EDITORIAL

Agent Harness - Novas arquiteturas de software que estão redesenhando as operações das empresas

Transição de API para o Agent Harness e a IA Física

Ao considerar a transição da API para Agent Harness, a arquitetura de um software proporciona uma mudança profunda nas capacidades de TI, que pode impactar diretamente a operação de uma empresa ou até mesmo o seu modelo de negócio. Atualmente, os programadores definem o "Agent Manifest" para descrever a propósito de um agente e, ao incorporar o harness, adicionam condições para que os sistemas possam simular comportamentos. Essa mudança revolucionária já começou. Um Agent Harness é preparado para ser executado num ambiente (runtime) que provê todos os recursos necessários aos modelos de IA, uma plataforma para torná-lo um agente com propósito funcional e seguro. Em suma, Agente = Modelo (cérebro) + Harness (corpo/ambiente). Essa formulação redesenha completamente a relação funcional entre os componentes de software que transformam as operações de uma empresa.

Numa abstração livre, um **Agent Harness transforma sistemas do tipo "chamar um modelo" em sistemas do tipo "implantar um funcionário".** Essa mudança é grande o suficiente para justificar que as novas arquiteturas de software geram novos modelos operacionais nas empresas.

Além disso, o **avanço da IA física** (veja: Prometheus – Jeff Bezos), com a aplicação de VLMs (Vision-Language Models) em práticas de manufatura, logística e operação industrial, multiplica as possibilidades de novas formas de executar tarefas, processos, workflows, modelos, e toda uma operação ou negócio empresarial.

Nas próximas seções deste artigo, esses conceitos serão relacionados — Manifesto do Agente, Agent-Harness, Comportamento, Segurança — em três camadas:

- Arquitetura de software
- Operações da empresa
- Modelos de negócios/receita

1. Arquitetura de software: das chamadas de API aos sistemas centrados em agentes

O uso tradicional de LLMs consiste em uma "chamada de função sem estado": você envia um prompt para uma API, recebe uma resposta, e o restante do aplicativo é código convencional.

Com o Agent Harness, a arquitetura gira em torno dos agentes, em vez de chamadas individuais ao modelo. Isso normalmente envolve a construção de blocos em várias arquiteturas.

1.1 Manifesto de um Agente como um contrato de primeira classe

O Manifesto do Agente se torna um novo tipo de contrato de interface, semelhante a uma especificação de API, mas mais abrangente, por exemplo:

- O que o agente pode fazer: ferramentas, acesso a dados, operações permitidas.
- Sob quais restrições: políticas, limites de segurança, regras de segurança.
- Como ele se comporta: objetivos, estilo de interação, estratégia de memória, regras de escalonamento.
- Como ele é composto: subagentes, fluxos de trabalho, dependências.

Do ponto de vista de uma arquitetura, isso possibilita:

- Registros de agentes: um serviço onde os agentes são publicados, versionados e descobertos de maneira muito semelhante à dos microsserviços atualmente.
- Interfaces para tipos de agente: outros serviços chamam um agente não apenas com "strings de prompt", mas com intenções estruturadas predefinidas no manifesto (por exemplo, CreateInvoice, TriageTicket, DraftProposal).

Isso leva você da "integração com LLM" para a arquitetura orientada a agentes, na qual os agentes são pares de serviços e aplicativos.

(...continua)

Zózimo De Souza Jr.
editor-chefe

PROJECT DESIGN
PDM
MANAGEMENT

Editor-Chefe & Publisher
Osmar Zózimo De Souza Jr.
zozimo@mundopm.com.br
projectdesignmanagement.com.br

INFORMAÇÕES

ISSN: 1807-8095

Emails:
assinaturas@mundopm.com.br
artigos@mundopm.com.br

Redes Sociais:
linkedin.com/in/mundopm/

1.2 O *Agent Harness como uma camada de execução*

Se o modelo é o cérebro, o Harness é o ambiente de execução que transforma previsões em comportamento controlado. E geralmente inclui estas camadas:

- Orquestração e planejamento: dividir os objetivos do usuário em etapas, decidir quais ferramentas chamar, quando parar e quando solicitar esclarecimentos.
- Gerenciamento de estado e memória: armazenar a memória de trabalho (tarefa atual) e a memória de longo prazo (interações passadas, perfis, conhecimento).
- Aplicação de políticas: verificar cada ação (chamada de ferramenta, acesso a dados, efeito externo) em relação às políticas de segurança, conformidade e proteção.
- Observação e controle: registros, métricas, rastreamentos, pontos de revisão humana, lógica de reversão para ações com efeitos colaterais.

