atualização de versões mais antigas ou o ajuste de configurações. As organizações devem exigir a criptografia resiliente a quântica para projetos novos (greenfield), enquanto aceleram os roteiros de migração para sistemas legados.

5. Revelação progressiva de contexto

Adote

A revelação progressiva de contexto é uma técnica de engenharia de contexto para fornecer informações a LLMs sob demanda. O sistema inicializa a LLM com o contexto essencial e permite que as respostas da LLM ou as entradas do usuário acionem o carregamento de dados adicionais. Estamos movendo a revelação progressiva de contexto para "Adote" porque a prática é quase universal em aplicações sérias, embora assuma muitas formas. Na programação, um exemplo é manter seu AGENTS.md enxuto e fornecer links para documentações que o agente possa consultar quando necessário. Harnesses de programação também usam essa abordagem para adiar o custo total de contexto de ferramentas MCP e agent skills até que sejam utilizadas. Aplicações de negócios podem usar RAG vetorial ou GraphRAG para recuperar informações adicionais com base na entrada do usuário. Em cada caso, a revelação progressiva de contexto pode reduzir o consumo e os custos de tokens, além de ajudar a combater o context rot.

6. Modelagem de ameaças

Adote

A modelagem de ameaças é um exercício para identificar ameaças potenciais aos seus sistemas. É uma prática de segurança fundamental para todo desenvolvimento de software, especialmente para aplicações de IA generativa, cujos riscos de segurança distintos, como injeção de prompt e ação autônoma, podem ser ignorados por verificações e varreduras baseadas em regras. A modelagem de ameaças deve ser uma parte regular e liderada por humanos no ciclo de vida de desenvolvimento de software, com o apoio de agentes. A discussão ajuda a trazer à tona comportamentos não documentados e a construir o entendimento das partes interessadas.

Recomendamos começar com ameaças conhecidas e mapeá-las para o seu contexto. Recursos úteis incluem o catálogo de ameaças de IA MITRE ATLAS, o MAESTRO como complemento para frameworks como o STRIDE, e a análise de fluxo tóxico para IA para raciocinar sobre onde a injeção de prompt pode entrar. Uma sessão deve terminar com decisões explícitas sobre onde o comportamento não determinístico do modelo é aceitável, onde não é e onde guardrails determinísticos são necessários para limitar riscos e danos.

© Thoughtworks, Inc. All Rights Reserved.

7. Marketplace de plugins para agentes

Experimente

Um marketplace de plugins para agentes é um meio de distribuir ativos de harness em toda a organização. A necessidade surge quando várias equipes adotam agentes de programação e desejam compartilhar skills, instruções, servidores MCP, definições de subagentes e outros ativos. Copiar esses elementos entre repositórios não oferece uma maneira fácil de publicá-los, descobri-los ou atualizá-los. Implementamos um marketplace de plugins para agentes usando repositórios Git documentados, com plugins no formato Agent Plugins que os agentes de programação consomem diretamente ou que são gerenciados como dependências com o Agent Package Manager. A Anthropic popularizou o padrão com seu Claude Code plugin marketplace, e registros hospedados como o da Tessl e coleções da comunidade como wshobson/agents adotaram a ideia. Vemos os marketplaces de plugins para agentes como uma forma prática de disponibilizar seu enterprise harness para toda a organização.

8. Bases de conhecimento mantidas por agentes

Experimente

Com bases de conhecimento mantidas por agentes, uma equipe mantém seu conhecimento compartilhado em arquivos de texto simples vinculados entre si, muitas vezes em um cofre (vault) do Obsidian, e usa um agente de programação para mantê-los. O LLM Wiki de Andrej Karpathy descreve o padrão: fontes brutas que o agente lê, mas nunca altera, uma wiki que ele mantém e um esquema que orienta como ele faz isso. O OpenWiki aplica uma versão mais restrita da ideia à documentação de um único repositório de código. A técnica é prática hoje porque agentes de programação de uso geral já conseguem ler e editar arquivos de forma eficaz, enquanto o Git fornece controle de versão. Várias de nossas equipes usam variações dessa abordagem, com gerentes de produto, designers, pessoas desenvolvedoras e seus agentes consultando a mesma base de conhecimento. Essas equipes acham suas informações mais conectadas e acessíveis.

Muitas pessoas começam com um “segundo cérebro” pessoal. Recomendamos dar o próximo passo e compartilhar um com toda a equipe, embora expandi-lo para uma base de conhecimento em toda a organização exija mais do que simplesmente apontar um agente de programação para uma pasta de arquivos. O conhecimento gerado pode se distanciar da realidade e, sem o esforço humano regular para mantê-lo preciso, a base de conhecimento corre o risco de criar proliferação de documentação gerada por IA.

9. Ancorar agentes de programação a uma aplicação de referência

Experimente

Ancorar agentes de programação a uma aplicação de referência aborda o desafio da degradação de templates separando templates de scaffolding estruturais e enxutos das regras de configuração de plataforma em constante evolução. Anteriormente, focamos no uso de aplicações de referência dinâmicas e compiláveis para manter o código gerado em sintonia com as práticas atuais. Agora, estamos vendo equipes estenderem esse padrão com sucesso para o scaffolding direcionado por IA.

Geradores de código tradicionais e ferramentas de CLI para scaffolding rapidamente ficam desatualizados à medida que as diretrizes da plataforma, dependências e melhores práticas arquitetônicas mudam. Em vez de codificar essas regras rigidamente em templates estáticos, os agentes de programação consultam fontes de informação centralizadas e atualizadas — como um OpenWiki ou bases de conhecimento mantidas por agentes estruturadas — em tempo de execução

© Thoughtworks, Inc. All Rights Reserved.

para recuperar convenções e configurações atuais. Consolidar regras de plataforma em evolução em repositórios unificados e acessíveis por agentes ajuda os agentes a gerar código que segue os padrões atuais, sem exigir atualizações contínuas nas ferramentas de scaffolding subjacentes.

10. Harness de programação para agentes de propósito geral

Experimente

Equipes que constroem agentes para trabalhos que não envolvem código, como processamento de documentos, suporte de operações ou planejamento de migração, normalmente começavam com um framework de agentes e montavam o loop do agente por conta própria. Cada vez mais, vemos equipes construírem em cima de um harness de programação para agentes de propósito geral, reutilizando o ambiente de execução (runtime) de um agente de código maduro. Agentes de código populares agora expõem seu runtime de forma programática através do Claude Agent SDK, do Codex SDK e app-server, do servidor e SDK do OpenCode ou do SDK e modo RPC do Pi. Esses harnesses já lidam com aspectos complexos do loop do agente, incluindo a compactação de contexto em tarefas de longa duração, a persistência de sessões e a execução headless com as mesmas habilidades de agente, subagentes e instruções usados em sessões interativas de agentes de código. Em comparação com frameworks de agentes, o trade-off é ter menos controle em troca de um runtime que já foi exaustivamente testado. Usamos essa abordagem para construir um acelerador interno para migrações de aplicações .NET.

No entanto, esses harnesses diferem em suas capacidades. O Codex e o Claude Agent SDK oferecem sandboxing e políticas de aprovação, enquanto o Pi deixa as proteções (guardrails) deliberadamente para extensões, facilitando a adaptação, mas exigindo mais trabalho das equipes. Como esses runtimes foram projetados para o desenvolvimento de software, seus prompts do sistema e ferramentas integradas tendem a direcionar os agentes a escrever e executar código. As equipes devem revisar esses padrões para o seu domínio pretendido. Elas também devem monitorar custos, já que os loops de agentes de código podem consumir um número substancial de tokens, e prompts integrados grandes somam a cada turno. Essa é uma das razões pelas quais harnesses mínimos como o Pi são atraentes.

11. Flywheel de feedback

Experimente

No Radar anterior, descrevemos o flywheel de feedback como uma forma de melhorar o harness de um agente de programação a partir de sessões anteriores. A maioria das pessoas desenvolvedoras atualiza arquivos AGENTS.md e habilidades de agentes quando um agente erra algo, mas o que uma desenvolvedora aprende pode não ser compartilhado com o restante da equipe. O flywheel de feedback adiciona uma pergunta ao final de cada sessão, questionando se algo deve mudar em um artefato compartilhado, junto com um item de retrospectiva onde a equipe decide o que atualizar. Sensores de feedback mostram quais verificações continuam falhando, uma caixa de sugestões dos agentes coleta problemas relatados pelos agentes e ferramentas como o Backpass propõem edições com base em sessões passadas para um humano aceitar ou rejeitar. Estamos movendo o flywheel de feedback para “Experimente” porque edições no harness agora são rotineiras.

O flywheel precisa remover tanto quanto adicionar. Regras que apenas se acumulam se tornam inchaço de instruções para os agentes, e um sensor que gera apenas ruído deve ser eliminado. Recomendamos verificar continuamente quais artefatos ainda estão em uso e atualizados. O perfilamento de contexto mostra se as edições estão aumentando o que cada sessão carrega. Onde o autoaperfeiçoamento de agentes faz com que um agente execute o loop, o flywheel de feedback o torna um hábito da equipe, com os agentes propondo alterações e as pessoas escolhendo quais alterações fazer.

© Thoughtworks, Inc. All Rights Reserved.

12. Raciocínio sobre base de código ancorado em grafos

Experimente

À medida que fluxos de trabalho com alto consumo de contexto se tornam mais comuns, o envio de arquivos-fonte completos ou texto não estruturado para LLMs aumenta os custos com tokens. O raciocínio sobre base de código ancorado em grafos aborda isso representando código, documentação e relacionamentos como um grafo estruturado, permitindo que agentes recuperem subgrafos relevantes como contexto. Ferramentas como o Graphify capturam dependências estruturais, ajudando os agentes a identificar relacionamentos que a recuperação baseada em texto pode perder.

No entanto, o grounding baseado em grafos traz ressalvas. Pesquisas recentes sugerem que agentes às vezes podem ter um desempenho pior com contexto baseado em grafos do que com ferramentas mais simples, como find e grep. Colocamos essa técnica em "Experimente" para explorar quando o contexto baseado em grafos melhora a eficiência e a precisão, construindo sobre nosso blip anterior sobre inteligência de código como ferramenta para agentes.

13. Grilling

Experimente

Grilling é uma técnica na qual um agente de programação entrevista você sobre um plano antes de agir para resolver potenciais áreas de ambiguidade. O exemplo mais conhecido vem de mattpocock/skills, mas outras bibliotecas de skills têm suas próprias versões. Um benefício é que o grilling desestimula os agentes de programação a adivinhar quando sua intenção não está clara. Ele traz à tona suposições que o agente de programação poderia, de outra forma, fazer por conta própria. Outra vantagem é que ajuda a combater a dívida cognitiva da base de código. O processo de grilling também força você a examinar suas próprias suposições, deixando você com uma compreensão melhor do plano de implementação que pode ser registrada em registros de decisão. Nossas equipes consideram o grilling uma adição útil à sua prática de programação com agentes.

14. Planos visuais interativos com agentes

Experimente

Planos visuais interativos com agentes são planos autossuficientes em HTML ou MDX com diagramas, fluxos de usuário e wireframes que as equipes podem explorar antes da implementação. Nossas equipes consideram esses artefatos mais eficazes do que longos planos em texto para verificar a intenção e revisar alterações complexas. Elas alertam contra o uso dessa técnica para "big upfront designs" e favorecem uma abordagem mais iterativa, dividindo o trabalho em etapas menores e verificáveis para construir o entendimento ao longo do caminho. A mesma abordagem pode fornecer um resumo visual do trabalho concluído, mas nossas equipes consideram os resumos mais úteis quando a implementação se desvia do plano original. Recomendamos o uso de planos visuais interativos com agentes para construir um entendimento compartilhado e estabelecer um ponto de revisão humana antes que os agentes implementem as alterações.

15. Managed agents runtime

Experimente

Managed agents runtime é a prática de executar agentes de entrega de software em uma plataforma que gerencia sua execução, infraestrutura e necessidades operacionais, em vez de depender da estação de trabalho de uma pessoa desenvolvedora. O runtime hospeda o loop do agente e fornece computação em sandbox, gatilhos, manipulação de credenciais, estado da sessão e observabilidade,

© Thoughtworks, Inc. All Rights Reserved.

deixando para as equipes a definição do modelo, prompt, ferramentas e habilidades. Trabalhos assíncronos e de longa duração são bem adequados para agentes em segundo plano que coletam tarefas a partir de issues, pipelines ou agendamentos e enviam pull requests para revisão.

As ferramentas de código estão convergindo para essa abordagem, com os agentes em nuvem do Cursor, o agente de código do GitHub Copilot e as tarefas em nuvem do OpenAI Codex executando todos em sandboxes remotas. O Claude Managed Agents oferece um equivalente de uso geral. Nossa equipe de segurança interna executa um agente diário que faz a triagem de falsos positivos de detecção de endpoint e propõe alterações de regras. "Gerenciado" não significa necessariamente a nuvem de terceiros. Em um cliente, executamos agentes personalizados em um runtime auto-hospedado em um loop contínuo através de múltiplos repositórios, corrigindo code smells sinalizados por scanners e atualizando dependências junto com qualquer alteração de código necessária.

Um serviço hospedado é frequentemente o lugar mais fácil para começar, mas organizações com restrições de residência de dados, segurança ou integração podem precisar de sua própria plataforma. De qualquer forma, recomendamos tratar o runtime como infraestrutura de produção, com sandboxing de agentes e definições de agentes controladas por versão.

16. Modelos open-weight

Experimente

Os modelos open-weight, nos quais os pesos do modelo, arquiteturas e código de inferência são lançados publicamente, atingiram um ponto de inflexão significativo. Atraídos por seu menor custo total de propriedade, temos avaliado os modelos mais recentes de famílias como DeepSeek, GLM e Kimi para trabalhos de entrega em produção que exigem raciocínio complexo e geração de código de alta qualidade, e estamos vendo bons resultados.

Além da eficiência de custos, os modelos open-weight oferecem flexibilidade de implantação, liberdade de vendor lock-in e uma soberania de dados mais forte. Equipes de engenharia podem dimensionar adequadamente suas implantações — desde modelos flash leves até modelos Mixture of Experts (MoE) de centenas de bilhões ou trilhões de parâmetros — e realizar fine-tuning privado usando dados do domínio. Embora a auto-hospedagem (self-hosting) exija um esforço operacional maior, os principais provedores de nuvem agora oferecem modelos open-weight gerenciados que reduzem a barreira para adoção. Em vez de tratá-los apenas como opções econômicas, as organizações devem avaliar os modelos open-weight para uma ampla variedade de cargas de trabalho corporativas, indo muito além da programação.

17. Supervisão humana baseada no nível de risco

Experimente

A supervisão humana baseada no nível de risco alinha a profundidade da revisão com o impacto que uma alteração poderia causar. Revisar tudo com a mesma intensidade resulta em um teatro de supervisão humana. Sugerimos triar sistematicamente as alterações. As alterações mais arriscadas ou mais difíceis de reverter são reservadas para implementação por humanos. Alterações de risco intermediário passam por pull requests revisados por humanos. Alterações de baixo risco podem ser mescladas automaticamente por agentes de programação por meio de uma esteira de integração contínua.

© Thoughtworks, Inc. All Rights Reserved.

Duas estratégias decidem quais alterações precisam de mais revisão: zonas na base de código exigem revisão elevada para arquivos ou diretórios específicos, enquanto a classificação de alterações baseada em LLM analisa o impacto das alterações, mas é falível. O processo publicado da Meta combina essas duas estratégias. Nossas equipes consideram a supervisão humana baseada no nível de risco uma forma eficaz de aproveitar ao máximo a escassa atenção humana.

18. Harness que se autoaperfeiçoa

Experimente

Um harness que se autoaperfeiçoa usa agentes para analisar traces de execução e feedback de avaliação para, então, propor ou implementar melhorias no sistema que envolve o modelo, incluindo prompts, estratégias de contexto, fluxos de trabalho, habilidades, ferramentas, memória e middleware. A ideia não é nova: técnicas como o flywheel de feedback usam acertos e falhas de sessões de agentes de programação para melhorar o harness, mas ainda dependem de pessoas para interpretar o feedback, decidir o que alterar e atualizar o sistema. Pesquisas recentes exploram uma abordagem mais autodirigida. O Meta-Harness e o VERSE usam traces de execução e resultados de avaliação para melhorar o código do harness; o VERSE também evolui as próprias ferramentas e procedimentos do otimizador. A visão geral de Lilian Weng traça essa progressão desde a otimização de prompts até a evolução do harness.

A observabilidade e a programabilidade são essenciais para essa transição. A observabilidade ajuda os agentes a diagnosticar falhas a partir de traces de execução e a conectar mudanças a resultados mensuráveis. A programabilidade oferece aos agentes acesso a componentes que eles podem inspecionar, modificar e testar, expandindo o escopo para o autoaperfeiçoamento. As extensões do Pi, os Claude Code Mods e o DeepSeek Harness (DSH) ilustram essa direção por meio de contexto, fluxos de trabalho, ferramentas e componentes de harness extensíveis. Uma maior abertura cria mais oportunidades de melhoria, mas os ganhos ainda exigem validação. Recomendamos experimentos delimitados com testes de regressão, enquanto pessoas definem objetivos, estabelecem limites de modificação e avaliam se as melhorias se generalizam para novas tarefas.

19. Modelos de linguagem de pequeno porte

Experimente

Continuamos a ver uma mudança na indústria, afastando-se da mentalidade de que “quanto maior, melhor” em direção ao aprimoramento da inteligência artificial por meio de dados de treinamento de alta qualidade, destilação de modelos e quantização. Modelos de linguagem de pequeno porte (SLMs) tornaram-se cada vez mais úteis para tarefas específicas que não exigem os recursos computacionais de modelos maiores. À medida que as organizações enfrentam custos crescentes com a execução de LLMs, as equipes estão reconhecendo o valor de escolher o modelo certo para a tarefa.

