AI IMPACT ACROSS THE SDLC · PLAYBOOK

# La IA cambia veinticuatro roles, no uno, y horas ahorradas no son dólares ahorrados.

Los agentes de IA no cambian un rol. Cambian veinticuatro, con intensidades distintas en momentos diferentes, y por eso un roadmap único de IA para toda la ingeniería es un error de categoría. Este playbook mapea el impacto rol por rol en un SDLC enterprise de 5.000 ingenieros, configura GitHub Copilot para cada rol, secuencia la adopción en cuatro olas a lo largo de doce meses y ata cada hora recuperada a una Acción de Conversión, porque el tiempo ahorrado no es dinero ahorrado hasta que un paso deliberado lo convierte en reducción de costo, crecimiento de ingreso o una mejor experiencia del colaborador.


AUTORA

Paula Silva

CARGO

AI-Native Software Engineer

CONTACTO

in/paulanunes


TIEMPO DE LECTURA

~ 45 minutos, de punta a punta

| 24 | 6 | 4 |
|---|---|---|
| PERSONAS | CAPÍTULOS | OLAS DE ADOPCIÓN |

12

MESES HASTA
ADOPCIÓN PLENA

CAPÍTULOS

Chapter 00

## Veinticuatro roles, no un roadmap

Por qué un roadmap único de IA es un error de categoría, por qué una persona es una unidad de trabajo con resultado medible, por qué el tiempo ahorrado no es dinero ahorrado, y el alcance: un SDLC enterprise de 5.000 ingenieros con roles legacy incluidos.

Contexto →

Chapter 01

## El mapa de personas en cuatro clusters

Las 24 personas del SDLC agrupadas en Estrategia y descubrimiento, Construcción y calidad, Operación y confiabilidad, y Gobernar y crecer, cada una con la capacidad de GitHub Copilot que necesita y el antes y después que mide.

Personas →

Chapter 02

## Configurando GitHub Copilot por perfil

Los cinco niveles de personalización, el agent mode con anclaje vía MCP, la elección de modelo en el selector de Copilot por tarea, y la meseta del nivel 1 al nivel 2 donde la mayoría de los equipos se estanca.

Método →

Chapter 03

## Cuatro olas a lo largo de doce meses

La línea de tiempo de ejecución: reducción de costo inmediata, optimización de proveedor y el parque legacy, crecimiento de ingreso vía menor time-to-value, y luego resiliencia y experiencia del colaborador.

Adopción →

Chapter 04

## Métricas guiadas por resultado y gobernanza

Las cinco Métricas Guiadas por Resultado que cargan el impacto, el control de IA que mueve cada una, la cadencia como indicador adelantado de patrocinio, y la postura de gobernanza que lo mantiene defendible.

Métricas →

Chapter 05

## Convirtiendo horas ahorradas en dólares

Por qué las ganancias se fugan sin una Acción de Conversión, las cuatro palancas que convierten horas en dinero, el costeo del programa en GitHub AI Credits, y el business case ola por ola.

Conversión →

CONSEJOS DE LECTURA

> **ATAJOS DE TECLADO**
>
> Presiona `/` para buscar. Usa `j` y `k` para moverte entre capítulos.
> Presiona `t` para cambiar tema. Presiona `g` para ir al inicio.

---

**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 Y TESIS

# Veinticuatro roles, no un roadmap

Los agentes de IA no cambian un rol. Cambian veinticuatro, con intensidades distintas en momentos diferentes. Ese único hecho rompe el plan enterprise más común, el roadmap único para toda la ingeniería. Este capítulo fija el marco para el resto del playbook: por qué un roadmap único es un error de categoría, por qué una persona es la unidad correcta de análisis, por qué el tiempo ahorrado no es dinero ahorrado, y qué contiene realmente un SDLC enterprise de 5.000 ingenieros, roles legacy incluidos.

---

## Por qué un roadmap único de IA es un error de categoría en el SDLC

Los agentes de IA no cambian un rol. Cambian veinticuatro roles, con intensidades distintas en momentos diferentes. Un Product Manager, un Staff Engineer, un Site Reliability Engineer, un Ingeniero de Datos y un Compliance Officer enfrentan, cada uno, un horizonte agéntico distinto. Cada uno necesita una combinación distinta de GitHub Copilot, Microsoft 365 y Azure para llegar a valor. Tratar "IA para ingeniería" como un roadmap único es un error de categoría. Asume un punto de partida uniforme, un formato de tarea uniforme y una curva de retorno uniforme. Ninguno de ellos se sostiene a lo largo del ciclo de vida del software.

El error es caro porque conduce al plan equivocado. Un roadmap único financia el caso de uso más ruidoso, por lo general la generación de código, y deja sin recursos a los roles donde el retorno es mayor y más lento en aparecer. Un Analista de Requisitos que cruza transcripciones con el conocimiento institucional, o un desarrollador COBOL que convierte treinta años de comportamiento no documentado en documentación consultable, no aparece en un dashboard de autocompletado de código. Cuando una sola métrica gobierna todo el programa, esas victorias quedan invisibles, y el programa parece haber rendido menos de lo debido. La corrección no es un mejor promedio. La corrección es mapear el impacto

rol por rol, para que cada persona reciba la herramienta, la profundidad de personalización y el plazo que su trabajo realmente exige.

## Una persona como unidad de trabajo, conjunto de decisiones y resultado medible

Una persona no es un cargo. Es una unidad de trabajo, un conjunto de decisiones y un resultado medible. Esa definición es lo que hace operativo al framework. Un cargo no dice nada sobre dónde ayuda un agente. Una unidad de trabajo dice qué tareas puede asumir el agente, qué decisiones quedan con el humano y qué resultado se mide al final. Para el Desarrollador Moderno, la unidad es una rama de feature, las decisiones son diseño y revisión, y el resultado es throughput. Para el Site Reliability Engineer, la unidad es un incidente, las decisiones son diagnóstico y remediación, y el resultado es el tiempo hasta recuperar.

GitHub Copilot le da a cada persona un punto de contacto distinto. Algunos viven en agent mode dentro del editor, delegando issues completas a un coding agent que abre un pull request con pruebas. Otros viven en el chat, haciendo preguntas a una base de código que no escribieron. La jerarquía de personalización es lo que ajusta el agente al rol: un .github/copilot-instructions.md válido para todo el repositorio, archivos de instrucciones por ruta, un AGENTS.md para el comportamiento del proyecto, custom agents para una persona nombrada y MCP servers que conectan el agente a los sistemas que la persona ya usa. El modelo en el selector de Copilot también es una elección por persona. Haiku 4.5 para ediciones rápidas y baratas; Opus 4.8 o Claude Fable 5 para razonamiento difícil en muchos archivos. Una unidad de trabajo, un conjunto de decisiones, un resultado, una configuración.

