AI IMPACT ACROSS THE SDLC · PLAYBOOK

AI changes twenty-four roles, not one, and hours saved are not dollars saved.

AI agents do not change one role. They change twenty-four, at different intensities and at different times, which is why a single AI roadmap for all of engineering is a category error. This playbook maps the impact persona by persona across a 5,000-engineer enterprise SDLC, configures GitHub Copilot for each role, sequences adoption in four waves across twelve months, and ties every recovered hour to a Conversion Action, because time saved is not money saved until a deliberate step turns it into cost reduction, revenue growth, or a better employee experience.

AUTHOR

Paula Silva

ROLE

AI-Native Software Engineer

CONTACT

in/paulanunes

READING TIME

~ 45 minutes, end to end

24	6	4
PERSONAS	CHAPTERS	ADOPTION WAVES

12

MONTHS TO FULL ADOPTION

CHAPTERS

Chapter 00

Twenty-four roles, not one roadmap

Why a single AI roadmap is a category error, why a persona is a unit of work with a measurable outcome, why time saved is not money saved, and the scope: a 5,000-engineer enterprise SDLC with legacy roles included.

Context →

Chapter 01

The persona map in four clusters

The 24 SDLC personas grouped into Strategy and discovery, Build and quality, Run and reliability, and Govern and grow, each with the GitHub Copilot capability it needs and the before and after it measures.

Personas →

Chapter 02

Configuring GitHub Copilot per persona

The five customization levels, agent mode with MCP grounding, choosing a model in the Copilot picker per task, and the level 1 to level 2 plateau where most teams stall.

Method →

Chapter 03

Four waves across twelve months

The execution timeline: immediate cost reduction, supplier optimization and the legacy estate, revenue growth through shorter time-to-value, then resilience and employee experience.

Adoption →

Chapter 04

Outcome-driven metrics and governance

The five Outcome-Driven Metrics that carry the impact, the AI control that moves each one, cadence as a leading indicator of sponsorship, and the governance posture that keeps it defensible.

Metrics →

Chapter 05

Turning hours saved into dollars

Why gains leak without a Conversion Action, the four levers that turn hours into money, costing the program in GitHub AI Credits, and the wave-by-wave business case.

Conversion →

READING TIPS

> **KEYBOARD SHORTCUTS**
>
> Press `/` to search. Use `j` and `k` to move between chapters. Press `t` to toggle theme. Press `g` to go to top.

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 CONTEXT AND THESIS

Twenty-Four Roles, Not One Roadmap

AI agents do not change one role. They change twenty-four, at different intensities and at different times. That single fact breaks the most common enterprise plan, the one roadmap for all of engineering. This chapter sets the frame for the rest of the playbook: why one roadmap is a category error, why a persona is the right unit of analysis, why time saved is not money saved, and what a 5,000-engineer enterprise SDLC actually contains, legacy roles included.

Why a single AI roadmap is a category error across the SDLC

AI agents do not change one role. They change twenty-four roles, at different intensities and at different times. A Product Manager, a Staff Engineer, a Site Reliability Engineer, a Data Engineer, and a Compliance Officer each face a different agentic horizon. Each needs a different combination of GitHub Copilot, Microsoft 365, and Azure services to reach value. Treating "AI for engineering" as one roadmap is a category error. It assumes a uniform starting point, a uniform task shape, and a uniform payoff curve. None of those hold across the software lifecycle.

The error is expensive because it drives the wrong plan. A single roadmap funds the loudest use case, usually code generation, and starves the roles where the return is larger and slower to appear. A Requirements Engineer who cross-references transcripts against institutional knowledge, or a COBOL developer who turns thirty years of undocumented behavior into queryable documentation, does not show up in a code-completion dashboard. When one metric governs the whole program, those wins are invisible, and the program looks like it underperformed. The fix is not a better average. The fix is to map impact role by role, so each persona gets the tooling, the customization depth, and the timeline that its work actually requires.

A persona as a unit of work, a decision set, and a measurable outcome

A persona is not a job title. It is a unit of work, a set of decisions, and a measurable outcome. That definition is what makes the framework operational. A job title tells you nothing about where an agent helps. A unit of work tells you which tasks the agent can take, which decisions stay with the human, and which outcome you measure at the end. For the Modern Software Developer the unit is a feature branch, the decisions are design and review, and the outcome is throughput. For the Site Reliability Engineer the unit is an incident, the decisions are diagnosis and remediation, and the outcome is time to recover.

GitHub Copilot gives each persona a different point of contact. Some live in agent mode inside the editor, delegating whole issues to a coding agent that opens a pull request with tests. Some live in chat, asking questions of a codebase they did not write. The customization hierarchy is what tunes the agent to the role: a repository-wide .github/copilot-instructions.md, path-scoped instruction files, an AGENTS.md for project behavior, custom agents for a named persona, and MCP servers that connect the agent to the systems the persona already uses. The model in the Copilot picker is a per-persona choice too. Haiku 4.5 for fast, cheap edits; Opus 4.8 or Claude Fable 5 for hard reasoning across many files. One unit of work, one decision set, one outcome, one configuration.

The thesis: time saved is not money saved

Productivity gains alone do not pay for AI. This is the thesis the rest of the playbook defends. An agent that saves a developer six hours a week has not saved the company anything yet. Time saved leaks back into rework, validation overhead, context switching, and meetings unless a deliberate step redirects it. Gartner research states the point plainly: productivity gains on their own will not pay for AI. The gap between hours recovered and dollars booked is where most AI business cases quietly fail, and it is why a skeptical CFO is often right to disbelieve the number on the slide.