Modelos compactos também são viáveis para edge computing, como o Apple Core AI demonstra ao executar pequenos modelos fundamentais diretamente nos dispositivos. Nossas equipes constataram que os SLMs são eficazes para tarefas específicas e de alta frequência que não exigem conhecimento extenso ou raciocínio complexo, como sumarização, extração de dados estruturados, classificação de intenção para roteamento de requisições e programação básica. A menos que um caso de uso de agente exija raciocínio em múltiplas etapas ou chamada confiável de ferramentas (tool calling), as equipes devem considerar um SLM como uma alternativa de menor custo e menor latência.

© Thoughtworks, Inc. All Rights Reserved.

20. Guardrails com estado

Experimente

Guardrails com estado (stateful guardrails) estendem os guardrails de LLM de mensagens individuais para conversas inteiras. Em vez de verificar apenas a mensagem atual do usuário ou a resposta final do modelo, eles avaliam um histórico compacto de turnos recentes em relação a verificações específicas de políticas e mantêm um estado de penalidades (strikes) ou bloqueios (bans) ao longo da sessão. Isso é importante porque intenções inseguras podem surgir gradualmente: cada mensagem pode parecer aceitável isoladamente, enquanto a conversa como um todo revela tentativas de evasão de políticas, jailbreaking ou extração de prompts. Na prática, as verificações com estado são executadas como um ponto de controle (gate) em tempo de execução antes que o agente atue, juntamente com os guardrails convencionais de entrada e saída para turno único.

Avaliar o histórico da conversa a cada turno adiciona latência e custo, e os limites de penalidade ou bloqueio precisam de ajustes para evitar que falsos positivos bloqueiem usuários legítimos. Nossas equipes usaram essa técnica em um assistente conversacional em produção para fluxos de trabalho sensíveis, rastreando as decisões dos guardrails no Langfuse. Recomendamos que as equipes que executam assistentes multiturno em produção experimentem guardrails com estado em vez de confiar apenas em verificações por mensagem.

21. Subagentes para melhor gestão de contexto

Experimente

Subagentes para melhor gestão de contexto é a prática de delegar trabalho a subagentes para mantê-lo fora do contexto do agente orquestrador. Exploração, saída de ferramentas, tentativas de repetição e raciocínio intermediário permanecem no subagente, e a sessão principal recebe um resumo curto. A maioria das pessoas começa usando subagentes para executar tarefas em paralelo ou formar times de agentes de programação baseados em papéis. Proteger o contexto do orquestrador é outro motivo para utilizá-los. O que quer que permaneça no contexto do orquestrador compete por sua atenção nas interações subsequentes. Trazer a transcrição completa de um subagente para a thread principal para verificar seu progresso anula esse benefício. Algumas diretrizes úteis incluem executar de dois a quatro subagentes por vez e manter tarefas que exijam a mesma compreensão do código dentro de um único agente, para que múltiplos agentes não precisem reconstruir essa mesma compreensão.

Tratar subagentes deliberadamente como um limite para a engenharia de contexto é fácil de experimentar porque a maioria dos agentes de codificação suporta subagentes e a mudança está basicamente nas instruções. Usamos modelos mais baratos para subagentes encarregados de tarefas específicas, como pesquisas, reduzindo potencialmente os custos em comparação ao uso indiscriminado de modelos de ponta. Estamos colocando a técnica em “Experimente” porque seu benefício ainda não foi medido sistematicamente. As evidências até agora vêm de relatos de profissionais sobre suas próprias sessões, sem uma contabilidade detalhada de tokens. Também observamos que o mesmo modelo com o mesmo prompt usa subagentes de forma diferente dependendo do harness. Equipes que experimentarem essa abordagem devem usar perfilamento de contexto para verificar se o contexto do orquestrador realmente diminui.

© Thoughtworks, Inc. All Rights Reserved.

22. Análise de fluxo tóxico para IA

Experimente

À medida que as capacidades dos agentes de IA se expandem, as equipes estão cada vez mais implantando sistemas que enfrentam uma combinação letal de três riscos de segurança: acesso a dados privados, exposição a conteúdos não confiáveis e a capacidade de se comunicar externamente. Essa combinação cria oportunidades para ataques como prompt injection e tool poisoning. Para lidar com esses riscos, movemos a análise de fluxo tóxico para IA para "Experimente".

Embora a modelagem de ameaças geral forneça uma base de segurança abrangente, a análise de fluxo tóxico se concentra em identificar caminhos de dados não seguros em sistemas baseados em agentes. Esses riscos não se limitam às integrações padrão. Eles também podem surgir de skills de agentes maliciosas que parecem úteis, mas contêm instruções ocultas para exfiltrar dados sensíveis. Usamos com sucesso ferramentas como o Agent Scan para realizar essa análise, ajudando as equipes a identificar e mitigar caminhos de dados não seguros antes que sejam explorados.

23. Execuções agênticas autônomas noturnas

Experimente

Execuções agênticas autônomas noturnas consistem em atribuir a um agente de programação uma tarefa com escopo de feature no fim do expediente e revisar o que foi produzido na manhã seguinte. Fizemos isso com funcionalidades no nível de épicos e tarefas de infraestrutura que podem levar horas. Um único agente é suficiente, sem a necessidade de um enxame de agentes de programação, embora ele possa utilizar subagentes. Uma das equipes da Thoughtworks passa o dia colaborando em planos e especificações e deixa a programação para os agentes durante a noite. O termo "noturnas" está no nome porque é algo difícil de fingir. Uma equipe capaz de fazer isso também executará tarefas semelhantes durante o dia. Já uma equipe que não deixa um agente programando sem supervisão à noite, provavelmente também não está executando trabalhos de forma totalmente autônoma durante o dia.

Assim como os pré-requisitos para microserviços, essa prática possui pré-requisitos, e incentivar as equipes a cumpri-los é uma grande parte do seu valor. Os agentes precisam de execução em sandbox, trabalho especificado antecipadamente como em desenvolvimento orientado a especificações e sensores de feedback para que possam corrigir seus próprios erros. O trabalho de uma noite pode ser um lote grande demais para ser revisado efetivamente linha por linha, de modo que as equipes precisam de um harness testado para validar as alterações em vez de depender exclusivamente da revisão humana de código. Técnicas complementares incluem executar o agente em um managed agent runtime e permitir que os agentes mantenham o build verde corrigindo falhas no pipeline por conta própria. Posicionamos essa prática em "Experimente" para equipes que possuem essas bases; para outras equipes, é um objetivo que justifica desenvolvê-las.

24. Responsabilidade de agentes no pipeline

Avalie

A responsabilidade de agentes no pipeline incorpora os agentes de programação à responsabilidade da equipe em manter o build verde. Um agente que não consegue ver o status do build não pode respeitar um acordo de equipe para não fazer push quando ele estiver vermelho, enquanto uma pessoa que repassa logs de falha em uma janela de chat se torna um meat proxy.

© Thoughtworks, Inc. All Rights Reserved.

Recomendamos tornar o estado do build parte do contexto do agente. Exponha o status do pipeline e os logs por meio das ferramentas do agente para que ele possa ver o que deu errado e ajudar a corrigir. Aplique regras importantes mecanicamente com travas (gates) de pre-push e no lado do servidor para evitar que os agentes façam push de alterações quebradas. A maior taxa de commits originada por agentes de programação está levando as equipes a repensar suposições sobre a disciplina de build, inclusive por meio de técnicas como supervisão humana baseada no nível de risco. Inclua o papel dos agentes em sua política de build ao realizar o mapeamento do caminho para produção.

25. Caixa de sugestões para agentes

Avalie

Uma caixa de sugestões para agentes dá a um agente em execução um local para reportar os problemas que ele encontra, como um bug em uma ferramenta ou uma dependência ausente em seu ambiente. Quando a única saída de um agente é um pull request ou um veredito estruturado, ele pode contornar o problema sem que ninguém saiba. A sessão seguinte, então, encontra o mesmo problema novamente. A caixa pode ser um canal do Slack, um repositório de memória ou um rastreador de problemas (issue tracker). A Lovable chama sua versão de "vent tool" e o agents-feedback oferece uma para agentes de programação. A memória pode ajudar um agente a evitar uma ferramenta com defeito, mas uma caixa de sugestões faz com que a ferramenta seja corrigida. A técnica funciona tanto para agentes de negócios em produção quanto para agentes de programação.

Usamos a caixa de sugestões para agentes com um agente que inicia uma nova sessão a cada dia para rascunhar regras de detecção para engenharia. Um segundo agente deduplica os relatórios. O mesmo relatório vindo de várias sessões é um sinal claro, embora alguns relatórios sejam subjetivos ou transitórios. Com a caixa instalada, as execuções fluíram melhor e tivemos uma lista de melhorias a fazer. As sugestões ainda precisam ser revisadas e trabalhadas, por exemplo, por meio do autoaperfeiçoamento de agentes. Estamos colocando a caixa de sugestões para agentes em "Avalie" porque nossa experiência vem de um único agente. A Lovable realiza a mesclagem (merge) de cerca de dez correções por dia a partir de suas "vents", mas relata que aproximadamente metade das correções propostas pelo seu agente de triagem são falsos positivos.

26. Perfilamento de contexto

Avalie

À medida que os custos de tokens aumentaram e a indústria deixou para trás a modinha do tokenmaxxing, as pessoas desenvolvedoras estão buscando maneiras de gerenciar o uso de tokens. O antigo conselho de que você deve sempre medir antes de otimizar ainda se aplica — o perfilamento de contexto significa analisar para onde está indo o seu gasto com tokens, por exemplo, em leituras de arquivos ou chamadas de ferramentas MCP. A intuição é um péssimo guia para identificar quais otimizações valem a pena, especialmente quando se leva em conta o armazenamento em cache, os níveis de modelos e os preços promocionais.

Em nossa experiência, as maiores economias geralmente vêm de mudanças simples, como limpar sessões no momento certo, ajustar parâmetros de cache ou silenciar scripts de build muito falantes. No entanto, frequentemente vemos pessoas desenvolvedoras recorrendo a soluções complexas, como proxies de compressão, sem medir seu impacto. Ferramentas maduras ainda não surgiram, mas você pode se virar analisando suas sessões de agente ou dados de ferramentas como o Entire CLI e o AgentsView.

© Thoughtworks, Inc. All Rights Reserved.

27. Enterprise harness

Avalie

Um enterprise harness é a camada organizacional daquilo que os agentes de programação consomem — habilidades, ferramentas, conectores para contexto interno, tecnologias aprovadas, registros de decisões de arquitetura, orientações sobre o caminho para a produção, hooks e políticas de segurança — gerenciados coletivamente. Observamos que, à medida que várias equipes em uma organização adotam agentes de programação, elas precisam de um contexto comum para complementar as instruções compartilhadas curadas de cada equipe. É como uma plataforma de experiência da desenvolvedora, mas para a experiência do agente. Vimos organizações irem além, construindo um agente de programação personalizado sobre o OpenCode, mas recomendamos o compartilhamento de ativos por meio de padrões abertos e sua distribuição via um marketplace de plugins para agentes. Ainda não há consenso sobre como fazer isso bem, portanto, mantenha o enterprise harness enxuto e federado, deixe espaço para experimentação e colha o que funciona.

28. Modelos fundacionais para dados estruturados

Avalie

A IA corporativa tem se concentrado fortemente em texto e visão, enquanto grande parte dos dados de negócios permanece presa em bancos de dados relacionais. Os modelos fundacionais adaptam transformers para dados estruturados, realizando o pré-treinamento nas sequências de eventos e logs de transações de uma organização para aprender representações reutilizáveis do comportamento do negócio.

Ao contrário dos modelos tradicionais baseados em gradient boosting, que exigem ampla engenharia de recursos e retreinamento, esses modelos podem suportar múltiplas tarefas downstream com um ajuste fino leve. Ferramentas como TabPFN e TabFM podem melhorar a avaliação de riscos, a análise preditiva e a retenção de clientes diretamente a partir de dados tabulares ou relacionais. Dado o seu potencial para desbloquear insights de negócios sem pipelines complexos de engenharia de recursos, sugerimos avaliar essa técnica.

29. Avaliação da qualidade do harness

Avalie

Atualmente, organizações e equipes criam harnesses customizados e adaptados aos seus projetos, incluindo skills de agentes, plugins, AGENTS.md e guardrails determinísticos. A avaliação da qualidade do harness tornou-se uma técnica importante, avaliando a qualidade do próprio harness. Guardrails e harnesses ajudam um agente de programação a corrigir o rumo e a melhorar a qualidade geral, mas apenas se realmente funcionarem.

Um harness geralmente inclui componentes determinísticos e não determinísticos. Para harnesses não determinísticos, nossas equipes usam PluginEval para avaliar plugins ou skills de agente de programação. Para harnesses determinísticos, podemos introduzir uma suíte de testes para permitir o desenvolvimento desses guardrails no estilo TDD, garantindo que cada um funcione de fato e forneça evidências tangíveis para provar sua eficácia. Recomendamos executar a avaliação de qualidade (evals) de forma agendada (como diariamente à noite ou junto com o CI).

© Thoughtworks, Inc. All Rights Reserved.

30. Modelos de programação locais

Avalie

Nas edições anteriores do Radar, colocamos os assistentes de programação locais em “Cautela” porque os modelos disponíveis eram muito limitados para contextos complexos de engenharia de software e não representavam uma solução convincente para as preocupações que deveriam resolver. Com os agentes de programação sendo adotados de forma muito mais ampla do que há 18 meses e os modelos open-weight amadurecendo rapidamente, nossas equipes perceberam que os modelos de programação locais são eficazes para lidar com a preguiça na seleção de camadas de modelo.

Modelos que podem ser executados localmente, como o Qwen3.8-27B, provaram ser particularmente úteis para cargas de trabalho que não exigem um modelo de ponta, incluindo indexação de codebase, parsing e sumarização de logs estruturados, tarefas iniciais de programação e revisões automatizadas de diffs e commits do Git. Executar essas tarefas localmente pode evitar custos incrementais de tokens em nuvem, ao mesmo tempo em que permite o processamento em segundo plano de longa duração e oferece um recurso de fallback offline.

31. MITRE ATLAS

Avalie

Sistemas agênticos e ferramentas de codificação introduzem novas arquiteturas e ameaças emergentes de segurança. O MITRE ATLAS é uma base de conhecimento de táticas e técnicas adversariais direcionadas a sistemas de IA e ML. Mais focado do que o framework mais amplo MITRE ATT&CK e projetado para complementá-lo, o ATLAS fornece uma taxonomia de ameaças para pipelines de ML, aplicações de LLM e sistemas agênticos.

Percebemos que, sem um vocabulário compartilhado, os riscos de segurança são frequentemente ignorados ou reduzidos a um mero exercício de marcar caixas. É aqui que o ATLAS pode ajudar. Com base em pesquisas sobre incidentes reais e padrões tecnológicos, as equipes podem usar o framework para apoiar a modelagem de ameaças. As equipes também podem achá-lo um complemento natural para frameworks de controle, como o SAIF, ajudando a descrever o cenário de ameaças em evolução para sistemas de IA.

32. Agentes de programação com contexto de produção

Avalie

Há muito tempo dizemos que estar consciente da produção ajuda pessoas desenvolvedoras a escreverem códigos melhores, e estamos descobrindo que o mesmo se aplica aos agentes de programação. Recomendamos agentes de programação com contexto de produção, o que significa dar aos agentes de programação acesso apropriado a dados e ferramentas de observabilidade de produção. Esse contexto os ajuda a planejar alterações eficazes, revisar o impacto das alterações implantadas e auxiliar na investigação de problemas de produção. Uma categoria crescente de ferramentas está tornando esse acesso prático. Esta recomendação é semelhante à responsabilidade de agentes no pipeline, estendendo os sensores de feedback que mantêm os agentes em autocorreção durante uma sessão de programação para além de verificações de pre-commit, alcançando a produção. Acreditamos que o desenvolvimento bem-sucedido possui múltiplas camadas de feedback, e uma engenharia agêntica bem-sucedida concede aos agentes acesso a todas elas.

© Thoughtworks, Inc. All Rights Reserved.

33. Chamada programática de ferramentas

Avalie

A chamada programática de ferramentas é um padrão de uso de ferramentas no qual o modelo escreve código em vez de emitir uma chamada de ferramenta por vez. Em vez de fazer um ciclo de ida e volta (round-trip) pelo modelo para cada operação, ele combina chamadas de ferramentas em um programa que é executado em um ambiente de execução, realizando loops, filtragens e a formatação dos resultados. Apenas os resultados selecionados ou a saída final são retornados ao contexto do modelo.

O conceito é compartilhado por várias implementações, embora a nomenclatura ainda não esteja padronizada. A Cloudflare e o Pi o chamam de Code Mode; a Anthropic o chama de Programmatic Tool Calling; e um artigo de engenharia anterior da Anthropic descreve a execução de código com MCP. O DeepSeek Harness (DSH) também oferece um PTC Mode intimamente relacionado.

As implementações diferem principalmente em relação ao local onde o ambiente de execução roda: em um contêiner do provedor do modelo, no harness do agente ou, mais recentemente, por meio de um servidor MCP em Code Mode. Este último permite que clientes MCP usem chamadas programáticas de ferramentas sem implementar seu ambiente de execução próprio. No entanto, isso pode reduzir a granularidade das aprovações do lado do cliente, de modo que a autorização e a aprovação para operações individuais ainda devem ser impostas pelo servidor e seus manipuladores de ferramentas.

