AI IMPACT ACROSS THE SDLC · PLAYBOOK

A IA muda vinte e quatro papéis, não um, e horas economizadas não são dólares economizados.

Agentes de IA não mudam um papel. Mudam vinte e quatro, com intensidades diferentes em momentos diferentes, e é por isso que um roadmap único de IA para toda a engenharia é um erro de categoria. Este playbook mapeia o impacto papel a papel em um SDLC enterprise de 5.000 engenheiros, configura o GitHub Copilot para cada papel, sequencia a adoção em quatro ondas ao longo de doze meses e amarra cada hora recuperada a uma Ação de Conversão, porque tempo economizado não é dinheiro economizado até que uma etapa deliberada o transforme em redução de custo, crescimento de receita ou uma experiência melhor para o colaborador.

AUTORA

Paula Silva

CARGO

AI-Native Software Engineer

CONTATO

in/paulanunes

TEMPO DE LEITURA

~ 45 minutos, fim a fim

24
PERSONAS

6
CAPÍTULOS

4
ONDAS DE ADOÇÃO

12

MESES ATÉ ADOÇÃO PLENA

CAPÍTULOS

Chapter 00

Vinte e quatro papéis, não um roadmap

Por que um roadmap único de IA é um erro de categoria, por que uma persona é uma unidade de trabalho com resultado mensurável, por que tempo economizado não é dinheiro economizado, e o escopo: um SDLC enterprise de 5.000 engenheiros com papéis legados incluídos.

Contexto →

Chapter 01

O mapa de personas em quatro clusters

As 24 personas do SDLC agrupadas em Estratégia e descoberta, Construção e qualidade, Operação e confiabilidade, e Governar e crescer, cada uma com a capacidade do GitHub Copilot que precisa e o antes e depois que mede.

Personas →

Chapter 02

Configurando o GitHub Copilot por perfil

Os cinco níveis de customização, o agent mode com ancoragem via MCP, a escolha de modelo no seletor do Copilot por tarefa, e o platô do nível 1 ao nível 2 onde a maioria dos times empaca.

Método →

Chapter 03

Quatro ondas ao longo de doze meses

A linha do tempo de execução: redução de custo imediata, otimização de fornecedor e o parque legado, crescimento de receita via menor time-to-value, e depois resiliência e experiência do colaborador.

Adoção →

Chapter 04

Métricas guiadas por resultado e governança

As cinco Métricas Guiadas por Resultado que carregam o impacto, o controle de IA que move cada uma, a cadência como indicador antecedente de patrocínio, e a postura de governança que mantém tudo defensável.

Métricas →

Chapter 05

Transformando horas economizadas em dólares

Por que os ganhos vazam sem uma Ação de Conversão, as quatro alavancas que transformam horas em dinheiro, o custeio do programa em GitHub AI Credits, e o business case onda a onda.

Conversão →

DICAS DE LEITURA

> **ATALHOS DE TECLADO**
>
> Aperte `/` para buscar. Use `j` e `k` para mover entre capítulos. Aperte `t` para alternar tema. Aperte `g` para ir ao topo.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

00 CONTEXTO E TESE

Vinte e quatro papéis, não um roadmap

Agentes de IA não mudam um papel. Mudam vinte e quatro, com intensidades diferentes em momentos diferentes. Esse único fato quebra o plano enterprise mais comum, o roadmap único para toda a engenharia. Este capítulo define a moldura para o resto do playbook: por que um roadmap único é um erro de categoria, por que uma persona é a unidade certa de análise, por que tempo economizado não é dinheiro economizado, e o que um SDLC enterprise de 5.000 engenheiros realmente contém, papéis legados incluídos.

Por que um roadmap único de IA é um erro de categoria no SDLC

Agentes de IA não mudam um papel. Mudam vinte e quatro papéis, com intensidades diferentes em momentos diferentes. Um Product Manager, um Staff Engineer, um Site Reliability Engineer, um Engenheiro de Dados e um Compliance Officer enfrentam, cada um, um horizonte agêntico diferente. Cada um precisa de uma combinação distinta de GitHub Copilot, Microsoft 365 e Azure para chegar a valor. Tratar "IA para engenharia" como um roadmap único é um erro de categoria. Ele assume um ponto de partida uniforme, um formato de tarefa uniforme e uma curva de retorno uniforme. Nenhum deles se sustenta ao longo do ciclo de vida do software.

O erro é caro porque leva ao plano errado. Um roadmap único financia o caso de uso mais barulhento, em geral geração de código, e deixa sem recurso os papéis onde o retorno é maior e mais lento de aparecer. Um Analista de Requisitos que cruza transcrições com o conhecimento institucional, ou um desenvolvedor COBOL que transforma trinta anos de comportamento não documentado em documentação consultável, não aparece em um dashboard de autocompletar código. Quando uma métrica governa o programa inteiro, essas vitórias ficam invisíveis, e o programa

parece ter rendido menos do que deveria. A correção não é uma média melhor. A correção é mapear o impacto papel a papel, para que cada persona receba a ferramenta, a profundidade de customização e o prazo que o seu trabalho de fato exige.

Uma persona como unidade de trabalho, conjunto de decisões e resultado mensurável

Uma persona não é um cargo. É uma unidade de trabalho, um conjunto de decisões e um resultado mensurável. Essa definição é o que torna o framework operacional. Um cargo não diz nada sobre onde um agente ajuda. Uma unidade de trabalho diz quais tarefas o agente pode assumir, quais decisões ficam com o humano e qual resultado se mede no fim. Para o Desenvolvedor Moderno, a unidade é um branch de feature, as decisões são design e revisão, e o resultado é throughput. Para o Site Reliability Engineer, a unidade é um incidente, as decisões são diagnóstico e remediação, e o resultado é o tempo até recuperar.

O GitHub Copilot dá a cada persona um ponto de contato diferente. Alguns vivem em agent mode dentro do editor, delegando issues inteiras a um coding agent que abre um pull request com testes. Outros vivem no chat, fazendo perguntas a uma base de código que não escreveram. A hierarquia de customização é o que ajusta o agente ao papel: um .github/copilot-instructions.md válido para todo o repositório, arquivos de instrução por caminho, um AGENTS.md para o comportamento do projeto, custom agents para uma persona nomeada e MCP servers que conectam o agente aos sistemas que a persona já usa. O modelo no seletor do Copilot também é uma escolha por persona. Haiku 4.5 para edições rápidas e baratas; Opus 4.8 ou Claude Fable 5 para raciocínio difícil em muitos arquivos. Uma unidade de trabalho, um conjunto de decisões, um resultado, uma configuração.