## La tesis: el tiempo ahorrado no es dinero ahorrado

Las ganancias de productividad por sí solas no pagan por la IA. Esta es la tesis que el resto del playbook defiende. Un agente que le ahorra seis horas por semana a un desarrollador todavía no le ha ahorrado nada a la empresa. El tiempo ahorrado se fuga de vuelta a retrabajo, sobrecarga de validación, cambio de contexto y reuniones, a menos que un paso deliberado lo redirija. La investigación de Gartner lo

plantea sin rodeos: las ganancias de productividad, por sí solas, no pagan por la IA. La distancia entre horas recuperadas y dólares registrados es donde la mayoría de los business cases de IA falla en silencio, y por eso un CFO escéptico muchas veces tiene razón en dudar del número en la diapositiva.

La disciplina que cierra esa distancia es una Acción de Conversión. Cada persona debe declarar cómo sus horas recuperadas se convierten en uno de tres resultados: reducción de costo, crecimiento de ingreso o mejora de la experiencia del colaborador. Un Tech Lead que recupera tiempo de revisión lo convierte entregando features antes, no sosteniendo más reuniones. Hay un costo del otro lado del balance, y GitHub Copilot lo hace explícito. Las ejecuciones de agente se miden contra los GitHub AI Credits, donde un crédito equivale a un centavo de dólar desde el 1 de junio de 2026. Esa etiqueta de precio es el punto. Fuerza la pregunta honesta que la tesis exige: ¿esta ejecución produjo más valor del que consumió?

> DEFINICIÓN
>
> El tiempo ahorrado no es dinero ahorrado. Cada persona debe declarar una Acción de Conversión que convierta las horas recuperadas en reducción de costo, crecimiento de ingreso o mejora de la experiencia del colaborador. Sin eso, las horas se fugan de vuelta a retrabajo y el business case no cierra.

## Alcance: un SDLC enterprise de 5.000 ingenieros, con roles legacy incluidos

El framework está acotado a una empresa real, no a una demo. La organización de referencia es una función de entrega de software de 5.000 ingenieros, organizada en ocho fases del ciclo de vida del software, desde la gestión de clientes y contratos hasta gobernanza, seguridad y compliance. Las veinticuatro personas son los roles que de hecho integran esa función. No son arquetipos elegidos para hacer que la IA se vea bien. Incluyen los roles que los playbooks genéricos omiten. Las empresas de esta escala son la norma, no la excepción: más del noventa por ciento de la Fortune 100 usa GitHub.

### Los roles legacy están en el alcance, no alrededor de él

La mayoría de las historias de adopción de IA asume, en silencio, código greenfield y stacks modernas. Una empresa de 5.000 ingenieros no es así. Ejecuta décadas de sistemas COBOL, Natural/ADABAS, DB2 y mainframe, mantenidos por desarrolladores senior que se están jubilando y se llevan conocimiento no documentado con ellos. El framework pone tres perfiles de desarrollador legacy en el alcance a propósito. GitHub Copilot soporta la sintaxis COBOL y Natural, y un custom agent puede reducir la comprensión de un programa de días a horas. El resultado de mayor valor es la captura permanente de conocimiento: el contexto que vivía en la cabeza de un ingeniero se vuelve documentación que todo el equipo puede consultar. Un roadmap que cubre solo los roles modernos no es un roadmap para esta empresa. Es un roadmap para la parte de ella que ya era fácil.

# Referencias

Fuentes primarias para el framework de las veinticuatro personas, la evidencia empírica de productividad y las capacidades de GitHub Copilot que este playbook configura por rol.


- Agentic SDLC Personas, sitio companion del framework, la fuente original del framework de veinticuatro personas sobre el que se apoya este playbook.
- Las veinticuatro personas en detalle, cada rol con sus tareas, herramientas y datos de antes y después.
- GitHub Copilot Documentation, agent mode, el coding agent, los custom agents y los archivos de instrucciones del repositorio que este framework configura por persona.
- Customize AI responses in VS Code, archivos de instrucciones, AGENTS.md, custom agents y agent skills, los cinco niveles de personalización de Copilot.
- GitHub, Quantifying GitHub Copilot's impact on developer productivity and happiness, el resultado de 55% de velocidad de tareas citado para las personas de desarrollo.
- GitHub, Quantifying GitHub Copilot's impact in the enterprise with Accenture, el estudio enterprise detrás de la cifra de 88% de código retenido.

- GitHub Universe 2024 press release, más del 90% de la Fortune 100 usa GitHub, la escala a la que apunta este framework.
- DORA 2025 State of DevOps research, rendimiento de entrega, frecuencia de despliegue y tasa de fallo en cambios.

---

**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 A QUIÉN AFECTA

# El mapa de personas en cuatro clusters: a quién transforman los agentes de IA, y cómo.

Los agentes de IA no cambian un rol. Cambian 24, con intensidades distintas en momentos diferentes. Tratar la IA para ingeniería como un único roadmap es un error de categoría, porque el Product Manager, el desarrollador COBOL, el Site Reliability Engineer y el Compliance Officer enfrentan horizontes agénticos distintos. Este capítulo agrupa las 24 personas en cuatro clusters según el tipo de trabajo que la IA cambia: Estrategia y descubrimiento, Construcción y calidad, Operación y confiabilidad, y Gobernar y crecer.

## El mapa: 24 personas, cuatro clusters, una unidad de trabajo

Una persona es una unidad de trabajo, un conjunto de decisiones y un resultado medible. Las 24 personas vienen de un SDLC enterprise real e incluyen roles legados como COBOL y Natural/ADABAS que los playbooks genéricos omiten. Agruparlas en cuatro clusters responde una pregunta práctica antes del detalle rol por rol: qué tipo de trabajo cambia realmente un agente de IA, y quién es su dueño. Cada cluster comparte un patrón de impacto, un conjunto de herramientas principales y un riesgo que decide si la ganancia es real o se fuga.

La lente a lo largo de todo el texto es GitHub Copilot. El agent mode ejecuta el trabajo de múltiples pasos. Un archivo .github/copilot-instructions.md y el AGENTS.md llevan las convenciones que cada repositorio ya codifica, para que la salida generada cumpla los estándares corporativos en lugar de adivinarlos. Los servidores MCP conectan al agente con los sistemas en los que vive cada persona, desde Azure Boards hasta la base de conocimiento corporativa. La facturación corre sobre GitHub AI Credits, donde un crédito equivale a un centavo de dólar, vigente desde el 1 de junio de 2026, así que cada persona mapea a una línea de costo que

un CFO puede leer. El selector de modelos trae Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro y la familia GPT-5.x, y el modelo correcto se elige por la tarea, no por el rol.

## Estrategia y descubrimiento: comprimir el overhead de reuniones y síntesis

Este cluster reúne al Business Manager, al Project Manager, al Product Owner, al Requirements Engineer, al Enterprise y Solution Architect, al Tech Lead, al Scrum Master y al UAT Analyst. El cuello de botella común no es el código. Es el overhead de reuniones y síntesis. El patrón de impacto es compresión. Las actas manuales que tomaban de 30 a 40 minutos pasan a 3 a 5 minutos de notas generadas por Copilot a partir de una transcripción de Teams. La compilación semanal de status que tomaba de 4 a 5 horas pasa a un reporte asistido por agente de 30 a 60 minutos, extraído de Azure DevOps Boards, GitHub Projects y Power BI.

