

// METODOLOGIA

Sciensa · Metodologia 2026

Como o Product Studio constrói produtos de inteligência:
fundamentos e prática

Scienza · Metodologia 2026

00	PARTE 0 Por que trabalhamos assim	3
01	PARTE 1 O modelo em quatro camadas	5
02	PARTE 2 Portfólio, apostas e gates	7
03	PARTE 3 Produto de IA, oito decisões e seis fases	9
04	PARTE 4 Entrega e cadência do time	13
05	PARTE 5 Engenharia e arquitetura emergente	17
06	PARTE 6 Governança, risco e conformidade	20
07	PARTE 7 Pessoas e agentes	22
08	PARTE 8 Adoção e maturidade	23

Anexos

— templates A a G · Apêndice H · Guia de Cultura de Engenharia (uso interno) · glossário · base científica · nível de confiança

25

Por que trabalhamos assim

O problema que a metodologia resolve

Construímos produtos cuja peça central é uma inferência (um score, uma classificação, uma recomendação) e não uma tela. Isso muda três coisas em relação ao software convencional, e as três estão documentadas na literatura:

1. **A qualidade não é binária.** Um sistema de aprendizado de máquina “funciona” dentro da distribuição em que foi avaliado e degrada fora dela; a dívida técnica se acumula em dado, configuração e ciclos de feedback, não apenas em código. Testar o código não basta: é preciso testar dado, modelo, infraestrutura e monitoramento.
2. **O produto é uma mudança de workflow, não um modelo.** O estudo de caso da Microsoft com mais de quinhentos engenheiros mostra que os gargalos de sistemas de IA estão em descoberta e gestão de dado, integração e operação, não em algoritmo. O MIT xPRO chama esse estágio de Business Process: qual etapa do processo muda, quem age antes e depois da IA, que política precisa existir.
3. **Alguém responde por cada inferência.** Sistemas que tomam ou preparam decisões sobre pessoas exigem trilha de auditoria de ponta a ponta e um nível de automação escolhido deliberadamente, não herdado da ferramenta.

Somamos a isso o contexto em que operamos: times pequenos e seniores, agentes de IA trabalhando ao lado das pessoas, e clientes cuja propriedade intelectual será auditada por terceiros. A metodologia existe para que essas condições produzam entrega consistente, e para que qualquer pessoa, dentro ou fora da Sciensa, consiga verificar isso sem precisar confiar na nossa palavra.

O que ela não é

- **Não é um framework certificável.** Não há papéis-cerimônia nem artefatos por obrigação. Cada prática desta metodologia tem uma razão escrita ao lado; quando a razão deixa de valer, a prática sai.
- **Não é Scrum, SAgile ou “ágil” como rótulo.** Usamos o que a evidência sustenta e descartamos o resto. As revisões sistemáticas sobre métodos ágeis são explícitas sobre a fragilidade da evidência agregada sob esse guarda-chuva; por isso a base científica deste documento (Anexo J) está organizada prática por prática, e não pelo rótulo.
- **Não é uma dependência tecnológica.** Os agentes de IA que usamos são parte do método, nunca parte do produto entregue ao cliente.

Os sete princípios

#	PRINCÍPIO	O QUE SIGNIFICA NA PRÁTICA
1	Software rodando é a única prova	Gate, showcase e marco só contam com sistema funcionando sobre dado real. Slide não é evidência; o que foi construído e ainda não rodou é ativo, e entra no gate quando roda
2	Hipótese antes de escopo	Toda aposta nasce como “acreditamos que... saberemos porque...”. O escopo é consequência da hipótese, não o contrário
3	Fatia fina, ponta a ponta, primeiro	A primeira entrega de qualquer sistema atravessa todas as camadas com um caso real antes de aprofundar qualquer uma delas
4	Decisão tem dono, prazo e default	Nenhuma decisão fica aberta sem nome, data e o que acontece se não for tomada. Decidimos no último momento responsável, não no último possível
5	Humano responsável por toda inferência	Cada modelo tem quem aprova método, threshold e publicação. Cada agente de IA tem quem revisa a saída
6	Menos, bem construído, documentado	Preferimos um catálogo curto e auditável a um amplo e frágil. Decisão de arquitetura é registrada no momento em que é tomada
7	Tudo versionado, tudo observável	Código, dado, modelo, prompt, eval set e decisão têm versão. O que não é observado não está em produção

O modelo em quatro camadas

Organizamos o trabalho em quatro camadas. Cada uma responde a uma pergunta diferente, tem seu próprio ritmo e sua própria evidência de que está funcionando. Elas não são fases: uma aposta atravessa as quatro em semanas, e um projeto tem dezenas de apostas em estágios diferentes ao mesmo tempo.

CAMADA	PERGUNTA QUE RESPONDE	RITMO	EVIDÊNCIA DE QUE FUNCIONA
Portfólio	Em que apostamos, e como saberemos que deu certo?	Ondas de 3 a 6 meses, gate no fim	Hipótese escrita na abertura, veredito medido no fechamento
Produto de IA	Que decisões este produto precisa ter tomado, e em que ordem?	Semanas, por aposta	As oito decisões registradas; os seis gates cruzados
Entrega	Como o time transforma isso em software toda semana?	Iteração de 1 a 2 semanas; showcase quinzenal	Fluxo medido (lead time, WIP); showcase com dado real
Engenharia	Como provamos, continuamente, que está certo?	Todo commit	Pipeline verde, fitness functions, métricas DORA

Como uma aposta atravessa as quatro camadas

- 01 Nasce no portfólio como hipótese, com medida de sucesso e uma janela de tempo.
- 02 Ganha um AI Product Brief (as oito decisões do backbone MIT) e percorre as seis fases operacionais, DEFINE a IMPLEMENT, em fatias.
- 03 Cada fatia entra no fluxo de entrega: kick-off, par, desk check, showcase.
- 04 Cada commit sai pela engenharia: pipeline, fitness functions, contrato, avaliação de modelo, deploy.
- 05 O showcase alimenta o veredito da hipótese, que fecha ou reabre a aposta.

O que atravessa as quatro camadas é o Company Brain (Parte 4): o repositório canônico do projeto, onde a hipótese, o brief, a decisão, o veredito e a fonte de cada um ficam registrados e ligados.

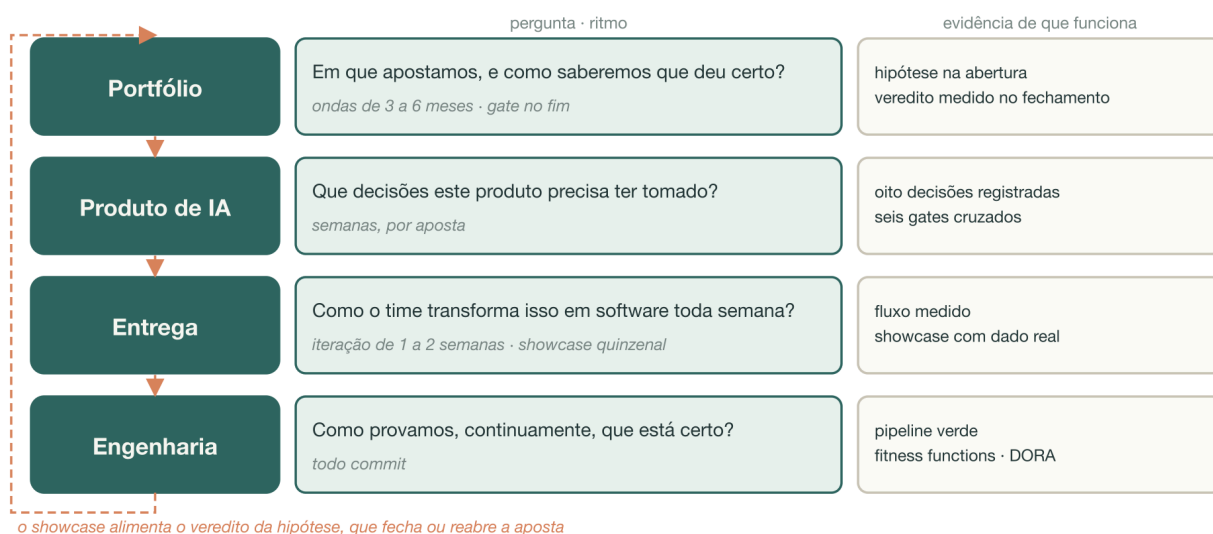


Figura 1 • As quatro camadas e o caminho de uma aposta

Por que quatro camadas, e não um processo único

A literatura de inovação de produto converge em um ponto: processos de estágio dão governança mas engessam; métodos iterativos dão velocidade mas perdem o portfólio de vista. A síntese que a evidência apoia é o híbrido: gates no nível do portfólio, iteração no nível do time. Separar as camadas é o que permite ter os dois sem que um contamine o outro: o gate não pergunta “quanto foi feito”, e a iteração não precisa justificar-se ao Steering.

Portfólio, apostas e gates

A árvore de valor

Descrevemos o portfólio como uma árvore: **visão** → **metas** → **apostas** → **iniciativas**, a árvore de valor do EDGE. A árvore substitui o backlog plano: cada iniciativa aponta para a aposta que serve, e cada aposta para a meta que move. O que não está na árvore não é feito. Quando dois documentos descrevem o mesmo trabalho com régua de tempo diferentes, é sinal de que existem duas árvores onde deveria haver uma.

O formato de hipótese

Toda aposta é escrita assim antes de qualquer linha de código:

Acreditamos que [capacidade] para [quem] vai produzir [resultado].
Saberemos que é verdade quando [medida observável] atingir [valor] até [data].
Se não for, [o que fazemos].

O formato vem da experimentação contínua em software e da disciplina de experimentos controlados: sem medida definida antes, todo resultado parece confirmar a aposta. Steve Blank resume a consequência para produto: o plano é uma lista de hipóteses não testadas, e o trabalho é testá-las na ordem do maior risco.