The discipline that closes the gap is a Conversion Action. Every persona has to declare how its recovered hours become one of three outcomes: cost reduction,

revenue growth, or improved employee experience. A Tech Lead who recovers review time converts it by shipping features sooner, not by holding more meetings. There is a cost on the other side of the ledger, and GitHub Copilot makes it explicit. Agent runs meter against GitHub AI Credits, where one credit is one cent since June 1 2026. That price tag is the point. It forces the honest question the thesis demands: did this run produce more value than it consumed?

> DEFINITION
>
> Time saved is not money saved. Every persona must declare a Conversion Action that turns recovered hours into cost reduction, revenue growth, or improved employee experience. Without it, the hours leak back into rework and the business case does not close.

Scope: a 5,000-engineer enterprise SDLC, legacy roles included

The framework is scoped to a real enterprise, not a demo. The reference organization is a 5,000-engineer software delivery function, organized across eight phases of the software lifecycle, from client and contract management through governance, security, and compliance. The twenty-four personas are the roles that actually staff that function. They are not archetypes chosen to make AI look good. They include the roles the generic playbooks skip. Enterprises at this scale are the norm, not the exception: more than ninety percent of the Fortune 100 use GitHub.

Legacy roles are in scope, not around it

Most AI adoption stories quietly assume greenfield code and modern stacks. A 5,000-engineer enterprise does not look like that. It runs decades of COBOL, Natural/ADABAS, DB2, and mainframe systems, maintained by senior developers who are retiring and taking undocumented knowledge with them. The framework puts three legacy developer profiles in scope on purpose. GitHub Copilot supports COBOL and Natural syntax, and a custom agent can cut program comprehension from days to hours. The higher-value outcome is permanent knowledge capture: the context that lived in one engineer's head becomes documentation the whole team

can query. A roadmap that covers only the modern roles is not a roadmap for this enterprise. It is a roadmap for the part of it that was already easy.

References

Primary sources for the twenty-four persona framework, the empirical productivity evidence, and the GitHub Copilot capabilities this playbook configures per role.

- Agentic SDLC Personas, companion framework site, the original source of the twenty-four persona framework this playbook builds on.
- The twenty-four personas in detail, each role with its tasks, tools, and before and after data.
- GitHub Copilot Documentation, agent mode, the coding agent, custom agents, and the repository instruction files this framework configures per persona.
- Customize AI responses in VS Code, instruction files, AGENTS.md, custom agents, and agent skills, the five levels of Copilot customization.
- GitHub, Quantifying GitHub Copilot's impact on developer productivity and happiness, the 55% task-speed result cited for the developer personas.
- GitHub, Quantifying GitHub Copilot's impact in the enterprise with Accenture, the enterprise study behind the 88% code-retention figure.
- GitHub Universe 2024 press release, more than 90% of the Fortune 100 use GitHub, the scale this framework targets.
- DORA 2025 State of DevOps research, delivery performance, deployment frequency, and change failure rate.

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 WHO IS AFFECTED

The persona map in four clusters: who AI agents change, and how.

AI agents do not change one role. They change 24, at different intensity at different times. Treating AI for engineering as a single roadmap is a category error, because the Product Manager, the COBOL developer, the Site Reliability Engineer, and the Compliance Officer each face a different agentic horizon. This chapter groups the 24 personas into four clusters by the kind of work AI changes: Strategy and discovery, Build and quality, Run and reliability, and Govern and grow.

The map: 24 personas, four clusters, one unit of work

A persona is a unit of work, a decision set, and a measurable outcome. The 24 personas come from a real enterprise SDLC, and they include legacy roles like COBOL and Natural/ADABAS that generic playbooks skip. Grouping them into four clusters answers a practical question before the role-by-role detail begins: which kind of work does an AI agent actually change, and who owns it. Each cluster shares an impact pattern, a set of primary tools, and a risk that decides whether the gain is real or leaks away.

The lens throughout is GitHub Copilot. Agent mode runs the multi-step work. A .github/copilot-instructions.md file and AGENTS.md carry the conventions each repository already encodes, so generated output matches corporate standards instead of guessing at them. MCP servers connect the agent to the systems a persona lives in, from Azure Boards to the corporate knowledge base. Billing runs on GitHub AI Credits, where one credit equals one cent, in effect since June 1, 2026, so every persona maps to a cost line a CFO can read. The model picker holds Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro, and the GPT-5.x family, and the right model is chosen by the task, not by the role.

Strategy and discovery: compressing meeting and synthesis overhead

This cluster covers the Business Manager, Project Manager, Product Owner, Requirements Engineer, Enterprise and Solution Architect, Tech Lead, Scrum Master, and UAT Analyst. Their shared bottleneck is not code. It is the overhead of meetings and synthesis. The impact pattern is compression. Manual meeting minutes that took 30 to 40 minutes become 3 to 5 minutes of Copilot-generated notes from a Teams transcript. Weekly status compilation that took 4 to 5 hours becomes a 30 to 60 minute agent-assisted report pulled from Azure DevOps Boards, GitHub Projects, and Power BI.