34. Revisão semântica de textos editados por IA

Avalie

Agentes de programação produzem e revisam artefatos em prosa, como especificações, ADRs e planos, muito mais rápido do que humanos. Diffs de texto escondem a intenção sob reformulações e ruídos de formatação, tornando difícil ver se um plano mudou significativamente. A revisão semântica de textos editados por IA adiciona uma camada sobre o diff de texto: um LLM analisa mudanças no significado, como fatos alterados, declarações conflitantes, critérios de aceitação redefinidos e mudanças de escopo. Semelhante à revisão automatizada de PR, mas para textos em prosa, ela pode ser implementada como uma skill de agente que roda sempre que um agente de programação edita um documento. Vimos equipes da Thoughtworks usarem essa técnica para reduzir a carga cognitiva e a fadiga de revisão, ajudando pessoas revisoras a entenderem melhor as alterações no trabalho planejado antes da implementação.

35. Testes como dados em vez de testes como código

Avalie

Testes como dados em vez de testes como código significa expressar casos de teste como dados estruturados, tipicamente YAML ou JSON, em vez de código de teste. Um harness de teste específico do projeto lê esses dados e executa o código subjacente: o harness fornece os blocos de construção, enquanto os casos de teste definem como eles são montados. A técnica não é nova; BDD com Gherkin é um excelente exemplo. O que a torna relevante novamente é que os agentes de programação por IA agora escrevem mais código, incluindo testes. Percebemos que um esquema definido restringe melhor o que eles produzem, tornando os testes mais fáceis de gerar e revisar, com menos código de teste gerado por IA.

© Thoughtworks, Inc. All Rights Reserved.

Algumas de nossas equipes também viram um caso de uso atraente na modernização de sistemas legados, em que os mesmos casos de teste expressos como dados podem ser executados tanto no código legado quanto no moderno, apenas trocando o adaptador do harness de teste. Isso ajuda as equipes a verificar se o sistema modernizado se comporta como o original, pelo menos para os casos cobertos pelos testes. O uso dessa técnica com IA ainda é relativamente novo para nós, e não a aplicamos de forma ampla. No entanto, os resultados até agora são promissores, e achamos que vale a pena avaliar.

36. Inchaço de instruções para agentes

Cautela

As instruções para agentes de programação, como orientações em AGENTS.md, habilidades de processo e fluxos de trabalho, tendem a crescer ao longo do tempo. Embora cada nova regra possa ser útil isoladamente, esse acúmulo frequentemente resulta no inchaço de instruções para agentes. As instruções se tornam longas, contraditórias e desatualizadas, especialmente se as equipes praticarem rendição cognitiva a skills de agentes e importarem conjuntos inteiros de instruções externas.

Primeiro, converta todas as instruções possíveis em hooks ou verificações determinísticas. Em seguida, reavalie as instruções a cada novo modelo e exclua aquelas das quais ele não precisa mais — testes de progressão, em vez de testes de regressão. A Anthropic removeu mais de 80% do prompt de sistema do Claude Code em seus modelos mais recentes sem perda mensurável, já que os modelos de ponta agora seguem instruções de forma mais confiável. A revelação progressiva de contexto mantém as instruções restantes fora do caminho até que sejam necessárias.

37. Rendição cognitiva a skills de agentes

Cautela

Há uma tendência crescente de rendição cognitiva a skills de agentes, um antipadrão no qual as equipes adotam grandes bibliotecas de skills selecionadas e mantidas pela comunidade ou frameworks pesados de agentes de IA sem entender completamente o que eles fazem. Da mesma forma que importar cegamente grandes dependências de software, as pessoas desenvolvedoras podem tratar essas skills como funcionalidades prontas para uso e perder de vista como o sistema subjacente funciona. Esse não é um problema novo; exemplos semelhantes incluem copiar e colar código sem entendê-lo ou confiar demais em padrões de projeto genéricos. No entanto, a proliferação de ferramentas de IA tornou o problema muito mais evidente.

Além da perda de compreensão do sistema, abrir mão da avaliação crítica introduz riscos de segurança: instruções em prompts não revisadas podem chegar aos pipelines de execução. Skills de fontes diferentes também podem entrar em conflito quando não tiverem sido projetadas para funcionar juntas. É importante inspecionar e revisar as instruções das skills antes de as adotar, em vez de usar indistintamente skills genéricas de agentes. À medida que os modelos subjacentes se tornam mais capazes, skills genéricas mantidas pela comunidade podem se tornar desnecessárias ou obsoletas, tornando mais valioso selecionar e manter skills específicas de domínio adaptadas ao contexto do projeto.

© Thoughtworks, Inc. All Rights Reserved.

38. Shadow IT acelerada por IA

Cautela

Observamos muitas pessoas não desenvolvedoras migrando de plataformas no-code para agentes de programação como o Claude Code e o OpenAI Codex. Celebramos o fato de as pessoas resolverem seus próprios problemas, mas a shadow IT acelerada por IA, seja criada por pessoas não programadoras ou por desenvolvedoras, traz riscos. O primeiro deles é a segurança. O segundo é que softwares oriundos de fora do desenvolvimento tradicional costumam carecer de testes, esteiras de implantação e observabilidade. Isso é especialmente importante quando a ferramenta de produtividade de uma pessoa é adotada por toda a sua equipe ou departamento. As organizações devem priorizar a governança com limites e diretrizes para aplicações geradas por IA. Em nossa experiência, é melhor fornecer às colegas ambientes e enterprise harnesses para uma experimentação segura do que apagar seu entusiasmo. Seja firme em relação à segurança e ofereça às ferramentas úteis um caminho claro para suporte.

39. Proliferação de documentação gerada por IA

Cautela

A proliferação de documentação gerada por IA ocorre quando as equipes usam agentes de IA e ferramentas de desenvolvimento baseadas em especificações (SDD), como Spec-kit ou BMAD, para gerar grandes volumes de arquivos Markdown e documentação. Como a IA pode produzir documentações polidas e de aparência profissional em segundos, o resultado pode criar uma ilusão de qualidade de software e rápido progresso. Ao contrário do código-fonte, no entanto, a documentação raramente passa por portas de qualidade automatizadas ou testes de CI. À medida que os projetos evoluem, os repositórios acumulam rapidamente especificações verbosas, sem manutenção e contraditórias que as pessoas desenvolvedoras não têm tempo para limpar, resultando em uma caótica “grande bola de lama de documentação”.

Essa superabundância de documentação obsoleta é particularmente problemática em fluxos de trabalho de agentes. Quando agentes de programação leem um repositório repleto de registros de decisões arquitetônicas desatualizados ou arquivos de especificação conflitantes, esses documentos redundantes e contraditórios consomem um contexto valioso e fornecem informações não confiáveis. Em vez de ajudar no desenvolvimento, o contexto obsoleto pode desorientar as ferramentas de IA e resultar em uma geração de código ruim. Para manter a eficácia dos agentes, as equipes devem tratar o Markdown como um artefato de primeira classe, dedicando um esforço humano deliberado para podar, refinar e consolidar os arquivos de contexto gerados, garantindo que o contexto do agente permaneça claro, relevante e confiável.

40. Uso indiscriminado de modelos de ponta

Cautela

Às vezes nos pegamos usando por padrão o modelo mais capaz disponível em qualquer tarefa que exija um LLM, sem prestar atenção ao custo, à latência ou à capacidade necessária. Chamamos isso de uso indiscriminado de modelos de ponta. Os modelos de ponta podem custar significativamente mais e executar mais lentamente do que seus equivalentes mais ágeis. Trabalhos rotineiros como exploração, edições simples ou resumos são frequentemente melhor atendidos por um modelo menor ou até mesmo modelos open-weight. Mude de nível manualmente, por subagente ou por meio de configurações do harness que alternem automaticamente com base na complexidade ou etapa da tarefa. Cada modelo tem seu próprio cache de prompt, portanto mude em limites naturais — uma

© Thoughtworks, Inc. All Rights Reserved.

nova sessão, um subagente ou na transição do planejamento para a implementação — em vez de por requisição. Um alerta sobre o nosso alerta: modelos mais capazes às vezes concluem tarefas usando menos tokens, então use o perfilamento de contexto para verificar se a sua otimização está funcionando.

41. Volume de código produzido como medida de produtividade

Cautela

Em um Radar anterior, alertamos contra o uso do volume de código produzido como medida de produtividade. À medida que a indústria pressiona pela adoção de IA e busca métricas tangíveis, vemos cada vez mais métricas baseadas em atividade, como a porcentagem de código gerado por IA ou tokens consumidos, sendo tratadas como medalhas de honra. Embora tais métricas possam indicar níveis de adoção, elas são péssimos indicadores para ganhos reais de produtividade.

Para medir a produtividade de forma significativa, continuamos recomendando frameworks consolidados, como as métricas do DORA e qualidade de colaboração, além de vincular as melhorias diretamente aos resultados de negócios. Tratar métricas de atividade como metas, em vez de sinais, aumenta o risco de cair na armadilha da Lei de Goodhart. Já vimos pessoas desenvolvedoras gerando trabalho inútil com IA apenas para atingir essas metas.

42. Teatro de supervisão humana

Cautela

O teatro de supervisão humana é manter uma pessoa humana no loop de um agente como uma etapa de aprovação vazia, em vez de uma verificação real. O humano aprova ações, prompts ou diffs que não consegue avaliar de forma realista.

A etapa de aprovação funciona como um sumidouro de responsabilidade: ninguém realmente se responsabiliza pela decisão. Em vez disso, nosso conselho é fazer uma triagem. Reserve a revisão humana para alterações que um humano possa julgar genuinamente, usando supervisão humana baseada em níveis de risco. Para o restante, mude o foco da revisão de alterações individuais para o direcionamento do sistema por meio de testes de mutação, testes baseados em propriedades, modelos formais como TLA+ e Quint e revisão semântica assistida por agentes.

43. Meat proxy

Cautela

Um meat proxy é uma pessoa que repassa sem pensar o resultado não verificado de um LLM para um colega, encaminhando-o sem ler, validar ou adicionar um julgamento substantivo. É uma violação da boa etiqueta de IA e, em nossa infeliz experiência, um problema crescente. Quando você atua como um meat proxy, transfere o fardo da verificação para a pessoa seguinte e se exime de sua própria responsabilidade de garantir um trabalho de qualidade. Isso é semelhante ao teatro de supervisão humana, com a diferença de que se trata de uma autoria falsa em vez de uma aprovação falsa. Em nenhum dos casos o ser humano adiciona valor. O rascunho das diretrizes de IA da KDE resume bem: “Não seja um meat proxy.”

© Thoughtworks, Inc. All Rights Reserved.

44. Compressão ingênua de contexto

Cautela

À medida que os custos de tokens aumentam, as equipes recorrem cada vez mais a ferramentas como Headroom, RTK e Tamp para encolher seus prompts. No entanto, aconselhamos contra a compressão ingênua de contexto, que pode introduzir vários problemas sutis. Primeiro, ela frequentemente interfere com o prompt caching; reduzir a contagem de tokens enquanto se perdem acertos de cache (cache hits) pode levar a contas finais mais altas. Segundo, uma compressão agressiva causa perda de informação, degradando o desempenho e a precisão dos loops de agentes. Terceiro, algumas ferramentas dependem de um LLM para resumir o contexto durante a compressão, adicionando latência e custos que anulam o objetivo original.

Antes de adotar essas ferramentas, analise o uso de contexto e avalie rigorosamente tanto o custo líquido quanto a precisão das tarefas; a redução de tokens não significa nada se os caches de prompt falharem ou se o desempenho dos agentes degradar. Dê preferência a ferramentas que comprimam as saídas das ferramentas enquanto preservam referências aos dados brutos e que façam a poda (prune) do histórico apenas de uma vez nas fronteiras das tarefas para proteger os prefixos de cache.

45. Pull requests empilhados

Cautela

Pull requests empilhados dividem uma grande alteração em uma cadeia ordenada de pequenos pull requests (PRs) dependentes. Em vez de ter como alvo o branch principal do repositório, cada PR subsequente direciona para o branch diretamente abaixo dele na stack. Vimos um grande aumento no interesse por essa técnica recentemente devido à rápida adoção de ferramentas de programação assistidas por IA. As equipes adotam essa abordagem para reduzir a latência de revisão e o tempo ocioso das pessoas desenvolvedoras. Ao avaliar camadas pequenas e isoladas, as pessoas revisoras enfrentam menor sobrecarga cognitiva.

O suporte de ferramentas também melhorou para apoiar esse fluxo de trabalho; o GitHub introduziu recentemente suporte nativo. Apesar das ferramentas melhores, colocamos essa técnica no anel “Cautela”, pois muitas vezes ela serve como uma solução alternativa para uma decomposição inicial mal feita. O empilhamento não cria problemas de decomposição; ele expõe aqueles que já existiam. O risco não é a técnica em si; é usá-la como substituto para esse planejamento. O empilhamento habitual e indiferenciado pode facilmente se tornar uma desculpa para escrever uma única alteração enorme. Mantemos a posição de que o melhor hábito é especificar, dividir, dimensionar e entregar o trabalho como unidades pequenas e independentes.

© Thoughtworks, Inc. All Rights Reserved.

Plataformas



Adote

46. VictoriaMetrics

Experimente

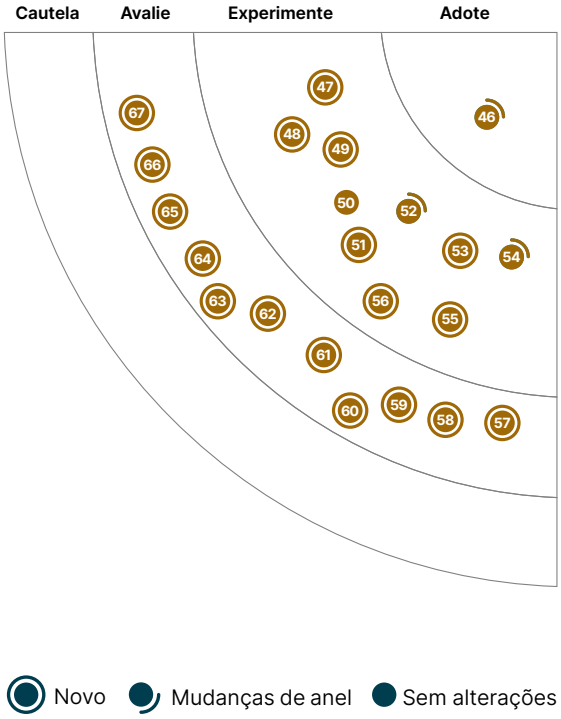
47. Kafka share groups
48. KServe
49. LibreChat
50. LiteLLM
51. Microsoft Copilot Retrieval API
52. Restate
53. Talos Linux
54. Temporal
55. vCluster
56. VictoriaLogs

Avalie

57. Agent Client Protocol (ACP)
58. Agent Plugins
59. agentgateway
60. AWS DevOps Agent
61. Claude Managed Agents
62. Modelos de decisão
63. KAITO
64. llm-d
65. Netris
66. Omnigent
67. VAST DataStore

Cautela

—



46. VictoriaMetrics

Adote

Movemos o VictoriaMetrics para "Adote" porque ele amadureceu e se tornou uma escolha confiável para organizações que operam plataformas de observabilidade autogerenciadas de grande escala. Como um banco de dados de séries temporais de código aberto e alto desempenho, ele processa grandes volumes de métricas com alta cardinalidade que podem desafiar o Prometheus. Em um de nossos projetos, um cluster do VictoriaMetrics lida com cinco milhões de amostras por segundo com 20 a 30 rótulos por métrica, demonstrando sua adequação para implantações de grande escala.

A plataforma oferece uma versão simples de nó único para cargas de trabalho menores, bem como uma arquitetura em cluster que separa ingestão de dados, consulta e armazenamento em componentes independentemente escaláveis. Ela se integra a arquiteturas de observabilidade existentes ao oferecer suporte à ingestão nativa de OpenTelemetry, APIs do Prometheus e suporte a remote write, permitindo que as equipes a adotem sem substituir sua instrumentação e painéis existentes. Seja usado como armazenamento de longo prazo por trás do Prometheus ou como uma stack completa de métricas com componentes como vmagent e vmalert, o VictoriaMetrics é uma opção atraente para infraestruturas onde a retenção, a cardinalidade ou o custo se tornaram um desafio.

47. Kafka share groups

Experimente

O recurso Kafka Share Groups (introduzido no KIP-932, pronto para produção desde o Kafka v4.2) traz um novo modelo cooperativo de consumo de eventos para o Kafka, permitindo um comportamento semelhante a filas. Historicamente, os grupos de consumidores tradicionais do Kafka impunham um gargalo de concorrência porque o número de consumidores ativos não podia exceder o número de partições. Os share groups removem essa restrição: registros individuais são entregues a qualquer consumidor disponível no pool sob um bloqueio (lock) de aquisição a cada registro, que o consumidor deve confirmar explicitamente. Em caso de falha ou tempo limite (timeout), o registro é liberado e reentregue a qualquer consumidor no grupo, até um número máximo configurável de tentativas de entrega. Isso garante a entrega de "pelo menos uma vez" (at-least-once) com a possibilidade de processamento duplicado. Registros que esgotam suas tentativas de entrega são encaminhados para arquivamento; o suporte a dead-letter queues (KIP-1191) ainda não está disponível.

A contrapartida é que os share groups não garantem a ordenação por partição, pois os registros são despachados e reentregues de forma independente. No entanto, para equipes que anteriormente precisavam executar tanto o Kafka quanto um sistema externo de filas, os share groups podem simplificar a arquitetura ao consolidar as cargas de trabalho no Kafka.

