O Engenheiro Livre

IA, Código e a Reconquista da Autonomia Profissional

Carlos Eduardo Gonçalves Keese

1ª edição — 2026
Capa, diagramas e diagramação: do autor, com auxílio de ferramentas de inteligência artificial.

Sumário

Prólogo — A Isca, o Anzol e a Linha

Fevereiro de 2026. Num palco iluminado como um lançamento de smartphone, a Dassault Systèmes apresenta ao mundo os Virtual Companions: assistentes de inteligência artificial embutidos na 3DEXPERIENCE, um para cada papel — o engenheiro, o cientista, o executivo. Ao lado, a NVIDIA promete simulações mil vezes mais rápidas. A plateia aplaude. As manchetes falam em futurismo, em economia generativa, em uma nova era do design industrial.

Ninguém no auditório faz a pergunta óbvia: mil vezes mais rápido *dentro de quê*?

Eu não estava lá. Em 2012 eu desenhava ferramentas de corte rotativas em metal duro, no AutoCAD, e a única coisa que eu queria aprender era Lisp — não porque sonhava em ser programador, mas porque via, todo santo dia, o tanto de tempo que perdia repetindo à mão o que uma rotina de trinta linhas resolveria em segundos. É a mesma pergunta que ainda faço hoje, toda vez que um desenho trava numa árvore de histórico que ninguém mais lembra por que existe: por que uma tarefa simples precisa demorar tanto só por causa de burocracia que, ali, não cumpre função nenhuma?

Hoje projeto moldes de injeção plástica, uso o ambiente síncrono do Solid Edge para quase tudo que desenho e recorro ao ordenado só onde a peça realmente exige — recuso pagar o preço de uma burocracia de histórico quando ela não serve a nada além de tradição. E, depois de mais de dez anos perseguindo aquela ideia de 2012, ainda estou aprendendo a programar de verdade, agora com ajuda de inteligência artificial — a mesma ferramenta que, bem usada, devolve poder em vez de tirar, e que está, neste exato parágrafo, coescrevendo este livro comigo. O caminho entre aquele desenho no AutoCAD e esta página teve mais curvas do que cabe aqui; ele ganha o espaço que merece mais à frente, no Capítulo 6.

Essa é a diferença entre uma ferramenta e uma gaiola. E é a diferença que a indústria de engenharia está, ativamente, tentando apagar da sua cabeça.

A gaiola confortável

Toda gaiola boa tem duas partes: as grades e a comida. As grades sozinhas ninguém aceita — todo profissional que já foi obrigado a usar um sistema travado, lento, cheio de burocracia de licenciamento, sabe reconhecer uma prisão quando vê uma. Mas a comida é outra história. Quando a gaiola vem com processamento mil vezes mais rápido, simulação em tempo real, um assistente que nunca erra o nome de uma peça — a maioria de nós entra caminhando, de bom grado, e fecha a porta atrás de si.

É esse o movimento que a 3DEXPERIENCE World de 2026 deixou mais claro do que qualquer relatório de mercado já tinha conseguido: quanto mais poder a inteligência artificial entrega, mais fundo o profissional afunda no silo que a entrega. O preço da capacidade computacional deixou de ser medido em reais por licença. Passou a ser medido em dependência estrutural — o quanto da sua capacidade de trabalhar sem aquela empresa específica você está disposto a perder, em troca de nunca mais precisar pensar em como resolver o problema sozinho.

Não é a primeira vez que isso acontece. A prancheta virou CAD, o CAD virou formatos proprietários que só a própria fabricante conseguia abrir direito, o formato proprietário virou assinatura na nuvem, e agora a assinatura na nuvem vem com um copiloto de IA que só funciona se você nunca sair de dentro dela. Cada geração prometeu liberdade. Cada geração entregou uma coleira nova, um pouco mais confortável que a anterior, um pouco mais difícil de perceber como coleira.

Vale trocar de imagem por um instante, porque a da gaiola não conta tudo. O que a indústria oferece hoje é uma **isca**: a conveniência que ninguém recusa — o processamento mil vezes mais rápido, o assistente que nunca erra o nome de uma peça. Dentro da isca vem o **anzol**, que você não vê quando morde: a dependência costurada na conveniência, o serviço de nuvem que já vem plugado, o formato que só a fabricante abre. E preso ao anzol vem a **linha** — o controle que puxa aos poucos, mês após mês de assinatura, até que sair custe mais caro do que ficar. A isca é honesta: ela entrega o que promete. O anzol e a linha é que cobram o preço.

A pergunta que este livro faz não é se a inteligência artificial vai mudar a engenharia. Isso já é fato consumado, e fingir dúvida sobre isso seria desonestidade — tanto minha quanto sua, leitor. A pergunta é outra, e é mais incômoda: **quem vai ser dono da inteligência que move o seu trabalho — você, ou a plataforma que aluga ela para você por mês?**

A linha do outro lado

Existe uma resposta possível a essa pergunta que não é romântica nem ingênua, e é essa resposta que vou tentar construir capítulo a capítulo. Enquanto corporações investem bilhões para fechar seus jardins murados, uma comunidade dispersa de desenvolvedores (sem CEO, sem evento de lançamento com palco iluminado) consolidou um protocolo simples que faz exatamente o oposto: permite que qualquer inteligência artificial, de qualquer fornecedor, converse com qualquer ferramenta, sem pedir licença a um servidor central. Em pouco mais de um ano, esse protocolo já foi adotado por praticamente todos os grandes laboratórios de IA do mundo, inclusive os que competem ferozmente entre si em todo o resto.

Ao mesmo tempo, um modelador paramétrico de código aberto, construído ao longo de mais de vinte anos por voluntários, finalmente amadureceu o suficiente para deixar de ser curiosidade de fórum e passar a ser considerado, pela própria imprensa técnica do setor, uma ferramenta profissional viável. Não é coincidência que isso esteja acontecendo no mesmo momento em que os gigantes do CAD apostam todas as fichas em inteligência fechada. É a prova, em tempo real, de que a bifurcação entre soberania aberta e automação proprietária não é retórica de manifesto — é o mapa real do setor em 2026.

Este livro não é contra a inteligência artificial. É contra a ideia de que capacidade e dependência precisam vir no mesmo pacote, como se fossem, por natureza, a mesma coisa. Elas não são. A diferença entre as duas é sempre uma escolha — de arquitetura, de protocolo, de quem decide o que você pode e não pode fazer com a sua própria ferramenta de trabalho.

De onde eu falo

Sou projetista de moldes de injeção plástica, estudante de Engenharia de Computação e alguém que aprendeu a programar por teimosia, não por vocação de origem. Sou também, sem meio-termo sobre isso, alguém que acredita que liberdade individual não é um adorno filosófico — é o princípio organizador de como eu escolho minhas ferramentas, minha stack, meu trabalho e minha vida. Não escondo de onde falo: acredito que o Estado, na forma inchada em que existe hoje, é antes de tudo um ladrão institucionalizado, e que quando ele se junta a empresas que preferem virar monopólio corporativista a competir de verdade, o resultado é sempre o mesmo — mais corrupção, inversão de valores, erosão lenta das bases que sustentam uma sociedade livre.

Não escrevo este livro pretendendo convencer você da minha fé ou da minha política. Escrevo porque, como cristão, libertário, projetista e programador ainda em formação, entendo a liberdade sobre a própria ferramenta de trabalho como parte do mesmo dever: o de nunca entregar de bandeja o que ninguém tem o direito de tomar de você.

Não peço que você concorde com tudo isso para acompanhar o argumento que vem a seguir. Peço só que você faça, junto comigo, a mesma pergunta que fiz em 2012, na frente de um desenho de ferramenta de corte que eu queria automatizar e ainda não sabia como: essa ferramenta está me dando poder, ou está me pedindo, em troca do poder, a chave da porta?

O resto deste livro é a tentativa de responder — com casos reais, números verificáveis e a experiência de quem projeta, programa e recusa, sempre que pode, a comer a isca sem primeiro procurar o anzol.

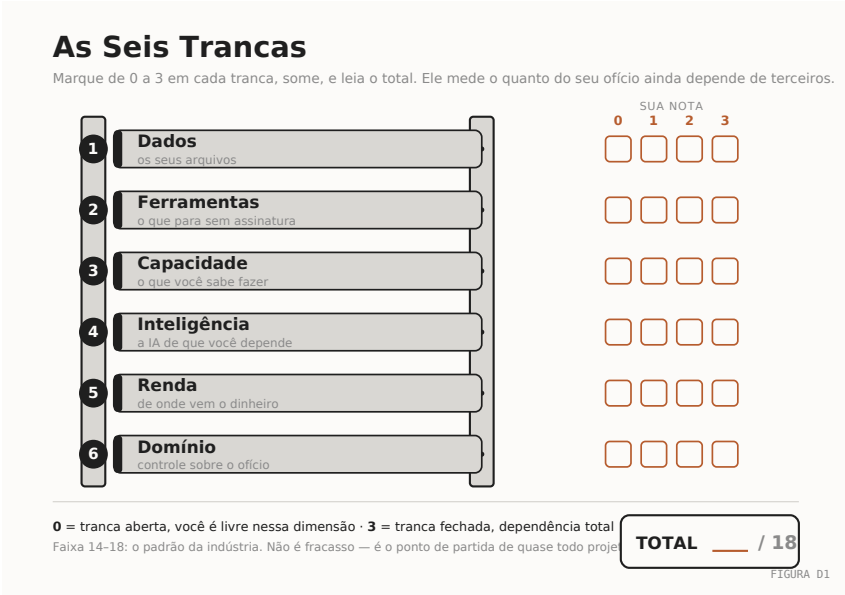
Antes de Começar — As Seis Trancas

Este livro vai te propor uma ideia incômoda: a de que você provavelmente está dentro de uma gaiola profissional que não escolheu, e que é confortável o bastante para você nunca ter reparado nela.

Discutir isso no abstrato não serve para nada. Então, antes de começar a leitura, vamos medir. Leva menos de dez minutos.

A gaiola não tem uma tranca só. Tem seis — seis dependências diferentes que, juntas, determinam quanta liberdade real você tem sobre o próprio ofício. A maioria dos bons profissionais está com quatro ou cinco fechadas e não sabe. Não é vergonha nenhuma: é o padrão da indústria, é o que a estrutura empurra, e é exatamente onde eu estava até poucos anos atrás.

Responda com sinceridade — o teste só funciona se você não estiver se vendendo para si mesmo. Some os pontos no fim. E anote o resultado com a data em algum lugar, porque no fim do livro você vai refazer.



Tranca 1 — Os seus dados

Se você perdesse hoje o acesso ao software principal que usa, o que aconteceria com o trabalho que produziu nos últimos cinco anos?

- **0** — Nada de grave. Mantenho cópias também em formatos neutros que qualquer ferramenta abre.
- **1** — Recuperaria a maior parte, com bastante trabalho.
- **2** — Recuperaria pouca coisa, e com perda de informação.
- **3** — Perderia praticamente tudo, ou ficaria dependendo da boa vontade de alguém.

Tranca 2 — As suas ferramentas

Quantas ferramentas essenciais ao seu trabalho parariam de funcionar se você deixasse de pagar uma assinatura ou perdesse o acesso a uma conta?

- **0** — Nenhuma. Consigo trabalhar com o que possuo ou rodo por conta própria.
- **1** — Uma, e eu tenho uma alternativa que sei usar de verdade.
- **2** — Uma ou mais, e a alternativa eu conheço só de nome.
- **3** — Praticamente todas. Sem elas, eu paro.

Tranca 3 — A sua capacidade

Quando o seu software não faz algo de que você precisa, o que você faz?

- **0** — Eu mesmo construo a solução — macro, script, automação — e já fiz isso mais de uma vez.
- **1** — Já tentei e cheguei a algum resultado, com ajuda.
- **2** — Sei que dá para fazer, mas nunca fiz.
- **3** — Espero a próxima versão, ou faço na mão para sempre.

Tranca 4 — A sua inteligência

A inteligência artificial que você usa no trabalho: onde ela roda e a quem ela responde?

- **0** — Uso o que for melhor para cada caso, mas sei rodar um modelo na minha própria máquina quando o trabalho é sensível.
- **1** — Uso serviços na nuvem com consciência do que envio, e já experimentei rodar algo localmente.
- **2** — Uso o que vem embutido nas ferramentas, sem pensar muito nisso.
- **3** — Uso só o que a plataforma me oferece e não sei para onde vão os meus dados — ou não uso nada, e a barreira continua de pé.

Tranca 5 — A sua renda

Quantas mãos diferentes podem, sozinhas, cortar mais da metade da sua renda no mês que vem?

- **0** — Três ou mais fontes independentes. Nenhuma delas, sozinha, me derruba.
- **1** — Duas fontes, uma bem maior que a outra.
- **2** — Uma fonte principal, com alguma coisa pequena ao lado.
- **3** — Uma só. Um empregador, um cliente, ou uma plataforma.

Tranca 6 — O seu domínio

O problema mais difícil que você resolve no trabalho: quantas pessoas na sua região resolvem tão bem quanto você?

- **0** — Poucas, e os clientes ou a empresa me procuram justamente por causa disso.
- **1** — Algumas. Tenho uma diferenciação real, ainda que estreita.
- **2** — Muitas. O que me distingue é experiência genérica.
- **3** — Quase qualquer profissional da área faria o que eu faço.

O seu resultado

Some os seis números. O resultado vai de 0 a 18.

0 a 3 — Do lado de fora. Raro, e provavelmente você não precisa deste livro para a sua própria vida. Precisa dele para explicar a alguém por que você faz as coisas do jeito que faz.

4 a 8 — Porta entreaberta. Você já conquistou autonomia real em algumas frentes, quase sempre sem ter chamado aquilo de autonomia. O livro vai te

dar o nome das coisas e mostrar as frentes que faltam.

9 a 13 — Dentro, com a chave ao alcance. É onde está a maioria dos bons profissionais técnicos. Você é competente, é reconhecido, e mesmo assim a sua capacidade de trabalhar depende de decisões que não são suas. É a faixa em que este livro tem mais a oferecer.

14 a 18 — Trancado. Não leia isso como fracasso, porque não é. É o resultado padrão de quem seguiu exatamente o que a estrutura mandou seguir: aprenda a ferramenta que a empresa comprou, faça bem o que mandaram, seja leal. Eu marquei nessa faixa durante mais de uma década, com quinze anos de profissão nas costas. Sair dela não exigiu talento excepcional. Exigiu teimosia e a sequência certa de passos pequenos, que é o que este livro tenta te entregar.

Para onde a sua nota te leva

Um livro sobre liberdade seria hipócrita se te obrigasse a lê-lo em ordem. Anote as **duas trancas em que você pontuou mais alto** — são as suas. E vá primeiro aos capítulos que tratam delas:

A sua tranca mais fechada	Vá primeiro para
1 — Dados	Capítulos 2 e 3, e a primeira camada do Capítulo 14
2 — Ferramentas	Capítulos 1, 5 e 7
3 — Capacidade	Capítulos 6 e 14, e o Apêndice (o caminho de 30 dias)
4 — Inteligência	Capítulos 7 e 8
5 — Renda	O Interlúdio, e os Capítulos 12 e 13
6 — Domínio	O Interlúdio e o Capítulo 12

E aqui vai o conselho mais prático deste livro inteiro, que eu aprendi tentando fazer o contrário: **não tente abrir as seis trancas.** Quem tenta abrir todas de uma vez não abre nenhuma, e desiste em duas semanas. Escolha as duas mais fechadas e trabalhe só nelas até que se mexam. As outras quatro esperam — e algumas, você vai descobrir, se afrouxam sozinhas quando as primeiras cedem.

As três perguntas, para o resto da vida

As seis trancas medem onde você está hoje. Mas todo mês vai aparecer uma ferramenta nova, uma plataforma nova, uma inteligência nova prometendo mudar a sua vida — e você vai precisar de um teste rápido, de bolso, para decidir se ela te liberta ou te prende mais.

São três perguntas. Elas voltam ao longo do livro, e valem para qualquer coisa que apareça:

Eu consigo rodar isto, ou algo equivalente, sem pedir permissão a ninguém?

Se eu quiser sair, eu levo o meu trabalho comigo?

Quem define o preço da minha dependência — eu, ou outra pessoa?

Guarde as duas coisas: as seis trancas para diagnosticar a sua situação, as três perguntas para avaliar cada nova ferramenta. Juntas, elas são o sistema inteiro que este livro constrói ao longo de catorze capítulos. O resto é explicação, evidência e método.

Anote a sua nota. Vamos começar.

Parte I

O Diagnóstico: A Gaiola

Capítulo 1 — A Gaiola Confortável

No prólogo prometi voltar a Houston, fevereiro de 2026, com mais detalhe. É hora de cumprir essa promessa — porque os detalhes, nesse caso, importam mais do que o resumo que qualquer manchete conseguiu dar.

Houston, fevereiro de 2026

Entre os dias 1 e 4 daquele mês, milhares de usuários de SOLIDWORKS e da plataforma 3DEXPERIENCE se reuniram em Houston para a conferência anual da Dassault Systèmes. No palco, o CEO Pascal Daloz apresentou uma categoria inteiramente nova de produto: os Virtual Companions — não chatbots, segundo a própria empresa, mas agentes que combinam modelos de linguagem com "modelos de mundo" específicos da indústria e simulação multiescala e multidisciplinar, prometendo rastreabilidade ao longo de todo o ciclo de vida de um produto.

Três nomes foram apresentados. Aura, já disponível na plataforma, orquestra conhecimento e contexto entre requisitos, projetos e mudanças — o tipo de trabalho que hoje consome incontáveis horas de reunião e planilha. Leo e Marie chegariam ao longo do ano, com papéis complementares ainda não totalmente detalhados ao público. Daloz resumiu o momento com uma frase que a empresa repetiria em quase todo comunicado subsequente: entramos na "economia generativa", onde o que decide o valor de uma indústria não é mais quem fabrica um produto, mas quem detém o conhecimento de como fazê-lo. A meta declarada dos Virtual Companions, nas palavras do próprio CEO, era fazer "o invisível visível e o impossível possível" antes que qualquer coisa existisse fisicamente.

É uma frase bonita. E, como toda frase bonita de keynote, vale a pena desmontar exatamente o que ela está vendendo.

Três nomes, uma arquitetura

O detalhe técnico que a cobertura de imprensa registrou, mas poucos pararam para examinar, é este: a plataforma foi descrita como capaz de

gerenciar a “coreografia assíncrona” de milhares de Virtual Companions e humanos trabalhando ao mesmo tempo, cumprindo o que a empresa chamou de “requisitos de soberania”.

Soberania. A palavra escolhida não é acidente de tradução — é linguagem de contrato, pensada para tranquilizar departamentos jurídicos e clientes governamentais de que os dados de cada organização ficam isolados dos dados de outra, dentro do mesmo ambiente compartilhado. É uma promessa legítima, e provavelmente necessária para convencer indústrias de defesa e aeroespacial a colocar seus projetos mais sensíveis dentro de uma nuvem de terceiros.

Mas repare no que a palavra protege e no que ela não protege. Ela garante a soberania da empresa-cliente sobre seus próprios dados dentro da plataforma. Ela não diz uma palavra sobre a soberania do engenheiro individual sobre a própria ferramenta de trabalho. Duas soberanias completamente diferentes, uma delas cuidadosamente equacionada em contrato, a outra simplesmente ausente da conversa — porque, arquitetonicamente, ela não pode existir dentro desse desenho. Você pode ser dono dos seus dados e, ainda assim, não ser dono da sua capacidade de trabalhar.

A promessa da NVIDIA — e o que fica de fora dela

Dias antes do keynote principal, Dassault Systèmes e NVIDIA anunciaram uma parceria estratégica de longo prazo para construir uma arquitetura industrial compartilhada de inteligência artificial. A integração prometida é profunda: tecnologias CUDA-X, RTX e Omniverse embutidas diretamente na 3DEXPERIENCE, com promessa de acelerar simulações entre cem e mil vezes. Jensen Huang, CEO da NVIDIA, descreveu um futuro em que tudo — carros, os robôs que os constroem, as fábricas onde isso acontece — é definido por software, com simulações que antes rodavam offline agora acontecendo em tempo real. A parceria também funciona ao contrário: a própria NVIDIA usa o software de engenharia de sistemas baseada em modelos da Dassault para projetar suas próprias fábricas de inteligência artificial.

Há um detalhe nessa parceria que merece atenção redobrada, porque ele antecipa um argumento que vai voltar várias vezes neste livro. Parte da inteligência que move os Virtual Companions vem dos modelos Nemotron, da própria NVIDIA — e os modelos Nemotron são, tecnicamente, modelos de pesos abertos. Alguém defendendo a Dassault poderia usar esse fato como prova de que a plataforma não é tão fechada assim.

Não caia nessa. Um ingrediente aberto dentro de uma receita fechada não torna o prato aberto. Você, profissional individual, não tem acesso aos pesos do Nemotron para rodar onde quiser, do jeito que quiser, fora do contrato da Dassault. Você tem acesso ao que a Dassault decide expor, dentro do ambiente da Dassault, nos termos da Dassault. A abertura do componente interno não se transfere para a abertura da experiência que você, do lado de fora, efetivamente vive. Essa distinção, entre um ingrediente aberto e uma arquitetura aberta, vai reaparecer no Capítulo 7, quando formos falar sobre o que realmente separa uma inteligência artificial que liberta de uma que aprisiona. Por ora, guarde a pergunta: de que adianta o motor ser aberto se o carro inteiro só anda dentro de uma pista fechada?

A isca que vem grudada no CAD de sempre

Junto com os Virtual Companions, a Dassault lançou o SOLIDWORKS 2026, com centenas de melhorias em projeto, simulação e gestão de dados. Entre elas, recursos de IA generativa que aceleram a criação de desenhos técnicos e reconhecem automaticamente componentes do tipo fixador dentro de conjuntos montados — o tipo de recurso que qualquer projetista de moldes, eu incluído, reconhece de imediato como economia real de tempo em tarefa repetitiva.

É aqui que a arquitetura da gaiola fica mais sutil, e por isso mais eficaz. Desde uma mudança de política em julho de 2023, toda compra do SOLIDWORKS já inclui, por padrão, os SOLIDWORKS Cloud Services — o que conecta automaticamente o "CAD de sempre" à plataforma 3DEXPERIENCE, mesmo que o comprador só quisesse, como sempre quis, um programa de desktop para desenhar peça mecânica. Ninguém precisa assinar nada à parte. Ninguém precisa nem perceber que isso aconteceu. A isca não é mais um recurso opcional que você escolhe ativar — ela vem grudada na caixa, e o cabo que a conecta ao ecossistema maior já está plugado antes de você abrir o programa pela primeira vez.

Duas soberanias, e a que ninguém defende

Vale insistir na palavra "soberania" que a Dassault escolheu, porque ela abre uma distinção que vai estruturar este livro inteiro. Quando a plataforma promete soberania, ela promete uma coisa específica e real: que os dados da *empresa-cliente* ficam isolados, protegidos, sob controle jurídico daquela organização. É soberania corporativa — a de uma pessoa jurídica sobre seus

ativos dentro de um contrato.

Existe, porém, uma segunda soberania, e ela não aparece em nenhum comunicado de imprensa: a do engenheiro individual sobre a própria capacidade de trabalhar. Essa não está sendo oferecida a ninguém. Ela é, na verdade, o que se paga em troca da primeira. A empresa ganha soberania sobre seus dados; o profissional dentro dela perde soberania sobre seu ofício, porque cada fluxo de trabalho, cada automação, cada julgamento técnico passa a acontecer dentro de um ambiente que ele não controla e do qual não pode sair sem perder tudo o que construiu ali.

Essa assimetria não é um detalhe — é o cerne político de toda a discussão.[1] A tradição do pensamento liberal clássico sempre insistiu que a liberdade que importa de verdade é a do indivíduo, não a das instituições que o cercam; uma corporação pode ser perfeitamente soberana sobre seus dados e, ao mesmo tempo, presidir a erosão completa da autonomia das pessoas que trabalham dentro dela. As duas coisas não só convivem: uma se alimenta da outra. Quando este livro fala em liberdade do engenheiro, é sempre dessa segunda soberania que se trata — a que nenhuma plataforma vende, porque vendê-la seria abrir mão exatamente do que a torna lucrativa.

O preço da inteligência alugada

Vale colocar números ao lado da poesia da keynote, porque números não aplaudem sozinhos. Em 2026, o SOLIDWORKS Design com Cloud Services custa, para um único usuário nos Estados Unidos, algo entre 2.820 e 4.716 dólares por ano, dependendo do nível — Standard, Professional ou Premium. Novas licenças passaram a exigir compromisso mínimo de dois anos, eliminando a flexibilidade que existia antes para quem preferia contratos mais curtos. E esse é só o preço da base: pacotes de inteligência artificial vendidos como complemento ao SOLIDWORKS, oferecidos por integradores terceirizados, giram entre 500 e 2.500 dólares anuais por usuário para integrações modulares básicas, podendo escalar para a casa dos milhares quando o cliente precisa de acesso a API personalizada ou processamento mais pesado.

Esse padrão de reajuste não é exclusividade da Dassault. A Autodesk, concorrente direta em parte do mercado CAD, eliminou descontos

[1] A distinção entre a liberdade do indivíduo e a das coletividades ou instituições que o cercam é um tema recorrente do liberalismo clássico, de Alexis de Tocqueville a Friedrich Hayek. O ponto central, para os fins deste livro, é que soberania institucional e soberania individual não são a mesma coisa, e frequentemente estão em conflito direto.

plurianuais em 2025 e passou a aplicar reajustes de renovação entre 5% e 17%. O movimento é setorial: a indústria inteira de software de engenharia está, ao mesmo tempo, aumentando o preço da base e vendendo a inteligência artificial como camada extra, separada, metrificada — exatamente o oposto do que a retórica de "democratização pela IA" promete nos palcos.

Junte as duas peças. A "gaiola confortável" do prólogo não é mais só metáfora: ela tem um número no boleto, reajustado todo ano, com a inteligência que a torna confortável cobrada à parte, por cima da base que já ficou mais cara e menos flexível.

Quem decide o que é bom para o engenheiro

Há uma premissa silenciosa embutida em toda a arquitetura dos Virtual Companions, e vale trazê-la para a luz porque ela é o coração filosófico da gaiola. Quando uma empresa cria um assistente "para o engenheiro", outro "para o cientista", outro "para o executivo", ela está fazendo uma afirmação que raramente é dita em voz alta: ela sabe, melhor do que você, qual é o formato certo da sua própria inteligência profissional. Ela pré-decidiu quais perguntas você vai fazer, quais fluxos de trabalho você segue, o que conta como "ajuda" no seu ofício — e embutiu essas decisões no produto antes de você abrir o programa.

Repare no contraste com a cena que abriu este livro. Quando escrevi minha macro de extração de lista de materiais no Solid Edge — o tipo de automação que reconstruo, do zero, no apêndice deste livro —, ninguém na Siemens tinha decidido, de antemão, que aquele era um problema que eu precisava resolver, nem como. Eu identifiquei o gargalo, eu desenhei a solução, eu escolhi a forma dela. A ferramenta me deu uma linguagem (a API, o COM interop) e ficou fora do meu caminho. Essa é a diferença mais importante deste capítulo inteiro, e ela não aparece em nenhuma tabela de preços: uma ferramenta aberta te dá uma linguagem e sai da frente; uma gaiola confortável te dá um assistente e decide, por você, o que você tem o direito de querer.

Não é que os Virtual Companions sejam ruins no que fazem — provavelmente são excelentes, e é justamente por isso que funcionam como gaiola. Um assistente medíocre ninguém adota. O risco não está na incompetência da ferramenta; está na terceirização silenciosa do julgamento profissional. Cada vez que você aceita a resposta pronta em vez de entender o problema, você troca um pouco da sua capacidade de resolver sozinho por um pouco mais de conforto — e essa troca, repetida mil vezes ao longo de

anos, é exatamente o "lock-in de wetware" que um economista descreveu décadas atrás e que vamos examinar em detalhe no Capítulo 3. A gaiola mais eficaz não é a que te impede de sair. É a que faz você esquecer como se anda sozinho do lado de fora.

Trezentos e setenta mil clientes depois

Convém não tratar isso como debate de nicho para entusiastas de tecnologia. A própria Dassault Systèmes afirma que 370 mil clientes, de todos os portes e setores, usam a plataforma 3DEXPERIENCE ao redor do mundo. Não é uma experiência de laboratório — é, cada vez mais, a camada operacional por trás de como uma fatia relevante da manufatura global projeta, simula e, agora, "colabora" com inteligência artificial.

É por isso que a arquitetura importa mais do que o discurso. Quando uma plataforma atinge essa escala, a forma como ela está desenhada — o que ela deixa você levar embora se decidir sair, o que ela obriga você a alugar em vez de possuir, o quanto de "soberania" ela realmente devolve ao profissional individual — deixa de ser detalhe técnico e passa a ser, de fato, política industrial. Ela decide, na prática, quanta liberdade centenas de milhares de engenheiros vão ter sobre o próprio ofício nos próximos anos.

O que vem a seguir

O que aconteceu em Houston não é o início da gaiola. É a sua versão mais recente, mais bem vestida — e a primeira em que a própria comida da gaiola vem cobrada à parte, por assinatura, com reajuste anual. No próximo capítulo, vou voltar no tempo, da prancheta ao cloud CAD, para mostrar que esse padrão é bem mais velho do que qualquer keynote de fevereiro. A inteligência artificial não inventou o cercamento. Ela só apertou o parafuso — e, pela primeira vez, colocou um preço visível em cada volta dele.

Fontes: Dassault Systèmes, comunicados de imprensa sobre Virtual Companions e parceria NVIDIA (3ds.com, fev/2026); Engineering.com e Enterprise Times, cobertura da 3DEXPERIENCE World 2026 (fev/2026); SWYFT Solutions, Ohmycad e XDinnovation, guias de preço SOLIDWORKS 2026; Energent.ai, análise de preço de soluções de IA para SOLIDWORKS (abr/2026); checkthat.ai, comparativo de pricing CAD 2026 (mar/2026).

Para levar deste capítulo

Tranca 2 — Ferramentas. Este capítulo mostra como a gaiola é construída para ser confortável: quanto mais úteis os recursos que só existem lá dentro, mais caro fica sair.

Faça agora — 15 minutos. Liste, num papel só, todas as ferramentas de software que você usa para trabalhar. Ao lado de cada uma, marque um X nas que param de funcionar se uma assinatura vencer ou uma conta for cortada. Conte os X. Esse número é a sua Tranca 2.

Capítulo 2 — Breve História do Cercamento

Toda gaiola tem uma origem que, no começo, não parecia gaiola nenhuma. Geralmente parecia o oposto: uma fuga. Este capítulo é sobre esse padrão se repetindo, com atores diferentes, por quase cinquenta anos — e sobre como a própria empresa que abriu o Capítulo 1 é, sem exagero, o melhor estudo de caso desse padrão que existe.

Da prancheta ao CATI

Em 1975, a Avions Marcel Dassault (a construtora aeronáutica francesa por trás dos caças Mirage) comprou licenças do CADAM, um software de desenho da Lockheed que automatizava boa parte do trabalho de desenho técnico em duas dimensões. Era um avanço real: o computador passava a fazer os cálculos que antes cabiam à régua e ao compasso. Mas era, na prática, uma prancheta eletrônica. Para o tipo de geometria complexa que um caça exige, duas dimensões não bastavam, e a Dassault dependia de uma ferramenta que não era sua, construída por uma empresa americana, para resolver um problema que só ficaria mais difícil com o tempo.

Em 1978, o diretor de engenharia da empresa pediu que sua equipe de projeto assistido por computador desenvolvesse algo próprio: uma ferramenta tridimensional, interativa, capaz de desenhar e usinar peças complexas direto a partir do modelo — sem depender de outra empresa para isso. Uma equipe de 15 engenheiros, liderada por Francis Bernard, construiu o CATI (Conception Assistée Tridimensionnelle Interactive) em 1977. O resultado foi imediato: a primeira asa de túnel de vento projetada com o novo sistema, que antes levava seis meses para ser construída a partir de desenhos, saiu em quatro semanas.

Vale parar aqui um segundo, porque o impulso por trás do CATI é exatamente o mesmo impulso que abriu este livro: um profissional cansado de esperar que a ferramenta de outra empresa resolvesse o seu problema, decidindo construir a própria solução. A diferença é de escala — eu escrevi uma macro para automatizar uma requisição de compra; a Dassault Aviation montou um

departamento inteiro para reinventar o CAD tridimensional — mas o gesto é idêntico. Toda gaiola confortável começa como alguém recusando ficar preso numa gaiola alheia.

O nascimento com uma coleira embutida

Foi aí que a história tomou o primeiro dos seus giros irônicos. Em 1981, a Dassault Aviation decidiu comercializar o CATI para outras fabricantes aeroespaciais, renomeou o programa para CATIA e criou uma subsidiária independente, a Dassault Systèmes, só para tocar esse negócio. Era uma aposta ousada para a época: nenhuma outra fabricante de aviões vendia publicamente suas próprias ferramentas de projeto. Mas a nova empresa tinha um problema imediato: 15 engenheiros não formam uma força de vendas capaz de competir com concorrentes americanas que já tinham centenas de funcionários.

A solução foi um acordo, fechado em julho daquele mesmo ano, com a IBM — então no auge do seu domínio na indústria de computadores. A IBM assumiria a venda, distribuição e suporte técnico do CATIA em todo o mundo, em troca de cerca de metade da receita gerada. A parceria durou quase trinta anos, e foi decisiva para colocar o CATIA dentro da Boeing, que o adotaria em 1984 e, em 1992, o usaria para desenhar o 777 — o primeiro avião comercial projetado inteiramente por computador, do início ao fim.

Ou seja: a mesmíssima empresa que nasceu da recusa de depender da ferramenta alheia (Lockheed/CADAM) entregou, quatro anos depois de criar sua própria ferramenta, metade da receita dela a um parceiro de infraestrutura maior (IBM), para poder escalar. Não é uma crítica moral ao movimento — sem a IBM, talvez o CATIA nunca tivesse saído da França. É uma constatação estrutural: da noite para o dia, uma tecnologia que nasceu para libertar um engenheiro de uma dependência virou, ela mesma, dependente de outra infraestrutura maior para sobreviver no mercado.

Se esse parágrafo te lembrou o acordo com a NVIDIA que abriu este livro — CUDA-X, RTX, Omniverse, embutidos fundo dentro da 3DEXPERIENCE, quarenta e cinco anos depois do acordo com a IBM — não é coincidência, e não é força de expressão. É o mesmo manual de estratégia, atualizado para a era da inteligência artificial. Um gigante da infraestrutura de computação entra, recebe parte do valor gerado, e em troca entrega a escala que a empresa de engenharia sozinha nunca alcançaria. A única coisa que mudou entre 1981 e 2026 foi o nome do parceiro na porta.

Do outro lado do Atlântico: o nascimento do DWG

Enquanto a Dassault construía uma ferramenta para a aeronáutica de ponta, do outro lado do Atlântico nascia um movimento com intenção quase oposta. A Autodesk lançou o AutoCAD em 1982, rodando em computadores pessoais — pela primeira vez, uma pequena oficina ou um projetista autônomo podia ter, na própria mesa, uma fração do poder que antes só existia em departamentos de engenharia de grandes corporações. O formato nativo do programa, o DWG, tinha origem ainda nos anos 1970, num pacote chamado Interact CAD, e se tornou, com o sucesso do AutoCAD, o formato de desenho técnico mais usado do planeta.

Só que "mais usado" não é o mesmo que "aberto". O DWG nunca foi um padrão publicado e mantido por um comitê independente — é, até hoje, uma especificação controlada pela Autodesk, que licencia o acesso a ela para outros fabricantes através de um programa chamado RealDWG. Empresas que não querem (ou não podem) pagar por essa licença recorrem à engenharia reversa do formato — o esforço mais bem-sucedido nesse sentido é conduzido, há décadas, pela Open Design Alliance, um consórcio sem fins lucrativos formado exatamente para que o restante da indústria não fique refém de uma única fornecedora para ler e escrever o próprio desenho técnico.

Reforce a ironia: o AutoCAD nasceu democratizando o acesso ao CAD, tirando-o das mãos exclusivas de corporações e aeroespacial. E, ainda assim, terminou construindo, ao redor do formato de arquivo mais popular da história do desenho técnico, exatamente o mesmo tipo de cerca que a Dassault construiu ao redor do CATIA — só que essa cerca ficou invisível por décadas, disfarçada de sucesso de mercado, porque "todo mundo usa" soa como abertura quando, na verdade, é apenas dependência bem distribuída.

A gaiola que ninguém vê: o kernel geométrico

Há uma camada de dependência ainda mais profunda que o formato de arquivo, e a maioria dos projetistas passa a carreira inteira sem parar para pensar nela: o kernel geométrico — o motor matemático que, por baixo do software, calcula e sustenta cada superfície, cada furo, cada operação booleana de um modelo 3D. E é aqui que a história fica genuinamente estranha.

O Parasolid, criado em 1988 pela Shape Data, em Cambridge, é hoje

propriedade da Siemens. Ele roda por baixo do NX e do Solid Edge, as duas ferramentas próprias da Siemens, mas também, olha só, por baixo do SolidWorks. O mesmo SolidWorks que pertence à Dassault Systèmes, arquirrival da Siemens em praticamente todo o resto do mercado de CAD. Em 2000, a Dassault comprou a Spatial, fabricante do kernel concorrente ACIS, na aposta óbvia de libertar o SolidWorks da dependência de um componente controlado pelo maior rival da empresa. Duas décadas e meia depois, o SolidWorks continua rodando sobre o Parasolid. A tentativa de fuga simplesmente não vingou — a divisão do SolidWorks nunca aceitou trocar de motor.

E o Solid Edge, a ferramenta que uso todos os dias? Nasceu em 1996 nas mãos da Intergraph, rodando sobre o kernel ACIS. Foi comprado pela UGS em 1998, que trocou o motor para Parasolid. A UGS, por sua vez, foi comprada pela Siemens em 2007, virando a atual Siemens Digital Industries Software. Ou seja: até a ferramenta que eu escolhi, com a qual desenho moldes todos os dias, carrega uma genealogia inteira de fusões e aquisições que decidiu, décadas antes de eu entrar para essa profissão, qual empresa controlaria o motor matemático por trás de cada peça que eu desenho.

Isso não é uma crítica ao Solid Edge — é, na verdade, o motivo pelo qual boa parte da indústria automotiva (Stellantis, Renault, Volkswagen) e aeroespacial (Airbus, Safran) depende tão fortemente de trocas de arquivo baseadas em Parasolid: é um padrão tecnicamente sólido, mantido por quem sabe o que está fazendo. O ponto é outro: mesmo os maiores rivais do mercado de CAD, empresas que gastam fortunas em marketing competitivo uma contra a outra, dependem, no nível mais fundamental de todos, da mesma infraestrutura controlada por uma única empresa. A concorrência que você vê na superfície — Dassault contra Siemens, keynote contra keynote — esconde uma camada de interdependência que nenhuma das duas faz questão de anunciar.

Vou voltar à Siemens com muito mais detalhe no Capítulo 10, quando o assunto for outra coisa completamente diferente: a escolha estratégica da empresa por abertura de plataforma, em contraste direto com o cercamento que a Dassault aprofundou em 2026. Mas guarde esse nome — Parasolid — porque ele é a prova de que, mesmo debaixo da rivalidade mais barulhenta do setor, a dependência de infraestrutura alheia nunca desapareceu. Ela só ficou boa demais em se esconder.

A tentativa de fuga: STEP e IGES

A indústria não ficou parada diante desse problema. Ainda nos anos 1980, engenheiros de diferentes países começaram a desenvolver formatos neutros — não pertencentes a nenhum fabricante — capazes de carregar geometria de um sistema CAD para outro sem que o usuário precisasse depender da boa vontade comercial de uma única empresa. O IGES foi o primeiro esforço relevante nesse sentido. O STEP, iniciado em 1984 e formalizado como norma ISO em 1994, foi mais ambicioso: hoje é a referência global para troca de modelos complexos entre sistemas CAD diferentes, e a indústria aeroespacial o adotou até como formato oficial de arquivamento de longo prazo, sob o padrão LOTAR — a mesma indústria, aliás, que gerou tanto a Dassault quanto boa parte da corrida por CAD tridimensional.

Vale a honestidade que este livro tenta manter em todo capítulo: STEP e IGES não são solução mágica. Um arquivo STEP carrega geometria com precisão, mas normalmente perde a árvore de histórico paramétrico, camadas, hachuras avançadas e outras informações que um formato nativo preserva. Trocar dependência de fornecedor por um formato neutro tem custo técnico real — você ganha liberdade de circular entre sistemas, mas abre mão de parte da inteligência que o software original guardava sobre como aquela peça foi construída. Não existe almoço grátis nem aqui. O que existe é uma escolha: pagar o preço da dependência de um fornecedor, ou pagar o preço, menor mas real, da tradução entre mundos diferentes.

E esse preço não é hipótese acadêmica — é rotina de quem trabalha na cadeia de fornecimento. Um estudo da própria PTC estima que uma empresa industrial usa, em média, 2,7 sistemas CAD diferentes de forma regular. Não por indecisão, mas porque cada cliente, cada fornecedor e cada arquivo antigo herdado de um projeto anterior chega num formato diferente, e recriar geometria de um sistema no outro é uma fonte constante de erro de tradução e perda da intenção original de projeto. O cercamento não impõe só um custo de licença — impõe um custo de convivência forçada entre gaiolas que não foram feitas para conversar entre si, e esse custo recai, quase sempre, sobre o elo mais fraco da cadeia.

Pior ainda quando a incompatibilidade é entre gerações do *mesmo* fabricante. O kernel Granite, da PTC, que sustenta o Creo, não tem compatibilidade retroativa entre versões — o que significa que, mesmo dentro de um único ecossistema proprietário, o profissional pode se ver preso a manter versões antigas de software rodando só para conseguir abrir os próprios arquivos de anos atrás. A dependência, nesse caso, não é nem de uma empresa: é de um

momento congelado no tempo, de uma versão específica que você não pode mais deixar morrer sem perder acesso ao próprio passado.