A tese: tempo economizado não é dinheiro economizado

Ganhos de produtividade sozinhos não pagam pela IA. Essa é a tese que o resto do playbook defende. Um agente que economiza seis horas por semana de um desenvolvedor ainda não economizou nada para a empresa. O tempo economizado

vaza de volta para retrabalho, sobrecarga de validação, troca de contexto e reuniões, a menos que uma etapa deliberada o redirecione. A pesquisa do Gartner coloca o ponto sem rodeios: ganhos de produtividade, por si sós, não pagam pela IA. A distância entre horas recuperadas e dólares registrados é onde a maioria dos business cases de IA falha em silêncio, e é por isso que um CFO cético muitas vezes está certo em duvidar do número no slide.

A disciplina que fecha essa distância é uma Ação de Conversão. Cada persona precisa declarar como suas horas recuperadas viram um de três resultados: redução de custo, crescimento de receita ou melhoria da experiência do colaborador. Um Tech Lead que recupera tempo de revisão converte esse tempo entregando features mais cedo, não realizando mais reuniões. Há um custo do outro lado do balanço, e o GitHub Copilot o torna explícito. As execuções de agente são medidas contra os GitHub AI Credits, onde um crédito equivale a um centavo de dólar desde 1 de junho de 2026. Essa etiqueta de preço é o ponto. Ela força a pergunta honesta que a tese exige: esta execução produziu mais valor do que consumiu?

> DEFINIÇÃO
>
> Tempo economizado não é dinheiro economizado. Cada persona precisa declarar uma Ação de Conversão que transforme horas recuperadas em redução de custo, crescimento de receita ou melhoria da experiência do colaborador. Sem isso, as horas vazam de volta para retrabalho e o business case não fecha.

Escopo: um SDLC enterprise de 5.000 engenheiros, com papéis legados incluídos

O framework é escopado para uma empresa real, não para uma demo. A organização de referência é uma função de entrega de software de 5.000 engenheiros, organizada em oito fases do ciclo de vida do software, da gestão de clientes e contratos até governança, segurança e compliance. As vinte e quatro personas são os papéis que de fato compõem essa função. Não são arquétipos escolhidos para fazer a IA parecer boa. Elas incluem os papéis que os playbooks

genéricos ignoram. Empresas nessa escala são a norma, não a exceção: mais de noventa por cento da Fortune 100 usam o GitHub.

Papéis legados estão no escopo, não em volta dele

A maior parte das histórias de adoção de IA assume, em silêncio, código greenfield e stacks modernas. Uma empresa de 5.000 engenheiros não é assim. Ela roda décadas de sistemas COBOL, Natural/ADABAS, DB2 e mainframe, mantidos por desenvolvedores seniores que estão se aposentando e levando conhecimento não documentado com eles. O framework coloca três perfis de desenvolvedor legado no escopo de propósito. O GitHub Copilot suporta sintaxe COBOL e Natural, e um custom agent pode reduzir a compreensão de um programa de dias para horas. O resultado de maior valor é a captura permanente de conhecimento: o contexto que vivia na cabeça de um engenheiro vira documentação que a equipe inteira pode consultar. Um roadmap que cobre apenas os papéis modernos não é um roadmap para esta empresa. É um roadmap para a parte dela que já era fácil.

Referências

Fontes primárias para o framework das vinte e quatro personas, a evidência empírica de produtividade e as capacidades do GitHub Copilot que este playbook configura por papel.

- Agentic SDLC Personas, site do framework companion, a fonte original do framework de vinte e quatro personas sobre o qual este playbook se apoia.
- As vinte e quatro personas em detalhe, cada papel com suas tarefas, ferramentas e dados de antes e depois.
- GitHub Copilot Documentation, agent mode, o coding agent, os custom agents e os arquivos de instrução do repositório que este framework configura por persona.
- Customize AI responses in VS Code, arquivos de instrução, AGENTS.md, custom agents e agent skills, os cinco níveis de customização do Copilot.
- GitHub, Quantifying GitHub Copilot's impact on developer productivity and happiness, o resultado de 55% de velocidade em tarefas citado para as personas de desenvolvimento.

- GitHub, Quantifying GitHub Copilot's impact in the enterprise with Accenture, o estudo enterprise por trás do número de 88% de código retido.
- GitHub Universe 2024 press release, mais de 90% da Fortune 100 usam o GitHub, a escala que este framework mira.
- DORA 2025 State of DevOps research, desempenho de entrega, frequência de deploy e taxa de falha em mudanças.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

01 QUEM É AFETADO

O mapa de personas em quatro clusters: quem os agentes de IA transformam, e como.

Agentes de IA não mudam um papel. Mudam 24, com intensidades diferentes em momentos diferentes. Tratar IA para engenharia como um único roadmap é um erro de categoria, porque o Product Manager, o desenvolvedor COBOL, o Site Reliability Engineer e o Compliance Officer enfrentam horizontes agênticos distintos. Este capítulo agrupa as 24 personas em quatro clusters pelo tipo de trabalho que a IA muda: Estratégia e descoberta, Construção e qualidade, Operação e confiabilidade, e Governar e crescer.

O mapa: 24 personas, quatro clusters, uma unidade de trabalho

Uma persona é uma unidade de trabalho, um conjunto de decisões e um resultado mensurável. As 24 personas vêm de um SDLC enterprise real e incluem papéis legados como COBOL e Natural/ADABAS que os playbooks genéricos ignoram. Agrupá-las em quatro clusters responde a uma pergunta prática antes do detalhe papel a papel: que tipo de trabalho um agente de IA realmente muda, e quem é o dono dele. Cada cluster compartilha um padrão de impacto, um conjunto de ferramentas principais e um risco que decide se o ganho é real ou vaza.

A lente ao longo de todo o texto é o GitHub Copilot. O agent mode executa o trabalho de múltiplas etapas. Um arquivo .github/copilot-instructions.md e o AGENTS.md carregam as convenções que cada repositório já codifica, para que a saída gerada bata com os padrões corporativos em vez de adivinhá-los. Servidores MCP conectam o agente aos sistemas em que cada persona vive, do Azure Boards à base de conhecimento corporativa. A cobrança roda sobre GitHub AI Credits, onde um crédito equivale a um centavo de dólar, em vigor desde 1 de junho de 2026, então cada persona mapeia para uma linha de custo que um CFO consegue ler. O

seletor de modelos traz Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro e a família GPT-5.x, e o modelo certo é escolhido pela tarefa, não pelo papel.

Estratégia e descoberta: comprimir o overhead de reuniões e síntese

Este cluster reúne o Business Manager, o Project Manager, o Product Owner, o Requirements Engineer, o Enterprise e Solution Architect, o Tech Lead, o Scrum Master e o UAT Analyst. O gargalo comum não é código. É o overhead de reuniões e síntese. O padrão de impacto é compressão. Atas manuais que levavam de 30 a 40 minutos viram de 3 a 5 minutos de notas geradas pelo Copilot a partir de uma transcrição do Teams. A compilação semanal de status que levava de 4 a 5 horas vira um relatório assistido por agente de 30 a 60 minutos, extraído de Azure DevOps Boards, GitHub Projects e Power BI.