Do ponto de vista da arquitetura de software, isso se parece com uma nova camada de "Ambiente de Execução de um Agente" que fica entre os:

- Front-ends/canais (aplicativo web, Slack, e-mail, APIs)
- Sistemas de back-end (bancos de dados, microsserviços, APIs de terceiros)
- Provedores de modelos (LLMs, RL, VLMs, outros modelos de ML)

Em vez de conectar diretamente os front-ends aos modelos, você conecta os front-ends ao agente num ambiente de execução, e esse agente utiliza os modelos nos bastidores.

1.3 *De chamadas síncronas a fluxos de trabalho orientados a objetivos*

As arquiteturas centradas em API são, em grande parte, do tipo solicitação/resposta e síncronas.

Arquiteturas centradas em agentes são operadas com:

- Interfaces orientadas a objetivos: "Resolva X dentro das restrições Y", em vez de "execute a função Z uma vez".
- Fluxos de trabalho orientados a eventos: os agentes respondem a eventos (novo ticket, novo pedido, anomalia detectada) e iniciam o trabalho de forma autônoma.
- Sessões de longa duração: o mesmo agente (ou identidade de agente) lida com um cliente ou caso ao longo do tempo, recorrendo à memória.
- Colaboração entre múltiplos agentes: agentes especializados (por exemplo, "FinanceAgent", "SupportAgent", "ComplianceAgent") coordenam-se em torno de um registro compartilhado (tíquete, caso, conta).

E, isso geralmente requer:

- Um mecanismo de fluxo de trabalho que possa ser acionado por agentes, e não apenas por etapas estaticamente codificadas.
- Um armazenamento de contexto compartilhado (bancos de dados vetoriais, memórias de casos, histórico de conversas) acessível por meio do Harness.
- Limites claros de autoridade por agente definidos pelo manifesto (ferramentas permitidas, dados, nível de risco).

1.4 *Segurança, governança e testes como questões centrais da arquitetura*

Assim que os agentes passam a agir, a governança torna-se uma questão de arquitetura, e não apenas de uma política:

- Serviços de proteção: componentes separados que verificam entradas, saídas e chamadas de ferramentas quanto à segurança e conformidade.
- Conjuntos de testes e ambientes de teste isolados: o comportamento do agente é testado com cenários, simulações e ambientes de simulação de ataques (red-teaming) antes da entrada em produção.
- Auditabilidade: registros estruturados do "processo de raciocínio" (planos, escolhas de ferramentas, políticas acionadas) para que os seres humanos possam compreender e depurar.

Isso se torna uma camada explícita da arquitetura: "Governança e Observabilidade do Agente", análoga ao que os gateways de API e as malhas de serviços (meshes) representam para os microsserviços.

2. Operações da empresa: de desenvolvedores "usando um modelo" para equipes "responsáveis por agentes"

Ao adotar um Agent Harness, as funções e os processos organizacionais em torno da TI geralmente precisam mudar.

2.1 Novas funções em torno de produto e engenharia

Será comum encontrar funções como:

- Proprietários de produto de agentes: definem o que um agente deve fazer, onde ele se encaixa nas jornadas e os KPIs pelos quais é responsável (por exemplo, tempo de resolução, taxas de erro).
- Engenheiros de segurança/políticas: codificam as regras organizacionais em manifestos, ferramentas e medidas de proteção.
- Engenheiros de confiabilidade de agentes (AIOps/MLOps): monitoram o desempenho, desvios, regressões e incidentes dos agentes.
- Responsáveis pelas ferramentas: equipes responsáveis pelas ferramentas que os agentes utilizam (por exemplo, "ferramentas de faturamento", "ferramentas de CRM"), tratando os agentes como "clientes" internos.

Operacionalmente, em vez de "lançar um recurso que chama o GPT", as equipes lançam e mantêm agentes como recursos persistentes com SLAs.

2.2 Redesenho de processos em torno da colaboração entre humanos e agentes

Como um agente é capaz de manter o contexto ao longo do tempo e agir de forma autônoma, os processos podem ser redesenhados:

- Orquestração do trabalho:
  - Os agentes lidam com a classificação, o encaminhamento e as primeiras ações (rascunhos, verificações, enriquecimento).
  - Os humanos lidam com julgamentos, aprovações, exceções e trabalho de relacionamento.
- Escalonamento e transferências:
  - Os manifestos definem quando um agente deve escalar: baixa confiança, conflitos de políticas, decisões de alto risco.
  - O Harness aplica essas regras e transmite o contexto estruturado para que os humanos não precisem refazer o trabalho.

- Treinamento contínuo em operações:
  - Os ciclos de feedback (aprovação/reprovação, correções) são incorporados às configurações, prompts, ferramentas ou até mesmo ao ajuste fino dos agentes.
  - As equipes de operações passam a tratar os SOPs como "material de treinamento para agentes", e não apenas como documentos para humanos.

Isso pode reduzir as etapas do processo, eliminar transferências de tarefas e alterar o tipo de trabalho que os humanos realmente realizam.