The strategic gain is not the minutes saved. It is where the recovered time goes: scope analysis, dependency negotiation, and stakeholder discovery, the work that only a person can do well. That redirection is also the risk. If capacity reallocation is not enforced in the operating cadence, the recovered time leaks straight back into more status meetings. Primary tooling for this cluster is Microsoft 365 Copilot, Copilot Studio, GitHub Projects with Ask mode, Azure DevOps Boards, and Power BI.

Build and quality: from typing code to specifying intent

This is the largest cluster: the Modern Software Developer, QA Engineer, Legacy Developer, COBOL Developer, Natural/ADABAS Developer, Application Security Engineer, Data Engineer, and ML Engineer. The impact pattern is a shift in the unit of work, from typing code to specifying intent. The keystroke stops being the unit; the spec becomes it. Personalization through instruction files, AGENTS.md, custom agents, and agent skills makes generated code match corporate standards on the first try, which removes most review iterations. Modern Developer throughput moves from about 40 to 80 or 100 function points per month, with better test coverage than before.

Two hard gates govern this cluster. Application security stays a required check, not a recommendation, and the productivity gain must be tied to a declared Conversion Action, or it leaks back into rework instead of reaching the balance sheet. Primary tooling is GitHub Copilot in Ask, Edit, Agent, Plan, and Coding Agent modes, GitHub

Copilot Code Review, GitHub Advanced Security, Microsoft Fabric, and Azure AI Foundry.

Legacy capture: COBOL and Natural/ADABAS

The higher-value outcome in this cluster is not new code. It is knowledge capture. Senior developers who carry 20 to 30 years of undocumented system behavior are retiring, and their context has never been written down. GitHub Copilot supports COBOL and Natural syntax natively. An explainer agent brings program comprehension from 2 to 5 days down to 2 to 4 hours, and the productivity gain on new code generation reaches 85%. The permanent result is that the knowledge which lived in one engineer becomes queryable documentation that outlives the retirement.

> DEFINITION
>
> The build cluster changes the unit of work, from the keystroke to the spec. The higher value is not the speed of new code. It is turning the legacy knowledge that lived in one person into documentation the organization can query.

Run and reliability: from reactive firefighting to proactive engineering

This cluster is the DevOps and Platform Engineer, Database Administrator, Release Manager, and Site Reliability Engineer. The impact pattern is a move from reactive firefighting to proactive engineering. Incident triage drops from 15 to 30 minutes to 3 to 5 minutes with an incident-commander agent. Post-mortems shrink from 4 to 8 hours to 1 to 2 hours. Release notes go from 3 to 5 hours per release to 30 to 60 minutes. Toil, the manual repetitive work a Site Reliability Engineer measures, moves from 40 to 50% of capacity to under 30%.

The Site Reliability Engineer shows the sharpest numbers in the whole framework. Automated alert correlation cuts incident detection from 5 to 10 minutes to 30 seconds. Root cause diagnosis drops from 2 to 3 hours of manual investigation to 10 minutes of automated analysis. Remediation through executable runbooks takes 2 to

5 minutes against the previous 30 minutes to 2 hours. The governing rule keeps the human in charge: automated remediation passes three supervised runs before promotion, and the agent stays advisory during a live incident while the human commander retains authority. Primary tooling is Azure Monitor, Application Insights, Microsoft Sentinel, and GitHub Actions deployment environments.

Govern and grow: continuous evidence replaces point-in-time audit

The last cluster is the Engineering Manager and IT Manager, the InfoSec and Compliance Officer, the Compliance and Audit Officer, and the UX and Product Designer. The impact pattern is that continuous evidence replaces point-in-time audit. The Engineering Manager owns adoption metrics from day one, which is why this persona sits in the first adoption wave and not the last. InfoSec moves from scanning at pull request time to continuous compliance collection on every push. A semiannual manual audit that took 3 to 4 weeks becomes a continuous automated process with a full evidence trail.

The designer belongs here too. UX research synthesis drops from 2 to 3 days per study to 4 to 8 hours, and accessibility audits become continuous instead of annual. Two risks govern this cluster. Public preview features must not run in production for critical workloads, and IP indemnification must be confirmed before adoption, through GitHub Copilot Enterprise and the Azure OpenAI Customer Copyright Commitment. The GitHub AI Credits ledger helps here: because one credit equals one cent, the audit trail already carries a cost line for every agent action. Primary tooling is GitHub Advanced Security, the GitHub Audit Log API, Microsoft Purview, Microsoft Sentinel, Power BI compliance dashboards, and Azure Policy.

> DEFINITION
>
> Four clusters, one discipline: measure the before and after per persona, tie every recovered hour to a declared outcome, and keep a human in authority where the work carries risk. That is what turns AI strategy into something a 5,000-person engineering organization can execute.

References

Primary sources for the persona impact patterns, the productivity benchmarks, and the GitHub Copilot personalization mechanisms cited in this chapter.

- GitHub, Research: quantifying GitHub Copilot impact on developer productivity and happiness, the source for the 55% faster task completion benchmark.
- GitHub, Research: quantifying GitHub Copilot impact in the enterprise with Accenture, the source for 88% of suggested code retained in final versions.
- GitHub Universe 2024, enterprise Copilot adoption, over 90% of the Fortune 100 use GitHub.
- DORA, 2025 State of DevOps research, elite delivery performance, deployment frequency, and change failure rate.
- GitHub Copilot documentation, agent mode, coding agent, and custom agents.
- Customize AI responses in VS Code, instructions files, AGENTS.md, custom agents, and agent skills.
- Agentic DevOps SDLC, 24 Personas, companion source, the role-by-role source for the four-cluster map.