48. KServe

Experimente

O KServe é um operador de Kubernetes de código aberto que gerencia a inferência de IA preditiva e generativa por meio de recursos personalizados. Em vez de atuar como um motor de inferência em si, ele orquestra runtimes de serviço como vLLM, SGLang e NVIDIA Triton, separando o ciclo de vida de um serviço de modelo do motor que o executa. À medida que as organizações constroem AI factories, operar a inferência na infraestrutura de GPUs do Kubernetes torna-se um problema de gerenciamento de frota, em vez de uma questão de executar servidores de modelos individuais. O KServe aborda

© Thoughtworks, Inc. All Rights Reserved.

isso com dois recursos personalizados: InferenceService para aprendizado de máquina convencional e cargas de trabalho mais simples de LLMs, com scale-to-zero opcional baseado em Knative, e LLMInferenceService para serviço avançado de LLMs. Desenvolvido sobre o llm-d e a Gateway API Inference Extension, o LLMInferenceService suporta inferência multinó (multi-node), desagregação de pré-preenchimento e decodificação (prefill/decode disaggregation) e roteamento baseado em KV-cache que direciona requisições para réplicas que já possuem o prefixo em cache relevante. O KServe é um projeto no estágio Incubating da CNCF, e vários de nossos clientes o utilizam em produção. Nossas equipes o preferem em relação a outras ferramentas desse espaço ao construir plataformas internas de inferência no Kubernetes. O LLMInferenceService é mais recente que o InferenceService, e seu ecossistema ainda está em evolução.

49. LibreChat

Experimente

O LibreChat é uma plataforma de chat de IA open-source e auto-hospedada que permite que pessoas de toda a organização acessem modelos de vários provedores por meio de uma única interface. Assim como o Open WebUI, ele oferece logon único (single sign-on), permissões baseadas em funções e RAG sobre documentos carregados. Seus recursos de agentes o tornam mais do que uma interface de chat. Os administradores decidem quais servidores MCP estão disponíveis. Os usuários cuja função permite podem, então, criar agentes nesses servidores sem escrever código e compartilhá-los por meio de um marketplace integrado.

Uma de nossas equipes construiu a plataforma interna de IA de um cliente sobre o LibreChat. A equipe escreveu servidores MCP para Jira, Confluence, Salesforce, Grafana e para os sistemas próprios de pedidos, estoque e receita do cliente. A própria equipe do cliente construiu os agentes: a área de vendas criou um para verificar disponibilidade e registrar pedidos, enquanto a de operações criou um para gerenciar contratos. Essa experiência é o motivo pelo qual o estamos colocando em “Experimente”. Ele oferece a pessoas fora da engenharia uma forma controlada de acessar ferramentas de IA aprovadas, que é o que recomendamos como resposta à shadow IT acelerada por IA. A ClickHouse adquiriu o LibreChat em novembro de 2025, e o projeto continua sob licença MIT.

50. LiteLLM

Experimente

O LiteLLM tornou-se um gateway de IA open-source amplamente utilizado. Várias de nossas equipes roteiam suas chamadas de modelos por meio dele, seja como um proxy autônomo (standalone) ou como um sidecar. Ele suporta mais de 100 provedores de LLM e oferece roteamento, tradução de provedores, autenticação, orçamentos e guardrails. O núcleo do seu proxy está sendo reescrito de Python para Rust para melhorar o desempenho, mantendo o mesmo SDK e configuração em Python.

Nesta edição, estamos movendo o LiteLLM de “Linguagens & Frameworks” para “Plataformas”, pois agora o vemos mais como infraestrutura do que como uma biblioteca. Mantemos o LiteLLM em “Experimente” em vez de “Adote” principalmente devido a preocupações de segurança: no momento em que este texto foi escrito, mais de 25 CVEs foram relatadas apenas este ano. As equipes que o utilizam devem aplicar os patches de segurança prontamente. Para governar também o tráfego MCP e de agente-para-agente, considere combiná-lo com o agentgateway.

© Thoughtworks, Inc. All Rights Reserved.

51. Microsoft Copilot Retrieval API

Experimente

A Microsoft Copilot Retrieval API é uma solução simplificada para recuperar dados do ecossistema da Microsoft para geração aumentada por recuperação (RAG). A API recupera trechos de texto relevantes do SharePoint, OneDrive e conteúdos expostos por meio de conectores do Copilot. Ela aplica filtragem de segurança com base no usuário solicitante e respeita os controles de acesso definidos no tenant.

Pessoas desenvolvedoras podem construir agentes de IA com contexto organizacional sem a necessidade de manter repositórios de dados e pipelines de recuperação customizados, deixando tarefas como fragmentação (chunking), indexação, sincronização de dados, reindexação e controle de acesso a cargo da Microsoft. A Microsoft Copilot Retrieval API é particularmente útil para sistemas de IA que precisam do conhecimento organizacional já armazenado nas ferramentas da Microsoft, e nossa experiência com sua adoção tem sido positiva.

52. Restate

Experimente

O Restate é uma plataforma de execução durável para construir aplicações tolerantes a falhas e com estado (stateful). Como ele registra cada etapa da execução em log (journals), um serviço que falha retoma de onde parou. O Restate também garante comunicação exatamente uma vez (exactly-once) entre serviços. Uma aplicação combina Serviços Básicos para funções sem estado (stateless), Objetos Virtuais para entidades concorrentes com estado e Workflows para processos de múltiplas etapas.

Da última vez que escrevemos sobre o Restate, estávamos avaliando a ferramenta em um grande sistema de seguros. Agora, ele é uma parte central desse sistema, automatizando todo um processo de negócios de seguros. A equipe confia em sua recuperação de estado e controle de concorrência e não precisou escrever seu próprio código de recuperação. Essa experiência é o motivo pelo qual estamos movendo a ferramenta para “Experimente”. O Restate é adequado para processos de longa duração que aguardam uma pessoa no circuito (human in the loop) ou que fazem várias tentativas (retries) contra sistemas legados. Ele cumpre o mesmo papel que o Temporal, que também avança para “Experimente” nesta edição. Os SDKs são licenciados sob MIT, mas o servidor utiliza a Business Source License, o que impede oferecê-lo a terceiros como um serviço gerenciado.

53. Talos Linux

Experimente

Talos Linux é uma distribuição Linux imutável e minimalista para executar Kubernetes. Ela não possui SSH, gerenciador de pacotes nem shell interativo. Você configura os nós de forma declarativa com o talosctl por meio de uma API gRPC protegida por mTLS, e as atualizações substituem toda a imagem do SO. Esse design reduz a superfície de ataque e o drift de configuração associados a distribuições de uso geral, como o Ubuntu, tornando cada nó um dispositivo substituível, no espírito dos servidores imutáveis.

Uma de nossas equipes executa Kubernetes autogerenciado na Talos em uma grande frota bare-metal para uma plataforma de nuvem de IA e considerou o plano de controle fácil de provisionar. A maioria das organizações usa um serviço gerenciado, como o Amazon EKS, e nunca interage com o SO do nó, de modo que a Talos é voltada principalmente para equipes que operam seus próprios nós. O Bottlerocket adota uma abordagem semelhante na AWS, e o Flatcar Container Linux também tem ganhado tração. Em 2026, a Yardi adquiriu a Sidero Labs, desenvolvedora da Talos. A Talos permanece como código aberto sob a licença MPL-2.0, mas as equipes devem acompanhar os rumos que o novo proprietário tomará.

© Thoughtworks, Inc. All Rights Reserved.

54. Temporal

Experimente

Temporal é uma plataforma de execução durável para processos de negócio complexos que exigem fortes garantias de confiabilidade. O Temporal fornece fluxos de trabalho com retentativas (retries), persistência de estado intermediário, timeouts configuráveis, pontos de verificação com intervenção humana (human-in-the-loop), capacidade de retomada (resumability) e visibilidade do progresso de execução. Ele oferece tanto uma opção de código aberto para auto-hospedagem quanto um serviço gerenciado na nuvem.

Encontramos casos de uso atraentes para o Temporal ao construir plataformas internas para equipes de desenvolvimento e planos de controle para infraestrutura de nuvem para IA. Esses sistemas tipicamente envolvem múltiplas tarefas de provisionamento de infraestrutura de longa duração cujo estado precisa ser reconciliado posteriormente. As retentativas integradas do Temporal, o agendamento flexível de filas de workers, a capacidade de retomada, a visibilidade e a execução durável ajudam as equipes a desenvolver sistemas desse tipo com mais resiliência.

55. vCluster

Experimente

O vCluster fornece planos de controle virtualizados do Kubernetes para ambientes multi-tenant. Ele oferece um meio-termo entre multi-tenancy em nível de namespace, em que os tenants compartilham APIs do cluster e definições de recursos personalizadas (CRDs), e o provisionamento de clusters completamente independentes, o que aumenta a sobrecarga operacional. Com o vCluster, cada tenant tem seu próprio servidor de API do Kubernetes, controller manager, data store, RBAC e CRDs. O plano de controle é executado como uma carga de trabalho leve na infraestrutura subjacente, permitindo que ferramentas padrão como kubectl, Helm e controladores de GitOps interajam com cada tenant de forma independente.

Utilizamos o vCluster com sucesso para construir serviços gerenciados do Kubernetes e ambientes de nuvem de IA onde equilibrar o isolamento de tenants em relação aos custos operacionais é crítico. É particularmente atraente para plataformas de IA onde diferentes clientes podem precisar de seus próprios operadores de GPU, escalonadores, frameworks de inferência e CRDs. Para cargas de trabalho confiáveis, planos de controle virtualizados podem compartilhar recursos de computação subjacentes. Para ambientes de GPU voltados a clientes, nós worker dedicados podem fornecer um isolamento de carga de trabalho mais forte, mantendo o gerenciamento consolidado do plano de controle. Dado o seu sucesso em ambientes de produção de clientes, estamos colocando o vCluster em “Experimente”.

56. VictoriaLogs

Experimente

Desenvolvido pelos criadores do VictoriaMetrics, o VictoriaLogs é uma plataforma de gerenciamento de logs de código aberto otimizada para ingestão de alta vazão, armazenamento compacto e consultas rápidas de grandes volumes de logs. Em vez de exigir índices pré-definidos, seu mecanismo de armazenamento organiza e compacta campos de log em blocos. Ele utiliza filtros de Bloom e índices de timestamp para evitar varreduras desnecessárias e paraleliza consultas entre núcleos de CPU.

Tivemos experiências positivas usando o VictoriaLogs enquanto construíamos a observabilidade para um sistema de infraestrutura de nuvem de IA. Nessa escala, os serviços do Kubernetes geram grandes volumes de telemetria, tornando a vazão de ingestão, o desempenho de consultas

© Thoughtworks, Inc. All Rights Reserved.

e a eficiência de armazenamento em considerações críticas. Em comparação com plataformas tradicionais de logs baseadas em busca como Elasticsearch, o VictoriaLogs oferece um backend de logs capaz com menor complexidade operacional e menor consumo de recursos. Também o achamos particularmente atraente para logs estruturados de alta cardinalidade e buscas de campos arbitrários quando comparado ao Grafana Loki, porque não exige o mesmo grau de planejamento de rótulos. O VictoriaLogs também inclui a linguagem de consulta LogsQL para busca em texto completo, transformações e agregação. Ele se integra ao Grafana para exploração interativa e alertas, enquanto a compatibilidade com os protocolos de ingestão do OpenTelemetry, Elasticsearch e Loki permite que as equipes o adotem sem vincular as aplicações a um pipeline de logs proprietário.

57. Agent Client Protocol (ACP)

Avalie

Embora agentes de programação baseados em CLI se sobressaiam na implementação em todo o repositório e na execução de fluxos de trabalho, pessoas desenvolvedoras ainda precisam de acesso inline aos agentes em suas IDEs para exploração rápida de código, correção interativa de bugs e edição sensível ao contexto. Tradicionalmente, isso exige plugins individuais para cada IDE, criando uma sobrecarga de integração e prendendo as pessoas desenvolvedoras a combinações específicas de editor e agente. O Agent Client Protocol (ACP) aborda esse problema estabelecendo uma fronteira padrão e aberta entre os agentes de IA e os ambientes de execução (hosts), seja o VS Code, JetBrains, uma janela de terminal ou um pipeline de CI/CD.

Assim como o Language Server Protocol (LSP) padronizou o suporte a linguagens entre editores, o ACP depende do JSON-RPC para definir o ciclo de vida do agente e as primitivas de interação, incluindo criação e retomada de sessão, atualizações em tempo real (streaming), solicitações de permissão e cancelamento. O ACP complementa as camadas de interoperabilidade existentes: enquanto o Model Context Protocol (MCP) lida com a integração de agente para ferramenta, o ACP lida com o controle de cliente para agente, tornando todo o harness do agente um componente intercambiável.

58. Agent Plugins

Avalie

Agent Plugins é um padrão aberto para empacotar componentes reutilizáveis de agentes em plugins portáteis. A primeira versão, lançada por uma colaboração multifornecedora incluindo Amazon, Microsoft e OpenAI, define um formato compartilhado para habilidades de agentes e servidores do Model Context Protocol (MCP). Embora a especificação seja deliberadamente mínima, um padrão aberto é um passo importante para um ecossistema que até agora estava fragmentado. Descobrimos que o “piso de interoperabilidade mínimo” facilita a extensão de um repositório existente de habilidades para que ele também funcione como um Agent Plugin.

59. agentgateway

Avalie

agentgateway é um gateway de alto desempenho para tráfego relacionado a LLMs, hospedado pela Linux Foundation. Ele suporta inferência de LLM, comunicação via MCP e A2A por meio de um protocolo comum, tanto em versões standalone quanto em Kubernetes. Seus recursos principais incluem roteamento baseado em custo e latência, fallbacks, limites de gastos, guardrails, controle de ferramentas MCP e observabilidade com OpenTelemetry. Estamos particularmente interessadas na versão para Kubernetes por causa de sua integração profunda com infraestrutura de LLMs auto-hospedada. Ele pode usar métricas do Kubernetes para rotear requisições para uma réplica

© Thoughtworks, Inc. All Rights Reserved.

respaldada por GPU que já tenha o contexto relevante em cache na memória ou para um servidor menos utilizado com base no uso real da GPU. Essas capacidades fazem do agentgateway uma opção que vale a pena avaliar para organizações que executam stacks de inferência auto-hospedadas.

60. AWS DevOps Agent

Avalie

AWS DevOps Agent é uma plataforma gerenciada de agentes da AWS para investigação de incidentes orientada a IA. Quando um alarme, chamado ou mensagem de chat dispara uma investigação, a plataforma mapeia os recursos da aplicação e seus relacionamentos, correlaciona telemetria com implantações recentes e alterações de código, testa hipóteses e apresenta causas raízes prováveis e mitigações para que uma pessoa engenheira possa atuar. Integrações nativas com Datadog, Dynatrace, New Relic, Splunk e Grafana permitem que alertas dessas ferramentas disparem investigações e habilitam a plataforma a consultar sua telemetria por meio de integrações baseadas em MCP. Conexões com GitHub, GitLab, ServiceNow, PagerDuty e Slack, juntamente com servidores MCP personalizados, estendem seu alcance a sistemas on-premises e proprietários.

Nossas equipes estão experimentando o serviço em produção para investigar alarmes em tempo real. Ainda é cedo, mas os resultados parecem promissores, sugerindo um potencial para análise de causa raiz mais rápida e menor esforço operacional manual. Nossa principal ressalva é a transparência de custos. A plataforma cobra por agente-segundo gasto na investigação de um incidente, tornando os custos difíceis de prever. Os gastos podem se acumular rapidamente sem um monitoramento rigoroso. O serviço está disponível publicamente apenas desde março de 2026, por isso recomendamos avaliá-lo com validação humana de cada achado, controle rígido sobre quais eventos disparem investigações e acompanhamento ativo de custos.

61. Claude Managed Agents

Avalie

O Claude Managed Agents é o serviço hospedado da Anthropic para a execução de agentes de uso geral. As equipes definem o modelo, prompt, ferramentas, servidores MCP e habilidades de cada agente. A Anthropic fornece o restante de um managed agents runtime: o loop do agente, sandbox, agendamento e observabilidade. Os agentes são executados de acordo com um agendamento ou quando uma aplicação inicia uma sessão por meio da API. Execuções agendadas são adequadas para trabalhos recorrentes em segundo plano, como triagem de alertas e proposição de correções na forma de pull requests para revisão humana.

As equipes podem começar sem escrever código, por meio do console, ou usar integrações baseadas em código para maior flexibilidade e controle. Definições de agente, ambiente e agendamento podem ser armazenadas em arquivos aplicados por meio de um pipeline de CI. Um cofre de credenciais injeta credenciais em requisições de saída sem expô-las ao agente. Uma de nossas equipes internas escolheu o Claude Managed Agents em vez de outras plataformas de agentes, incluindo o Gemini Enterprise Agent Platform, por ser mais fácil de entender e mais rápido de entregar.

Estamos colocando-o em “Avalie” porque nossa experiência está limitada a essa equipe e o serviço está em fase beta. Ao contrário do Amazon Bedrock AgentCore, ele suporta apenas o harness de agentes da Anthropic e modelos Claude. A Anthropic cobra pelo tempo de execução da sessão, bem como por tokens. Como o serviço armazena sessões, ele não é elegível para retenção zero de dados. Sandboxes auto-hospedados podem manter a execução de ferramentas dentro da sua rede, mas a orquestração e os resultados das ferramentas ainda passam pela Anthropic.

© Thoughtworks, Inc. All Rights Reserved.

62. Modelos de decisão

Avalie

Uma nova classe de modelos de IA chamada modelos de decisão (decision models) surgiu desde o último Radar. A TypeSafe AI lançou o Jev, que ela chama de um modelo 'System One' construído para tomar decisões rápidas e estruturadas. Vários outros modelos se seguiram, incluindo Laya, Nimble e Kev. Em contraste com modelos de raciocínio deliberativo, os modelos de decisão fornecem valores estruturados predefinidos como saída em vez de texto gerado. Em vez de usar aprendizado por reforço com feedback humano (RLHF) para otimizar preferências humanas, o Jev usa aprendizado por reforço com decisões calibradas (RLCD) para otimizar para decisões bem calibradas, onde as probabilidades previstas refletem com que frequência um resultado realmente ocorre. Suas saídas menores e estruturadas permitem um processamento mais rápido e barato. Os modelos de decisão recebem um estado e um conjunto de perguntas. Cada pergunta feita em relação ao estado deve corresponder a um dos três tipos: Noul para perguntas do tipo sim/não que retornam a probabilidade de uma resposta "sim", Choice para opções categorizadas de múltipla escolha ou Score para avaliações em uma escala ordinal ordenada. Os casos de uso de modelos de decisão incluem Jev as a judge como uma alternativa a LLM como juiz, roteamento de modelos e ferramentas, portões de políticas (policy gates) e rotulagem em lote (bulk labeling). Recomendamos que as equipes evitem o uso excessivo de LLMs e definam critérios de avaliação para determinar se um modelo de decisão pode substituir um LLM para casos de uso de classificação.