La ganancia estratégica no son los minutos ahorrados. Es a dónde va el tiempo recuperado: análisis de alcance, negociación de dependencias y descubrimiento con stakeholders, el trabajo que solo una persona hace bien. Ese redireccionamiento es también el riesgo. Si la reasignación de capacidad no se impone en la cadencia operacional, el tiempo recuperado se fuga directo de vuelta a más reuniones de status. El tooling principal de este cluster es Microsoft 365 Copilot, Copilot Studio, GitHub Projects con Ask mode, Azure DevOps Boards y Power BI.

## Construcción y calidad: de escribir código a especificar intención

Este es el cluster más grande: el Modern Software Developer, el QA Engineer, el Legacy Developer, el COBOL Developer, el Natural/ADABAS Developer, el Application Security Engineer, el Data Engineer y el ML Engineer. El patrón de impacto es un cambio en la unidad de trabajo, de escribir código a especificar intención. El tecleo deja de ser la unidad; la spec pasa a serlo. La personalización mediante archivos de instrucción, AGENTS.md, custom agents y agent skills hace que el código generado cumpla los estándares corporativos a la primera, lo que elimina la mayoría de las

iteraciones de review. El rendimiento del Modern Developer sube de cerca de 40 a 80 o 100 function points por mes, con mejor cobertura de pruebas que antes.

Dos gates duros gobiernan este cluster. La seguridad de aplicación sigue siendo un gate obligatorio, no una recomendación, y la ganancia de productividad debe estar atada a una Acción de Conversión declarada, o se fuga de vuelta a retrabajo en lugar de llegar al balance. El tooling principal es GitHub Copilot en los modos Ask, Edit, Agent, Plan y Coding Agent, GitHub Copilot Code Review, GitHub Advanced Security, Microsoft Fabric y Azure AI Foundry.

### Captura de legado: COBOL y Natural/ADABAS

El resultado de mayor valor en este cluster no es el código nuevo. Es la captura de conocimiento. Desarrolladores senior que acumulan de 20 a 30 años de comportamiento de sistema no documentado se están jubilando, y ese contexto nunca se escribió. GitHub Copilot soporta la sintaxis COBOL y Natural de forma nativa. Un agente explicador reduce la comprensión de un programa de 2 a 5 días a 2 a 4 horas, y la ganancia de productividad en la generación de código nuevo llega al 85%. El resultado permanente es que el conocimiento que vivía en una sola persona se convierte en documentación consultable que sobrevive a la jubilación.

> DEFINICIÓN
>
> El cluster de construcción cambia la unidad de trabajo, del tecleo a la spec. El mayor valor no es la velocidad del código nuevo. Es convertir el conocimiento de legado que vivía en una persona en documentación que la organización puede consultar.

## Operación y confiabilidad: del combate reactivo a la ingeniería proactiva

Este cluster es el DevOps y Platform Engineer, el Database Administrator, el Release Manager y el Site Reliability Engineer. El patrón de impacto es pasar del combate reactivo de incendios a la ingeniería proactiva. El triage de incidentes cae de 15 a 30 minutos a 3 a 5 minutos con un agente comandante de incidente. Los post-mortems

se reducen de 4 a 8 horas a 1 a 2 horas. Las release notes pasan de 3 a 5 horas por release a 30 a 60 minutos. El toil, el trabajo manual y repetitivo que un Site Reliability Engineer mide, baja de 40 a 50% de la capacidad a menos de 30%.

El Site Reliability Engineer muestra los números más expresivos de todo el framework. La correlación automatizada de alertas reduce la detección de incidentes de 5 a 10 minutos a 30 segundos. El diagnóstico de causa raíz cae de 2 a 3 horas de investigación manual a 10 minutos de análisis automatizado. La remediación mediante runbooks ejecutables toma de 2 a 5 minutos frente a los 30 minutos a 2 horas anteriores. La regla que gobierna mantiene al humano al mando: la remediación automatizada pasa tres ejecuciones supervisadas antes de promoverse, y el agente permanece consultivo durante un incidente en vivo mientras el comandante humano mantiene la autoridad. El tooling principal es Azure Monitor, Application Insights, Microsoft Sentinel y los entornos de deployment de GitHub Actions.

## Gobernar y crecer: la evidencia continua reemplaza la auditoría puntual

El último cluster es el Engineering Manager e IT Manager, el InfoSec y Compliance Officer, el Compliance y Audit Officer y el UX y Product Designer. El patrón de impacto es que la evidencia continua reemplaza la auditoría puntual. El Engineering Manager asume las métricas de adopción desde el primer día, y por eso esta persona queda en la primera ola de adopción, no en la última. El InfoSec pasa del escaneo en el momento del pull request a la recolección continua de compliance en cada push. Una auditoría manual semestral que tomaba de 3 a 4 semanas pasa a ser un proceso automatizado continuo con traza de evidencias completa.

El diseñador también pertenece aquí. La síntesis de investigación de UX cae de 2 a 3 días por estudio a 4 a 8 horas, y las auditorías de accesibilidad pasan a ser continuas en lugar de anuales. Dos riesgos gobiernan este cluster. Las funciones en public preview no deben correr en producción para workloads críticos, y la indemnización de IP debe confirmarse antes de la adopción, vía GitHub Copilot Enterprise y el Azure OpenAI Customer Copyright Commitment. El ledger de GitHub AI Credits ayuda aquí: como un crédito equivale a un centavo de dólar, la traza de auditoría ya lleva una línea de costo para cada acción del agente. El tooling principal

es GitHub Advanced Security, la GitHub Audit Log API, Microsoft Purview, Microsoft Sentinel, dashboards de compliance en Power BI y Azure Policy.

> DEFINICIÓN
>
> Cuatro clusters, una disciplina: medir el antes y el después por persona, atar cada hora recuperada a un resultado declarado y mantener a un humano con autoridad donde el trabajo lleva riesgo. Eso es lo que convierte la estrategia de IA en algo que una organización de 5.000 ingenieros puede ejecutar.

# Referencias

Fuentes primarias para los patrones de impacto por persona, los benchmarks de productividad y los mecanismos de personalización de GitHub Copilot citados en este capítulo.