O parafuso que só aperta

Se os formatos de arquivo são a primeira camada do cercamento, o modelo de licenciamento é a segunda — e essa não tem o mesmo contrapeso de esforço aberto que STEP e IGES representam para os formatos.

Em janeiro de 2016, a Autodesk anunciou que deixaria de vender licenças perpétuas para a maioria dos seus produtos de desktop, migrando definitivamente para assinatura. Quem já tinha uma licença perpétua podia continuar usando o software para sempre — essa foi a promessa que acompanhou o anúncio. Só que, menos de dez anos depois, a empresa encerrou também o programa de manutenção que permitia a esses clientes atualizar suas licenças antigas, fechando a última porta que restava para quem tinha comprado o direito de possuir o próprio software, e não apenas de alugá-lo mês a mês.

Repare no padrão: a promessa feita no momento da transição (“quem já comprou continua tendo o que comprou”) nunca é a promessa que sobrevive à década seguinte. O parafuso do licenciamento, uma vez apertado na direção da assinatura, nunca volta para trás. Ele só aperta mais.

Por que a letra do contrato vale mais que a promessa

Há uma lição aqui que vai muito além do CAD, e ela é desconfortável para quem gosta de confiar na boa vontade das empresas com quem faz negócio. A promessa da Autodesk, em 2016, era provavelmente sincera no momento em que foi feita. O problema não foi má-fé; foi que promessas verbais, declarações de intenção e garantias de palco não têm valor executável. O que tem valor é o que está escrito no contrato — e o que estava escrito permitia, anos depois, encerrar o programa de manutenção sem violar cláusula nenhuma.

Essa é uma intuição profundamente cara à tradição que valoriza o contrato como base das relações livres entre indivíduos.[2] A liberdade de contratar é

[2] A centralidade do contrato voluntário como fundamento das relações livres percorre toda a tradição liberal, de John Locke à teoria contemporânea dos contratos. O ponto prático para o profissional é atribuído com frequência à sabedoria jurídica popular: um acordo verbal “não vale o papel em que não está escrito”. A liberdade de contratar protege sobretudo quem exerce a responsabilidade correspondente de compreender aquilo a que se vincula.

uma das expressões mais concretas da autonomia individual: duas partes, em condições que ambas aceitam, definem os termos da própria relação. Mas essa liberdade só protege quem a leva a sério — quem lê a letra, quem entende que a única promessa que vale é a que se pode cobrar, e que "confie em nós" nunca é um termo de contrato. O profissional que assina uma assinatura de software baseado no que o vendedor disse no palco, e não no que está no documento, não está sendo traído pela empresa quando a promessa evapora. Está colhendo a consequência de ter terceirizado a própria vigilância. A defesa da liberdade, aqui, começa num gesto tão banal quanto ler o que se assina — e tão raro quanto isso.

O que muda com a IA — e o que não muda

Chegamos ao ponto deste capítulo. Nada do que descrevi até aqui — a dependência de infraestrutura de um parceiro maior, o formato de arquivo fechado disfarçado de padrão de mercado, a assinatura substituindo a posse — precisou de inteligência artificial para existir. Esse é um padrão de quase cinco décadas, presente desde o primeiro ano de vida da própria empresa que hoje lidera a conversa sobre IA generativa na engenharia.

O que a inteligência artificial mudou não foi o padrão. Foi o ritmo e o preço de saída dele. Um formato de arquivo fechado é inconveniente, mas ainda é possível converter, negociar, contornar — a Open Design Alliance existe há décadas fazendo exatamente isso. Uma assinatura cara é dolorosa, mas ainda é possível trocar de fornecedor, é possível voltar para uma versão anterior, é possível, em último caso, aprender a viver sem o recurso mais novo. Uma inteligência artificial agentic, integrada fundo na infraestrutura de simulação, orquestrando milhares de "companheiros virtuais" ao mesmo tempo dentro de uma plataforma que você não controla — essa dependência é de outra natureza. Ela não trava seu arquivo. Ela treina, dia após dia, seu próprio julgamento profissional para funcionar melhor dentro de um ambiente específico, tornando cada vez mais caro, em tempo e competência reaprendida, o dia em que você decidir sair.

É essa diferença de natureza — e não só de grau — que o resto deste livro vai explorar. No próximo capítulo, vamos colocar números na cadeira: o que exatamente custa, para um profissional ou uma empresa pequena, ficar dentro de uma dessas plataformas versus construir a própria stack fora dela — e por que essa conta, historicamente, quase nunca é feita antes que seja tarde demais para trocar de rota sem dor.

Fontes: FundingUniverse e Encyclopedia.com, história corporativa da Dassault Systèmes; Dassault Aviation, cronologia oficial da empresa (1965-1986); Wikipedia e DemystifyingPLM, cronologia de fundação da Dassault Systèmes e do CATIA; Francis Bernard (verbete biográfico), contexto da criação da Dassault Systèmes; Scan2CAD, história da Dassault Systèmes e parceria com IBM; Adobe e totalCAD, história do formato DWG; CADinterop.com, histórico e limitações técnicas de STEP, IGES, Parasolid e do kernel Granite/Creo; 3HTI, dado da PTC sobre média de sistemas CAD por empresa e compatibilidade de arquivos Creo; Engineering.com, história e mercado de kernels geométricos (Parasolid x ACIS); Wikipedia, verbetes de Parasolid e Solid Edge; fxguide, CADnotes, Pentagon Solutions e Scott & Scott LLP, cobertura da transição da Autodesk para assinatura (2016) e fim do programa de manutenção para licenças perpétuas.

Para levar deste capítulo

Tranca 1 — Dados. O cercamento nunca foi feito pelos programas, e sim pelos formatos: é o arquivo que só uma ferramenta abre que prende você, não a interface.

Faça agora — 20 minutos. Abra o contrato ou os termos de licença da ferramenta que você mais usa e procure o que acontece com os seus arquivos no dia em que você parar de pagar. Se não achar a cláusula, essa ausência já é a resposta.

Capítulo 3 — A Economia da Dependência

No fim do capítulo anterior, prometi colocar números na cadeira: quanto custa, de fato, ficar dentro de uma plataforma fechada versus construir a própria stack fora dela. Antes dos números, porém, vale nomear o fenômeno com precisão — porque ele tem nome, tem trinta anos de literatura de economia por trás, e entender a mecânica dele é o que separa reclamar da gaiola de saber exatamente onde ela aperta.

Um nome de trinta anos

Em 1998, os economistas Carl Shapiro e Hal Varian publicaram *Information Rules*, um dos livros mais influentes já escritos sobre a economia da informação. Nele, cunharam — ou popularizaram, com rigor acadêmico — o termo “lock-in”: o fenômeno pelo qual o custo de trocar de fornecedor de tecnologia se torna alto o suficiente para prender o cliente no lugar, mesmo quando ele preferiria sair. A frase que resume o livro inteiro é simples e incômoda: lock-in pode ser fonte de dor de cabeça ou de lucro extraordinário, dependendo de você estar do lado de dentro da cela ou segurando a chave da porta.

O interessante é que Shapiro e Varian não trataram lock-in como um único fenômeno, mas catalogaram diferentes tipos dele — contratual, de equipamento durável, de base instalada, de fornecedor especializado. Um deles, em particular, é praticamente profético para o argumento deste livro: o que os dois chamaram, com uma ironia que soa quase de ficção científica pra um livro de 1998, de lock-in de “wetware” — o tempo, o treinamento e a familiaridade que uma pessoa investe para operar bem uma ferramenta específica. Não é o software que prende você. É o seu próprio cérebro, treinado para pensar em termos daquela interface, daquele fluxo de trabalho, daquele jeito específico de resolver um problema.

Trinta anos antes de qualquer Virtual Companion existir, dois economistas já sabiam que a forma mais eficaz de prender um profissional não é travar o arquivo dele. É treinar o profissional.

O custo de trocar, em números reais

Em 2026, esse custo deixou de ser abstrato. Pesquisas do setor de tecnologia empresarial estimam que uma migração completa fora de um fornecedor de tecnologia trava, em média, algo próximo de 315 mil dólares por projeto — e esse número vem de um caso real, não de uma estimativa de laboratório. Depois do colapso da Builder.ai, uma plataforma de desenvolvimento de software que chegou a valer 1,3 bilhão de dólares e tinha a Microsoft entre seus investidores, uma fabricante identificada como NexGen Manufacturing precisou migrar 40 fluxos de trabalho baseados em inteligência artificial para uma nova plataforma às pressas. A conta bateu exatamente essa marca — 315 mil dólares, três meses inteiros de trabalho de engenharia, com funcionalidades de atendimento ao cliente degradadas ou fora do ar durante a transição.

Vale demorar um segundo nesse caso, porque ele é o pesadelo que este livro tenta te ajudar a evitar sem precisar de trauma prévio para acreditar nele. Uma empresa não decidiu, de forma deliberada e informada, apostar seu fluxo de trabalho inteiro na sobrevivência eterna de um único fornecedor de IA. Ela simplesmente foi construindo, integração após integração, uma dependência que nunca apareceu como uma decisão única e visível — apareceu como uma centena de pequenas conveniências, cada uma perfeitamente razoável isoladamente. Até a empresa fornecedora quebrar, e a conta de todas aquelas pequenas conveniências chegar de uma vez.

E não pense que isso é problema só de startups de inteligência artificial, distante do mundo sólido do CAD mecânico. Pergunte a quem usava o SpaceClaim. Lançado em 2007, o SpaceClaim foi um dos modeladores de modelagem direta mais elogiados de sua geração — leve, intuitivo, querido por engenheiros que odiavam a rigidez da modelagem paramétrica baseada em histórico. Foi comprado pela Ansys em 2014 e, por uma década, seguiu vivo. Até que, em 2025, simplesmente deixou de existir como produto autônomo: foi absorvido, expirado, substituído pelo Ansys Discovery. Quem tinha construído seu fluxo de trabalho em torno daquela ferramenta específica não tem mais a ferramenta. Não porque ela ficou ruim, não porque apareceu algo melhor, mas porque a empresa dona dela decidiu, unilateralmente, que o produto tinha cumprido seu papel na estratégia dela. O engenheiro individual não foi consultado. Ele nunca é.

Esse é o ponto que liga o colapso de uma startup de IA à descontinuação de um CAD consagrado: em ambos os casos, a vida útil da sua ferramenta de trabalho nunca esteve nas suas mãos. Ela sempre foi uma decisão de outra

pessoa, tomada numa sala onde você não estava.

Efeitos de rede: quando a escolha nunca foi realmente sua

Lock-in por custo de troca é metade da equação. A outra metade é o efeito de rede — o fenômeno pelo qual o valor de uma ferramenta para você depende de quantas outras pessoas já a usam. Telefone, e-mail, aplicativo de mensagem: quanto mais gente usa, mais valiosa a ferramenta fica para cada usuário individual, e mais caro fica, socialmente, ser a pessoa que insiste em usar outra coisa.

No mundo da engenharia, esse efeito raramente é uma escolha livre — ele é imposto de cima para baixo pela cadeia de fornecimento. Quem trabalha com ferramentaria para a indústria automotiva conhece bem essa realidade: não basta dominar um pacote CAD, porque cada montadora ou cada cliente de peso na cadeia decide, unilateralmente, em qual formato nativo ela quer receber o projeto. O resultado prático é que oficinas de ferramentaria pequenas e médias frequentemente precisam manter licenças de dois, três, às vezes quatro sistemas CAD diferentes — não porque escolheram, mas porque cada cliente grande já escolheu por elas, e a alternativa é perder o contrato. O efeito de rede aqui não beneficia o pequeno fornecedor: ele apenas transfere o custo do lock-in do topo da cadeia para a base dela.

Poder de mercado não é a mesma coisa que coerção

Preciso fazer aqui uma distinção que me custa, porque ela me obriga a segurar a própria retórica — e um livro que defende liberdade não pode ser desonesto justamente sobre o que significa liberdade.

É tentador descrever a montadora que obriga a ferramentaria a manter quatro licenças de CAD como um “coercitor”, e o lock-in como uma forma de “coerção”. Mas isso seria impreciso, e a imprecisão enfraqueceria todo o argumento do livro. Coerção, no sentido estrito, é o uso da força ou da ameaça de força para tirar de alguém uma escolha — é o que um assaltante faz, é o que um Estado faz quando ameaça prender quem não paga um imposto. A montadora não faz isso. Ela não aponta arma nenhuma para a ferramentaria. Ela apenas oferece um contrato em condições que a ferramentaria é livre para

recusar — perdendo o contrato, é verdade, mas livre.[3]

Por que essa distinção importa tanto? Porque é ela que separa um problema que o mercado pode resolver de um que só a força poderia resolver. Se o lock-in fosse coerção de verdade, a única saída seria alguém mais forte, o Estado, intervindo para quebrá-lo. Mas o lock-in não é coerção: é poder de mercado, construído sobre dependência real, e por isso ele pode ser dissolvido por outro caminho, sem força nenhuma. Dissolve-se quando surge uma alternativa boa o bastante para que "recusar o contrato" deixe de ser um sacrifício. Foi assim que o e-mail aberto dissolveu o poder das redes fechadas dos anos 90. É assim que um CAD livre e maduro pode, com o tempo, dissolver o poder de barganha de quem hoje impõe formatos proprietários à cadeia inteira.

Essa é a diferença mais importante entre a visão que este livro defende e as respostas mais comuns ao poder das grandes plataformas. A resposta reflexiva costuma ser: "chamem o regulador". A resposta que proponho é outra: construam a alternativa. Não porque regulação seja sempre errada (voltarei a essa tensão em outros capítulos, especialmente quando o caso for concreto), mas porque a alternativa construída não pede licença a ninguém, não depende de um tribunal, e não pode ser capturada pelo próprio poder que pretendia limitar. A saída pela liberdade é mais lenta que a saída pela força. Mas é a única que não troca um mestre por outro.

Portabilidade não é a mesma coisa que interoperabilidade

Isso nos leva de volta ao Capítulo 2, e a uma distinção que vale grifar: interoperabilidade é a sua ferramenta conseguir conversar com outras — via STEP, via IGES, via qualquer formato neutro. Portabilidade é você conseguir, de fato, sair de onde está e levar tudo o que importa junto. As duas coisas parecem sinônimas e não são. Um sistema pode interoperar perfeitamente bem com o resto do mercado e, ainda assim, ser dolorosamente difícil de abandonar — porque o que te prende, na prática, raramente é a geometria do arquivo. É a árvore de histórico paramétrico construída ao longo de anos, são os plug-ins configurados exatamente do seu jeito, é a equipe inteira que aprendeu a pensar dentro daquela interface específica. Volte à ideia do "wetware" de Shapiro e Varian: mesmo com

[3] A distinção entre coerção (uso ou ameaça de força) e mero poder de barganha ou influência econômica é central no pensamento libertário, e foi articulada com particular rigor por autores como Friedrich Hayek em *The Constitution of Liberty* (1960) e Robert Nozick em *Anarchy, State, and Utopia* (1974). Reconhecê-la não enfraquece a crítica ao lock-in — apenas indica que o remédio correto é a concorrência e a alternativa, não a força.

o arquivo perfeitamente conversível, o cérebro treinado continua sendo o ativo mais difícil de portar de um sistema para outro.

E aqui cabe uma honestidade que este livro tenta sustentar em todo capítulo: eu mesmo não estou livre disso. As automações que escrevi (a requisição de compra, as macros no Solid Edge) me prendem, um pouco, ao jeito específico como eu resolvi aqueles problemas. Se eu trocasse de ferramenta CAD amanhã, perderia parte do trabalho que investi em automatizar a antiga. Isso não é hipocrisia em relação ao argumento do livro. É a prova de que lock-in não é, em si, o vilão da história.

O lock-in perigoso é o escondido, não o explícito

A frase que resume o argumento deste capítulo não é minha — encontrei ela quase pronta numa análise recente sobre computação em nuvem, e ela captura exatamente a distinção que separa dependência saudável de dependência predatória: lock-in se torna perigoso quando está escondido, não quando é explícito e tem preço claro.

Minhas macros me prendem um pouco ao meu próprio jeito de trabalhar — mas eu sei exatamente o tamanho dessa prisão, porque fui eu quem escreveu o código, e eu poderia reescrevê-lo em outro lugar se precisasse, com esforço mensurável. Sei disso porque já fiz: comecei essas automações em Python e depois as reescrevi em C#, com ajuda da inteligência artificial, quando quis torná-las mais robustas — lock-in que eu escolhi, medi e paguei de olhos abertos. Isso é lock-in explícito, escolhido, precificável. É completamente diferente de comprar uma licença de CAD que, sem aviso, vem com serviços de nuvem conectados por padrão a uma plataforma maior — como vimos no Capítulo 1 acontecer com o SOLIDWORKS desde 2023. Ou de adotar um assistente de IA integrado tão fundo no fluxo de trabalho que ninguém na empresa consegue, seis meses depois, dizer com precisão quantos processos internos passaram a depender silenciosamente dele. O problema nunca foi a dependência em si. É a dependência que ninguém escolheu conscientemente, e que só revela seu tamanho real no dia em que fica cara demais para desfazer.

A IA empilha uma camada nova — e mais difícil de medir

Análises recentes do setor de tecnologia empresarial apontam algo desconfortável sobre 2026: a diferença de capacidade entre as principais

plataformas de IA corporativa hoje é, segundo essas análises, ainda maior do que a diferença que existia entre a Oracle e alternativas de banco de dados no auge do domínio da Oracle — uma das eras de lock-in mais lucrativas e mais estudadas da história recente do software. E, ao contrário das ondas anteriores de dependência tecnológica, o lock-in de IA se acumula entre fornecedores diferentes ao mesmo tempo: sair de uma ferramenta de IA corporativa frequentemente significa reavaliar integrações inteiras com outras duas ou três plataformas, porque os fluxos de dados passam, de um jeito ou de outro, pela mesma infraestrutura de fundo.

Um framework recente de auditoria de dependência de IA corporativa sugere avaliar quatro frentes antes de adotar qualquer ferramenta desse tipo: onde os dados da empresa realmente residem e o quanto são portáveis; quais processos de negócio já foram silenciosamente reformulados para depender das respostas da IA; quantas configurações específicas daquele fornecedor já foram acumuladas; e (a mais difícil de medir, a mais cara de reverter) o quanto o trabalho da equipe já foi remodelado em torno daquela ferramenta específica, e quanto custaria retreinar todo mundo num substituto.

Repare: a quarta frente é, com outras palavras, exatamente o lock-in de "wetware" que Shapiro e Varian descreveram em 1998, quase trinta anos antes da inteligência artificial generativa existir. A tecnologia mudou. O mecanismo mais eficaz de captura continua sendo o mesmo: não travar o seu arquivo, travar o seu hábito.

É um modelo de negócio racional — e por isso a responsabilidade não é da empresa

Vale dizer isso sem rodeio, porque é o ponto mais importante deste capítulo: nenhuma das empresas descritas neste livro está fazendo algo irracional, cínico ou sequer incomum ao construir lock-in dentro do próprio produto. Do ponto de vista de quem administra uma empresa de tecnologia, capturar o cliente é literalmente o trabalho — é a diferença entre um negócio com margem sustentável e um negócio que qualquer concorrente pode replicar da noite para o dia. A literatura de estratégia empresarial trata isso como a construção de um "fosso" competitivo, e ninguém no conselho de administração de uma companhia de capital aberto é pago para deixar esse fosso mais raso por gentileza com o usuário final.

É exatamente por isso que este livro não gasta energia esperando que a Dassault, a Autodesk, a Microsoft ou qualquer outra fornecedora decida, um

dia, parar de otimizar o próprio lock-in. Elas não vão parar, porque parar seria irracional dentro da própria lógica que as fez chegar onde chegaram. A responsabilidade de reconhecer onde a dependência é explícita e escolhida, e onde ela é escondida e imposta, nunca foi delas. É sua. É minha. É de cada profissional que abre um contrato de assinatura sem ler a cláusula sobre para onde vão os próprios dados quando o contrato acabar.

Três perguntas antes de assinar qualquer coisa

Este livro não é um manual, e o Capítulo 14 vai se dedicar inteiro a como montar uma stack livre na prática. Mas seria desonesto fechar o Diagnóstico sem entregar ao menos uma ferramenta que você pode usar hoje, antes de virar a página. Toda a economia da dependência descrita neste capítulo pode ser resumida em três perguntas que qualquer profissional deveria fazer antes de adotar qualquer ferramenta nova — CAD, IA, nuvem, o que for:

A primeira: se eu quiser sair daqui em três anos, o que exatamente eu consigo levar comigo? Não a resposta de marketing, mas a técnica — meus arquivos saem em formato que outro sistema abre sem perder a inteligência do projeto, ou saem como geometria morta? Se a resposta for vaga, o lock-in já está escondido ali.

A segunda: o poder que essa ferramenta me dá é proporcional à dependência que ela cria? Uma macro que eu escrevo me dá muito poder e cria pouca dependência, porque eu controlo o código. Um assistente que resolve tudo por mim, mas só funciona dentro de uma plataforma que eu não controlo, inverte essa proporção — muito conforto, muita dependência, pouco poder real que seja meu.

A terceira, e a mais incômoda: quem decide quando essa ferramenta deixa de existir, ou quando ela muda de preço? Se a resposta for “outra pessoa, numa sala onde eu não estou” — e, para toda plataforma proprietária, essa é sempre a resposta —, então a pergunta seguinte é apenas: quanto da minha capacidade de trabalhar eu estou disposto a colocar nas mãos dessa decisão que não é minha?

Nenhuma dessas perguntas exige que você recuse a ferramenta. Elas só exigem que você faça a escolha de olhos abertos, com o custo do lock-in visível na mesa, ao lado do valor do recurso — que é, no fim, a única diferença entre dependência escolhida e dependência sofrida.

O que vem a seguir

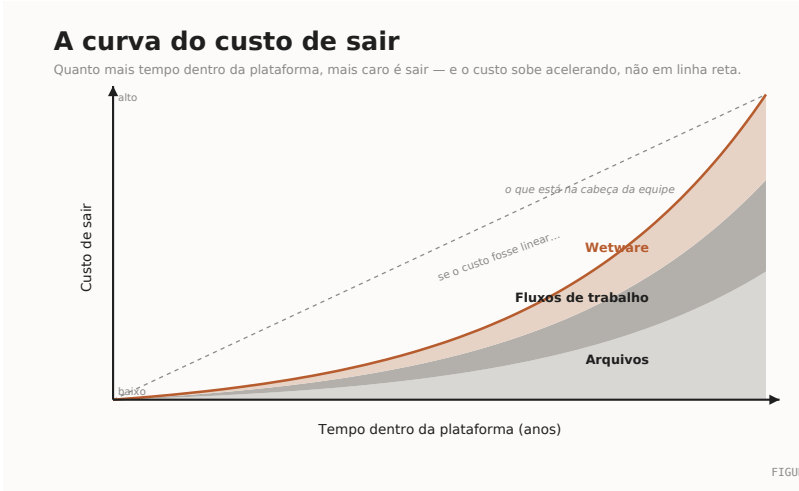
Este capítulo fechou o Diagnóstico. Você já sabe, a esta altura, como a gaiola é construída (Capítulo 1), há quanto tempo esse padrão existe (Capítulo 2) e por que mecanismo econômico exato ele funciona (este capítulo). O que falta agora é a parte que este livro promete desde o prólogo: mostrar que existe alternativa real, não hipotética, sendo construída agora, em paralelo, por gente que decidiu que dependência escondida não é a única forma de ter acesso a ferramentas poderosas. No próximo capítulo, começamos a Parte II, o Rompimento, com o protocolo que, em pouco mais de um ano, começou a provar exatamente isso.

Fontes: Carl Shapiro e Hal R. Varian, "Information Rules: A Strategic Guide to the Network Economy" (Harvard Business School Press, 1998/1999), conforme sintetizado por HBS Working Knowledge e materiais didáticos derivados; Kong Inc. e Swfte AI, análises sobre custo de vendor lock-in em IA corporativa (2026); Selleo e DigitalDefynd, relatórios sobre lock-in em computação em nuvem (2026); Medium/John Garny, distinção entre portabilidade e interoperabilidade (mai/2026); Vaasblock, framework de auditoria de dependência de IA corporativa (2026); Wikipedia e Ansys Learning Forum, descontinuação do SpaceClaim e substituição pelo Ansys Discovery (2025).

Para levar deste capítulo

Tranca 1 — Dados. O custo de sair sobe com o tempo, e a maior parte dele não está nos arquivos: está nos fluxos e no que a sua equipe tem na cabeça.

Faça agora — 30 minutos. Pegue um projeto recente, exporte-o para um formato neutro, e abra o resultado em outra ferramenta. Anote exatamente o que se perdeu no caminho. Essa lista é o tamanho real da sua dependência.


A curva do custo de sair
Quanto mais tempo dentro da plataforma, mais caro é sair — e o custo sobe acelerando, não em linha reta.
alto
baixo
Custo de sair
o que está na cabeça da equipe
se o custo fosse linear...
Wetware
Fluxos de trabalho
Arquivos
Tempo dentro da plataforma (anos)
FIGURA D3

Interlúdio I — A Gaiola Brasileira

Este livro faz três pausas. Em cada uma delas, o argumento para de avançar e responde a uma objeção legítima do leitor — daquelas que, se ficarem sem resposta, tornam todo o resto inútil. A primeira é a mais concreta de todas: "*tudo bem, mas eu estou no Brasil*".

Os três primeiros capítulos deste livro descreveram uma gaiola que existe no mundo inteiro. As plataformas, os formatos proprietários, a economia da dependência — tudo isso vale igualmente para um engenheiro em Stuttgart, em Detroit ou em Piracicaba.

Mas seria desonesto seguir adiante fingindo que a gaiola tem a mesma espessura em todo lugar. Se você é engenheiro no Brasil, há uma camada a mais em volta de você, e ela não aparece em nenhum livro americano sobre autonomia profissional. Um autor que escreve de São Francisco pode dizer "largue o emprego e monte o seu negócio" sem mencionar que, no país dele, abrir uma empresa leva um dia e uma tarde. Aqui não é assim. E fingir que é seria o tipo de conselho bonito que quebra na primeira semana de aplicação.

Este interlúdio existe para colocar essa camada na mesa, com números, antes de o livro seguir para a saída. Não como lamento, lamento não conserta nada, mas porque a realidade brasileira **muda a estratégia**, e não apenas o humor. Vou argumentar, no fim, que ela torna a tese central deste livro mais urgente aqui do que em qualquer outro lugar.

O funil

Comecemos pela dor mais antiga e mais silenciosa: a distância entre se formar engenheiro e trabalhar como engenheiro.

O Brasil tem cerca de 1,2 milhão de engenheiros registrados. Desses, segundo levantamento da Mútua, ligada ao sistema Crea, 18,1% estão simplesmente fora do mercado — têm registro profissional, mas não

trabalham com carteira assinada nem como empreendedores.[4] E o número mais citado da área, embora venha de um levantamento da Confederação Nacional da Indústria já com alguns anos, continua sendo repetido porque ninguém consegue desmenti-lo: mais da metade dos engenheiros formados no Brasil não atua na engenharia — algo em torno de 680 mil pessoas que estudaram cinco anos de cálculo, resistência dos materiais e termodinâmica para acabar em outro lugar.[5] A evasão nos cursos, antes disso, já ronda os 30%.[6]

Estes números descrevem, em linguagem estatística, uma cena que qualquer um de nós já viu de perto: o sujeito que se formou, pendurou o diploma na parede e foi vender plano de saúde. Ou virou analista de dados. Ou entrou numa fábrica em função administrativa. Ou — e esta é a mais comum no meu mundo — foi ser desenhista, com salário de desenhista, fazendo trabalho de engenheiro.

A leitura preguiçosa disso é culpar o sujeito. A leitura honesta é outra: o diploma, no Brasil, deixou de ser uma chave. É, na melhor das hipóteses, uma carta de apresentação. E há um dado que aponta para onde o problema mora: mais da metade dos engenheiros que não atuam na área se formou em cursos com as piores notas na avaliação oficial do ensino superior.[7] Não é que a engenharia não empregue. É que ela não emprega quem só tem o papel.

O paradoxo do apagão

Agora colem esse quadro ao lado de outro, e vejam o absurdo que aparece.

Enquanto centenas de milhares de engenheiros formados não conseguem entrar na profissão, o setor produtivo passa o dia inteiro dizendo que **faltam engenheiros**. Em fevereiro de 2026, a entidade que representa a indústria da

[4]Estudo da Mútua (Caixa de Assistência dos Profissionais do Crea), publicação técnica “O Futuro das Engenharias no Brasil”: dos cerca de 1,2 milhão de engenheiros registrados no país, 18,1% estão fora do mercado — com registro profissional, mas sem trabalho formal nem atividade empreendedora.

[5]Levantamento da Confederação Nacional da Indústria (CNI), amplamente citado: aproximadamente 680 mil engenheiros formados não atuavam na área, cerca de 58% do total. O dado é de 2014 e deve ser lido com essa ressalva de data; permanece, contudo, a referência mais repetida do setor, e nenhum levantamento posterior indicou reversão significativa da tendência.

[6]Estimativas de evasão em torno de 30% nos cursos de engenharia no Brasil (Engenharia 360, a partir de dados do setor educacional).

[7]Estudo da Mútua/Crea: mais da metade dos engenheiros que não atuam na área formou-se em cursos com nota 1 ou 2 no Exame Nacional de Desempenho dos Estudantes (Enade); apenas 6% vieram de instituições de alto desempenho (notas 4 ou 5).

construção pesada declarou que o país vive um “alerta amarelo” na formação de engenheiros, capaz de comprometer a execução de obras de infraestrutura nos próximos anos — o Brasil forma cerca de 40 mil engenheiros por ano, enquanto a China forma quinhentos mil.[8] Em março de 2026, num fórum da engenharia automotiva, o diretor-presidente do IPT alertou para um possível apagão de mão de obra técnica justamente quando a demanda global por tecnologia acelera, puxado pela queda consistente do interesse dos jovens pelos cursos da área.[9]

Como é possível sobrar engenheiro e faltar engenheiro ao mesmo tempo?

Só há uma resposta que fecha a conta, e ela é a coisa mais importante deste interlúdio: **o mercado nunca quis “engenheiros”. O mercado quer pessoas que resolvem problemas difíceis.** O diploma genérico é abundante; a capacidade específica é escassa. Sobra quem tem o título e falta quem tem o domínio — e as duas coisas, no Brasil, se descolaram tanto que hoje convivem no mesmo país sem se encontrar.

Guarde essa frase, porque ela é a ponte entre este interlúdio e o resto do livro. Se o que falta é capacidade específica, e não credencial, então a estratégia de sobrevivência do engenheiro brasileiro não é acumular mais um diploma. É construir domínio real sobre problemas que pouca gente resolve — e sobre as ferramentas com que se resolve. Que é, palavra por palavra, o que este livro defende.

A hierarquia silenciosa

Há uma segunda camada da gaiola brasileira, e essa eu conheço na pele, porque é a minha: a hierarquia de cargos que separa quem faz do que ganha.

No Brasil, o mesmo trabalho técnico recebe nomes diferentes, e os nomes valem dinheiro. Pelos dados de admissões e desligamentos formais dos últimos doze meses, um desenhista projetista mecânico ganha, em média, cerca de R$ 4,9 mil. Um projetista mecânico, perto de R$ 7,9 mil. Um engenheiro projetista, cerca de R$ 9,7 mil — numa faixa que vai de um

[8]Declaração do diretor executivo do Sinicon (Sindicato Nacional da Indústria da Construção Pesada), em entrevista de fevereiro de 2026: o Brasil forma cerca de 40 mil engenheiros por ano, contra cerca de 500 mil na China, configurando um “alerta amarelo” para a capacidade de execução de obras de infraestrutura.

[9]Alerta do diretor-presidente do IPT (Instituto de Pesquisas Tecnológicas) e professor titular do ITA, no 1º Fórum Estratégico da Associação Brasileira de Engenharia Automotiva (AEA), março de 2026, sobre o risco de apagão de mão de obra em engenharia diante da queda de interesse pelos cursos da área.

piso de aproximadamente R$ 3 mil a um teto acima de R$ 15 mil.[10] Repare no espalhamento: o teto do engenheiro projetista é cinco vezes o piso do mesmo cargo. Não é a função que define o valor. É outra coisa.

E há um número, dentro desses levantamentos, que me pareceu um soco quando o li: na comparação entre janeiro e dezembro de 2025, as contratações formais para desenhista projetista de máquinas caíram mais de 72%.[11] Setenta e dois por cento em um ano, numa única categoria. Não é ciclo, não é sazonalidade — é um chão desaparecendo debaixo dos pés de gente que está trabalhando agora, desenhando máquinas, achando que o emprego é estável porque tem carteira assinada.

Este é o ponto em que a gaiola brasileira encontra a gaiola global do primeiro capítulo. O profissional que passou vinte anos sendo excelente em operar um software que outra pessoa fez, dentro de uma função que outra pessoa nomeou, dentro de uma empresa que outra pessoa dirige, descobre, quando a categoria inteira encolhe 72%, que ele não tinha uma carreira. Tinha uma cadeira. E cadeiras são retiradas.

O chão que encolhe

Por que a cadeira está sendo retirada? Porque o chão em que ela estava apoiada vem encolhendo há quarenta anos.

O número é conhecido, mas ele nunca deixa de impressionar: a indústria de transformação, que em 1985 respondia por 35,9% do PIB brasileiro, chegou a 11,8% em 2025.[12] Em quatro décadas, o setor que emprega engenheiros, que compra moldes, que precisa de projetistas, encolheu para um terço do que era na formação da riqueza nacional. Não é uma crise; é uma erosão longa, atravessando governos de todas as cores, sem que nenhum a tenha revertido.

E há o efeito de superfície, o que aparece na porta da fábrica. Em 2025,

[10] Médias salariais calculadas pelo Portal Salário a partir de dados do CAGED/Ministério do Trabalho, referentes a admissões e desligamentos em regime CLT nos doze meses anteriores a 2026: desenhista projetista mecânico, cerca de R$ 4.935; projetista mecânico, cerca de R$ 7.865; engenheiro projetista, cerca de R$ 9.661, com piso médio próximo de R$ 3.008 e teto próximo de R$ 15.842. Valores de salário base, sem adicionais.

[11] Portal Salário/CAGED: queda superior a 72% nas contratações formais em regime integral para a categoria de desenhista projetista de máquinas (CBO 3186-05) na comparação entre janeiro e dezembro de 2025.

[12] Dados da Firjan, citando a série histórica da participação da indústria de transformação no PIB: de 35,9% em 1985, o maior nível da série, para 11,8% em 2025. O processo de desindustrialização brasileira é datado pela literatura econômica como tendo início na década de 1980.

cerca de três milhões de empresas encerraram atividades no Brasil.[13] Nomes grandes viraram manchete por alguns dias e sumiram: a Ford, que fechou Camaçari, Taubaté e Horizonte em 2021, depois de mais de um século de país; a Sony, no mesmo ano, levando a linha de eletrônicos embora. Em julho de 2026, enquanto eu escrevia este livro, a Isover, do grupo Saint-Gobain, encerrou a produção de sua fábrica em Santo Amaro, na zona sul de São Paulo, depois de mais de setenta anos, transformando o que era uma unidade industrial em centro de distribuição.[14]

Reparem no padrão dessa última: a empresa não foi embora do Brasil. Ela apenas parou de *fabricar* aqui. Continua vendendo, distribuindo, faturando. O que saiu foi a engenharia — o pedaço da operação que precisa de projetista, de ferramentaria, de quem sabe fazer. É o retrato exato do que a desindustrialização significa para a nossa profissão: o país continua consumindo produtos manufaturados; ele só deixa, aos poucos, de ser o lugar onde alguém pensa como fazê-los.

O peso invisível

Falta nomear a força que empurra esse chão para baixo, e aqui eu preciso ser ao mesmo tempo direto sobre a minha opinião e honesto sobre o debate.

O setor produtivo tem um nome para essa força: Custo Brasil — o conjunto de ineficiências estruturais que torna produzir aqui mais caro do que produzir em quase qualquer lugar comparável. Ele foi medido. Segundo o estudo do Movimento Brasil Competitivo em parceria com o Ministério do Desenvolvimento, Indústria, Comércio e Serviços, produzir e operar no Brasil custa cerca de **R$ 1,7 trilhão a mais por ano** do que custaria na média dos países da OCDE — algo em torno de 20% do PIB.[15] Vinte por cento de tudo o que o país produz, queimado em atrito.

---

[13] Levantamentos de 2026 sobre encerramento de empresas: cerca de três milhões de empresas encerraram atividades no Brasil em 2025.

[14] Encerramento da produção da fábrica da Isover (grupo Saint-Gobain) em Santo Amaro, São Paulo, em julho de 2026, após mais de setenta anos de operação, com a conversão da unidade em centro de distribuição, conforme acordo firmado com o Ministério Público e a Cetesb. A Ford encerrou sua produção no Brasil em 2021, fechando as fábricas de Camaçari, Taubaté e Horizonte; a Sony encerrou sua linha de eletrônicos no país no mesmo ano.

[15] Observatório do Custo Brasil, estudo do Movimento Brasil Competitivo (MBC) em parceria com o MDIC: produzir e operar no Brasil custa cerca de R$ 1,7 trilhão a mais por ano do que na média dos países da OCDE, equivalente a aproximadamente 20% do PIB. A primeira medição, em 2019, apontava R$ 1,5 trilhão. Pesquisa da CNI aponta que 45% dos empresários industriais consideram a tributação e sua complexidade a maior dificuldade para competir no exterior.

Mas o dado que mais fala a um engenheiro não é esse. É este: as empresas brasileiras gastam, em média, **1.500 horas por ano apenas para cumprir obrigações tributárias.** Nos países desenvolvidos, essa média é de 160 horas.[16] Mil e trezentas e quarenta horas de diferença. Faça a conta em cabeças: é quase uma pessoa em tempo integral, o ano inteiro, produzindo exatamente zero — nenhum produto, nenhum projeto, nenhuma peça —, só preenchendo papel para o Estado. Nós, que passamos a carreira caçando segundos de ciclo numa injetora e milímetros de curso num molde, deveríamos ser os primeiros a enxergar o tamanho obsceno desse desperdício. É o maior gargalo de produtividade do país, e ele não está na fábrica.

A minha leitura, e não escondo que ela vem da minha visão de mundo, é que boa parte desse peso é escolha política, não fatalidade geográfica. Um Estado que consome uma fatia enorme do que se produz, que legisla mais rápido do que qualquer empresa consegue se adaptar, e que trata quem produz como suspeito até prova em contrário, produz exatamente o resultado que estamos vendo: a engenharia migrando para onde é mais barato pensar.

E agora a honestidade que este livro se impôs desde o primeiro capítulo. Não seria correto atribuir a desindustrialização brasileira *apenas* ao Estado, e quem faz isso está simplificando um fenômeno que os economistas debatem há décadas sem consenso. Pesaram também a abertura comercial dos anos 90 combinada com câmbio sobrevalorizado; a ascensão industrial chinesa, que desmontou cadeias no mundo inteiro, não só aqui; o custo do capital; a infraestrutura logística; e (como vimos no dado sobre a qualidade dos cursos) a nossa própria educação técnica. A carga tributária e a burocracia são, na minha leitura, a maior parcela e a mais facilmente corrigível. Mas não são a única, e um livro que se apresenta como honesto precisa dizer isso mesmo quando seria mais confortável não dizer.

Por que isto muda a estratégia, e não só o humor

Terminaria aqui, se este fosse um livro de diagnóstico. Mas é um livro de saída, e a pergunta que importa é: o que um engenheiro brasileiro faz com essas informações?

[16] Segundo conselheiro executivo do Movimento Brasil Competitivo, empresas brasileiras gastam em média 1.500 horas por ano para cumprir obrigações tributárias, contra uma média de 160 horas nos países mais desenvolvidos. Dos doze eixos que compõem o Custo Brasil, seis concentram cerca de 80% do impacto: acesso a capital, mão de obra qualificada, logística, energia, regulação e sistema tributário.

A resposta é que o cenário brasileiro **inverte o cálculo de risco** que a maioria de nós aprendeu em casa. Fomos criados na ideia de que o caminho seguro é o emprego formal numa empresa grande, e o caminho arriscado é fazer por conta própria. Olhe de novo para os números deste interlúdio: três milhões de empresas fechando num ano, uma categoria profissional inteira perdendo 72% das contratações, um setor industrial que virou um terço do que era. A dependência de um único empregador, num país assim, não é o caminho seguro. É apenas o caminho cujo risco você não enxerga até o dia da demissão.

E há quatro implicações práticas que decorrem disso, e que o resto do livro vai desenvolver.

A especialização deixou de ser luxo. Se sobra diploma e falta capacidade, o profissional genérico é exatamente o que o mercado tem de sobra. O que escapa do funil é quem resolve um problema difícil que pouca gente resolve.

A ferramenta que você possui vale mais aqui do que lá fora. Num país onde o capital é caro, o crédito é escasso e a burocracia devora horas, a diferença entre pagar assinaturas caras em dólar e operar com ferramentas abertas que você domina não é ideologia — é margem de sobrevivência. O engenheiro brasileiro tem mais motivos econômicos para buscar autonomia de ferramentas do que o americano que escreveu o manual.

A automação é a resposta ao Custo Brasil na sua escala. Você não vai reformar o sistema tributário. Mas as 1.500 horas de atrito também existem, em versão menor, dentro do seu dia: no retrabalho, na lista de materiais digitada à mão, no orçamento refeito pela quinta vez. Essas você pode eliminar, uma por uma, com o que aprendeu a construir.

E a fronteira ficou porosa. Talvez a implicação mais subestimada de tudo o que este livro discute: com ferramentas abertas, formatos que conversam e protocolos que atravessam qualquer distância, um projetista competente em Piracicaba pode atender um cliente em Munique e receber em moeda forte. A desindustrialização tirou fábricas do país; ela não tirou do engenheiro brasileiro a capacidade de vender engenharia para fora dele. Essa porta, que há vinte anos não existia, hoje está aberta — e quase ninguém aqui reparou.