Ondas e gates

Trabalhamos em ondas de três a seis meses. Cada onda abre com hipóteses escritas e fecha com um gate (**go, pivot, pause ou stop**) decidido sobre as medidas, não sobre o andamento. Adotamos explicitamente o híbrido Agile-Stage-Gate que a pesquisa em gestão de inovação vem validando: o gate pertence ao portfólio, a iteração pertence ao time, e o gate pergunta “a hipótese sobreviveu?”, nunca “quanto foi entregue?”.

Uma regra decorre disso e é aplicada sem exceção: **o gate avalia evidência, não capacidade**. Um componente construído antes da hora é ativo, e ativo antecipado é o que dá folga à onda seguinte; mas ele só entra na avaliação do gate depois de rodar com dado real. Até lá, conta no balanço do projeto, não no veredito da hipótese.

Avaliação em quatro camadas

Medimos cada aposta em quatro camadas, como o MIT xPRO propõe, e é assim que evitamos lançar “porque o modelo parece bom”:

CAMADA	EXEMPLOS DE MEDIDA	QUEM PRODUZ
Modelo	Precisão por classe, cobertura, calibração de confiança, taxa de falha crítica, groundedness	Applied Data Scientist

CAMADA	EXEMPLOS DE MEDIDA	QUEM PRODUZ
Experiência	Compreensão, confiança calibrada do usuário, esforço, adoção, taxa de correção humana	Product Designer
Operação	Latência, custo por tenant, disponibilidade, tempo de pipeline, tempo de recuperação	DevSecOps / SRE
Negócio	Tempo poupado, conversão, expansão por módulo, custo evitado, risco reduzido	CPO, com o cliente

A regra de avaliação é a do documento MIT consolidado: o sistema precisa demonstrar comportamento útil no escopo, experiência controlável, operação sustentável e resultado superior ao baseline, nas quatro camadas.

Governança leve

A governança do portfólio acontece em dois fóruns. O **Steering** (quinzenal ou mensal) decide escopo, risco e economics; o **Stage Gate** (fim de onda) decide continuidade. Ambos recebem medidas, não relatórios de andamento. É a “governança leve” do EDGE: financia-se resultado, não output, e a autonomia do time é a contrapartida da transparência das medidas.

Produto de IA, oito decisões e seis fases

Esta é a camada em que adotamos o backbone do programa *Designing and Building AI Products and Services*, do MIT xPRO, e o operacionalizamos. Seguimos a distinção que o próprio documento consolidado faz: os quatro estágios e as oito decisões são estrutura do MIT; as seis fases, a escada de autonomia, os canvases e os gates são adaptação nossa. Ao citar externamente, atribuímos ao MIT apenas o que é do MIT.

Os quatro estágios e as oito decisões

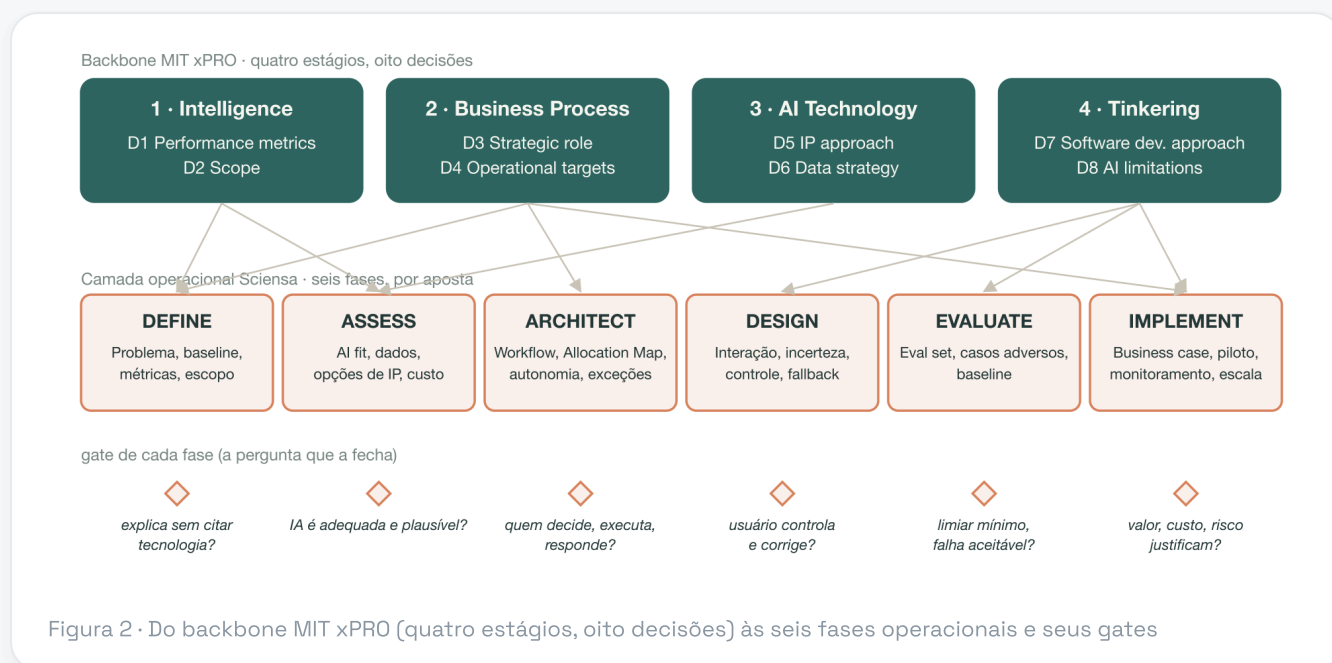
ESTÁGIO	DECISÃO	PERGUNTA QUE RESPONDEMOS	ONDE FICA REGISTRADA
1 • Intelligence	D1 Performance metrics	Como medimos que o comportamento inteligente é bom o suficiente?	AI Product Brief · bloco Metrics
1 • Intelligence	D2 Scope	Onde precisa funcionar bem, e onde não precisa?	Brief · bloco Scope
2 • Business Process	D3 Strategic role	A IA torna o produto melhor, resolve uma parcela maior do problema, ou cria efeito de sistema?	Brief · Strategic role
2 • Business Process	D4 Operational targets	Qual etapa do processo muda, quem age antes e depois, que política e que ativo complementar precisam existir?	Brief · Business process; service blueprint
3 • AI Technology	D5 IP approach	Build, buy, adapt ou compose?	ADR
3 • AI Technology	D6 Data strategy	Que dado sustenta o comportamento, quem é dono, como obtemos e avaliamos, qual feedback melhora o sistema?	Brief · Data; datasheet
4 • Tinkering	D7 Software development approach	Como construímos, testamos e mantemos flexibilidade num ambiente que muda rápido?	Parte 5 desta metodologia
4 • Tinkering	D8 AI limitations	Que classes de falha existem e como cada uma é tratada?	Registro de riscos; model card

A ordem é a razão de adotarmos o modelo: comportamento e métrica vêm antes de processo; processo vem antes de tecnologia. É a inversão da sequência comum, “temos um modelo, onde usamos?”. Uma decisão pode ficar provisória, e o registro diz *provisória, visitar em [gate]*; o que não pode é ficar implícita.

As seis fases operacionais

As seis fases organizam o trabalho de uma aposta até que as oito decisões estejam tomadas e o sistema em operação. **Rodam por aposta, não por projeto.** Uma aposta percorre DEFINE a IMPLEMENT em semanas, e várias apostas estão em fases diferentes ao mesmo tempo. Se as fases virassem etapas do projeto, seria cascata com vocabulário de IA, e a evidência contra cascata em ambiente de incerteza tem quase quarenta anos.

FASE	ÂNCORA MIT	ARTEFATOS	GATE (A PERGUNTA QUE FECHA A FASE)
DEFINE	Intelligence + Business Process	Problem statement, baseline do processo, métricas de produto e de inteligência, escopo inicial, mapa de stakeholders	O time explica o problema e o comportamento desejado sem citar uma tecnologia?
ASSESS	Intelligence + AI Technology	AI Fit Canvas, inventário de dados, opções build/buy/adapt/compose, riscos técnicos, estimativa de custo e latência	Há evidência de que IA é adequada e tecnicamente plausível para o escopo?
ARCHITECT	Business Process + HCI + Superminds	Service blueprint, Human-AI Allocation Map, escada de autonomia, mapa de exceções, políticas de aprovação	Está claro quem percebe, recomenda, decide, executa e responde quando algo dá errado?
DESIGN	HCI + Tinkering	Fluxos principais, estados de incerteza, protótipo, fallbacks, regras de reversão e confirmação	O usuário consegue orientar, entender, controlar, corrigir e interromper a IA?
EVALUATE	Tinkering	Eval set versionado, casos adversos, testes com usuários, métricas de operação, comparação contra baseline	O sistema atinge o limiar mínimo e falha de forma aceitável para o contexto?
IMPLEMENT	Business Process + Tinkering + business case	Business case, plano de piloto, modelo de ownership, plano de monitoramento, critérios de escala	Valor, custo, risco e governança justificam ampliar o uso?



As ferramentas

AI Product Brief. Um documento de uma página por aposta, com doze blocos: problema, intelligence, metrics, scope, business process, strategic role, technology, data, human + AI, failure, business case, next experiment. É a entrada do kick-off e o registro vivo das oito decisões. Template no Anexo A.

AI Fit Canvas. Nove dimensões (natureza da tarefa, variabilidade, dados e contexto, erro tolerável, latência, explicabilidade, custo, integrações, reversibilidade, observabilidade) para decidir se IA é o mecanismo certo antes de escolher qual. Anexo B.

Human-AI Allocation Map e escada de autonomia. Para cada função cognitiva do sistema (sense, remember, create, decide, act, learn) dizemos quem faz: humano, IA ou ambos. A unidade de design deixa de ser “um usuário usando uma ferramenta” e passa a ser o sistema de inteligência coletiva. Para cada ação da IA escolhemos um degrau: **informar, sugerir, recomendar, preparar, executar com supervisão, executar autonomamente.** A escada é a nossa operacionalização de uma linha de pesquisa de duas décadas sobre tipos e níveis de automação, e a escolha do degrau segue as diretrizes de interação humano-IA que a pesquisa em HCI consolidou. Em especial: deixar claro o que o sistema pode fazer e quão bem, mostrar por que fez o que fez, e permitir correção eficiente.