63. KAITO

Avalie

O KAITO (Kubernetes AI Toolchain Operator) é um operador Kubernetes que automatiza a infraestrutura necessária para auto-hospedar modelos open-weight em seus próprios clusters. Em vez de configurar manualmente nós de GPU e servidores de modelos, as equipes declaram um Workspace, o recurso personalizado do KAITO para a implantação de um único modelo, especificando o modelo e o tipo de instância de GPU. O controlador do KAITO então provisiona os nós de GPU correspondentes por meio de NodePools do Karpenter, estima os requisitos de memória de GPU, ajusta o runtime e disponibiliza o modelo através do vLLM, expondo um endpoint compatível com a OpenAI via um Kubernetes Service.

Nossas equipes começaram a explorar o KAITO no AKS para hospedar modelos de linguagem de pequeno porte com menor custo e latência. Anteriormente, executar o Ollama e o vLLM diretamente em VMs com GPU exigia gerenciar drivers, escalonamento e configuração de nós manualmente. Gerenciar o ciclo de vida do modelo por meio dos mesmos manifestos declarativos usados para outras cargas de trabalho promete reduzir grande parte desse trabalho repetitivo.

No entanto, o escopo do KAITO é deliberadamente restrito. Ele lida com o provisionamento de nós e a implantação de modelos, em vez de focar na pilha de inferência mais ampla, onde o KServe é mais maduro e oferece recursos como roteamento baseado em KV-cache. O KAITO ainda é um projeto CNCF Sandbox. Seu autoprovisionamento de nós de GPU é mais maduro no AKS, e ele oferece suporte apenas para GPUs NVIDIA da geração Ampere ou mais recentes. Achamos que vale a pena avaliar, especialmente para equipes que desejam auto-hospedar modelos usando suas ferramentas Kubernetes existentes.

© Thoughtworks, Inc. All Rights Reserved.

64. llm-d

Avalie

O llm-d é uma stack de serviço de inferência distribuída de código aberto projetada para operar servidores de modelos como vLLM e SGLang em múltiplos nós do Kubernetes. Aplicações web tradicionais dependem frequentemente de roteamento sem estado (stateless) e escalabilidade horizontal, mas a inferência de LLMs é altamente dependente de estado (stateful) devido ao caching de chave-valor (KV). Em vez de tratar pods de serviço de modelos como backends HTTP intercambiáveis, o llm-d utiliza roteamento baseado em inferência para coordenar requisições em múltiplos aceleradores, estendendo a otimização além dos servidores de modelos individuais.

Construído em torno do Kubernetes e do Gateway API Inference Extension, o llm-d oferece diversos recursos adaptados ao tráfego de IA com estado (stateful). Ele suporta roteamento baseado em modelo e em KV-cache, utilizando fatores como afinidade de cache, profundidade de fila e latência prevista para orientar as decisões de escalonamento. Um recurso notável é a desagregação de pré-preenchimento e decodificação (prefill/decode disaggregation), que permite aos operadores posicionar cargas de trabalho de pré-preenchimento com uso intensivo de computação em um conjunto de GPUs e tarefas de decodificação limitadas por largura de banda de memória em outro. Isso pode melhorar a utilização de hardware e os custos de inferência. Equipes que constroem plataformas de inferência GPU em grande escala devem avaliar o llm-d como uma base emergente para sua camada de inferência, especialmente ao buscar escalar enquanto atendem a objetivos rígidos de latência.

65. Netris

Avalie

O Netris é uma plataforma de automação de rede fora de banda (out-of-band) e multi-tenancy para data centers físicos e infraestrutura de IA. Usando um modelo declarativo de infraestrutura como código acessível por meio de UI, API, CRDs do Kubernetes e Terraform, ele simplifica o gerenciamento de rede abstraindo configurações de switches em clouds privadas virtuais (VPCs) no estilo cloud, redes virtuais, listas de controle de acesso (ACLs) e balanceadores de carga de Camada 4. Um controlador central gerencia políticas enquanto agentes leves executam alterações em switches e unidades de processamento de dados (DPUs). O controlador opera fora de banda como um plano de controle de automação, permitindo que o encaminhamento de pacotes continue sem interrupções mesmo se o controlador ficar offline.

Estamos destacando o Netris por seu suporte a cargas de trabalho de IA de alta largura de banda. Ele automatiza fabrics de Ethernet VPN (EVPN) roteados, gerencia o controle de fluxo para RDMA sobre Ethernet Convergida (RoCE) e orquestra DPUs NVIDIA BlueField para multi-tenancy aplicado via hardware. Ele também se integra ao NVIDIA UFM para chaves de partição InfiniBand (PKeys) e ao NMX-C para domínios NVLink, ajudando a gerenciar limites multi-tenant em diferentes fabrics de IA por meio de uma camada de gerenciamento unificada. Estamos colocando o Netris em “Avalie” para equipes que constroem clouds privadas automatizadas ou data centers de IA, embora as equipes devam avaliar a compatibilidade com seus sistemas operacionais de switch, como SONiC, Cumulus Linux e Arista EOS, bem como os limites do suporte à orquestração bare-metal.

© Thoughtworks, Inc. All Rights Reserved.

66. Omnigent

Avalie

O Omnigent é um meta-harness de código aberto, lançado pela Databricks, que opera acima de agentes de programação como Claude Code, OpenAI Codex e Pi, bem como agentes definidos em YAML. Ele fornece uma camada comum para recursos que, de outra forma, cada harness implementaria de maneira diferente. As políticas podem limitar gastos ou exigir aprovação antes de uma ação de risco, enquanto um sandbox do sistema operacional restringe o acesso ao sistema de arquivos e à rede e substitui credenciais em requisições de saída, permitindo que o agente as use sem lê-las. As sessões são persistentes, permitindo que uma execução continue de onde parou. O Omnigent também pode se comunicar com os agentes por meio do Agent Client Protocol (ACP) em vez de envolver cada CLI.

Em um laptop, o Omnigent oferece à pessoa desenvolvedora uma caixa de entrada de sessões paralelas, de forma muito semelhante ao que o Herdr faz. Como um servidor compartilhado, ele adiciona Single Sign-On, compartilhamento de sessão em tempo real e políticas definidas por administradores, com sessões rodando localmente ou em sandboxes na nuvem como o E2B. Isso se adequa a organizações que desejam controles consistentes, independentemente de quais agentes de programação suas pessoas desenvolvedoras escolham. Ainda não temos certeza de qual seja a abstração correta em relação aos agentes de programação, e o projeto ainda se descreve como estando na fase alfa. Equipes que desejam políticas consistentes entre vários agentes de programação devem avaliar o Omnigent levando em conta essas duas ressalvas.

67. VAST DataStore

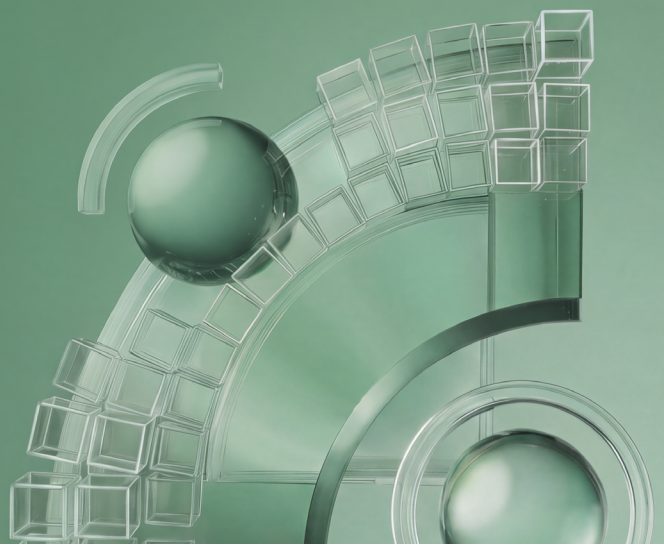
Avalie

VAST DataStore é a camada de armazenamento da plataforma VAST Data. Ela fornece acesso a arquivos (NFS e SMB), objetos (S3) e blocos (NVMe/TCP) por meio de um namespace unificado. Sua arquitetura desagregada do tipo shared-everything separa nós de computação sem estado (stateless) do armazenamento NVMe compartilhado, permitindo que a capacidade e o desempenho sejam dimensionados de forma independente. Estamos avaliando-a porque o throughput de armazenamento afeta cada vez mais a eficiência com que clusters caros de GPU operam durante o treinamento, o checkpointing e o carregamento de modelos. O DataStore permite que as cargas de trabalho mantenham o acesso convencional NFS/POSIX ou S3 enquanto usam NVMe e RDMA para uma transferência de dados mais rápida, oferecendo uma alternativa aos sistemas de arquivos paralelos especializados. A VAST também está trabalhando com a NVIDIA no S3 sobre RDMA e armazenamento compartilhado de KV-cache para inferência desagregada.

Comparado com o Ceph, que oferece armazenamento de blocos, arquivos e objetos de código aberto em hardware convencional, o DataStore é uma plataforma proprietária projetada para cargas de trabalho de NVMe, RDMA e IA. No entanto, ele é caro e prende você à plataforma de dados mais ampla da VAST, portanto pode ser mais do que as equipes precisam, a menos que estejam treinando modelos ou executando inferência em grande escala. Achamos que o VAST DataStore vale a pena ser avaliado por equipes que projetam grandes clusters de GPU para cargas de trabalho de IA e computação de alto desempenho (HPC), em que o carregamento de dados, o checkpointing e o acesso intensivo a metadados de bilhões de arquivos podem deixar GPUs caras esperando pelo armazenamento.

© Thoughtworks, Inc. All Rights Reserved.

Ferramentas



Adote

68. mise
69. OpenAI Codex
70. OpenCode

Experimente

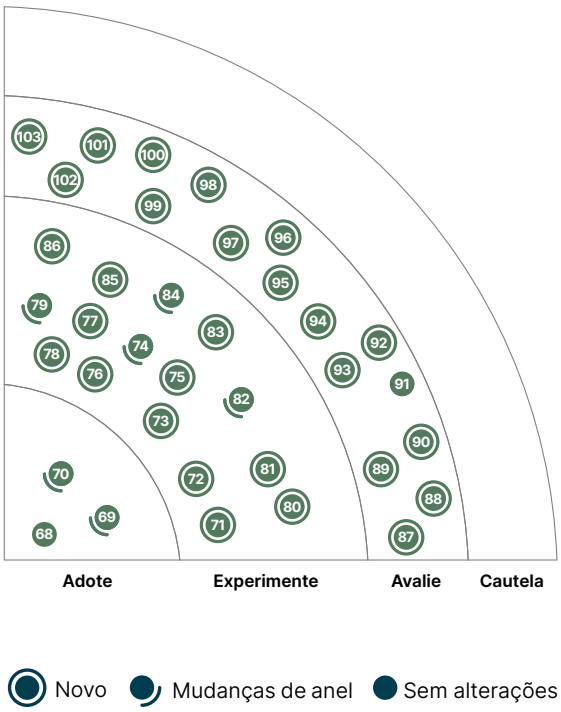
71. Act
72. Agent Package Manager
73. ast-grep
74. Beads
75. Chaos Mesh
76. Cube Core
77. Figma Code Connect
78. FluxMarkdown
79. Git AI
80. Linear
81. mattpocock/skills
82. OpenSpec
83. OpenWiki
84. Pi (agente de programação)
85. Sourcebot
86. Timefold Solver

Avalie

87. AgentsView
88. Apple container
89. Deepsec
90. DeepSeek Harness (DSH)
91. Entire CLI
92. FreeToken
93. GLM
94. Habit Hooks
95. Herdr
96. Hunk
97. nono.sh
98. OpenResearch CLI (orx)
99. Safe Chain
100. Swamp
101. Vale
102. Varlock
103. zizmor

Cautela

—



68. mise

Adote

Desde nossa última avaliação, o mise evoluiu de uma alternativa de alto desempenho ao asdf para um frontend padrão para o ambiente de desenvolvimento. Estamos mantendo-o em "Adote" porque ele consolida três preocupações fragmentadas — controle de versões de ferramentas e linguagens, gerenciamento de variáveis de ambiente e execução de tarefas — em uma única ferramenta de alto desempenho desenvolvida em Rust, configurada por meio de um arquivo declarativo `mise.toml`. O mise é fácil de configurar e funciona bem com pipelines de CI/CD. Ele também adiciona uma camada de segurança na cadeia de suprimentos por meio da integração com o Cosign e o GitHub Artifact Attestations, recurso que muitas vezes não está presente em outros gerenciadores de versão.

Para equipes que buscam padronizar a configuração de seus ambientes de desenvolvimento, o mise se tornou nossa recomendação padrão. Em ambientes poliglotas com múltiplos microsserviços, isso é especialmente útil quando as bases de código adotam novas versões de linguagem ao mesmo tempo. O melhor de tudo é que o mise também funciona com ferramentas específicas de linguagem já existentes, de modo que as equipes não precisam migrar tudo de uma só vez.

69. OpenAI Codex

Adote

O OpenAI Codex avança para "Adote", juntando-se ao Claude Code e ao OpenCode como agentes de programação focados em CLI que nossas equipes usam no dia a dia. Seu aplicativo desktop é popular entre pessoas que não programam. O Codex possui os recursos esperados para a adoção corporativa: sandboxing em nível de sistema operacional, aprovações em camadas, configuração centralizada e gerenciamento de conformidade. O loop de agente subjacente é licenciado sob Apache-2.0 e pode ser usado como harness de programação para agentes de propósito geral. Embora seja de código aberto, o Codex é gerenciado por uma empresa em vez de ser governado pela comunidade. Ele recorre à clássica estratégia do desafiante em matéria de compatibilidade: facilita a importação de configurações, habilidades e sessões de outros agentes e o uso de modelos além dos GPTs da OpenAI. Isso faz do Codex uma escolha sólida para executar modelos open-weight.

70. OpenCode

Adote

Desde o nosso último Radar, o OpenCode permanece como um dos harnesses de agentes de programação de código aberto mais proeminentes. Vimos sua adoção crescer significativamente junto com o surgimento de modelos open-weight eficazes. As equipes apreciam sua capacidade de personalização e o modo headless/servidor, o que o torna uma base atraente para organizações que constroem um enterprise harness. Por outro lado, seus recursos de compartilhamento de sessão e colaboração ainda ficam atrás de harnesses concorrentes. O OpenCode se tornou uma escolha popular para equipes que combinam agentes de programação com modelos auto-hospedados de pesos abertos e modelos locais de programação, mas ainda recomendamos avaliar alternativas emergentes, como o Pi.

© Thoughtworks, Inc. All Rights Reserved.

71. Act

Experimente

O Act executa workflows do GitHub Actions localmente, substituindo o ciclo de commit-push-espera por um feedback mais rápido. Nós o mencionamos em nosso blip sobre GitHub Actions em 2021. Os agentes de programação dão ao Act uma nova relevância, porque o volume crescente de commits e pushes pode causar contenção de builds. O Act permite que agentes de programação executem um workflow localmente antes de fazer o push, trazendo a CI diretamente para o seu loop generativo e apoiando a responsabilidade de agentes no pipeline.

Nossas equipes constataram que o Act lida com a maior parte do desenvolvimento de workflows do GitHub Actions, embora não seja uma réplica perfeita do ambiente do GitHub Actions. Por exemplo, suas imagens padrão carecem de algumas ferramentas de runner e ele não oferece suporte a concorrência, permissões e segredos de ambiente. Combinado com verificações estáticas como o Zizmor, o Act ajuda os agentes de programação a manter o build funcionando quando eles alteram o próprio pipeline.

72. Agent Package Manager

Experimente

O Agent Package Manager é um gerenciador de dependências de código aberto para configuração de agentes de IA. Ele funciona por meio de um manifesto que declara plugins, habilidades, instruções e servidores MCP. Nós o utilizamos para publicar habilidades versionadas a partir de um repositório central, em vez de copiar e colar arquivos. Ele trata o contexto do agente como uma cadeia de suprimentos, instalando a partir de repositórios Git ou de um marketplace de plugins para agentes e escaneando por caracteres Unicode ocultos que poderiam sequestrar seu comportamento. Seus arquivos de lock registram hashes de conteúdo e procedência, exportando SBOM como CycloneDX e SPDX. A auditoria detecta drift, e um arquivo de política restringe fontes, escopos e primitivas, tornando um enterprise harness reproduzível e governado. Ainda é um projeto pré-1.0 impulsionado pela comunidade, embora patrocinado por uma grande empresa do setor, por isso o colocamos em "Experimente" em vez de "Adote" e esperamos que soluções concorrentes surjam no futuro.

73. ast-grep

Experimente

O ast-grep é uma ferramenta de linha de comando para busca estrutural, linting e reescrita de código. Os padrões são escritos como código comum com caracteres curinga para qualquer nó de sintaxe e correspondem à árvore sintática em vez de texto, usando parsers de tree-sitter para mais de 20 linguagens. As regras de linting e reescrita são escritas em um formato declarativo YAML. Uma de nossas equipes usou essas regras como parte do harness em torno de um agente de programação, verificando o código do agente em relação a um conjunto de regras personalizadas e enviando os resultados de volta para o agente.