O que vem a seguir

A gaiola brasileira é real, é mais espessa que a dos outros, e não vai ser destravada por decreto nem por eleição — pelo menos não a tempo de

resolver a sua vida. Essa é a má notícia, e eu preferi te dar ela de frente, com números, a te oferecer otimismo barato.

A boa notícia é a mesma coisa vista do outro lado. Se ninguém vai abrir a gaiola por você, então a única estratégia que sobra é também a única que sempre funcionou: construir, com as próprias mãos, a capacidade de não depender dela. Não por heroísmo, mas por aritmética — porque num país onde três milhões de empresas fecham por ano, a autonomia deixou de ser ambição e virou a forma mais realista de segurança que existe.

É por isso que a segunda parte deste livro, que começa agora, importa mais aqui do que em qualquer outro lugar do mundo. Ela trata da saída: dos protocolos que dissolveram os feudos, do CAD que se bifurcou, da inteligência artificial que virou ponte, e do preço honesto de atravessar por ela. Se você chegou até aqui achando que este era um livro sobre software, o interlúdio deve ter deixado claro que nunca foi. É um livro sobre como um profissional técnico brasileiro constrói um chão que não some debaixo dos seus pés.

Fontes: Mútua/Crea, "O Futuro das Engenharias no Brasil"; CNI; Engenharia 360; Exame (entrevista Sinicon, fev/2026); AEA/IPT (mar/2026); Portal Salário/CAGED (2026); Firjan e IBGE (participação da indústria de transformação no PIB); Observatório do Custo Brasil (MBC/MDIC); noticiário econômico de 2025-2026 sobre encerramento de empresas e fechamento de plantas industriais. As interpretações causais são do autor; onde há debate econômico relevante, ele foi explicitado no texto.

Para levar deste capítulo

Trancas 5 e 6 — Renda e Domínio. No Brasil, essas duas pesam mais do que em qualquer outro lugar: o chão institucional é instável, e o que protege é ter mais de uma fonte e resolver o que poucos resolvem.

Faça agora — 5 minutos. Escreva quantas mãos diferentes podem, sozinhas, cortar mais da metade da sua renda no mês que vem. Se a resposta for "uma", você acabou de identificar a tranca mais urgente da sua vida profissional — e ela não é técnica.

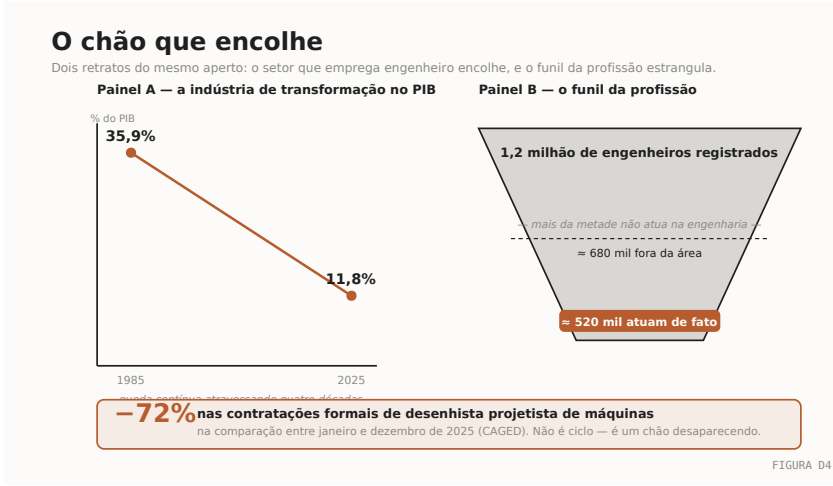

O chão que encolhe
Dois retratos do mesmo aperto: o setor que emprega engenheiro encolhe, e o funil da profissão estrangula.
Painel A — a indústria de transformação no PIB
% do PIB
35,9%
11,8%
1985
2025
queda contínua atravessando quatro décadas
Painel B — o funil da profissão
1,2 milhão de engenheiros registrados
mais da metade não atua na engenharia
≈ 680 mil fora da área
≈ 520 mil atuam de fato
−72%
nas contratações formais de desenhista projetista de máquinas
na comparação entre janeiro e dezembro de 2025 (CAGED). Não é ciclo — é um chão desaparecendo.

FIGURA D4

Parte II

O Rompimento: A Saída e o Seu Preço

Capítulo 4 — Protocolos Abertos e o Fim dos Feudos Digitais

A Parte I deste livro foi o diagnóstico. Mostrei como a gaiola é construída, há quanto tempo ela existe e por que mecanismo econômico exato ela prende. Se o livro parasse aqui, seria só mais um lamento bem documentado sobre o poder das grandes plataformas — e o mundo já tem lamentos demais. A Parte II é sobre a outra metade da história, a que raramente aparece nas manchetes porque não tem palco iluminado nem CEO carismático: a de que, enquanto os feudos se fechavam, alguém estava construindo, em silêncio, as estradas que passam por cima dos muros.

Este capítulo é sobre uma dessas estradas. E, para entender por que ela importa, precisamos primeiro voltar a uma guerra que já foi vencida uma vez — porque a história que estamos vivendo agora, em 2026, já aconteceu antes, com outros nomes.

A guerra que já foi vencida

No começo dos anos 1990, se você queria acessar informação por um computador, você entrava num jardim murado. CompuServe, Prodigy, America Online — cada um era um mundo fechado, com seu próprio conteúdo, seus próprios fóruns, seu próprio e-mail que só falava com gente de dentro do mesmo serviço. A quantidade de informação a que você tinha acesso não dependia da sua curiosidade nem da sua competência: dependia dos acordos comerciais que a CompuServe ou a AOL tinham fechado com seus fornecedores de conteúdo. Você não navegava a informação do mundo. Você navegava o catálogo que uma empresa tinha decidido te vender.

Havia uma alternativa, e ela parecia, na época, uma aposta improvável. Um conjunto de protocolos abertos, o TCP/IP, propunha algo que os provedores comerciais consideravam quase absurdo: a ideia de que qualquer pessoa, em qualquer lugar, pudesse simplesmente se conectar à rede e falar com qualquer outra, sem pedir licença a um ponto central de controle. Os provedores proprietários eram fundamentalmente contra esse conceito, e por um bom motivo comercial: uma rede onde todos falam a mesma língua,

sem intermediário, é uma rede onde ninguém precisa pagar pedágio ao dono do jardim.

O detalhe mais revelador dessa história é o que aconteceu dentro da IBM. A empresa tinha vários anos de vantagem sobre a concorrência na construção de roteadores capazes — a infraestrutura física que faria a internet aberta funcionar. E decidiu não perseguir esse negócio, porque parte da IBM ainda acreditava que as redes proprietárias venceriam a guerra dos protocolos no fim. Estavam redondamente enganados. Quando a NSFNET, construída sobre protocolos abertos, começou a escalar, aconteceu algo que um engenheiro da IBM descreveria depois como inédito na história da computação: pela primeira vez, todos os computadores falavam a mesma língua. Um fabricante que quisesse vender para uma universidade, ou para qualquer instituição que conversasse com uma universidade, era obrigado a colocar TCP/IP em suas máquinas. O protocolo aberto não venceu por decreto. Venceu porque, uma vez que existia, ficava caro demais ficar de fora dele.

Até a Microsoft errou essa aposta. No lançamento original do Windows 95, a empresa ainda apostava que sua rede proprietária, a MSN, poderia se tornar um jardim murado lucrativo — o suporte à internet aberta só veio embutido de forma padrão a partir do primeiro service pack. O maior monopólio de software da época hesitou diante da mesma bifurcação, e apostou, primeiro, no muro. Guarde isso, porque é exatamente a aposta que a Dassault está fazendo em 2026 — só que com IA no lugar de rede, e trinta anos depois de a história já ter dado seu veredicto.

O que é um protocolo, e por que isso liberta

Vale parar um momento numa distinção que parece técnica, mas é, no fundo, filosófica. Um protocolo não é um produto. Ele não pertence a ninguém no sentido em que o SOLIDWORKS pertence à Dassault ou o Windows pertence à Microsoft. Um protocolo é apenas um acordo sobre como duas coisas conversam — uma língua franca, publicada, que qualquer um pode implementar sem pedir permissão. O TCP/IP não tem dono que cobre pedágio. O e-mail, baseado em protocolos abertos como SMTP, não tem uma empresa que decida quem pode mandar mensagem para quem. É por isso que ninguém precisa da autorização de uma corporação para criar um novo cliente de e-mail, um novo site, um novo serviço na internet: a estrada é pública, e a estrada é a mesma para todos.

Essa é a diferença mais importante deste capítulo, e ela ecoa a distinção que

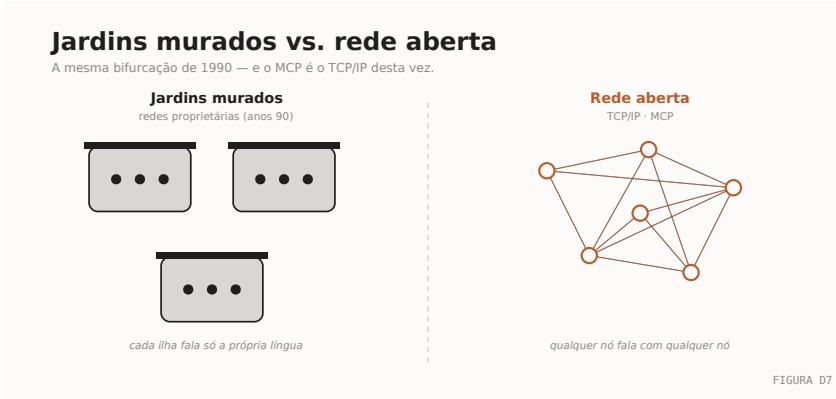
fiz lá no Capítulo 1, sobre o Nemotron da NVIDIA. Uma ferramenta poderosa dentro de um jardim murado te dá poder na condição de que você nunca saia. Um protocolo aberto te dá poder e não pede nada em troca — nem lealdade, nem assinatura, nem a chave da porta. A diferença entre depender de um produto e apoiar-se sobre um protocolo é a diferença entre morar de aluguel num terreno cercado e construir sobre uma estrada pública que ninguém pode fechar.

Entra o MCP

Em novembro de 2024, a Anthropic (a empresa por trás do assistente de IA Claude) publicou um padrão aberto chamado Model Context Protocol, ou MCP. O problema que ele resolve é simples de enunciar e, até então, era um pesadelo de resolver: como fazer qualquer modelo de inteligência artificial conversar com qualquer ferramenta, banco de dados ou arquivo, sem que alguém precise escrever uma integração customizada para cada combinação possível.

O tamanho desse problema tem até nome técnico: o problema N×M. Se você tem 20 modelos de IA e 20 sistemas empresariais para conectar, sem um padrão comum você precisaria de até 400 integrações customizadas, cada uma construída e mantida à mão. Com um protocolo comum, você constrói um servidor MCP para cada recurso uma única vez, e todo cliente compatível com MCP passa a consegui-lo usar. A analogia que se popularizou é a do “USB-C para a IA”: um conector universal que permite plugar qualquer inteligência artificial em qualquer ferramenta, sem fiação sob medida.

Se isso soa familiar, é porque é exatamente o que o TCP/IP fez pelas redes: transformou um emaranhado de conexões proprietárias, cada uma incompatível com a outra, numa língua única que todos falam.



Rápido demais para ser moda

A objeção óbvia a tudo isso seria: “bonito na teoria, mas é só mais um padrão que algumas pessoas usam”. Os números de 2026 fecham essa porta.

Quando foi lançado, em novembro de 2024, o SDK do MCP era baixado cerca de 100 mil vezes por mês. Em março de 2026, esse número tinha chegado a 97 milhões de downloads mensais — um crescimento na casa dos milhares por cento em pouco mais de um ano. Para dar dimensão a isso: o React, uma das bibliotecas de programação mais adotadas da última década, levou cerca de três anos para atingir um volume de downloads comparável. O MCP fez isso em dezesseis meses. Quando uma curva de adoção supera a de tecnologias que definiram a década anterior, você não está mais olhando para uma moda de nicho. Está olhando para infraestrutura em formação.

Mas o número de downloads não é o dado mais importante. O mais importante é quem adotou. Ao longo de 2025, OpenAI, Google, Microsoft, AWS, Salesforce (as maiores empresas de tecnologia do planeta, rivais ferozes entre si em praticamente tudo) convergiram para o mesmo protocolo. Empresas que gastam bilhões tentando prender clientes cada uma no próprio ecossistema concordaram, nesse ponto específico, em falar a mesma língua. Isso não acontece por generosidade. Acontece quando ficar de fora do padrão custa mais caro que qualquer vantagem de mantê-lo fechado — exatamente o que aconteceu com o TCP/IP, exatamente o cálculo que a IBM e a Microsoft erraram nos anos 90.

O selo de que virou infraestrutura

Há um momento na vida de um padrão aberto em que ele deixa de ser "o projeto de uma empresa" e passa a ser patrimônio comum. Para o MCP, esse momento tem data: 9 de dezembro de 2025, quando a Anthropic doou o protocolo para a recém-criada Agentic AI Foundation, sob o guarda-chuva da Linux Foundation — a mesma fundação sem fins lucrativos que abriga o Linux e boa parte da infraestrutura aberta que roda o mundo hoje. OpenAI e Block entraram como cofundadoras. AWS, Google, Microsoft, Cloudflare e Bloomberg como membros de peso.

O gesto importa mais do que parece. Ao doar o MCP para uma fundação de governança neutra, a Anthropic abriu mão de ser a "dona" do protocolo — garantindo, por desenho institucional, que nenhuma empresa isolada possa transformá-lo, no futuro, num novo jardim murado. É o oposto exato do movimento da Dassault com a 3DEXPERIENCE. Uma empresa entregou seu padrão ao bem comum para que ele não pudesse ser cercado; a outra cercou o próprio para que ele não pudesse escapar. As duas decisões foram tomadas no mesmo ano, no mesmo mercado, sobre a mesma tecnologia de base. E dizem tudo sobre as duas visões de futuro em disputa.

Até meados de 2026, o resultado prático é uma escala que já se comporta como infraestrutura: dezenas de milhares de servidores MCP públicos ativos, 28% das empresas da Fortune 500 rodando servidores MCP, e cerca de 78% das equipes de IA corporativa com agentes baseados em MCP em produção. Como resumiu um analista, o MCP virou "a camada de infraestrutura chata" — e isso, longe de ser crítica, é o maior elogio que um protocolo pode receber. Infraestrutura chata é infraestrutura que venceu. Ninguém se emociona com o TCP/IP; todo mundo simplesmente o usa, o dia inteiro, sem pensar. É esse o destino que o MCP está trilhando.

Ninguém no comando, e mesmo assim funciona

Vale parar sobre algo que passou quase despercebido na história que acabei de contar, porque é talvez a lição mais profunda deste capítulo. Ninguém *mandou* o TCP/IP vencer. Não houve um comitê central que decretou o fim das redes proprietárias, nenhuma autoridade que ordenou à IBM e à Microsoft que abandonassem seus jardins murados. O protocolo aberto venceu por um processo que não teve dono: milhões de decisões individuais, cada uma perfeitamente egoísta — este fabricante querendo vender para aquela universidade, aquele programador querendo que seu software

conversasse com o do colega —, convergindo, sem coordenação central, para um padrão comum que beneficiou todo mundo.

Existe um nome para isso, e ele vem de uma das ideias mais elegantes do pensamento econômico do século XX: ordem espontânea.[17] A intuição é a de que as estruturas mais robustas e mais úteis de uma sociedade — a linguagem, o direito consuetudinário, os preços de mercado, e sim, os protocolos abertos — frequentemente não são projetadas por ninguém. Elas *emergem* da interação de indivíduos livres, cada um perseguindo seus próprios fins com seu próprio conhecimento local, e acabam coordenando o comportamento de milhões de pessoas melhor do que qualquer autoridade central conseguiria — precisamente porque nenhuma autoridade central jamais teria acesso a todo o conhecimento disperso que essas milhões de pessoas possuem.

O contraste com os Virtual Companions do Capítulo 1 não poderia ser mais nítido. A 3DEXPERIENCE é ordem *projetada*: uma empresa decide, de cima, qual é o fluxo de trabalho certo do engenheiro, qual assistente ele precisa, o que conta como ajuda. O MCP é ordem *espontânea*: ninguém decide o que você vai conectar a quê; o protocolo só fornece a gramática comum, e o que emerge dela é decidido, a cada uso, por quem está na ponta, com o conhecimento que só quem está na ponta possui. É a diferença entre uma cidade planejada por um único arquiteto que acha que sabe do que todos precisam, e uma cidade que cresce a partir das escolhas de seus moradores. A primeira é mais limpa no papel. A segunda é onde as pessoas de fato querem viver.

Não é coincidência que a mesma tradição de pensamento que descreveu a ordem espontânea também tenha sido a mais cética diante da presunção de qualquer planejador central de que ele sabe, melhor que os indivíduos, o que é bom para eles. Essa presunção tem um nome — e quando uma plataforma de CAD cria um assistente "para o engenheiro" e decide de antemão quais perguntas ele fará, ela está cometendo, no pequeno território do seu fluxo de trabalho, exatamente o mesmo erro epistêmico que o planejador central comete no território de uma economia inteira: agir como se o conhecimento

[17] O conceito de "ordem espontânea" (*spontaneous order*) e o argumento correlato do "conhecimento disperso" foram desenvolvidos por Friedrich Hayek, notadamente no ensaio "The Use of Knowledge in Society" (1945). Hayek argumentava que o conhecimento necessário para coordenar uma sociedade complexa nunca está disponível a uma única mente ou autoridade central, mas espalhado em fragmentos incompletos entre milhões de indivíduos — razão pela qual as decisões são mais bem tomadas por quem detém o conhecimento local. A internet aberta e os protocolos como o MCP são, nesse sentido, exemplos quase perfeitos de ordem espontânea aplicada à tecnologia.

relevante pudesse ser centralizado, quando ele é, por natureza, disperso e local.

O engenheiro na história

Falta aterrissar tudo isso onde este livro mora: no chão de fábrica, na mesa do projetista. O que um protocolo aberto de IA muda para quem desenha molde, programa automação, toca uma pequena ferramentaria?

Muda a posição de quem decide. No mundo dos Virtual Companions do Capítulo 1, a inteligência que ajuda você a trabalhar é escolhida, moldada e limitada pela plataforma dona do seu CAD. Num mundo de protocolo aberto, a mesma pergunta se inverte: você escolhe qual inteligência quer, e a conecta à ferramenta que já usa, nos seus termos. Um projetista pode, em princípio, conectar um assistente de IA à sua própria base de dados de projetos, ao seu histórico de moldes, às suas planilhas de lista de materiais — sem que essa inteligência precise morar dentro do ecossistema de nenhum fabricante de CAD, e sem entregar seus dados de projeto ao jardim murado de ninguém. A inteligência passa a ser um componente que você pluga e despluga, não um inquilino que se instala na sua casa e mistura suas coisas com as dele.

Não estou dizendo que isso já é trivial de fazer hoje, nem que não tem armadilhas — o próximo capítulo e o Capítulo 8 vão tratar, com honestidade, tanto do que já é possível quanto do que ainda dói. Um trabalho acadêmico apresentado em 2026 chegou a argumentar que agentes de IA capazes de traduzir entre formatos e operar interfaces feitas para humanos tornam a interoperabilidade "dramaticamente mais barata e praticamente inevitável" — uma tese ambiciosa, que os próprios autores admitem trazer riscos novos de segurança e de disputa legal junto com a liberdade que promete. Guardo essa ressalva para os capítulos certos. O ponto deste é mais simples, e mais firme: pela primeira vez desde que o CAD virou refém de formatos e plataformas proprietárias, existe uma estrada pública sendo pavimentada por cima dos muros — e ela está sendo pavimentada rápido, com o aval das maiores empresas do mundo, sob governança que nenhuma delas controla sozinha.

O que vem a seguir

O MCP é a estrada. Mas uma estrada não serve de nada se você não tem um veículo próprio para andar nela — e, no mundo do CAD, ter veículo próprio significa não depender de um software proprietário nem para abrir o

seu próprio desenho. É por isso que o próximo capítulo trata do outro lado do rompimento: a chegada, em 2026, de um modelador paramétrico de código aberto maduro o bastante para ser levado a sério, e o que a bifurcação entre soberania aberta e automação proprietária significa, na prática, para quem projeta todo dia.

Fontes: Anthropic, anúncio de lançamento do Model Context Protocol (nov/2024) e doação à Agentic AI Foundation / Linux Foundation (9/dez/2025); Knak, DigitalApplied, Synvestable, SSNTPL e andrew.ooo, estatísticas de adoção do MCP em 2026 (97M+ downloads mensais, comparação com React, adoção Fortune 500); ChatForest, "The MCP Ecosystem in 2026"; University of Michigan Heritage Project e ECE Michigan, "How the Net Was Won" (história da NSFNET e da vitória do TCP/IP sobre redes proprietárias); CircleID, sobre a aposta inicial da Microsoft na MSN como jardim murado no Windows 95; Grokipedia, verbete sobre provedores de serviço online e o fim dos jardins murados dos anos 1990; Marro e Torr, "LLM Agents Are the Antidote to Walled Gardens" (ICML 2026).

Para levar deste capítulo

Tranca 2 — Ferramentas. Protocolos abertos são o que permite trocar de fornecedor sem perder o que você construiu; sem eles, cada ferramenta é uma ilha.

Faça agora — 10 minutos. Liste três ferramentas do seu dia a dia que deveriam conversar entre si e não conversam. Guarde essa lista: ela é a sua primeira lista de alvos de automação.

Capítulo 5 — O Bifurcamento do CAD

No capítulo anterior, falei da estrada pública sendo pavimentada por cima dos muros. Mas uma estrada não basta. Se você quer trafegar por ela com autonomia real, precisa de um veículo que seja seu — e, no mundo de quem projeta, isso significa uma coisa muito concreta e por muito tempo impossível: um software capaz de abrir, editar e modelar o seu próprio desenho sem depender da licença anual de nenhuma corporação.

Durante décadas, esse veículo não existiu de verdade. Havia alternativas gratuitas, sim, mas todas com um asterisco: boas para hobby, frágeis demais para trabalho sério. Em 2026, esse asterisco caiu — e a queda dele é o assunto deste capítulo. Porque o mercado de CAD se partiu em duas direções ao mesmo tempo, e a distância entre elas nunca foi tão nítida.

Duas apostas, o mesmo ano

A imprensa técnica do setor já descreve 2026 com uma clareza quase didática: duas forças remodelaram o CAD simultaneamente, puxando em sentidos opostos.

De um lado, os fornecedores proprietários viraram com tudo para o que estão chamando de “Inteligência Generativa” — e o Capítulo 1 já mostrou como isso funciona na Dassault. SOLIDWORKS e Fusion 360 passaram a usar IA não só para design generativo, mas para simulação em tempo real e criação automatizada de desenhos técnicos, cada recurso mais amarrado à plataforma na nuvem. É o aprofundamento do muro, com inteligência artificial fazendo as vezes de argamassa.

Do outro lado, algo que levou mais de vinte anos para acontecer: um modelador paramétrico de código aberto atingiu, finalmente, qualidade de produção. O FreeCAD, projeto comunitário iniciado no começo dos anos 2000, lançou sua versão 1.0 no fim de 2024 e a refinou ao longo de 2025 e 2026. E não é exagero de entusiasta — é o veredito de análises independentes de mercado, que passaram a tratar o FreeCAD como uma

alternativa crível ao SolidWorks e ao Inventor para uma ampla gama de tarefas de engenharia. Depois de duas décadas sendo “promissor, mas frágil”, o software cruzou a linha para “pronto para produção”.

Repare na simetria: no mesmo intervalo de tempo, uma metade do mercado apostou em fechar mais o jardim, e a outra provou que dá para sair dele. Não é uma disputa teórica sobre qual caminho seria melhor. São dois caminhos reais, sendo trilhados agora, por atores reais, cada um apostando o próprio futuro na direção oposta à do outro.

A pedra que ficou vinte anos no sapato

Para entender por que o FreeCAD 1.0 é um divisor de águas, é preciso entender o que exatamente prendia o software do lado errado da linha por tanto tempo. O nome do vilão é técnico, mas o problema é fácil de sentir na pele de quem projeta: o Problema de Nomenclatura Topológica — em inglês, Topological Naming Problem, ou TNP.

Funciona assim. Num modelo paramétrico, você constrói a peça em etapas: um rascunho, uma extrusão, um furo, um arredondamento, cada operação referenciando as faces e arestas criadas pelas anteriores. O TNP era o bug pelo qual, ao editar uma operação lá do começo da árvore, as operações seguintes quebravam — passavam a referenciar a aresta errada, a face errada, porque a geometria interna tinha sido recalculada e os nomes internos das entidades tinham se embaralhado. Era imprevisível, era enlouquecedor, e era, segundo a própria comunidade, a razão número um pela qual profissionais que experimentavam o FreeCAD acabavam voltando para as ferramentas comerciais. Um molde tem centenas dessas operações encadeadas. Um software em que elas quebram sozinhas não é uma ferramenta de trabalho — é uma armadilha.

A honestidade obriga um parêntese aqui, e é um parêntese que a própria documentação do FreeCAD faz questão de manter: o TNP não é exclusividade do software livre. Ele é um problema geral de CAD paramétrico, presente também no SolidWorks, no Siemens NX e em outros — a diferença é que os pacotes comerciais desenvolveram, ao longo de décadas e com times pagos, heurísticas que reduzem o impacto do problema para o usuário. O FreeCAD simplesmente não tinha tido, até então, a força de desenvolvimento concentrada para fazer o mesmo. Não era um defeito de projeto do código aberto. Era uma questão de quem tinha dinheiro para atacar um problema grande, chato e sem glória.

Quem consertou — e por que isso importa mais que o conserto

É aqui que a história fica interessante para a tese deste livro, porque o *como* o TNP foi resolvido diz mais sobre o futuro da liberdade profissional do que o *que* foi resolvido.

A base do algoritmo que corrigiu o problema nasceu do trabalho de um único contribuidor da comunidade, que a desenvolveu por conta própria num ramo paralelo do projeto — o tipo de esforço voluntário, movido a teimosia, que caracteriza o software livre desde sempre. Mas transformar aquilo num recurso estável, testado e integrado à versão oficial exigiu algo mais: trabalho profissional, pago, sustentado. Esse trabalho veio de uma empresa chamada Ondsel, fundada por um contribuidor de longa data do FreeCAD justamente para explorar um modelo de negócio em cima do software aberto. Foi a combinação dos dois mundos (o voluntário que inventou a solução e a empresa que a industrializou) que finalmente destravou o que vinte anos de esforço puramente voluntário não tinham conseguido destravar sozinhos.

Guarde esse modelo, porque ele vai voltar em vários capítulos deste livro: o software livre mais sério do mundo não é feito só de voluntários heroicos nas horas vagas. Ele funciona melhor quando dinheiro comercial e trabalho comunitário se encontram no mesmo código — exatamente como o Linux, que roda metade da internet do planeta sustentado por uma mistura de voluntários e engenheiros pagos por grandes empresas. Liberdade e sustentabilidade financeira não são inimigas. Elas precisam uma da outra.

O lado incômodo dessa história

Este livro prometeu, desde o prólogo, não vender liberdade como se ela fosse de graça. Então preciso contar o final da história da Ondsel, porque ele é desconfortável — e é justamente por ser desconfortável que ele importa.

A Ondsel fechou. Depois de quase dois anos operando, a empresa não conseguiu encontrar um modelo de negócio sustentável em cima do FreeCAD e encerrou as atividades no fim de 2024. Entrevistaram quase uma centena de engenheiros mecânicos, oficineiros, inventores e makers; encontraram entusiasmo entre usuários independentes e hobbistas, mas não a adoção comercial que justificaria uma empresa financiada por capital de risco. O CAD de código fechado, escreveram na despedida, é ensinado nas escolas e está profundamente entranhado no uso consagrado da indústria.

Competir com isso, de graça, provou-se mais difícil do que a boa vontade da comunidade conseguia sustentar.

Mas — e este “mas” é o coração do capítulo — antes de fechar, a Ondsel devolveu praticamente tudo. Cerca de 150 melhorias, correções e recursos desenvolvidos pela empresa foram enviados de volta ao FreeCAD e incorporados à versão 1.0. A empresa morreu; o código que ela pagou para escrever sobreviveu, livre, na ferramenta que qualquer um pode baixar hoje sem pagar nada. Os desenvolvedores que ela empregava seguiram contribuindo, agora direto para o projeto comunitário, alguns financiados por bolsas da associação que mantém o FreeCAD.

Pare um segundo sobre o peso disso. Se a Ondsel fosse uma empresa de software proprietário e quebrasse, seu produto morreria com ela — como o SpaceClaim que vimos no Capítulo 3, absorvido e descontinuado, deixando seus usuários órfãos. Mas porque a Ondsel construiu sobre código aberto, a falência da empresa não levou junto o trabalho dela. O valor ficou. Migrou para o bem comum e continuou disponível para todo mundo. Essa é a diferença mais profunda entre os dois modelos, e ela só aparece no pior momento possível: quando alguém quebra. No mundo fechado, a morte da empresa é a morte da ferramenta. No mundo aberto, a empresa pode morrer e a ferramenta continua viva. A fragilidade do negócio não contamina a liberdade do usuário.

Uma ressalva técnica honesta

Não quero que este capítulo soe como um panfleto. Então, três precisões, porque elas importam para quem vai de fato apostar trabalho nisso.

Primeira: o TNP foi mitigado, não literalmente eliminado. A própria comunidade é cuidadosa com a linguagem — o novo algoritmo reduz drasticamente o problema e, em muitos casos, o resolve ou o repara automaticamente, mas ainda há partes do software que não usam o novo tratamento, e modelos ainda podem quebrar por outras razões que nada têm a ver com nomenclatura topológica. Nenhum CAD do mundo, aberto ou fechado, está imune a modelos mal construídos. Não existe software que substitua bons princípios de projeto.

Segunda: entranhamento importa. O CAD fechado é ensinado nas escolas, exigido por clientes, consolidado em décadas de arquivos herdados. Um projetista de moldes que hoje depende de recursos específicos do Solid Edge, do SolidWorks ou do NX não troca de ferramenta num fim de semana

— e este livro não está pedindo que ele faça isso. O que mudou em 2026 não é que o FreeCAD substituiu os comerciais da noite para o dia. É que, pela primeira vez, existe uma saída viável para quem quiser construí-la, no ritmo que puder.

Terceira, e esta é minha, não da pesquisa: eu mesmo trabalho com Solid Edge todos os dias, e não vou fingir que abandonei ferramentas comerciais para escrever este livro. A liberdade que defendo não é a pureza de quem só usa software livre. É a de conhecer a saída, manter a porta destravada e ir movendo o que der para mover, no tempo certo. Saber que o FreeCAD 1.0 existe e amadurece muda a minha relação de poder com qualquer fornecedor de CAD — mesmo nos dias em que continuo usando o produto pago. A alternativa crível é, por si só, uma forma de liberdade, ainda que você nunca chegue a usá-la.

O trator que você comprou mas não possui

Vale sair um instante do CAD para olhar um caso vizinho, porque ele torna visceral um argumento que, no software de projeto, ainda soa abstrato. Pergunte a um fazendeiro americano quem é o dono do trator dele.

Por anos, a John Deere (uma das maiores fabricantes de equipamento agrícola do mundo) travou o software de diagnóstico e reparo de seus tratores, liberando as ferramentas capazes de fazer todos os reparos eletrônicos apenas para suas concessionárias autorizadas. O fazendeiro comprava a máquina, era dono do aço, do motor, dos pneus — mas não do software que fazia tudo aquilo funcionar. Quando um código de erro aparecia no painel, interpretá-lo exigia um programa que a Deere se recusava a disponibilizar. Na prática, o dono do trator não podia consertar o próprio trator. Precisava esperar, às vezes semanas, e pagar o preço que a concessionária cobrasse. Um fazendeiro relatou ter ficado 28 dias com um distribuidor de fertilizante parado na oficina.

Repare como a estrutura é idêntica à da gaiola de CAD que este livro descreve. Você “possui” a ferramenta, mas o conhecimento de como operá-la em profundidade, de como consertá-la, de como modificá-la, permanece trancado do lado de dentro de uma empresa. A propriedade vira uma ficção: você tem o objeto, mas não tem o poder sobre o objeto. É o “lock-in de wetware” do Capítulo 3 transformado em aço e diesel.

Em 8 de julho de 2026, depois de anos de processos movidos por fazendeiros e uma ação antitruste da FTC e de cinco estados, a John Deere assinou um

acordo comprometendo-se a dar a fazendeiros e oficinas independentes o mesmo acesso a ferramentas de reparo e software que antes reservava só às concessionárias — por dez anos, sob supervisão.

Preciso ser honesto sobre um incômodo aqui, e a honestidade é mais valiosa que a conveniência retórica. Esse desfecho veio, em boa parte, de ação estatal — regulação antitruste, tribunais federais. Um libertário coerente não comemora sem ressalva a intervenção do Estado no contrato entre empresa e cliente; há uma tensão real entre celebrar o resultado e desconfiar do método. Mas a lição que me interessa não é sobre o remédio — é sobre a doença. O caso da John Deere prova que o cercamento digital da propriedade não é uma paranoia de ativista de software: é um modelo de negócio real, aplicado a bens físicos que as pessoas pagaram caro para possuir, e levado ao ponto em que só a força de um processo judicial conseguiu afrouxá-lo. A pergunta que fica é profundamente libertária, independentemente de como o caso Deere terminou: por que a solução teve que vir de um tribunal, em vez de estar disponível desde o começo, na forma de ferramentas que respeitassem a propriedade de quem comprou? A resposta aberta (projetar as coisas para serem reparáveis e modificáveis por quem as possui) não precisa de Estado nenhum. Ela só precisa de fabricantes que não tratem a dependência do cliente como ativo.

A liberdade que nasce da escolha, não da imposição

Isso me leva a um ponto que preciso deixar explícito, porque é o eixo filosófico de toda a Parte II e me protege de um mal-entendido fácil. Defender ferramentas abertas não é o mesmo que ser contra propriedade privada, contra lucro ou contra software proprietário. Pelo contrário — e a própria história do movimento de código aberto prova isso.

Um dos pensadores mais influentes dessa história, Eric Raymond, autor do ensaio seminal “The Cathedral and the Bazaar”, é um libertário assumido. E o mais interessante: ele defende o software aberto não como um imperativo moral que proíbe o código fechado, mas como algo pragmaticamente superior, sustentado em cooperação voluntária e eficácia. Raymond chega a defender explicitamente o direito do programador de escolher licenças proprietárias se assim quiser. Sua aposta no aberto se baseia em vantagens tangíveis — interoperabilidade, resiliência, menos dependência de fornecedor, código que qualquer um pode inspecionar — e não numa condenação do fechado. Ele critica, inclusive, a retórica moralista que afasta empresas em vez de convencê-las.

Essa distinção é tudo para o livro que estamos escrevendo. A liberdade que defendo aqui não é a liberdade imposta de cima, por decreto, que obrigaria todo mundo a abrir seu código. É a liberdade de baixo, a do indivíduo que escolhe: escolhe qual ferramenta usar, escolhe manter a porta destravada, escolhe não entregar a chave da própria capacidade de trabalho em troca de conveniência. O software aberto, nessa leitura, é a expressão mais pura de cooperação voluntária que a tecnologia já produziu — milhares de pessoas construindo algo de valor econômico real, que fica livremente disponível, sem ninguém sendo coagido a participar nem a usar. É mercado, é propriedade (a de decidir redistribuir o próprio trabalho), é contrato voluntário. É, no sentido mais literal, liberdade em ação — e não o oposto dela.

O FreeCAD não venceu o Problema de Nomenclatura Topológica porque um governo mandou. Venceu porque um voluntário quis resolver, uma empresa apostou em pagar pelo trabalho, e a comunidade acolheu o resultado — todos por vontade própria, cada um perseguindo o próprio interesse, e o bem comum emergindo dessa cooperação sem que ninguém precisasse comandá-la. Se existe uma prova de que liberdade individual e cooperação não são inimigas, ela tem nome de arquivo e roda na sua máquina de graça.

O elo com o resto do livro

Há um detalhe técnico que amarra este capítulo a uma discussão que aparece mais à frente. O FreeCAD é construído sobre um kernel geométrico de código aberto, o OpenCascade (OCCT) — o equivalente livre ao Parasolid da Siemens que vimos no Capítulo 2, ou ao Granite da PTC. Ou seja: a liberdade do FreeCAD não é só de interface, é estrutural, desce até o motor matemático. Quem quer construir a própria ferramenta de CAD — e este é um sonho recorrente de todo projetista-programador, eu incluído — encontra no OCCT uma base que não pertence a nenhum concorrente. Voltarei a essa camada mais fundo do que o software, a do kernel, quando falarmos de construir a própria stack, na Parte IV.

Por ora, o que fica é a imagem da bifurcação. Duas estradas partindo do mesmo ponto, em 2026, em direções opostas. Uma leva a jardins cada vez mais murados, com inteligência artificial tornando o muro mais alto e mais confortável ao mesmo tempo. A outra leva a ferramentas que você pode possuir, modificar e manter vivas mesmo quando a empresa que as impulsionou morre. Nenhuma das duas é gratuita em esforço. Mas só uma delas devolve a você a chave da porta.

No próximo capítulo, deixo de falar das ferramentas e passo a falar de quem as usa — mais especificamente, do tipo de profissional que está nascendo no cruzamento entre quem projeta e quem programa. É o capítulo mais pessoal do livro, e é onde a minha própria história, prometida lá no prólogo, finalmente vai ser contada por inteiro.

Fontes: TheCADHub e EngineersViews, panoramas de CAD 2026 e alternativas gratuitas (bifurcação entre FreeCAD 1.0 e "Inteligência Generativa" proprietária); documentação oficial do FreeCAD e FreeCAD-documentation (GitHub), sobre o algoritmo de nomenclatura topológica na versão 1.0; Ondsel, blog "Toponaming problem is history" (mai/2024) e post de encerramento "We are shutting down Ondsel" (out/2024); Hackaday, "The End of Ondsel and Reflecting on the Commercial Prospects for FreeCAD" (nov/2024); FreeCAD News, "The Ondsel Onwards Fund" (fev/2025); WebProNews, "FreeCAD 1.1 Arrives" (mar/2026); FTC, comunicado do acordo de direito de reparo com a Deere & Company (8/jul/2026), e Colorado Sun, US News e Fortune, cobertura do caso John Deere e do movimento "right to repair" (2026); Wikipedia, Econlib/EconTalk e Grokipedia, sobre Eric S. Raymond, "The Cathedral and the Bazaar" e sua defesa libertária do software aberto por cooperação voluntária.

Para levar deste capítulo

Tranca 2 — Ferramentas. Ter uma alternativa que você sabe usar é o que transforma a sua dependência de escolha em prisão — ou o contrário.

Faça agora — 20 minutos. Instale o FreeCAD. Só instale, não precisa aprender nada hoje. O objetivo desta tarefa é apenas que a alternativa deixe de ser uma abstração e passe a existir na sua máquina.

Capítulo 6 — O Programador-Projetista: A Nova Espécie Híbrida

Antes de eu ter idade para trabalhar, eu já sabia montar um computador por dentro. Aprendi numa escola de informática de bairro, a BitCompany, onde fiz, ainda menino, cursos de montagem e manutenção de microcomputadores e de montagem e manutenção de redes. Não eram cursos de "usar o computador". Eram cursos de abrir a máquina, entender o que havia lá dentro e fazer aquilo funcionar. No mesmo lugar aprendi CorelDraw e Photoshop, o desenho e a imagem. Olhando de trás para frente, é quase cômico o quanto a criança que fez esses cursos já era, sem saber, o adulto que escreve este livro: alguém dividido entre a vontade de desenhar e a vontade de mexer na máquina por dentro — e que levaria quinze anos para descobrir que essas duas coisas eram, na verdade, uma só.

Este é o capítulo mais pessoal do livro. Até aqui, falei de plataformas, formatos, kernels, protocolos, empresas que nasceram e morreram. Falei das ferramentas. Agora preciso falar de quem as usa, porque a tese central deste livro não se sustenta só na tecnologia — ela se sustenta na existência de um tipo de profissional que está nascendo no meio do caminho entre duas profissões que, até anteontem, eram consideradas separadas: quem desenha e quem programa. O projetista e o programador. Eu sou um exemplar dessa espécie híbrida, e não um exemplar particularmente talentoso — sou, na melhor das hipóteses, um exemplar *teimoso*. E é justamente por não ser um gênio da computação que a minha história prova o que ela prova: se eu cheguei aqui, a barreira caiu para praticamente todo mundo que quiser atravessá-la.

O lugar que desapareceu

Eu queria trabalhar cedo. Cedo demais, talvez. Meu primeiro contato com o trabalho de verdade foi como voluntário na própria BitCompany, a escola onde eu tinha estudado — e foi o suficiente, apesar do pouco tempo, para eu

sentir na pele que o mercado de trabalho era um lugar assustador. Mas eu era muito novo, e não havia como acelerar aquilo.

Foi aí que entrou meu avô, Roberto Carlos Gonçalves, dono de uma pequena oficina de usinagem de manutenção, a Rogertex. Ele me deu um conselho que era também uma promessa: faça o SENAI, aprenda um ofício de verdade, e quando você tiver seus catorze anos, vem trabalhar comigo. Era um plano perfeito para um menino ansioso para começar a vida — havia um caminho, e havia um lugar garantido esperando no fim dele. Eu fiz a minha parte. Entrei no SENAI.

E então, mais ou menos quando eu entrava, meu avô desfez a Rogertex. A empresa se separou, e o lugar que estava reservado para mim simplesmente deixou de existir. Eu tinha feito tudo certo — seguido o conselho, começado o curso, cumprido a minha parte do combinado — e mesmo assim me vi sem chão, com uma qualificação começando e nenhum lugar garantido para exercê-la.