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 METHOD AND ARCHITECTURE

Configuring GitHub Copilot per persona: five levels, one picker, grounded output.

A generic Copilot helps every persona a little. A configured Copilot changes how a persona works. The difference is not the model. It is the context you give it and the way you ground it. This chapter describes the five customization levels, agent mode with MCP grounding, model choice in the picker, and the plateau where most teams stall.

The five customization levels

GitHub Copilot is configured in five levels, each one adding specificity on top of the last. All five work with Copilot in the IDE, without GitHub Enterprise Cloud. Level 1 is base Copilot: the picker, inline completion, chat, and agent mode, with no knowledge of your conventions. It is generic by design. Level 2 is a .github/copilot-instructions.md file at the repository root. Copilot reads it on session start and applies it to every request in that repository: naming rules, security constraints, and the corporate conventions a new hire would need. This is the single highest-return file in the hierarchy, and it takes an afternoon to write.

Level 3 narrows the scope. Glob-scoped instruction files, .instructions.md with an applyTo pattern, carry rules that apply only to matching files: *.java, *.sql, *.py. A SQL rule does not reach a Java prompt, and a Java rule does not reach a test file. Level 4 is custom agents, declared in .agent.md, each a specialized persona with its own tools and handoffs. The @cobol-explainer agent, the @test-generator agent, and the @code-reviewer agent pass work between themselves as formal handoffs. Level 5 is Agent Skills, defined in SKILL.md: on-demand, task-specific workflows with bundled templates that Copilot loads only when the task calls for them. Custom agents are in public preview; Agent Skills are experimental.

Map the level to the persona

The level a persona needs is set by the work, not by seniority. A Business Manager who wants meeting minutes needs Level 1 and M365 Copilot, nothing more. A Modern Software Developer benefits from Level 2 conventions and Level 3 glob rules for each language in the stack. A COBOL developer needs a Level 4 @cobol-explainer agent to compress program comprehension from days to hours. A Site Reliability Engineer needs Level 5 skills that wrap executable runbooks. Configure the level the persona will actually use, then climb when the ceiling is reached.

Agent mode and MCP grounding

Instruction files tell Copilot what you prefer. Grounding tells Copilot what is true. Agent mode is where the two combine: the agent plans a change across files, runs tools, reads results, and iterates, rather than completing a single line. On its own, agent mode still infers facts about the codebase. The Model Context Protocol closes that gap. An MCP server exposes deterministic facts to the agent on demand: resolved dependency versions, service interfaces, schema definitions, internal API contracts. The agent queries the server instead of guessing, and the generated code matches the corporate standard on the first try instead of the third.

The payoff is measured in rework avoided. GitHub research found that 88% of Copilot-suggested code was retained in final versions when the model had the context it needed. Grounding raises that number by removing the class of errors that come from inference: a deprecated method, a wrong dependency version, a service boundary the agent could not see. Configure the MCP servers once at the platform level, point every persona Copilot at them, and each persona inherits the same source of truth. The instruction files carry preference. The MCP servers carry fact. Agent mode acts on both.

> DEFINITION
>
> Instruction files carry preference. MCP servers carry fact. Agent mode acts on both. Preference without fact produces confident code that violates a constraint the agent never saw. Ground first, then instruct.

Choosing a model in the Copilot picker

The Copilot picker exposes several models, and the right one depends on the task, not on a leaderboard. Claude Fable 5 and Opus 4.8 are the heavy reasoning options: architecture decisions, legacy comprehension, and multi-file refactors where a wrong plan is expensive. Sonnet 5 and the GPT-5.x models are the everyday coding options: they carry most feature work, endpoint scaffolding, and test generation at a lower cost per request. Haiku 4.5 is the fast option for high-volume, low-stakes iterations: completions, small edits, and quick questions where latency matters more than depth. Gemini 3.1 Pro is the long-context and multimodal option, useful when the task spans large files or includes diagrams and screenshots.

Model choice has a cost dimension. Since June 1, 2026, Copilot usage is metered in GitHub AI Credits, where one credit is one US cent. Heavier models draw more credits per request, so matching the model to the task is a budget decision as well as a quality one. The pattern that holds across personas: default to a mid-tier model for routine work, escalate to a heavy model when the plan matters, and drop to a fast model for bulk edits. A Site Reliability Engineer diagnosing an incident and a developer scaffolding a CRUD endpoint should not be on the same model, and with the picker they do not have to be.

One picker, many personas

The picker is per request, so a single developer moves between models within one session. Ask Opus 4.8 to plan a refactor, switch to Sonnet 5 to implement it, and drop to Haiku 4.5 for the mechanical edits. The instruction files and MCP grounding stay constant across the switch, so the context does not reset when the model does. The model changes how the work is reasoned about. The configuration keeps it correct.

Where teams stall: the level 1 to level 2 plateau

Most teams do not stall at Level 4 or Level 5. They stall between Level 1 and Level 2. They install Copilot, use the inline completion, and never write the .github/copilot-instructions.md file. GitHub research reports 55% faster task completion from Copilot, and a team that reaches that number can believe the work is finished. It is not. The file that would encode the team conventions never gets written, and every

session pays for that absence by re-inferring the same rules. The plateau is not technical. The highest-return step simply looks optional, and it is not.