O ganho estratégico não são os minutos economizados. É para onde o tempo recuperado vai: análise de escopo, negociação de dependências e descoberta com stakeholders, o trabalho que só uma pessoa faz bem. Esse redirecionamento é também o risco. Se a realocação de capacidade não for imposta na cadência operacional, o tempo recuperado vaza direto de volta para mais reuniões de status. O tooling principal deste cluster é Microsoft 365 Copilot, Copilot Studio, GitHub Projects com Ask mode, Azure DevOps Boards e Power BI.

Construção e qualidade: de digitar código a especificar intenção

Este é o maior cluster: o Modern Software Developer, o QA Engineer, o Legacy Developer, o COBOL Developer, o Natural/ADABAS Developer, o Application Security Engineer, o Data Engineer e o ML Engineer. O padrão de impacto é uma mudança na unidade de trabalho, de digitar código para especificar intenção. O toque de tecla deixa de ser a unidade; a spec passa a ser. A personalização por arquivos de instrução, AGENTS.md, custom agents e agent skills faz o código gerado bater com os padrões corporativos na primeira tentativa, o que elimina a maior parte das iterações de review. A vazão do Modern Developer sobe de cerca de 40 para 80 ou 100 function points por mês, com cobertura de testes melhor do que antes.

Dois gates rígidos governam este cluster. A segurança de aplicação continua um gate obrigatório, não uma recomendação, e o ganho de produtividade precisa estar amarrado a uma Ação de Conversão declarada, senão vaza de volta para retrabalho em vez de chegar ao balanço. O tooling principal é o GitHub Copilot nos modos Ask, Edit, Agent, Plan e Coding Agent, o GitHub Copilot Code Review, o GitHub Advanced Security, o Microsoft Fabric e o Azure AI Foundry.

Captura de legado: COBOL e Natural/ADABAS

O resultado de maior valor neste cluster não é código novo. É captura de conhecimento. Desenvolvedores seniores que carregam de 20 a 30 anos de comportamento de sistema não documentado estão se aposentando, e esse contexto nunca foi escrito. O GitHub Copilot suporta a sintaxe COBOL e Natural nativamente. Um agente explicador reduz a compreensão de um programa de 2 a 5 dias para 2 a 4 horas, e o ganho de produtividade na geração de novo código chega a 85%. O resultado permanente é que o conhecimento que vivia em um único engenheiro vira documentação consultável que sobrevive à aposentadoria.

> DEFINIÇÃO
>
> O cluster de construção troca a unidade de trabalho, do toque de tecla para a spec. O maior valor não é a velocidade do código novo. É transformar o conhecimento de legado que estava na cabeça de uma pessoa em documentação que a organização consegue consultar.

Operação e confiabilidade: do combate reativo à engenharia proativa

Este cluster é o DevOps e Platform Engineer, o Database Administrator, o Release Manager e o Site Reliability Engineer. O padrão de impacto é sair do combate reativo a incêndios para a engenharia proativa. A triagem de incidentes cai de 15 a 30 minutos para 3 a 5 minutos com um agente comandante de incidente. Os post-mortems encolhem de 4 a 8 horas para 1 a 2 horas. As release notes vão de 3 a 5 horas por release para 30 a 60 minutos. O toil, o trabalho manual e repetitivo que um Site Reliability Engineer mede, sai de 40 a 50% da capacidade para menos de 30%.

O Site Reliability Engineer mostra os números mais expressivos de todo o framework. A correlação automatizada de alertas reduz a detecção de incidentes de 5 a 10 minutos para 30 segundos. O diagnóstico de causa raiz cai de 2 a 3 horas de investigação manual para 10 minutos de análise automatizada. A remediação por runbooks executáveis leva de 2 a 5 minutos contra os 30 minutos a 2 horas anteriores. A regra que governa mantém o humano no comando: a remediação automatizada passa por três execuções supervisionadas antes de ser promovida, e o agente permanece consultivo durante um incidente ao vivo enquanto o comandante humano mantém a autoridade. O tooling principal é Azure Monitor, Application Insights, Microsoft Sentinel e os ambientes de deployment do GitHub Actions.

Governar e crescer: evidência contínua substitui auditoria pontual

O último cluster é o Engineering Manager e IT Manager, o InfoSec e Compliance Officer, o Compliance e Audit Officer e o UX e Product Designer. O padrão de impacto é que a evidência contínua substitui a auditoria pontual. O Engineering Manager assume as métricas de adoção desde o primeiro dia, e é por isso que essa persona fica na primeira onda de adoção, não na última. O InfoSec sai da varredura no momento do pull request para a coleta contínua de compliance a cada push. Uma auditoria manual semestral que levava de 3 a 4 semanas vira um processo automatizado contínuo com trilha de evidências completa.

O designer também pertence aqui. A síntese de pesquisa de UX cai de 2 a 3 dias por estudo para 4 a 8 horas, e as auditorias de acessibilidade passam a ser contínuas em vez de anuais. Dois riscos governam este cluster. Recursos em public preview não podem rodar em produção para workloads críticos, e a indenização de IP precisa ser confirmada antes da adoção, via GitHub Copilot Enterprise e o Azure OpenAI Customer Copyright Commitment. O ledger de GitHub AI Credits ajuda aqui: como um crédito equivale a um centavo de dólar, a trilha de auditoria já carrega uma linha de custo para cada ação do agente. O tooling principal é GitHub Advanced Security, a GitHub Audit Log API, o Microsoft Purview, o Microsoft Sentinel, dashboards de compliance no Power BI e o Azure Policy.

DEFINIÇÃO

Quatro clusters, uma disciplina: medir o antes e o depois por persona, amarrar cada hora recuperada a um resultado declarado e manter um humano com autoridade onde o trabalho carrega risco. É isso que transforma estratégia de IA em algo que uma organização de 5.000 engenheiros consegue executar.

Referências

Fontes primárias para os padrões de impacto por persona, os benchmarks de produtividade e os mecanismos de personalização do GitHub Copilot citados neste capítulo.

- GitHub, Research: quantifying GitHub Copilot impact on developer productivity and happiness, a fonte do benchmark de 55% mais rápido na conclusão de tarefas.
- GitHub, Research: quantifying GitHub Copilot impact in the enterprise with Accenture, a fonte dos 88% do código sugerido mantido nas versões finais.
- GitHub Universe 2024, enterprise Copilot adoption, mais de 90% da Fortune 100 usa o GitHub.
- DORA, 2025 State of DevOps research, desempenho de entrega elite, frequência de deployment e taxa de falha em mudanças.
- GitHub Copilot documentation, agent mode, coding agent e custom agents.
- Customize AI responses in VS Code, arquivos de instruções, AGENTS.md, custom agents e agent skills.
- Agentic DevOps SDLC, 24 Personas, companion source, a fonte papel a papel para o mapa de quatro clusters.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

02 MÉTODO E ARQUITETURA

Configurando o GitHub Copilot por perfil: cinco níveis, um seletor, saída ancorada.