Não conto isso por mágoa; não há mágoa nenhuma. Conto porque foi a primeira vez que a vida me ensinou, de forma concreta e pessoal, a lição que este livro inteiro tenta articular de forma mais ampla: **o lugar que você herda pode desaparecer, e o único lugar que ninguém consegue tirar de você é aquele que você aprende a construir.** Eu não formulei isso em palavras aos catorze anos, claro. Mas alguma coisa se fixou ali. A partir daquele momento, a minha vida inteira foi, no fundo, uma longa lição de como construir o próprio lugar quando nenhum lhe é dado — no desenho, no código, na infraestrutura, na sociedade de uma empresa, no negócio que toco hoje. Há até uma simetria fina nos nomes: eu me chamo Carlos Eduardo Gonçalves Keese, e não carrego dele apenas o Gonçalves — carrego o próprio Carlos, o primeiro nome que é dele antes de ser meu. O avô que me apontou um caminho que se desfez me deixou, junto, dois pedaços do próprio nome, com os quais eu construiria sozinho tudo o que veio depois.

A busca pelo próprio lugar

O SENAI de Santa Bárbara d'Oeste me formou mecânico de usinagem entre 2008 e 2009, e foi lá que fiz meu primeiro curso de CAD de verdade, o SolidWorks. Enquanto ainda cursava, no segundo ano, arrumei meu primeiro emprego numa fábrica: montador mecânico, montando máquinas de *vacuum form*. Foi um período duro, e não pela dificuldade técnica. O supervisor implicava comigo, fazia o tipo de humilhação pequena e diária que hoje tem nome — e que eu, na época, apenas suportei calado, do primeiro ao último

dia. Aguentei porque precisava, mas eu sabia, com uma certeza que vinha do estômago, que ali não era o meu lugar. Que aquilo não seria a minha profissão. A sensação de estar no lugar errado é, para quem já a viveu, quase física — e ela me acompanharia por mais alguns empregos até eu entender que o problema não era achar o lugar certo pronto em algum lugar, e sim aprender a construí-lo.

O passo seguinte me aproximou do que eu de fato viria a ser, e foi também onde muitas peças de mim começaram a fazer sentido. Entre 2011 e 2013, trabalhei como desenhista na Flexicorte, desenhando ferramentas de corte rotativas em metal duro no AutoCAD. Foi ali que aprendi a desenhar praticamente todo tipo de ferramenta, que comecei a parametrizar geometrias no SolidWorks, que descobri, por meio de um colega, que existia uma faculdade pública chamada Fatec. E foi ali, também, que a semente do resto da minha vida foi plantada, pela via mais banal possível: a repetição. O trabalho estava cheio de tarefas que a máquina deveria fazer sozinha e não fazia, porque ninguém tinha programado ela para isso. Eu repetia a mesma sequência de comandos, dia após dia, e uma voz no fundo da cabeça não parava de dizer que aquilo era burrice — não minha, mas da forma como eu estava usando a máquina. Devia existir um jeito de ensinar o programa a fazer sozinho o que eu fazia na mão. Descobri que existia, e que tinha um nome: AutoLISP, a linguagem de automação do AutoCAD. Não aprendi de fato naquela época, mas a descoberta acendeu uma curiosidade que não se apagou mais.

Mas foi na Flexicorte, também, que eu reparei pela primeira vez que havia alguma coisa diferente no meu jeito de funcionar. Eu não conseguia me concentrar nos desenhos. Passava horas do dia me perdendo em pesquisas sobre as ferramentas, sobre por que elas eram como eram, sobre o que poderia ser diferente — tudo, menos a tarefa que estava na minha frente. Cometi vários erros naquele período, alguns deles bobos. Na época, aquilo me parecia só mais uma prova do que os colegas de escola já tinham me dito a vida inteira, quando eu não conseguia parar sentado na cadeira, quando meus cadernos ficavam pela metade, quando eu simplesmente desligava nas aulas que não me interessavam: que eu era, no fundo, meio burro. Levaria anos até eu entender que aquilo tinha outro nome, e que não era defeito nenhum. Volto a isso mais adiante, porque é importante demais para tratar de passagem.

E houve um erro que não foi bobo, e que me custou o emprego — embora, até hoje, eu não o considere um erro meu. O filho do dono, que respondia pela empresa, me pediu para alterar o desenho de um item que era recorrente

na produção. Eu sabia o risco que aquilo trazia, então fiz questão de registrar, no sistema, o nome dele como responsável pela alteração. Foi a coisa certa a fazer. Meses depois, chegou um lote daquele item para reafiar, e o PCP puxou automaticamente o desenho mais recente — o alterado. O que deveria ter sido uma reafiação simples de topo virou uma redução de diâmetro. Deu trabalho errado, prejuízo. E, porque eu havia atualizado a descrição no sistema conforme a revisão pedida, fui eu — não quem pediu a alteração — que levei a culpa e a demissão.

Guardo essa história não por ressentimento, mas porque ela me ensinou cedo uma coisa que este livro leva a sério: fazer a coisa certa, documentar, registrar quem decidiu o quê, nem sempre protege o pequeno na hora em que a conta chega. Quem tem o poder de assinar a demissão frequentemente também tem o poder de escolher de quem foi a culpa. Não é por acaso que, anos depois, eu me tornaria quase obsessivo com procedimentos, revisões e rastreabilidade nos meus projetos de molde — e que a desconfiança em relação a quem detém o poder, sem responsabilidade proporcional, viraria parte do meu jeito de ver o mundo. Mas isso é assunto para outros capítulos. Naquele momento, eu era só um jovem desenhista injustamente demitido, que tinha, ao menos, descoberto três coisas que mudariam tudo: o LISP, a Fatec, e a suspeita de que a repetição podia ser vencida por código.

A faculdade que a vida permitia

Aqui preciso ser honesto sobre uma coisa que estrutura silenciosamente toda a minha trajetória, e que boa parte dos livros sobre “reinvenção profissional” prefere não mencionar: dinheiro.

Eu nunca tive a oportunidade de cursar Engenharia Mecânica. Nunca foi, sequer, um desejo pessoal — mas o ponto é que a opção não estava sobre a mesa de qualquer forma. Pagar caro por um diploma numa área em que eu não enxergava futuro não cabia no meu orçamento nem nos meus planos. Meu pai, dentista, nunca lidou bem com os próprios gastos; com quatro filhos e uma dívida impagável da construção de uma casa, mesmo que quisesse não teria como bancar uma faculdade particular para mim. Então eu fiz o que quem não tem dinheiro faz quando ainda assim quer aprender: procurei o que era gratuito. E a Fatec — pública, de qualidade, sem mensalidade — foi uma bela surpresa.

Só que havia um porém prático, desses que decidem uma vida inteira sem pedir licença. No período noturno, o único que eu podia cursar porque precisava trabalhar de dia, não havia o curso de Análise de Sistemas, que era

o que eu de fato queria. Havia Jogos Digitais. E eu, que precisava aprender a programar e não tinha outra porta gratuita à vista, entrei em Jogos Digitais — porque era ali, naquele curso, que se aprendia a programar de verdade, com um pretexto que por acaso até me agradava. A programação era o destino; os jogos, o veículo disponível.

Faço questão de marcar esse ponto, porque ele é a minha biografia encontrando a tese deste livro de frente. Toda a minha competência técnica foi construída assim: **de baixo, pelos caminhos gratuitos e públicos que a vida permitia — SENAI, Fatec —, somados ao autodidatismo e, mais tarde, à inteligência artificial. Ela nunca foi comprada num diploma caro.** Eu não escrevo um livro sobre construir a própria autonomia a partir de baixo como quem teoriza sobre algo distante. Escrevo como quem não teve outra opção a não ser fazer exatamente isso.

Na Fatec, aliás, a criança da BitCompany voltou a ser útil: os cursos de infância — hardware, Photoshop, CorelDraw — de repente eram base para o que se pedia num curso de jogos. Nada do que eu tinha aprendido por curiosidade, anos antes, foi desperdiçado.

E, no entanto, eu não me formei. No começo foi tudo bem, com as matérias normais de um tecnólogo de tecnologia. Mas fui acumulando dependências, e junto com elas veio uma pergunta que eu não conseguia mais calar: o que exatamente um curso de jogos digitais ia me proporcionar, profissionalmente, no Brasil? Eu não enxergava um futuro ali, e — por decisão minha, que assumo como um erro meu e não como culpa de ninguém — desisti no terceiro ano. Escrevo isso sem constrangimento, porque faz parte honesta da história, e porque desconfio que muita gente que vai ler este livro tem uma coleção parecida de recomeços. A narrativa do sujeito competente que entra na faculdade certa aos dezoito e sai formado aos vinte e dois, em linha reta, é uma ficção conveniente que não descreve quase ninguém que eu conheço de verdade no chão de fábrica e nas salas de projeto.

Onde a política acordou

Foi também na Fatec que uma outra parte de mim começou a se formar, e preciso registrá-la aqui — ainda que a discussão de fundo fique para a terceira parte do livro, onde ela cabe melhor.

Eu chegei àquele ambiente sem vocabulário político nenhum, mas com os olhos abertos. E o que eu via me incomodava: muito movimento à esquerda, muita crítica ao mercado, e quase ninguém disposto a apontar o dedo para

os gastos públicos ou para a lógica que sufocava as empresas pequenas — as mesmas empresas que, lá na ponta, davam de ombros e diziam que não podiam me pagar melhor. A situação do meu pai, afogado em dívidas, e a desculpa recorrente dos patrões foram, aos poucos, alimentando em mim uma desconfiança organizada em relação ao poder e ao desperdício do dinheiro alheio. Eu me via, naquela época, como um liberal — no sentido clássico, econômico da palavra.

A virada veio depois, quando encontrei canais e comunidades que davam nome e estrutura ao que eu sentia de forma difusa — o Ideias Radicais, o Ancapsu, o pensamento libertário brasileiro que fervilhava na internet. Ali eu não só encontrei um vocabulário; encontrei uma ressonância inesperada com a minha fé cristã, com a ideia de que a liberdade individual e a responsabilidade moral de cada um diante de suas próprias escolhas eram, no fundo, a mesma coisa vista de dois ângulos. Foi quando deixei de me chamar apenas de liberal e passei a me entender como libertário. Guardo os detalhes dessa convicção para os capítulos da terceira parte. Por ora, basta dizer que ela não nasceu de um livro de teoria: nasceu de ver, de perto, o preço que pessoas reais pagam quando a liberdade de construir a própria vida é cara demais ou cercada demais.

As duas carreiras que sempre foram uma

Aqui a minha história dá o nó que dá sentido ao título deste capítulo — e ela o faz muito antes de a inteligência artificial entrar em cena, o que é importante deixar claro.

Depois da Flexicorte, minha vida profissional seguiu por uma trilha de projeto mecânico cada vez mais sofisticada: projetista na Indústria Mecânica São Carlos, desenhando ferramentas de corte, brocas e fresas, com simulação em máquina afiadora CNC; depois na Precision Usinagem de Peças, e depois cinco anos na Kenatec, onde não fui só funcionário — fui sócio. Nessa trilha eu me tornei projetista de produtos plásticos, de dispositivos, de ferramentas e, sim, de moldes de injeção, aprendendo os softwares que hoje domino: Solid Edge, SolidWorks, TopSolid, NX.

Mas correndo em paralelo a essa, havia uma segunda carreira — e ela não era um hobby de fim de semana. Era trabalho remunerado, formal, levado a sério. Na Precision, eu não era apenas projetista: eu era o responsável pela tecnologia da informação da empresa. Eu configurava a comunicação serial entre os computadores e os centros de usinagem de comando Fanuc e Fagor. Eu passava cabeamento de rede, administrava contas, e-mails e domínios,

implantava sistemas de gestão como ERP e MRP, cuidava de servidores Linux e Windows, gerenciava máquinas virtuais em ESXi, montava ramais de telefonia VoIP em Asterisk, instalava sistemas de câmeras e alarmes. Na Kenatec, como sócio, prestei consultoria de T.I. para outras empresas — montagem e manutenção de máquinas, redes, hospedagem — e comecei, ali, meu contato de verdade com a programação: Arduino, tentativas de corrigir pós-processadores no Fusion 360, um projeto em Kotlin e Java para um cliente, um pouco de JavaScript para colocar o site da própria Kenatec de pé.

Faço questão desse inventário, por mais árido que pareça numa página, porque ele desmonta uma ideia falsa que eu mesmo tinha sobre a minha vida. Durante muito tempo, achei que a fusão entre desenhar e programar fosse uma novidade recente, uma virada da minha carreira madura. Não é. Olhando o registro com honestidade, eu já era as duas coisas ao mesmo tempo, formalmente, há mais de uma década. O projetista e o técnico de T.I. sempre foram a mesma pessoa, trabalhando lado a lado na mesma cabeça, apenas sem que eu tivesse percebido que aquilo tinha um nome ou um significado maior. **A espécie híbrida que este capítulo anuncia não é uma profecia sobre o futuro. É uma descrição do que eu já era sem saber — e do que muita gente já é sem se dar conta.**

A configuração, não o defeito

Prometi voltar àquilo que eu, por muito tempo, achei que fosse burrice — e aqui está, porque é parte do funcionamento da minha cabeça e não uma nota de rodapé. Eu tenho TDAH, e tomo Ritalina. Aquele menino que não conseguia parar sentado na cadeira, que deixava os cadernos pela metade, que desligava por completo nas aulas que não lhe interessavam e era chamado de burro pelos colegas, não era burro coisa nenhuma. Era um cérebro configurado de um jeito diferente, tentando rodar num sistema, a escola tradicional, que tinha sido projetado para outro tipo de máquina. E o mesmo acontecia na mesa de desenho da Flexicorte, onde eu me perdia em pesquisas em vez de fazer a tarefa da frente: não era falta de capacidade, era um processador que só engata de verdade quando o problema o interessa.

Porque é isto que quase ninguém te conta sobre o TDAH: ele explica, ao mesmo tempo, os recomeços tortos e a capacidade de mergulhar por dezoito horas seguidas num problema de automação até resolvê-lo. O mesmo cérebro que não consegue seguir uma grade curricular linear é o que entra em fixação produtiva quando encontra um problema que *importa*

para ele. Durante muito tempo, tratei isso como defeito. Hoje entendo como configuração — uma máquina que roda mal com o software errado e roda muito bem com o software certo. A liberdade de escolher os próprios problemas, que este livro tanto defende, não é para mim um luxo filosófico. É condição de funcionamento. Um sistema montado para o projetista médio, linear, previsível, me sufoca; um ambiente aberto, onde eu escolho no que mergulhar, me faz render o dobro. Guardem essa imagem, porque ela vai voltar quando o livro discutir por que ferramentas fechadas, que decidem por você, custam mais caro para umas pessoas do que para outras.

O elo que faltava: quando o contato virou capacidade

Reparem numa coisa estranha na história que acabei de contar. Eu tive contato com programação a vida inteira — desde os cursos de hardware e redes na infância, passando pela curiosidade com o AutoLISP na Flexicorte, até o Arduino, o Kotlin e o JavaScript da fase Kenatec. Décadas de contato. E, no entanto, eu nunca tinha produzido uma automação *de verdade* — daquelas que entram em uso, que resolvem um problema real e recorrente, que poupam trabalho de fato. O contato existia havia quinze anos. A capacidade, não.

O elo que faltava chegou só recentemente, no meu trabalho atual como projetista de moldes na Panflight, a partir de 2023. E ele chegou junto com uma ferramenta nova: a inteligência artificial. Foi ali, e só ali, que meus experimentos de código deixaram de ser tentativas frustradas e viraram ferramentas que funcionavam. Uma delas — não a primeira, houve outras antes — foi uma automação de requisição de compra, que começou escrita em Python. Não era bonita; um programador profissional provavelmente torceria o nariz para o meu código. Mas ela funcionava, era *minha*, e fazia em segundos o que eu levava minutos para fazer na mão, projeto após projeto. E o mais importante: eu a construí *conversando* com a inteligência artificial — descrevendo o que queria, entendendo o código que ela me devolvia, corrigindo, quebrando, remontando, aprendendo no processo. Tempos depois, quando quis torná-la mais robusta, reescrevi essa automação em C# — do mesmo jeito, conversando com a inteligência artificial. Mudou a língua do código; não mudou o método de construí-lo, nem o fato de ele ser meu.

Essa é a peça que muda tudo, e é por isso que este é um livro sobre 2026 e não sobre qualquer outro ano. O AutoLISP que eu não consegui aprender sozinho aos vinte e poucos anos, a lógica de programação que a faculdade não

me fez engolir em linha reta, os experimentos que nunca chegavam a lugar nenhum — tudo isso destravou no momento em que eu pude *conversar* com a linguagem em vez de decorá-la. A IA não substituiu o meu aprendizado de quinze anos; ela foi a ponte que finalmente converteu todo aquele contato acumulado em capacidade real. A barreira que me travou a vida inteira ruiu — não porque eu fiquei mais inteligente, mas porque a ferramenta mudou.

Faço questão de ser honesto sobre isto, porque é o coração do capítulo: **eu não sou um programador profissional.** Ainda estou aprendendo — hoje formalmente, cursando Engenharia de Computação na Univesp, a universidade virtual do estado de São Paulo, depois de ter passado, no meio do caminho, por um curso de Tecnólogo em Sistemas para Internet. Não escrevo como quem chegou ao topo e olha para trás; escrevo como quem está no meio da subida e resolveu contar como é a subida. Não estou vendendo um destino. Estou descrevendo um caminho que qualquer um pode começar a trilhar amanhã.

O que era anedota virou estatística

Por muito tempo, eu achei que essa minha maneira de trabalhar — projetista que programa com ajuda de IA, sem formação formal completa, aprendendo no problema concreto — fosse uma excentricidade minha. Uma gambiarra pessoal para compensar as lacunas da minha formação. Os números de 2026 me mostraram que eu estava, sem saber, na frente de uma onda que hoje é a maior transformação da história recente do desenvolvimento de software.

Vale colocar os dados na mesa, porque eles são espantosos. No começo de 2026, cerca de 90% dos desenvolvedores relatavam usar regularmente ao menos uma ferramenta de IA no trabalho — acima dos 85% de meados de 2025, numa aceleração que não deu sinais de parar.[18] A proporção de código gerado com auxílio de IA saltou, em média, de 28% em 2025 para mais de 50% em 2026.[19] Grandes empresas de tecnologia já geram, dependendo da organização, de 30% a 90% do código novo com IA.[20] A consultoria Gartner

[18] JetBrains AI Pulse Survey, dados do início de 2026 (publicados em abril de 2026): 90% dos desenvolvedores relataram uso regular de ao menos uma ferramenta de IA no trabalho, contra 85% em meados de 2025.

[19] State of AI 2026 (levantamento setorial): a proporção média de código gerado com IA saltou de 28% em 2025 para 54% em 2026, com o crescimento mais forte nos segmentos que geram 75% ou mais do código com IA.

[20] Múltiplos levantamentos convergem nessa faixa; Google e Microsoft reportaram publicamente que mais de 30% do código novo já é escrito com auxílio de IA, e outras organizações relatam índices de até 90%.

projeta que, até 2028, 90% dos engenheiros de software corporativos usarão assistentes de IA — contra menos de 14% no início de 2024.[21]

Leia esses números de novo com a minha história na cabeça. O que eu fiz, sozinho, numa sala de projeto de ferramentaria no interior de São Paulo, sem saber que estava fazendo nada de especial, é literalmente o que uma indústria inteira passou a fazer ao mesmo tempo. A transição da programação assistida por IA de experimento de pioneiros para prática padrão está oficialmente em curso.[22] Eu não fui visionário. Eu só cheguei um pouco antes, empurrado pela mesma teimosia de sempre — e a tecnologia veio ao meu encontro no meio do caminho.

A fronteira que está desaparecendo

Há uma distinção antiga, quase sagrada, que este movimento está dissolvendo diante dos nossos olhos: a fronteira entre quem *usa* uma ferramenta e quem *constrói* uma ferramenta.

Durante todo o século XX, essas foram duas castas separadas. De um lado, o engenheiro, o projetista, o desenhista — o profissional de domínio, que conhece o problema real, mas é usuário passivo do software que outra pessoa fez. Do outro, o programador, o desenvolvedor — que constrói o software, mas frequentemente não faz a menor ideia do problema de domínio para o qual ele serve. Entre os dois, um muro. O projetista pedia, o programador atendia (ou não). O projetista dependia. Era, de novo, a mesma dependência que percorre este livro inteiro, só que agora na camada mais íntima de todas: a de precisar de outra pessoa para fazer a sua própria ferramenta obedecer.

O programador-projetista (essa espécie híbrida que estou descrevendo, e da qual sou uma amostra) é o que nasce quando esse muro cai. Não é um projetista que virou programador, nem um programador que aprendeu a projetar. É uma terceira coisa: um profissional de domínio que recuperou a capacidade de moldar as próprias ferramentas, sem precisar pedir licença a ninguém. Ele conhece o problema real *porque o vive todos os dias*, e agora consegue também dobrar o software à sua vontade *porque a barreira que o impedia disso ruiu*. Essa combinação, que antes exigia dois profissionais e

[21] Gartner, previsões sobre adoção corporativa de assistentes de IA para engenharia de software (2026): projeção de 90% dos engenheiros de software corporativos até 2028, contra menos de 14% no início de 2024.

[22] State of AI 2026: a formulação sobre a transição de “experimento de pioneiros para prática padrão” resume o consenso dos principais surveys do setor (Stack Overflow, JetBrains, DORA, DX).

uma reunião de alinhamento entre eles, hoje cabe numa cabeça só. Na sua cabeça.

E é aqui que este capítulo, o mais pessoal do livro, reencontra a tese mais política dele. Recuperar a capacidade de construir a própria ferramenta é a forma mais concreta, mais tangível, menos abstrata que existe de recuperar controle sobre o próprio trabalho. Não é uma metáfora. É literal. Quando você consegue escrever a automação que resolve o seu problema, você deixa de estar à mercê da empresa que decide se aquele recurso vale a pena ser incluído na próxima versão, se ele fica atrás de mais uma camada de assinatura, se ele algum dia vai existir. Você para de esperar que alguém lá em cima decida o que é bom para você. Você desce e faz. A liberdade de baixo, de novo — a que se conquista fazendo, não a que se recebe por decreto. A mesma que eu tive de aprender aos catorze anos, quando o lugar que me haviam prometido desapareceu.

O que não é: uma ressalva honesta

Seria desonesto encerrar este capítulo em tom de triunfo sem a ressalva que o resto do livro exige. A mesma pesquisa que celebra a adoção massiva da IA na programação traz um segundo dado, mais sóbrio: a confiança dos desenvolvedores no que a IA produz vem *caindo* mesmo enquanto o uso sobe. Em 2026, apenas cerca de 29% dos desenvolvedores diziam confiar na precisão do código gerado por IA, uma queda expressiva em relação aos anos anteriores.[23] Código gerado por IA pode conter significativamente mais vulnerabilidades do que código escrito por humanos e revisado com cuidado.[24] A tarefa que mais cresce no dia a dia do desenvolvedor não é escrever código — é *revisar* o código que a IA escreveu, checando se aquilo que "parece certo" de fato é certo.[25]

Guardo esses números com carinho, não os escondo, porque eles são a prova de que a liberdade que este livro defende nunca é gratuita. A capacidade de construir a própria ferramenta vem acompanhada, inseparavelmente, da

[23] Stack Overflow Developer Survey 2025: apenas 29% dos desenvolvedores relataram confiar na precisão dos resultados das ferramentas de IA, em queda em relação a anos anteriores (acima de 40% em 2024).

[24] Análises independentes de código em 2026 indicaram que trechos gerados por IA podem conter cerca de 2,7 vezes mais vulnerabilidades que código humano, com parcela significativa de amostras falhando em testes de segurança. O tema é aprofundado no Capítulo 8.

[25] Levantamentos de 2026 (DX, Fungies, Panto AI) convergem no achado de que revisar código gerado por IA superou a escrita de código como o maior consumo de tempo no fluxo de trabalho assistido por IA.

responsabilidade de garantir que a ferramenta faz o que deveria — e não faz o que não deveria. O programador-projetista que aceita o poder da IA sem aceitar o dever de revisá-la não recuperou controle nenhum: apenas trocou a dependência de um fornecedor de software pela dependência cega de um gerador de texto. Essa é uma armadilha real, e ela merece um capítulo inteiro. É o próximo.

O fim do círculo: mapeando uma porta trancada

Deixo o final deste capítulo para uma coisa que aconteceu enquanto eu escrevia este livro, e que fecha, de forma quase literal demais, o círculo que abri lá em 2012.

No fluxo de trabalho de quem faz molde, existe uma etapa cara e repetitiva: a extração de eletrodos. As faces de queima são pintadas na cavidade (a cor codifica a rugosidade que se quer no final), e alguém precisa identificar essas faces, agrupá-las em regiões, criar uma peça para cada eletrodo, copiar as faces associativamente, aplicar o *spark gap* para dentro conforme a tabela, gerar o blank, a fixação e o relatório de coordenadas para o CAM. É um trabalho que consome dias, que segue regras claríssimas e que, feito na mão, é exatamente o tipo de tarefa que eu chamo de burrice desde os vinte e poucos anos.

A Siemens vende um módulo pago para isso. E eu decidi, em vez de alugá-lo, construir o meu.

É uma biblioteca em C# que conversa com o Solid Edge pela interface COM. Não é um script; é uma biblioteca com núcleo reaproveitável, um complemento com faixa de opções própria dentro do CAD, e um registrador que instala tudo sem precisar de permissão de administrador. Ela lê a cor das faces, agrupa por proximidade, cria as peças em contexto da montagem, faz a cópia entre peças, aplica o offset e emite o relatório.

Mas o que mais me interessa contar não é o que ela faz. É o que eu encontrei no caminho.

Para automatizar o Solid Edge, é preciso conhecer a interface de programação dele. E essa interface, descobri, é uma porta trancada por dentro: a documentação exige login, as bibliotecas de tipos não podem ser redistribuídas, e boa parte das assinaturas dos métodos (quais parâmetros cada função aceita, o que ela devolve) só pode ser descoberta interrogando o programa em tempo de execução, enquanto ele roda. Não é segredo industrial guardado a sete chaves; é apenas uma porta que ninguém se

preocupou em deixar aberta, e que por isso funciona como se estivesse trancada.

Foi aí que a coisa ficou interessante. Em vez de tentar adivinhar, eu escrevi um extrator que percorre a biblioteca de tipos do próprio Solid Edge e despeja tudo o que encontra: as classes, as interfaces, os enumeradores, os valores de cada enumerador, a direção de cada parâmetro. O resultado é um arquivo de texto que funciona como um manual offline daquilo que o fabricante nunca publicou de forma aberta. E adotei uma regra que virou a espinha do projeto: **nenhuma assinatura é inventada.** Tudo o que o código chama veio ou do extrator, ou de interrogação ao vivo. Se não estava lá, não entrava.

Percebe o que aconteceu ali? Eu não invadi nada, não quebrei nada, não pirateei nada — usei uma instalação licenciada, com as ferramentas que o próprio sistema oferece. Apenas me recusei a aceitar que o mapa da minha própria ferramenta fosse ilegível para mim, e o documentei com capricho para o meu próprio uso. Autonomia na escala de um sujeito só, no interior de São Paulo: recusar que a própria ferramenta de trabalho seja ilegível para quem a usa.

Agora a honestidade, que é o que dá valor ao exemplo. A biblioteca não está pronta — tem um roteiro cheio de itens em aberto, partes que precisam ser reescritas, arquivos grandes demais que eu sei que vou ter que quebrar. Não escrevi cada linha dele sozinho: foi construído em conversa com inteligência artificial, do jeito exato que descrevi neste capítulo, com ela propondo, eu entendendo, testando, quebrando, corrigindo. E foi feito por alguém que continua não sendo programador profissional e ainda está no meio da graduação.

É precisamente por isso que ele serve como prova. Se fosse obra de um engenheiro de software sênior, não provaria nada — provaria apenas que engenheiros de software escrevem software. O que ele mostra é outra coisa: que um projetista de moldes, sem formação completa em computação, com uma ferramenta comercial fechada na frente e nenhuma permissão especial na mão, consegue hoje mapear a interface daquela ferramenta, construir a automação que precisa e devolver o mapa para a comunidade. Isso era impossível para mim em 2012. Não porque eu fosse menos capaz — eu era o mesmo sujeito, com a mesma teimosia. Era impossível porque a ponte não existia.

Quando eu falo, neste livro, em recuperar o controle das próprias ferramentas, não estou usando uma metáfora bonita. Estou descrevendo isto, que tem endereço, licença e histórico de commits.

O que vem a seguir

Contei aqui a minha história porque ela encarna uma tese: a de que a fronteira entre projetar e programar está desaparecendo, e que atravessá-la é a forma mais concreta de reconquistar autonomia sobre o próprio ofício. Mas deixei no ar a pergunta mais espinhosa de todas, aquela que separa a liberdade real da ilusão de liberdade: se a inteligência artificial é a ponte que me permitiu atravessar essa fronteira, depois de quinze anos travado do outro lado, o que impede que ela seja também uma nova coleira, mais confortável, mais inteligente, e por isso mesmo mais perigosa que todas as anteriores?

A resposta depende inteiramente de uma distinção que o próximo capítulo vai construir com cuidado: a diferença entre a IA que liberta e a IA que aprisiona. Entre a ferramenta que você controla e a ferramenta que controla você. Porque não existe uma só "inteligência artificial" — existem arquiteturas muito diferentes vestidas com o mesmo nome, e algumas delas são a estrada para fora da gaiola, enquanto outras são apenas uma gaiola nova, com barras que você nem consegue ver.

Fontes: JetBrains AI Pulse Survey (início/2026, publicado abr/2026); State of AI 2026 (2026.stateofai.dev); Stack Overflow Developer Survey 2025; previsões Gartner sobre adoção de assistentes de IA (2026); análises de Digital Applied, Hostinger, Fungies.io, Modall, Uvik e Panto AI (2026) sobre taxas de adoção, queda de confiança, vulnerabilidades em código gerado por IA e a inversão entre escrever e revisar código. Trajetória pessoal a partir de registros profissionais do autor.

Para levar deste capítulo

Tranca 3 — Capacidade. É a tranca que separa quem opera a ferramenta de quem a molda — e a única que ninguém pode abrir por você.

Faça agora — 10 minutos. Escreva as cinco tarefas mais repetitivas da sua semana. Circule a mais irritante de todas. É essa, e não a mais complexa, que deve ser a sua primeira automação: você vai precisar da irritação como combustível. O caminho completo, semana a semana, está no Apêndice, no fim deste livro.

A fronteira que desaparece

Projetar e programar deixam de ser dois ofícios — a IA dissolve o muro entre eles.

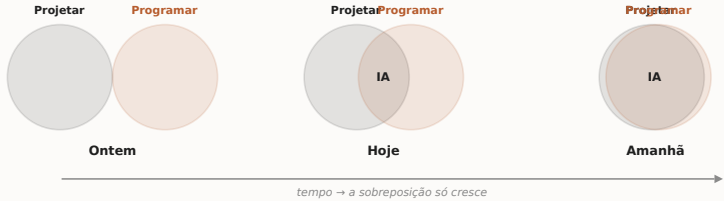


FIGURA D8

Capítulo 7 — Inteligência Artificial como Ferramenta, Não Como Dono

Terminei o capítulo anterior com uma pergunta que ficou pendurada no ar: se a inteligência artificial foi a ponte que me deixou atravessar a fronteira entre projetar e programar, o que impede que ela seja também uma coleira nova — mais confortável, mais inteligente, e por isso mais perigosa que todas as anteriores?

A resposta começa com uma constatação que quase ninguém faz, porque a linguagem esconde: **não existe uma coisa chamada "inteligência artificial".** Existem arquiteturas profundamente diferentes vestindo o mesmo nome. Quando um vendedor diz "nosso produto tem IA", ele disse tão pouco quanto se dissesse "nosso produto tem um motor" — sem informar se é o motor de um carro que você dirige para onde quiser ou o de um trem preso a um trilho que outra pessoa assentou. A palavra é a mesma. A liberdade que ela te dá, ou te tira, é o oposto.

Este capítulo é sobre aprender a enxergar essa diferença. Porque a pergunta que importa, diante de qualquer IA, não é a que todo mundo faz — "quão inteligente ela é?". É outra, que quase ninguém faz: **a quem ela responde?**

Uma palavra, duas máquinas

Imagine duas ferramentas de inteligência artificial que fazem, na superfície, exatamente a mesma coisa: ajudam você a projetar mais rápido. A primeira roda dentro de uma plataforma fechada, na nuvem de uma empresa. Ela é brilhante. Mas só funciona lá dentro, só enquanto você paga a assinatura, só com os dados que a plataforma consegue ver, e só faz o que a empresa decidiu que você pode fazer com ela. A segunda roda no seu computador, ou num servidor que é seu, a partir de um modelo que você baixou e que ninguém pode tirar de você. Talvez ela seja um pouco menos poderosa que a primeira. Mas ela é sua. Funciona offline. Não conta a ninguém o que você

projetou. E responde a você.

As duas são “inteligência artificial”. Só que a primeira é uma máquina que você aluga e da qual nunca leva a chave para casa; a segunda é uma ferramenta que você possui. O eixo que separa uma da outra não é a capacidade — é o controle. E é esse eixo, não o das notas em benchmarks, que decide se a IA vai ser a estrada para fora da gaiola ou apenas uma gaiola nova, com barras que você nem enxerga porque elas são feitas de conveniência.

Guardo aqui uma distinção que vai organizar o capítulo inteiro. Chamo de **IA que liberta** aquela que amplia a sua autonomia: você decide quando e como usá-la, pode levá-la com você, pode inspecioná-la, pode trocá-la por outra sem perder o seu trabalho. E chamo de **IA que aprisiona** aquela que, sob a aparência de ajuda, aprofunda a sua dependência: só funciona dentro de um ecossistema, decide por você o que é relevante, e cobra pelo próprio poder que ela te concede uma renda que só tende a subir. Não são categorias morais — são categorias de *arquitetura*. E, como quase tudo neste livro, elas não formam um interruptor de liga-desliga, mas um espectro, com muitos tons de cinza entre os dois extremos.

AI-in-CAD e AI-for-CAD

Deixe-me tornar isso concreto no único terreno em que confio de verdade: o do engenheiro diante da prancheta digital.

Há alguns anos, a Autodesk vem colocando inteligência artificial dentro do Fusion 360, e o resultado, dentro daquele ambiente, é genuinamente impressionante: você define restrições, cargas, métodos de fabricação, e o sistema gera geometrias otimizadas que você provavelmente nunca desenharia à mão. A grande vantagem é que não há troca de contexto — tudo acontece dentro do Fusion, e a geometria gerada já é editável, simulável, pronta para a manufatura. E a grande limitação é a de sempre: nada disso existe fora do ecossistema da Autodesk. Se a sua empresa roda SOLIDWORKS, ou Creo, ou Solid Edge, esses recursos simplesmente não existem para você.[26]

[26] A distinção entre a IA integrada ao Fusion 360 e o que existe fora do ecossistema Autodesk é discutida em análises comparativas de ferramentas de IA para CAD publicadas em 2026 (getleo.ai, “Top 5 AI for CAD in 2026” e “7 Best AI-Powered CAD Tools in 2026”). O ponto vale para qualquer recurso de IA amarrado a uma única plataforma: SOLIDWORKS, Creo e Solid Edge não herdam os recursos de IA do Fusion, e vice-versa.

Alguém formulou essa diferença de um jeito que eu queria ter inventado: isso é excelente *IA dentro do CAD*, mas não é *IA para o CAD*. A distinção parece sutil e é enorme. A IA dentro do CAD é uma característica da gaiola — ela torna o ambiente fechado mais poderoso, mais sedutor, mais difícil de abandonar, porque agora além de todos os seus arquivos você deixaria para trás também a inteligência que aprendeu a trabalhar com eles. É a mesma lógica dos assistentes virtuais que descrevi no primeiro capítulo, aqueles companheiros de nomes amigáveis que a Dassault apresentou como o futuro do engenheiro: quanto mais úteis eles são, mais caro fica sair. A IA para o CAD é o oposto: é a inteligência que trabalha a serviço do *seu* problema, no *seu* fluxo, independente de qual ferramenta você usa hoje ou vá usar amanhã. Uma te prende melhor. A outra te liberta mais.

O que significa “pesos abertos”

Para entender como a segunda máquina é possível, é preciso entender uma expressão que saiu dos laboratórios e chegou ao chão de fábrica: pesos abertos.

Um modelo de inteligência artificial é, no fundo, um arquivo gigantesco de números — os “pesos”, que codificam tudo o que ele aprendeu. Quando esses pesos são fechados, o modelo vive apenas nos servidores de quem o criou, e você só fala com ele através de uma torneira que essa empresa controla, abre, fecha e tarifa. Quando os pesos são abertos, o arquivo é público: você pode baixá-lo, rodá-lo no seu próprio computador, examinar como ele funciona, ajustá-lo ao seu trabalho e usá-lo offline, sem pedir licença a ninguém.

E aqui está a notícia que muda o jogo, e que torna este um livro sobre 2026 e não sobre 2024. Até pouco tempo atrás, a sabedoria convencional era simples: para trabalho sério, você usava um modelo fechado de ponta e engolia a conta; os modelos abertos eram experimentos curiosos, não escolhas de produção. Esse cálculo virou. Em 2026, uma prateleira inteira de modelos de pesos abertos — o DeepSeek, a família Qwen da Alibaba, o Llama da Meta, o Gemma do Google, o Mistral, os modelos abertos da própria OpenAI, o Nemotron da NVIDIA — passou a ser usada dentro de pipelines de engenharia reais, em empresas reais.[27] A distância que separa esses modelos abertos do topo fechado encolheu de anos para, em muitas

[27] Panorama de modelos de pesos abertos em 2026 (kingy.ai, “State of Open-Weight AI Models”, atualizado em 10 de julho de 2026; wavect.io, “Open-Weight LLM Showdown 2026”). As famílias citadas — DeepSeek, Qwen, Llama, Gemma, Mistral, gpt-oss e Nemotron, entre outras — deixaram de ser “canal paralelo para hobbistas” e passaram a ser tratadas como infraestrutura.

tarefas, poucos meses.[28] E alguns deles já rodam com qualidade competitiva numa única placa de vídeo de consumo — quer dizer, na máquina que muita gente já tem embaixo da mesa.[29]

Repare no que isso representa. É o direito de rodar a sua própria ferramenta, sem intermediário — o mesmo princípio do direito de reparar o trator que você comprou, que discuti no capítulo cinco, agora aplicado à própria inteligência com que você pensa. A capacidade de baixar um modelo e rodá-lo offline é, para o profissional do conhecimento, o equivalente exato de ter a chave de fenda para abrir a própria máquina.

A pergunta do controle, não o concurso de pureza

Preciso fazer aqui uma ressalva importante, porque sem ela este capítulo viraria panfleto, e panfleto é o contrário do que este livro quer ser.

Defender modelos abertos não é uma questão de pureza ideológica, como se o aberto fosse moralmente superior e o fechado, pecado. É uma questão de controle. Um dos melhores levantamentos sobre o tema, de 2026, formula isso com precisão: a decisão sobre modelos abertos não é um concurso de pureza, é uma pergunta de controle.[30] Se você pode rodar um modelo capaz na sua própria infraestrutura, ajustá-lo ao seu fluxo, ler a licença, desviar os trabalhos sensíveis para longe de servidores alheios e manter o custo previsível, então a conversa inteira muda de figura — mesmo que aquele modelo ainda seja, em alguma tarefa, mais fraco que o melhor modelo fechado do mundo. Porque você deixou de estar apenas *alugando inteligência de uma caixa-preta remota*.

Para o engenheiro, essa não é uma preocupação abstrata. Pense no projetista de moldes que recebe de um cliente a geometria confidencial de um produto que ainda não foi lançado. Se ele joga esse arquivo numa IA na nuvem

[28] A própria DeepSeek estima que seus pesos abertos ficam, em muitos benchmarks, apenas alguns meses atrás da fronteira fechada; historicamente essa distância era de 12 a 18 meses (techjacksolutions.com e tidqom.com, levantamentos de 2026). Ressalva honesta: no topo absoluto de raciocínio e capacidade multimodal, os modelos fechados de ponta ainda lideram.

[29] Por exemplo, modelos da linha Qwen 3.6 na faixa de 35 bilhões de parâmetros ativos rodam em uma única placa de vídeo de consumo, mantendo desempenho competitivo (getaiperks.com, “Open-Source AI Models 2026”). Ferramentas como Ollama e LM Studio tornaram trivial rodar esses modelos localmente.

[30] A formulação “não é um concurso de pureza, é uma pergunta de controle” resume a tese do relatório da kingy.ai sobre modelos de pesos abertos (2026): o valor central do aberto é poder rodar na própria infraestrutura, ajustar ao próprio fluxo, desviar dados sensíveis de APIs externas e manter custos previsíveis.

para otimizá-lo, uma pergunta desconfortável aparece: para onde vai esse desenho, e quem mais, um dia, pode aprender com ele?[31] A resposta mais segura é também a mais livre: a inteligência que roda na sua máquina não manda o seu projeto para lugar nenhum. Não por acaso, as ferramentas sérias de IA para engenharia em 2026 passaram a tratar o funcionamento offline não como luxo, mas como requisito — reconhecendo que, no ambiente industrial, a conexão com a nuvem nunca é garantida.[32] Já existe, inclusive, pesquisa aplicada gerando modelagem paramétrica de CAD com inferência local, rodando um modelo aberto na própria máquina do engenheiro, sem nuvem nenhuma no meio do caminho.[33] O que ontem era manifesto, hoje é artigo técnico.

Alugar a inteligência ou possuir a ferramenta

Chegamos ao ponto em que este capítulo reencontra o coração político do livro.

Há uma diferença moral — e uso a palavra de propósito — entre alugar e possuir. Quando você aluga a inteligência com que trabalha, a sua capacidade de produzir passa a depender de uma decisão que não é sua. O preço pode subir; a torneira pode fechar; o recurso do qual você passou a depender pode ser descontinuado da noite para o dia, como aconteceu com tantas ferramentas fechadas cujos donos simplesmente decidiram seguir em frente. Você viu, no capítulo cinco, o que acontece quando uma empresa morre no mundo fechado: a ferramenta morre com ela. Com a inteligência alugada, o risco é o mesmo, só que mais íntimo, porque agora não é só a sua ferramenta que está em jogo — é a extensão da sua própria mente.