Climbing is a sequence, not a leap. Write the copilot-instructions.md first, because it is one file and it applies everywhere. Add glob-scoped instructions next, one language at a time, as the conventions for each stack stabilize. Introduce custom agents only where a repeated workflow justifies a named persona: the @code-reviewer that recovers review time, the @cobol-explainer that captures retiring knowledge. Add Agent Skills last, for the task-specific workflows a persona runs often enough to template. Each level should earn the next. A team that climbs deliberately reaches a place where Copilot behaves like it belongs to the codebase, because it does.

> DEFINITION
>
> The highest-return step in the whole hierarchy is the cheapest one to skip: writing the first copilot-instructions.md file. A team stuck on base Copilot is paying full price for a fraction of the value.

References

Primary sources for the customization hierarchy, agent mode, and the productivity evidence.

- Customize AI responses in VS Code, the reference for instruction files, custom agents, and Agent Skills.
- GitHub Copilot Documentation, the reference for agent mode, the coding agent, and custom agents.
- GitHub, Quantifying GitHub Copilot impact on developer productivity and happiness, the source for the 55% faster task completion figure.
- GitHub and Accenture, Quantifying GitHub Copilot impact in the enterprise, the source for the 88% code retention figure.
- The 24 personas in detail, the per-persona configuration and outcome matrix this chapter draws on.

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 EXECUTION TIMELINE

Four waves across twelve months: how the 24 personas come online.

The 24-persona framework is not a launch, it is a sequence. Adoption runs in four waves across twelve months, and each wave adds personas whose value depends on the infrastructure the previous wave put in place. This chapter walks the timeline wave by wave, names the personas in each, and ties every wave to the Gartner-validated bucket its returns land in: cost reduction, revenue growth, or improved employee experience.

Wave 1, immediate gains: seven personas, cost reduction

Months 1 to 3. Seven profiles deliver visible return within weeks because their core tasks map onto what GitHub Copilot does in agent mode out of the box: text synthesis, code generation, and data aggregation. Business Manager and Project Manager are the fastest wins outside engineering. Microsoft 365 Copilot turns a 30-minute manual meeting-minutes process into automatic extraction of decisions, action items, and owners from a Teams transcript. The Modern Software Developer shows the sharpest numbers. A REST endpoint that takes 4 to 6 hours of manual work drops to 30 to 45 minutes with agent mode and scaffold generation, and the Coding Agent adds a parallel async lane where a developer delegates 2 to 3 issues per day to a background agent that opens a PR with tests. Function points per month roughly double, from about 40 to a range of 80 to 100, per the Forrester Total Economic Impact study.

QA and Test Lead cuts test-suite generation from 2 to 3 weeks down to 3 to 5 days, with a 75 percent coverage floor enforced automatically. Tech Lead recovers about 60 percent of review time as Copilot Code Review handles the repetitive checks and the lead keeps the architectural decisions. DBA and Engineering Manager round out the wave. The Engineering Manager has a specific job from day one: instrument

adoption. Daily-active-use above 70 percent within 60 days is the predictive indicator for the whole program, and the Manager wires GitHub Copilot Metrics API output into Power BI so anecdotal feedback becomes a business case with NPV and payback period. Billing runs on GitHub AI Credits, one credit priced at one US cent since June 2026, which makes model spend across the Copilot picker a line the Manager can track against hours recovered.

Wave 2, expansion: eight personas, supplier optimization and the legacy estate

Months 4 to 6. Eight profiles come online, and each needs more setup than Wave 1. MCP servers, a repository .github/copilot-instructions.md, and persona-specific AGENTS.md files go in before Copilot reaches full utility. The Requirements Engineer gets an agent that queries the corporate SharePoint knowledge base, cross-references transcripts against institutional knowledge, and flags conflicts before they reach development. The Enterprise Architect takes on the modernization core: decades of COBOL, Natural/ADABAS, DB2, and mainframe code that must eventually move to Spring Boot or microservices. An exploration agent maps the legacy codebase, produces C4 diagrams, and writes feasibility assessments, compressing weeks of analysis into days.

Three legacy developer profiles sit at the center of this wave, and they carry the institutional-knowledge problem. Senior developers holding 20 to 30 years of undocumented system knowledge are retiring. Copilot reads COBOL and Natural syntax natively, and a @cobol-explainer custom agent cuts program comprehension from 2 to 5 days down to 2 to 4 hours. The higher-value outcome is permanent knowledge capture: what lived in the head of a single engineer becomes queryable documentation. DevOps and Platform Architect and InfoSec close the wave. The ROI bucket here is cost reduction through supplier and outsourcing optimization, and the predictive indicator the program tracks is three contracts renegotiated with AI-driven leverage, because a legacy estate an internal team can now read and modify changes the terms of every outsourcing conversation.

Wave 3, specialization: eight personas, revenue growth via shorter time-to-value

Months 7 to 9. Eight specialized profiles come online with domain-specific custom agents that only work once the foundational infrastructure from Waves 1 and 2 is mature. Product Owner manages backlogs of 500 or more items with AI-driven WSJF prioritization and automatic user-story generation. Scrum Master and Agile Coach automate retrospectives, sprint reviews, and cross-team dependency maps. Data Engineer sits at the intersection of 50 or more legacy and modern sources feeding Microsoft Fabric for a thousand or more Power BI users, and pipeline creation time falls by about 80 percent while data-quality monitoring becomes continuous instead of reactive.