Um Copilot genérico ajuda um pouco cada perfil. Um Copilot configurado muda como o perfil trabalha. A diferença não é o modelo. É o contexto que você dá a ele e a forma como você o ancora. Este capítulo descreve os cinco níveis de customização, o agent mode com ancoragem via MCP, a escolha de modelo no seletor e o platô onde a maioria dos times empaca.

Os cinco níveis de customização

O GitHub Copilot é configurado em cinco níveis, cada um adicionando especificidade sobre o anterior. Os cinco funcionam com o Copilot no IDE, sem o GitHub Enterprise Cloud. O Nível 1 é o Copilot base: o seletor, a completação inline, o chat e o agent mode, sem conhecimento das suas convenções. É genérico por natureza. O Nível 2 é um arquivo .github/copilot-instructions.md na raiz do repositório. O Copilot o lê ao abrir a sessão e o aplica a toda requisição naquele repositório: regras de nomenclatura, restrições de segurança e as convenções corporativas que um recém-chegado precisaria. É o arquivo de maior retorno da hierarquia, e leva uma tarde para escrever.

O Nível 3 estreita o escopo. Arquivos de instrução com escopo por glob, .instructions.md com um padrão applyTo, carregam regras que se aplicam só aos arquivos correspondentes: *.java, *.sql, *.py. Uma regra de SQL não alcança um prompt de Java, e uma regra de Java não alcança um arquivo de teste. O Nível 4 são os custom agents, declarados em .agent.md, cada um um perfil especializado com suas próprias tools e handoffs. O agente @cobol-explainer, o agente @test-generator e o agente @code-reviewer passam trabalho entre si como handoffs formais. O Nível 5 são as Agent Skills, definidas em SKILL.md: workflows específicos por tarefa, sob demanda, com templates embutidos que o Copilot carrega só

quando a tarefa exige. Os custom agents estão em public preview; as Agent Skills são experimentais.

Mapeie o nível ao perfil

O nível de que um perfil precisa é definido pelo trabalho, não pela senioridade. Um Gestor de Negócio que quer atas de reunião precisa do Nível 1 e do M365 Copilot, nada mais. Um Desenvolvedor Moderno se beneficia das convenções do Nível 2 e das regras glob do Nível 3 para cada linguagem do stack. Um desenvolvedor COBOL precisa de um agente @cobol-explainer de Nível 4 para comprimir a compreensão de um programa de dias para horas. Um Site Reliability Engineer precisa de skills de Nível 5 que envolvam runbooks executáveis. Configure o nível que o perfil realmente vai usar e depois suba quando o teto for atingido.

Agent mode e ancoragem via MCP

Os arquivos de instrução dizem ao Copilot o que você prefere. A ancoragem diz ao Copilot o que é verdadeiro. O agent mode é onde os dois se combinam: o agente planeja uma mudança em vários arquivos, roda tools, lê resultados e itera, em vez de completar uma única linha. Por si só, o agent mode ainda infere fatos sobre a base de código. O Model Context Protocol fecha essa lacuna. Um servidor MCP expõe fatos determinísticos ao agente sob demanda: versões de dependências resolvidas, interfaces de serviço, definições de schema, contratos de APIs internas. O agente consulta o servidor em vez de chutar, e o código gerado atende ao padrão corporativo na primeira tentativa em vez da terceira.

O retorno se mede em retrabalho evitado. A pesquisa do GitHub constatou que 88% do código sugerido pelo Copilot foi mantido nas versões finais quando o modelo tinha o contexto de que precisava. A ancoragem eleva esse número ao remover a classe de erros que vem da inferência: um método depreciado, uma versão errada de dependência, uma fronteira de serviço que o agente não conseguia ver. Configure os servidores MCP uma vez no nível da plataforma, aponte o Copilot de cada perfil para eles, e cada perfil herda a mesma fonte de verdade. Os arquivos de instrução carregam preferência. Os servidores MCP carregam fato. O agent mode age sobre os dois.

DEFINIÇÃO

Os arquivos de instrução carregam preferência. Os servidores MCP carregam fato. O agent mode age sobre os dois. Preferência sem fato produz código confiante que viola uma restrição que o agente nunca viu. Ancore primeiro, depois instrua.

Escolhendo um modelo no seletor do Copilot

O seletor do Copilot expõe vários modelos, e o certo depende da tarefa, não de um ranking. Claude Fable 5 e Opus 4.8 são as opções de raciocínio pesado: decisões de arquitetura, compreensão de legado e refatorações em vários arquivos onde um plano errado sai caro. Sonnet 5 e os modelos GPT-5.x são as opções de codificação do dia a dia: carregam a maior parte do trabalho de features, o scaffolding de endpoints e a geração de testes a um custo menor por requisição. Haiku 4.5 é a opção rápida para iterações de alto volume e baixo risco: completações, edições pequenas e perguntas rápidas onde a latência importa mais que a profundidade. Gemini 3.1 Pro é a opção de contexto longo e multimodal, útil quando a tarefa abrange arquivos grandes ou inclui diagramas e capturas de tela.

A escolha de modelo tem uma dimensão de custo. Desde 1 de junho de 2026, o uso do Copilot é medido em GitHub AI Credits, onde um crédito equivale a um centavo de dólar. Modelos mais pesados consomem mais créditos por requisição, então casar o modelo com a tarefa é uma decisão de orçamento além de qualidade. O padrão que vale entre os perfis: use um modelo de tier médio como padrão para o trabalho de rotina, escale para um modelo pesado quando o plano importa e desça para um modelo rápido nas edições em massa. Um Site Reliability Engineer diagnosticando um incidente e um desenvolvedor gerando um endpoint CRUD não deveriam estar no mesmo modelo, e com o seletor eles não precisam estar.

Um seletor, muitos perfis

O seletor é por requisição, então um mesmo desenvolvedor transita entre modelos dentro de uma sessão. Peça ao Opus 4.8 para planejar uma refatoração, troque para o Sonnet 5 para implementá-la e desça para o Haiku 4.5 nas edições mecânicas. Os

arquivos de instrução e a ancoragem via MCP permanecem constantes na troca, então o contexto não reinicia quando o modelo muda. O modelo muda como o trabalho é raciocinado. A configuração o mantém correto.