Figura 3 • A escada de autonomia

Classes de risco. Adversarial, generalização, viés e falhas generativas. As três primeiras vêm do MIT; a quarta é extensão nossa, alinhada às taxonomias de risco de modelos de linguagem e ao NIST AI RMF. Cada classe tem um controle nomeado na Parte 6.

Entrega e cadência do time

Product Studio

Trabalhamos como Product Studio: um time misto Sciensa e cliente, decisões compartilhadas, accountability da Sciensa sobre arquitetura, engenharia, qualidade e entrega; o cliente mantém a liderança metodológica e as decisões de produto. O núcleo é pequeno e sênior: profissionais principal e staff que decidem e produzem diretamente, sem camadas intermediárias de coordenação. Enablement técnico entra conforme a entrega pede.

A forma do time segue a evidência. Times pequenos com autonomia superam times grandes com coordenação. A estrutura de comunicação do time determina a estrutura do sistema que ele produz, o que faz da topologia do time uma decisão de arquitetura. E a segurança psicológica é o preditor mais forte de aprendizado em times, razão pela qual o showcase e a retrospectiva são onde se diz que algo não deu certo, e não onde se esconde.

Lean Inception

Toda onda, e todo projeto, abre com uma Lean Inception: uma semana, ou três dias na versão compacta, em que time e cliente alinham visão do produto, o que ele é e não é, personas, jornadas, funcionalidades, revisão técnica, de UX e de negócio, sequenciador e canvas de MVP. A saída não é um backlog. É o conjunto de hipóteses da onda, o primeiro AI Product Brief de cada aposta e a fila de decisões inicial. A prática descende do inception da Thoughtworks e da tradição de descoberta rodando em paralelo à entrega.

Cadência

FÓRUM	CADÊNCIA	QUEM	O QUE ACONTECE
Stand-up	Diária, 10 min	Time e enablement ativo	Bloqueios e pares do dia
IPM + fila de decisões	Semanal	CPO, Data Product, Domain Lead, Architect, time	Puxa as histórias da semana; revisa cada decisão aberta: dono, prazo, default
Showcase	Quinzenal	Cliente, Sciensa, usuários convidados	Software rodando com dado real. Sem slide
Retrospectiva	Quinzenal, após o showcase	Time	O que muda na próxima quinzena: uma ação, com dono
Model & Data Council	Mensal	Dados, IA, domínio, segurança	Eval sets, thresholds, drift, viés, sobre dashboard e não relatório
Steering	Quinzenal ou mensal	Sponsors e lideranças	Escopo, risco, economics

FÓRUM	CADÊNCIA	QUEM	O QUE ACONTECE
Stage Gate	Fim de onda	Steering + sponsors	Veredito das hipóteses: go, pivot, pause, stop

Fluxo

- **Kanban com limite de WIP.** Uma história por par em andamento. A base é a Lei de Little: tempo de atravessamento = trabalho em andamento ÷ vazão. Para entregar mais rápido com o mesmo time, reduz-se o que está em andamento, não se aumenta o esforço. A revisão sistemática de Kanban em software confirma redução de lead time e ganho de visibilidade como os efeitos mais consistentes.
- **Iteração de uma semana** enquanto o esqueleto do sistema não roda de ponta a ponta; duas semanas depois disso.
- **Story kick-off.** Antes de começar uma história, quem desenvolve, quem define e quem valida conversam por quinze minutos sobre o que “pronto” significa para ela.
- **Desk check.** Antes de fechar, quem desenvolveu mostra a história rodando para quem validou no kick-off. Substitui a fase de QA no fim.
- **Definition of Done.** Testada, integrada no tronco, aprovada no pipeline, implantada no ambiente da onda, observável, e com a decisão registrada se houve decisão. Anexo D.
- **Par por padrão** em arquitetura, integração entre sistemas e em qualquer coisa que só uma pessoa sabe fazer. As meta-análises sobre pair programming encontram efeito modesto em qualidade e negativo em esforço, e forte em transferência de conhecimento. Usamos o par pelo terceiro efeito, não pelo primeiro: em times de cinco, a pessoa que sabe sozinha é o risco.

A fila de decisões

O bloqueio mais comum em produto de IA não é código: é decisão não tomada. Mantemos uma fila única, revisada toda semana no IPM, em que cada decisão tem **dono, prazo e default**, isto é, o que acontece se ninguém decidir até a data. Decidimos no último momento responsável, não no último possível. Decisões de arquitetura viram ADR; decisões de produto viram registro DP no repositório de documentação do projeto.

O Company Brain

Todo projeto tem um repositório canônico de conhecimento: um vault de notas ligadas entre si, versionado em Git, que chamamos de Company Brain. Nele vivem a árvore de valor, as hipóteses, os briefs, as decisões (DP e ADR), a fila de decisões, o glossário, as personas, os benchmarks, as referências teóricas e uma nota por reunião. A regra que o torna canônico é uma só: **tudo nasce como nota, e arquivos soltos são exportações**. Este documento, um README, uma proposta, um ADR no repositório de código: cada um é gerado a partir das notas, e quando diverge delas, é o arquivo que está errado.

Cada nota segue um esquema fixo: tipo, status, data de atualização, as fontes de onde veio e uma seção de evidências que aponta para a frase exata da fonte. O que não tem fonte é hipótese e fica marcado como tal. As decisões têm código, dono e data. O vault é um grafo: uma decisão liga o conceito que ela muda, a reunião em que foi tomada, a spec que a implementa e a fonte que a motivou, e a pergunta “por que isto é assim?” tem um caminho de cliques até a resposta. Template da nota no Anexo C.

O Company Brain faz três coisas na metodologia. É a **memória que sobrevive às pessoas**: em times de cinco, o conhecimento que só existe na cabeça de alguém é o risco maior, e a nota é a forma de tirá-lo de lá. É o **contexto que os agentes de IA leem**: o Product Agent sintetiza reuniões em notas, o Data & Ontology Agent mantém o glossário e os contratos, e todos respondem a partir do que está registrado, com fonte, e não do que o modelo supõe; uma alucinação encontra um lugar onde ser conferida. E é a **trilha que a diligência pede**: quem audita a propriedade intelectual encontra cada decisão com autoria, data, alternativas e a evidência que a sustenta, sem que ninguém precise reconstruir a história de memória.

Duas regras de fronteira. O vault é do projeto e vive no ambiente do projeto; dado pessoal do cliente entra pseudonimizado, como em qualquer outro artefato. E o vault não é o produto: o que sai para o cliente continua sendo código, dado e documentação exportada, sem o repositório de trabalho dos agentes.

Como medimos o Company Brain

Um vault que ninguém mede vira pasta de arquivos em seis meses. Medimos o Company Brain em dois eixos, e os números entram na retrospectiva como qualquer outra métrica de entrega.

O primeiro eixo é **organização**: quão bem o cérebro está construído. Usa os três níveis da Parte 8. No nível 1 existe esquema (pastas, tipos, templates), as notas têm frontmatter e se ligam, e não há órfãs. No nível 2 toda nota tem fonte e a seção de evidências aponta para a frase citada, as decisões têm código, dono e data, os status são mantidos vivos, e as métricas do vault (percentual com fonte, percentual com evidência, links quebrados, órfãs, frescor por pasta) são revisadas na retro. No nível 3 os agentes escrevem com um humano aprovando, as exportações são geradas por pipeline a partir das notas, e a rubrica de entendimento por produto alimenta os gates.

O segundo eixo é **cobertura da jornada do produto**: que etapas o cérebro registra com densidade. As etapas são sete: tese e mercado; discovery (domínio, ontologia, decisões); arquitetura; construção (specs, ADRs, contratos); verificação (evals, showcases, testes); piloto e operação (runbooks, SLOs, incidentes); aprendizado (retrospectivas, vereditos de gate). Um cérebro maduro tem notas nas sete, e a proporção entre elas diz em que fase o projeto realmente está, independentemente do que o cronograma afirma.

O AI Orchestrator

Usamos agentes de IA como parte do método: um **Product Agent** (pesquisa, síntese de entrevistas, backlog, documentação), um **Data & Ontology Agent** (profiling, mapeamento de fontes, contratos, qualidade), um **Model & Evaluation Agent** (experimentos, geração de casos, avaliação, comparação) e um **Engineering Agent** (código assistido, testes, revisão, análise de impacto). Três regras não negociáveis:

- 01 **Cada agente tem um humano que aprova.** O CPO aprova tese, prioridade e release; o Data Product Lead aprova semântica; o Data Scientist aprova método, thresholds e publicação; o engenheiro responsável revisa, testa e autoriza merge e deploy.
- 02 **A saída do agente passa pelo mesmo pipeline que a de qualquer pessoa.** Não há atalho.
- 03 **O agente nunca é dependência do produto entregue.** O que sai para o cliente é código, dado e documentação, sem instruções, hooks ou artefatos de agente.