ML and AI Engineer closes the feedback loop for the platform: model selection across the Copilot picker, prompt optimization, evaluation frameworks, and AI governance such as hallucination detection and data-leak prevention. Compliance and Audit Officer compresses a 3 to 4 week semiannual manual audit into a continuous process with full evidence trails, a direct response to LGPD, ISO 27001, and sector regulation. UX and Product Designer generates Figma prototypes in minutes and enables same-day stakeholder validation of flows that used to take weeks. That is the through-line of the wave: every persona here shortens time-to-value, which is why the ROI bucket shifts from cost to revenue growth, and the program target is a median time-to-value down by at least 30 percent.

Wave 4, maturity: SRE, resilience, and employee experience

Months 10 to 12. One profile reaches full maturity last, the Site Reliability Engineer, and its numbers are the most striking in the framework. Automated alert correlation cuts incident detection from 5 to 10 minutes down to 30 seconds. Root-cause diagnosis drops from 2 to 3 hours of manual investigation to about 10 minutes of automated analysis. Remediation through executable runbooks takes 2 to 5 minutes against the previous 30 minutes to 2 hours. For systems where downtime costs millions per minute, these are not marginal gains. They change how reliability is engineered.

The ROI bucket for Wave 4 is improved employee experience and resilience, the third Gartner-validated bucket, and the predictive indicator is an eNPS uplift of at least 8 points among AI users. On-call quality of life is not a soft benefit here. It is the retention argument that a CFO review accepts alongside cost and revenue, and it is what keeps the senior engineers whose judgment the whole program depends on.

The program enters continuous refinement

By month 12 the program stops adding personas and starts tuning. Consolidated DORA metrics, ROI reported to the C-level, and an AI-first engineering culture across all 24 roles become the steady state. Custom agents get sharper against real usage, copilot-instructions files get corrected against real output, and MCP integrations get pruned. The waves were sequential and not optional. Refinement is permanent.

> DEFINITION
>
> The four-wave model is sequenced, not simultaneous. Wave 1 lands cost reduction through experience compression, Wave 2 through supplier and outsourcing optimization plus the legacy estate, Wave 3 turns shorter time-to-value into revenue growth, and Wave 4 converts resilience and employee experience into a retention argument. After month 12 the program enters continuous refinement across all 24 personas.

References

Primary sources for the four-wave adoption model, the persona catalog, and the productivity evidence behind the timeline.

- Agentic DevOps SDLC Personas, companion site, the source catalog for the 24 personas and the four-wave adoption roadmap.
- GitHub Copilot Documentation, agent mode, the Coding Agent, and custom agents referenced across all four waves.
- Customize AI responses in VS Code, instructions files, AGENTS.md, custom agents, and agent skills, the customization layers Wave 2 depends on.

- GitHub Research, quantifying GitHub Copilot impact on developer productivity and happiness, the 55 percent faster task completion behind Wave 1 developer gains.
- DORA, 2025 State of DevOps research, the delivery metrics consolidated when the program reaches month 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 MEASURE AND GOVERN

Outcome-driven metrics and governance: what to measure, and how to keep it defensible.

A pilot that cannot survive a CFO review does not scale. This chapter turns adoption into a defensible business case. It names the five Outcome-Driven Metrics that carry most of the persona-level impact, pairs each one with the AI control that moves it, and sets the governance posture that keeps the program auditable at enterprise scale.

The ODM matrix: metric, lever owner, and the outcome the CFO tracks

Per Gartner G00799085 and G00841077, an Outcome-Driven Metric has direct line of sight to a business outcome. Five ODMs carry most of the persona-level impact, and each one names two things beyond the number itself: the team that owns the lever, and the outcome the CFO tracks. Time-to-market is owned by Product, the Tech Lead, and the Release Manager, and it reports to revenue growth. Cycle time is owned by Developers, QA, and DevOps, and it reports to cost reduction and throughput. Defect escape rate is owned by QA, AppSec, and SRE, and it reports to penalty avoidance and NPS. Customer NPS is owned by Product, UX, and Support, and it reports to retention and ACV growth. Revenue per engineer is owned by the Engineering Manager with the CFO partner, and it reports to headcount efficiency.

The ownership column is not decoration. A metric with no owner drifts, and a metric with no CFO-tracked outcome never reaches the business case. The outcome column is the translation layer between engineering and finance. Revenue per engineer is the ratio that survives a CFO review, because it is the only one expressed in the units the finance team uses. Productivity gains alone do not pay for AI, per Gartner G00837723. Every persona must declare a Conversion Action that turns recovered hours into one of three buckets: cost reduction, revenue growth, or

improved employee experience. Without that step, time leaks back into rework and meetings, and the CFO is right to disbelieve the case.