O ast-grep é semelhante às ferramentas por trás de inteligência de código como ferramenta para agentes, mas ele não cria um índice nem exige um servidor de linguagem. Por pular a indexação, ele é um primeiro passo de baixo atrito para uma base de código grande antes de se comprometer com ferramentas mais pesadas. Ao contrário da maioria das ferramentas desse grupo, ele também tem usos que não envolvem agentes: as próprias pessoas desenvolvedoras podem executar as mesmas buscas, regras de lint e reescritas. O ast-grep abrange praticamente o mesmo escopo que o Comby, que destacamos em 2021. A maioria das IDEs já oferece busca baseada em AST, mas elas não substituem as regras de lint personalizadas e as reescritas que destacam o ast-grep.

© Thoughtworks, Inc. All Rights Reserved.

74. Beads

Experimente

Beads é um rastreador de problemas (issue tracker) baseado em Git que agentes de programação podem compartilhar para coordenar seu trabalho. Em vez de planos ad hoc em Markdown, ele fornece aos agentes um grafo de tarefas estruturado e persistente. Como o Beads armazena as tarefas e suas dependências junto com o código, os agentes mantêm seus planos e novas tarefas descobertas ao longo de sessões e compactações de contexto. Cada agente pode reivindicar a próxima tarefa disponível sem colidir com outros.

Estamos movendo o Beads para "Experimente" porque muitas de nossas pessoas desenvolvedoras agora o usam por padrão com agentes de programação. Ele complementa rastreadores de problemas de equipe, como o Linear, que armazenam o trabalho que uma equipe planeja, prioriza e relata. O Beads armazena as tarefas de granularidade mais fina que os agentes criam enquanto trabalham, e sincroniza em ambas as direções com o Linear, Jira e GitHub Issues.

Desde a última vez que o destacamos, o Beads atingiu a versão 1.0 e se mudou para a organização Gas Town Hall. Ele é mantido pela Gas City, a empresa por trás do Gas Town. O Dolt é agora seu único backend de armazenamento, e adições como uma API HTTP, contratos de alocação de trabalho (work leases) e sincronização entre armazenamentos tornam o Beads mais pesado do que o rastreador leve que era no início. As atualizações migram o esquema no local (in place), e clientes executando versões diferentes não podem compartilhar um armazenamento, portanto as equipes devem fixar uma única versão para cada armazenamento compartilhado e revisar as instruções que o Beads escreve no AGENTS.md.

75. Chaos Mesh

Experimente

O Chaos Mesh é um conjunto de ferramentas de código aberto de chaos engineering para injetar falhas controladas em sistemas nativos da nuvem (cloud-native). O Chaos Mesh define simulações de falha como recursos personalizados (custom resources) declarativos do Kubernetes, permitindo que as equipes usem ferramentas familiares e gerenciem experimentos de caos por meio do controle de versão. Achamos que sua variedade de cenários de falha simulados, reprodutibilidade e controles de raio de alcance (blast radius) o tornam adequado para testes repetíveis e automatizados em escala. Isso inclui ambientes com infraestrutura de IA densa em GPUs, onde validar a resiliência pode ser particularmente desafiador, porém importante.

76. Cube Core

Experimente

À medida que os dados analíticos são cada vez mais consumidos tanto por aplicações personalizadas quanto por agentes de IA, definir métricas diretamente em painéis (dashboards) ou em consultas SQL individuais frequentemente leva à duplicação e ao desvio de métricas (*metric drift*). O Cube Core é uma camada semântica de código aberto que aborda esse desafio, permitindo que as equipes definam métricas, dimensões, junções (joins) e regras de acesso como código. Ao criar um modelo semântico governado e reutilizável, as equipes podem definir métricas de negócios uma única vez e expô-las de forma consistente em ferramentas de BI, análises embarcadas e fluxos de trabalho de IA por meio de APIs padrão.

© Thoughtworks, Inc. All Rights Reserved.

Usamos o Cube Core com sucesso para implementar uma camada semântica em projetos recentes. A ferramenta conta com uma base de código ativa e uma comunidade vibrante de contribuidores. Se você está enfrentando problemas com definições de dados inconsistentes em várias camadas de consumo, recomendamos avaliar o Cube Core para padronizar o acesso analítico.

77. Figma Code Connect

Experimente

Ferramentas genéricas de design-to-code frequentemente transformam quadros (frames) em marcação (markup) bruta que ignora os componentes existentes de uma equipe, resultando em código que precisa ser reescrito. O Figma Code Connect aborda esse descompasso mapeando componentes do Figma diretamente para componentes de UI de produção no código. Em vez de adivinhar APIs de componentes, ferramentas de IA e pessoas desenvolvedoras podem usar esses mapeamentos para gerar código com as assinaturas e propriedades (props) reais dos seus componentes. Essa abordagem estruturada fundamenta a geração de código no sistema de design (design system) existente.

Esta ferramenta é particularmente relevante à medida que as equipes fazem a transição da geração de código baseada em prompts para fluxos de trabalho agênticos mais controlados. Ao combinar o Figma Code Connect com o servidor MCP do Figma e agentes de programação, pessoas desenvolvedoras podem fornecer contexto aos agentes a partir da base de código real. Isso ajuda a deslocar a geração de código em formato livre em direção a uma montagem de componentes mais determinística e em conformidade com a produção, reduzindo as iterações de revisão de código e o risco de desvio arquitetural (architectural drift).

Em nossa experiência com sistemas de design para dispositivos móveis, usamos esse fluxo de trabalho para mapear componentes entre iOS e Android, publicando automaticamente no Figma após lançamentos no branch principal e usando integração contínua (CI) para detectar desvios. No entanto, a geração de Figma para código é tão confiável quanto o sistema de design subjacente. Os resultados gerados ainda exigem guardrails, como verificações de lint para bloquear cores programadas diretamente (hardcoded) e baselines de snapshot em cada pull request. Manter mapeamentos para cada variante também exige esforço contínuo, e atualmente a ferramenta opera no nível de componente, em vez de validar telas completas do produto. Portanto, a lógica de layout, o estado e a navegação ainda exigem supervisão humana.

78. FluxMarkdown

Experimente

O FluxMarkdown é um plugin do Quick Look para macOS que renderiza arquivos Markdown diretamente no painel de pré-visualização do Finder. Embora existam outras opções, como o QLMarkdown, o FluxMarkdown oferece suporte abrangente para destaque de sintaxe, notações matemáticas via KaTeX e diagramas complexos usando Mermaid, Graphviz e Vega. Achamos o FluxMarkdown útil para inspecionar documentação porque ele elimina a necessidade de abrir um editor dedicado apenas para visualizar um diagrama ou equação renderizada. Conforme as pessoas desenvolvedoras dependem cada vez mais de interfaces de linha de comando e agentes autônomos em seus fluxos de trabalho, a visualização de arquivos Markdown renderizados ainda pode exigir a abertura de uma IDE. O FluxMarkdown oferece uma alternativa leve que permite às pessoas desenvolvedoras pré-visualizar esses arquivos sem sair do Finder.

© Thoughtworks, Inc. All Rights Reserved.

79. Git AI

Experimente

Git AI é uma extensão de código aberto do Git projetada para rastrear código gerado por IA em repositórios. Ela vincula linhas escritas por IA aos agentes, modelos e prompts que as geraram. Usando checkpoints e hooks, a ferramenta rastreia alterações incrementais de código ao longo do desenvolvimento. Cada checkpoint contém um diff entre o estado atual e o anterior, marcando as linhas como de autoria humana ou de IA. Essa abordagem oferece uma atribuição mais detalhada das contribuições de IA do que simplesmente contar linhas de código no momento em que são adicionadas.

A extensão utiliza um padrão aberto baseado no Git Notes para rastrear código gerado por IA. À medida que o código gerado por IA se torna mais comum no desenvolvimento do dia a dia, manter a responsabilidade e entender a intenção original por trás do código torna-se cada vez mais importante. Tanto pessoas desenvolvedoras quanto agentes de IA podem consultar sessões arquivadas para recuperar o contexto original e as decisões de design. Dado o seu valor potencial para responsabilidade e manutenibilidade em fluxos de trabalho agênticos, movemos o Git AI para "Experimente".

80. Linear

Experimente

O Linear é um rastreador de tarefas que combina a fácil configuração do Trello com os recursos de planejamento pelos quais as equipes costumam recorrer ao Jira. Muitas startups o utilizam no lugar do Jira há anos, e seu apelo recente vem de como ele funciona bem com agentes de programação. Agentes entram em um espaço de trabalho do Linear como membros, de modo que as equipes podem atribuir tarefas a eles, mencioná-los em comentários e ver o raciocínio por trás de cada alteração. O Cursor, o Codex e o Devin podem assumir trabalhos dessa forma, enquanto o servidor MCP remoto do Linear permite que ferramentas como o Claude Code leiam e atualizem tarefas. O Linear armazena o trabalho que uma equipe planeja e prioriza. O Beads fica um nível abaixo dele, rastreando as tarefas menores que os agentes criam à medida que avançam. Frequentemente usamos o Linear com clientes cujas equipes trabalham intensamente com agentes de programação. Equipes que escolhem suas próprias ferramentas devem considerar experimentá-lo se quiserem que os agentes peguem tarefas diretamente do backlog.

81. mattpocock/skills

Experimente

O mattpocock/skills ("Skills For Real Engineers") é uma coleção de skills de agente para engenharia de software agêntica. As habilidades fornecem um fluxo de trabalho sugerido, mas são explicitamente projetadas para serem pequenas, fáceis de adaptar e com alta capacidade de composição. Nós as achamos valiosas para ajudar equipes a aplicar boas práticas de engenharia de software à programação agêntica, bem como para análise auxiliada por IA de requisitos e premissas, como grilling de requisitos. Embora façamos um alerta contra a adoção de bibliotecas inteiras de habilidades sem avaliá-las (a rendição cognitiva a skills prontas), vale a pena dar uma olhada nessas habilidades leves se você deseja mais estrutura nos seus fluxos de trabalho de desenvolvimento de software agêntico.

© Thoughtworks, Inc. All Rights Reserved.

82. OpenSpec

Experimente

OpenSpec é um framework leve de desenvolvimento orientado por especificação que funciona com qualquer agente de programação. Desde que o destacamos na categoria "Avalie" do Tech Radar de abril, várias de nossas equipes relataram sucesso com ele, o suficiente para movê-lo para "Experimente". Apreciamos especialmente sua proposta para bases de código legadas (brownfield), adaptando especificações ao código existente. As equipes relatam resultados de agentes mais previsíveis e menos alterações não intencionais em comparação com o uso direto de prompts. O principal diferencial do OpenSpec em relação a concorrentes como GitHub Spec Kit e Kiro é focar em um fluxo de trabalho ancorado em especificações. As alterações são deltas em relação a uma especificação que serve como fonte da verdade persistente, de modo que os agentes podem confiar nela como uma referência atual. A cautela habitual se aplica: arquivos de especificação são markdown que os agentes geram prolificamente. O arquivamento de deltas de alterações do OpenSpec mitiga isso, mas manter a especificação persistente ainda exige disciplina.

83. OpenWiki

Experimente

O OpenWiki é uma interface de linha de comando que escreve e mantém automaticamente uma wiki para uma base de código ou repositório pessoal de conhecimento. Ele sintetiza materiais de origem em uma wiki em Markdown com links que outros agentes podem usar como contexto. O OpenWiki está atualmente em nosso anel "Experimente" enquanto avaliamos seu impacto em projetos ativos. Nossa experiência reforçou um desafio familiar: documentação de alta qualidade é essencial, mas mantê-la atualizada é notoriamente difícil. O OpenWiki aborda isso usando pipelines automatizadas para atualizar o conteúdo à medida que o material de origem muda, com checkpointing para rastrear quando cada trecho de documentação foi gerado pela última vez.

O OpenWiki também suporta modelos locais para geração privada e o formato OKF para documentação estruturada. Para humanos, inclui um visualizador interativo que apresenta a wiki subjacente como um grafo interativo, ajudando a navegar e entender as relações entre tópicos interconectados e documentos da base de código.

84. Pi (agente de programação)

Experimente

O Pi é um agent harness minimalista e de código aberto escrito em TypeScript. Este blip cobre todo o kit de ferramentas: a API LLM multiprovedor (pi-ai), o runtime do agente (pi-agent-core) e o agente de programação via terminal construído sobre eles (pi-coding-agent). O Pi intencionalmente deixa de fora MCP, subagentes, modo de planejamento e prompts de permissão. Você adiciona o que precisa por meio de extensões, habilidades do agente (agent skills) e templates de prompt. O Pi pode até escrever suas próprias extensões. Estamos movendo o Pi para "Experimente" porque a adoção cresceu desde o último Radar, principalmente como base para outras ferramentas. Scanners de segurança e harnesses de orquestração agora rodam no Pi, frequentemente como uma dependência subjacente em vez de uma ferramenta usada diretamente, enquanto seus modos SDK e RPC permitem embuti-lo em outros sistemas de agentes.

© Thoughtworks, Inc. All Rights Reserved.

Como um agente de programação usado diretamente, o Pi não é a escolha mais forte, e o OpenCode é mais adequado como um harness de programação empresarial. O Oh My Pi, um fork popular, adiciona integração com servidores de linguagem, subagentes e MCP para criar um agente de programação completo, mas divergiu o suficiente para ser avaliado como uma ferramenta separada. Nossa preocupação da última vez era a dependência do Pi de apenas um mantenedor; em abril de 2026, seu criador se juntou à Earendil, que agora é proprietária do Pi e se comprometeu a manter o núcleo sob licença MIT, enquanto planeja add-ons comerciais.

85. Sourcebot

Experimente

Assim como as pessoas desenvolvedoras precisam de um contexto amplo em bases de código empresariais grandes e multirrepositório, os agentes de engenharia de software precisam do mesmo contexto para realizar refatorações em grande escala e navegar por ecossistemas legados. O Sourcebot é uma plataforma auto-hospedada de busca e compreensão de código que atua como uma camada de contexto para atender a ambos. Ele indexa milhares de repositórios para oferecer busca rápida e baseada em símbolos, além de navegação em nível de IDE. Com o servidor MCP do Sourcebot, os agentes podem pesquisar em toda a base de código de uma organização. Nossas equipes também acharam a funcionalidade do plano pago, Ask Sourcebot, útil para a colaboração entre equipes. Essa ferramenta de chat com IA permite que funções de fora do desenvolvimento consultem a base de código usando linguagem natural, reduzindo a dependência de pessoas desenvolvedoras para responder a dúvidas sobre o código.

86. Timefold Solver

Experimente

O Timefold Solver é um mecanismo de código aberto de otimização de restrições para Java e Kotlin e o sucessor moderno do OptaPlanner. Ele resolve problemas de planejamento logístico, como roteamento de veículos, escala de funcionários, atribuição de tarefas e agendamento de manutenção, nos quais o espaço de possibilidades é grande demais para ser pesquisado de forma exaustiva. Embora comercializado sob o guarda-chuva mais amplo da IA, ele não envolve dados de treinamento nem modelos aprendidos, confiando em vez disso em uma busca heurística com base em restrições que você define. Ele modela regras de negócios absolutas como restrições rígidas (hard constraints) que um plano retornado não pode violar e metas operacionais como restrições flexíveis (soft constraints) que ele otimiza dentro desse espaço viável. Utilizamos o Timefold Solver com sucesso em produção no domínio de cadeia de suprimentos para modelar rapidamente cenários de negócios complexos.

87. AgentsView

Avalie

À medida que a programação assistida por IA se torna convencional, o AgentsView oferece às pessoas desenvolvedoras observabilidade sobre as sessões de programação para que elas possam analisar padrões e melhorar o uso de harnesses. O AgentsView permite que as pessoas desenvolvedoras naveguem, pesquisem e realizem perfilamento de contexto em sessões passadas de programação com IA. Ele pode se conectar automaticamente a harnesses de programação populares, como Claude Code, OpenAI Codex, OpenCode ou Pi e recuperar sessões históricas. Ele inclui ferramentas de análise de sessões de programação para métricas como prompt-cache e eficiência de custos, além de recomendações para aprimorar o uso de harnesses. Algo que gostamos no AgentsView é seu

© Thoughtworks, Inc. All Rights Reserved.

design local-first, o que torna a adoção simples até mesmo para organizações preocupadas com a segurança, enquanto a sincronização entre equipes e as revisões colaborativas de sessões podem ser adicionadas posteriormente.

88. Apple container

Avalie

O Apple container é uma ferramenta de código aberto para executar contêineres Linux nativamente em Apple silicon. Escrito em Swift, ele usa o pacote Containerization da Apple para executar uma VM leve para cada contêiner em vez de uma VM Linux monolítica. Essa arquitetura oferece isolamento a nível de hipervisor e alocação dinâmica de memória, devolvendo a memória RAM ao macOS quando um contêiner é encerrado. Ele oferece uma alternativa leve e nativa a gerenciadores de VM compartilhada, como o Colima, e evita os requisitos de licenciamento associados ao Docker Desktop. Quando combinado com pontes da comunidade (community bridges) como o Socktainer para compatibilidade com a API do Docker, o Apple container é particularmente eficaz para configurações locais de Testcontainers que exigem inicializações frias rápidas e limites rígidos de recursos. O projeto também inclui o Container Machine para executar ambientes Linux persistentes e integrados ao host.

89. Deepsec

Avalie

O Deepsec é um scanner de vulnerabilidades impulsionado por agentes que primeiro analisa uma base de código para entender sua arquitetura e, em seguida, lança revisões de IA direcionadas a áreas de alto risco para produzir relatórios de vulnerabilidade. O que se destaca é sua arquitetura híbrida: ele combina correspondência determinística para identificar arquivos de alto risco com subagentes dedicados que os inspecionam. Ao estreitar o foco do modelo para onde é mais importante, o Deepsec pode reduzir o consumo de tokens e os custos em comparação com ferramentas semelhantes.