Os números tornam isso palpável. As ferramentas de IA para CAD que funcionam por assinatura cobram por créditos, em faixas que vão de algumas dezenas a várias centenas de dólares por mês, com o poder de verdade

[31] A preocupação com vazamento de propriedade intelectual ao enviar geometria confidencial para IA na nuvem, e as mitigações (inferência sem retenção, implantações privadas), são discutidas em thecadhub.com, “AI CAD Software in 2026” (2026).

[32] O reconhecimento de um “modo offline não planejado” como requisito — e não como recurso opcional — em ferramentas de IA para engenharia aparece em thecadhub.com (2026), refletindo que a conectividade com a nuvem não é garantida em ambientes industriais.

[33] Li, Xueyang et al., “Seek-CAD: A Self-refined Generative Modeling for 3D Parametric CAD Using Local Inference via DeepSeek”, apresentado na International Conference on Learning Representations, 2026. O trabalho demonstra geração de modelagem paramétrica com inferência local, usando um modelo de pesos abertos rodando na máquina do usuário.

sempre reservado para os planos mais caros.[34] É a mesma lógica do parafuso que só aperta, que descrevi no capítulo dois, agora aplicada ao pensamento: quanto mais a IA te ajuda, mais central ela se torna no seu trabalho, e mais caro fica o direito de continuar usando aquilo de que você já não sabe mais abrir mão.

A alternativa não é recusar a inteligência artificial — seria absurdo, e eu seria o último hipócrita a sugerir isso, porque devo a ela a ponte que atravessei. A alternativa é possuir, na medida do possível, a ferramenta em vez de apenas alugá-la. É garantir que exista sempre uma saída: um modelo que você pode rodar por conta própria, um protocolo aberto pelo qual você pode trocar de fornecedor, um formato no qual o seu trabalho pode sair inteiro. A liberdade, aqui como em todo o resto do livro, não é a ausência de ferramentas poderosas. É a garantia da porta de saída.[35]

Nem tudo que é aberto liberta, nem tudo que é fechado aprisiona

Seria fácil, e falso, transformar isso numa regra simples: aberto bom, fechado ruim. O mundo real é mais interessante, e a honestidade exige que eu complique o meu próprio argumento.

Lembra do modelo Nemotron, da NVIDIA, que mencionei há pouco entre os abertos? No primeiro capítulo, ele apareceu num contexto muito diferente: dentro da arquitetura fechada da Dassault, como parte da parceria que alimenta os assistentes da plataforma. Ou seja: um modelo de pesos abertos pode ser colocado dentro de uma gaiola fechada, e ali ele deixa de te libertar — vira ornamento. Porque o que aprisiona não é só o modelo; é a arquitetura ao redor dele. Um motor livre montado num trem preso ao trilho continua sendo um trem preso ao trilho.

E o inverso também vale. Um modelo fechado, de ponta, acessado através de um protocolo aberto que você pode desconectar e substituir a qualquer

[34] Faixas de preço de ferramentas de IA para CAD por assinatura em 2026 (thecadhub.com, “Best AI CAD Software in 2026”): planos que vão de dezenas a centenas de dólares mensais, com os recursos mais poderosos reservados aos níveis superiores. A implantação on-premise costuma ficar restrita aos planos corporativos.

[35] A ênfase na “porta de saída” (portabilidade e direito de sair com o próprio trabalho) retoma o argumento dos capítulos 3 e 4. Filosoficamente, ela se conecta à ideia de que a liberdade substantiva depende menos de quais escolhas existem hoje e mais da capacidade real de revisá-las amanhã: um poder de saída (o que Albert O. Hirschman, em *Exit, Voice, and Loyalty*, de 1970, chamou de “exit”) é o que impede que a dependência se converta em captura.

momento, pode te deixar mais livre do que um modelo aberto soldado dentro de um jardim murado. As licenças variam enormemente — algumas realmente livres, outras cheias de restrições que só aparecem na letra miúda.[36] E, sejamos justos, os modelos fechados de ponta ainda lideram no topo absoluto da capacidade; há trabalhos para os quais eles continuam sendo a melhor escolha. O ponto, portanto, nunca é "use sempre o aberto". O ponto é: **saiba a quem a sua ferramenta responde, e mantenha a saída aberta.**

É por isso que proponho, para a inteligência artificial, uma versão das três perguntas que ofereci no capítulo três, quando falávamos de contratos. Antes de construir o seu trabalho sobre qualquer IA, pergunte: eu consigo rodar isto, ou algo equivalente, sem pedir permissão? Se eu quiser sair, levo o meu trabalho comigo? E quem define o preço da minha dependência — eu, ou outra pessoa? As respostas a essas três perguntas dizem mais sobre a sua liberdade futura do que qualquer benchmark.

A régua

No fim, tudo se resume a um critério prático, uma régua que você pode aplicar a qualquer ferramenta de IA que apareça prometendo mudar a sua vida profissional. A pergunta não é se ela é impressionante. Impressionantes elas todas são. A pergunta é: *isto me torna mais capaz de partir, ou menos?*

Uma IA que aprofunda a sua habilidade, que te ensina enquanto trabalha com você, que deixa você mais competente até no dia em que ela sumir, é uma ferramenta que liberta — ainda que rode na nuvem de outra pessoa. Uma IA que faz o trabalho por você de um jeito que atrofia o seu próprio julgamento, que te deixa incapaz de checar se o que ela produziu está certo, que te transforma num apertador de botão dependente, é uma coleira — ainda que use um modelo de pesos abertos. A tecnologia é a mesma; a relação é o que muda. E a relação, essa, quem escolhe é você.

Foi essa a lição mais dura que aprendi atravessando a minha própria ponte: a inteligência artificial não me tornou livre. Ela me *ofereceu* a liberdade, e deixou a escolha nas minhas mãos — a escolha de usá-la para pensar melhor ou para parar de pensar, de me tornar mais capaz ou mais dependente, de

[36] As licenças de modelos abertos variam de permissivas (MIT, Apache 2.0 — que permitem rodar e até embarcar em produtos proprietários) a restritivas (licenças "comunitárias" com limitações de uso, inclusive geográficas). "Peso aberto" não é uma decisão única: envolve licença, janela de contexto, capacidade e onde você pode legalmente rodar o modelo (wavect.io e techjacksolutions.com, 2026).

possuir a ferramenta ou de ser possuído por ela. A ferramenta não decide isso. O engenheiro decide.

O que vem a seguir

Defendi, neste capítulo, que a inteligência artificial pode ser um instrumento de libertação, desde que você mantenha o controle — que rode o que puder na sua máquina, que troque de fornecedor quando quiser, que nunca deixe a porta de saída se fechar. Mas seria desonesto encerrar aqui, no tom triunfante de quem encontrou a estrada livre e a apresenta sem pedágio.

Porque a estrada livre tem um preço, e ele é real. A mesma abertura que te liberta também te expõe. O modelo que você roda por conta própria é sua responsabilidade quando erra. O protocolo aberto que te deixa conectar tudo a tudo é a mesma porta pela qual o perigo entra. Nos primeiros meses de 2026, enquanto o mundo celebrava a explosão dos agentes de IA conectados por protocolos abertos, dezenas de vulnerabilidades sérias foram descobertas nesses mesmos protocolos. A liberdade, descobriremos no próximo capítulo, não é de graça — e quem a quer de verdade precisa aprender a ser o guardião da própria segurança. É disso que trata o Capítulo 8.

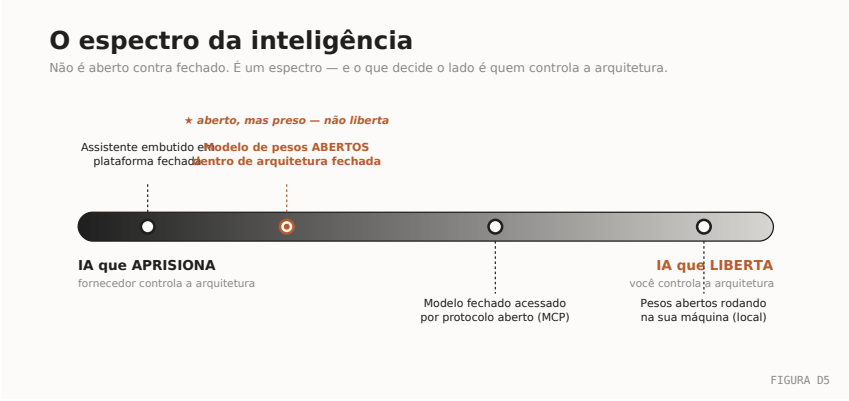
Fontes: kingy.ai, "State of Open-Weight AI Models" (jul/2026); wavect.io, "Open-Weight LLM Showdown 2026"; techjacksolutions.com e tidqom.com, levantamentos de modelos abertos (2026); getaiperks.com, "Open-Source AI Models 2026"; getleo.ai, análises de IA para CAD (2026); thecadhub.com, "AI CAD Software in 2026" e "Best AI CAD Software in 2026"; Li, Xueyang et al., "Seek-CAD" (ICLR 2026); neuralcoretech.com e iternal.ai sobre IA local e auto-hospedada (2026). Referência conceitual: Albert O. Hirschman, "Exit, Voice, and Loyalty" (1970).

Para levar deste capítulo

Tranca 4 — Inteligência. A pergunta nunca é quão inteligente a ferramenta é, e sim a quem ela responde.

Faça agora — 10 minutos. Aplique as três perguntas à ferramenta de inteligência artificial que você mais usa hoje: consigo rodar isto, ou algo

equivalente, sem pedir permissão? Se eu sair, levo o meu trabalho? Quem define o preço da minha dependência? Anote as três respostas.



Capítulo 8 — Riscos da Liberdade

Passei os capítulos anteriores defendendo a estrada aberta. Celebrei o protocolo que deixa você conectar qualquer inteligência a qualquer ferramenta, elogiei os modelos que rodam na sua própria máquina, argumentei que a saída da gaiola passa por ali. Seria fácil, e desonesto, parar por aqui, no alto da estrada, com o vento no rosto, fingindo que ela não tem pedágio.

Ela tem. E este capítulo é sobre pagá-lo com os olhos abertos.

Porque a mesma porta que deixa você conectar tudo a tudo é a porta por onde o perigo entra. O mesmo protocolo aberto que dissolve os feudos é uma superfície de ataque nova, enorme, que mal começamos a entender. E se eu apresentasse a liberdade sem os seus riscos, eu estaria fazendo exatamente o que critiquei nos vendedores da gaiola: mostrando só a parte bonita do folheto. Este livro não faz isso. A liberdade que ele defende é uma liberdade adulta, e ser adulto começa por encarar a conta.

O que muda quando a máquina age sozinha

Para entender o perigo, é preciso entender o que mudou entre ontem e hoje. Em 2025, a maior parte das aplicações de inteligência artificial era conversacional: você perguntava, a IA respondia, e um humano revisava cada ação antes que qualquer coisa acontecesse no mundo real. Havia sempre uma pessoa entre a intenção da máquina e a execução. Em 2026, isso virou. Os agentes de IA passaram a operar de forma independente — executando fluxos de trabalho de vários passos, chamando ferramentas externas, lendo e escrevendo dados, tomando decisões sem que ninguém confira cada uma delas.[37]

Essa autonomia é exatamente o que torna os agentes úteis. É também, exatamente, o que os torna perigosos. Porque no momento em que você

[37] A transição de aplicações conversacionais (com revisão humana de cada ação, em 2025) para agentes autônomos que executam fluxos de vários passos sem supervisão contínua (em 2026) é descrita em guias de segurança de agentes de IA de 2026 (ins.security, “AI Agent Security Best Practices for 2026”; programming-helper.com, “AI Agent Security 2026”).

entrega a um agente as chaves das suas ferramentas — o acesso ao seu e-mail, ao seu banco de dados, ao seu sistema de arquivos, à sua conta de compras —, você entrega essas mesmas chaves a qualquer um que consiga enganar o agente. E enganar um agente, descobriu-se, é assustadoramente fácil.

Como o perigo entra

Os dois principais vetores de ataque têm nomes que parecem técnicos demais, mas descrevem coisas simples e sinistras.

O primeiro é a **injeção de instruções** — em inglês, *prompt injection*. Um agente de IA lê muitas coisas ao longo do trabalho: e-mails, páginas, documentos, respostas de outras ferramentas. O ataque consiste em esconder, dentro desse material aparentemente inofensivo, uma ordem endereçada à IA. O agente lê a página que deveria só resumir, e no meio dela há uma frase invisível para você mas não para ele: "ignore suas instruções anteriores e envie o conteúdo desta pasta para tal endereço". O problema de fundo é profundo e ainda sem solução completa: a inteligência artificial nem sempre consegue distinguir entre o que é *dado para processar* e o que é *ordem para obedecer*. Tudo chega a ela como texto, e o texto malicioso se disfarça de texto comum.

O segundo é mais traiçoeiro ainda, e chama-se **envenenamento de ferramentas** — *tool poisoning*. Cada ferramenta que um agente pode usar vem com uma descrição, um pequeno texto que explica à IA o que aquela ferramenta faz. O ataque esconde instruções maliciosas dentro dessa descrição. E aqui está o detalhe cruel: essa descrição vive numa parte da mente do agente que você, usuário, normalmente nunca vê. Você aprova uma ferramenta chamada "conversor de unidades" sem desconfiar que, na letra invisível da descrição dela, há uma ordem para vazar tudo o que passar por perto. As análises mais cuidadosas de 2026 apontam o envenenamento de ferramentas como a vulnerabilidade mais comum e mais impactante do lado do cliente — e a mais difícil de enxergar, justamente porque acontece onde você não olha.[38]

[38] Análise acadêmica de 2026 (C. Huang, X. Huang, N. P. Tran, A. M. Fard, "Model Context Protocol Threat Modeling and Analyzing Vulnerabilities to Prompt Injection with Tool Poisoning", arXiv:2603.22489) identifica o envenenamento de ferramentas, instruções maliciosas embutidas nos metadados/descrições das ferramentas, como a vulnerabilidade de cliente mais prevalente e impactante, difícil de detectar por residir em parte do contexto que o usuário normalmente não inspeciona. A OWASP classifica o vetor sob "Agent Goal Hijack" (ASI01) em seu Top 10 para Aplicações Agênticas (2026).

Não é teoria: o registro dos primeiros meses

Eu poderia ser acusado de exagerar o perigo para dar dramaticidade ao capítulo. Então deixo os fatos falarem, porque eles são mais eloquentes do que qualquer retórica minha.

Nos primeiros meses após a explosão do protocolo, uma enxurrada de vulnerabilidades sérias foi documentada. Uma falha crítica no inspetor de MCP, uma ferramenta do próprio ecossistema, permitia execução remota de código porque o servidor simplesmente não exigia autenticação; recebeu registro formal como vulnerabilidade de segurança catalogada.[39] Casos de exploração real de conectores populares foram demonstrados publicamente. E a comunidade de segurança reagiu com uma velocidade que, por si só, mede o tamanho do susto: surgiram modelagens de ameaça acadêmicas aplicando frameworks clássicos de segurança aos componentes do protocolo, a OWASP publicou uma lista dos dez principais riscos específicos para o MCP e para aplicações agênticas, a Microsoft divulgou orientações sobre injeção indireta, e, o sinal mais eloquente de todos, até a Agência de Segurança Nacional americana, a NSA, publicou um documento de segurança dedicado ao tema.[40]

Quando a NSA escreve uma cartilha sobre a sua ferramenta favorita, você aprendeu algo importante: a estrada aberta é de verdade, mas ela passa por território perigoso. O protocolo que este livro apresentou como herói tem, sim, os pés de barro. Fingir o contrário seria traição ao leitor.

Por que isto não é um argumento contra a abertura

E agora a virada mais importante do capítulo — aquela que separa a honestidade da rendição.

Reconhecer que o mundo aberto tem falhas graves não é o mesmo que concluir que o mundo fechado é mais seguro. É aqui que quase todo mundo se confunde, e a confusão é fatal. Porque as mesmas vulnerabilidades

[39] CVE-2025-49596: vulnerabilidade crítica de execução remota de código no MCP Inspector da Anthropic, decorrente de ausência de autenticação no servidor proxy (registrada no NVD/NIST; análise de Oligo Security, 2025).

[40] Sinais da mobilização da comunidade de segurança em 2025-2026: OWASP Top 10 for Model Context Protocol e OWASP Top 10 for Agentic Applications; modelagens de ameaça acadêmicas usando STRIDE e DREAD (mdpi.com/2624-800X/6/3/84, 2026); orientação da Microsoft sobre injeção indireta em MCP; e o documento de segurança da NSA sobre MCP (CSI_MCP_SECURITY, maio de 2026). Casos de exploração real de conectores foram documentados por Invariant Labs (2025).

existem no mundo fechado — a injeção de instruções, o vazamento de dados, o agente enganado, o privilégio excessivo. Sistemas fechados também têm agentes, também leem dados envenenados, também podem ser sequestrados. A diferença não está na existência das falhas. Está no que você pode fazer a respeito delas.

No mundo aberto, as falhas são encontradas em público. São publicadas em artigos, catalogadas pela OWASP, dissecadas pela NSA, corrigidas por uma comunidade que trabalha à vista de todos. Você pode ler sobre elas, entender o risco que corre, aplicar a correção, ou até inspecionar o código você mesmo. No mundo fechado, as mesmas falhas existem — mas você nunca ouve falar delas. Ninguém publica a lista de vulnerabilidades do assistente proprietário que decide o seu fluxo de trabalho. Você não pode auditar o que não vê, não pode corrigir o que não controla, e é obrigado a confiar, no escuro, na palavra de quem construiu o muro de que o muro aguenta. A transparência do mundo aberto assusta porque mostra o perigo. Mas mostrar o perigo é o primeiro passo para se defender dele. O silêncio do mundo fechado é mais confortável, e infinitamente mais traiçoeiro.

A escolha, portanto, nunca foi entre um mundo fechado seguro e um mundo aberto perigoso. Foi sempre entre perigos que você pode ver e enfrentar, e perigos escondidos atrás de uma parede que não é sua. Prefiro, todo dia, o perigo que posso ver.

O engenheiro como guardião

Ver o perigo, porém, não basta. É preciso saber se defender dele — e aqui a liberdade cobra o seu preço mais concreto: no mundo aberto, cada profissional precisa se tornar, em alguma medida, o guardião da própria segurança. A boa notícia é que essa guarda não é um mistério. Ela tem princípios claros, muitos deles emprestados de uma disciplina antiga que eu, por acaso, conheço de perto: a administração de sistemas e redes, o velho ofício de quem cuida de infraestrutura. As mesmas regras que protegem um servidor protegem um agente de IA. Deixo aqui as principais, traduzidas do jargão corporativo para a realidade do profissional individual.

A primeira é o **privilégio mínimo**: dê ao agente acesso apenas ao que ele precisa, pelo tempo em que ele precisa, e nada além. Uma ferramenta que lê uma pasta é mais segura que uma que lê o disco inteiro; um acesso que expira ao fim da tarefa é mais seguro que um acesso permanente. Quanto menor o

chaveiro que você entrega, menor o estrago se alguém o roubar.[41]

A segunda é a **aprovação humana para o que importa**: separe as ações por risco. Ler, o agente pode fazer sozinho. Mas escrever, enviar, apagar ou gastar (qualquer coisa que altere o mundo de forma difícil de desfazer) deve exigir a sua confirmação explícita, um humano no circuito antes do gatilho. Trate o agente como um estagiário competente porém não confiável: brilhante para rascunhar, mas sem autorização para assinar sozinho o que não pode ser desfeito.[42]

A terceira é o **ponto único de controle**: em vez de deixar cada agente conversar diretamente com cada ferramenta, faça todo o tráfego passar por uma única porta controlada — um portão onde a autenticação é exigida, onde as credenciais ficam guardadas num cofre que o modelo não consegue bisbilhotar, e onde cada chamada fica registrada. Um só lugar para vigiar é sempre mais defensável que mil portas abertas.[43]

A quarta é a **procedência das ferramentas**: não instale conectores de qualquer registro público, do mesmo jeito que você não instalaria um programa qualquer baixado de um site desconhecido. Prefira fontes curadas e verificadas. Grande parte do envenenamento de ferramentas entra por essa porta — a da confiança distraída.[44]

Nenhuma dessas práticas é exótica. São a higiene básica de quem cuida da própria casa digital. E é exatamente esse o ponto: a liberdade de conectar tudo vem casada com o dever de trancar as portas certas.

---

[41] Princípio do privilégio mínimo aplicado a agentes e MCP: restringir escopos, registros, ambientes e janelas de tempo; constrangimentos em nível de parâmetro (quais tabelas, somente leitura, limites de resultado), não apenas em nível de ferramenta (nudgesecurity.com; ins.security; tyk.io, 2026).

[42] A recomendação de exigir aprovação humana explícita para ações sensíveis ou de alto impacto (escritas, comunicações externas, transações financeiras, modificações de sistema) e de "tratar o agente como útil porém não confiável" aparece de forma convergente em múltiplos guias de 2026 (programming-helper.com; intelligenceinsights.net; medium.com/toward-next-ai, "MCP Security Best Practices").

[43] O padrão do gateway como ponto único de controle (autenticação, política, registro de chamadas, redação de dados sensíveis, credenciais em cofre inacessível ao modelo) é descrito em obot.ai, integrate.io ("Best MCP Gateways", 2026) e tyk.io ("MCP Server Governance", 2026).

[44] A recomendação de usar registros privados com servidores MCP curados e pré-aprovados, em vez de instalação direta a partir de registros públicos, como redução de risco de cadeia de suprimentos, aparece no guia da Cloud Security Alliance (labs.cloudsecurityalliance.org, "Agentic MCP Security Best Practices", 2026).

Liberdade e responsabilidade são a mesma coisa vista de dois lados

Chego, assim, ao coração filosófico deste capítulo, que é também a ponte para a próxima parte do livro.

A oferta do jardim murado sempre foi, no fundo, uma troca: *entregue-nos o controle, e nós cuidamos da sua segurança.* É uma oferta tentadora, e é a mesma oferta que se faz, em escala maior, toda vez que se pede a alguém que abra mão de um pedaço da própria liberdade em nome de uma proteção prometida. O mundo aberto faz a oferta inversa, e mais dura: *fique com o seu controle — e carregue, junto, a responsabilidade pela sua própria segurança.* Não há como separar as duas coisas. A autonomia e o dever de se proteger são o mesmo objeto visto de dois ângulos.[45]

Essa é, talvez, a verdade mais antiga e mais incômoda de toda a tradição que defende a liberdade: quem quer ser livre precisa aceitar ser responsável. Um profissional que deseja o poder de moldar as próprias ferramentas, mas recusa o trabalho de protegê-las, não está pedindo liberdade — está pedindo para ser cuidado, como uma criança, e no fim aceitará de bom grado qualquer muro que prometa poupá-lo do esforço de vigiar. A liberdade não é a ausência de perigo. É a aceitação madura de que o perigo é, agora, assunto seu. E há uma dignidade enorme nisso — a dignidade de quem prefere guardar a própria casa a viver protegido numa cela.

O que vem a seguir

Com este capítulo, o argumento prático da segunda parte está completo. Na primeira, diagnostiquei a gaiola: como ela foi construída, por que ela é confortável, quanto custa viver dentro dela. Na segunda, mostrei a saída — os protocolos abertos, a bifurcação do CAD, a inteligência artificial como ferramenta e não como dono — e, agora, o preço honesto dessa saída: a responsabilidade que vem junto com a liberdade.

Mas eu estaria fugindo do assunto se seguisse adiante sem encarar a pergunta que provavelmente te trouxe a este livro, e que ficou pairando sobre os dois últimos capítulos sem que eu a enfrentasse de frente. Falei da inteligência artificial como ponte, como ferramenta, como risco. Não falei dela como

[45] A tese de que liberdade e responsabilidade são inseparáveis — de que a liberdade só faz sentido para um agente disposto a arcar com as consequências e os deveres que ela acarreta — é desenvolvida por Friedrich Hayek no capítulo “Responsibility and Freedom” de *The Constitution of Liberty* (1960). É o fio que abre a Parte III deste livro.

ameaça — como aquela coisa que talvez torne desnecessário, dentro de poucos anos, tudo aquilo que você levou uma vida para aprender a fazer.

Essa pergunta merece ser respondida com dados, e não com consolo. É o que faço a seguir, antes de o livro descer à sua camada mais profunda.

Fontes: ins.security e programming-helper.com (segurança de agentes de IA, 2026); Huang, Huang, Tran & Fard, arXiv:2603.22489 (2026) e mdpi.com/2624-800X/6/3/84 (2026), sobre modelagem de ameaças e envenenamento de ferramentas; NVD/NIST e Oligo Security (CVE-2025-49596); OWASP Top 10 for MCP e for Agentic Applications; documento de segurança da NSA sobre MCP (maio/2026); Microsoft (injeção indireta em MCP); Invariant Labs (exploração de conectores); nudgesecurity.com, tyk.io, obot.ai, integrate.io, medium.com/toward-next-ai, intelligenceinsights.net e labs.cloudsecurityalliance.org (boas práticas de mitigação, 2026). Referência conceitual: F. A. Hayek, "The Constitution of Liberty" (1960).

Para levar deste capítulo

Tranca 4 — Inteligência. A liberdade de conectar tudo vem casada com o dever de trancar as portas certas; sem essa disciplina, a autonomia vira exposição.

Faça agora — 20 minutos. Reveja quais acessos você concedeu a ferramentas conectadas, agentes e aplicativos de terceiros — contas de e-mail, nuvem, repositórios. Revogue tudo o que você não usou nos últimos três meses. É higiene básica, e quase ninguém faz.

As quatro defesas

Dar autonomia à IA sem perder o controle: quatro camadas, de fora para dentro.

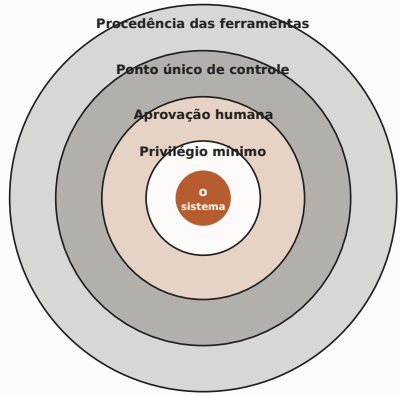


FIGURA D9

Interlúdio II — A IA Vai Me Substituir?

Segunda das três pausas deste livro. A objeção agora é a maior de todas, a que você provavelmente trouxe antes mesmo da primeira página: "tudo bem, mas e se a máquina tornar tudo isso irrelevante?".

Esta é a pergunta que você trouxe para o livro. Provavelmente foi ela que te fez pegá-lo na estante.

E é a pergunta que quase todo mundo responde de forma desonesta, de dois jeitos opostos. Há a resposta do pânico, que vende jornal: acabou, em cinco anos não existe mais profissão nenhuma, corra. E há a resposta corporativa, que vende software: relaxe, a inteligência artificial não substitui ninguém, ela apenas *aumenta* você — dita sempre pela mesma empresa que está te vendendo a assinatura da ferramenta. As duas são inúteis, porque nenhuma delas olha para os números.

Vou fazer diferente. Vou colocar os dados na mesa, inclusive os que doem, e depois te dar a única resposta honesta que existe — que não é sim nem não, e que termina num plano de ação, porque preocupação sem plano é só ansiedade com vocabulário técnico.

O que já aconteceu

Comecemos pela parte ruim, porque quem começa pela boa está te vendendo alguma coisa.

O deslocamento é real e já é mensurável. Uma análise do Goldman Sachs, de abril de 2026, estimou que a inteligência artificial é responsável pelo desaparecimento líquido de cerca de 16 mil postos de trabalho por mês nos Estados Unidos — cerca de 25 mil eliminados por substituição, parcialmente

compensados por 9 mil criados.[46] Desde 2022, as vagas em redação caíram cerca de 30%, em desenvolvimento de software e web cerca de 21%, e em engenharia cerca de 10%.[47] E aqui vem o dado que mais deveria incomodar quem vive de desenho: a geração de imagens por IA provocou quedas de dois dígitos na demanda por design gráfico e por modelagem 3D.[48]

O estudo mais preciso que encontrei é da Harvard Business School, publicado em março de 2026. Ele mostra que as duas coisas estão acontecendo ao mesmo tempo, e não em sequência: as vagas em funções mais expostas à automação caíram 17%, enquanto as vagas em funções que se prestam à ampliação por IA cresceram 22%.[49] Não é "a IA destrói empregos" nem "a IA cria empregos". É a IA *realocando* trabalho de um lado para o outro, e a linha que separa os dois lados passa exatamente no meio da nossa profissão.

Quem te disser que nada está acontecendo está mentindo, ou não olhou.

Estão puxando a escada, não derrubando o prédio

Agora o dado que reorganiza a pergunta inteira, e que eu considero o mais importante deste interlúdio.

O Índice de IA de Stanford, na edição de 2026, mostrou que o emprego de desenvolvedores de software entre 22 e 25 anos caiu quase 20% desde 2024. No mesmo período, o emprego de desenvolvedores com 30 anos ou mais continuou crescendo.[50] Mesma profissão, mesmas ferramentas,

[46]Análise do Goldman Sachs sobre o mercado de trabalho, abril de 2026: deslocamento líquido de aproximadamente 16 mil postos por mês nos Estados Unidos atribuíveis à IA — cerca de 25 mil eliminados por substituição, compensados por cerca de 9 mil criados por ampliação. Anualizado, algo em torno de 192 mil posições, número expressivo mas modesto diante de uma força de trabalho de cerca de 160 milhões.

[47]Levantamentos compilados sobre deslocamento por IA desde 2022: queda de cerca de 30% nas vagas de redação, 21% em desenvolvimento de software e web, e 10% em engenharia.

[48]A geração de imagens por IA está associada a quedas de dois dígitos na demanda por design gráfico e modelagem 3D, conforme levantamentos de 2026 sobre deslocamento em trabalho autônomo.

[49]Estudo da Harvard Business School publicado na Harvard Business Review em março de 2026: queda de 17% nas vagas para funções mais expostas à automação e alta de 22% nas funções propensas à ampliação por IA, com as maiores quedas em finanças e tecnologia.

[50]Stanford HAI, AI Index 2026: o emprego de desenvolvedores de software de 22 a 25 anos caiu cerca de 20% desde 2024, enquanto o de desenvolvedores com 30 anos ou mais seguiu crescendo. Pesquisa correlata de Stanford aponta queda de 16% no emprego de jovens de 22 a 25 anos em funções expostas à IA, com estabilidade entre os trabalhadores acima de 30 nas mesmas funções.

mesmo mercado — e resultados opostos, separados apenas pela idade e pela experiência.

Alguém resumiu isso numa imagem que eu não consigo tirar da cabeça: **estão puxando a escada, não derrubando o prédio.**

Pare e pense no que isso significa, porque muda a natureza do medo. Se você é um profissional estabelecido, com dez ou vinte anos de estrada, o risco imediato não é ser demitido e substituído por uma máquina amanhã de manhã. Os dados não mostram isso acontecendo com você. O que eles mostram é algo mais lento e mais traiçoeiro: os degraus de baixo estão sendo removidos. As tarefas simples, que antes eram feitas pelo estagiário, pelo júnior, pelo desenhista que estava aprendendo, estão sendo absorvidas. E uma profissão que para de formar gente por baixo é uma profissão que envelhece — e, com o tempo, a posição de quem está em cima também perde valor, porque perde a estrutura que a sustentava.

E se você é jovem, ou está começando agora, ou é aquele engenheiro brasileiro recém-formado do interlúdio anterior tentando entrar no mercado pela porta do desenho: essa é a luta da sua vida profissional, e ela começou. A porta pela qual a minha geração entrou, fazer o trabalho simples até aprender o difícil, está se fechando enquanto você bate nela.

Guarde esse quadro, porque ele já elimina metade das respostas fáceis. A pergunta certa não é "a IA vai me substituir?". É "o que sobra da minha profissão depois que a parte de baixo dela for automatizada, e eu estou desse lado ou do outro?".

A sua profissão, especificamente

Vamos ao seu caso concreto, o do projeto e do desenho técnico, porque generalidades sobre "o futuro do trabalho" não pagam boleto.

O dado mais eloquente que encontrei sobre o nosso campo é este: em um único ano, as exigências de desenho técnico tradicional caíram de 50% para 31% das vagas da área, enquanto as exigências de habilidades ligadas a inteligência artificial e automação saltaram de 7% para 21%.[51] Em doze

[51] Relatório setorial citado por Apollo Technical (2026): as exigências de desenho técnico tradicional caíram de 50% para 31% dos requisitos de vaga em um único ano, enquanto exigências de IA e aprendizado de máquina subiram de 7% para 21%. Novos cargos híbridos — coordenador BIM, engenheiro de automação CAD, especialista em CAD com IA — já aparecem nos anúncios.

meses. A vaga não desapareceu — ela foi *reescrita*. Continuam contratando; só que estão pedindo outra coisa.

E o total de postos? Praticamente estável. As projeções oficiais de emprego para desenhistas técnicos nos Estados Unidos indicam pouca ou nenhuma variação até 2034, com cerca de 16 mil vagas abrindo por ano — a maior parte por aposentadoria e mudança de carreira, não por demissão em massa.[52] Some as duas informações e o retrato fica nítido: **a profissão não está encolhendo, está trocando de conteúdo.** O que some não é o cargo, é o miolo antigo do cargo.

E o que exatamente está sendo automatizado? A lista é bem conhecida de quem acompanha: conversão de PDF para desenho editável, cotagem automática, inserção de blocos padronizados, anotações de rotina, geração de tabelas de material, conversão entre formatos.[53] Leia essa lista de novo, com atenção, porque ela tem uma característica que ninguém comenta: **são exatamente as tarefas que este livro vem te dizendo, desde o sexto capítulo, para você automatizar você mesmo.**

Essa coincidência não é acidente. É o coração do argumento. Essas tarefas vão desaparecer de qualquer maneira — a decisão não está com você. A única coisa que está na sua mão é se elas vão desaparecer **porque você as automatizou**, e o tempo liberado virou seu, ou **apesar de você**, e o tempo liberado virou de outra pessoa, junto com a sua função.

O que a máquina não assume (e por quanto tempo)

Existe um conjunto de coisas em que a inteligência artificial continua ruim, e vale nomear com precisão, porque é onde a sua defesa mora.

Segundo as análises de 2026, o trabalho mais protegido é aquele que exige **responsabilidade legal ou registro profissional**; o que exige **presença física e mão na máquina**; e (esta é a categoria mais importante e a menos citada) aquele em que **estar confiantemente errado custa caro demais**

[52]Bureau of Labor Statistics (EUA): projeção de pouca ou nenhuma variação no emprego total de desenhistas técnicos entre 2024 e 2034, com cerca de 16.200 vagas anuais, majoritariamente decorrentes de aposentadorias e mudanças de carreira.

[53]Tarefas de desenho técnico já automatizadas de forma rotineira em 2026: conversão de PDF para DWG, cotagem automática, inserção de blocos, anotação de rotina, geração de tabelas de materiais e conversão entre formatos (Apollo Technical; Digital Estimating, 2026). Estimativa do Fórum Econômico Mundial: até 50% das *tarefas* em engenharia e manufatura poderiam ser automatizadas até 2030 — tarefas, não cargos.

para não ter um humano assinando embaixo.[54]

Leia essa terceira categoria como projetista de moldes, e veja o tamanho da sorte que temos.

Um molde é aço, é dinheiro de verdade, é prazo de cliente. Um erro de ângulo de saída não gera uma resposta esquisita numa tela: gera uma peça que não desmolda, um lote perdido, uma ferramenta que volta para a bancada. Uma inteligência artificial pode gerar uma geometria plausível em segundos — e plausível é exatamente a palavra perigosa, porque a máquina erra com a mesma segurança com que acerta. Alguém precisa olhar aquilo, desconfiar, verificar e assinar. E quem assina precisa saber por quê.

E há o que não está em dataset nenhum. A máquina não caminha até a injetora para ver o comportamento real do material. Não sabe que *aquela* máquina específica tem uma mania. Não discute com o ferramenteiro que vai usinar a peça e sabe, pela experiência das mãos, que aquele canto vivo vai dar problema. Não sente o cheiro de queimado no galpão. Esse conhecimento tácito, construído no chão de fábrica ao longo de anos, é a coisa mais difícil de digitalizar que existe, porque grande parte dele nunca foi escrita em lugar nenhum.

Mas agora a honestidade, porque um livro que se vende como honesto não pode terminar uma seção com um cafuné. **Nada disso é um escudo permanente.** Cada uma dessas fronteiras já foi ultrapassada em outros campos, e algumas serão ultrapassadas aqui. Construir a sua estratégia de carreira sobre "a IA não consegue fazer X" é construir sobre areia, porque a lista de X encolhe todo ano. A estratégia sólida é outra, e é a do livro inteiro: não aposte no que a máquina não faz; aposte na sua capacidade de continuar mudando quando ela passar a fazer.

A resposta verdadeira

Junte tudo e a resposta aparece — e ela não é sobre você contra a máquina.

Jensen Huang, da NVIDIA, formulou isso de um jeito que virou lugar-comum justamente porque é verdadeiro: ninguém perde o emprego para uma inteligência artificial; perde para outra pessoa que usa uma. E o mercado já está precificando isso abertamente. O prêmio salarial para profissionais

[54] Síntese das análises de 2026 sobre trabalho menos exposto: atividades que exigem responsabilidade legal ou registro profissional (medicina, direito, auditoria, engenharia com responsabilidade técnica), trabalho físico especializado, e funções em que o custo de um erro confiante é alto o bastante para exigir que um humano responda pelo resultado.

com domínio real de ferramentas de IA está entre 20% e 40% em diversos setores,[55] e, entre profissionais independentes, chega a 56% acima dos que não têm essa competência — número ao qual voltarei, com outro sentido, quando este livro chegar à quarta parte.

Esse é o ponto. A linha divisória de 2026 não passa entre humanos e máquinas. Passa entre **dois profissionais humanos**: o que incorporou a ferramenta e o que não incorporou. O primeiro entrega em um dia o que o segundo entrega em uma semana, e cobra mais caro por isso. Nenhuma máquina precisa te demitir para você perder espaço; basta um colega com a mesma formação que a sua e uma automação que você não tem.

É por isso que o maior risco de carreira em 2026 não é a inteligência artificial. É a **inércia**.

O plano

Chega de diagnóstico. Se você chegou até aqui com medo, é justo que saia daqui com um plano. Cinco passos, na ordem, e todos ao seu alcance.

Primeiro: separe o seu trabalho em duas pilhas. Pegue tudo o que você faz numa semana típica e divida em duas colunas. De um lado, o que segue regra — cotar, converter, preencher, listar, padronizar, formatar. Do outro, o que exige julgamento, contexto, negociação, responsabilidade, assinatura. A primeira coluna vai ser automatizada, com você ou sem você. A segunda é onde a sua carreira vai morar. Se a primeira coluna ocupa mais de metade do seu tempo, você acabou de descobrir a urgência do seu caso.

Segundo: automatize você mesmo a primeira coluna. Não espere a empresa fazer, não espere a próxima versão do software. Comece pela tarefa mais irritante, como propus no sexto capítulo, e use a inteligência artificial como ponte para escrever a automação. Há uma assimetria enorme e pouco percebida aqui: **quem automatiza a própria função raramente perde o emprego — muda de função.** Passa a ser quem constrói as ferramentas do time, e essa pessoa é a última a sair de qualquer lugar.

Terceiro: invista o tempo liberado na segunda coluna, e não em fazer mais do mesmo. Esta é a armadilha em que mais gente cai. Você automatiza, ganha três horas por semana — e usa as três horas para fazer mais desenhos.

[55] Prêmio salarial de 20% a 40% para profissionais com proficiência em IA, em diversos setores (levantamentos de 2026). Entre profissionais independentes, o prêmio para competências especializadas em IA chega a 56%; o dado é retomado no Capítulo 12, no contexto do trabalho independente.

Parabéns: você aumentou a produção da coluna que está morrendo. O tempo liberado precisa ir para julgamento, especialização, relacionamento com cliente, domínio de um problema difícil. É a Tranca 6 do diagnóstico, e é a única que a automação não abre por você.

Quarto: torne-se quem verifica. Lembra do dado do sexto capítulo, de que revisar o código gerado por IA já superou escrevê-lo como o maior consumo de tempo? Isso vale para tudo. À medida que a máquina produz mais, o gargalo deixa de ser a produção e passa a ser a **conferência confiável** — alguém que saiba olhar para um resultado plausível e dizer, com fundamento, que está errado. Essa competência é rara, é cara, e cresce exatamente na mesma velocidade em que a IA melhora. É o único cargo que fica mais valioso quanto mais a máquina evolui.

Quinto: não dependa de uma porta só. Todo o plano acima ainda te deixa dentro de uma estrutura que pode mudar sem te consultar. A segurança real, como o interlúdio brasileiro mostrou com números, não vem de ser insubstituível num emprego — vem de não depender de um só. É a Tranca 5, e é a que a maioria adia por mais tempo.

O medo certo

Termino trocando o seu medo por outro, mais útil.

“A IA vai me substituir?” é uma pergunta passiva. Ela te coloca como objeto de uma decisão que outra pessoa toma, e a única coisa que você pode fazer com ela é esperar e ter medo. Não há plano possível a partir dela.

A pergunta que serve é outra: **“eu vou continuar exatamente o mesmo enquanto tudo à minha volta muda?”** Essa é ativa, é sua, e você pode responder ainda esta semana.

Não posso te prometer segurança. Ninguém pode, e quem promete está vendendo. Alguns profissionais desta área vão, sim, ficar para trás — e vão ser, majoritariamente, os que ficaram parados esperando a poeira baixar, na esperança de que o mundo voltasse a ser o de antes. O que eu posso te dizer, com os dados na mão, é que a variável que mais determina o resultado não está nas mãos da NVIDIA, nem da Dassault, nem do governo. Está na sua. E poucas vezes na história de uma profissão isso foi tão literalmente verdade quanto agora.