agentes · parte do método

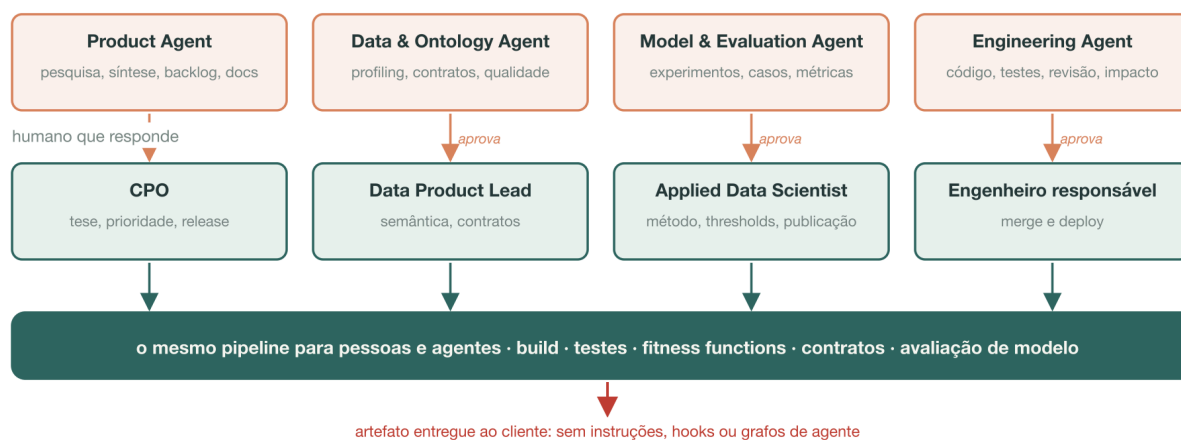


Figura 4 · O AI Orchestrator: cada agente tem um humano que aprova, e todos passam pelo mesmo pipeline

A evidência sobre IA no trabalho de conhecimento é clara nos dois sentidos. Há ganho real de produtividade e qualidade em tarefas dentro da fronteira de competência do modelo, e perda de qualidade quando a tarefa está fora dela e o humano confia sem verificar: a “fronteira irregular”, medida em experimento controlado com consultores. Em código, os estudos sobre geração assistida mostram aceleração mensurável e, ao mesmo tempo, expectativa acima da experiência quando quem desenvolve não revisa o que recebe. A regra 1 existe por causa desses achados.

Engenharia e arquitetura emergente

Arquitetura é o que é difícil de mudar

Adotamos a definição de Martin Fowler: arquitetura é o conjunto de decisões que são difíceis de mudar depois, “as coisas importantes, sejam elas quais forem”. O trabalho de arquitetura, portanto, não é desenhar o sistema inteiro antes; é reduzir o número de coisas difíceis de mudar e proteger as poucas que restam. Isso tem duas consequências diretas para produtos de IA.

A primeira é que o componente de menor meia-vida do sistema é o modelo, seguido do prompt, do provedor e do formato do dado. Uma arquitetura planejada em torno de um modelo específico está obsoleta antes de entrar em produção. O MIT xPRO formula isso como a decisão D7: arquitetura flexível para trocar modelos, dados e componentes. A segunda é que, em sistemas de ML, as fronteiras entre módulos erodem por construção: dado de um lado vira feature do outro, e o acoplamento aparece onde ninguém desenhou. Fronteiras não se mantêm por desenho; mantêm-se por teste.

Por que emergente

Trabalhamos com design emergente e arquitetura evolutiva. **Emergente** não significa ausência de desenho: significa desenho feito continuamente, sobre código que funciona, sob a disciplina de testes e refatoração. É a posição de Fowler em *Is Design Dead?*. **Evolutiva** significa que a arquitetura tem um mecanismo explícito de mudança guiada e incremental ao longo de múltiplas dimensões, com fitness functions como guia, a posição que a Thoughtworks consolidou em *Building Evolutionary Architectures*.

A base teórica é mais antiga que os nomes. Parnas mostrou, nos anos 1970, que a decomposição em módulos deve seguir as decisões de projeto que têm maior probabilidade de mudar (esconder cada uma dentro de um módulo), e não o fluxo de processamento. A escola de modularidade do MIT formalizou isso como um portfólio de opções reais: cada fronteira bem colocada é uma opção de substituir o módulo sem tocar no resto, e o valor da arquitetura é o valor dessas opções. Estudos empíricos posteriores mediram que sistemas mais modulares evoluem mais e sobrevivem mais. E as leis da evolução de software já diziam: um sistema em uso ou evolui ou se torna progressivamente menos útil, e sua complexidade cresce a menos que haja trabalho deliberado para contê-la.

Para um produto cujo núcleo será substituído várias vezes (modelo, provedor, estratégia de retrieval), esse é o único enquadramento compatível com a realidade. Planejar a arquitetura para durar é planejar para perder as opções.

As práticas

PRÁTICA	O QUE FAZEMOS
Decisões registradas	Cada decisão de arquitetura vira um ADR no momento em que é tomada, com contexto, alternativas e consequências. A arquitetura é o conjunto dessas decisões
Fitness functions	Cada característica arquitetural que importa (isolamento por tenant, privacidade, explicabilidade, fronteiras de módulo, latência, custo) é protegida por um teste automatizado que roda no pipeline. A lista é revisada a cada gate, no formato de cenários de atributo de qualidade
Fronteiras por domínio	Módulos seguem o domínio, não a tecnologia: linguagem ubíqua, contextos delimitados, e portas e adaptadores isolando o núcleo do que muda (modelo, banco, interface)
Monólito modular primeiro	Começamos com um sistema implantável por contexto, com fronteiras internas explícitas e testadas. Extraímos serviços quando a fronteira já provou que é estável, nunca antes
Arquitetura sacrificial	Partes do sistema são construídas sabendo que serão descartadas quando o aprendizado chegar: a primeira esteira de OCR, o primeiro provedor de embedding. Descartar é sucesso, não falha
Estrangulamento, não reescrita	Sistemas legados são substituídos por partes, com a nova implementação crescendo em volta da antiga até ela poder ser desligada
Refatoração contínua	Refatorar é parte de toda história, não um projeto. A hipótese da resistência de design: o esforço em design interno paga-se em semanas, não em anos
Esqueleto ambulante	Toda capacidade nova começa pela implementação mais fina que atravessa todas as camadas com um caso real. Só depois aprofundamos. O risco maior em sistemas de dados é integração, e ele é atacado primeiro
Contratos entre sistemas	Sistemas que conversam têm contrato versionado: schema, exemplo e teste executável nos dois lados. A quebra aparece no pipeline de quem consome, não em produção
Integração e entrega contínuas	Todo commit passa por build, testes, análise estática e segurança. O pipeline é a espinha dorsal da arquitetura: é ele que torna a mudança barata
Observabilidade desde o primeiro deploy	Logs estruturados, métricas, traces e SLOs; custo por tenant. Perguntas novas sobre o sistema em produção sem código novo

Dado como produto

Cada tabela, evento ou modelo publicado é um produto de dados: tem dono, contrato, qualidade medida, linhagem e SLO. Qualidade de dado é definida pela ótica de quem consome (acurácia, completude, atualidade, consistência, credibilidade) e não pela do pipeline. Datasets que treinam ou avaliam modelos têm datasheet. Onde o produto é mineração de processos, o contrato mínimo do event log (caso, atividade, timestamp) segue o Process Mining Manifesto.

Modelos e sistemas compostos

Os sistemas de IA que construímos são compostos: modelo, retrieval, ferramentas, regras, dado proprietário e avaliação, e o desempenho vem da composição, não do modelo-base. Modelos seguem o mesmo pipeline que código, com três acréscimos:

1. Dado, modelo, prompt e eval set têm versão, e a versão está no artefato publicado.
2. A promoção é por avaliação, não por opinião. Um modelo sobe de ambiente quando passa no eval set versionado com o threshold aprovado e registrado pelo Data Scientist.
3. Model card para cada modelo em produção: uso pretendido, dados, métricas por segmento, limitações conhecidas.

O ML Test Score é a rubrica que usamos para saber se um sistema de ML está pronto para produção: testes de dado, de modelo, de infraestrutura e de monitoramento. A avaliação de sistemas generativos segue a disciplina de evals com conjunto de referência, casos adversos e métricas por tarefa.

Caminho até a produção, inclusive a entrega ao cliente

Mapeamos o caminho de um commit até a mão do cliente e automatizamos cada passo. Isso inclui a transferência de repositório: quando o contrato prevê entrega de código, o pipeline produz um artefato limpo (sem instruções ou artefatos de agentes de IA, sem segredos, com histórico conforme o contrato) e o publica no repositório do cliente. Não é operação manual, e não depende de quem está de plantão.

Métricas de entrega

Medimos as quatro métricas DORA por repositório desde a primeira semana: frequência de deploy, lead time de mudança, taxa de falha de mudança e tempo de restauração. Ao lado da métrica de produto mais concreta de cada projeto, são o baseline sem o qual o gate é opinião.

Governança, risco e conformidade

Decision rights

TEMA	CLIENTE	SCIENSA	CONJUNTO
Metodologia e taxonomias	Aprova	Formaliza e implementa	Avalia impacto
Produto e prioridade	CPO / Product owner	Recomenda e executa	Roadmap e gates
Arquitetura e engenharia	Consulta	Accountable	Decisões de alto impacto
Segurança e privacidade	Define políticas e risco	Implementa controles	Aceite de risco
Modelos e thresholds	Valida domínio	Constrói e avalia	Aprova produção
Clientes-piloto	Relacionamento e acesso	Onboarding e suporte técnico	Critérios de sucesso

Classes de risco de IA e seus controles

CLASSE	O QUE É	CONTROLE
Adversarial	Entradas que exploram fragilidades e alteram o comportamento	Casos adversos no eval set; quarentena de entrada; limites de ação por degrau de autonomia
Generalização	Funciona na distribuição observada e degrada fora dela	Métricas por segmento na model card; monitoramento de drift; gate de cobertura antes de escalar
Viés	Resultado sistematicamente enviesado por dado, objetivo ou implementação	Avaliação por grupo; Model & Data Council mensal; rótulo humano com autoria e papel
Falhas generativas	Alucinação, conflito de fontes, incerteza mal comunicada	Groundedness medido; estados de incerteza na interface; fallback e escalonamento humano

A taxonomia segue o MIT xPRO, estendida pelas taxonomias de risco de modelos de linguagem e alinhada ao NIST AI RMF. O que garante que os controles existem, e não apenas constam, é uma auditoria interna de ponta a ponta: escopo, mapeamento, coleta de artefatos, teste e reflexão, repetida a cada gate. Datasets têm a sua própria trilha de responsabilidade, do requisito à publicação.