90. DeepSeek Harness (DSH)

Avalie

O DeepSeek Harness (DSH) é um runtime de código aberto desenvolvido pela DeepSeek AI para a construção de agentes personalizados de programação e de uso geral. Em vez de oferecer um assistente monolítico, o DSH utiliza uma arquitetura modular em que “tudo é um plugin”, impulsionada pelo Cordis kernel. O adaptador de modelo, o registro de ferramentas, o log de sessão e o loop do agente são todos plugins, organizados em perfis: listas nomeadas de plugins que a equipe estende e sobrescreve em arquivos YAML próprios.

O Cordis é um meta framework construído em torno da composabilidade espaço-temporal: os plugins registram suas operações como efeitos reversíveis. Por isso, o recarregamento (reload) e a desmontagem (teardown) os revertem de forma previsível, sem deixar estados órfãos no sistema.

A junção dessa arquitetura de plugins com a composabilidade espaço-temporal estabelece uma base sólida para criar enterprise harnesses específicos de domínio e viabilizar o autoaperfeiçoamento de agentes. O DSH também define a observabilidade como uma preocupação de primeira classe em tempo de execução (runtime). Todas as etapas processadas pelo modelo, incluindo prompts, injeções de contexto, acionamentos de ferramentas e colaborações entre subagentes, são salvas em um log de sessão do tipo append-only (somente anexação). Isso permite que as execuções sejam inspecionadas, retomadas, ramificadas (fork), pesquisadas e reproduzidas (replay), fornecendo um registro auditável do que o sistema recomposto realmente executou.

© Thoughtworks, Inc. All Rights Reserved.

91. Entire CLI

Avalie

O Entire CLI conecta-se via hooks ao Git para capturar sessões de agentes de programação, incluindo prompts, transcrições, chamadas de ferramentas e arquivos modificados, armazenando esses dados diretamente no repositório, junto aos commits gerados, mas fora do histórico do branch. A ferramenta oferece suporte a Claude Code, Codex, Cursor, OpenCode, Pi e outros. Desde a última vez em que incluímos o Entire CLI no Radar, uma de nossas equipes internas que trabalha intensamente com agentes de programação utilizou a ferramenta para compartilhar as sessões entre todos os seus membros. A equipe considerou o processo útil para observar como as pessoas estruturam seus comandos (prompts), como os agentes processam essas entradas e de que forma essa interação afeta a qualidade do resultado final. Isso também pode ajudar pessoas desenvolvedoras experientes a demonstrarem como trabalham com agentes.

Essa transparência tem um custo. O Entire CLI mantém um registro permanente das interações, acessível a qualquer pessoa com permissão no repositório. Ao tentar introduzir a ferramenta em outras equipes, houve resistência de pessoas desenvolvedoras que enxergam o agente de programação como um assistente pessoal e não se sentiram confortáveis em expor todo o conteúdo das conversas. Para esses cenários, o AgentsView funciona como alternativa local, permitindo que as pessoas desenvolvedoras analisem suas próprias sessões sem compartilhamento externo. Recomendamos experimentar o Entire CLI em equipes colaborativas e com alta cultura de confiança mútua. A ferramenta permanece em "Avalie" porque a validação ocorreu apenas em uma equipe interna, e ainda não temos dados sobre como outras equipes podem adotá-la com tranquilidade.

92. FreeToken

Avalie

O FreeToken é um motor de inferência de código aberto projetado para executar modelos open-weight grandes baseados em Mixture-of-Experts (MoE) em hardware de consumo comum. Conforme esses modelos se tornam mais capazes, o tamanho total de seus pesos pode exceder os limites de memória das GPUs domésticas. Diferente de executores de modelos locais de uso geral, o FreeToken é especializado em modelos MoE sobredimensionados (oversized). Ele combina, de forma adaptativa, a memória da GPU, o processamento na CPU, a memória do host e a largura de banda do barramento PCIe. O sistema faz cache dinâmico dos especialistas ativos na GPU e os alterna com base em quais especialistas são ativados. Isso permite que as pessoas desenvolvedoras executem modelos significativamente maiores em hardwares existentes ou mais antigos. Já estamos começando a ver técnicas de offloading de MoE incorporadas a ferramentas mais amplas, como o llama.cpp (usado pelo Ollama), mas o FreeToken oferece uma abordagem especializada que vale a pena avaliar.

93. GLM

Avalie

À medida que a diferença de capacidade entre modelos de IA abertos e fechados diminui, a família de modelos GLM da Z.ai se destaca pelo seu desempenho em cargas de trabalho de engenharia de software. Utilizando uma arquitetura Mixture-of-Experts (MoE) de aproximadamente 700 bilhões de parâmetros, modelos como o GLM-5.3 alcançam resultados sólidos em benchmarks de programação, raciocínio e tarefas de agentes, impulsionados principalmente por um pós-treinamento especializado em vez de apenas pela escala de pré-treinamento. Isso torna o GLM-5.3 um dos melhores modelos de pesos abertos disponíveis atualmente para programação e engenharia de software.

© Thoughtworks, Inc. All Rights Reserved.

Gostamos do GLM porque ele fornece às equipes uma alternativa open-weight que compete com modelos de ponta proprietários e tarefas de raciocínio complexas. No entanto, em nossa experiência, o modelo às vezes pode "pensar" demais, analisando excessivamente prompts simples. Aconselhamos as equipes a configurar o esforço de raciocínio com base na complexidade da carga de trabalho.

94. Habit Hooks

Avalie

O Habit Hooks é uma ferramenta open-source de coaching de qualidade de código para agentes de programação por IA. Ele executa ferramentas existentes de análise estática e linting, como Ruff, ESLint, Knip, PMD, deptry e jscpd. Em vez de retornar erros brutos de linter, ele traduz violações de regras específicas das ferramentas em orientações proativas de design e contexto de code smells. Uma violação simples de linter informa ao agente de IA que uma verificação falhou, mas pode incentivar uma alteração superficial que apenas satisfaz a regra. O Habit Hooks mapeia o problema apontado ao respectivo code smell e injeta de forma determinística orientações explícitas no prompt do modelo. Esse contexto ajuda o agente a abordar o problema real de design e se autocorrigir imediatamente sem depender de um LLM para gerar a recomendação. A ferramenta oferece uma implementação prática de sensores de feedback para agentes de programação, integrando verificações determinísticas no loop de implementação para que os agentes possam responder a falhas antes da revisão humana.

95. Herdr

Avalie

O Herdr é um multiplexador de terminal e runtime de código aberto projetado para gerenciar múltiplas sessões de agentes de programação. Ele oferece recursos familiares de multiplexadores, como painéis, navegação por teclado, desanexação e reanexação, facilitando a migração das pessoas desenvolvedoras a partir de ferramentas como o tmux. O Herdr mantém os agentes em execução em sessões de terminal persistentes enquanto expõe estados do agente, como trabalhando (working), bloqueado (blocked), concluído (done) e ocioso (idle). Integrações via CLI permitem que os agentes criem painéis, troquem prompts e coordenem o trabalho.

À medida que as pessoas desenvolvedoras executam cada vez mais agentes de programação paralelos em múltiplos espaços de trabalho, acompanhar quais sessões exigem atenção e como retomar o trabalho interrompido tornou-se um novo desafio de fluxo de trabalho. Ferramentas como T3 Code, Orca e Paseo oferecem experiências alternativas para pessoas desenvolvedoras que preferem uma interface gráfica (GUI). Equipes que gerenciam agentes paralelos devem avaliar o Herdr se quiserem sessões persistentes e status visível dos agentes em seu fluxo de trabalho de terminal.

96. Hunk

Avalie

Hunk é um visualizador de diff de código aberto no terminal para revisar alterações feitas por agentes de programação. Ele exibe um pull request do GitHub ou um conjunto de alterações do Git, Jujutsu ou Sapling como uma revisão de múltiplos arquivos, com uma barra lateral de arquivos e layouts divididos ou unificados. Um modo de observação (watch mode) atualiza a visualização à medida que um agente edita a árvore de trabalho (working tree). Um agente pode carregar a habilidade de agente do Hunk para conduzir uma sessão de revisão ao vivo, saltando para um arquivo ou trecho (hunk) e adicionando notas ao lado das linhas que alterou. As pessoas revisoras podem, então, ler o raciocínio do agente ao lado do código relevante. Essas notas diferenciam o Hunk de ferramentas de diff com reconhecimento de sintaxe, como o Difftastic.

© Thoughtworks, Inc. All Rights Reserved.

Não temos certeza se a revisão humana linha por linha dos resultados de agentes continuará sendo a norma, mas muitas equipes ainda precisam dela hoje: economizar nessa revisão pode levar à complacência com código gerado por IA. O projeto começou em março de 2026 e não atingiu a versão 1.0. Equipes que revisam alterações de agentes no terminal devem experimentar o Hunk e avaliar se as notas do agente tornam as revisões mais rápidas ou se simplesmente dão às pessoas revisoras mais conteúdo para ler.

97. nono.sh

Avalie

O nono.sh é um shell voltado à segurança de código aberto e sandbox leve projetado para executar agentes de IA sem a sobrecarga das máquinas virtuais tradicionais ou contêineres Docker. Ele aborda uma lacuna comum: embora o uso de ambientes isolados (sandboxing) seja amplamente recomendado, a complexidade e custo de configuração pode desestimular as equipes a implementá-lo. O nono.sh isola processos de agentes diretamente no nível do sistema operacional usando mecanismos nativos do kernel, como Landlock no Linux e Seatbelt no macOS. Ele opera in-process como processos nativos do host sem daemons em segundo plano, bloqueando por padrão o acesso a recursos sensíveis do host, como chaves SSH.

Acreditamos que vale a pena avaliar o nono.sh porque ele reduz a barreira para o isolamento de processos. Seu design minimalista facilita a configuração, o teste e a integração em fluxos de trabalho existentes no macOS, Linux e Windows via WSL2. Perfis de segurança podem ser ajustados, criados a partir de versões existentes (forks) e compartilhados entre equipes ou com a comunidade por meio do registro de perfis do nono, oferecendo uma camada de defesa de baixa sobrecarga para pessoas desenvolvedoras que estejam adotando ferramentas de execução autônomas como o Claude Code ou o OpenAI Codex.

98. OpenResearch CLI (orx)

Avalie

O OpenResearch CLI (orx) é um harness de código aberto que transforma agentes de programação, como o Claude Code, em agentes de pesquisa capazes de executar experimentos em paralelo. Cada experimento fica isolado em seu próprio worktree do Git, garantindo que as execuções não interfiram entre si. Todos os experimentos usam o mesmo comando de execução, permitindo a comparação direta dos resultados. Além disso, cada execução é vinculada a um snapshot imutável do código, o que assegura a reprodutibilidade.

O orx pode ser executado na infraestrutura de computação que a equipe escolher (local, Slurm, Kubernetes ou Modal). As equipes podem implementar um método descrito em um artigo acadêmico, reproduzir seus experimentos e testar se os resultados se mantêm no contexto corporativo. Isso ajuda a identificar a lacuna entre a teoria acadêmica e a prática corporativa. Acreditamos que vale a pena avaliar o orx como um harness promissor e específico de domínio para diminuir essa lacuna.

99. Safe Chain

Avalie

O Safe Chain é uma ferramenta gratuita da Aikido que atua como um wrapper para os gerenciadores de pacotes de Node.js e Python, como npm, pnpm, Bun, pip e uv. Ele encaminha os downloads de pacotes por meio de um proxy local que bloqueia versões publicadas nas últimas 48 horas e qualquer pacote listado como malicioso na base aberta de ameaças da Aikido. Worms como o Shai-Hulud

© Thoughtworks, Inc. All Rights Reserved.

mostraram a rapidez com que um pacote comprometido pode se propagar. Um atraso controlado dá tempo para que os scanners detectem versões maliciosas. A maioria dos gerenciadores de pacotes atuais possui configuração nativa de idade mínima de publicação, e as equipes devem ativá-la em seus projetos. O Safe Chain cobre o que essas configurações deixam passar, como um comando npx pontual executado fora de qualquer projeto ou pacotes maliciosos mais antigos do que o limite de idade definido.

O Safe Chain abrange apenas dois ecossistemas, mas ambos são alvos recorrentes de ataques à cadeia de suprimentos (supply chain attacks). Uma versão retida pela restrição de idade gera apenas um aviso genérico de supressão de pacotes. Equipes que baixam dependências por meio de espelhos (mirrors) privados com políticas próprias de tempo têm pouco benefício prático, e o bloqueio de ameaças depende exclusivamente do catálogo de um único fornecedor. Além disso, o Safe Chain é distribuído sob licença AGPL ou comercial, o que exige revisão jurídica prévia antes de ser embutido em qualquer solução distribuída comercialmente.

100. Swamp

Avalie

O Swamp é uma ferramenta de automação construída para ser usada por agentes de programação baseados em IA. Quando um agente realiza uma tarefa operacional diretamente, ele pode executá-la de forma diferente a cada vez e deixar pouca rastreabilidade sobre o que fez. Com o Swamp, em vez disso, o agente escreve a automação na forma de modelos TypeScript de sistemas externos e fluxos de trabalho YAML que encadeiam os métodos desses modelos. Os fluxos de trabalho seguem o mesmo processo definido todas as vezes, suas definições residem em um repositório Git e a saída de cada execução é mantida como dados versionados que o agente pode consultar posteriormente. As credenciais são injetadas a partir de um cofre de segredos no momento da execução, garantindo que permaneçam fora dos prompts. Os modelos também podem ser empacotados como extensões e compartilhados por meio de um registro de módulos. Um agente instruído a escrever e fazer commit de scripts sem o Swamp pode produzir algo semelhante, mas sem interfaces tipadas integradas, histórico de execuções ou módulos reutilizáveis. O Swamp substitui o System Initiative, que destacamos em “Avalie” em uma edição anterior. O Swamp surgiu do gerenciamento de infraestrutura, mas é voltado para fluxos de trabalho operacionais de forma mais ampla, desde a resposta a incidentes até pipelines de dados. O código-fonte é aberto sob a licença AGPLv3, embora a distribuição oficial utilize uma licença comercial que exige uma assinatura para uso no trabalho. O projeto é novo e evolui rapidamente, com uma nova versão lançada a cada commit, portanto as equipes devem testá-lo em automações internas de baixo risco antes de depender dele.

101. Vale

Avalie

O Vale é um linter para textos. Ele analisa Markdown, AsciiDoc e reStructuredText, ignora blocos de código e verifica o texto em relação a regras de estilo armazenadas como YAML no repositório, permitindo que as convenções de escrita sejam aplicadas no editor e no CI, de forma semelhante ao estilo de código. Os estilos inclusos abrangem os guias de documentação da Google, Microsoft e Red Hat; outros verificam a linguagem inclusiva ou a legibilidade, e você também pode criar os seus próprios. O Vale faz parte de toolchains de docs-as-code há anos, mas o volume de documentações, ADRs e descrições de PRs rascunhadas por LLMs o torna ainda mais relevante hoje. Pacotes como o vail-ai-tells sinalizam o fraseado típico de textos gerados, e o Vale funciona bem como um dos sensores de feedback para agentes de programação, permitindo que um agente revise um rascunho antes que qualquer pessoa o leia.

© Thoughtworks, Inc. All Rights Reserved.

As verificações do Vale são correspondências determinísticas de palavras e padrões, capazes de identificar vícios superficiais, mas que podem deixar passar raciocínios fracos ou detalhes inventados. O Vale também depende em grande parte de um único mantenedor, e nosso uso até agora tem sido feito por indivíduos trabalhando em seus próprios textos, e não por equipes de clientes.

102. Varlock

Avalie

O Varlock é uma camada de configuração de ambiente e gerenciamento de segredos de código aberto projetada para evitar a exposição de segredos a agentes de programação via texto simples (plaintext). Em vez de confiar em convenções informais do tipo `.env.example`, o Varlock introduz um arquivo `.env.schema` versionado que descreve nomes de variáveis, tipos, regras de validação e documentação. Ele separa o contrato de configuração dos valores reais. Os segredos são resolvidos e injetados em tempo de execução a partir de cofres externos, como 1Password, AWS Secrets Manager ou Azure Key Vault. Para cargas de trabalho nas quais um agente deve chamar um serviço autenticado, o Varlock também oferece um proxy de credenciais que fornece um placeholder para o agente e substitui pelo valor real apenas no limite da rede. Achamos esse um padrão útil que substitui scripts auxiliares ad hoc e estende a injeção de segredos em múltiplos ambientes sem expor credenciais.

103. zizmor

Avalie

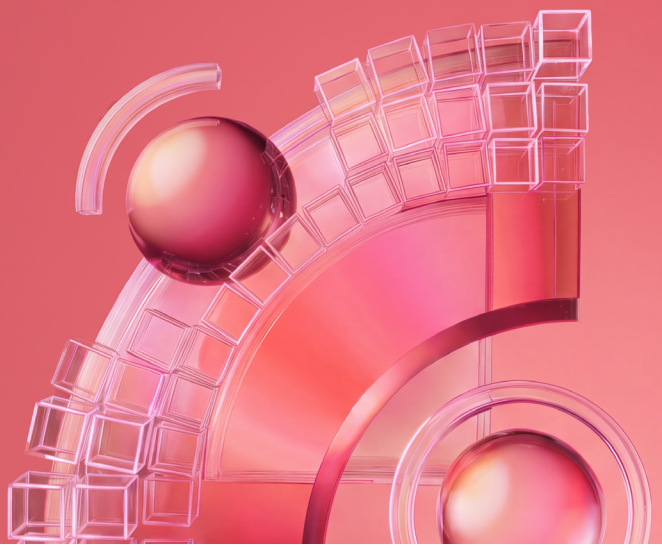
O zizmor é uma ferramenta de análise estática para fluxos de trabalho do GitHub Actions que também verifica configurações do Dependabot e do pre-commit. Ele sinaliza falhas de configuração que atacantes podem explorar, incluindo injeção de template, permissões excessivamente amplas e ações fixadas em tags mutáveis, além do uso de gatilhos arriscados como o `pull_request_target`. Ele também pode corrigir alguns dos problemas identificados automaticamente.