Onde os times empacam: o platô do nível 1 ao nível 2

A maioria dos times não empaca no Nível 4 ou no Nível 5. Empaca entre o Nível 1 e o Nível 2. Instalam o Copilot, usam a completação inline e nunca escrevem o arquivo .github/copilot-instructions.md. A pesquisa do GitHub reporta 55% mais rapidez na conclusão de tarefas com o Copilot, e um time que atinge esse número pode acreditar que o trabalho está terminado. Não está. O arquivo que codificaria as convenções do time nunca é escrito, e cada sessão paga por essa ausência reinferindo as mesmas regras. O platô não é técnico. O passo de maior retorno simplesmente parece opcional, e não é.

Subir é uma sequência, não um salto. Escreva o copilot-instructions.md primeiro, porque é um arquivo só e se aplica a tudo. Adicione as instruções com escopo por glob em seguida, uma linguagem por vez, à medida que as convenções de cada stack se estabilizam. Introduza custom agents só onde um workflow repetido justifica um perfil nomeado: o @code-reviewer que recupera tempo de revisão, o @cobol-explainer que captura conhecimento em vias de se aposentar. Adicione as Agent Skills por último, para os workflows específicos por tarefa que um perfil roda com frequência suficiente para virar template. Cada nível deve merecer o próximo. Um time que sobe de forma deliberada chega a um ponto em que o Copilot se comporta como se pertencesse à base de código, porque pertence.

> DEFINIÇÃO
>
> O passo de maior retorno de toda a hierarquia é o mais barato de pular: escrever o primeiro arquivo copilot-instructions.md. Um time preso no Copilot base paga o preço cheio por uma fração do valor.

Referências

Fontes primárias para a hierarquia de customização, o agent mode e a evidência de produtividade.

- Customize AI responses in VS Code, a referência para arquivos de instrução, custom agents e Agent Skills.
- GitHub Copilot Documentation, a referência para agent mode, coding agent e custom agents.
- GitHub, Quantifying GitHub Copilot impact on developer productivity and happiness, a fonte do número de 55% mais rapidez na conclusão de tarefas.
- GitHub and Accenture, Quantifying GitHub Copilot impact in the enterprise, a fonte do número de 88% de código mantido.
- The 24 personas in detail, a matriz de configuração e resultado por perfil sobre a qual este capítulo se apoia.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

03 LINHA DO TEMPO DE EXECUÇÃO

Quatro ondas ao longo de doze meses: como os 24 perfis entram em operação.

O framework dos 24 perfis não é um lançamento, é uma sequência. A adoção corre em quatro ondas ao longo de doze meses, e cada onda acrescenta perfis cujo valor depende da infraestrutura que a onda anterior colocou de pé. Este capítulo percorre a linha do tempo onda a onda, nomeia os perfis de cada uma e amarra cada onda ao balde validado pelo Gartner em que seus retornos caem: redução de custo, crescimento de receita ou melhoria da experiência do colaborador.

Onda 1, ganho imediato: sete perfis, redução de custo

Meses 1 a 3. Sete perfis entregam retorno visível em semanas porque suas tarefas principais se encaixam no que o GitHub Copilot faz em agent mode de forma nativa: síntese de texto, geração de código e agregação de dados. Gestor de Negócio e Gestor de Projeto são os ganhos mais rápidos fora da engenharia. O Microsoft 365 Copilot transforma um processo manual de 30 minutos de elaboração de ata em extração automática de decisões, action items e responsáveis a partir de uma transcrição do Teams. O Desenvolvedor Moderno mostra os números mais expressivos. Um endpoint REST que leva de 4 a 6 horas de trabalho manual cai para 30 a 45 minutos com agent mode e geração de scaffold, e o Coding Agent acrescenta uma via paralela assíncrona em que o desenvolvedor delega de 2 a 3 issues por dia a um agente em background que abre um PR com testes. Os pontos de função por mês praticamente dobram, de cerca de 40 para uma faixa de 80 a 100, segundo o estudo Forrester Total Economic Impact.

QA e Test Lead reduz a geração de suítes de testes de 2 a 3 semanas para 3 a 5 dias, com um piso de cobertura de 75 por cento aplicado automaticamente. O Tech Lead recupera cerca de 60 por cento do tempo de revisão, porque o Copilot Code

Review cuida das verificações repetitivas e o lead fica com as decisões arquiteturais. DBA e Engineering Manager completam a onda. O Engineering Manager tem uma tarefa específica desde o primeiro dia: instrumentar a adoção. Uso diário acima de 70 por cento em 60 dias é o indicador preditivo de todo o programa, e o Manager liga a saída da GitHub Copilot Metrics API ao Power BI para que o feedback anedótico vire um business case com VPL e período de payback. A cobrança corre em GitHub AI Credits, um crédito valendo um centavo de dólar desde junho de 2026, o que torna o gasto de modelos no Copilot picker uma linha que o Manager acompanha contra as horas recuperadas.

Onda 2, expansão: oito perfis, otimização de fornecedor e o parque legado

Meses 4 a 6. Oito perfis entram em operação, e cada um exige mais configuração do que a Onda 1. MCP servers, um .github/copilot-instructions.md no repositório e arquivos AGENTS.md específicos por perfil entram antes que o Copilot atinja plena utilidade. O Analista de Requisitos ganha um agente que consulta a base de conhecimento corporativa no SharePoint, cruza transcrições com o conhecimento institucional e sinaliza conflitos antes que cheguem ao desenvolvimento. O Arquiteto Enterprise assume o núcleo da modernização: décadas de código COBOL, Natural/ADABAS, DB2 e mainframe que precisam eventualmente migrar para Spring Boot ou microsserviços. Um agente de exploração mapeia a base legada, produz diagramas C4 e escreve avaliações de viabilidade, comprimindo semanas de análise em dias.

Três perfis de desenvolvedor legado ficam no centro desta onda, e carregam o problema do conhecimento institucional. Desenvolvedores seniores com 20 a 30 anos de conhecimento de sistemas não documentado estão se aposentando. O Copilot lê a sintaxe COBOL e Natural de forma nativa, e um agente customizado @cobol-explainer reduz a compreensão de um programa de 2 a 5 dias para 2 a 4 horas. O resultado de maior valor é a captura permanente de conhecimento: o que vivia na cabeça de um único engenheiro vira documentação consultável. DevOps e Platform Architect e InfoSec fecham a onda. O balde de ROI aqui é redução de custo via otimização de fornecedor e outsourcing, e o indicador preditivo que o programa acompanha são três contratos renegociados com alavancagem de IA, porque um

parque legado que a equipe interna agora consegue ler e modificar muda os termos de toda conversa de outsourcing.