Privacidade e prontidão para diligência

Privacidade é decisão de modelagem, não retrofit: em produto de dado comportamental, retrofit é reescrita. Adotamos as oito estratégias de design de privacidade: minimizar, separar, agregar, ocultar, informar, controlar, impor, demonstrar, e os sete princípios de Privacy by Design. “Demonstrar” é a estratégia que interessa à diligência: o pipeline, as fitness functions, os ADRs e as model cards são a evidência que um comprador ou investidor pede, produzida como subproduto do trabalho e não como projeto à parte. O Company Brain é onde essa evidência fica ligada: decisão, alternativa, fonte e autoria num grafo que se navega, não numa pasta que se procura. Isolamento por tenant é testado, não convencionado.

Explicabilidade

Toda inferência sobre uma pessoa é rastreável até a evidência que a sustenta: origem, método, versão. A explicação é desenhada para quem a recebe, não para quem construiu o modelo. A pesquisa em ciências sociais sobre explicação mostra que explicações úteis são contrastivas, seletivas e conversacionais; a definição do que conta como interpretável precisa ser explícita e testável. Ferramentas como LIME e SHAP entram como instrumento, não como resposta.

Guardrails dos agentes de IA

Menor privilégio, segregação por cliente, logs completos, versionamento de prompts e modelos, avaliação antes de produção, revisão humana obrigatória e bloqueio de ações externas não autorizadas. O panorama de diretrizes de ética em IA converge nesses pontos; a diferença entre aderir e declarar está na Parte 5: cada guardrail é uma fitness function ou um controle de pipeline.

Pessoas e agentes

Os cinco papéis do núcleo High Impact

PAPEL	ACCOUNTABILITY	O QUE APROVA
CPO / Product Executive	Visão da categoria, estratégia, priorização, economics, validação de mercado	Tese, prioridade, release, business case
Product Designer / UX	Experiência das decisões, jornadas, usabilidade, coerência visual	Escada de autonomia, estados de incerteza, jornadas críticas
Data Product Lead	Transformar a metodologia do cliente em produtos de dados governados	Semântica, contratos, taxonomias, métricas
Principal Data Engineer	Disponibilidade, qualidade, segurança e operação da fundação de dados	Execução de pipelines, linhagem, catálogo
Applied Data Scientist	Desempenho, avaliação e explicabilidade dos modelos	Método, thresholds, publicação de modelo

A composição privilegia profissionais principal e staff, capazes de tomar decisões e produzir entregáveis diretamente. Não há camada de coordenação entre eles e o trabalho.

Enablement, ativado conforme a entrega

Principal Architect / AI Platform, Product Engineer full-stack, Endpoint / Systems Engineer, Knowledge Engineer, DevSecOps / SRE e QA Automation, de 0,2 a 2,0 FTE conforme o estágio da onda. O Architect é o guardião das fitness functions; o Knowledge Engineer, das perguntas de competência da ontologia; o DevSecOps, do caminho até a produção.

Agentes

Product Agent, Data & Ontology Agent, Model & Evaluation Agent e Engineering Agent, cada um pareado com o humano que aprova a sua saída (Parte 4). São parte do método de entrega e não aparecem no produto.

Papéis do cliente

Sponsor executivo com poder de remover bloqueios e decidir prioridades; Domain Lead disponível diariamente para externalizar método e validar resultados; responsável por taxonomias e critérios de impacto; responsável por economics; acesso a cliente-piloto ou dataset representativo; políticas corporativas e regras de captura aprovadas para implementação. Estes papéis são condição, e sem eles o método não sai da cabeça de alguém e o cronograma não fecha.

Adoção e maturidade

Como um projeto entra na metodologia

Um projeto está dentro da metodologia quando o checklist de kickoff (Anexo F) fecha: árvore de valor com ao menos uma hipótese escrita; Inception agendada; briefs das apostas da primeira onda; fila de decisões com donos; repositórios com pipeline e Definition of Done; fóruns com data no calendário; decision rights assinados; papéis do cliente nomeados. Nenhum item é opcional. Um projeto pode começar sem o checklist completo; não pode chegar ao primeiro showcase sem ele.

Níveis de maturidade

NÍVEL	PORTFÓLIO	PRODUTO DE IA	ENTREGA	ENGENHARIA	COMPANY BRAIN
1 • Praticando	Ondas e gates existem	Briefs existem	Fóruns acontecem na cadência	Pipeline verde; testes; ADRs	Esquema, templates e frontmatter; notas ligadas; sem órfãs
2 • Medindo	Hipóteses com medida e valor-alvo	Oito decisões registradas por aposta; classes de risco com controle	Lead time e WIP medidos; fila de decisões sem item vencido	DORA por repositório; fitness functions para cada NFR; contratos executáveis	Toda nota com fonte e evidência; decisões com código, dono e data; métricas do vault na retro
3 • Aprendendo	Gates com veredito medido; apostas encerradas por evidência	Model cards e evals versionados; decisões revisitadas em gate; arquitetura sacrificial exercida	Retrospectiva com ação cumprida; showcase nunca sem dado real	Promoção de modelo por avaliação; caminho até a produção automatizado até o repositório do cliente	Agentes escrevem com aprovação; exportações geradas por pipeline; rubrica de entendimento alimenta os gates; as sete etapas da jornada com notas

O que a versão 2026 fixa

- O backbone do MIT xPRO como camada de produto de IA, com as oito decisões como registro obrigatório de cada aposta.
- Arquitetura emergente e evolutiva como posição, com fitness functions como mecanismo e modularidade como portfólio de opções.
- Agentes de IA como parte do método, com humano que aprova, e a regra de que não entram no produto entregue.
- Fitness functions e trilha de decisão como resposta à diligência: evidência produzida pelo trabalho, não por projeto à parte.
- A regra do gate: o gate avalia evidência, não capacidade.

Como esta metodologia muda

Ela é versionada como o resto. Cada mudança é um ADR do método, discutido na retrospectiva e adotado por onda. A versão 2027 nasce do que os próximos projetos ensinarem, e o registro dessas lições começa no primeiro showcase.

// ANEXOS

Anexos

Templates A a G · Apêndice H · Guia de Cultura de Engenharia (uso interno) · glossário · base científica · nível de confiança



Anexo A · AI Product Brief

Uma página por aposta. Preenchido na Inception, revisado em cada gate das seis fases.

BLOCO	PERGUNTA	DECISÃO MIT
Problema	Qual problema real existe, para quem e em qual contexto?	
Intelligence	Qual comportamento inteligente queremos criar?	
Metrics	Como saberemos se esse comportamento é bom? Métrica de modelo e de uso	D1
Scope	Onde deve funcionar e quais casos ficam fora?	D2
Business process	Qual workflow muda e qual meta operacional melhora? Quem age antes e depois?	D4
Strategic role	A IA melhora o produto, resolve uma parcela maior do problema ou cria efeito de sistema?	D3
Technology	Build, buy, adapt ou compose? Qual stack e composição?	D5
Data	Que dados e contexto sustentam a solução? Quem é dono? Qual feedback?	D6
Human + AI	Quem sente, lembra, cria, decide, age e aprende? Degrau de autonomia de cada ação	HCI / Superminds
Failure	Quais falhas são aceitáveis, quais exigem fallback ou escalonamento? Classe de risco	D8
Business case	Qual valor, custo, risco e roadmap?	Lawler / business case
Next experiment	Qual o menor experimento que reduz a maior incerteza?	D7

Anexo B • AI Fit Canvas

DIMENSÃO	PERGUNTA
Natureza da tarefa	Classificar, prever, detectar, gerar, recomendar, buscar, planejar ou executar?
Variabilidade	O problema é difícil de resolver com regras determinísticas?
Dados e contexto	Existe informação suficiente, autorizada e atualizada?
Erro tolerável	Que tipo de erro é aceitável? Qual exige humano?
Latência	Qual tempo de resposta é necessário?
Explicabilidade	Que justificativa o usuário ou o regulador precisa?
Custo	Qual custo por uso, tarefa ou usuário é sustentável?
Integrações	O sistema precisa consultar dados ou agir em ferramentas externas?
Reversibilidade	A ação pode ser desfeita?
Observabilidade	Saberemos quando a solução está degradando?

Anexo C · Fichas de hipótese, de decisão e de nota do Company Brain

HIPÓTESE

CAMPO	CONTEÚDO
Acreditamos que	capacidade, para quem, resultado
Saberemos quando	medida observável, valor, data
Camada de avaliação	modelo / experiência / operação / negócio
Se não for	o que fazemos
Onda e gate	em que gate o veredito é dado

DECISÃO

CAMPO	CONTEÚDO
Código	ADR-nnn (arquitetura) ou DP-nnn (produto)
Pergunta	uma frase
Dono	uma pessoa
Prazo	data
Default	o que acontece se não for decidida até o prazo
Decisão MIT	D1 a D8, quando aplicável
Status	aberta / provisória (revisitar em gate) / tomada

NOTA DO COMPANY BRAIN

CAMPO	CONTEÚDO
type	plataforma, produto, decisao, conceito, pessoa, benchmark, referencia, reuniao, fonte
status	rascunho, hipótese, ativo, superado
fontes	as notas de fonte (F) de onde a nota veio
atualizado	data da última revisão
Em uma frase	o que a nota afirma, numa frase
Corpo	o conteúdo, com número, nome, data e arquivo
Relações	pertence a, depende de, relacionado, contrasta com
Evidências nas fontes	a frase exata da fonte que sustenta cada afirmação
Em aberto	o que ainda não se sabe, e quem decide

Anexo D • Definition of Done

Uma história está pronta quando, e somente quando:

1. tem teste automatizado que falharia se o comportamento sumisse;
2. está integrada no tronco, e o pipeline está verde;
3. as fitness functions do repositório passam;
4. está implantada no ambiente da onda;
5. é observável: log, métrica ou trace que mostra que está sendo usada;
6. passou por desk check com quem participou do kick-off;
7. se houve decisão de arquitetura ou de produto, ela está registrada;
8. se toca contrato entre sistemas, o teste de contrato passa nos dois lados.