2.3 *Governança e operações de risco*

Como os agentes podem agir, a gestão de riscos torna-se mais operacional:

- "Níveis de permissão" claros para os agentes (somente leitura, somente rascunho, pode acionar a API, pode fazer alterações até o valor X etc.).
- Fluxos de trabalho de aprovação: o agente produz um resultado; certas categorias são enviadas automaticamente, enquanto outras exigem aprovação humana.
- Política como código: regras de conformidade, jurídicas e de risco codificadas no Harness, não apenas em PDFs.

As empresas que acertam nesse ponto tratam o Harness como seu ponto central de controle do comportamento da IA e realizam gerenciamento de mudanças, aprovações e auditorias por meio dele.

3. Modelos de negócios: de "features com IA" a "agente como serviço"

Depois de ter um Agent Harness robusto, você pode criar novas ofertas que não fariam sentido apenas com APIs.

3.1 *Produtos de automação no nível de processo ou função*

A empresa, em vez de ofertar "uma API de IA" ou "IA integrada ao produto", a empresa pode ofertar um produto inteligente:

- "Células de trabalho de agentes": agentes responsáveis por resultados comerciais específicos (por exemplo, "agente de cobrança que reduz o DSO", "agente de triagem de suporte", "agente de pesquisa de vendas").
- Processo como serviço: você opera um processo impulsionado por agentes de ponta a ponta para os clientes (por exemplo, recebimento de reclamações, integração com fornecedores, análise de KYC).

O Harness é o que permite que a empresa garanta:

- Segurança (você pode comprovar o que o agente está autorizado a fazer).
- Observabilidade (você pode medir seu trabalho).
- Controlabilidade (você pode alterar o comportamento de forma centralizada por meio de manifestos e de ferramentas).

3.2 *Precificação baseada em resultados ou no uso*

Como os agentes podem ser vinculados diretamente a métricas de negócios, você pode mudar os modelos de precificação:

- Por resultado: custo por ticket resolvido, lead qualificado, fatura processada ou caso de fraude evitado.
- Por agente / por licença: como "comprar 100 agentes digitais que podem, cada um, lidar com N tarefas simultâneas".
- Capacidade em camadas: camadas mais altas desbloqueiam ferramentas e permissões mais poderosas para os agentes (por exemplo, acesso básico de somente leitura versus autoridade total para ações).

O Manifesto do Agente, neste caso, torna-se não apenas uma configuração técnica, mas também uma definição comercial de SKU: ele codifica o que o cliente está pagando (capacidades, limites, restrições).

3.3 *Ecossistemas e mercados de agentes*

Com um Harness consistente a empresa pode gerar uma integração ampla com:

- Terceiros, que podem publicar agentes ou ferramentas que se integram ao seu ambiente com segurança, pois o Harness garante o isolamento e a aplicação das políticas.
- Equipes internas, que podem desenvolver agentes específicos para cada domínio e compartilhá-los em uma plataforma interna para reutilização por toda a empresa.

Isso leva você a um modelo de plataforma em que seu valor não está apenas nos seus próprios agentes, mas também no ecossistema que pode ser executado com segurança no seu Harness.

4. Como fazer a transição de forma pragmática de uma API para o Agent Harness

Para tornar isso mais concreto, um caminho mais sensato é um passo a passo conforme sugerido a seguir:

1. Introduzir o Manifesto do Agente.
2. Comece formalizando um ou dois agentes na forma de manifesto: objetivos, ferramentas, políticas, memória. Trate isso como um novo tipo de interface em sua arquitetura.
3. Implemente um ambiente de execução de agente / harness
4. Implemente ou adote um ambiente de execução que:

- *Interprete manifestos*
- *Orquestre chamadas de modelos e chamadas de ferramentas*
- *Aplique políticas e registre tudo*

5. Reestruturar um fluxo de trabalho existente
6. Escolha um domínio bem delimitado (triagem de suporte, FAQ interna, reconciliação de faturas) e reprojete-o de forma que:

- *O agente seja responsável pelo fluxo de trabalho*
- *Os humanos supervisionem e lidem com exceções*
- *As métricas e os limites de segurança estejam claramente definidos*

7. Institucionalize a governança e a responsabilidade pelos agentes
8. Defina quem é o responsável por cada agente, como as alterações são testadas e implantadas e como os incidentes são tratados.
9. Generalize para uma "Plataforma de Agentes" interna
10. Assim que o padrão funcionar, exponha o Harness a outras equipes para que elas possam definir seus próprios agentes com manifestos, em vez de escrever integrações LLM personalizadas.