Onda 3, especialização: oito perfis, crescimento de receita via menor time-to-value

Meses 7 a 9. Oito perfis especializados entram em operação com agentes customizados de domínio que só funcionam depois que a infraestrutura de base das Ondas 1 e 2 está madura. O Product Owner gerencia backlogs de 500 ou mais itens com priorização por WSJF orientada por IA e geração automática de user stories. Scrum Master e Agile Coach automatizam retrospectivas, sprint reviews e mapas de dependência entre equipes. O Data Engineer fica na interseção de 50 ou mais fontes legadas e modernas alimentando o Microsoft Fabric para mil ou mais usuários de Power BI, e o tempo de criação de pipelines cai cerca de 80 por cento enquanto o monitoramento de qualidade de dados vira contínuo em vez de reativo.

ML e AI Engineer fecha o ciclo de feedback da plataforma: seleção de modelos no Copilot picker, otimização de prompts, frameworks de avaliação e governança de IA como detecção de alucinações e prevenção de vazamento de dados. Compliance e Audit Officer comprime uma auditoria manual semestral de 3 a 4 semanas em um processo contínuo com trilhas de evidências completas, resposta direta à LGPD, ISO 27001 e regulação setorial. UX e Product Designer gera protótipos no Figma em minutos e viabiliza validação de stakeholders no mesmo dia para fluxos que antes levavam semanas. Esse é o fio condutor da onda: cada perfil aqui encurta o time-to-value, e por isso o balde de ROI passa de custo para crescimento de receita, com a meta do programa sendo um time-to-value mediano menor em pelo menos 30 por cento.

Onda 4, maturidade: SRE, resiliência e experiência do colaborador

Meses 10 a 12. Um perfil atinge plena maturidade por último, o Site Reliability Engineer, e seus números são os mais expressivos do framework. A correlação automatizada de alertas reduz a detecção de incidentes de 5 a 10 minutos para 30 segundos. O diagnóstico de causa raiz cai de 2 a 3 horas de investigação manual

para cerca de 10 minutos de análise automatizada. A remediação via runbooks executáveis leva de 2 a 5 minutos contra os 30 minutos a 2 horas anteriores. Para sistemas onde o downtime custa milhões por minuto, esses não são ganhos marginais. Eles mudam como a confiabilidade é projetada.

O balde de ROI da Onda 4 é melhoria da experiência do colaborador e resiliência, o terceiro balde validado pelo Gartner, e o indicador preditivo é uma alta de eNPS de pelo menos 8 pontos entre usuários de IA. A qualidade de vida do on-call não é um benefício vago aqui. É o argumento de retenção que uma revisão de CFO aceita ao lado de custo e receita, e é o que segura os engenheiros seniores de cujo julgamento todo o programa depende.

O programa entra em refinamento contínuo

Ao chegar ao mês 12, o programa para de acrescentar perfis e começa a ajustar. Métricas DORA consolidadas, ROI reportado ao C-level e uma cultura de engenharia AI-first em todos os 24 perfis viram o estado permanente. Agentes customizados ficam mais afiados contra o uso real, arquivos copilot-instructions são corrigidos contra a saída real e integrações MCP são podadas. As ondas foram sequenciais e não opcionais. O refinamento é permanente.

> DEFINIÇÃO
>
> O modelo de quatro ondas é sequenciado, não simultâneo. A Onda 1 entrega redução de custo via compressão de experiência, a Onda 2 via otimização de fornecedor e outsourcing mais o parque legado, a Onda 3 converte menor time-to-value em crescimento de receita, e a Onda 4 converte resiliência e experiência do colaborador em um argumento de retenção. Após o mês 12, o programa entra em refinamento contínuo em todos os 24 perfis.

Referências

Fontes primárias para o modelo de adoção em quatro ondas, o catálogo de perfis e a evidência de produtividade por trás da linha do tempo.

- Agentic DevOps SDLC Personas, site complementar, o catálogo de origem dos 24 perfis e do roadmap de adoção em quatro ondas.
- GitHub Copilot Documentation, agent mode, o Coding Agent e os custom agents referenciados nas quatro ondas.
- Customize AI responses in VS Code, instructions files, AGENTS.md, custom agents e agent skills, as camadas de customização de que a Onda 2 depende.
- GitHub Research, quantifying GitHub Copilot impact on developer productivity and happiness, os 55 por cento de conclusão de tarefas mais rápida por trás dos ganhos de desenvolvedor da Onda 1.
- DORA, 2025 State of DevOps research, as métricas de entrega consolidadas quando o programa chega ao mês 12.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

04 MEDIR E GOVERNAR

Métricas guiadas por resultado e governança: o que medir, e como manter defensável.

Um piloto que não sobrevive a uma revisão de CFO não escala. Este capítulo transforma adoção em um business case defensável. Ele nomeia as cinco Métricas Guiadas por Resultado que carregam a maior parte do impacto por persona, pareia cada uma com o controle de IA que a move, e define a postura de governança que mantém o programa auditável em escala enterprise.

A matriz ODM: métrica, dono da alavanca e o resultado que o CFO acompanha

Conforme Gartner G00799085 e G00841077, uma Métrica Guiada por Resultado tem visão direta para um resultado de negócio. Cinco ODMs carregam a maior parte do impacto por persona, e cada uma nomeia duas coisas além do próprio número: o time que é dono da alavanca e o resultado que o CFO acompanha. Time-to-market pertence a Product, ao Tech Lead e ao Release Manager, e responde a crescimento de receita. Cycle time pertence a Developers, QA e DevOps, e responde a redução de custo e throughput. Defect escape rate pertence a QA, AppSec e SRE, e responde a evitar multa e NPS. Customer NPS pertence a Product, UX e Support, e responde a retenção e crescimento de ACV. Revenue per engineer pertence ao Engineering Manager com o parceiro de CFO, e responde a eficiência de headcount.

A coluna de propriedade não é decoração. Uma métrica sem dono se dispersa, e uma métrica sem resultado acompanhado pelo CFO nunca chega ao business case. A coluna de resultado é a camada de tradução entre engenharia e finanças. Revenue per engineer é a razão que sobrevive a uma revisão de CFO, porque é a única expressa nas unidades que o time de finanças usa. Ganhos de produtividade sozinhos não pagam pela IA, conforme Gartner G00837723. Toda persona precisa declarar uma Ação de Conversão que transforme horas recuperadas em um de três

baldes: redução de custo, crescimento de receita ou melhoria da experiência do colaborador. Sem essa etapa, o tempo vaza de volta para retrabalho e reuniões, e o CFO está certo em duvidar do caso.

Pareie cada métrica com o controle de IA que a move

Uma métrica sem controle é anedótica. Um controle sem métrica não tem defesa de negócio. Então cada ODM é pareada com o controle de IA que a move, o sinal de origem que a mede e a cadência em que é lida. Time-to-market se move com Spec Kit e agentes L4 customizados, lê de GitHub Releases e Azure DevOps Release, e é revisada semanalmente. Cycle time se move com Copilot Code Review rodando em todo PR, lê de GitHub Insights e Azure DevOps Analytics, e é revisada semanalmente. Defect escape rate se move com gates do GitHub Advanced Security mais um agente de QA, lê de Defender for Cloud, Sentinel e GHAS, e é revisada mensalmente. Customer NPS se move com um sintetizador de pesquisa UX mais um auditor de acessibilidade, lê de Microsoft Clarity e surveys no produto, e é revisada trimestralmente. Revenue per engineer se move com um tracker de adoção mais uma política de realocação de capacidade, lê da GitHub Copilot Metrics API e de relatórios financeiros, e é revisada trimestralmente.