- GitHub, Research: quantifying GitHub Copilot impact on developer productivity and happiness, la fuente del benchmark de 55% más rápido en la finalización de tareas.
- GitHub, Research: quantifying GitHub Copilot impact in the enterprise with Accenture, la fuente del 88% del código sugerido que se mantiene en las versiones finales.
- GitHub Universe 2024, enterprise Copilot adoption, más del 90% de la Fortune 100 usa GitHub.
- DORA, 2025 State of DevOps research, desempeño de entrega elite, frecuencia de deployment y tasa de fallo en cambios.
- GitHub Copilot documentation, agent mode, coding agent y custom agents.
- Customize AI responses in VS Code, archivos de instrucciones, AGENTS.md, custom agents y agent skills.
- Agentic DevOps SDLC, 24 Personas, companion source, la fuente rol por rol para el mapa de cuatro 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 Y ARQUITECTURA

# Configurando GitHub Copilot por perfil: cinco niveles, un selector, salida anclada.

Un Copilot genérico ayuda un poco a cada perfil. Un Copilot configurado cambia cómo trabaja el perfil. La diferencia no es el modelo. Es el contexto que le das y la forma en que lo anclas. Este capítulo describe los cinco niveles de personalización, el agent mode con anclaje vía MCP, la elección de modelo en el selector y la meseta donde la mayoría de los equipos se estanca.

## Los cinco niveles de personalización

GitHub Copilot se configura en cinco niveles, cada uno añadiendo especificidad sobre el anterior. Los cinco funcionan con Copilot en el IDE, sin GitHub Enterprise Cloud. El Nivel 1 es el Copilot base: el selector, la completación inline, el chat y el agent mode, sin conocimiento de tus convenciones. Es genérico por naturaleza. El Nivel 2 es un archivo .github/copilot-instructions.md en la raíz del repositorio. Copilot lo lee al iniciar la sesión y lo aplica a cada petición en ese repositorio: reglas de nomenclatura, restricciones de seguridad y las convenciones corporativas que un recién llegado necesitaría. Es el archivo de mayor retorno de la jerarquía, y lleva una tarde escribirlo.

El Nivel 3 estrecha el alcance. Los archivos de instrucción con alcance por glob, .instructions.md con un patrón applyTo, llevan reglas que se aplican solo a los archivos coincidentes: *.java, *.sql, *.py. Una regla de SQL no llega a un prompt de Java, y una regla de Java no llega a un archivo de prueba. El Nivel 4 son los custom agents, declarados en .agent.md, cada uno un perfil especializado con sus propias tools y handoffs. El agente @cobol-explainer, el agente @test-generator y el agente @code-reviewer se pasan trabajo entre sí como handoffs formales. El Nivel 5 son las Agent Skills, definidas en SKILL.md: flujos de trabajo específicos por tarea, bajo

demanda, con templates incluidos que Copilot carga solo cuando la tarea lo exige. Los custom agents están en public preview; las Agent Skills son experimentales.

### Mapea el nivel al perfil

El nivel que un perfil necesita lo define el trabajo, no la seniority. Un Gestor de Negocio que quiere actas de reunión necesita el Nivel 1 y M365 Copilot, nada más. Un Desarrollador Moderno se beneficia de las convenciones del Nivel 2 y de las reglas glob del Nivel 3 para cada lenguaje del stack. Un desarrollador COBOL necesita un agente @cobol-explainer de Nivel 4 para comprimir la comprensión de un programa de días a horas. Un Site Reliability Engineer necesita skills de Nivel 5 que envuelvan runbooks ejecutables. Configura el nivel que el perfil realmente va a usar y después sube cuando alcances el techo.

## Agent mode y anclaje vía MCP

Los archivos de instrucción le dicen a Copilot lo que prefieres. El anclaje le dice a Copilot lo que es verdadero. El agent mode es donde los dos se combinan: el agente planifica un cambio en varios archivos, corre tools, lee resultados e itera, en lugar de completar una sola línea. Por sí solo, el agent mode todavía infiere hechos sobre la base de código. El Model Context Protocol cierra esa brecha. Un servidor MCP expone hechos determinísticos al agente bajo demanda: versiones de dependencias resueltas, interfaces de servicio, definiciones de schema, contratos de APIs internas. El agente consulta el servidor en lugar de adivinar, y el código generado cumple el estándar corporativo a la primera en lugar de a la tercera.

El retorno se mide en retrabajo evitado. La investigación de GitHub encontró que el 88% del código sugerido por Copilot se mantuvo en las versiones finales cuando el modelo tenía el contexto que necesitaba. El anclaje eleva ese número al quitar la clase de errores que viene de la inferencia: un método obsoleto, una versión equivocada de dependencia, una frontera de servicio que el agente no podía ver. Configura los servidores MCP una vez a nivel de plataforma, apunta el Copilot de cada perfil hacia ellos, y cada perfil hereda la misma fuente de verdad. Los archivos de instrucción llevan preferencia. Los servidores MCP llevan hecho. El agent mode actúa sobre ambos.

DEFINICIÓN

Los archivos de instrucción llevan preferencia. Los servidores MCP llevan hecho. El agent mode actúa sobre ambos. Preferencia sin hecho produce código seguro de sí mismo que viola una restricción que el agente nunca vio. Ancla primero, después instruye.

## Eligiendo un modelo en el selector de Copilot

El selector de Copilot expone varios modelos, y el correcto depende de la tarea, no de un ranking. Claude Fable 5 y Opus 4.8 son las opciones de razonamiento pesado: decisiones de arquitectura, comprensión de legacy y refactorizaciones en varios archivos donde un plan equivocado sale caro. Sonnet 5 y los modelos GPT-5.x son las opciones de codificación del día a día: cargan la mayor parte del trabajo de features, el scaffolding de endpoints y la generación de pruebas a un costo menor por petición. Haiku 4.5 es la opción rápida para iteraciones de alto volumen y bajo riesgo: completaciones, ediciones pequeñas y preguntas rápidas donde la latencia importa más que la profundidad. Gemini 3.1 Pro es la opción de contexto largo y multimodal, útil cuando la tarea abarca archivos grandes o incluye diagramas y capturas de pantalla.

La elección de modelo tiene una dimensión de costo. Desde el 1 de junio de 2026, el uso de Copilot se mide en GitHub AI Credits, donde un crédito equivale a un centavo de dólar. Los modelos más pesados consumen más créditos por petición, así que casar el modelo con la tarea es una decisión de presupuesto además de calidad. El patrón que se sostiene entre perfiles: usa un modelo de tier medio por defecto para el trabajo de rutina, escala a un modelo pesado cuando el plan importa y baja a un modelo rápido en las ediciones masivas. Un Site Reliability Engineer diagnosticando un incidente y un desarrollador generando un endpoint CRUD no deberían estar en el mismo modelo, y con el selector no tienen por qué estarlo.

### Un selector, muchos perfiles

El selector es por petición, así que un mismo desarrollador transita entre modelos dentro de una sesión. Pídele a Opus 4.8 que planifique una refactorización, cambia a

Sonnet 5 para implementarla y baja a Haiku 4.5 para las ediciones mecánicas. Los archivos de instrucción y el anclaje vía MCP permanecen constantes en el cambio, así que el contexto no se reinicia cuando el modelo lo hace. El modelo cambia cómo se razona el trabajo. La configuración lo mantiene correcto.