Anexo E • Checklist de fitness functions

Para cada requisito não funcional declarado no projeto, uma linha. Revisado a cada gate.

REQUISITO	FITNESS FUNCTION	ONDE RODA	BLOQUEIA?
Privacidade desde a modelagem	Lint de PII em código e contratos; teste de minimização; retenção verificada por migração	Pipeline	Sim
Isolamento por tenant	Teste de integração que tenta cruzar tenants e precisa falhar	Pipeline	Sim
Explicabilidade	Teste que rejeita recomendação sem origem, método e versão	Pipeline	Sim
Bilíngue estrutural	Lint de string fora de recurso de i18n	Pipeline	Sim
Fronteiras de módulo	Regras de dependência (ArchUnit ou equivalente)	Pipeline	Sim
Latência e tempo de pipeline	Verificação contra SLO no ambiente da onda	Pós-deploy	Alerta
Custo por tenant	Orçamento com alerta	Contínuo	Alerta
Segredos	Scan de segredos no commit	Pipeline	Sim
Artefatos de agente de IA	Verificação de que instruções, hooks e grafos de agente não estão no artefato entregue	Pipeline de entrega	Sim

Anexo F • Checklist de kickoff

- ☐ Árvore de valor com visão, metas e ao menos uma aposta com hipótese escrita
- ☐ Lean Inception agendada, com cliente presente
- ☐ AI Product Brief das apostas da primeira onda
- ☐ Fila de decisões com dono, prazo e default para cada item
- ☐ Repositórios com pipeline, Definition of Done e ADR inicial
- ☐ Fóruns no calendário: stand-up, IPM, showcase, retro, council, steering, gate
- ☐ Decision rights assinados
- ☐ Papéis do cliente nomeados: sponsor, Domain Lead, dono das taxonomias, dono do economics, acesso a piloto
- ☐ Métricas DORA e métrica de produto com baseline na semana 1

Anexo G • Agenda de Lean Inception compacta (3 dias)

DIA	MANHÃ	TARDE
1	Visão do produto • É / Não é / Faz / Não faz • Objetivos	Personas • Jornadas
2	Brainstorm de funcionalidades • Revisão técnica, de UX e de negócio	Sequenciador • Hipóteses da onda
3	Canvas de MVP • AI Product Brief por aposta	Fila de decisões • Showcase da Inception ao Steering

Guia de Cultura de Engenharia

Este apêndice é o guia interno. O corpo do documento diz o que fazemos e por quê; este apêndice diz como, no dia a dia, e o que fazemos quando alguém, inclusive o cliente, pede para pular. Serve de referência para quem entra na Sciensa e de régua para quem já está.

H.1 • Os cinco valores

VALOR	O QUE SIGNIFICA
Parar a linha	Pipeline vermelho é a prioridade de todo mundo. Quem quebrou conserta ou reverte em dez minutos; ninguém integra sobre vermelho. É o jidoka da produção enxuta: o defeito para a linha para não se propagar
Sem janelas quebradas	Um teste ignorado, um warning tolerado, um TODO sem dono: cada um sinaliza que o próximo também pode ficar. Corrigimos o pequeno antes que vire o normal
Você constrói, você opera	Quem escreve o código responde por ele em produção. Não há time de “sustentação” para quem entregou
Erro é informação	Postmortem sem culpa em até 48 horas de qualquer incidente, com ao menos uma ação com dono. Culpar a pessoa esconde a causa; entender o sistema a expõe
Simple primeiro	Passa nos testes, revela intenção, não duplica, tem o menor número de elementos, nessa ordem. O resto é especulação

H.2 • Os defaults de código

Um projeto pode se desviar de um default. Registra a razão em ADR.

DEFAULT	REGRA
Tronco	Uma branch principal. Branches de trabalho vivem horas, no máximo um dia. Sem force-push em branch compartilhada
Commits pequenos, integração diária	Cada pessoa integra no tronco ao menos uma vez por dia. Trabalho que não cabe num dia é fatiado, não escondido numa branch
Teste primeiro	Teste de unidade e de contrato escritos antes ou junto do código. A pirâmide: muitos testes de unidade, alguns de integração e contrato, poucos de ponta a ponta, e fitness functions atravessando os três
Revisão em horas	Pull request pequeno, revisado em menos de quatro horas úteis. Revisão moderna é sobre entendimento compartilhado e transferência de conhecimento, tanto quanto sobre defeito

DEFAULT	REGRA
Refatoração dentro da história	Não existe “história de refatoração”. A regra do escoteiro: o código fica um pouco melhor do que foi encontrado
Feature toggles	Deploy contínuo; exposição por decisão de produto. Toggle tem dono e data de remoção
Infraestrutura como código	Nenhum ambiente configurado à mão. Se não está no repositório, não existe
Segredo nunca no repositório	Scan de segredos no commit e no pipeline, bloqueante. Rotação documentada
Observável antes de pronto	Log estruturado, métrica e trace são parte da história, não da próxima
ADR no momento da decisão	Uma decisão de arquitetura tomada e não registrada é uma decisão que vai ser tomada de novo

H.3 • Como pareamos

- Par por padrão em arquitetura, integração entre sistemas, segurança, e em qualquer tarefa que só uma pessoa sabe fazer.
- Rotação de par a cada história ou a cada dois dias, o que vier primeiro.
- Ensemble (grupo inteiro) para o esqueleto ambulante e para incidentes.
- Par remoto é par: tela compartilhada, driver e navigator alternando a cada 25 minutos.
- A pesquisa é honesta sobre o custo: pareamos pela transferência de conhecimento e pela redução do fator-ônibus, e sabemos que gastamos mais horas-pessoa por isso.

H.4 • Como usamos agentes de IA no código

1. **O agente escreve; a pessoa responde.** Todo código gerado é lido, entendido e testado por quem autoriza o merge. Se não consegue explicar, não integra.
2. **Teste primeiro vale para o agente também.** Pedimos o teste antes do código, ou escrevemos o teste nós mesmos.
3. **Tudo pelo pipeline.** O agente não faz merge, não faz deploy, não toca em produção.
4. **Prompts e instruções de projeto são versionados,** e ficam fora do artefato entregue ao cliente (fitness function no Anexo E).
5. **Frente irregular:** o agente acelera o que já sabemos fazer e engana no que não sabemos. Tarefa nova para o time é tarefa em par com humano no volante.
6. **Sem agente em decisão de arquitetura.** O agente propõe alternativas; o ADR é escrito e assinado por pessoa.

H.5 · Quando o pipeline quebra, quando produção quebra

- **Pipeline vermelho:** quem quebrou tem dez minutos para consertar ou reverter. Passou, reverte. Ninguém integra em cima.
- **Incidente em produção:** quem está de plantão conduz; comunica em canal aberto; restaura primeiro, investiga depois.
- **Postmortem em 48 horas,** sem culpa, com linha do tempo, causas contribuintes e ações com dono e prazo. Publicado para o time inteiro.
- **SLO e orçamento de erro:** quando o orçamento acaba, a próxima iteração é de confiabilidade, decisão de produto tomada com o CPO.

H.6 · Dívida técnica

- Toda dívida deliberada é um ADR com data de revisão. No quadrante de dívida técnica de Fowler, prudente e deliberada é a única aceitável.
- Dívida inadvertida descoberta vira item com dono na retro.
- Não existe “sprint de dívida”. Existe refatoração dentro da história e uma linha de orçamento de confiabilidade quando o SLO pede.
- Métrica: tempo de ciclo por história ao longo das ondas. Se sobe, a dívida está cobrando.

H.7 · Métricas que olhamos toda quinzena

MÉTRICA	FONTE	O QUE FAZEMOS COM ELA
Frequência de deploy	Pipeline	Se cai, algo ficou grande demais
Lead time de mudança	Commit → produção	Se sobe, revisão ou pipeline estão lentos
Taxa de falha de mudança	Incidentes / deploys	Se sobe, testes ou toggles estão faltando
Tempo de restauração	Incidente → restaurado	Se sobe, observabilidade ou rollback estão fracos
WIP e tempo de ciclo por história	Quadro	Se WIP sobe, o par está quebrado
Itens vencidos na fila de decisões	Fila	Se maior que zero, o IPM falhou

H.8 • Quando pedem para pular

- **“Não precisa de teste, é só um ajuste.”** O teste é o que permite o próximo ajuste ser “só um ajuste”. Escrevemos.
- **“Deixa o pipeline vermelho, depois a gente vê.”** Depois não existe. Revertemos.
- **“O cliente quer a demo com slide.”** Mostramos software rodando, com dado real, e deixamos o slide para a abertura. Se o software não roda, é isso que o cliente precisa saber.
- **“Manda o repositório como está.”** O pipeline de entrega roda a fitness function de artefatos e segredos. Se falhar, não sai.
- **“O agente já testou.”** O agente não responde por nada. A pessoa que autoriza o merge responde.

H.9 • Onboarding

Primeiro commit em produção na primeira semana, por menor que seja. É o esqueleto ambulante aplicado a pessoas: prova que o caminho até a produção funciona para quem acabou de chegar, e que nada nele depende de conhecimento tácito.