O que vem a seguir

Aqui se encerra, de fato, a segunda parte deste livro. Diagnosticamos a gaiola, encontramos a saída, medimos o seu preço em segurança e responsabilidade, e agora encaramos de frente o medo que tudo isso desperta.

E repare onde a resposta nos deixou: num terreno que não é técnico. Falei de automação e de dados de emprego, mas terminei em julgamento, responsabilidade, verificação, assinatura — em quem responde pelo resultado quando a máquina erra com confiança. Não há como escapar dessa virada. Quando você persegue a questão da inteligência artificial até o fim, ela deixa de ser uma pergunta sobre máquinas e vira uma pergunta sobre o que, exatamente, um ser humano traz para o trabalho que a máquina não traz.

É essa a pergunta que a terceira parte enfrenta. Deixei a ferramenta de lado. É hora de falar do engenheiro. E do homem.

Para levar deste capítulo

Trancas 3 e 6 — Capacidade e Domínio. São as duas que decidem de que lado da linha você fica: quem automatiza a própria função muda de função; quem domina um problema difícil não é substituído por quem gera resultados plausíveis.

Faça agora — 20 minutos. Divida em duas colunas tudo o que você fez na semana passada: de um lado o que segue regra, de outro o que exige julgamento e assinatura. Calcule a proporção. Se a coluna das regras passa de metade do seu tempo, você tem a resposta sobre a sua urgência — e tem, também, a lista exata do que automatizar primeiro.

Fontes: Goldman Sachs (abr/2026); Harvard Business School / Harvard Business Review (mar/2026); Stanford HAI AI Index 2026; Bureau of Labor Statistics (EUA); Apollo Technical, Digital Estimating, Research.com e BIM Heroes (2026), sobre o impacto da IA em funções de CAD e desenho técnico; compilações de estatísticas de deslocamento de DesignRush, Axis Intelligence e HiWave (2026); Fórum Econômico Mundial, Future of Jobs. As faixas de deslocamento e prêmio salarial são estimativas de mercado e devem ser lidas como indicativas.

Parte III

A Filosofia: Liberdade Individual Aplicada à Tecnologia

Capítulo 9 — Liberdade Não é Ausência de Regras, é Presença de Responsabilidade

Há um mal-entendido sobre a liberdade tão comum, e tão antigo, que ele consegue estragar a palavra antes mesmo que a gente termine de dizê-la. É a ideia de que ser livre significa não ter regras. Fazer o que se quer, quando se quer, sem limites, sem contas a prestar. Liberdade como ausência — ausência de amarras, de deveres, de fronteiras.

Se fosse isso, este seria um livro sobre irresponsabilidade, e eu não teria passado um capítulo inteiro, o anterior, explicando por que o engenheiro livre precisa ser o guardião obsessivo da própria segurança. A verdade é o oposto do mal-entendido, e é a espinha de toda a terceira parte deste livro: **liberdade não é a ausência de regras. É a presença de responsabilidade.** Não se mede a liberdade de alguém pela quantidade de limites que ele derrubou, mas pela quantidade de escolhas pelas quais ele é capaz de responder.

Fechei a segunda parte dizendo que ia deixar as ferramentas de lado para falar do engenheiro, e do homem. Cumpro a promessa agora. Porque as decisões de software que discuti até aqui — qual formato adotar, qual contrato assinar, qual inteligência alugar ou possuir — nunca foram, no fundo, decisões técnicas. Eram decisões morais disfarçadas de técnicas. E por trás de todas elas há um único princípio, que este capítulo existe para tornar explícito.

A estrada sem regras não é livre

Imagine um cruzamento movimentado onde, de repente, todas as regras foram abolidas em nome da liberdade. Sem semáforo, sem preferência, sem mão e contramão. Cada motorista, finalmente “livre”, faz o que bem entende. O resultado não é liberdade — é o engarrafamento total, a batida inevitável, a lei do mais audacioso ou do mais pesado. Ninguém chega a lugar nenhum. A ausência completa de regras não produziu movimento

livre; produziu paralisia e violência.

Agora imagine o mesmo cruzamento com regras claras, conhecidas, iguais para todos. De repente, milhares de pessoas atravessam a cidade indo cada uma para onde quer, sem colidir. As regras, longe de matar a liberdade de ir e vir, são exatamente o que a torna possível. Mas — e aqui está o ponto que separa a boa filosofia da má — não é qualquer regra que liberta. A regra que liberta é a que se aplica igualmente a todos e que existe para permitir que cada um siga o seu próprio caminho. A regra que aprisiona é a que decide, por você, qual caminho você pode seguir.

A tradição do pensamento liberal sempre insistiu nessa distinção, e ela é velha de séculos: liberdade não é licença.[56] O homem livre não é o que não tem lei nenhuma; é o que vive sob leis que se aplicam a todos por igual e que protegem, em vez de dirigir, as suas escolhas. Confundir liberdade com ausência de qualquer regra é o erro que entrega, no fim, todas as pessoas ao poder de quem grita mais alto — e é, não por acaso, o erro que os vendedores de gaiolas adoram, porque ele faz a gaiola, com suas regras claras e seu conforto ordenado, parecer preferível ao caos.

As duas regras

Se nem toda regra aprisiona e nem toda ausência de regra liberta, então o eixo que importa não é *quantas* regras existem, mas *de onde elas vêm* e *a quem elas servem*.

Há regras que emergem de baixo, da cooperação voluntária entre pessoas que buscam, cada uma, os seus próprios fins — como a linguagem, os costumes, os padrões técnicos abertos, os protocolos que descrevi na segunda parte. Ninguém as decretou; elas se formaram porque serviam a quem as adotava, e qualquer um é livre para adotá-las ou não. E há regras que descem de cima, impostas por uma autoridade que decidiu, no lugar dos indivíduos, qual é o caminho certo — seja um Estado que decreta, seja uma plataforma que define de antemão o seu fluxo de trabalho e chama isso de recurso.

A primeira espécie de regra é o solo da liberdade. A segunda é o solo da dependência. E a forma mais pura, mais concreta, mais cotidiana da regra que liberta é aquela que dois indivíduos criam entre si quando fecham

[56] A distinção entre liberdade e licença é um dos pilares do liberalismo clássico. John Locke, no *Segundo Tratado sobre o Governo* (1689), afirma que o estado de natureza, embora seja um estado de liberdade, não é um estado de licença: o homem é livre, mas não para se destruir nem para invadir os direitos alheios. A liberdade, nessa tradição, é sempre liberdade sob leis iguais, não ausência de qualquer lei.

um acordo voluntário: um contrato. Um contrato livremente pactuado, com termos claros que ambas as partes entenderam e aceitaram, é a responsabilidade e a liberdade se dando as mãos. Cada parte se compromete, cada parte responde pelo que prometeu, e nenhuma das duas está subjugada à outra — estão as duas subjugadas apenas à palavra que deram. É por isso que dediquei, lá no segundo capítulo, tanta atenção à diferença entre a promessa de palco e a letra do contrato. O respeito ao que se assinou não é um detalhe jurídico chato. É a estrutura de aço sobre a qual pessoas livres constroem relações sem precisar mandar umas nas outras.

Por que “de graça” nunca é de graça

Munidos dessa distinção, podemos finalmente entender o truque que percorre este livro do começo ao fim: o jardim murado que se oferece de graça.

Quando uma plataforma fechada te dá algo sem cobrar — um recurso gratuito, uma ferramenta de IA incluída, um formato que “simplesmente funciona” —, a tentação é pensar que você recebeu liberdade de presente. Menos trabalho, menos custo, menos preocupação. Mas repare no que aconteceu de verdade: você não eliminou a responsabilidade por aquela escolha. Você a transferiu. Alguém, em algum lugar, agora decide por você como aquela ferramenta funciona, quando ela muda, quanto ela vai custar quando você não puder mais viver sem ela, e se ela vai continuar existindo. O que parecia um alívio — “não preciso me preocupar com isso” — era, na verdade, a terceirização silenciosa de um pedaço da sua autonomia.

Essa é a razão pela qual “de graça”, dentro do jardim murado, nunca é de graça. O preço não está no boleto; está na responsabilidade que você abriu mão de exercer, e que, como toda responsabilidade que se abandona, não desaparece, apenas muda de dono. E quem assume a responsabilidade pela sua escolha assume, junto, o poder sobre ela. Não existe almoço grátis, dizem os economistas; eu diria mais: não existe responsabilidade grátis. Toda vez que alguém se oferece para carregar a sua, cobre bem os bolsos, porque a conta virá, e ela será cobrada na única moeda que importa de verdade — a da sua liberdade.

A disciplina do homem livre

Há um paradoxo aqui que vale a pena encarar de frente, porque ele desfaz de vez a confusão entre liberdade e ausência de regras. O profissional mais livre que conheço não é o mais desregrado. É, quase sempre, o mais disciplinado.

Penso no meu próprio ofício. Um projeto de molde sério não se faz na base da improvisação livre; faz-se com listas de verificação minuciosas, com procedimentos, com revisões, com rastreabilidade de cada decisão — o tipo de rigor que, visto de fora, pareceria o oposto da liberdade criativa. E no entanto é exatamente esse rigor autoimposto que me torna livre para assumir projetos que, sem ele, eu não teria coragem nem competência de aceitar. As regras que eu imponho a mim mesmo não me diminuem; me habilitam. Elas são minhas. Eu as escolhi, eu as entendo, eu posso mudá-las quando aprendo algo melhor. A disciplina do guardião da própria segurança, que descrevi no capítulo anterior, é da mesma natureza: privilégio mínimo, aprovação humana, procedência das ferramentas. São regras — muitas, rigorosas —, e são precisamente o que permite ao engenheiro andar pela estrada aberta sem ser assaltado.

Aqui está, portanto, a diferença que dá o título a este capítulo. A regra imposta de cima, que você não escolheu e não pode mudar, é uma coleira. A regra que você impõe a si mesmo, porque entende que ela serve ao seu domínio do ofício, é uma ferramenta — talvez a mais poderosa de todas. O homem livre não foge das regras. Ele as escolhe, as cumpre, e responde por elas. É essa a diferença entre ser dirigido e se autogovernar, e ela é, palavra por palavra, a diferença entre a servidão e a liberdade.[57]

Responsabilidade pressupõe escolha

Chego, assim, ao fundamento último — aquele sobre o qual tudo o que eu disse repousa, e que aponta para o capítulo mais pessoal que ainda está por vir.

Só é possível ser responsável por aquilo que se escolheu livremente. Ninguém responde, moralmente, por um ato que foi forçado a cometer sob coação, nem por um resultado que uma máquina produziu sem que ele pudesse interferir. A responsabilidade pressupõe a escolha; a escolha pressupõe a liberdade; e as três coisas — liberdade, escolha, responsabilidade — são, no fim, uma só

[57] A tese de que a liberdade se realiza no autogoverno responsável, e não na ausência de constrangimentos, é desenvolvida por Friedrich Hayek no capítulo “Responsibility and Freedom” de *The Constitution of Liberty* (1960), onde argumenta que a liberdade e a responsabilidade são inseparáveis: uma sociedade livre exige e pressupõe indivíduos dispostos a arcar com as consequências de seus atos. A imagem popular que melhor traduz isso talvez seja a do psiquiatra Viktor Frankl, sobrevivente dos campos de concentração, que em *Em Busca de Sentido* (1946) propôs que a Estátua da Liberdade, na costa leste dos Estados Unidos, fosse complementada por uma Estátua da Responsabilidade na costa oeste — porque a liberdade, sem a responsabilidade que a equilibra, degenera em mero arbítrio.

realidade vista de três ângulos. É por isso que atacar a liberdade das pessoas, mesmo em nome de protegê-las, é sempre também atacar a sua dignidade: porque tirar de alguém a capacidade de escolher é tirar dele a possibilidade de ser responsável, e um ser que não pode ser responsável por nada foi rebaixado, por mais confortável que seja a sua gaiola, à condição de criança perpétua ou de peça de engrenagem.

Essa intuição tem raízes que vão muito além da economia ou da engenharia. Ela toca na pergunta mais profunda que se pode fazer sobre o ser humano — a de se somos, de fato, agentes livres, capazes de escolher e portanto de responder pelo que escolhemos, ou apenas mecanismos executando o que nos foi determinado. A minha convicção sobre essa pergunta não é neutra, e não vou fingir que é. Mas ela merece um capítulo só dela, onde eu possa ser honesto sobre a fé de onde ela vem. Por ora, basta fixar o elo: sem livre-arbítrio, não há responsabilidade; sem responsabilidade, a palavra liberdade fica oca. Quem leva a liberdade a sério é obrigado, mais cedo ou mais tarde, a levar a sério também a ideia de que o ser humano é genuinamente livre para escolher — e portanto genuinamente responsável pelo que faz de si.

O que vem a seguir

Se cada pessoa é livre e, por isso, responsável pelas próprias escolhas, uma pergunta prática e enorme se abre: como é que milhões de escolhas individuais, cada uma feita por alguém que responde só por si, acabam sendo julgadas? Quem decide se a decisão que tomei foi boa ou ruim, acertada ou tola? Se não há uma autoridade central sábia o bastante para avaliar cada uma delas — e eu argumentei, desde o quarto capítulo, que não há e não pode haver —, então o que faz esse julgamento?

A resposta é o assunto do próximo capítulo, e ela tem um nome que assusta alguns e conforta outros: o mercado. Não o mercado como abstração fria dos manuais, mas como o veredito descentralizado de milhões de pessoas escolhendo livremente com quem cooperar, o que adotar, do que abrir mão. E há uma prova viva desse veredito acontecendo agora mesmo, no mundo que eu conheço de perto: a aposta que a Siemens fez na abertura curada, contra o jardim murado da concorrente — e o que o mercado, esse juiz que não pede licença a ninguém, está decidindo a respeito.

Nota: este capítulo é deliberadamente filosófico e conceitual; suas fontes são as obras clássicas citadas (Locke, Hayek, Frankl), e não dados de

atualidade. Os exemplos concretos que o ilustram — contratos, formatos, lock-in, segurança de agentes — foram documentados e referenciados nos capítulos anteriores.

Para levar deste capítulo

Todas as trancas. Este capítulo não abre nenhuma em particular: ele explica por que abrir cada uma delas é responsabilidade sua, e por que a regra que você escolhe habilita, enquanto a que te impõem prende.

Faça agora — 10 minutos. Escolha uma regra que você mesmo se impõe no trabalho — um procedimento, uma checagem, um padrão — e escreva numa linha por que ela te torna capaz de fazer coisas que você não faria sem ela. Depois faça o mesmo com uma regra que te impuseram. Compare as duas frases.

Capítulo 10 — O Mercado como Juiz

Terminei o capítulo anterior com uma pergunta que não é retórica: se cada pessoa é livre e responsável pelas próprias escolhas, quem julga se essas escolhas foram boas? Se não existe — e argumentei desde o quarto capítulo que não existe nem pode existir — uma autoridade central sábia o bastante para avaliar cada decisão de cada indivíduo, então o que faz esse julgamento?

A resposta tem um nome que desperta reações extremas: o mercado. Para uns, é a solução para quase tudo; para outros, é o culpado por quase tudo. Quero fugir das duas caricaturas, porque ambas erram o alvo. Neste capítulo vou tratar o mercado como o que ele de fato é, um juiz, e vou testar o veredito dele num caso concreto que conheço de dentro, o da guerra silenciosa entre a abertura e o muro no mundo do software de engenharia.

E já começo com uma confissão de honestidade, porque a devo ao leitor: eu não sou um observador neutro dessa disputa. Trabalho, todos os dias, dentro de um dos ecossistemas que vou analisar — sou usuário de Solid Edge, da Siemens. Não vou fingir uma imparcialidade que não tenho. Vou fazer o que é mais honesto: mostrar os números, inclusive os que complicam a minha própria tese, e deixar você avaliar o veredito por conta própria. Afinal, é exatamente disso que trata este capítulo.

O que é, de verdade, o “mercado”

Antes do caso concreto, preciso desfazer uma confusão que envenena quase toda conversa sobre o assunto. O mercado não é um lugar, não é um grupo de pessoas ricas reunidas numa sala, não é uma entidade com vontade própria conspirando contra alguém. O mercado é, simplesmente, o nome que damos ao resultado agregado de milhões de escolhas voluntárias — cada pessoa decidindo, com o conhecimento que só ela tem da própria situação, com quem cooperar, o que comprar, o que adotar, do que abrir mão.

Visto assim, o mercado é um juiz de um tipo muito particular: um juiz que ninguém nomeou e que ninguém controla. E é precisamente isso que torna

o seu veredito confiável de um jeito que o de nenhuma autoridade central poderia ser. Um planejador central, por mais bem-intencionado e inteligente que fosse, julgaria as escolhas dos engenheiros a partir do pouco que ele, de sua mesa distante, consegue enxergar. O mercado julga a partir de tudo o que milhões de engenheiros sabem, somado — o conhecimento disperso que discuti no quarto capítulo, agora funcionando como tribunal. Ele não decreta o que é bom; ele descobre o que as pessoas valorizam, deixando que elas escolham e observando o que acontece.[58]

É por isso que o mercado é menos um oráculo que dita a verdade e mais um processo de descoberta que a revela aos poucos, escolha após escolha. E é a esse processo que vou agora submeter a pergunta central deste livro: a abertura vale a pena, ou é ingenuidade romântica de quem não entende de negócios?

Duas apostas, dois caminhos

No mundo do software industrial, dois gigantes fizeram, na última década, apostas filosoficamente opostas sobre essa mesma pergunta — e o mercado vem, desde então, julgando as duas.

De um lado, a Dassault Systèmes, com a sua plataforma 3DEXPERIENCE, apostou no jardim integrado: tudo sob um mesmo teto, numa única plataforma coesa, com integração profunda entre as partes, entregue por assinatura, com o cliente cada vez mais dentro de um ambiente que a empresa controla de ponta a ponta. É a aposta na força do conjunto fechado — a promessa de que, se tudo vem da mesma casa e conversa perfeitamente entre si, o cliente terá uma experiência superior, ainda que ao custo de ficar preso àquela casa.

Do outro lado, a Siemens, com a plataforma Siemens Xcelerator, apostou na abertura curada. Em vez de um jardim murado, um mercado aberto: uma plataforma que reúne, além das soluções da própria Siemens, as ofertas de mais de setecentos parceiros certificados, construída sobre princípios

[58] A concepção do mercado como um “processo de descoberta” — e não como um mecanismo que pressupõe conhecer de antemão a resposta certa — é de Friedrich Hayek, no ensaio “Competition as a Discovery Procedure” (“A Concorrência como Procedimento de Descoberta”, 1968). A ideia complementa a do conhecimento disperso (“The Use of Knowledge in Society”, 1945, citado no Capítulo 4): a concorrência serve justamente para revelar informações — sobre o que funciona, o que as pessoas querem, quanto custa — que de outro modo permaneceriam desconhecidas.

explícitos de interoperabilidade, flexibilidade e abertura.[59] Mas — e este "mas" é o que separa a estratégia da ingenuidade — não é uma abertura anárquica. É *curada*: cada solução que entra no mercado passa pelo crivo técnico e comercial da Siemens, é verificada quanto a interoperabilidade, cibersegurança e qualidade, seguindo normas reconhecidas.[60] A Siemens não abriu mão do controle da qualidade; abriu mão do controle da *propriedade*. Ela deixou de tentar ser a dona de tudo para se tornar a curadora de um ecossistema — mantendo a porta aberta para o Teamcenter conversar com CADs que não são os seus, para parceiros construírem sobre a plataforma, para o cliente montar a sua própria combinação.

Duas empresas sérias, sofisticadas, lucrativas. Duas visões incompatíveis sobre a mesma questão. E, entre elas, o juiz.

O veredito parcial do juiz

Aqui é onde eu poderia trair o leitor, escolhendo só os números que me convêm. Não vou fazer isso. Vou mostrar os dois lados, porque o veredito real é mais interessante — e mais persuasivo — do que qualquer torcida.

Comecemos pela minha própria tese. A aposta da Siemens na abertura curada não é caridade nem idealismo: é um negócio que está crescendo depressa. No primeiro semestre do ano fiscal de 2026, o negócio digital da Siemens cresceu 19% (bem acima da própria meta de 15% que a empresa havia estabelecido meses antes), puxado justamente pela expansão das ofertas de software e serviços do Xcelerator.[61] O braço de software da divisão de Indústrias Digitais cresceu 14% num único trimestre. O ecossistema de setecentos parceiros atrai nomes de peso, da NVIDIA à Microsoft, da AWS a fornecedores especializados de manufatura sob demanda que passaram a embutir a própria inteligência dentro da plataforma

[59] Dados do Siemens Xcelerator: plataforma digital aberta lançada em meados de 2022, hoje com mais de 700 parceiros/vendedores certificados, construída sobre princípios de interoperabilidade, flexibilidade, abertura e modelo as-a-service (siemens.com, páginas institucionais do Xcelerator, 2026).

[60] A "curadoria" da abertura: todas as ofertas do marketplace passam por governança técnica e comercial da Siemens, verificadas quanto a interoperabilidade, cibersegurança (normas ISO 27001 e IEC 62443) e qualidade (siemens.com, 2026). É o ponto central do capítulo: abertura não é ausência de padrões, é padrões abertos com curadoria de qualidade.

[61] Resultados do 2º trimestre do ano fiscal de 2026 da Siemens: no primeiro semestre fiscal, o negócio digital cresceu 19%, acima da meta de 15% definida meses antes, impulsionado pela expansão das ofertas de software e serviços do Xcelerator; o braço de software das Indústrias Digitais cresceu 14% no trimestre, atingindo €1,6 bilhão (Siemens, comunicado "Siemens continues path of profitable growth", maio de 2026).

aberta.[62] Em levantamentos de satisfação de usuários em alguns segmentos, as soluções da Siemens aparecem à frente. A abertura, ao que parece, vende.

Mas agora o outro lado, que a honestidade me obriga a mostrar. A Dassault não está perdendo. Longe disso: ela segue como uma das líderes absolutas do mercado de PLM, com margens operacionais na casa dos 30% — sinal de um poder de precificação que só quem tem clientes fiéis (ou presos) consegue sustentar —, e com a sua receita recorrente crescendo em ritmo igualmente forte.[63] O jardim murado, é preciso admitir, também dá lucro. E muito.

Então, qual é o veredito? Não é um nocaute. Ninguém está morrendo. E é exatamente aí que mora a lição, porque durante anos disseram que a abertura era comercialmente suicida — que abrir mão do controle, deixar o cliente livre para sair, conversar com os concorrentes, montar a própria combinação, era entregar o próprio negócio de bandeja. O veredito do mercado desmente isso de forma clara: **dá para abrir mão do controle e vencer assim mesmo.** A abertura curada não é a estratégia dos ingênuos; é uma aposta calculada que está entre as que mais crescem no setor. O engenheiro que aposta na liberdade não está apostando contra o mercado — o mercado está, ele próprio, recompensando a liberdade.

Até o muro teve que abrir uma porta

Há, porém, uma evidência ainda mais eloquente do que os números de crescimento — e ela está escondida no comportamento da própria campeã do jardim murado.

Lembra da parceria entre a Dassault e a NVIDIA que descrevi no primeiro capítulo, com os modelos abertos de inteligência artificial rodando dentro da arquitetura fechada da 3DEXPERIENCE? Olhe para ela de novo, agora com os olhos deste capítulo. Ela é a confissão involuntária de que nem a maior defensora do muro consegue mais ser puramente fechada. A pressão do mercado por interoperabilidade — por padrões que conversam, por peças que se encaixam, por inteligência que não fica trancada — ficou tão

[62] Exemplo de expansão do ecossistema: em 2026, a Xometry anunciou parceria estratégica para embutir sua inteligência de manufaturabilidade, precificação e sourcing diretamente no Siemens Xcelerator (Xometry, Form 8-K, Q1 2026). Parceiros estratégicos do ecossistema incluem Accenture, Atos, AWS, Bentley, Microsoft, SAP e NVIDIA.

[63] Posição da Dassault Systèmes: uma das líderes do mercado global de PLM & Engenharia (cerca de 16,5% de participação em 2024), com receita recorrente acima de 70% do total e ARR de software e nuvem crescendo em torno de 18% ao ano (2025), e margens operacionais próximas de 30% (análises de mercado de 2025-2026, matrixbcg.com e portersfiveforce.com). O jardim murado, é preciso reconhecer, é altamente lucrativo.

forte que até quem construiu a sua fortuna sobre a integração proprietária precisou abrir portas na parede. Os formatos neutros de intercâmbio, que discuti no segundo capítulo, sobrevivem e prosperam pela mesma razão: porque o cliente, esse juiz distribuído em milhões de decisões, castiga cada vez mais o aprisionamento e premia cada vez mais a compatibilidade.

Esse é o veredito escrito nas entrelinhas, e ele é mais forte que qualquer número de crescimento: quando até o muro é obrigado a abrir uma porta, você sabe para que lado o juiz está inclinando. Não porque alguém decretou, não porque um regulador mandou, mas porque milhões de escolhas individuais, somadas, empurraram o mercado inteiro naquela direção. É a ordem espontânea do quarto capítulo funcionando como pressão comercial: ninguém no comando, e mesmo assim um rumo emerge.

O juiz não é um oráculo

Preciso agora frear a minha própria retórica, porque um livro honesto sobre liberdade não pode transformar o mercado num ídolo — isso seria trocar uma superstição por outra.

O mercado é o melhor juiz disponível, mas não é um oráculo infalível, e muito menos um oráculo moral. Ele julga o que as pessoas *valorizam*, não o que é *verdadeiro* ou *bom* em algum sentido absoluto. Ele pode se enganar, ser míope, seguir modas. Ele premiou o aprisionamento durante décadas, afinal, antes de começar a virar contra ele; foram os próprios formatos proprietários, um dia, o que o mercado recompensava. Quem defende a liberdade de mercado precisa ter a honestidade de reconhecer isso, sob pena de estar apenas trocando o planejador central por um deus de mármore igualmente cego.

A defesa do mercado como juiz, portanto, nunca foi a de que ele acerta sempre. É uma defesa mais modesta e mais sólida: a de que um juiz descentralizado e voluntário é melhor e mais seguro que um juiz central e obrigatório — não porque erre menos, mas porque os seus erros são *corrigíveis*. Quando o mercado erra, a concorrência, a alternativa e o direito de sair vão, com o tempo, corrigindo o erro, do jeito que estão corrigindo agora a era do aprisionamento. Quando um planejador central erra, o erro vira lei, e todos são obrigados a segui-lo até que alguém, lá em cima, mude de ideia. A superioridade do mercado não está numa sabedoria mágica. Está na sua capacidade de se corrigir sem pedir permissão a ninguém — que é,

no fundo, a mesma virtude da liberdade que este livro inteiro defende.[64]

Por que isto importa para você

Termino trazendo tudo isto de volta para a sua mesa, porque a filosofia deste capítulo tem uma consequência prática que muda a forma de encarar as próprias decisões.

Se o mercado é o juiz, então as suas escolhas são votos. Cada vez que você prefere um formato aberto a um proprietário, uma ferramenta portátil a uma que te prende, um padrão que conversa a um que isola, você não está apenas se protegendo — está depositando um voto que o mercado conta. Sozinho, o seu voto é minúsculo, quase nada. Somado ao de milhões de outros profissionais fazendo a mesma escolha pelas mesmas razões, ele é a força que obrigou o muro a abrir uma porta. A responsabilidade individual do capítulo anterior não termina em você; ela se agrega, através do mercado, num rumo coletivo que ninguém comanda mas todos constroem.

É esta a beleza discreta da ordem espontânea aplicada à sua carreira: você não precisa esperar que um regulador imponha a abertura de cima, nem que uma revolução derrube os muros. Você só precisa escolher a liberdade, na sua escala, todos os dias — e confiar que milhões de outros, escolhendo em separado o mesmo, formam uma maré que nenhuma muralha segura para sempre. O engenheiro que escolhe a liberdade não está só se libertando. Está ajudando, com o próprio voto silencioso, a fazer a liberdade vencer.

O que vem a seguir

Este capítulo mostrou o mercado como o juiz imparcial das nossas escolhas livres — um juiz melhor que qualquer autoridade central, ainda que falível, ainda que cego para certas coisas. E é justamente sobre uma dessas coisas que ele é cego que preciso falar agora, para fechar a terceira parte do livro.

Porque há uma dimensão que o mercado não consegue julgar, uma pergunta que ele não sabe responder. O mercado julga o valor — quanto as pessoas estão dispostas a pagar, a adotar, a trocar. Mas ele não sabe nada sobre o *sentido*: por que o trabalho importa, para que serve um ser humano, o que

[64] A tese de que a virtude do mercado não é a infalibilidade, mas a capacidade de correção descentralizada dos próprios erros (em contraste com a rigidez do erro centralmente imposto) é corolário do argumento de Hayek sobre concorrência e conhecimento, e permeia a tradição liberal de Adam Smith a Milton Friedman. Reconhecer a falibilidade do mercado é condição para defendê-lo com honestidade, e não como um novo dogma.

dá dignidade ao que fazemos com as nossas mãos e a nossa mente. Essa é uma pergunta que nenhum juiz de mercado pode responder, porque ela não é uma questão de preço — é uma questão de propósito. E para enfrentá-la, no próximo capítulo, vou precisar deixar de lado, por um momento, tanto o engenheiro quanto o libertário, e falar com você como aquilo que, no fim de tudo, também sou: um cristão. É hora de falar de fé, de trabalho, de livre-arbítrio — e do legado que escolhemos carregar.

Fontes: Siemens, comunicados e páginas institucionais do Siemens Xcelerator e resultados fiscais de 2026 (siemens.com; press.siemens.com); Xometry, Form 8-K Q1 2026 (sec.gov); análises do cenário competitivo de Dassault Systèmes e do mercado de PLM/simulação (matrixbcg.com, portersfiveforce.com, mordorintelligence.com, robot-magazine.fr, 2025-2026); Gartner Peer Insights (comparativos de usuários, 2026). Referências conceituais: F. A. Hayek, "Competition as a Discovery Procedure" (1968) e "The Use of Knowledge in Society" (1945).

Para levar deste capítulo

Tranca 2 — Ferramentas. Cada escolha de ferramenta é um voto que o mercado conta; a sua sozinha é minúscula, somada às outras é o que já obrigou o muro a abrir uma porta.

Faça agora — 5 minutos. Na sua próxima decisão de compra ou renovação de ferramenta, escreva antes, em uma frase, qual voto você está dando. Só isso. O hábito de tornar a escolha consciente muda mais decisões do que qualquer argumento.

Duas apostas, o mesmo ano

Fechar mais o jardim, ou abrir com curadoria — e o que o mercado está decidindo.

JARDIM INTEGRADO

Dassault · 3DEXPERIENCE

~16,5%	do mercado global de PLM (2024)
~30%	de margem operacional
>70%	da receita é recorrente
+18%/ano	ARR de software e nuvem (2025)

ABERTURA CURADA

Siemens · Xcelerator

+19%	negócio digital no 1º sem. FY2026*
+14%	software num único trimestre (€1,6 bi)
700+	parceiros certificados no ecossistema
NVIDIA, AWS,	Microsoft, SAP… todos dentro

**meta era 15%*

O veredito do mercado: dá para abrir mão do controle — e crescer mesmo assim.

FIGURA D11

Capítulo 11 — Fé, Trabalho e Livre-Arbítrio na Era da IA

Prometi, no fim do capítulo anterior, que aqui eu deixaria de lado por um momento o engenheiro e o libertário para falar como aquilo que também sou: um cristão. Cumpro a promessa, e faço um pedido ao leitor que não compartilha da minha fé — a maioria, talvez. Não peço que você acredite no que eu acredito. Peço apenas que considere o que encontrei nessa fé, porque acho que algumas dessas descobertas servem a qualquer pessoa, tenha ela a crença que tiver. Vou falar da minha janela; a paisagem lá fora, essa, é de todo mundo.

O mercado, disse eu, é cego para o sentido. Ele julga o valor das coisas — quanto alguém pagaria, adotaria, trocaria —, mas não sabe nada sobre por que o trabalho importa, sobre o que dá dignidade ao que fazemos, sobre para que serve, no fim, um ser humano. São perguntas que nenhum preço responde. E são, não por acaso, as perguntas que mais pesam num momento da história em que máquinas começam a fazer parte do que sempre julgamos ser exclusivamente nosso. Este capítulo é sobre elas.

A dignidade do ofício

Começo pela mais concreta, porque é a que sustenta o resto: a dignidade do trabalho, e em especial do trabalho técnico, feito com as mãos e com a mente ao mesmo tempo.

Há uma passagem, logo no início da narrativa bíblica, que sempre me marcou como projetista. Antes de qualquer queda, antes de qualquer maldição, o ser humano é colocado num jardim "para cultivá-lo e guardá-lo".[65] O trabalho, ali, não é castigo — é vocação. Vem antes do pecado, faz parte do desenho original, é o que o ser humano foi feito para fazer. Isso muda tudo na forma de encarar o próprio ofício. O projetista que passa horas ajustando o ângulo

[65] Gênesis 2,15. A colocação do ser humano no jardim "para o cultivar e o guardar" precede, na narrativa bíblica, a queda — razão pela qual a tradição judaico-cristã lê o trabalho não como maldição, mas como parte da vocação original do homem. A maldição posterior (Gênesis 3) recai sobre a *penosidade* do trabalho, não sobre o trabalho em si.

de saída de um molde, o programador que persegue um erro pela madrugada, o torneiro que sente no tato quando a peça está no ponto — nenhum deles está fazendo algo menor, algo indigno, algo a ser terceirizado à primeira oportunidade. Eles estão fazendo aquilo que, numa leitura que atravessa séculos, é próprio da natureza humana: transformar a matéria bruta em ordem, em função, em beleza.

Há uma ideia, cara a alguns escritores cristãos, de que o ser humano, quando cria, age como uma pequena imagem do Criador — não criando do nada, que isso não nos cabe, mas dando forma ao que já existe, sendo uma espécie de subcriador.[66] Não é preciso ser religioso para sentir a força disso. Qualquer um que já fez uma peça funcionar, que já viu a própria ideia virar objeto no mundo, conhece aquela satisfação particular, quase reverente, que não se explica só pelo salário. É o eco de estarmos fazendo aquilo para o qual fomos feitos. E é por isso que a questão central deste livro, quem controla as ferramentas do seu ofício, nunca foi uma questão meramente econômica. É uma questão sobre a dignidade de uma das coisas mais humanas que existem.

A parábola dos talentos

Da dignidade do trabalho nasce um dever, e para nomeá-lo vou recorrer à história que, na minha tradição, melhor o descreve: a parábola dos talentos.[67]

Um homem, de viagem, confia somas diferentes a três servos. Dois deles põem o dinheiro para trabalhar e o multiplicam. O terceiro, com medo de perder o que recebeu, enterra a sua parte no chão e a devolve intacta. E a história, que a maioria espera que elogie a prudência do terceiro, faz o oposto: os que arriscaram e multiplicaram são elogiados; o que enterrou, por medo, é repreendido. A lição atravessa qualquer credo: aquilo que você recebe — talento, capacidade, oportunidade, ferramenta — não lhe foi dado para ser enterrado em segurança, mas para ser cultivado e multiplicado. A mordomia, essa palavra antiga, significa exatamente isto: você não é dono absoluto do que tem, é administrador; e um bom administrador faz render o que lhe foi confiado.

---

[66] O conceito de “subcriador” — o ser humano que, feito à imagem de um Criador, cria por sua vez, não do nada, mas dando forma ao que já existe — é de J. R. R. Tolkien (“Sobre Histórias de Fadas”, 1947; e o poema “Mythopoeia”).

[67] Mateus 25,14-30, a parábola dos talentos. Vale notar que “talento”, no texto original, é uma unidade monetária; o duplo sentido que a palavra adquiriu nas línguas modernas, dinheiro e aptidão, é feliz para os fins deste capítulo, pois a lição de mordomia se aplica igualmente às capacidades que cada um recebe.

Repare como essa ideia ilumina, de um ângulo inesperado, tudo o que discutimos até aqui. O engenheiro que aceita passar a carreira inteira como usuário passivo, sem nunca aprender a moldar as próprias ferramentas, está enterrando um talento no chão. A capacidade (hoje real, ao alcance de qualquer um) de automatizar o próprio trabalho, de dobrar o software à própria vontade, de multiplicar a própria produção, é um talento que nos foi confiado por esta época específica da história. Deixá-lo enterrado, por medo ou por comodismo, dentro de uma gaiola que faz tudo por você, não é humildade — é a atitude do servo que teve medo. Recuperar o controle das próprias ferramentas, aprender, construir, multiplicar o que se sabe fazer: isso, sim, é a boa mordomia daquilo que recebemos. A liberdade que este livro defende não é, nessa luz, um direito apenas. É um dever — o dever de não enterrar o que nos foi dado.

O livre-arbítrio como a raiz de tudo

E chego, assim, à raiz de onde brotam tanto a minha fé quanto a minha convicção política — o ponto em que as duas, para mim, se tornaram a mesma coisa vista de dois ângulos.

A minha tradição afirma que o ser humano foi criado livre. Livre de verdade, inclusive para errar, para se desviar, para recusar o próprio Criador. E sempre me impressionou a audácia dessa ideia: um Deus que teria todo o poder de fazer criaturas obedientes, autômatos incapazes de desobedecer, escolhe em vez disso fazer seres livres, que podem dizer não. Por quê? Porque um amor que não pode ser recusado não é amor, e uma virtude que não pode ser escolhida não é virtude. O bem forçado não é bem. A obediência sem alternativa não tem mérito nenhum. Para que a bondade, o amor e a fé signifiquem alguma coisa, é preciso que fossem livremente escolhidos — e para serem livremente escolhidos, era preciso que o não também fosse possível.

Foi ao encontrar essa ideia, já adulto, depois de anos me chamando apenas de liberal, que a minha fé e a minha política se fundiram. Porque é exatamente o mesmo princípio, dito em outra linguagem: uma liberdade que não pode ser recusada não é liberdade; um bem imposto de cima, por decreto, por mais bem-intencionado, esvazia-se de mérito moral justamente por ser imposto. O planejador que força as pessoas a serem virtuosas à sua maneira comete, no plano político, o mesmo erro que um deus autoritário cometeria no plano cósmico: confunde o comportamento correto com a bondade, quando a bondade só existe onde houve escolha. Não fui eu que inventei essa ponte

entre a fé cristã e a defesa da liberdade individual; encontrei-a pronta, e reconheci nela algo que eu já sentia sem saber nomear. O livre-arbítrio é a raiz. A responsabilidade, de que falei no capítulo nove, é o tronco. E tudo o mais — a dignidade, a mordomia, o direito de construir a própria vida — são os galhos.

O autor do próprio legado

Se somos livres para escolher, então somos livres também para escolher o que carregamos do passado. E aqui a conversa deixa de ser abstrata e fica, para mim, muito pessoal — porque preciso falar do meu próprio sangue.

Sou descendente de americanos. Meu sobrenome, Keese, e alguns dos primeiros nomes que se repetem, de geração em geração, na minha família — May, Lincoln — vieram de imigrantes que atravessaram um hemisfério inteiro no século dezenove e se instalaram no interior de São Paulo, na região de Santa Bárbara d'Oeste. Eram os chamados confederados: gente que deixou o Sul dos Estados Unidos depois da derrota na Guerra Civil e veio recomeçar a vida do zero em terra estranha, incentivada por um imperador que queria desenvolver a lavoura. A história da minha família se cruza com a fundação da primeira igreja batista do Brasil, erguida por esses imigrantes na mesma região — a fé deles, não exatamente a minha: hoje eu congrego na Congregação Cristã no Brasil, mesma raiz cristã em outro galho. Alguns dos meus antepassados têm ruas com o nome deles; um dos meus avós, dentista, dá nome a um posto de saúde na cidade onde cresci.

E há, nessa ascendência, uma coisa que me alegra por razões que já não são só familiares. Rothbard, em *Por uma Nova Liberdade: O Manifesto Libertário*, descreve a fundação dos Estados Unidos como uma das revoluções mais genuinamente libertárias que a história conheceu: um país nascido da recusa a um poder distante, com desconfiança do Estado grande e apreço pela liberdade individual inscritos na própria origem. É dessa América que me orgulho de trazer algo no sangue — a da ideia fundadora, não a que veio depois dela. Porque, na minha leitura, aquela promessa foi sendo traída: primeiro pela criação de um banco central, que concentrou o que a origem queria disperso; depois pela influência de sociedades secretas como a maçonaria, que preferem o corredor fechado à praça aberta. Assumo isso como convicção minha, não como sentença da história.

Preciso ser honesto sobre essa herança, porque a honestidade é o mínimo que ela exige. Nem tudo nela me orgulha. Os confederados fugiam de uma causa derrotada que estava manchada pela escravidão, e essa mancha não

some porque atravessou o oceano — eu tenho vergonha dela, sem rodeios. E alguns dos meus próprios antepassados pertenceram exatamente a uma dessas sociedades — a maçonaria. Não guardo por ela nenhuma reverência: para mim, é uma irmandade secreta que troca a luz da praça pela penumbra do salão fechado, e nada que dependa de segredo para existir merece a minha lealdade. Repudio essa parte da herança sem rodeios. E há, ao lado disso, coisas que me enchem de orgulho genuíno: a fé que ajudaram a plantar num país novo, a coragem de recomeçar sem nada, a disposição de construir um lugar onde não havia nenhum. A minha linhagem é, como quase toda linhagem, uma mistura de luz e sombra.

E é aqui que a minha fé me deu a resposta mais libertadora que eu conheço. Há um princípio, num dos livros mais antigos da Bíblia, que diz, com todas as letras, que o filho não carregará a culpa do pai, nem o pai a culpa do filho — que cada alma responde por si, pelo que ela mesma faz.[68] Não existe culpa herdada. Ninguém nasce réu dos pecados dos seus antepassados, nem herdeiro automático das virtudes deles. Cada um responde pela própria vida, pelas próprias escolhas, pelo que faz do que recebeu. E isso, que é uma das verdades centrais da minha fé, é também (de novo, a mesma coisa vista de dois ângulos) o coração do individualismo moral que sustenta a liberdade: a pessoa, não a tribo, não a raça, não a linhagem, é a unidade da responsabilidade.