## Donde los equipos se estancan: la meseta del nivel 1 al nivel 2

La mayoría de los equipos no se estanca en el Nivel 4 ni en el Nivel 5. Se estanca entre el Nivel 1 y el Nivel 2. Instalan Copilot, usan la completación inline y nunca escriben el archivo .github/copilot-instructions.md. La investigación de GitHub reporta un 55% más de rapidez en completar tareas con Copilot, y un equipo que alcanza ese número puede creer que el trabajo está terminado. No lo está. El archivo que codificaría las convenciones del equipo nunca se escribe, y cada sesión paga por esa ausencia reinfiriendo las mismas reglas. La meseta no es técnica. El paso de mayor retorno simplemente parece opcional, y no lo es.

Subir es una secuencia, no un salto. Escribe el copilot-instructions.md primero, porque es un solo archivo y se aplica a todo. Añade las instrucciones con alcance por glob después, un lenguaje a la vez, a medida que las convenciones de cada stack se estabilizan. Introduce custom agents solo donde un flujo de trabajo repetido justifica un perfil con nombre: el @code-reviewer que recupera tiempo de revisión, el @cobol-explainer que captura conocimiento a punto de jubilarse. Añade las Agent Skills al final, para los flujos específicos por tarea que un perfil corre con frecuencia suficiente para hacer template. Cada nivel debe merecer el siguiente. Un equipo que sube de forma deliberada llega a un punto donde Copilot se comporta como si perteneciera a la base de código, porque pertenece.

> DEFINICIÓN
>
> El paso de mayor retorno de toda la jerarquía es el más barato de saltarse: escribir el primer archivo copilot-instructions.md. Un equipo atascado en el Copilot base paga el precio completo por una fracción del valor.

# Referencias

Fuentes primarias para la jerarquía de personalización, el agent mode y la evidencia de productividad.


- Customize AI responses in VS Code, la referencia para archivos de instrucción, custom agents y Agent Skills.
- GitHub Copilot Documentation, la referencia para agent mode, coding agent y custom agents.
- GitHub, Quantifying GitHub Copilot impact on developer productivity and happiness, la fuente del número de 55% más de rapidez en completar tareas.
- GitHub and Accenture, Quantifying GitHub Copilot impact in the enterprise, la fuente del número de 88% de código mantenido.
- The 24 personas in detail, la matriz de configuración y resultado por perfil sobre la que se apoya este capítulo.


---


**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 LÍNEA DE TIEMPO DE EJECUCIÓN

# Cuatro olas a lo largo de doce meses: cómo entran en operación los 24 perfiles.

El framework de los 24 perfiles no es un lanzamiento, es una secuencia. La adopción corre en cuatro olas a lo largo de doce meses, y cada ola agrega perfiles cuyo valor depende de la infraestructura que la ola anterior dejó montada. Este capítulo recorre la línea de tiempo ola por ola, nombra los perfiles de cada una y ata cada ola al cubo validado por Gartner en el que caen sus retornos: reducción de costo, crecimiento de ingreso o mejora de la experiencia del colaborador.

## Ola 1, ganancia inmediata: siete perfiles, reducción de costo

Meses 1 a 3. Siete perfiles entregan retorno visible en semanas porque sus tareas principales encajan con lo que GitHub Copilot hace en agent mode de forma nativa: síntesis de texto, generación de código y agregación de datos. Gestor de Negocio y Gestor de Proyecto son las ganancias más rápidas fuera de ingeniería. Microsoft 365 Copilot convierte un proceso manual de 30 minutos de elaboración de actas en una extracción automática de decisiones, action items y responsables a partir de una transcripción de Teams. El Desarrollador Moderno muestra los números más expresivos. Un endpoint REST que toma de 4 a 6 horas de trabajo manual se reduce a 30 a 45 minutos con agent mode y generación de scaffold, y el Coding Agent agrega un carril paralelo asíncrono donde el desarrollador delega de 2 a 3 issues por día a un agente en segundo plano que abre un PR con pruebas. Los puntos de función por mes prácticamente se duplican, de cerca de 40 a un rango de 80 a 100, según el estudio Forrester Total Economic Impact.

QA y Test Lead reduce la generación de suites de pruebas de 2 a 3 semanas a 3 a 5 días, con un piso de cobertura del 75 por ciento aplicado automáticamente. El Tech Lead recupera cerca del 60 por ciento del tiempo de revisión, porque Copilot Code

Review se encarga de las verificaciones repetitivas y el lead se queda con las decisiones arquitecturales. DBA y Engineering Manager completan la ola. El Engineering Manager tiene una tarea específica desde el primer día: instrumentar la adopción. Uso diario por encima del 70 por ciento en 60 días es el indicador predictivo de todo el programa, y el Manager conecta la salida de la GitHub Copilot Metrics API a Power BI para que el feedback anecdótico se vuelva un business case con VAN y período de payback. La facturación corre en GitHub AI Credits, un crédito valiendo un centavo de dólar desde junio de 2026, lo que convierte el gasto de modelos en el Copilot picker en una línea que el Manager sigue contra las horas recuperadas.

## Ola 2, expansión: ocho perfiles, optimización de proveedor y el parque legacy

Meses 4 a 6. Ocho perfiles entran en operación, y cada uno exige más configuración que la Ola 1. MCP servers, un .github/copilot-instructions.md en el repositorio y archivos AGENTS.md específicos por perfil entran antes de que Copilot alcance su plena utilidad. El Analista de Requisitos obtiene un agente que consulta la base de conocimiento corporativa en SharePoint, cruza transcripciones con el conocimiento institucional y señala conflictos antes de que lleguen al desarrollo. El Arquitecto Enterprise asume el núcleo de la modernización: décadas de código COBOL, Natural/ADABAS, DB2 y mainframe que eventualmente deben migrar a Spring Boot o microservicios. Un agente de exploración mapea la base legacy, produce diagramas C4 y escribe evaluaciones de viabilidad, comprimiendo semanas de análisis en días.

Tres perfiles de desarrollador legacy están en el centro de esta ola, y cargan el problema del conocimiento institucional. Desarrolladores senior con 20 a 30 años de conocimiento de sistemas no documentado están jubilándose. Copilot lee la sintaxis COBOL y Natural de forma nativa, y un agente personalizado @cobol-explainer reduce la comprensión de un programa de 2 a 5 días a 2 a 4 horas. El resultado de mayor valor es la captura permanente de conocimiento: lo que vivía en la cabeza de un solo ingeniero se vuelve documentación consultable. DevOps y Platform Architect e InfoSec cierran la ola. El cubo de ROI aquí es reducción de costo vía optimización de proveedor y outsourcing, y el indicador predictivo que el programa sigue son tres contratos renegociados con palanca de IA, porque un parque legacy que el equipo

interno ahora puede leer y modificar cambia los términos de toda conversación de outsourcing.