Pair every metric with the AI control that moves it

A metric without a control is anecdotal. A control without a metric has no business defense. So each ODM is paired with the AI control that moves it, the source signal that measures it, and the cadence at which it is read. Time-to-market moves on Spec Kit and Level 4 custom agents, reads from GitHub Releases and Azure DevOps Release, and is reviewed weekly. Cycle time moves on Copilot Code Review running on every PR, reads from GitHub Insights and Azure DevOps Analytics, and is reviewed weekly. Defect escape rate moves on GitHub Advanced Security gates plus a QA agent, reads from Defender for Cloud, Sentinel, and GHAS, and is reviewed monthly. Customer NPS moves on a UX research synthesizer plus an accessibility auditor, reads from Microsoft Clarity and in-product surveys, and is reviewed quarterly. Revenue per engineer moves on an adoption tracker plus a capacity reallocation policy, reads from the GitHub Copilot Metrics API and finance reports, and is reviewed quarterly.

The source signal matters as much as the control. Every metric here reads from a system of record, not a self-report, which is what lets it survive a steering review. Adoption reads from the GitHub Copilot Metrics API, not from a survey. Cost is measurable in the same frame: model usage in agent mode is billed in GitHub AI Credits, where one credit is one cent, in effect since June 1 2026. That makes cost-per-outcome a real number. Divide credits spent by releases shipped or defects caught. Selecting a cheaper model in the Copilot picker for a routine task and a stronger one for a hard task becomes a lever the metric can price.

Cadence discipline is a leading indicator of sponsorship

Cadence is not a reporting formality. It is matched to how fast each metric can move. The Engineering Manager owns the weekly and monthly cadences, which cover time-to-market, cycle time, and defect escape rate, because those move on engineering decisions. The CTO owns the quarterly cadence, which covers

Customer NPS and revenue per engineer, because those move on business decisions. Matching the cadence to the metric keeps every review actionable: a weekly metric read quarterly is stale, and a quarterly metric read weekly is noise.

The cadence carries diagnostic value beyond the numbers it reports. A missed cadence signals that the program is losing executive sponsorship before any KPI does. When the quarterly review quietly slips off the calendar, the CTO has stopped defending the investment, and the metrics will follow one or two quarters later. Cadence is cheaper to monitor than any KPI, and it fails earlier. That combination is what makes it the first indicator to watch, ahead of the dashboard it is supposed to feed.

> DEFINITION
>
> Cadence is the earliest warning you have. A metric can trend down for a full quarter before anyone acts on it. A cadence that gets skipped tells you this week that sponsorship is slipping. Watch the calendar before you watch the dashboard.

Governance posture: HITL gates, public preview limits, and IP indemnification

Three governance controls keep the program defensible at enterprise scale. The first is human-in-the-loop gates. Discount the gross hours saved by the review time each persona requires under its posture, typically 15 to 35 percent. That discount is not a number to hide. It is the honest figure the business case reports, and a persona with a higher discount is not failing. It is one whose output carries more risk and correctly earns more scrutiny. The second control is public preview limits. Level 4 custom agents are in public preview and Level 5 Agent Skills are experimental, and public preview features must not be deployed to production for critical workloads. The rule is stated plainly so that a metric gain never smuggles an unsupported dependency into a regulated path.

The third control is IP indemnification. Confirm it before scaling, through GitHub Copilot Enterprise and the Azure OpenAI Customer Copyright Commitment, so that

generated code carries a contractual backstop. Evidence trails complete the posture. GitHub Advanced Security, the Audit Log API, Microsoft Purview, and Microsoft Sentinel turn compliance from a semiannual manual audit into a continuous automated process with a full record. Governance is a per-persona attribute, not a program-wide switch. An SRE running executable runbooks against production and a UX designer generating throwaway prototypes sit at different risk tiers and carry different gates. The posture is what lets the same platform serve a regulated workload and an exploratory one without collapsing them into one policy.

References

Primary sources for the metrics model, the AI controls that move each metric, and the governance posture.

- DORA 2025 Research, elite delivery benchmarks used to frame cycle time and change failure rate.
- GitHub Copilot Documentation, agent mode, coding agent, custom agents, and the Copilot Metrics API used as source signals.
- Customize AI responses in VS Code, the personalization levels that define the Level 4 and Level 5 preview limits.
- Research: quantifying GitHub Copilot's impact on developer productivity and happiness, the task speed and PR cycle time evidence behind the cycle time metric.
- Research: quantifying GitHub Copilot's impact in the enterprise with Accenture, enterprise retention of suggested code, evidence for the adoption tracker.
- Microsoft Learn, Microsoft Purview, Microsoft Sentinel, and Defender for DevOps for evidence trails and governance.

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 CONVERSION AND CLOSE

Turning hours saved into dollars: the Conversion Action and its four levers.

A developer who saves six hours a week has not saved the company any money yet. The hours are real. The dollars are not, until a deliberate Conversion Action redirects the recovered time. This chapter closes the business case. It names why gains leak, the four levers that turn hours into money, how to cost the program in GitHub AI Credits at one cent each, and how the wave-by-wave case and the companion calculator report a single defensible figure.

The leak: why productivity gains evaporate without a Conversion Action

A developer who saves six hours a week, the figure McKinsey reports for AI-assisted development, has not saved the company any money yet. The hours are real. The savings are not, until something deliberate happens to the recovered time. This is the gap most AI business cases skip. Gartner states it plainly in report G00837723, Productivity Gains Alone Won't Pay for AI: time saved leaks back into rework, validation overhead, context switching, and unused capacity unless a Conversion Action redirects it. The leak is the default. Left alone, recovered hours refill with meetings, longer reviews, and idle slack. The productivity chart still looks good. The financial statement never moves.