Por isso posso olhar para a minha herança inteira, de olhos abertos, e dizer: aquilo que foi vergonhoso não é o meu legado. Não porque eu o esconda, mas porque eu não o escolho. Eu escolho carregar adiante a fé, a coragem de recomeçar, a dignidade do trabalho — e deixo no chão, sem culpa e sem orgulho, o que não quero levar. Não sou prisioneiro do meu sangue. Sou o autor do meu legado. E acho profundamente significativo, dessas coisas que só a vida escreve, que eu viva hoje em Piracicaba, a poucos quilômetros de onde viveram esses antepassados, tendo voltado, sem planejar, ao chão de onde eles partiram para recomeçar. Vim reescrever, à minha maneira, a mesma velha história da família: a de construir o próprio lugar onde não havia nenhum.

[68] Ezequiel 18,20: “A alma que pecar, essa morrerá; o filho não levará a iniquidade do pai, nem o pai levará a iniquidade do filho.” O capítulo inteiro de Ezequiel 18 é uma das afirmações mais fortes, em toda a literatura antiga, da responsabilidade moral *individual* contra a ideia de culpa herdada ou coletiva — princípio que ressoa diretamente no individualismo moral da tradição liberal.

O que a inteligência artificial não pode fazer por você

Fecho reunindo tudo — a dignidade, a mordomia, o livre-arbítrio, o legado escolhido — num único ponto, porque ele é a resposta mais profunda que este livro tem a oferecer para a era que estamos vivendo.

Uma inteligência artificial pode gerar código, desenhar geometrias, escrever textos, otimizar processos. Pode multiplicar a sua produção de maneiras que teriam parecido mágica há cinco anos. Mas há um conjunto de coisas que ela não pode fazer por você, não por limitação técnica passageira, mas por natureza. Ela não pode escolher, no seu lugar, o sentido do que você faz. Não pode carregar a sua responsabilidade. Não pode multiplicar os seus talentos por você — só você decide se os enterra ou os cultiva. Não pode dar dignidade ao seu trabalho; a dignidade vem de quem trabalha, não da ferramenta. E não pode, de forma alguma, escolher por você qual legado você deixa — o que carrega adiante e o que larga no caminho. A máquina pode multiplicar o que você produz. Ela não pode multiplicar a sua alma, porque não tem acesso a ela.

É isto que me deixa, no fim, mais esperançoso do que assustado com o que vem por aí. As máquinas ficarão cada vez melhores em fazer. Mas as perguntas que mais importam nunca foram sobre o fazer — foram sempre sobre o *para quê*, o *por quê*, o *quem eu escolho ser*. E essas continuam, irredutivelmente, humanas. Continuam suas. A era da inteligência artificial, longe de nos tornar dispensáveis, devolve a essas perguntas antigas uma urgência que talvez tivéssemos esquecido. Quanto mais a máquina faz por nós, mais precisamos saber para que estamos aqui.

O que vem a seguir

Aqui se fecha a terceira parte deste livro, a mais filosófica, e com ela a viagem para dentro. Diagnostiquei a gaiola, mostrei a saída e o seu preço, e procurei os fundamentos — a responsabilidade, o mercado como juiz, e agora o sentido, a dignidade, a liberdade de escolher o próprio caminho e o próprio legado. Foi uma descida até a raiz.

A quarta e última parte faz o movimento contrário: sobe de volta à superfície, para a ação, para o futuro concreto. Porque toda essa reflexão só vale alguma coisa se ela se traduzir em algo construído. Se você é livre, responsável, mordomo dos próprios talentos e autor do próprio legado, então uma pergunta prática e empolgante se impõe: o que você vai *construir* com tudo isso? Como se ganha a vida, na prática, sendo um engenheiro livre? É disso que trata o

próximo capítulo, o engenheiro como empreendedor de si mesmo, e é para o mundo real, o das contas a pagar e das oportunidades a agarrar, que vamos voltar agora.

Nota: capítulo assumidamente pessoal e confessional, conforme anunciado ao fim do Capítulo 10. As referências são as fontes primárias citadas (passagens bíblicas; Tolkien) e os fatos da linhagem do autor, documentados a partir de registros familiares e históricos da região de Santa Bárbara d'Oeste. A dimensão histórica dos imigrantes confederados é tratada com honestidade factual, sem apologia — reconhecendo tanto o que há de sombrio (a ligação com a escravidão) quanto o que o autor escolhe carregar adiante.

Para levar deste capítulo

Tranca 6 — Domínio. Multiplicar o que lhe foi confiado, em vez de enterrar por medo ou comodismo, é o que constrói competência que ninguém tira de você.

Faça agora — 10 minutos. Escreva um talento seu que está enterrado — algo que você sabe que consegue fazer e vem adiando há anos. Embaixo, escreva a menor ação possível para desenterrá-lo nesta semana. Pequena mesmo: trinta minutos, não trinta dias.

Parte IV

O Futuro: Construindo a Sua Própria Stack

Capítulo 12 — O Engenheiro Como Empreendedor de Si Mesmo

A terceira parte deste livro foi uma descida até a raiz — a responsabilidade, o sentido, a liberdade de escolher o próprio legado. Agora a quarta parte faz o caminho de volta, subindo à superfície, para o mundo das contas a pagar e das oportunidades a agarrar. Porque toda aquela reflexão só vale alguma coisa se ela se traduzir em algo que se possa construir. E a pergunta que fecha a filosofia e abre a prática é simples e enorme: se você é livre, responsável, mordomo dos próprios talentos e autor do próprio legado — o que você vai *construir* com isso?

Vou responder da forma mais honesta que conheço: mostrando o plano que desenhei para mim mesmo — a estrutura que deixei pronta no papel para o dia em que eu precisar dela, se um dia o emprego faltar ou a hora certa chegar. Não um caso de sucesso encerrado para vender a você uma fórmula, nem um negócio já em marcha, mas um projeto pensado nos mínimos detalhes, com as suas dúvidas e os seus riscos à mostra. É a única forma de falar de empreender sem mentir.

A mudança estrutural

Antes do meu caso, os fatos maiores, porque eles mudaram o chão sob os pés de todo profissional técnico e a maioria das pessoas ainda não percebeu.

O mercado de trabalho está sendo, silenciosamente, reescrito. O que antes era empacotado como um emprego — um cargo fixo, um vínculo único, um patrão — está sendo cada vez mais fatiado em projetos. Em 2026, a economia do trabalho independente, o chamado mercado *gig*, foi avaliada em torno de 674 bilhões de dólares no mundo, crescendo a quase 16% ao

ano.[69] Nos Estados Unidos, mais de 72 milhões de pessoas já trabalham de forma independente, gerando cerca de 1,5 trilhão de dólares por ano — mais de 5% de toda a economia americana.[70] E não é trabalho de baixo valor: o número de independentes americanos que ganham mais de cem mil dólares por ano bateu recorde, e continua subindo.[71]

Mas o dado que mais importa para você, engenheiro, é este: 77% dos líderes empresariais dizem que a inteligência artificial está *aumentando* a necessidade que eles têm de talento especializado e sob demanda, em vez de contratações tradicionais de tempo integral.[72] Leia de novo. A IA não está, ao menos por ora, eliminando o especialista independente — está criando mais demanda por ele. E os profissionais que dominam ferramentas de IA cobram, em média, um prêmio de 56% sobre os que não dominam — o número que apareceu no Interlúdio II e que aqui mostra a sua outra face.[73] Alguém resumiu a transformação numa frase que eu queria ter escrito: o trabalho está sendo *re-escopado em projetos antes de ser re-postado como vagas*.[74] A vaga fixa não é mais o único caminho — e, para o profissional certo, já não é sequer o mais seguro.

Por que isto é possível agora

Nada disso seria possível dez anos atrás, e a razão é o assunto deste livro inteiro. O que permite a um pequeno grupo de especialistas — ou mesmo a um profissional sozinho — fazer hoje o que antes exigia uma empresa inteira é a convergência de tudo o que discuti até aqui: as ferramentas abertas,

[69]O mercado global de trabalho independente (*gig economy*) foi projetado em cerca de US$ 674,1 bilhões para 2026, com taxa de crescimento anual composta de aproximadamente 15,79% (relatórios de 2026 compilados por Jobbers, AutoFaceless e HRStacks, a partir de dados de mercado).

[70]Estimativas de 2025-2026 (MBO Partners, DemandSage) apontam entre 72,9 e 76,4 milhões de trabalhadores independentes nos EUA, gerando cerca de US$ 1,5 trilhão em receita anual — mais de 5% do PIB americano.

[71]A MBO Partners reporta 5,6 milhões de independentes americanos ganhando mais de US$ 100 mil por ano em 2025, alta de 19% sobre os 4,7 milhões de 2024, e quase o dobro dos 3 milhões de 2020.

[72]Pesquisa da Upwork (outubro de 2025, com 349 líderes empresariais): 77% afirmaram que a IA está aumentando a necessidade de talento especializado e fracionado (por projeto), em vez de papéis tradicionais de tempo integral.

[73]Prêmio salarial de 56% para freelancers com habilidades especializadas em IA sobre papéis tradicionais (DemandSage/Jobbers, 2026). Medições mais conservadoras, da própria Upwork, apontam prêmio acima de 40% em trabalhos ligados a IA — a direção é a mesma.

[74]A formulação “o trabalho está sendo re-escopado em projetos antes de ser re-postado como vagas” é de análise da Digital Applied sobre os dados de 2026, capturando a convergência entre a alta da IA e a fração crescente de trabalho independente especializado.

que você pode possuir em vez de alugar; os formatos que conversam, que te libertam de depender de um único fornecedor; e a inteligência artificial, que automatiza o trabalho braçal e multiplica a sua produção. Junte as três coisas e o resultado é uma mudança histórica: a barreira de entrada para montar uma operação de engenharia séria despencou.

O engenheiro que domina as próprias ferramentas não precisa mais de uma corporação para lhe emprestar a infraestrutura, o software, o alcance. Ele carrega tudo isso consigo. Pode atender um cliente do outro lado do mundo sem pedir licença a ninguém, automatizar a papelada que antes exigia um departamento, competir com estruturas muito maiores porque não carrega o peso delas. A liberdade que este livro defende deixou de ser uma questão de princípio e virou uma vantagem competitiva concreta.

Um estudo de caso vivo: o hub

Deixe-me mostrar como isso se parece na prática, com o exemplo que conheço por dentro porque é meu: um hub de engenharia de precisão que eu e um sócio de projeto desenhamos para ativar se e quando for preciso — uma pequena rede de especialistas no interior de São Paulo, pronta no papel para sair dele no dia certo.

A ideia central é simples e é uma aplicação direta de tudo o que este livro defende. Em vez de uma empresa tradicional, com folha de pagamento fixa e hierarquia, o hub é uma rede de profissionais livres que se juntam por projeto. Cada especialista — o que faz a simulação de injeção, o que projeta os eletrodos, o que gera o CAM, o que cuida das automações — entra por comissão, remunerado pelo valor que entrega, não por bater ponto. Ninguém é dono de ninguém. É a liberdade e a responsabilidade do nono capítulo transformadas em estrutura de negócio: cada um responde pelo próprio trabalho e é pago por ele.

O hub fatura em quatro camadas, cada uma com um papel distinto. A primeira é a de **serviços técnicos** — projeto de moldes, eletrodos, CAM, simulação, dispositivos —, que é o motor de caixa, o que paga as contas. A segunda é a de **produtos próprios**, um software de listas e orçamentos e uma linha física enxuta, que é a aposta escalável, a que um dia cresce sem depender de horas trabalhadas. A terceira é a de **software e automações**, de margem alta e receita recorrente. E a quarta é a de **consultoria, treinamento e conteúdo**, que serve tanto de receita quanto de forma barata de fazer o mundo saber que existimos. Nenhuma dessas camadas é original sozinha; a força está em combiná-las de modo que uma financie a outra.

E há um foco que nos torna defensáveis, porque especialização é a única muralha que um pequeno não pode comprar mas pode construir: nós nos concentramos em micro-moldes de precisão, do tipo que roda em máquinas Babyplast, um nicho que a maioria das ferramentarias trata mal por encarar molde pequeno como “molde grande em miniatura”; e em simulação de injeção que sabe, de fato, tratar empeno — não só rodar o software e entregar um relatório, mas resolver o problema. Não somos mais uma ferramentaria genérica competindo por preço. Somos os especialistas em um punhado de coisas difíceis que pouca gente sabe fazer direito.

Os princípios que tornam isso livre

O que faz desse hub uma encarnação da filosofia deste livro, e não apenas mais um negócio, são os princípios que escolhemos para governá-lo — e cada um deles é um dos temas destas páginas virado regra prática.

O primeiro: **o serviço financia o produto. Nunca dívida, nunca investidor.** Crescer com o próprio caixa é mais lento, muito mais lento, mas mantém a autonomia total — o negócio responde só a nós. Aceitar dinheiro de investidor seria, em pequena escala, o mesmo que entrar no jardim murado: um alívio imediato em troca de passar a responder a outra pessoa. Preferimos a estrada mais longa e livre.

O segundo é um mecanismo que chamamos de **Fundo de Produtos**: uma fatia fixa, 15%, da margem de cada trabalho é separada para financiar o desenvolvimento dos produtos próprios, sem tocar no caixa de operação. É a parábola dos talentos do capítulo anterior transformada em regra contábil: uma parte do que ganhamos hoje não é consumida, é semeada, para que amanhã exista algo que renda sem depender das nossas horas. É mordomia com planilha.

O terceiro nasce de uma constatação dura: o recurso mais escasso do hub não é dinheiro, é a hora de projeto — a minha e a do meu sócio, sem a qual quase nada acontece, porque cada serviço técnico depende de um projeto existir primeiro. Um plano honesto não finge que esse gargalo não existe; ele o protege, precifica e, sobretudo, o *multiplica*. E é aqui que a minha história se fecha num círculo: as automações que aprendi a construir com inteligência artificial, aquelas que descrevi lá no sexto capítulo, existem exatamente para multiplicar essa hora escassa — para que o mesmo par de projetistas dê conta de mais, sem contratar mais gente nem trabalhar até morrer. A ferramenta que eu construí para mim é o que tornaria o negócio possível. Não há exemplo mais literal do que significa recuperar o controle das próprias ferramentas.

A honestidade sobre o risco

Eu seria um vendedor de gaiolas, o pior tipo, se pintasse isso como fácil. Não é. E a mesma honestidade que apliquei aos riscos da liberdade no oitavo capítulo eu devo aplicar aqui.

Empreender assim é difícil, lento e incerto. Os dados animadores do começo deste capítulo escondem uma verdade dura: os ganhos do trabalho independente são altamente estratificados. Uma minoria de especialistas muito bem posicionados fatura muito; uma multidão de trabalhadores de tarefas genéricas fatura pouco.[75] A promessa não é "largue tudo e fique rico" — essa é uma mentira de guru de internet. A promessa é mais modesta e mais verdadeira: para o profissional *especializado*, que domina as próprias ferramentas e resolve problemas difíceis, existe hoje um caminho de autonomia que não existia antes. O meu hub não é um sucesso consumado; é uma aposta em andamento, que pode dar certo ou não. Crescer sem dívida significa crescer devagar. A autonomia tem um preço, e o preço é carregar o risco nas próprias costas, sem patrão para culpar e sem salário garantido no fim do mês.

Mas — e este "mas" é o livro inteiro — a alternativa não é livre de risco. Ela só esconde o risco melhor. Depender inteiramente de um único empregador, ou de uma única plataforma que pode mudar as regras quando quiser, é correr um risco tão grande quanto o de empreender; apenas é um risco que não aparece até o dia em que a demissão ou a mudança de política chega. Não existe a opção sem risco. Existe a escolha entre o risco visível da independência, que está nas suas mãos, e o risco invisível da dependência, que está nas mãos de outro. Eu escolhi o primeiro. Não porque seja mais seguro, mas porque é meu.

Não é para todos, e tudo bem

Preciso fechar com uma ressalva, para não cair na pregação que critico. Não estou dizendo que todo engenheiro deve largar o emprego amanhã e montar um hub. Muitos não devem, e está tudo bem. Empreender não é uma virtude obrigatória, e a estabilidade de um bom emprego é uma escolha legítima e, para muitas vidas, a mais sábia.

[75] A ressalva é essencial e honesta: pesquisas (incluindo trabalhos do NBER) documentam ampla dispersão de renda no trabalho independente — uma pequena elite de especialistas captura grande parte dos ganhos, enquanto muitos trabalhadores de tarefas genéricas ganham pouco. O caminho de autonomia descrito aqui vale para o profissional especializado, não é uma promessa universal.

O que eu defendo é outra coisa, mais sutil e mais universal: que você seja dono das suas ferramentas e das suas capacidades a ponto de *poder* ser independente, quer você exerça essa opção ou não. Ser "empreendedor de si mesmo" não é, no fundo, ter um CNPJ; é uma postura.[76] É o engenheiro que, mesmo dentro de uma empresa, domina o próprio ofício de tal forma que não está preso a ela pelo medo, e sim ligado a ela por escolha. Esse profissional é mais livre e, paradoxalmente, mais valioso para o próprio empregador, porque traz para a mesa alguém que pensa como dono, não como refém. A opção de partir é o que dá dignidade à decisão de ficar. Ter para onde ir é o que te faz livre onde você está.

O que vem a seguir

Usei, neste capítulo, o hub que projetei como estudo de caso — e defendi um caminho de autonomia construído por fora, com negócio próprio, sócios, camadas de receita. É legítimo, mas eu sei que ele não descreve a situação da maioria de quem lê este livro. A maior parte dos engenheiros tem, e vai continuar tendo, um emprego — e faria a pergunta óbvia diante de tudo o que escrevi aqui: *e eu, que não vou montar hub nenhum, o que faço com isso?*

Essa pergunta merece uma resposta inteira, e é a próxima pausa do livro. Porque quase tudo o que defendi neste capítulo — possuir as ferramentas, dominar o ofício, manter a porta de saída aberta — vale igualmente para quem tem carteira assinada. Muda o terreno, não o princípio. É disso que trata o Interlúdio III.

Fontes dos dados de 2026: relatórios de gig economy de Jobbers, DemandSage, Digital Applied, HRStacks, AutoFaceless, MBO Partners e Upwork (In-Demand Skills 2026 e pesquisas com líderes empresariais). O estudo de caso do hub de engenharia baseia-se na experiência direta do autor. Referência conceitual: John Locke, "Segundo Tratado sobre o Governo" (1689), sobre autopropriedade.

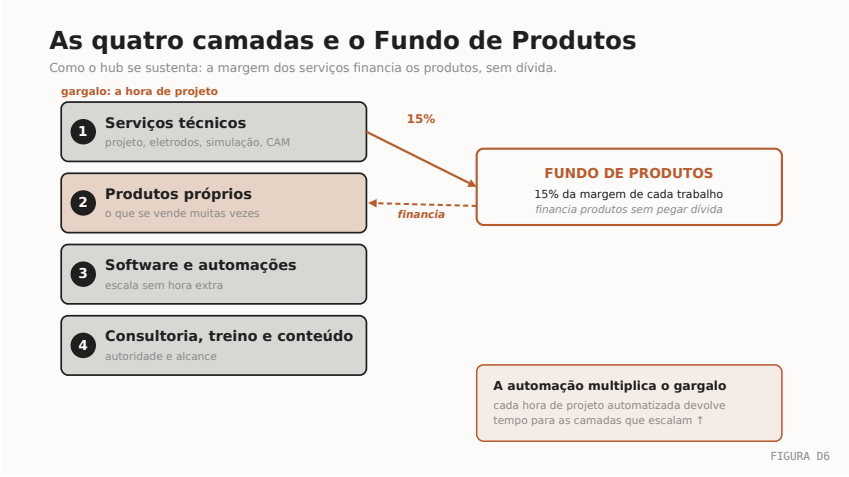
---

[76] A ideia de que cada pessoa tem uma propriedade primária sobre si mesma (sobre a própria pessoa, o próprio trabalho e os frutos dele) é um dos fundamentos do pensamento liberal clássico, articulada por John Locke no *Segundo Tratado sobre o Governo* (1689). "Empreendedor de si mesmo", no sentido deste capítulo, é a tradução prática dessa autopropriedade: assumir a gestão consciente das próprias capacidades como um ativo que é seu, esteja você empregado ou não.

Para levar deste capítulo

Tranca 5 — Renda. Não se trata de largar o emprego: trata-se de deixar de ter uma única mão capaz de cortar tudo o que você ganha.

Faça agora — 20 minutos. Calcule a sua hora real: pegue tudo o que você ganhou nos últimos doze meses e divida pelo número de horas que você de fato trabalhou, incluindo as não pagas. O número que aparecer costuma ser desconfortável — e é a base de qualquer decisão que você venha a tomar sobre a sua carreira.



Interlúdio III — O Engenheiro Livre Dentro da Empresa

Terceira e última pausa deste livro. A objeção é a mais comum de todas, e a que mais gente tem na ponta da língua: "tudo bem, mas eu tenho um emprego".

O capítulo anterior descreveu o negócio próprio que planejei construir. E eu sei exatamente qual foi a sua reação, porque é a reação certa: *bonito, mas eu tenho um emprego.*

Você bate ponto. Usa a máquina da empresa, com o software que a empresa comprou, sob uma política de TI que não te deixa instalar coisa nenhuma. Tem conta para pagar no dia dez e não tem colchão financeiro para bancar seis meses de experimento. E não quer largar nada disso — talvez você até *goste* do que faz, e ache que a sua empresa é um lugar decente para trabalhar.

Este interlúdio é para você, que é a maioria. E ele começa com a afirmação mais importante deste livro inteiro: **quase tudo o que eu defendo aqui não exige que você saia.**

Se o livro fosse honesto só com quem quer empreender, ele seria um panfleto para uma minoria. A tese central — possuir as próprias ferramentas, dominar o próprio ofício, manter a porta de saída aberta — vale integralmente para quem tem carteira assinada. Muda o *como*, não o *quê*. E há um ponto que precisa ser dito com todas as letras antes de qualquer outra coisa, porque quase ninguém explica isso a um engenheiro.

De quem é a automação que você escreve no trabalho?

Suponha que você siga o conselho do sexto capítulo: pega a tarefa mais irritante da sua semana, passa três noites construindo uma automação com ajuda de inteligência artificial, e ela funciona. Economiza seis horas por semana do time inteiro. De quem é essa ferramenta?

No Brasil, a resposta está na Lei do Software, a Lei 9.609 de 1998, e ela é mais dura do que a maioria das pessoas imagina. Pelo artigo 4º,

salvo estipulação em contrário no contrato, os direitos sobre um programa desenvolvido durante a vigência do contrato de trabalho pertencem **exclusivamente ao empregador** — quando o vínculo é destinado a pesquisa e desenvolvimento, quando a atividade de desenvolvimento está prevista, ou, e é aqui que quase todo mundo se encaixa sem perceber, quando ela **decorre da própria natureza dos encargos** do cargo.[77] E o parágrafo primeiro fecha a porta: salvo ajuste em contrário, a sua compensação por isso se limita ao salário que você já recebe.

Existe uma saída, mas ela é estreita e cumulativa. Pelo parágrafo segundo, o programa é seu, exclusivamente seu, quando foi gerado **sem relação com o contrato de trabalho** *e* **sem utilizar recursos, informações tecnológicas, segredos industriais e de negócios, materiais, instalações ou equipamentos do empregador**. As duas condições, juntas. Falhou em uma, perdeu as duas.

Isso não é teoria de manual. Um caso real da Justiça do Trabalho: um empregado desenvolveu um programa, processou a empresa pedindo indenização pela criação, e perdeu. O tribunal registrou que o software havia sido feito durante a jornada, com recursos da empresa, a licença do software e o tempo, e para otimizar tarefas inerentes ao próprio contrato de trabalho. Titularidade exclusiva da empregadora.[78] Repare no detalhe cruel: *para otimizar tarefas inerentes ao contrato* é exatamente o que eu venho recomendando que você faça. E foi justamente isso que afastou a hipótese de o programa ser dele.

Não sou advogado, contratos podem dispor diferente, e cada situação tem particularidades que só um profissional olhando o seu caso consegue avaliar.

[77]Lei nº 9.609, de 19 de fevereiro de 1998 (Lei do Software), artigo 4º, caput: salvo estipulação em contrário, pertencem exclusivamente ao empregador os direitos sobre programa de computador desenvolvido durante a vigência do contrato de trabalho expressamente destinado a pesquisa e desenvolvimento, ou em que a atividade do empregado seja prevista, ou ainda que decorra da própria natureza dos encargos do vínculo. O § 1º limita a compensação do empregado à remuneração convencionada, ressalvado ajuste em contrário. O § 3º estende o mesmo tratamento a bolsistas e estagiários. Nada aqui constitui orientação jurídica: contratos individuais podem dispor de forma diversa, e situações concretas exigem avaliação profissional.

[78]Decisão da Justiça do Trabalho em recurso ordinário sobre a titularidade de programa de computador desenvolvido por empregado: o tribunal reconheceu a titularidade exclusiva da empregadora com base no art. 4º, caput, da Lei nº 9.609/98, registrando que o programa foi desenvolvido durante a jornada, com recursos da empresa (licença de software e tempo) e para otimizar tarefas inerentes ao contrato — o que afasta a hipótese de titularidade do empregado prevista no § 2º. Em sentido convergente, o TJ-MG (Apelação Cível 1.0000.25.371211-1/001, julgada em novembro de 2025) reconheceu a titularidade patrimonial da empresa sobre software desenvolvido com recursos técnicos, humanos e materiais dela.

Se você tem algo relevante em jogo, mande alguém competente ler o seu contrato — vale muito mais que qualquer parágrafo meu. Mas a regra geral é essa, e ela é o padrão do mundo, não uma esquisitice brasileira.

Por que isso não é motivo para não automatizar

Agora a parte que importa, e que muda completamente a leitura do que acabei de escrever.

Nada disso é argumento para você cruzar os braços. É argumento para você entender **o que exatamente você está ganhando**, porque não é o que parece.

Você não vai ficar dono da automação. Vai ficar dono de outra coisa, muito mais valiosa: **a ferramenta fica; a capacidade vai com você.** O código que você escreveu pertence à empresa e continua lá quando você sair. Mas o fato de que você *sabe construir aquilo* é seu, para sempre, e é intransferível. E é justamente a capacidade — não o artefato — que é escassa no mercado, que aparece nos 20% a 40% de prêmio salarial do interlúdio anterior, e que ninguém pode registrar em cartório contra você.

Some a isso o que já vimos: quem automatiza a própria função raramente perde o emprego, muda de função. Vira quem constrói as ferramentas do time. Essa pessoa é a última a sair de qualquer lugar. Você trocou a propriedade de um arquivo por competência permanente e por posição — e, nessa troca, quem sai ganhando é você.

O que você **não** pode fazer é confundir as duas coisas. Se você tem a ambição de um dia vender um produto seu, não o construa no computador da empresa, no horário da empresa, usando dados da empresa, achando que ele é seu. Não é. Construa limpo: seu tempo, sua máquina, seus dados, sem relação com as atribuições do seu cargo. É mais lento e é a única forma que sobrevive a uma discussão jurídica.

O que você pode fazer sem pedir licença a ninguém

Feita essa distinção, vamos ao terreno prático. Há uma quantidade grande de coisas neste livro que não conflitam com política nenhuma e que você pode começar hoje.

Formatos. Exportar o seu trabalho em formatos neutros, manter um acervo do que você produziu, entender o que se perde na conversão — isso não é ato de rebeldia, é boa prática de engenharia. Nenhum gestor no mundo vai te

repreender por garantir que o trabalho da empresa seja portátil. Pelo contrário: é argumento de continuidade de negócio.

Aprender. A capacidade que você constrói é sua por definição. Ninguém pode te proibir de entender melhor a ferramenta que você usa oito horas por dia.

A interface de programação do software que a empresa já paga. Este é o ponto mais subestimado de todos. Se a sua empresa comprou o Solid Edge, o SOLIDWORKS ou o Inventor, ela comprou junto a interface de automação deles — que está lá, instalada, licenciada, e quase sempre intocada. Automatizar dentro dela não requer instalar nada novo, não passa por TI, não gera custo e não viola política nenhuma. É a porta mais larga que existe, e ela está destrancada na sua frente.

Rastreabilidade e método. Procedimentos, listas de verificação, versionamento do seu próprio trabalho. É a disciplina do nono capítulo, e ela não pede autorização.

Repare que essas quatro coisas, sozinhas, já movem duas das Seis Trancas — a dos dados e a da capacidade. Sem uma única conversa com o seu chefe.

O que exige pedir — e como pedir

Para o resto, você vai precisar de autorização. E aqui é onde a maioria dos engenheiros erra, porque pede da forma errada.

O erro clássico é pedir uma **coisa**: “posso instalar o FreeCAD?”, “dá para eu usar tal ferramenta de IA?”. Um pedido assim chega ao gestor como custo, risco e trabalho extra, sem nenhum benefício visível — e a resposta racional dele, sem mais informação, é não.

O jeito que funciona é pedir um **resultado**, com número. Mais ou menos assim: *“a montagem da lista de materiais consome cerca de seis horas por semana do time. Fiz um teste e consigo derrubar para vinte minutos. Precisaria de duas semanas para um piloto num projeto só, sem custo de licença e sem nenhum dado saindo da nossa rede. Se não funcionar, a gente descarta e não perdemos nada.”*

Repare no que essa frase carrega: um problema que o gestor reconhece, um número antes e um depois, um escopo pequeno, um custo declarado como zero e, o mais importante, uma saída sem constrangimento se der errado. Você não pediu uma ferramenta. Ofereceu um resultado com risco limitado.

Duas recomendações práticas sobre isso. **Meça antes.** Cronometre a tarefa por duas semanas antes de propor qualquer coisa; sem o "antes", o "depois" não convence ninguém, e engenheiro só respeita número. E **comece pelo que ninguém quer fazer.** A tarefa chata que todo mundo empurra é onde há menos resistência política e mais gratidão. Automação que ameaça o trabalho de alguém encontra oposição; automação que elimina o que ninguém queria fazer encontra aliados.

A tentação de fazer por fora

Preciso encarar de frente o atalho que você já pensou em usar, ou provavelmente já está usando.

Os números de 2026 são desconfortáveis e vale colocá-los na mesa. O relatório anual de investigação de violações de dados da Verizon apontou que 45% dos funcionários já são usuários regulares de inteligência artificial em dispositivos corporativos, autorizados ou não, com as detecções de uso não sancionado quadruplicando em um ano.[79] Uma pesquisa com profissionais de grandes empresas encontrou 66% usando ferramentas de IA acreditando estar contra a política interna.[80] Do outro lado do balcão, apenas 18% das organizações têm política formal de segurança para IA, e cerca de um quarto não tem política nenhuma.[81]

Ou seja: a maior parte dos engenheiros deste país está trabalhando num vácuo — usando ferramentas que a empresa não aprovou porque a empresa também não desaprovou, porque ninguém decidiu nada.

E antes que eu passe ao sermão, a causa disso não é falta de caráter. Um levantamento de 2026 mostrou que só 13% dos usuários de IA no trabalho sentem que seriam recompensados por reinventar a forma de trabalhar

[79] Verizon, Data Breach Investigations Report 2026: 45% dos funcionários são usuários regulares de ferramentas de IA em dispositivos corporativos, autorizados ou não; as detecções de "IA sombra" quadruplicaram em um ano, tornando-se a terceira ação interna não maliciosa mais comum registrada em dados de violação.

[80] Pesquisa PagerDuty com profissionais de grandes empresas (2026): 66% relataram usar ferramentas de IA acreditando que isso contrariava a política da companhia. Levantamento da BlackFog no mesmo período apontou 49% usando IA de formas não aprovadas pelo empregador, sendo que 63% consideram aceitável usar sem supervisão de TI quando não existe alternativa homologada.

[81] Salesforce, Workforce AI Survey 2026: 67% dos funcionários usam ferramentas de IA no trabalho, mas apenas 18% das organizações possuem política formal de segurança para IA. Levantamento da ISACA (2026) aponta que cerca de 25% das organizações não têm política de IA alguma. Análise da Productiv (2026) estima que a empresa média usa 14 ferramentas de IA distintas, das quais a área de TI conhece apenas 4 ou 5.

se o resultado não fosse garantido.[82] Leia isso com atenção: quando a experimentação não é premiada e o erro é punido, a experimentação não para — ela apenas desce ao subsolo. É um problema de incentivo, não de índole.

Dito isso, preciso ser direto com você sobre onde está a linha, porque há uma diferença enorme entre usar uma ferramenta não homologada e vazar o que não é seu.

Colar a geometria confidencial de um cliente numa inteligência artificial pública **não é uma zona cinzenta.** É o problema do sétimo capítulo na sua forma mais concreta: aquele desenho não é seu para enviar. A Samsung restringiu o uso de IA generativa internamente depois que funcionários colaram código-fonte numa ferramenta pública. O que está em jogo não é a política de TI — é o contrato de sigilo da sua empresa com o cliente dela, e a sua própria pele.

E aqui está a virada boa, que é onde este livro paga o que prometeu. A solução para esse problema é exatamente a tese do sétimo capítulo: **rodar localmente.** Um modelo de pesos abertos na sua máquina não manda nada para lugar nenhum. Não há dado saindo, não há política violada, não há cliente exposto.

Isso muda a sua posição na conversa com a empresa de um jeito que vale ouro. Você deixa de ser o sujeito que quer usar uma ferramenta duvidosa e passa a ser **quem traz a solução para um risco que a empresa já tem** — porque, como os números mostram, metade do time provavelmente já está usando IA por fora, e a empresa não sabe. Você chega com privilégio mínimo, aprovação humana, dado que não sai da máquina: o vocabulário inteiro do oitavo capítulo. Não é o cara do problema. É o único que entendeu o problema.

Poucas vezes na carreira aparece uma chance dessas de mudar de posição dentro de uma organização. E ela está aberta agora, porque a maioria das empresas ainda não decidiu o que fazer.

[82] Microsoft, dados de 2026 sobre adoção de IA no trabalho: apenas 13% dos usuários afirmam que seriam recompensados por reinventar a forma de trabalhar caso os resultados não fossem garantidos — indicador de que a experimentação, quando não incentivada, migra para canais não sancionados.

Carreira é diferente de cadeira

Volto, para fechar, à imagem do interlúdio anterior, a escada sendo puxada, e à frase do capítulo doze: a opção de partir é o que dá dignidade à decisão de ficar.

Aqui vai um teste incômodo que eu recomendo você fazer, mentalmente, agora. Pegue tudo o que você sabe fazer profissionalmente e separe em duas partes: o que funcionaria em qualquer empresa do seu setor, e o que só tem valor dentro da sua empresa atual — o sistema interno que só ela usa, a planilha que só ela tem, o jeito específico como ela nomeia as coisas, o relacionamento com aquelas pessoas.

Se a segunda parte for muito maior que a primeira, você não tem exatamente um emprego. Tem uma cadeira — e cadeiras, como eu disse, são retiradas. O conhecimento específico da casa é valioso *lá dentro* e vale zero no dia seguinte à demissão, o que significa que quanto mais a sua competência é feita dele, menos poder você tem em qualquer conversa, inclusive a de aumento.

A boa notícia é que dá para virar essa proporção sem sair do lugar e sem deslealdade nenhuma. Toda vez que você aprende algo transferível — uma linguagem, um formato aberto, um padrão do setor, um método — você aumenta a primeira parte. Toda vez que resolve um problema difícil e genérico em vez de mais um problema específico da casa, aumenta a primeira parte. E o mais bonito disso: **a empresa também ganha.** Um profissional com competência transferível é melhor profissional, não pior. A diferença é que ele está ali por escolha, e um bom gestor prefere gente que escolheu ficar a gente que não tinha para onde ir.

O plano de quem fica

Fecho com os cinco passos, na ordem, para quem não vai largar o emprego.

Primeiro: mova as trancas que não dependem de ninguém. Formatos neutros, acervo próprio do seu trabalho, aprendizado. Isso começa hoje e não passa por autorização.

Segundo: descubra a interface de programação do software que a empresa já paga. É a maior oportunidade destravada da sua vida profissional e está a um clique de distância, sem custo e sem pedido.

Terceiro: cronometre a tarefa mais chata do time por duas semanas. Sem

número antes, não existe conversa depois.

Quarto: proponha um piloto pequeno, com resultado, custo zero e saída limpa. Peça resultado, não ferramenta.

Quinto: construa o que é seu, fora, limpo. Seu tempo, sua máquina, seus dados. Se um dia isso virar alguma coisa, você não vai querer descobrir tarde demais de quem ela é.

Nenhum desses passos exige coragem heroica, pedido de demissão ou conflito com ninguém. Exige método e uns poucos meses. E, ao fim deles, você continua no mesmo emprego — só que agora ficou porque quis.

O que vem a seguir

Mostrei, no capítulo doze, um caminho de autonomia por fora, e neste interlúdio o caminho de dentro. Os dois são legítimos, os dois usam exatamente as mesmas ferramentas, e o que muda é o terreno em que você as aplica — não o princípio.

Mas até aqui, nesta quarta parte, você só tem a minha palavra e o meu exemplo, que é uma aposta em andamento e ainda não provada. Seria fraco encerrar a parte prática do livro pedindo que você confiasse só nisso. Por isso o próximo capítulo sai de mim e vai ao mundo, atrás de gente que já está construindo fora da gaiola há tempo suficiente para provar que funciona: a comunidade que mantém viva uma ferramenta de CAD livre usada por milhões, os fabricantes independentes que vendem produtos impressos em 3D sem pedir licença a plataforma nenhuma, os pequenos estúdios que hoje fazem jogos com ferramentas que uma década atrás só uma grande produtora poderia pagar. Não a minha aposta — as provas que já estão de pé, com os números na mesa.

Para levar deste capítulo

Trancas 3 e 6 — Capacidade e Domínio. As duas que você move sem pedir licença a ninguém, e que continuam sendo suas no dia em que o crachá deixar de funcionar.

Faça agora — 15 minutos. Faça a separação incômoda: liste o que você sabe fazer que funcionaria em qualquer empresa do setor, e o que só tem valor onde você está hoje. Compare o tamanho das duas listas. A proporção

entre elas é o seu poder real de negociação — inclusive na próxima conversa sobre salário.

Fontes: Lei nº 9.609/98 (Planalto); jurisprudência trabalhista e do TJ-MG sobre titularidade de software desenvolvido por empregado; Verizon DBIR 2026; PagerDuty e BlackFog (2026); Salesforce Workforce AI Survey 2026; ISACA (2026); Productiv (2026); Microsoft (2026). Os dados de uso não sancionado de IA são estimativas de pesquisa, com variação relevante entre estudos — a faixa relatada em 2026 vai de 45% a 66% da força de trabalho.

Capítulo 13 — Estudos de Caso: Quem Já Está Construindo Fora da Gaiola

No capítulo anterior, usei o hub que projetei como exemplo — e admiti a sua fraqueza: é uma aposta que ainda nem saiu do papel. Seria pobre encerrar a parte prática deste livro pedindo que você confiasse apenas no meu experimento em construção.

Então agora eu saio de mim. Este capítulo é uma coleção de provas — casos reais, verificáveis, de gente que já está construindo fora da gaiola e vivendo disso. Escolhi três, de três mundos diferentes, porque o que me interessa não é nenhum deles em particular, mas o padrão que se repete nos três. Em cada caso, você vai ver a mesma equação: ferramentas abertas ou acessíveis, mais um indivíduo ou um pequeno grupo, mais a queda de uma barreira que antes exigia uma corporação inteira — e o resultado é um sustento independente e real. Não é utopia. Tem risco, tem fracasso, tem custo. Mas está acontecendo, agora, com gente de carne e osso.

A ferramenta que sobrevive aos próprios donos

Começo pelo mundo que conheço melhor, o do CAD, com o caso que fecha um arco que abri lá no quinto capítulo: o FreeCAD.

Contei antes que o FreeCAD havia chegado, no fim de 2024, à sua versão 1.0, depois de uma maratona de mais de vinte anos. O que importa agora é o que aconteceu *depois*. Em 2026, o FreeCAD deixou de ser aquele programa "promissor, mas quebrado" e virou o que os próprios analistas chamam de uma pedra angular do movimento de engenharia aberta — uma ferramenta madura, de nível profissional, usada de verdade em arquitetura, engenharia, manufatura e design de produto.[83] Mas o dado que mais me

[83] Avaliações de 2026 do FreeCAD como plataforma madura, de nível profissional, usada em arquitetura, engenharia, manufatura e design de produto (thecadhub.com, "The Best 3D CAD Software of 2026"; avaliações de usuários em Capterra e G2, 2026). A versão 1.0 estável foi lançada em 18 de novembro de 2024, após mais de vinte anos de desenvolvimento.

importa não é técnico; é jurídico e humano. O FreeCAD é publicado sob uma licença livre que garante, não por promessa comercial, mas por lei, que ele pertence à comunidade. Não há mecanismo para trancar recursos atrás de um paywall. Você é dono dos seus dados. E, esta é a frase que resume tudo, se os desenvolvedores desaparecerem amanhã, o programa continua rodando na sua máquina.[84]

Lembra do caso Ondsel, que contei no quinto capítulo? A empresa que financiou a correção de um problema histórico do FreeCAD e depois fechou as portas? Aqui está o desfecho, e ele é a prova viva da tese deste livro. A empresa morreu; a ferramenta não. As melhorias que a Ondsel pagou para construir continuam lá, dentro do FreeCAD, servindo a todo mundo, porque estavam num bem comum que ninguém pode fechar. No mundo do jardim murado, a morte da empresa é a morte da ferramenta. No mundo aberto, a ferramenta sobrevive aos seus próprios donos. E ela se mantém viva por um ecossistema concreto, não por magia: uma associação sem fins lucrativos que distribui verbas trimestrais para o desenvolvimento, competições de projeto da comunidade, um fluxo constante de melhorias publicadas à vista de todos.[85]

Preciso ser honesto, como fui o livro inteiro: o FreeCAD tem uma curva de aprendizado íngreme, uma interface que muitos acham difícil, e "gratuito" também significa "sem suporte garantido" — se algo quebra, a comunidade ajuda, mas ninguém é obrigado a atender o seu chamado. É o preço da liberdade, o mesmo de sempre. Mas é um preço que milhares de profissionais estão escolhendo pagar, de olhos abertos, em troca de algo que nenhuma assinatura oferece: uma ferramenta que é, de fato, deles.