## Ola 3, especialización: ocho perfiles, crecimiento de ingreso vía menor time-to-value

Meses 7 a 9. Ocho perfiles especializados entran en operación con agentes personalizados de dominio que solo funcionan una vez que la infraestructura base de las Olas 1 y 2 está madura. El Product Owner gestiona backlogs de 500 o más ítems con priorización por WSJF orientada por IA y generación automática de user stories. Scrum Master y Agile Coach automatizan retrospectivas, sprint reviews y mapas de dependencias entre equipos. El Data Engineer está en la intersección de 50 o más fuentes legacy y modernas que alimentan Microsoft Fabric para mil o más usuarios de Power BI, y el tiempo de creación de pipelines cae cerca del 80 por ciento mientras el monitoreo de calidad de datos se vuelve continuo en lugar de reactivo.

ML y AI Engineer cierra el ciclo de retroalimentación de la plataforma: selección de modelos en el Copilot picker, optimización de prompts, frameworks de evaluación y gobernanza de IA como detección de alucinaciones y prevención de fuga de datos. Compliance y Audit Officer comprime una auditoría manual semestral de 3 a 4 semanas en un proceso continuo con trazas de evidencias completas, respuesta directa al LGPD, ISO 27001 y regulación sectorial. UX y Product Designer genera prototipos en Figma en minutos y habilita validación de stakeholders el mismo día para flujos que antes tomaban semanas. Ese es el hilo conductor de la ola: cada perfil aquí acorta el time-to-value, y por eso el cubo de ROI pasa de costo a crecimiento de ingreso, con la meta del programa siendo un time-to-value mediano menor en al menos 30 por ciento.

## Ola 4, madurez: SRE, resiliencia y experiencia del colaborador

Meses 10 a 12. Un perfil alcanza la plena madurez al final, el Site Reliability Engineer, y sus números son los más expresivos del framework. La correlación automatizada de alertas reduce la detección de incidentes de 5 a 10 minutos a 30 segundos. El

diagnóstico de causa raíz cae de 2 a 3 horas de investigación manual a cerca de 10 minutos de análisis automatizado. La remediación mediante runbooks ejecutables toma de 2 a 5 minutos frente a los 30 minutos a 2 horas anteriores. Para sistemas donde el tiempo de inactividad cuesta millones por minuto, estas no son ganancias marginales. Cambian cómo se diseña la confiabilidad.

El cubo de ROI de la Ola 4 es mejora de la experiencia del colaborador y resiliencia, el tercer cubo validado por Gartner, y el indicador predictivo es un alza de eNPS de al menos 8 puntos entre usuarios de IA. La calidad de vida del on-call no es un beneficio vago aquí. Es el argumento de retención que una revisión de CFO acepta junto a costo e ingreso, y es lo que retiene a los ingenieros senior de cuyo juicio depende todo el programa.

### El programa entra en refinamiento continuo

Al llegar al mes 12, el programa deja de agregar perfiles y empieza a ajustar. Métricas DORA consolidadas, ROI reportado al C-level y una cultura de ingeniería AI-first en los 24 perfiles se vuelven el estado permanente. Los agentes personalizados se afinan contra el uso real, los archivos copilot-instructions se corrigen contra la salida real y las integraciones MCP se podan. Las olas fueron secuenciales y no opcionales. El refinamiento es permanente.

> DEFINICIÓN
>
> El modelo de cuatro olas es secuenciado, no simultáneo. La Ola 1 entrega reducción de costo vía compresión de experiencia, la Ola 2 vía optimización de proveedor y outsourcing más el parque legacy, la Ola 3 convierte menor time-to-value en crecimiento de ingreso, y la Ola 4 convierte resiliencia y experiencia del colaborador en un argumento de retención. Después del mes 12, el programa entra en refinamiento continuo en los 24 perfiles.

## Referencias

Fuentes primarias para el modelo de adopción en cuatro olas, el catálogo de perfiles y la evidencia de productividad detrás de la línea de tiempo.

- Agentic DevOps SDLC Personas, sitio complementario, el catálogo de origen de los 24 perfiles y del roadmap de adopción en cuatro olas.
- GitHub Copilot Documentation, agent mode, el Coding Agent y los custom agents referenciados en las cuatro olas.
- Customize AI responses in VS Code, instructions files, AGENTS.md, custom agents y agent skills, las capas de personalización de las que depende la Ola 2.
- GitHub Research, quantifying GitHub Copilot impact on developer productivity and happiness, el 55 por ciento de finalización de tareas más rápida detrás de las ganancias de desarrollador de la Ola 1.
- DORA, 2025 State of DevOps research, las métricas de entrega consolidadas cuando el programa llega al mes 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 Y GOBERNAR

# Métricas guiadas por resultado y gobernanza: qué medir, y cómo mantenerlo defendible.

Un piloto que no sobrevive a una revisión del CFO no escala. Este capítulo convierte la adopción en un business case defendible. Nombra las cinco Métricas Guiadas por Resultado que cargan la mayor parte del impacto por persona, empareja cada una con el control de IA que la mueve, y define la postura de gobernanza que mantiene el programa auditable a escala enterprise.

## La matriz ODM: métrica, dueño de la palanca y el resultado que sigue el CFO

Según Gartner G00799085 y G00841077, una Métrica Guiada por Resultado tiene línea directa de vista a un resultado de negocio. Cinco ODMs cargan la mayor parte del impacto por persona, y cada una nombra dos cosas más allá del propio número: el equipo dueño de la palanca y el resultado que sigue el CFO. Time-to-market pertenece a Product, al Tech Lead y al Release Manager, y responde a crecimiento de ingreso. Cycle time pertenece a Developers, QA y DevOps, y responde a reducción de costo y throughput. Defect escape rate pertenece a QA, AppSec y SRE, y responde a evitar multa y NPS. Customer NPS pertenece a Product, UX y Support, y responde a retención y crecimiento de ACV. Revenue per engineer pertenece al Engineering Manager con el socio del CFO, y responde a eficiencia de headcount.

La columna de propiedad no es decoración. Una métrica sin dueño se dispersa, y una métrica sin resultado seguido por el CFO nunca llega al business case. La columna de resultado es la capa de traducción entre ingeniería y finanzas. Revenue per engineer es la razón que sobrevive a una revisión del CFO, porque es la única expresada en las unidades que usa el equipo de finanzas. Las ganancias de productividad por sí solas no pagan por la IA, según Gartner G00837723. Cada persona debe declarar una Acción de Conversión que transforme las horas

recuperadas en uno de tres cubos: reducción de costo, crecimiento de ingreso o mejora de la experiencia del colaborador. Sin ese paso, el tiempo se fuga a retrabajo y reuniones, y el CFO tiene razón en dudar del caso.