The framework treats conversion as a required field, not an optional one. Every one of the 24 personas in this playbook declares a Conversion Action explicitly, alongside its unit of work and its governance posture. The chain is written as a formula so it survives a finance review. Initial time saved, minus the productivity leak, minus value not realized, equals actual productivity gained. That figure is multiplied by a Conversion Action to reach financial ROI in dollars. A persona with no Conversion Action produces a productivity number and zero dollars. A CFO is right to disbelieve any business case that stops at the productivity number.

The conversion chain and its four levers

Four levers turn actual productivity gained into money. The first is the HITL discount. Gross hours saved are never net, because a human still reviews the agent output. The review time is set by the persona governance posture and typically runs 15 to 35 percent of the gross saving. A COBOL explainer agent under light review keeps most of its hours. An AppSec workflow behind a hard gate keeps fewer. Discounting first keeps the rest of the calculation honest.

The second lever is capacity reuse. Recovered hours have to land somewhere measurable: an innovation backlog, dependency negotiation, customer discovery, or scope refinement. If the destination cannot be named, the hours are assumed to have leaked. The third lever is error cost reduction. A defect caught at pull-request time costs close to nothing. The same defect caught in production can trigger a contract penalty. Moving detection earlier multiplies the value of every hour the agent saved. The fourth lever is throughput gains: more releases per quarter at constant headcount, which reads as more revenue per engineer. That is the ratio a CFO review keeps. GitHub research puts the gross signal at 55 percent faster task completion and a pull-request cycle time cut from 9.6 to 2.4 days, and the Accenture study reports 88 percent of suggested code retained. Those are inputs. The four levers convert them.

Which lever leads, by wave

No single lever carries the whole case, and the mix shifts across the program. Wave 1 leans on capacity reuse. Wave 2 adds supplier and contract renegotiation. Wave 3 adds revenue uplift from shorter time-to-value. Wave 4 adds risk avoidance and retention savings driven by eNPS. A persona states which levers apply to it, so double counting stays visible and can be removed.

> DEFINITION
>
> The four levers, in order: discount the gross hours for human review, land the remaining capacity somewhere measurable, price in the defects caught early, and count the extra releases shipped at constant headcount. Skip any one and the dollar figure is overstated.

Costing the program in GitHub AI Credits, one credit equals one cent

A business case needs a cost side, and since June 1, 2026, GitHub prices agent usage in AI Credits, where one credit equals one cent. That turns the cost side into arithmetic rather than estimation. Agent mode runs, coding agent tasks, and model calls draw credits against the plan. Because one credit is 0.01 dollars, a team reads its monthly credit consumption off the billing view and states it in dollars without a spreadsheet. The denominator of the ROI ratio becomes as defensible as the numerator.

The model chosen in the Copilot picker is a cost lever of its own. The picker offers Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro, and the GPT-5.x models. Routine work does not need the most expensive option. A Haiku 4.5 or Sonnet 5 run costs fewer credits than an Opus 4.8 run and is often enough for scaffolding, test generation, or a first-pass review. The heavier models are held for the tasks that earn their credits. Context hygiene lowers the cost too: a repository with a maintained .github/copilot-instructions.md file and the right MCP servers wired in spends fewer credits per task, because the agent stops re-deriving context it could have read once. Lower reconstruction means a lower credit draw, which improves the cost side of every persona.

The wave-by-wave business case and the companion calculator by persona mix

The program is costed and converted wave by wave, because the levers and the sponsors change with each wave. Wave 1 covers seven personas across months one to three, converts mainly through capacity reallocation, and is defended by daily active use above 70 percent within 60 days. Wave 2 covers eight personas across months four to six and adds supplier and outsourcing renegotiation. Wave 3 covers eight personas across months seven to nine and shifts the primary bucket to revenue growth through shorter time-to-value. Wave 4 brings the Site Reliability Engineer online across months ten to twelve and converts through risk avoidance and eNPS-driven retention. Each wave names an executive sponsor and a predictive indicator, so the case is auditable before the next wave starts.

The companion calculator combines all of this by persona mix. Give it the count of each persona, the wave, the governance posture that sets the HITL discount, and the chosen Conversion Actions, and it returns a dollar figure that already nets out the leak. Change the mix and the figure changes with it. It runs on the same published SDLC outcome-driven metrics model that anchors the 24 personas, so the calculator and the playbook cannot drift apart. The output is one number a steering committee can act on.

> DEFINITION
>
> Hours saved are an input, not a result. The result is the figure that survives a steering review: actual productivity gained, discounted for human review, multiplied by a named Conversion Action, costed against AI Credits at one cent each, and stated in dollars.

References

Primary sources for the conversion model, the productivity baselines it converts, and the GitHub AI Credits cost side.

- GitHub Copilot Documentation, the reference for agent mode, the coding agent, and AI Credits billing where one credit equals one cent.
- GitHub, Quantifying GitHub Copilot's impact on developer productivity and happiness, the gross productivity signal: 55 percent faster task completion.
- GitHub, Quantifying GitHub Copilot's impact in the enterprise with Accenture, enterprise evidence: 88 percent of suggested code retained in final versions.
- DORA, State of DevOps 2025 research, delivery performance baselines for the throughput lever.
- VS Code, Customize AI responses, instructions files and custom agents that lower the credit draw per task.
- Agentic DevOps SDLC, 24 Personas companion source, the full persona detail and the conversion model this chapter draws on.

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