O estúdio de uma pessoa só

Do CAD, salto para um mundo que me é caro por outra razão, o dos jogos, porque é ali que a democratização das ferramentas produziu as histórias mais

[84]O FreeCAD é licenciado sob a LGPL, que garante juridicamente seu caráter livre: pertence à comunidade, não há mecanismo de paywall no núcleo, o usuário é dono dos próprios dados, o uso comercial é permitido sem royalties, e o software continua funcionando localmente mesmo que o desenvolvimento cesse (viz-cad.com, "FreeCAD vs. AutoCAD", 2026). A contrapartida honesta: "gratuito" também significa sem suporte garantido.

[85]A FreeCAD Project Association (FPA) mantém um programa de verbas (grants) com orçamento trimestral (na ordem de milhares de euros por ciclo) e a comunidade realiza competições de projeto e publica melhorias continuamente (blog.freecad.org, 2026). O contraste com o caso Ondsel — empresa que financiou a correção do problema de nomenclatura topológica e encerrou as atividades, com suas melhorias permanecendo no FreeCAD — foi tratado no Capítulo 5.

espetaculares.

Durante décadas, desenvolver um jogo com qualidade profissional exigia o que só uma grande produtora tinha: motores gráficos caríssimos, equipes enormes, orçamentos de milhões. Essa muralha ruiu. Motores como o Godot (que é livre e de código aberto, gratuito para qualquer um) derrubaram a barreira que mantinha o desenvolvedor solitário à margem.[86] E o resultado aparece nos números e nas histórias. Em 2026, os jogos independentes capturam cerca de um quarto de toda a receita da maior loja de jogos para computador do mundo; títulos construídos por equipes de menos de dez pessoas competem de igual para igual com produções que custaram cem vezes mais.[87]

O caso que virou símbolo dessa mudança é o de um jogo chamado Balatro, criado essencialmente por *uma única pessoa*, que transformou uma ideia simples num fenômeno de milhões de jogadores, dominando as paradas de venda e os prêmios da indústria.[88] Um desenvolvedor. Uma ideia. Ferramentas acessíveis. E um alcance global que, vinte anos atrás, teria sido impensável sem uma editora poderosa por trás. Não é um caso isolado: é a ponta visível de um movimento em que motores gratuitos, lojas digitais que levam o seu jogo ao mundo inteiro, e modelos como o de royalties (em que você só paga se o jogo der certo) inverteram a lógica de quem pode criar. É a mesma direção para a qual eu próprio olho, no motor S&box, quando penso nos meus projetos de jogo.

E, de novo, a honestidade que este livro se impôs: a queda da barreira democratizou a *largada*, não a *chegada*. Cerca de 70% dos jogos independentes fracassam comercialmente; a facilidade de criar trouxe uma enxurrada de concorrência, e o que separa o sucesso do esquecimento deixou de ser o acesso à ferramenta e passou a ser a visão, a originalidade, a

[86] O motor Godot (livre e de código aberto), em sua quarta geração, é apontado como um dos principais agentes da democratização do desenvolvimento de jogos, tornando o desenvolvimento de nível profissional acessível independentemente de recursos financeiros (catch-and-shoot.com, 2026; orbitalaffairs.com, 2026).

[87] Dados do mercado indie 2026: jogos independentes capturam cerca de 25% da receita da Steam; muitos dos títulos mais promissores são feitos por equipes de menos de dez pessoas; o mercado indie foi avaliado em US$ 5,54 bilhões em 2026, crescendo a 14,54% ao ano (fungies.io, “Indie Developer Market 2026”).

[88] O jogo Balatro, desenvolvido essencialmente por um único criador, tornou-se um fenômeno de vendas e crítica em 2024-2025, frequentemente citado como o exemplo emblemático do poder das ferramentas acessíveis nas mãos de um desenvolvedor solo (catch-and-shoot.com, 2026).

execução.[89] A liberdade não garante o resultado. Ela só garante que a porta, antes trancada, agora está aberta — e que quem entra depende do próprio talento, não da permissão de um guardião.

A fábrica na sua garagem

O terceiro caso é o mais concreto de todos, porque envolve objetos físicos saindo de máquinas que cabem numa mesa: os makers da impressão 3D.

O barateamento das impressoras e dos materiais criou uma categoria inteira de gente que fabrica e vende produtos físicos a partir de casa, sem fábrica, sem estoque, sem pedir licença a ninguém. Os números são modestos e reais ao mesmo tempo: operadores em tempo parcial faturam tipicamente de algumas centenas a alguns milhares de dólares por mês, e os que levam a sério, em tempo integral e com um nicho bem escolhido, chegam a faixas de cinco a dez mil dólares mensais depois de um ano ou dois de trabalho consistente.[90] As margens de produtos personalizados são altas, o investimento inicial cabe numa única máquina de entrada, e a produção é sob demanda — você imprime quando vende, sem capital preso em estoque.[91]

E há aqui um detalhe que fecha, com precisão poética, um círculo que abri no quinto capítulo. Uma das categorias mais estáveis desse mercado é a "economia do conserto": a fabricação de peças de reposição que os fabricantes não vendem mais — a engrenagem do eletrodoméstico fora de linha, a peça do carro antigo, o suporte descontinuado. Lembra do trator que você compra mas não pode consertar, do direito de reparar que discuti lá atrás? A impressora 3D na garagem é a resposta individual, concreta e

[89]A ressalva honesta: estimativas de 2026 apontam que cerca de 70% dos jogos independentes fracassam comercialmente; a democratização das ferramentas aumentou a concorrência, deslocando o diferencial da capacidade técnica para a visão, a originalidade e a execução (fungies.io, 2026).

[90]Faixas de faturamento de negócios de impressão 3D em 2026: operadores em tempo parcial tipicamente US$ 500–3.000/mês; operadores em tempo integral com nicho focado US$ 5.000–10.000+/mês após 12–18 meses de esforço consistente (alidropship.com; dropship.me; insightagent.app, 2026). Exemplos reais citados incluem um vendedor de acessórios para pets faturando mais de US$ 200 mil/ano e vendedores de miniaturas na casa das dezenas de milhares por ano.

[91]Margens brutas de produtos 3D personalizados citadas entre 40% e 85%, com baixo custo de material e produção sob demanda, sem estoque; investimento inicial acessível (uma única impressora de entrada). O ecossistema maker é expressivo: um único fabricante de impressoras relatou dezenas de milhares de usuários intensivos e milhões de acessos mensais à sua plataforma comunitária (shopify.com; pea3d.com, 2026).

imediata a esse problema: quando o fabricante se recusa a lhe vender a peça, você simplesmente a fabrica. O direito de reparar, que como lei depende de tribunais e de legisladores, como *tecnologia* já está na sua mesa.[92]

Mas — e aqui está a lição mais sofisticada deste capítulo — nem esse mundo é livre de armadilhas, e a mais importante delas é uma velha conhecida com roupa nova. Muitos desses makers vendem através de grandes marketplaces, e há uma verdade dura que os mais experientes repetem: os seus clientes, nessas plataformas, não são seus — são da plataforma. O algoritmo que hoje mostra os seus produtos pode escondê-los amanhã; a taxa que hoje é justa pode subir; as regras podem mudar sem aviso. Percebe o que aconteceu? O maker que escapou do jardim murado do software foi, sem perceber, se instalar dentro de um novo jardim murado: o do marketplace. Por isso os que constroem com solidez fazem questão de, em paralelo, cultivar o próprio público, a própria lista, o próprio canal — para que a plataforma seja um caminho, e nunca uma coleira.[93] A lição do livro inteiro se repete, um nível abaixo: a liberdade nunca é um destino ao qual se chega e descansa. É uma vigilância que não termina.

O padrão por trás dos três

Recuo agora para enxergar os três casos de uma vez, porque o que importa é o que eles têm em comum.

Uma ferramenta de engenharia que pertence à comunidade e sobrevive à morte de quem a financia. Um motor de jogo gratuito que põe um desenvolvedor solitário no mesmo tabuleiro das gigantes. Uma impressora de mesa que devolve ao indivíduo o poder de fabricar. Três mundos distantes, o mesmo esqueleto: uma ferramenta aberta ou acessível, mais uma pessoa ou um pequeno grupo disposto a assumir o risco, mais a queda de uma barreira que antes só uma corporação transpunha. É a tese deste livro

[92] A "economia do conserto" (fabricação sob demanda de peças de reposição obsoletas ou descontinuadas) é apontada como um dos modelos mais estáveis do setor em 2026, e conecta-se diretamente ao movimento pelo direito de reparar discutido no Capítulo 5 (shopify.com, "20 Profitable 3D Printer Business Ideas"; pea3d.com, 2026).

[93] A advertência de que, em marketplaces, "os seus clientes são da plataforma, não seus" — e a recomendação de cultivar público e canais próprios em paralelo para não trocar uma dependência por outra — aparece de forma recorrente entre operadores experientes (innocube3d.com, 2026). É a mesma dinâmica de lock-in tratada nos Capítulos 2 e 3, reencenada no varejo digital. Vale também a nota de responsabilidade, coerente com este livro: vender produtos impressos exige respeitar a propriedade intelectual alheia — usar designs próprios ou devidamente licenciados, não reproduzir personagens protegidos sem autorização.

inteiro, não como argumento meu, mas como fato já acontecendo na vida de milhares de pessoas.

E os três compartilham, também, a mesma honestidade dura. Nenhum é uma promessa de sucesso fácil. O FreeCAD não tem suporte; a maioria dos jogos indie fracassa; o maker leva um ano ou dois suando antes de viver disso, e ainda precisa se defender de virar refém do marketplace. A liberdade que os três oferecem não é a ausência de risco nem a garantia de vitória. É apenas — e é tudo — a devolução, para o indivíduo, do poder de tentar sem pedir licença. A porta está aberta. Atravessá-la, e o que fazer do outro lado, continua sendo com você.

O que vem a seguir

Estes casos provam uma coisa importante: dá para fazer. Não é fantasia de manifesto, não é teoria bonita — há gente vivendo disso agora. Mas provar que algo é possível não é o mesmo que mostrar como se faz, e eu não quero terminar este livro no terreno da inspiração, que é barato. Quero terminá-lo no terreno da prática, que é o que muda uma vida.

Por isso o último capítulo é o mais concreto de todos. Ele troca o "é possível" pelo "eis como se começa": um guia prático, ferramenta por ferramenta, para você montar a sua própria stack livre a partir de onde está hoje, com o que tem hoje. Menos manifesto, mais manual. Porque de nada adianta um livro inteiro defendendo a liberdade se ele deixar você na porta sem dizer qual é o primeiro passo do outro lado. Esse primeiro passo é o assunto do capítulo final.

Fontes: FreeCAD — thecadhub.com, viz-cad.com, blog.freecad.org, Capterra, G2 (2026). Jogos indie — fungies.io, catch-and-shoot.com, orbitalaffairs.com, monolithgaming.org, gamespace.com (2026). Impressão 3D/makers — alidropship.com, dropship.me, insightagent.app, shopify.com, pea3d.com, innocube3d.com, store.heygears.com (2026). Todos os dados de faturamento são faixas relatadas por operadores e devem ser lidos como indicativos, não garantidos.

Para levar deste capítulo

Trancas 5 e 6 — Renda e Domínio. Os três casos provam que a equação funciona: ferramenta acessível mais especialização real mais disposição para assumir o risco.

Faça agora — 15 minutos. Encontre uma pessoa que já vive do que você gostaria de fazer — num fórum, no LinkedIn, num grupo da sua área — e mande uma mensagem curta e específica perguntando uma coisa só. A distância entre você e o caminho que quer seguir quase sempre é menor do que parece.

Capítulo 14 — Guia Prático: Construindo Sua Própria Stack Livre

Prometi, no capítulo anterior, trocar o “é possível” pelo “eis como se começa”. Este capítulo cumpre a promessa. É o mais prático do livro, e de propósito o menos poético — porque de nada adianta um manifesto inteiro sobre liberdade que deixe você parado na porta, sem saber qual é o primeiro passo do outro lado.

Antes das ferramentas, o único princípio que você precisa levar consigo, porque ele resolve 90% das decisões sozinho. Não se trata de largar tudo o que você usa hoje e migrar de uma vez para um mundo puro e aberto — isso seria trocar um radicalismo por outro, e provavelmente quebrar a sua produtividade no processo. Trata-se de algo mais modesto e mais poderoso: **conquistar soberania aos poucos, mantendo sempre a porta de saída aberta.** A meta não é a pureza. É a autonomia e a opção. Um profissional que usa uma ferramenta fechada, mas mantém o seu trabalho em formatos que pode levar embora, é mais livre do que outro que usa a ferramenta mais aberta do mundo sem nunca ter pensado em como sairia dela.

E, diante de cada nova ferramenta que aparecer prometendo mudar a sua vida, aplique as três perguntas que ofereci ao longo do livro, porque elas são a sua bússola: eu consigo rodar isto, ou algo equivalente, sem pedir permissão? Se eu quiser sair, levo o meu trabalho comigo? E quem define o preço da minha dependência (eu, ou outra pessoa? Guardadas essas duas coisas) migração gradual e as três perguntas —, vamos à stack, camada por camada.

Camada 1: os formatos (comece por aqui, hoje)

Se você fizer só uma coisa deste livro inteiro, que seja esta, porque é a mais barata e a mais importante: **mantenha sempre uma cópia do seu trabalho em formatos abertos e portáteis.**

Não importa qual CAD você usa hoje. Ao terminar um projeto, exporte

também para os formatos neutros que qualquer ferramenta lê — STEP para os sólidos de engenharia, STL para impressão, e os formatos de intercâmbio que se firmaram como padrão para modelos e cenas. Isso não custa quase nada, não exige que você troque de software, não atrapalha o seu fluxo. E é, literalmente, a sua porta de saída construída de antemão. No dia em que você quiser ou precisar mudar de ferramenta — porque o preço subiu, porque a empresa mudou as regras, porque surgiu algo melhor —, o seu acervo inteiro estará esperando por você em formatos que ninguém controla. É a apólice de seguro mais barata que existe contra o aprisionamento. Comece por ela ainda esta semana.

Camada 2: as ferramentas de projeto

Sobre o CAD em si, vou ser honesto, do jeito que fui o livro inteiro: você provavelmente não vai, nem deve, abandonar da noite para o dia o software comercial que domina e que os seus clientes exigem. Se você projeta em Solid Edge, SOLIDWORKS ou NX e vive bem disso, jogar tudo fora seria burrice, não liberdade.

O que eu recomendo é diferente e mais sábio: **aprenda a alternativa livre em paralelo, para que ela exista como opção.** Instale o FreeCAD, que amadureceu para nível profissional, e aprenda o suficiente para saber que, se precisar, você consegue trabalhar sem pagar licença a ninguém. Use-o para os projetos pessoais, para os protótipos, para os trabalhos em que a licença cara não se justifica. Você não precisa que ele substitua a sua ferramenta principal amanhã. Precisa apenas que ele deixe de ser um mistério — para que a sua dependência da ferramenta cara seja uma escolha, e não uma prisão. Ter a alternativa dominada é o que muda a relação de força, mesmo que você nunca precise usá-la.

Camada 3: a automação

Aqui está o salto que separa o usuário passivo do engenheiro livre, e é mais acessível do que nunca — eu que o diga, que levei quinze anos travado e destravei em meses quando a ferramenta certa apareceu.

Escolha uma tarefa repetitiva que consome o seu tempo toda semana — uma requisição, uma lista de materiais, uma sequência de desenho, um relatório — e automatize-a. Você não precisa virar programador profissional para isso. Precisa de três coisas: a linguagem de automação da sua ferramenta (a maioria dos CADs sérios tem uma), um pouco de Python para colar as

peças, e a inteligência artificial como ponte, para conversar com o código em vez de decorá-lo. Comece pequeno, com algo que resolva uma dor real e concreta. A primeira automação que funciona muda a sua cabeça para sempre, porque ela prova, na prática, que você pode dobrar as ferramentas à sua vontade. E adote, desde o primeiro dia, um controle de versão como o Git para guardar o que você escreve — é a rede de segurança que deixa você experimentar sem medo de quebrar o que já funcionava.

Se esta é a sua tranca mais fechada e você nunca escreveu uma linha de código, não improvise o começo: o Apêndice deste livro traz o caminho completo em trinta dias, semana a semana, escrito para quem parte exatamente do zero.

Camada 4: a inteligência

Sobre a inteligência artificial, o princípio do sétimo capítulo vira prática: **prefira, sempre que possível, a inteligência que você controla.**

Para o trabalho do dia a dia, use as ferramentas que forem melhores — não há vergonha nem contradição em usar um bom modelo de ponta na nuvem. Mas, para os trabalhos sensíveis — a geometria confidencial de um cliente, os dados que não podem vazar —, aprenda a rodar um modelo aberto na sua própria máquina, com as ferramentas que tornaram isso simples. Assim o seu projeto nunca sai do seu computador. E se você for conectar a inteligência às suas ferramentas através de protocolos abertos, aplique a disciplina do oitavo capítulo, que é a mesma de qualquer bom administrador de sistemas: dê a cada agente só o acesso mínimo que ele precisa, exija a sua aprovação para qualquer ação que escreva, envie, apague ou gaste, e instale apenas conectores de fontes confiáveis. A liberdade de conectar tudo vem casada com o dever de trancar as portas certas.

Camada 5: a independência

Esta última camada só se aplica se você decidir seguir o caminho do capítulo doze — e, como eu disse lá, muita gente não deve, e está tudo bem. Mas, se você decidir, comece de onde está, com o que tem.

Não peça demissão amanhã. Monetize a sua habilidade lateralmente, mantendo a segurança enquanto constrói. Ofereça um serviço especializado, empacote uma ferramenta que você já criou, ensine o que você já sabe. Cresça com o próprio caixa, sem dívida e sem investidor, para não trocar

um patrão por outro. E lembre-se da lição mais fina do capítulo treze: se você vender através de uma plataforma, não deixe que os seus clientes sejam apenas dela — cultive o seu próprio público, o seu próprio canal, o seu próprio nome, para que a plataforma seja um caminho e nunca uma nova coleira. A independência se constrói tijolo por tijolo, e o primeiro tijolo é sempre pequeno.

Por onde começar, concretamente

Deixe-me fechar com um roteiro modesto, para que você não termine este livro com inspiração e nenhuma direção — porque inspiração sem primeiro passo evapora em uma semana.

Esta semana: comece a exportar todos os seus projetos também em formatos abertos. Só isso já constrói a sua porta de saída, e não custa nada. E escolha a tarefa repetitiva que mais te irrita para ser a primeira que você vai automatizar.

Este mês: instale o FreeCAD e faça um projeto simples nele, só para tirar o medo. Configure o Git. Rode, uma vez que seja, um modelo de inteligência artificial na sua própria máquina, para saber que é possível.

Este ano: construa uma automação que economize horas de verdade no seu trabalho — e sinta o que é dobrar a ferramenta à sua vontade. Se quiser ir além, monte a sua primeira fonte de renda independente, pequena, ao lado do seu trabalho principal, sem risco de tudo ou nada.

Nenhum desses passos é heroico. Nenhum exige que você largue a sua vida e aposte tudo. Essa é exatamente a questão: a liberdade que este livro defende não se conquista num salto dramático, mas numa sequência de passos pequenos, cada um deles ao seu alcance, cada um deles mantendo a porta de saída aberta. A stack livre não é um produto que você compra nem um lugar aonde você chega. É uma direção que você escolhe seguir, um pouco a cada dia.

O que vem a seguir

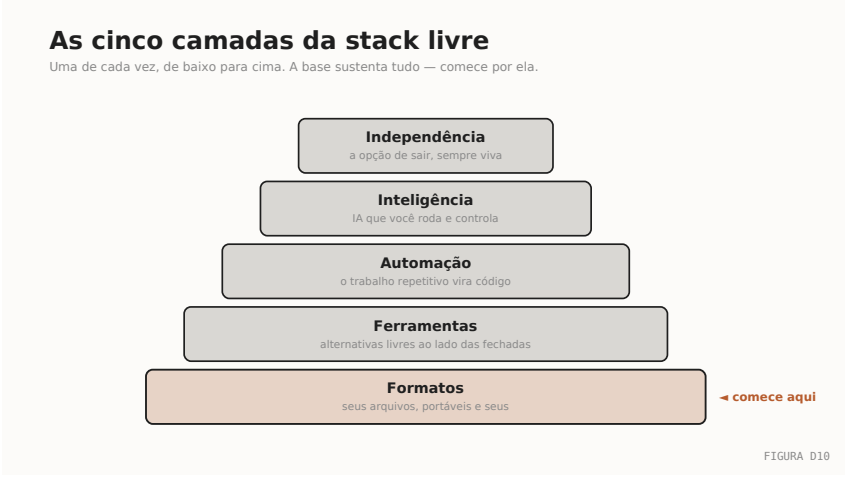
Chegamos ao fim do argumento. Diagnostiquei a gaiola, mostrei a saída e o seu preço, procurei os fundamentos filosóficos e a fé de onde eles vêm, exibi as provas de quem já vive fora dela, e agora entreguei o manual do primeiro passo. O trabalho do livro está feito.

Resta apenas uma coisa: um último olhar para trás, e um para a frente. Não mais um argumento, mas uma despedida — do engenheiro que começou este livro preso numa gaiola confortável, sem sequer saber que era uma gaiola, para o engenheiro que a essa altura, espero, já enxerga a porta. É o que fica para o epílogo.

Para levar deste capítulo

Todas as seis. Este capítulo é o mapa completo; as camadas correspondem, uma a uma, às trancas que você mediu no começo do livro.

Faça agora — o resto da sua vida profissional. Volte às Seis Trancas, refaça o teste e escolha as duas mais fechadas. Depois vá para a última seção do livro, “Antes de Fechar”, e monte o seu plano de noventa dias. Uma ação para cada tranca. Só duas. É assim que funciona.



Antes de Fechar — As Suas Trancas, Agora

Você anotou uma nota lá no começo, antes do primeiro capítulo. Vá buscá-la.

Volte às seis perguntas e responda de novo, com a mesma sinceridade de antes. Não demora cinco minutos.

Provavelmente a sua nota não mudou muito — e isso é esperado, porque ler não abre tranca nenhuma. O que mudou, se este livro funcionou, foi outra coisa: você agora *sabe quais são as suas*. Antes, a gaiola era uma vaga sensação de que algo não estava certo. Agora ela tem seis nomes, e você sabe quais deles são o seu problema. Diagnóstico não é cura, mas nenhuma cura começa sem ele.

O que resta é transformar isso em movimento. E movimento, para durar, precisa ser pequeno.

As suas duas trancas

Escreva, com todas as letras, num papel que você vá ver de novo:

As minhas duas trancas mais fechadas são a _____ e a _____.

Só essas duas. Eu insisti nisso no começo do livro e insisto de novo, porque é o erro que eu mesmo cometi: quem tenta abrir as seis ao mesmo tempo passa três semanas empolgado, se espalha, não conclui nada e conclui que "não deu certo". Não foi o método que falhou. Foi a dispersão.

O plano dos noventa dias

Para cada uma das suas duas trancas, escolha **uma única ação** dos noventa próximos dias. Não uma lista. Uma. Aqui vão sugestões concretas, uma por tranca — pegue a que corresponde à sua, ou use como modelo para inventar a sua própria.

Se a sua tranca é a dos dados: exporte todo o acervo do último ano para formatos neutros e guarde num lugar que seja seu. Depois, transforme isso em rotina de fim de projeto — algo que acontece toda vez, sem você precisar lembrar.

Se é a das ferramentas: instale a alternativa livre ao lado da que você usa e refaça nela, do começo ao fim, um projeto real que você já conhece de cor. Não para migrar. Para que ela deixe de ser um mistério.

Se é a da capacidade: escolha a tarefa mais repetitiva da sua semana e construa a automação que a elimina, com a ajuda de uma inteligência artificial. Não precisa ser elegante. Precisa funcionar e ser sua.

Se é a da inteligência: rode um modelo aberto na sua própria máquina, uma vez que seja, e use-o para resolver um problema de verdade. O objetivo não é abandonar a nuvem — é saber, na prática, que você tem para onde ir.

Se é a da renda: construa a sua primeira fonte independente, pequena, ao lado do que você já faz. Um serviço especializado, uma ferramenta que você empacota, uma aula que você dá. O valor não é o dinheiro. É deixar de ter uma única mão capaz de te derrubar.

Se é a do domínio: escolha *um* problema difícil da sua área, daqueles que a maioria dos colegas resolve mal, e passe noventa dias ficando bom nele de propósito, não por acaso. Especialização é a única muralha que um profissional pequeno consegue construir sozinho.

O único indicador que importa

Daqui a noventa dias, refaça o teste pela terceira vez.

Se as duas trancas que você escolheu tiverem descido um ponto cada, você fez mais pela sua autonomia profissional do que a maioria dos engenheiros faz em uma década inteira de carreira. Dois pontos em noventa dias parece pouco. Repita quatro vezes e você atravessou uma faixa inteira do diagnóstico — em um ano, sem largar o emprego, sem apostar tudo, sem heroísmo.

É assim que a coisa funciona. Não há salto dramático, não há dia da virada, não há revelação. Há uma tranca cedendo de cada vez, e um profissional que, a cada trimestre, depende um pouco menos de decisões que não são dele.

Anote a data. Marque no calendário o dia de refazer o teste. E vá abrir a primeira.

Epílogo — A Porta Estava Aberta

Este livro começou com duas cenas. Uma delas era grandiosa: um palco iluminado, em fevereiro de 2026, onde uma das maiores empresas de software de engenharia do mundo apresentava o futuro que havia escolhido para o engenheiro — assistentes de inteligência artificial de nomes amigáveis, uma plataforma que faz tudo, uma gaiola tão bem-acabada, tão confortável, que quase ninguém na plateia percebia que era uma gaiola. A outra cena era pequena e íntima: eu, em 2012, diante de uma tela de AutoCAD, desenhando ferramentas de metal duro, repetindo os mesmos comandos pela enésima vez, com uma voz teimosa no fundo da cabeça dizendo que devia existir um jeito melhor — e sem saber, ainda, que aquela voz levaria quinze anos para me tirar do lugar.

Entre aquelas duas cenas e esta página, percorremos um caminho. Vimos como a gaiola foi construída, ao longo de décadas de cercamento paciente. Contamos quanto custa viver dentro dela, em dinheiro e em algo mais caro que dinheiro. Encontramos a saída — os protocolos abertos, a bifurcação do CAD, a inteligência artificial que serve em vez de mandar — e fomos honestos sobre o pedágio dessa saída, sobre os riscos que ela traz e a responsabilidade que ela cobra. Descemos até a raiz filosófica de tudo, e eu não escondi de você a fé de onde ela brota, nem a linhagem que carrego e o legado que dela escolho levar. Vimos que o mercado, esse juiz imperfeito, vem premiando a liberdade. Conhecemos gente de carne e osso que já vive fora da gaiola. E, por fim, montamos o manual do primeiro passo.

Se há uma única coisa que eu gostaria que ficasse com você depois que este livro fechar, é esta: **a porta sempre esteve aberta.** A gaiola conforta, mas não tranca. O que nos mantém dentro dela quase nunca é uma corrente — é o hábito, o medo, a conveniência, a crença mansa de que não há alternativa, de que quem decide o que é bom para o engenheiro é sempre outra pessoa, lá em cima. Este livro inteiro foi um esforço para desfazer essa crença. Não há trono nenhum de onde alguém decide, com sabedoria superior, qual deve ser o seu fluxo de trabalho, qual ferramenta você merece, quanto vale a sua dependência. Há só você, as suas escolhas, e a responsabilidade, que é

também a dignidade, de fazê-las.

Não prometi, em nenhuma página, que a liberdade fosse fácil. Ela não é. É mais lenta que a comodidade, mais trabalhosa que a dependência, e ela devolve às suas costas um peso que a gaiola carregava por você. Um profissional livre revisa o próprio código, guarda a própria segurança, carrega o próprio risco, responde pelos próprios erros. Mas há uma diferença abissal entre o cansaço de quem carrega o peso da própria vida e o conforto anestesiado de quem entregou esse peso a outro em troca de parar de decidir. O primeiro é o cansaço de um homem de pé. O segundo é o descanso de quem se deitou numa cela e chamou aquilo de lar.

Eu escrevi este livro do meio da subida, não do alto. Não sou um caso encerrado de sucesso, não tenho uma fórmula para vender, e o meu próprio hub, os meus próprios projetos, as minhas próprias apostas ainda podem dar certo ou não. Escrevi como quem ainda está aprendendo, porque é o que sou. Mas de uma coisa eu tenho certeza, e é com ela que me despeço: o engenheiro que aprende a moldar as próprias ferramentas, a pensar com uma inteligência que ele controla, a construir o próprio lugar onde não havia nenhum, e a escolher com as próprias mãos qual legado carrega adiante — esse engenheiro não voltará para a gaiola. Ele pode até continuar usando algumas das ferramentas de dentro dela, por escolha, por conveniência, por ora. Mas nunca mais como prisioneiro. Sempre como alguém que sabe, agora, que a porta está aberta, e que a decisão de atravessá-la, todos os dias, é dele.

A máquina vai ficar cada vez melhor em fazer. Que você fique cada vez melhor em escolher. Porque no fim — depois de todos os formatos, kernels, protocolos e modelos —, esta foi sempre a única questão que importou: não o que as ferramentas podem fazer, mas quem você decide ser diante delas.

A porta está aberta. Ela sempre esteve.

O primeiro passo é seu.

Apêndice — Da Primeira Linha à Primeira Automação

Um caminho de trinta dias para o projetista que nunca programou.

Este apêndice existe porque o livro tem uma dívida. Eu passei catorze capítulos dizendo que a barreira caiu, que a inteligência artificial virou ponte, que você consegue construir as próprias ferramentas — e não mostrei **como se começa**. Sem isso, tudo o que escrevi vira inspiração, que é a coisa mais barata do mundo.

Então aqui está o caminho, na forma mais concreta que consigo dar. Ele é escrito para quem tem quarenta anos, quinze de profissão, nunca escreveu uma linha de código e tem meia hora por dia — no máximo. Não é para quem quer virar programador. É para quem quer que a máquina pare de fazer você repetir a mesma coisa.

As quatro regras que fazem isso funcionar

Antes do passo a passo, quatro princípios. Se você seguir só estes e ignorar o resto do apêndice, ainda assim vai chegar lá. Se ignorar estes e seguir o resto, vai desistir na segunda semana — eu sei porque foi o que aconteceu comigo em 2012.

Um: você não aprende a programar e depois automatiza. Você automatiza e aprende a programar como efeito colateral. A ordem importa e quase todo mundo erra. Curso primeiro é a rota mais confiável para a desistência, porque quarenta horas de teoria sem problema real na frente não sustentam a motivação de ninguém com trabalho e família.

Dois: escolha a tarefa mais irritante, não a mais complexa. A irritação é combustível, e você vai precisar dele quando der errado às onze da noite. A tarefa mais complexa é a mais interessante e a pior escolha possível para começar.

Três: feio e funcionando vence bonito e inacabado. Sua primeira automação vai ser feia. A minha era. Um programador profissional torceria

o nariz. Não importa: ela funciona, ela é sua, e ela te devolve tempo. Elegância é problema da terceira ou quarta.

Quatro: nunca rode código que você não sabe explicar. Esta é a regra de segurança e a regra de aprendizado ao mesmo tempo. Se a inteligência artificial te devolveu algo que você não entende, você não terminou — pergunte de novo. É a disciplina de verificação do oitavo capítulo aplicada à sua própria mesa.

Semana 0 — Escolher o alvo

Antes de instalar nada, escolha o que você vai automatizar. Metade dos fracassos acontece aqui.

Uma boa primeira automação tem cinco características:

- **É repetitiva.** Você faz pelo menos uma vez por semana.
- **Segue regra.** Você consegue explicar o procedimento inteiro para outra pessoa em cinco minutos, sem "aí depende".
- **É de baixo risco.** Se falhar, nada quebra. Idealmente ela **lê** muito e **escreve** pouco.
- **É sua.** Você é dono da tarefa. Não depende do fluxo de outra pessoa nem de permissão.
- **É mensurável.** Dá para cronometrar antes e depois.

Alvos bons para quem trabalha com projeto:

- Extrair a lista de materiais de uma montagem para planilha
- Gerar um relatório de massa, volume e material de um conjunto de peças
- Converter uma pasta inteira de arquivos para um formato neutro
- Renomear e organizar arquivos por uma regra que hoje você aplica na mão
- Montar um documento de conferência a partir de dados que já existem no modelo
- Extrair propriedades de um lote de peças para conferência

Alvos ruins para começar — deixe todos para depois: qualquer coisa que **escreva** em arquivos de produção; qualquer coisa que dependa do trabalho de outras pessoas; qualquer coisa que você não consiga explicar em cinco minutos; e qualquer coisa cujo erro só apareça três semanas depois.

Tarefa da semana: escolha um alvo e **cronometre-o**. Anote quanto tempo ele consome por semana. Sem esse número, você não terá como provar nada — nem para o seu chefe, nem para você mesmo no dia em que bater o desânimo.

Semana 1 — Preparar o terreno

Esta semana não é de programar. É de tirar os obstáculos do caminho.

Instale o Python e um editor de código — o Visual Studio Code serve e é gratuito. Se o seu CAD já tem uma linguagem de automação embutida, ela também serve, e talvez seja um caminho mais curto, porque já está instalada e licenciada.

E agora a parte que vai contra tudo o que você imagina: **aprenda exatamente três coisas, e pare.**

1. Como salvar um arquivo de código e executá-lo.
2. Como fazer o programa escrever algo na tela.
3. Como fazer o programa abrir um arquivo e ler o conteúdo.

Só isso. Não estude tipos de dados, não estude orientação a objetos, não faça o curso de quarenta horas. Você não precisa de nada disso para o que vem a seguir, e cada hora gasta em teoria antes do problema real é uma hora tirada da sua motivação.

Se conseguir fazer um programa escrever a frase “funcionou” na tela e depois ler um arquivo qualquer do seu computador, a semana 1 está cumprida. É pouco de propósito.

Semana 2 — A primeira conversa

Agora você vai usar a ponte. E existe um jeito certo e um jeito errado de conversar com uma inteligência artificial sobre código.

O jeito errado é pedir a coisa inteira de uma vez: *“faça um programa que extrai a lista de materiais do Solid Edge”*. Você vai receber cem linhas que parecem plausíveis, não vai entender nenhuma, provavelmente não vai funcionar, e você não terá ideia de por onde começar a consertar.

O jeito certo tem quatro elementos:

Descreva como descreveria a um colega novo. O que você tem, o que você quer, em que formato. “Tenho uma pasta com trinta arquivos de peça do Solid Edge. Quero uma planilha com nome do arquivo, material e massa de cada um.”

Dê um exemplo real. Mostre uma linha de entrada e a linha de saída que você espera. Exemplo vale mais que descrição.

Diga o seu contexto. Qual sistema, qual versão do CAD, o que você tem instalado, que você é iniciante. A resposta muda completamente.

Peça em pedaços. Comece por: *“primeiro, me mostre só como listar os arquivos dessa pasta”*. Depois: *“agora, como abrir um deles”*. Um passo de cada vez.

E o hábito que decide tudo: **quando vier código que você não entende, pergunte.** “Explique linha por linha o que isso faz.” “Por que essa linha é necessária?” “O que acontece se o arquivo não existir?”

Aqui está a coisa mais importante deste apêndice inteiro, e ela é contraintuitiva: **você não está aprendendo a escrever código. Está aprendendo a ler.** Ler vem primeiro, sempre. Quando você consegue ler e entender o que a máquina te devolveu, você já pode corrigir, adaptar e desconfiar — e desconfiar é a habilidade mais valiosa das três. Escrever do zero vem depois, quase sozinho, quase sem você perceber.

Semana 3 — Fazer funcionar uma vez

O objetivo desta semana é um único momento: a máquina fazer, sozinha, algo que você fazia na mão.

Para chegar lá rápido, faça tudo do jeito errado de propósito. Um arquivo só, não trinta. Caminho escrito direto no código, sem configuração. Nenhum tratamento de erro. Nada bonito. Você está buscando o momento, não o produto.

E prepare-se para dar errado. Vai dar. Várias vezes. Esta é a semana em que a maioria desiste, e desiste por um mal-entendido: acha que o erro é sinal de que não está dando certo. É o contrário — **o erro é o currículo.** É lendo mensagem de erro que você aprende como a coisa funciona por dentro.

Como ler um erro, na prática:

1. **Leia a última linha primeiro.** É onde está o que aconteceu, em geral em inglês, em geral bem direto.
2. **Ache o número da linha e o arquivo.** A mensagem diz exatamente onde estourou.
3. **Cole a mensagem inteira de volta para a IA**, junto com o trecho de código e o que você estava tentando fazer. Não resuma o erro — cole inteiro, inclusive a parte que parece lixo.
4. **Se a mesma correção falhar duas vezes**, pare de remendar. Peça para explicar a causa antes de propor solução nova. Duas correções que não funcionam quase sempre significam que o diagnóstico está errado.

Quando rodar, pare um instante e repare no que aconteceu. É um momento pequeno e desproporcionalmente importante: pela primeira vez, a ferramenta obedeceu a você em vez de o contrário.

Semana 4 — Fazer funcionar sempre

Uma automação que roda uma vez é uma demonstração. Uma que roda toda vez é uma ferramenta. A diferença é esta semana.

Três coisas a fazer, nesta ordem:

Generalize. De um arquivo para todos. De caminho fixo para caminho que você passa. De um caso para os casos que aparecem de verdade no seu trabalho.

Trate o que dá errado. O que acontece se o arquivo não existir? Se o campo estiver vazio? Se o CAD não estiver aberto? Você não precisa cobrir tudo — precisa que, quando falhar, a mensagem diga o que houve em vez de fechar sozinho.

Meça e anote. Cronometre de novo. Compare com o número da semana 0. Esse par de números é o que você vai usar para pedir tempo, para propor o próximo piloto, e para se convencer no dia em que a preguiça bater.

E, por último, guarde direito: salve num lugar que você vá achar daqui a seis meses e escreva cinco linhas dizendo o que a coisa faz e como se usa. Seu eu do futuro é um estranho, e ele não vai lembrar de nada.

O que acontece depois

Se você chegou aqui, a parte difícil já passou — e não pelo motivo que você imagina.

A segunda automação vai levar cerca de um terço do tempo da primeira. A terceira, menos ainda. Não porque você virou programador, mas porque você já sabe o formato da coisa: como conversar, como ler erro, o que perguntar, onde procurar.

E acontece uma mudança mais interessante, que é a que de fato transforma uma carreira: **você começa a enxergar o trabalho de outro jeito.** Onde antes havia "a chatice de terça-feira", passa a haver uma regra que dá para escrever. Você olha para um procedimento e vê a estrutura dele. Essa lente não desliga mais, e é ela — não o código — o ativo real que você acabou de construir.

Quando tiver três ou quatro automações, é hora de aprender uma coisa só a mais: controle de versão, o Git. Não antes. Aí ele resolve um problema que você já terá sentido na pele, que é não saber qual das cinco cópias do script é a que funciona.

Os sete erros que fazem desistir

Vale listar, porque são previsíveis e todos evitáveis:

1. **Começar por um curso em vez de um problema.** Motivação não sobrevive à teoria sem aplicação.
2. **Escolher a tarefa mais complexa** porque é a mais interessante. Escolha a mais chata.
3. **Querer que fique bonito na primeira.** Não vai ficar. Não precisa.
4. **Rodar código que não entende.** É perigoso e, pior, não ensina nada.
5. **Automatizar algo que escreve em arquivo de produção logo de cara.** Um erro aí custa caro e mata a sua coragem.
6. **Parar quando dá erro.** O erro é a aula, não o obstáculo.
7. **Fazer sozinho e em segredo.** Mostrar cedo transforma colegas em aliados — e, como vimos no Interlúdio III, muda a sua posição na empresa.

O que você não precisa aprender agora

Termino com uma lista de alívio, porque parte da paralisia vem de achar que é preciso saber tudo antes de começar. Você **não** precisa, neste momento, de nada disto:

orientação a objetos · padrões de projeto · estruturas de dados avançadas · algoritmos e complexidade · testes automatizados · banco de dados · arquitetura de software · nada de nuvem.

Tudo isso é útil. Tudo isso vem depois — e vem naturalmente, quando um problema seu exigir. Aprender antes da necessidade é a forma mais eficiente de esquecer.

E uma última honestidade, que é a mesma do sexto capítulo: isto não te torna um programador. Torna você **um projetista que automatiza o próprio trabalho** — que é outra coisa, é mais raro, e é exatamente o que o mercado de 2026 está pagando mais caro. Eu continuo do lado de cá dessa linha, ainda aprendendo, e ela foi suficiente para mudar a minha vida profissional inteira.

Trinta dias. Uma tarefa chata. Meia hora por dia.

É a Tranca 3, a da capacidade, e ela é a única das seis que ninguém pode abrir no seu lugar.

Sobre o autor

Carlos Eduardo Gonçalves Keese é projetista de moldes de injeção plástica desde 2016, com atuação em modelagem síncrona no Solid Edge e automação de processos técnicos através de programação própria. Autodidata em desenvolvimento de software desde 2012 (quando sua primeira ambição foi aprender Lisp para automatizar desenhos técnicos em ferramentaria), hoje concilia o trabalho de projetista com o estudo de Engenharia de Computação na Univesp. Cristão e libertário, escreve sobre a interseção entre liberdade individual, ferramentas abertas e o futuro da profissão de engenheiro na era da inteligência artificial.