Em resumo, como o Agent Harness padroniza a forma como os modelos interagem com ferramentas, dados, políticas e pessoas, ele permite uma:

- Reestruturação da arquitetura, passando de "pontos de chamada de LLM" para "ambientes de execução de agentes e manifestos" como contratos de primeira classe.
- Transformação das operações, de modo que as equipes sejam responsáveis por agentes dinâmicos com autoridade definida, em vez de recursos estáticos com algum aprendizado de máquina (ML) incorporado.
- Criação de novos modelos de negócios baseados em resultados automatizados e trabalhadores digitais reutilizáveis e governados.

5. Implicações e Discussão

Os líderes em empresas de tecnologia de ponta apresentam uma ampla variedade de visões sobre esse assunto. Alguns destacam que:

- Agentes de codificação e "engenharia de harness": alguns argumentam que os repositórios de código são especialmente adequados para a IA porque o contexto é textual, centralizado e autocorretivo (testes, erros de compilação). Outros estão cada vez mais céticos em relação ao MCP (protocolo de contexto de modelo), preferindo uma memória desorganizada no estilo de arquivos Markdown (à la OpenClaw), pois ela carrega mais contexto do que as rígidas arquiteturas multiagentes baseadas em API.
- O futuro é "harness de agentes": Alguns preveem que as empresas de tecnologia precisarão de um "harness de agentes" — documentação e ferramentas otimizadas para uso em IA, e não para o design de interface de usuário humano — atuando em paralelo às APIs tradicionais.
- Mudanças nos negócios da empresa: Agentes de IA para atendimento ao cliente agora lidam com chamadas telefônicas, chat e web para muitas empresas de tecnologia, automatizando 70–90% dos casos. Alguns passarão a utilizar a tarifação baseada em resultados (pagamento somente quando um agente resolve um caso) em vez da baseada no uso, o que alguns consideram mais alinhado com os clientes e mais disruptivo para a economia do software legado.
- Sistemas de registro x Sistemas de engajamento: Alguns argumentam que os agentes de IA podem minar a durabilidade do software tradicional de "sistemas de registro/cadastro" (CRM, plataformas de marketing), enquanto sistemas semelhantes a livros contábeis (ledgers - livros razão) permanecem duráveis.
- Produtividade da IA: Alguns defendem a tese central — "a unidade atômica da produtividade da IA é um processo, não uma pessoa". As empresas têm dificuldade em capturar os ganhos da IA porque estão organizadas em torno de departamentos/organogramas, em vez de processos de ponta a ponta.
- Previsões para 2026: Um "momento AlphaGo" científico para a IA, a adoção generalizada de agentes e a maior parte do código do Vale do Silício sendo escrita por IA, em vez de ser codificada manualmente.

O modelo de criação de empresas de tecnologia muda após a IA, com **foco na transição de uma hierarquia baseada em funções para generalistas com alta autonomia,** possibilitada pela IA. Destacando os seguintes pontos:

1. Os líderes técnicos (não os gerentes) tornam-se mais importantes. Os agentes de IA aceleram essa dinâmica.
2. Ascensão do "hipergeneralista" / "engenheiro de produto". Anteriormente, os generalistas — pessoas com bom gosto, alguma habilidade em infraestrutura e profundo entendimento do cliente, mas sem uma única especialidade técnica aprofundada — eram deixados de lado à medida que as empresas cresciam, porque "não havia lugar" para quem sabia fazer de tudo.
3. Organizações mais planas, hierarquia menos orientada à especialização. Porque o impacto individual se torna muito maior com a IA.
4. Reorientação em torno de "alta autonomia + segurança", e não em torno de credenciais. Executam suas próprias ideias diretamente, em vez de precisarem de convencer ou montar uma equipe.

Em resumo, o modelo antigo (silos funcionais de engenheiros/gerentes de produto/designers com proporções definidas e processos com transferência de tarefas – "handoff") conferia poder estrutural aos especialistas e às camadas de gestão. Na era pós-IA, muitos acreditam que a unidade mais valiosa e duradoura é o profissional generalista com bom gosto e percepção do cliente, capaz de usar a IA para executar um processo de ponta a ponta — levando as empresas a adotar estruturas mais planas, organizadas em torno de processos e indivíduos empoderados, em vez de departamentos e proporções de pessoal.

A IA física não contradiz esta tese — ela amplia seus limites. Os modelos VLA estão, essencialmente, tentando criar essa mesma propriedade para o mundo físico, ou seja, transformar fluxos de sensores e demonstrações práticas em algo tão treinável e transferível como um repositório de código. Os setores de logística, manufatura e serviços em ambientes estruturados (aeroportos, armazéns, hotéis) são onde esses limites/fronteiras estão se movendo visivelmente, e onde o "modelo para construir uma empresa de tecnologia após o surgimento da IA" — descritas como hipergeneralistas, com propriedade sobre os processos e uma economia baseada em resultados — está começando também a ser aplicado nas operações das empresas que envolvem a parte física (Indústria 4.0), e não apenas numa tela digital.

Zózimo De Souza Jr.
editor-chefe