## Empareja cada métrica con el control de IA que la mueve

Una métrica sin control es anecdótica. Un control sin métrica no tiene defensa de negocio. Así que cada ODM se empareja con el control de IA que la mueve, la señal de origen que la mide y la cadencia en la que se lee. Time-to-market se mueve con Spec Kit y agentes L4 personalizados, lee de GitHub Releases y Azure DevOps Release, y se revisa semanalmente. Cycle time se mueve con Copilot Code Review corriendo en todo PR, lee de GitHub Insights y Azure DevOps Analytics, y se revisa semanalmente. Defect escape rate se mueve con gates de GitHub Advanced Security más un agente de QA, lee de Defender for Cloud, Sentinel y GHAS, y se revisa mensualmente. Customer NPS se mueve con un sintetizador de research UX más un auditor de accesibilidad, lee de Microsoft Clarity y surveys en producto, y se revisa trimestralmente. Revenue per engineer se mueve con un tracker de adopción más una política de reasignación de capacidad, lee de la GitHub Copilot Metrics API y de reportes financieros, y se revisa trimestralmente.

La señal de origen importa tanto como el control. Cada métrica aquí lee de un sistema de registro, no de un autorreporte, y eso es lo que la hace sobrevivir a una revisión de steering. La adopción lee de la GitHub Copilot Metrics API, no de una survey. El costo es medible en el mismo marco: el uso de modelo en agent mode se factura en GitHub AI Credits, donde un crédito es un centavo, vigente desde el 1 de junio de 2026. Eso hace del costo por resultado un número real. Divide los créditos gastados por releases entregados o defectos capturados. Elegir un modelo más barato en el Copilot picker para una tarea de rutina y uno más fuerte para una tarea difícil se vuelve una palanca que la métrica puede tarificar.

## La disciplina de cadencia es un indicador adelantado de patrocinio

La cadencia no es una formalidad de reporte. Está ajustada a la velocidad con la que cada métrica puede moverse. El Engineering Manager es dueño de las cadencias semanal y mensual, que cubren time-to-market, cycle time y defect escape rate, porque esas se mueven con decisiones de ingeniería. El CTO es dueño de la cadencia trimestral, que cubre Customer NPS y revenue per engineer, porque esas se mueven con decisiones de negocio. Ajustar la cadencia a la métrica mantiene cada revisión accionable: una métrica semanal leída trimestralmente está vieja, y una métrica trimestral leída semanalmente es ruido.

La cadencia carga valor diagnóstico más allá de los números que reporta. Una cadencia perdida señala que el programa está perdiendo patrocinio ejecutivo antes de que lo haga cualquier KPI. Cuando la revisión trimestral se cae silenciosamente del calendario, el CTO dejó de defender la inversión, y las métricas seguirán uno o dos trimestres después. La cadencia es más barata de monitorear que cualquier KPI, y falla más temprano. Esa combinación es lo que la vuelve el primer indicador a observar, por delante del dashboard que se supone que debe alimentar.

> DEFINICIÓN
>
> La cadencia es la advertencia más temprana que tienes. Una métrica puede caer durante un trimestre entero antes de que alguien actúe. Una cadencia que se salta te dice esta semana que el patrocinio se está escurriendo. Observa el calendario antes de observar el dashboard.

## Postura de gobernanza: gates HITL, límites de public preview e indemnización de IP

Tres controles de gobernanza mantienen el programa defendible a escala enterprise. El primero son los gates human-in-the-loop. Descuenta de las horas brutas ahorradas el tiempo de revisión que exige la postura de cada persona, típicamente 15 a 35 por ciento. Ese descuento no es un número para esconder. Es la cifra honesta que reporta el business case, y una persona con un descuento mayor no

está fallando. Es una persona cuyo output carga más riesgo y correctamente merece más escrutinio. El segundo control son los límites de public preview. Los agentes L4 personalizados están en public preview y los L5 Agent Skills son experimentales, y las funciones en public preview no deben ir a producción en workloads críticos. La regla se dice con claridad para que una ganancia de métrica nunca contrabandee una dependencia sin soporte hacia un camino regulado.

El tercer control es la indemnización de IP. Confírmala antes de escalar, vía GitHub Copilot Enterprise y el Azure OpenAI Customer Copyright Commitment, para que el código generado cargue un respaldo contractual. Las trazas de evidencia completan la postura. GitHub Advanced Security, la Audit Log API, Microsoft Purview y Microsoft Sentinel convierten el compliance de una auditoría manual semestral en un proceso automatizado continuo con registro completo. La gobernanza es un atributo por persona, no un interruptor único del programa. Un SRE corriendo runbooks ejecutables contra producción y un UX designer generando prototipos descartables están en niveles de riesgo distintos y cargan gates distintos. La postura es lo que permite a la misma plataforma servir un workload regulado y uno exploratorio sin colapsarlos en una sola política.

# Referencias

Fuentes primarias para el modelo de métricas, los controles de IA que mueven cada métrica y la postura de gobernanza.


- DORA 2025 Research, benchmarks de entrega de elite usados para enmarcar cycle time y tasa de fallo en cambios.
- GitHub Copilot Documentation, agent mode, coding agent, custom agents y la Copilot Metrics API usada como señal de origen.
- Customize AI responses in VS Code, los niveles de personalización que definen los límites de preview de los niveles 4 y 5.
- Research: quantifying GitHub Copilot's impact on developer productivity and happiness, la evidencia de velocidad de tarea y tiempo de PR detrás de la métrica cycle time.
- Research: quantifying GitHub Copilot's impact in the enterprise with Accenture, retención enterprise del código sugerido, evidencia para el tracker de adopción.

- Microsoft Learn, Microsoft Purview, Microsoft Sentinel y Defender for DevOps para trazas de evidencia y gobernanza.

---

**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 CONVERSIÓN Y CIERRE

# Convirtiendo horas ahorradas en dólares: la Acción de Conversión y sus cuatro palancas.

Un desarrollador que ahorra seis horas por semana todavía no le ahorró dinero a la empresa. Las horas son reales. Los dólares no lo son, hasta que una Acción de Conversión deliberada redirige el tiempo recuperado. Este capítulo cierra el business case. Nombra por qué las ganancias se fugan, las cuatro palancas que convierten horas en dinero, cómo costear el programa en GitHub AI Credits a un centavo cada uno, y cómo el business case ola por ola y la calculadora complementaria reportan una única cifra defendible.

## La fuga: por qué las ganancias de productividad se evaporan sin una Acción de Conversión

Un desarrollador que ahorra seis horas por semana, la cifra que McKinsey reporta para el desarrollo asistido por IA, todavía no le ahorró dinero a la empresa. Las horas son reales. El ahorro no lo es, hasta que algo deliberado ocurre con el tiempo recuperado. Esa es la brecha que la mayoría de los business cases de IA se saltan. Gartner lo dice con todas las letras en el informe G00837723, Productivity Gains Alone Won't Pay for AI: el tiempo ahorrado se fuga de regreso a retrabajo, overhead de validación, cambio de contexto y capacidad no utilizada, a menos que una Acción de Conversión lo redirija. La fuga es el estado por defecto. Libradas a su suerte, las horas recuperadas se vuelven a llenar con reuniones, revisiones más largas y tiempo ocioso. El gráfico de productividad sigue viéndose bien. El estado financiero nunca se mueve.