O sinal de origem importa tanto quanto o controle. Toda métrica aqui lê de um sistema de registro, não de um autorrelato, e é isso que a faz sobreviver a uma revisão de steering. Adoção lê da GitHub Copilot Metrics API, não de uma survey. Custo é mensurável no mesmo quadro: uso de modelo em agent mode é cobrado em GitHub AI Credits, onde um crédito é um centavo, em vigor desde 1 de junho de 2026. Isso torna custo por resultado um número real. Divida créditos gastos por releases entregues ou defeitos capturados. Escolher um modelo mais barato no Copilot picker para uma tarefa de rotina e um mais forte para uma tarefa difícil vira uma alavanca que a métrica consegue precificar.

Disciplina de cadência é um indicador antecedente de patrocínio

Cadência não é formalidade de reporte. Ela é ajustada à velocidade com que cada métrica pode se mover. O Engineering Manager é dono das cadências semanal e mensal, que cobrem time-to-market, cycle time e defect escape rate, porque essas se movem com decisões de engenharia. O CTO é dono da cadência trimestral, que cobre Customer NPS e revenue per engineer, porque essas se movem com decisões de negócio. Ajustar a cadência à métrica mantém toda revisão acionável: uma métrica semanal lida trimestralmente está velha, e uma métrica trimestral lida semanalmente é ruído.

A cadência carrega valor diagnóstico além dos números que reporta. Uma cadência perdida sinaliza que o programa está perdendo patrocínio executivo antes que qualquer KPI o faça. Quando a revisão trimestral silenciosamente sai do calendário, o CTO parou de defender o investimento, e as métricas seguirão um ou dois trimestres depois. Cadência é mais barata de monitorar do que qualquer KPI, e falha mais cedo. Essa combinação é o que a torna o primeiro indicador a observar, à frente do dashboard que ela deveria alimentar.

> DEFINIÇÃO
>
> Cadência é o aviso mais precoce que você tem. Uma métrica pode cair por um trimestre inteiro antes que alguém aja. Uma cadência que é pulada te diz nesta semana que o patrocínio está escorregando. Observe o calendário antes de observar o dashboard.

Postura de governança: gates HITL, limites de public preview e indenização de IP

Três controles de governança mantêm o programa defensável em escala enterprise. O primeiro são os gates human-in-the-loop. Desconte das horas brutas economizadas o tempo de revisão que a postura de cada persona exige, tipicamente 15 a 35 por cento. Esse desconto não é um número para esconder. É a cifra honesta que o business case reporta, e uma persona com desconto maior não está falhando.

É uma persona cujo output carrega mais risco e corretamente merece mais escrutínio. O segundo controle são os limites de public preview. Os agentes L4 customizados estão em public preview e os L5 Agent Skills são experimentais, e recursos em public preview não podem ir para produção em workloads críticos. A regra é dita de forma clara para que um ganho de métrica nunca contrabandeie uma dependência sem suporte para um caminho regulado.

O terceiro controle é a indenização de IP. Confirme-a antes de escalar, via GitHub Copilot Enterprise e o Azure OpenAI Customer Copyright Commitment, para que o código gerado carregue um respaldo contratual. As trilhas de evidência completam a postura. GitHub Advanced Security, a Audit Log API, Microsoft Purview e Microsoft Sentinel transformam compliance de uma auditoria manual semestral em um processo automatizado contínuo com registro completo. Governança é um atributo por persona, não um interruptor único do programa. Um SRE rodando runbooks executáveis contra produção e um UX designer gerando protótipos descartáveis estão em níveis de risco diferentes e carregam gates diferentes. A postura é o que permite à mesma plataforma servir um workload regulado e um exploratório sem colapsá-los em uma única política.

Referências

Fontes primárias para o modelo de métricas, os controles de IA que movem cada métrica e a postura de governança.

- DORA 2025 Research, benchmarks de entrega de elite usados para enquadrar cycle time e taxa de falha em mudanças.
- GitHub Copilot Documentation, agent mode, coding agent, custom agents e a Copilot Metrics API usada como sinal de origem.
- Customize AI responses in VS Code, os níveis de personalização que definem os limites de preview dos níveis 4 e 5.
- Research: quantifying GitHub Copilot's impact on developer productivity and happiness, a evidência de velocidade de tarefa e tempo de PR por trás da métrica cycle time.
- Research: quantifying GitHub Copilot's impact in the enterprise with Accenture, retenção enterprise do código sugerido, evidência para o tracker de adoção.

- Microsoft Learn, Microsoft Purview, Microsoft Sentinel e Defender for DevOps para trilhas de evidência e governança.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

05 CONVERSÃO E FECHAMENTO

Transformando horas economizadas em dólares: a Ação de Conversão e suas quatro alavancas.

Um desenvolvedor que economiza seis horas por semana ainda não economizou dinheiro para a empresa. As horas são reais. Os dólares não são, até que uma Ação de Conversão deliberada redirecione o tempo recuperado. Este capítulo fecha o business case. Ele nomeia por que os ganhos vazam, as quatro alavancas que transformam horas em dinheiro, como custear o programa em GitHub AI Credits a um centavo cada, e como o business case onda a onda e a calculadora complementar reportam um único número defensável.

O vazamento: por que os ganhos de produtividade evaporam sem uma Ação de Conversão

Um desenvolvedor que economiza seis horas por semana, o número que a McKinsey reporta para desenvolvimento assistido por IA, ainda não economizou dinheiro para a empresa. As horas são reais. A economia não é, até que algo deliberado aconteça com o tempo recuperado. Essa é a lacuna que a maioria dos business cases de IA pula. A Gartner diz isso com todas as letras no relatório G00837723, Productivity Gains Alone Won't Pay for AI: o tempo economizado vaza de volta para retrabalho, overhead de validação, troca de contexto e capacidade não usada, a menos que uma Ação de Conversão o redirecione. O vazamento é o padrão. Deixadas por conta própria, as horas recuperadas se enchem de novo com reuniões, revisões mais longas e ociosidade. O gráfico de produtividade continua bonito. A demonstração financeira nunca se move.