Anexo I · Glossário

TERMO	SIGNIFICADO NESTE DOCUMENTO
ADR	Architecture Decision Record, registro de uma decisão de arquitetura com contexto, alternativas e consequências
AI Product Brief	Documento de uma página por aposta com as oito decisões do backbone MIT
Aposta	Iniciativa de produto formulada como hipótese com medida de sucesso
Árvore de valor	Visão → metas → apostas → iniciativas; substitui o backlog plano
Desk check	Demonstração da história rodando antes de fechá-la
DORA	As quatro métricas de desempenho de entrega: frequência de deploy, lead time, taxa de falha, tempo de restauração
DP	Decisão de produto registrada
Escada de autonomia	Seis degraus de agência da IA: informar, sugerir, recomendar, preparar, executar com supervisão, executar autonomamente
Esqueleto ambulante	A implementação mais fina que atravessa todas as camadas com um caso real
Eval set	Conjunto versionado de casos com resposta esperada, usado para avaliar e promover modelos
Fitness function	Teste automatizado que protege uma característica arquitetural
Gate	Decisão de portfólio no fim de uma onda: go, pivot, pause ou stop
IPM	Iteration Planning Meeting, planejamento semanal, onde a fila de decisões é revisada
Model card	Ficha de um modelo em produção: uso pretendido, dados, métricas por segmento, limitações
Onda	Janela de entrega de três a seis meses, com hipóteses na abertura e gate no fechamento
Product Studio	Modelo de atuação: time misto, decisões compartilhadas, accountability Sciensa sobre engenharia e entrega
Showcase	Demonstração quinzenal de software rodando com dado real
Story kick-off	Conversa de quinze minutos antes de começar uma história sobre o que “pronto” significa
WIP	Work in progress, trabalho em andamento; limitado por par

Anexo J · Base científica

O corpo do documento diz o que fazemos. Este anexo diz em que evidência cada prática se apoia e, quando a evidência é modesta ou contrária, diz isso também. O nível de confiança de cada fonte está no Anexo K.

PARTE O · POR QUE TRABALHAMOS ASSIM

AFIRMAÇÃO OU PRINCÍPIO	O QUE A EVIDÊNCIA DIZ	FONTES
A qualidade de sistemas de ML não é binária; a dívida mora em dado, configuração e ciclos de feedback	Sculley et al. descrevem a dívida técnica específica de ML: fronteiras que erodem, dependências de dado, feedback loops ocultos, configuração. Breck et al. propõem uma rubrica de 28 testes em quatro áreas (dado, modelo, infraestrutura, monitoramento)	Sculley et al. (2015); Breck et al. (2017)
O produto de IA é uma mudança de workflow, não um modelo	Estudo de caso na Microsoft com mais de 500 engenheiros: os maiores desafios estão em dado, integração e operação. Survey de casos de deploy confirma o padrão	Amershi et al. (2019a); Paleyes, Urma & Lawrence (2022)
Alguém responde por cada inferência	Framework de auditoria interna de ponta a ponta (SMACTR); modelo de tipos e níveis de automação como escolha de projeto, não herança da ferramenta	Raji et al. (2020); Parasuraman, Sheridan & Wickens (2000)
“Ágil” como rótulo não é evidência	Revisão sistemática de 33 estudos: evidência fragmentada e de qualidade desigual; conclusões só fazem sentido prática por prática	Dybå & Dingsøyr (2008)
P1 · Software rodando é a única prova	Entrega contínua correlaciona com desempenho de TI e organizacional em surveys de larga escala (State of DevOps)	Humble & Farley (2010); Forsgren & Humble (2016); Forsgren, Humble & Kim (2018)
P2 · Hipótese antes de escopo	Modelo RIGHT de experimentação contínua; experimentos controlados exigem métrica definida antes do experimento; o plano de negócio como lista de hipóteses	Fagerholm et al. (2017); Kohavi et al. (2009); Blank (2013)
P3 · Fatia fina, ponta a ponta, primeiro	Modelo espiral: atacar o maior risco primeiro; walking skeleton como primeira entrega	Boehm (1988); Cockburn (2004)
P4 · Decisão com dono, prazo e default	Decidir no último momento responsável (lean software development)	Poppendieck & Poppendieck (2003)

AFIRMAÇÃO OU PRINCÍPIO	O QUE A EVIDÊNCIA DIZ	FONTES
P5 • Humano responsável por toda inferência	Níveis de automação; HCAI: alta automação e alto controle humano não são opostos num eixo; auditoria interna	Parasuraman et al. (2000); Shneiderman (2020); Raji et al. (2020)
P6 • Menos, bem construído, documentado	Arquitetura como conjunto de decisões de projeto; dívida técnica da metáfora à teoria	Jansen & Bosch (2005); Kruchten, Nord & Ozkaya (2012)
P7 • Tudo versionado, tudo observável	Arquitetura de referência MLOps; ML Test Score	Kreuzberger, Kühn & Hirschl (2023); Breck et al. (2017)

PARTE 1 • O MODELO EM QUATRO CAMADAS

AFIRMAÇÃO	O QUE A EVIDÊNCIA DIZ	FONTES
Gates no portfólio, iteração no time	Stage-Gate original; revisão do que vem depois; o híbrido Agile-Stage-Gate documentado em empresas industriais, com ganhos de velocidade e adaptação sem perder governança	Cooper (1990); Cooper (2014); Cooper & Sommer (2016)

PARTE 2 • PORTFÓLIO

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Árvore de valor e governança leve	EDGE (Thoughtworks): visão, metas, apostas, iniciativas; financiar resultado, não output	Highsmith, Luu & Robinson (2019)
Formato de hipótese	Sistemas de experimentação; fechar o “open loop” entre opinião e dado; modelo RIGHT; experimentos controlados na web	Bosch (2012); Olsson & Bosch (2014); Fagerholm et al. (2017); Kohavi et al. (2009)
Calibração de confiança como medida de modelo	Redes modernas são mal calibradas; a calibração é mensurável e corrigível	Guo et al. (2017)
Avaliação em quatro camadas (não só modelo)	Explicações de IA podem aumentar a confiança do usuário sem aumentar o acerto do time humano + IA; por isso medir experiência e resultado	Bansal et al. (2021)

PARTE 3 • PRODUTO DE IA

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Backbone: quatro estágios, oito decisões	Programa MIT xPRO; o Delta Model como base da decisão de papel estratégico	MIT xPRO; Hax & Wilde (1999)
Comportamento e métrica antes de tecnologia	Projetos que chegam à produção diferem dos que não chegam em dado e integração, não em algoritmo	Amershi et al. (2019a); Paleyes et al. (2022)
Sistema sociotécnico, não modelo	Cinco armadilhas de abstração (enquadramento, portabilidade, formalismo, efeito cascata, solucionismo); engenharia de sistemas sociotécnicos	Selbst et al. (2019); Baxter & Sommerville (2011)
Superminds e Allocation Map	Inteligência coletiva como configuração de pessoas e máquinas: sense, remember, create, decide, learn	Malone & Bernstein (2015); Malone (2018)
Escada de autonomia	Modelo de dez níveis de automação por estágio de processamento; HCAI	Parasuraman, Sheridan & Wickens (2000); Shneiderman (2020)
Escolha do degrau segue diretrizes de HCI	Dezoito diretrizes de interação humano-IA validadas com 49 praticantes em 20 produtos	Amershi et al. (2019b)
Classes de risco	Taxonomia de 21 riscos em seis áreas para modelos de linguagem; NIST AI RMF	Weidinger et al. (2022); NIST (2023)
Datasheets e model cards	Documentação padronizada de datasets e de modelos	Geburu et al. (2021); Mitchell et al. (2019)
Fases por aposta, não cascata	Modelo espiral orientado a risco	Boehm (1988)

PARTE 4 • ENTREGA

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Time pequeno e sênior, sem camada de coordenação	Custo de comunicação cresce com o tamanho do time; times auto-organizados em desenvolvimento de produto	Brooks (1995); Takeuchi & Nonaka (1986)
Topologia do time é decisão de arquitetura	Lei de Conway; Team Topologies	Conway (1968); Skelton & Pais (2019)
Showcase e retro como lugar de dizer o que não deu certo	Segurança psicológica é o preditor mais forte de aprendizado em times (estudo com 51 times)	Edmondson (1999)

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Lean Inception	Prática documentada; descoberta em paralelo à entrega (origem do dual-track); design thinking	Caroli (2018); Sy (2007); Brown (2008)
Limite de WIP	Lei de Little; revisão sistemática de Kanban em software: redução de lead time e ganho de visibilidade	Little (1961); Ahmad, Markkula & Oivo (2013)
Par por padrão, com honestidade sobre o custo	Meta-análise de 18 estudos: efeito pequeno em qualidade, médio em duração (mais rápido), médio negativo em esforço. Benefícios de transferência de conhecimento vêm de estudos qualitativos	Hannay et al. (2009)
Fila de decisões com default	Último momento responsável; ADR; arquitetura como decisões	Poppendieck & Poppendieck (2003); Nygard (2011); Jansen & Bosch (2005)
AI Orchestrator com humano que aprova	Experimento de campo com 758 consultores: dentro da fronteira de competência do modelo, +40% de qualidade; fora dela, 19 pontos percentuais a menos de acerto. Em código: aceleração mensurável e expectativa acima da experiência quando não há revisão	Dell'Acqua et al. (2023); Peng et al. (2023); Ziegler et al. (2022); Vaithilingam, Zhang & Glassman (2022)

PARTE 4 • COMPANY BRAIN

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Tudo nasce como nota	Conhecimento tácito só vira ativo da organização quando é externalizado e combinado (modelo SECI)	Nonaka (1994); Nonaka & Takeuchi (1995)
Repositório canônico, com esquema e dono	O valor de um sistema de gestão de conhecimento vem do processo e da cultura em volta dele, não da ferramenta; em engenharia de software, o ganho aparece em reúso, decisão e onboarding	Alavi & Leidner (2001); Rus & Lindvall (2002); Bjørnson & Dingsøyr (2008)
Decisão registrada no momento em que é tomada	Decisões de arquitetura não registradas “evaporam” e são retomadas; o registro leve com contexto, alternativas e consequências resolve	Jansen & Bosch (2005); Tyree & Akerman (2005)
Notas ligadas como memória externa	Memex; aumentar o intelecto humano com ferramentas de associação; hipertexto	Bush (1945); Engelbart (1962); Conklin (1987)
O vault como grafo de conhecimento	Grafos como forma de integrar conhecimento heterogêneo com proveniência	Hogan et al. (2021)

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Agentes respondem a partir do registrado	Recuperação de contexto antes da geração reduz respostas sem base	Lewis et al. (2020)