El framework trata la conversión como campo obligatorio, no opcional. Cada una de las 24 personas de este playbook declara una Acción de Conversión explícitamente, junto a su unidad de trabajo y su postura de gobernanza. La cadena se escribe como fórmula para sobrevivir a una revisión financiera. Tiempo inicial ahorrado, menos la

fuga de productividad, menos el valor no realizado, igual a la productividad real ganada. Esa cifra se multiplica por una Acción de Conversión para llegar al ROI financiero en dólares. Una persona sin Acción de Conversión produce un número de productividad y cero dólares. Un CFO tiene razón en desconfiar de cualquier business case que se detenga en el número de productividad.

## La cadena de conversión y sus cuatro palancas

Cuatro palancas convierten la productividad real ganada en dinero. La primera es el descuento HITL. Las horas brutas ahorradas nunca son netas, porque un humano todavía revisa la salida del agente. El tiempo de revisión lo fija la postura de gobernanza de la persona y suele estar entre 15 y 35 por ciento del ahorro bruto. Un agente explicador de COBOL bajo revisión ligera conserva la mayor parte de sus horas. Un flujo de AppSec detrás de un gate estricto conserva menos. Descontar primero mantiene honesto el resto del cálculo.

La segunda palanca es el reúso de capacidad. Las horas recuperadas tienen que aterrizar en algo medible: un backlog de innovación, negociación de dependencias, descubrimiento de cliente o refinamiento de alcance. Si el destino no se puede nombrar, se asume que las horas se fugaron. La tercera palanca es la reducción del costo de error. Un defecto detectado en el momento del pull request cuesta casi nada. El mismo defecto detectado en producción puede activar una penalidad contractual. Adelantar la detección multiplica el valor de cada hora que el agente ahorró. La cuarta palanca son las ganancias de throughput: más releases por trimestre con el mismo headcount, lo que se lee como más ingreso por ingeniero. Esa es la razón que una revisión de CFO conserva. La investigación de GitHub sitúa la señal bruta en 55 por ciento más rápido en la finalización de tareas y un tiempo de ciclo de pull request recortado de 9,6 a 2,4 días, y el estudio de Accenture reporta 88 por ciento del código sugerido retenido. Esos son insumos. Las cuatro palancas los convierten.

### Qué palanca lidera, por ola

Ninguna palanca por sí sola carga el business case, y el mix cambia a lo largo del programa. La Wave 1 se apoya en el reúso de capacidad. La Wave 2 suma renegociación de proveedor y de contrato. La Wave 3 suma incremento de ingreso

vía menor time-to-value. La Wave 4 suma evitación de riesgo y ahorro de retención guiado por eNPS. Cada persona declara qué palancas le aplican, así el doble conteo queda visible y se puede eliminar.

> DEFINICIÓN
>
> Las cuatro palancas, en orden: descontar las horas brutas por la revisión humana, aterrizar la capacidad restante en algo medible, poner precio a los defectos detectados temprano y contar los releases extra entregados con el mismo headcount. Salta cualquiera y la cifra en dólares queda sobreestimada.

## Costeando el programa en GitHub AI Credits, un crédito equivale a un centavo

Un business case necesita un lado de costo, y desde el 1 de junio de 2026 GitHub tarifica el uso de agentes en AI Credits, donde un crédito equivale a un centavo. Eso convierte el lado de costo en aritmética, no en estimación. Las ejecuciones en agent mode, las tareas del coding agent y las llamadas al modelo consumen créditos del plan. Como un crédito es 0,01 dólar, un equipo lee su consumo mensual de créditos en la pantalla de billing y lo expresa en dólares sin planilla. El denominador de la razón de ROI queda tan defendible como el numerador.

El modelo elegido en el selector de Copilot es una palanca de costo por sí misma. El selector ofrece Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro y los modelos GPT-5.x. El trabajo de rutina no necesita la opción más cara. Una ejecución con Haiku 4.5 o Sonnet 5 cuesta menos créditos que una con Opus 4.8 y muchas veces basta para scaffolding, generación de pruebas o una primera revisión. Los modelos más pesados se reservan para las tareas que justifican sus créditos. La higiene de contexto también reduce el costo: un repositorio con un archivo .github/copilot-instructions.md mantenido y los servidores MCP correctos conectados gasta menos créditos por tarea, porque el agente deja de rederivar un contexto que pudo haber leído una vez. Menos reconstrucción significa menor consumo de créditos, lo que mejora el lado de costo de cada persona.

# El business case ola por ola y la calculadora complementaria por mix de personas

El programa se costea y se convierte ola por ola, porque las palancas y los patrocinadores cambian con cada ola. La Wave 1 cubre siete personas en los meses uno a tres, convierte principalmente por reasignación de capacidad y se defiende por uso diario por encima del 70 por ciento en 60 días. La Wave 2 cubre ocho personas en los meses cuatro a seis y suma renegociación de proveedor y outsourcing. La Wave 3 cubre ocho personas en los meses siete a nueve y desplaza el cubo primario hacia crecimiento de ingreso vía menor time-to-value. La Wave 4 trae al Site Reliability Engineer en los meses diez a doce y convierte por evitación de riesgo y retención guiada por eNPS. Cada ola nombra un patrocinador ejecutivo y un indicador predictivo, así el business case es auditable antes de que arranque la siguiente ola.

La calculadora complementaria combina todo esto por mix de personas. Dale el conteo de cada persona, la ola, la postura de gobernanza que fija el descuento HITL y las Acciones de Conversión elegidas, y devuelve una cifra en dólares que ya neta la fuga. Cambia el mix y la cifra cambia con él. Corre sobre el mismo modelo publicado de métricas guiadas por resultado del SDLC que ancla las 24 personas, así la calculadora y el playbook no pueden divergir. La salida es un único número sobre el que un comité de steering puede actuar.

> DEFINICIÓN
>
> Las horas ahorradas son un insumo, no un resultado. El resultado es la cifra que sobrevive a una revisión de steering: productividad real ganada, descontada por la revisión humana, multiplicada por una Acción de Conversión nombrada, costeada contra AI Credits a un centavo cada uno, y expresada en dólares.

# Referencias

Fuentes primarias para el modelo de conversión, los baselines de productividad que convierte y el lado de costo en GitHub AI Credits.

- GitHub Copilot Documentation, la referencia para agent mode, el coding agent y el billing en AI Credits donde un crédito equivale a un centavo.
- GitHub, Quantifying GitHub Copilot's impact on developer productivity and happiness, la señal bruta de productividad: 55 por ciento más rápido en la finalización de tareas.
- GitHub, Quantifying GitHub Copilot's impact in the enterprise with Accenture, evidencia enterprise: 88 por ciento del código sugerido retenido en las versiones finales.
- DORA, State of DevOps 2025, baselines de desempeño de entrega para la palanca de throughput.
- VS Code, Customize AI responses, archivos de instrucciones y agentes personalizados que reducen el consumo de créditos por tarea.
- Agentic DevOps SDLC, 24 Personas, fuente complementaria, el detalle completo de las personas y el modelo de conversión en que se apoya este capítulo.

---

**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