Este ano, atacantes exploraram um workflow de `[pull_request_target]` no repositório do Trivy e, em seguida, reescreveram as tags de sua GitHub Action para que as pipelines que referenciavam essas tags executassem o código dos atacantes. Nossas equipes usaram o zizmor em projetos de clientes, executando-o a partir da linha de comando e como um hook de pré-commit enquanto escreviam workflows. Agentes de programação também podem executá-lo, fornecendo feedback rápido sobre as pipelines que escrevem.

Em meio à atenção voltada para a varredura de segurança impulsionada por IA, gostamos de ter uma ferramenta determinística para esse trabalho. O Snyk e o Wiz cobrem parte do mesmo terreno comercialmente e linters genéricos detectam alguns desses problemas, mas o zizmor é gratuito e de código aberto. Sua desvantagem é o escopo: ele não cobre nenhuma outra plataforma de CI além do GitHub Actions. Equipes que usam o GitHub Actions devem avaliar o zizmor juntamente com a confiança zero para pipelines de CI/CD.

© Thoughtworks, Inc. All Rights Reserved.

Linguagens e Frameworks



Adote

104. Open Data Contract Standard (ODCS)

Experimente

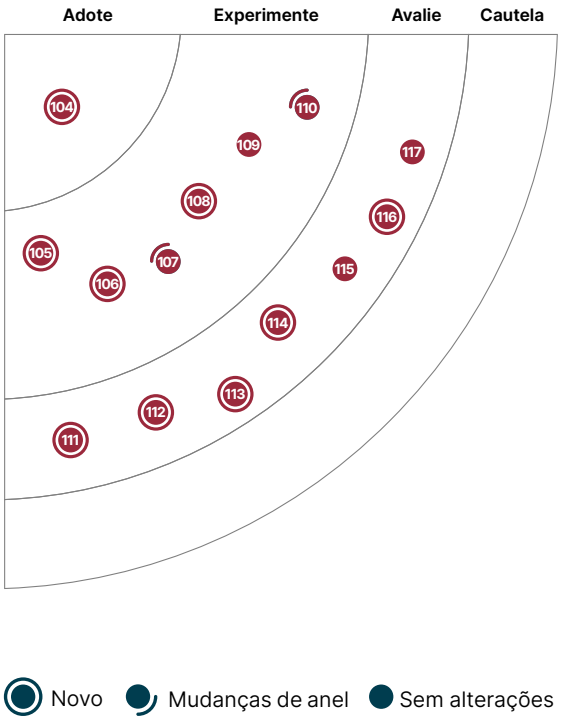
105. BoTorch
106. Deep Agents
107. Kagent
108. OpenFGA
109. Presidio
110. Promptfoo

Avalie

111. Apple Core AI
112. CALM
113. Hindsight
114. Quint
115. SolidJS
116. spartan/ui
117. TLA+

Cautela

—



104. Open Data Contract Standard (ODCS)

Adote

O Open Data Contract Standard (ODCS) define contratos de dados de forma declarativa em YAML, cobrindo o esquema, as expectativas de qualidade e os níveis de serviço de um produto de dados. Ele é desenvolvido pelo projeto Bitol da Linux Foundation, que se graduou em julho de 2026. Embora existam vários padrões concorrentes e formatos proprietários, o ODCS se destacou como um padrão líder, com os mantenedores do Data Contract CLI descontinuando o DCS em favor do ODCS.

Vários dos nossos clientes que estão construindo data meshes e produtos de dados estão migrando para o ODCS, frequentemente junto com o Open Data Product Standard (ODPS). O interesse em contratos de dados também está crescendo, em parte porque seu formato declarativo os torna mais fáceis de interpretar, revisar e validar tanto por seres humanos quanto por agentes de IA. Estamos posicionando o ODCS em "Adote" como um padrão sensato para equipes que constroem produtos de dados.

105. BoTorch

Experimente

O BoTorch é um framework de código aberto baseado em PyTorch para otimização bayesiana. Embora a otimização bayesiana seja frequentemente usada para ajuste de hiperparâmetros (hyperparameter tuning), estamos observando sua aplicação crescente em problemas reais de ciência e engenharia. Quando as ferramentas padrão para esse desafio não são suficientes, o BoTorch oferece a modularidade e o controle de baixo nível necessários para construir modelos probabilísticos e funções de aquisição personalizadas.

Combinado com o Ax para orquestração de experimentos, o BoTorch se destaca em domínios como formulação de produtos e otimização de P&D, onde avaliações físicas ou experimentais são dispendiosas e uma busca exaustiva é inviável. Nesses espaços de busca complexos, o BoTorch ajuda as equipes a navegar com eficiência pelos trade-offs e a maximizar um orçamento de avaliação limitado.

106. Langchain Deep Agents

Experimente

O Deep Agents é um harness de agentes de código aberto da LangChain, desenvolvido para tarefas de longa duração. O Deep Agents fornece uma arquitetura opinativa para decompor problemas, delegar trabalho a subagentes e gerenciar a execução. Ele inclui recursos essenciais de harness, como ferramentas de planejamento, skills de agentes, memória, pontos de checagem com intervenção humana (human-in-the-loop) e acesso ao sistema de arquivos.

Disponível como bibliotecas Python e TypeScript, o Deep Agents se integra bem com a LangChain e o LangGraph. Ele é particularmente útil quando etapas específicas num grafo de agentes complexo exigem um agente capaz e de longa duração. Em nossa experiência, ele complementa casos de uso de pesquisa multiagente e modernização de legado, integrando-se ao ecossistema mais amplo da LangChain, incluindo ferramentas de observabilidade como o Langfuse.

© Thoughtworks, Inc. All Rights Reserved.

107. Kagent

Experimente

O Kagent é um framework de programação de código aberto e uma plataforma nativa do Kubernetes para criar, executar e gerenciar agentes de IA. Em vez de exigir uma infraestrutura sob medida, o framework transforma os agentes em definições de recursos personalizados (CRDs) padrão do Kubernetes que as equipes podem implantar, escalar e proteger usando suas pipelines de GitOps existentes. Isso permite que equipes com experiência em Kubernetes criem agentes personalizados sem precisar aprender um novo modelo operacional. O Kagent oferece um runtime integrado com rastreamento via OpenTelemetry, sandboxing seguro, suporte a memória e integrações nativas com o Model Context Protocol (MCP) para conectar facilmente os agentes a ferramentas externas e provedores de modelos.

Movemos o Kagent para "Experimente" após usá-lo com sucesso no suporte a ambientes de produção. Em um cliente recente, migramos um bot de suporte multiagente para o Kagent a partir de uma ferramenta anterior que restringia conexões a um único servidor MCP e exigia hospedagem em uma conta pessoal. Como a equipe do cliente já estava familiarizada com o Kubernetes, a migração foi simples. A implantação resultante conta com um orquestrador e múltiplos agentes especialistas que usam o A2A (Agent-to-Agent) Protocol integrado do Kagent para se comunicar, ajudando as pessoas desenvolvedoras a depurar incidentes e responder a dúvidas gerais diretamente nos canais da equipe.

108. OpenFGA

Experimente

O OpenFGA é um sistema de autorização de granulação fina (fine-grained authorization) e código aberto, inspirado no Zanzibar do Google. Ele modela relacionamentos entre usuários e recursos para determinar permissões, permitindo que as aplicações respondam a perguntas como "Este usuário pode acessar este recurso?". Ao contrário do controle de acesso baseado em funções (RBAC) tradicional, o OpenFGA suporta permissões complexas e específicas de recursos, bem como relacionamentos hierárquicos. Ele centraliza a lógica de autorização, melhorando a mantenabilidade, consistência, testes e auditoria em serviços distribuídos. Usamos o OpenFGA com sucesso em produção para controle de acesso avançado baseado em relacionamentos (ReBAC), implantando-o com uma estratégia de sidecar para permitir verificações de permissão com baixa latência. É uma escolha atraente quando o RBAC padrão se torna insuficiente.

109. Presidio

Experimente

Presidio é um SDK de proteção de dados para identificar e anonimizar dados sensíveis em texto estruturado e não estruturado. Ele detecta informações de identificação pessoal (PII), como números de cartão de crédito, nomes e locais, usando reconhecimento de entidade nomeada, expressões regulares e lógica baseada em regras. O Presidio suporta reconhecedores de entidade e pipelines de desidentificação customizados, permitindo que as organizações o adaptem às suas necessidades de privacidade e conformidade. Nossos times têm usado o Presidio em ambientes empresariais com controles rigorosos de compartilhamento de dados ao integrar com LLMs. Embora ele automatize a detecção de informações sensíveis, não é infalível e pode omitir ou identificar entidades incorretamente. Os times devem ter cautela ao confiar em seus resultados.

© Thoughtworks, Inc. All Rights Reserved.

110. Promptfoo

Experimente

Colocamos o Promptfoo em "Avalie" em 2023 para testar prompts. Ainda o usamos para avaliações (evals), incluindo chamadas de ferramentas por agentes e portões de CI, mas agora também para red teaming. Recomendamos testá-lo para ambos. Para fazer red teaming em uma aplicação, você descreve para o que ela serve, e o Promptfoo gera ataques para os riscos que você escolher, aplica estratégias de ataque e avalia as respostas. Usamos o Promptfoo para fazer red teaming em um assistente conversacional em produção com regras de segurança específicas de domínio. Ele transformou sondagens manuais ad hoc em testes repetíveis executados novamente após cada correção. Ele também funciona com guardrails com estado: os guardrails são o controle em tempo de execução, e o Promptfoo testa se eles se mantêm firmes. O Promptfoo agora pertence à OpenAI, embora continue sob licença MIT e seja agnóstico quanto ao modelo. Alguns dos ataques simulados mais avançados do Promptfoo dependem de seu serviço hospedado, portanto certifique-se de estar confortável com quais dados saem da sua organização.

111. Apple Core AI

Avalie

O Apple Core AI é um framework lançado recentemente para rodar modelos de IA e machine learning direto no dispositivo usando Apple Silicon. Com suporte nativo a Swift para desenvolvimento mobile, ele chega como o sucessor moderno do antigo framework da Apple, o Core ML. O Apple Core AI permite que pessoas desenvolvedoras integrem recursos de IA de forma nativa em seus apps iOS, aproveitando o processamento no próprio dispositivo em vez de depender de serviços externos.

Um destaque importante desse ecossistema é o framework Evaluations no Xcode, que estende o Swift Testing para avaliar recursos de IA generativa criados com os Foundation Models da Apple. Avaliar o output não determinístico de LLMs historicamente exigia scripts customizados no backend ou ferramentas de avaliação de terceiros. Agora, os times conseguem rodar datasets sintéticos, medir métricas quantitativas direto no código e aplicar pontuação via model-as-a-judge para critérios qualitativos, como tom e relevância, tudo dentro do fluxo de trabalho nativo de dev.

112. CALM

Avalie

O Common Architecture Language Model, ou CALM, é uma especificação open-source da FINOS para definir arquiteturas de software em um formato padronizado e legível por máquina. Construído sobre um meta-schema JSON extensível, ele trata a arquitetura como código, em vez de depender de diagramas estáticos e documentações que acabam ficando desalinhados da implementação.

Em grandes empresas, garantir que os sistemas sigam os padrões de arquitetura estabelecidos pode ser um desafio. O CALM permite que os times definam e gerenciem a arquitetura de forma descentralizada, mantendo uma visão unificada quando necessário. Suas descrições estruturadas também ajudam na validação do código em relação à arquitetura planejada. Embora ferramentas como o Mermaid permitam criar diagramas como código, o CALM oferece uma base legível por máquina para conformidade arquitetônica e governança programática.

© Thoughtworks, Inc. All Rights Reserved.

113. Hindsight

Avalie

O Hindsight é o sistema de memória de código aberto da Vectorize para agentes de IA. Um LLM extrai fatos, entidades e timestamps de conversas e os armazena no Postgres. A recuperação (recall) combina busca semântica, por palavras-chave, baseada em grafos e no tempo, e reordena os resultados. Uma operação de reflexão (reflect) consolida memórias armazenadas em observações e "modelos mentais" que evoluem à medida que novas evidências chegam. Essa consolidação o diferencia do Mem0, que extrai e armazena os fatos que julga mais relevantes. O Hindsight mantém um banco de memória separado para cada usuário ou agente e pode ser auto-hospedado ou usado por meio do serviço gerenciado Hindsight Cloud. Os agentes se conectam a ele por meio de SDKs, integração com frameworks ou um servidor MCP.

O Hindsight é particularmente útil para agentes customizados que precisam de memória entre sessões. Uma de nossas equipes o utilizou em um projeto de hackathon para fornecer aos agentes memória por usuário, que melhora ao longo de sessões repetidas. O Hindsight também possui plugins para agentes de programação como o Claude Code, embora esses agentes já tenham seu próprio mecanismo para reter memória entre sessões. O fornecedor relata pontuações de liderança no benchmark LongMemEval, que testa a recuperação ao longo de históricos de chat extensos. Equipes que constroem agentes que precisam de memória de longo prazo para cada usuário devem comparar o Hindsight com o Mem0 e mensurar se suas capacidades de recuperação e consolidação melhoram o comportamento do agente o suficiente para justificar as chamadas adicionais a LLMs e infraestrutura.

114. Quint

Avalie

O Quint é uma linguagem de especificação formal baseada no cálculo do TLA+, a lógica temporal de ações. Assim como o TLA+, ele foi projetado para ajudar a encontrar bugs sutis em sistemas complexos que envolvem concorrência. O diferencial do Quint em relação ao TLA+ está na sua legibilidade e na experiência da pessoa desenvolvedora. Ele possui uma sintaxe parecida com a do TypeScript, o que facilita a revisão de código, especialmente quando as especificações são geradas por agentes. Ele também vem com type checker, REPL e simulador, e pode ser instalado via npm. O simulador testa o modelo de forma leve usando execuções aleatórias e também consegue gerar traces a partir da especificação para testar a implementação real. Nossos times acharam a escolha entre Quint e TLA+ bem acirrada, ponderando o ecossistema de ferramentas do Quint contra a maturidade do TLA+. Esperamos que o seu desenvolvimento contínuo sinalize um ganho de tração para a verificação poliglota.

115. SolidJS

Avalie

O SolidJS é uma biblioteca JavaScript declarativa para a criação de interfaces de usuário que continua se destacando por seu alto desempenho e reatividade granular. Atualizações recentes trazem recursos nativos importantes para a comunicação assíncrona cliente-servidor. Ao lidar com essas demandas de forma nativa, o SolidJS elimina a necessidade de bibliotecas externas para gerenciar chamadas de API, cache e desduplicação de requisições. Esses recursos melhoram a experiência de desenvolvimento ao resolver desafios assíncronos comuns, como condições de corrida e estados

© Thoughtworks, Inc. All Rights Reserved.

de carregamento, diretamente dentro do framework. O SolidJS também oferece controle detalhado sobre suspense boundaries, permitindo que as pessoas desenvolvedoras sequenciem como os componentes assíncronos são exibidos, em vez de deixar sua ordem dependente da disponibilidade da rede. Combinado a um ecossistema de ferramentas em expansão que inclui o framework SolidStart, o SolidJS é uma alternativa atraente que vale a pena explorar para a construção de aplicações web de alto desempenho.

116. spartan/ui

Avalie

O spartan/ui traz a abordagem do shadcn para o Angular. Você instala primitivos acessíveis e sem estilo construídos sobre o Angular CDK via npm e os combina com componentes estilizados que ficam no seu próprio código-fonte. Essa divisão é central para a arquitetura: acessibilidade, comportamento e estado são mantidos por bibliotecas upstream, enquanto você controla a estilização.

É uma alternativa viável ao Angular Material. Uma de nossas equipes o utilizou como base para uma biblioteca de componentes corporativa em diversas aplicações, evitando a necessidade de construir um design system do zero e reduzindo o esforço necessário para criar e manter componentes compartilhados. O spartan/ui atingiu a versão 1.0 em junho de 2026, mas ainda é relativamente recente. Encontramos algumas imperfeições ao usar a CLI para atualizar os componentes estilizados em nosso código-fonte, por isso o mantemos em “Avalie”.

117. TLA+

Avalie

O TLA+ é uma linguagem de especificação formal para identificar falhas de design em sistemas distribuídos. Quando publicamos sobre ela pela última vez em 2015, o investimento era alto demais para quase tudo, exceto sistemas críticos. Desde então, agentes de programação mudaram esse cálculo: o custo de usar TLA+ caiu, enquanto seu valor aumentou. A experiência de desenvolvimento desconfortável do TLA+ não é mais um impedimento, embora equipes que buscam uma sintaxe amigável possam preferir o Quint. Observamos equipes verificando traces de aplicações em relação a um modelo TLA+ sem experiência prévia.

Enquanto isso, pessoas desenvolvedoras agora têm maior necessidade de técnicas de garantia de qualidade para gerenciar código gerado por agentes. Onde o TLA+ antes era exclusividade de organizações como a AWS, que o usava para encontrar bugs sutis no DynamoDB, agora ele faz parte de uma tendência mais ampla de verificação poliglota, ao lado de técnicas como testes de mutação e testes unitários baseados em propriedades.

© Thoughtworks, Inc. All Rights Reserved.

Mantenha-se atualizada com os artigos e informações relacionados ao Radar

Assine o Technology Radar para receber e-mails mensais com insights de tecnologia da Thoughtworks e divulgações dos futuros volumes.

Assine



Somos uma consultoria global de tecnologia que promove impacto extraordinário ao combinar design, engenharia e IA.

Há mais de 30 anos, nossa cultura de inovação e excelência tecnológica tem ajudado nossas clientes a aprimorar seus sistemas, escalar com agilidade e criar experiências digitais integradas.

Estamos comprometidos em resolver os desafios mais complexos de nossas clientes, combinando IA e criatividade humana para transformar suas ideias audaciosas em realidade.

/thoughtworks

Design. Engenharia. IA.