PARTE 5 • ENGENHARIA

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Arquitetura é o que é difícil de mudar	Coluna em IEEE Software	Fowler (2003)
Design emergente	<i>Is Design Dead?</i> ; design emergente como natureza do desenvolvimento profissional; as quatro regras de design simples	Fowler (2004); Bain (2008); Beck (2000)
Arquitetura evolutiva e fitness functions	Mudança guiada e incremental em múltiplas dimensões; governança automatizada	Ford, Parsons, Kua & Sadalage (2022); Ford et al. (2021)
Modularidade como portfólio de opções	Information hiding como critério de decomposição; Design Rules (MIT); modularidade como opções reais; teoria de opções aplicada a design de software; estudo empírico de código aberto e proprietário	Parnas (1972); Baldwin & Clark (2000); Sullivan et al. (2001); MacCormack, Rusnak & Baldwin (2006)
Sistemas em uso evoluem ou decaem	Leis da evolução de software	Lehman (1980)
Fronteiras de ML erodem por construção	Dívida técnica oculta em ML	Sculley et al. (2015)
Cenários de atributo de qualidade, não adjetivos	ATAM; Software Architecture in Practice	Kazman, Klein & Clements (2000); Bass, Clements & Kazman (2012)
Fronteiras por domínio; portas e adaptadores	DDD; arquitetura hexagonal	Evans (2003); Cockburn (2005)
Monólito modular primeiro; arquitetura sacrificial; estrangulamento; refatoração; resistência de design	Bliki e livros de Fowler	Fowler (2015); Fowler (2014); Fowler (2004, atualizado 2024); Fowler (2018); Fowler (2007)
Contratos entre sistemas	Consumer-driven contracts	Robinson (2006)

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Integração e entrega contínuas; DORA	CI; CD; correlação com desempenho; revisão sistemática de CI/CD; agenda de engenharia contínua	Fowler (2006); Humble & Farley (2010); Forsgren, Humble & Kim (2018); Shahin, Ali Babar & Zhu (2017); Fitzgerald & Stol (2017)
Observabilidade desde o primeiro deploy	Observability engineering; SRE	Majors, Fong-Jones & Miranda (2022); Beyer et al. (2016)
Dado como produto	Data Mesh; princípios do paradigma; qualidade de dado pela ótica do consumidor; datasheets	Dehghani (2022); Machado, Costa & Santos (2022); Wang & Strong (1996); Gebru et al. (2021)
Mineração de processos	Process Mining Manifesto: event log com caso, atividade e timestamp	van der Aalst et al. (2012)
Sistemas de IA compostos; RAG	O desempenho vem da composição, não do modelo-base	Zaharia et al. (2024); Lewis et al. (2020)
Entrega contínua para modelos; promoção por avaliação; evals	CD4ML; MLOps; ML Test Score; avaliação de LLMs; model cards	Sato, Wider & Windheuser (2019); Kreuzberger, Köhl & Hirschl (2023); Breck et al. (2017); Chang et al. (2024); Mitchell et al. (2019)

PARTE 6 • GOVERNANÇA, RISCO E CONFORMIDADE

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Classes de risco e controles	Taxonomia de riscos de LMs; NIST AI RMF	Weidinger et al. (2022); NIST (2023)
Auditoria interna de ponta a ponta; trilha de datasets	Framework SMACTR; responsabilidade sobre datasets do requisito à publicação	Raji et al. (2020); Hutchinson et al. (2021)
Privacidade por design	Oito estratégias de design; sete princípios fundacionais	Hoepman (2014); Cavoukian (2009)
Isolamento por tenant testado, não convencionado	Manutenção de SaaS multi-tenant	Bezemer & Zaidman (2010)

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Explicabilidade desenhada para quem recebe	Explicações úteis são contrastivas, seletivas e conversacionais; interpretabilidade precisa de definição testável; LIME; SHAP	Miller (2019); Doshi-Velez & Kim (2017); Ribeiro, Singh & Guestrin (2016); Lundberg & Lee (2017)
Guardrails convergentes	84 documentos de diretrizes de ética em IA convergem em transparência, justiça, não maleficência, responsabilidade e privacidade	Jobin, Ienca & Vayena (2019)

APÊNDICE H • CULTURA DE ENGENHARIA

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Parar a linha	Jidoka no sistema Toyota; lean software development	Ohno (1988); Poppendieck & Poppendieck (2003)
Sem janelas quebradas	Pragmatic Programmer; teoria das janelas quebradas	Hunt & Thomas (1999); Wilson & Kelling (1982)
Você constrói, você opera	Entrevista de Werner Vogels sobre o modelo da Amazon	Vogels (2006)
Erro é informação	Postmortems sem culpa; a “nova visão” do erro humano; segurança psicológica	Allspaw (2012); Dekker (2014); Edmondson (1999)
Simples primeiro	Quatro regras de design simples (XP)	Beck (2000)
Tronco, commits pequenos, integração diária	Trunk-based correlaciona com alto desempenho de entrega; CI	Forsgren, Humble & Kim (2018); Fowler (2006)
Teste primeiro	Quatro times industriais: 40 a 90% menos defeitos, 15 a 35% mais tempo inicial; limites da evidência sobre TDD; pirâmide de testes	Nagappan et al. (2008); Karac & Turhan (2018); Cohn (2009)
Revisão em horas, PR pequeno	Revisão moderna: expectativas versus resultados; práticas convergentes entre empresas; cobertura e participação em revisão correlacionam com menos defeitos	Bacchelli & Bird (2013); Rigby & Bird (2013); McIntosh et al. (2014)

PRÁTICA	O QUE A EVIDÊNCIA DIZ	FONTES
Refatoração dentro da história; toggles; infraestrutura como código	Refactoring; feature toggles; IaC	Fowler (2018); Hodgson (2017); Morris (2016)
SLO e orçamento de erro	SRE	Beyer et al. (2016)
Dívida técnica deliberada como ADR	Quadrante de dívida técnica; dívida técnica como teoria	Fowler (2009); Kruchten, Nord & Ozkaya (2012)
Agentes de IA: fronteira irregular	Experimento de campo com consultores	Dell'Acqua et al. (2023)

Anexo K · Nível de confiança das fontes

Seguimos a prática do documento consolidado do MIT xPRO: dizer o que é fonte primária revisada por pares, o que é livro ou relatório técnico, o que é literatura cinza e o que é adaptação nossa. Quem faz diligência precisa saber onde termina a evidência e começa a escolha.

NÍVEL	O QUE É	QUEM ESTÁ AQUI
Revisado por pares	Artigo em periódico ou conferência com revisão	Ahmad et al.; Alavi & Leidner; Amershi et al. (a, b); Bjørnson & Dingsøy; Conklin; Hogan et al.; Nonaka; Rus & Lindvall; Tyree & Akerman; Bacchelli & Bird; Bansal et al.; Baxter & Sommerville; Bezemer & Zaidman; Boehm; Bosch; Breck et al.; Chang et al.; Cooper (1990, 2014); Cooper & Sommer; Dybå & Dingsøy; Edmondson; Fagerholm et al.; Fitzgerald & Stol; Fowler (2003); Gebru et al.; Guo et al.; Hannay et al.; Hoepman; Hutchinson et al.; Jansen & Bosch; Jobin et al.; Karac & Turhan; Kohavi et al.; Kreuzberger et al.; Kruchten et al.; Lehman; Lewis et al.; Little; Lundberg & Lee; MacCormack et al.; Machado et al.; McIntosh et al.; Miller; Mitchell et al.; Nagappan et al.; Olsson & Bosch; Paleyes et al.; Parasuraman et al.; Parnas; Raji et al.; Ribeiro et al.; Rigby & Bird; Sculley et al.; Selbst et al.; Shahin et al.; Shneiderman; Sullivan et al.; Sy; Vaithilingam et al.; van der Aalst et al.; Wang & Strong; Weidinger et al.; Ziegler et al.
Livro ou relatório técnico	Editora acadêmica ou técnica; relatório de instituto	Bain; Baldwin & Clark; Bass et al.; Engelbart; Nonaka & Takeuchi; Beck; Beyer et al.; Brooks; Caroli; Cavoukian; Cockburn (2004); Cohn; Dehghani; Dekker; Evans; Ford et al. (2021, 2022); Forsgren, Humble & Kim; Fowler (2018); Highsmith et al.; Humble & Farley; Hunt & Thomas; Kazman et al.; Majors et al.; Malone; Malone & Bernstein; Morris; NIST; Ohno; Poppendieck & Poppendieck; Skelton & Pais
Literatura cinza	Blog, site, curso	Allspaw; Cockburn (2005); Fowler (2004–2015, bliki); Hodgson; MIT xPRO (programa e documento consolidado); Nygard; Robinson; Sato et al.; Zaharia et al.
Adaptação Sciensa	Escolhas nossas, apoiadas nas fontes acima mas não prescritas por elas	As quatro camadas; as seis fases operacionais; a escada de autonomia em seis degraus; AI Product Brief, AI Fit Canvas e Allocation Map como templates; a regra “o gate avalia evidência, não capacidade”; a fila de decisões com default; as três regras do AI Orchestrator; a quarta classe de risco (falhas generativas); o Company Brain com a regra “tudo nasce como nota”; o Apêndice H

NÍVEL	O QUE É	QUEM ESTÁ AQUI
Periódico de gestão, working paper ou pré-print	HBR, MIT SMR, HBS Working Paper, arXiv, ACM Queue, The Atlantic	Blank; Brown; Bush; Conway; Dell'Acqua et al.; Doshi-Velez & Kim; Forsgren & Humble; Hax & Wilde; Peng et al.; Takeuchi & Nonaka; Vogels; Wilson & Kelling

Ao citar esta metodologia externamente, atribuímos ao MIT xPRO apenas os quatro estágios e as oito decisões; ao Thoughtworks, as práticas de Inception, showcase e entrega contínua na forma em que as descrevem; e a nós mesmos, a última linha da tabela.



Metodologia 2026

Sciensa · Metodologia 2026

STRATEGY · DIGITAL · TECH

Alameda Campinas, 802 · 12º andar · São Paulo · SP