O framework trata a conversão como campo obrigatório, não opcional. Cada uma das 24 personas deste playbook declara uma Ação de Conversão explicitamente, ao lado de sua unidade de trabalho e de sua postura de governança. A cadeia é escrita

como fórmula para sobreviver a uma revisão financeira. Tempo inicial economizado, menos o vazamento de produtividade, menos o valor não realizado, igual à produtividade real ganha. Esse número é multiplicado por uma Ação de Conversão para chegar ao ROI financeiro em dólares. Uma persona sem Ação de Conversão produz um número de produtividade e zero dólar. Um CFO tem razão em desacreditar qualquer business case que pare no número de produtividade.

A cadeia de conversão e suas quatro alavancas

Quatro alavancas transformam a produtividade real ganha em dinheiro. A primeira é o desconto HITL. As horas brutas economizadas nunca são líquidas, porque um humano ainda revisa a saída do agente. O tempo de revisão é definido pela postura de governança da persona e costuma ficar entre 15 e 35 por cento da economia bruta. Um agente explicador de COBOL sob revisão leve mantém a maior parte de suas horas. Um fluxo de AppSec atrás de um gate rígido mantém menos. Descontar primeiro mantém honesto o resto do cálculo.

A segunda alavanca é o reuso de capacidade. As horas recuperadas precisam cair em algo mensurável: um backlog de inovação, negociação de dependência, descoberta com cliente ou refinamento de escopo. Se o destino não pode ser nomeado, presume-se que as horas vazaram. A terceira alavanca é a redução do custo de erro. Um defeito pego no momento do pull request custa quase nada. O mesmo defeito pego em produção pode acionar uma multa contratual. Antecipar a detecção multiplica o valor de cada hora que o agente economizou. A quarta alavanca são os ganhos de throughput: mais releases por trimestre com o mesmo headcount, o que se lê como mais receita por engenheiro. Essa é a razão que uma revisão de CFO mantém. A pesquisa do GitHub coloca o sinal bruto em 55 por cento mais rápido na conclusão de tarefas e um tempo de ciclo de pull request cortado de 9,6 para 2,4 dias, e o estudo da Accenture reporta 88 por cento do código sugerido retido. Esses são insumos. As quatro alavancas os convertem.

Qual alavanca lidera, por onda

Nenhuma alavanca sozinha carrega o business case, e o mix muda ao longo do programa. A Wave 1 se apoia no reuso de capacidade. A Wave 2 acrescenta renegociação de fornecedor e de contrato. A Wave 3 acrescenta uplift de receita via

menor time-to-value. A Wave 4 acrescenta evitamento de risco e economia de retenção guiada por eNPS. Cada persona declara quais alavancas se aplicam a ela, então a dupla contagem fica visível e pode ser removida.

> DEFINIÇÃO
>
> As quatro alavancas, em ordem: descontar as horas brutas pela revisão humana, colocar a capacidade restante em algo mensurável, precificar os defeitos pegos cedo e contar os releases extras entregues com o mesmo headcount. Pule qualquer uma e o número em dólares fica superestimado.

Custeando o programa em GitHub AI Credits, um crédito equivale a um centavo

Um business case precisa de um lado de custo, e desde 1º de junho de 2026 o GitHub precifica o uso de agentes em AI Credits, onde um crédito equivale a um centavo. Isso transforma o lado de custo em aritmética, não em estimativa. Execuções em agent mode, tarefas do coding agent e chamadas de modelo consomem créditos do plano. Como um crédito é 0,01 dólar, um time lê seu consumo mensal de créditos na tela de billing e o expressa em dólares sem planilha. O denominador da razão de ROI fica tão defensável quanto o numerador.

O modelo escolhido no seletor do Copilot é uma alavanca de custo por si só. O seletor oferece Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro e os modelos GPT-5.x. Trabalho de rotina não precisa da opção mais cara. Uma execução com Haiku 4.5 ou Sonnet 5 custa menos créditos que uma com Opus 4.8 e muitas vezes basta para scaffolding, geração de testes ou uma primeira revisão. Os modelos mais pesados ficam reservados às tarefas que justificam seus créditos. A higiene de contexto também reduz o custo: um repositório com um arquivo .github/copilot-instructions.md mantido e os servidores MCP certos conectados gasta menos créditos por tarefa, porque o agente para de rederivar um contexto que poderia ter lido uma vez. Menos reconstrução significa menor consumo de créditos, o que melhora o lado de custo de cada persona.

O business case onda a onda e a calculadora complementar por mix de personas

O programa é custeado e convertido onda a onda, porque as alavancas e os patrocinadores mudam a cada onda. A Wave 1 cobre sete personas nos meses um a três, converte principalmente por realocação de capacidade e é defendida por uso diário acima de 70 por cento em 60 dias. A Wave 2 cobre oito personas nos meses quatro a seis e acrescenta renegociação de fornecedor e outsourcing. A Wave 3 cobre oito personas nos meses sete a nove e desloca o balde primário para crescimento de receita via menor time-to-value. A Wave 4 traz o Site Reliability Engineer nos meses dez a doze e converte por evitamento de risco e retenção guiada por eNPS. Cada onda nomeia um patrocinador executivo e um indicador preditivo, então o business case é auditável antes de a próxima onda começar.

A calculadora complementar combina tudo isso por mix de personas. Informe a contagem de cada persona, a onda, a postura de governança que define o desconto HITL e as Ações de Conversão escolhidas, e ela devolve um número em dólares que já desconta o vazamento. Mude o mix e o número muda junto. Ela roda sobre o mesmo modelo publicado de métricas guiadas por resultado do SDLC que ancora as 24 personas, então a calculadora e o playbook não podem divergir. A saída é um único número sobre o qual um comitê de steering pode agir.

> DEFINIÇÃO
>
> Horas economizadas são um insumo, não um resultado. O resultado é o número que sobrevive a uma revisão de steering: produtividade real ganha, descontada pela revisão humana, multiplicada por uma Ação de Conversão nomeada, custeada contra AI Credits a um centavo cada, e expressa em dólares.

Referências

Fontes primárias para o modelo de conversão, os baselines de produtividade que ele converte e o lado de custo em GitHub AI Credits.

- GitHub Copilot Documentation, a referência para agent mode, o coding agent e o billing em AI Credits onde um crédito equivale a um centavo.
- GitHub, Quantifying GitHub Copilot's impact on developer productivity and happiness, o sinal bruto de produtividade: 55 por cento mais rápido na conclusão de tarefas.
- GitHub, Quantifying GitHub Copilot's impact in the enterprise with Accenture, evidência enterprise: 88 por cento do código sugerido retido nas versões finais.
- DORA, State of DevOps 2025, baselines de desempenho de entrega para a alavanca de throughput.
- VS Code, Customize AI responses, arquivos de instruções e agentes customizados que reduzem o consumo de créditos por tarefa.
- Agentic DevOps SDLC, 24 Personas, fonte complementar, o detalhe completo das personas e o modelo de conversão em que este capítulo se apoia.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps