BO·TT·EGA
IT minds

9 092 development sessions
from 44 users

24.14B total tokens paid
95% cached

$19 550.06 monthly cost if using only
premium models

85% cost reduction just by
selecting right model

$16 534.12 potential savings vs
premium-only approach

B = US billion (10^9) tokens

EXECUTIVE BRIEFING

AI Coding Agent Cost Optimization

Operational Data Analysis & Cost Projections
Following **GitHub Copilot**'s Transition to **Usage-Based Billing**

Cut AI coding costs by 85% without sacrificing code quality

Results from working on enterprise systems — proven on 9 092 real-world development sessions

To: Board of Directors / Executive Management
Date: July 1, 2026
Reporting Window: June 2026 — Full 30-Day Cycle
Prepared by: Bottega IT Minds — AI Transformations

THE BOTTOM LINE

Benchmarks lie, invoices don't

The right model choice + harness engineering = 84.2–84.7% cost savings with production-grade output.

This report analyzes 9 092 development sessions from 44 engineers working on real enterprise code. It establishes a baseline for enterprise-wide deployment under GitHub Copilot's new usage-based billing.

Why this matters: On June 1, 2026, Microsoft moved **GitHub Copilot** from flat-rate subscriptions to **Usage-Based Billing (UBB)** — token-based pricing for Input, Output, and Cached Input.

Scope: This analysis covers models in the GitHub Copilot catalog. Open-source models (Llama, DeepSeek) on private infrastructure are excluded.

Benchmarks vs cost: Opus scores ~11pp higher on benchmarks (per vendors) while costing 6.5× more than Mini. Haiku trails Mini on 3 of 4 benchmarks yet costs 1.3× more. GPT-5.5 costs a whopping 12.8× more than Mini.

Benchmark	GPT-5.4 Mini	Haiku 4.5	Opus 4.8	GPT-5.5 (long)
SWE-bench Verified	76.0%	73.3%	88.6%	82.6%
Terminal-Bench	60.0%	41.0%	74.6%	82.7%
OSWorld-Verified	72.1%	50.7%	83.4%	78.7%
GPQA Diamond	88.0%	71.2%	93.6%	93.6%
Average pp vs Mini	**—**	**-15.0pp**	**+11.0pp**	**+10.4pp**
Total cost vs Mini	**1×**	**1.3×**	**6.5×**	**12.8×**

SECTION 2

Key Strategic Insights

Premium models cost more without proportional gains

Claude Opus costs 6.5× more than GPT-5.4 Mini but benchmarks show only ~11pp improvement. Smaller models iterate faster, which directly accelerates development velocity.

Smart operators beat big models

A small model with precise codebase mapping, architecture docs, and clear specs outperforms a large model left to guess.

Context caching is the #1 cost lever

Cached tokens account for *95% of all volume* in agentic loops. Models with aggressive cache discounts win on invoice.

One terminal fix saves double digits

Bash commands drive 41.8% of tool costs — from verbose logs flooding the context window. Savings in the tens of dollars per user per month.

← Benchmark sources: benchmarklist.com, Anthropic (evals.report), OpenAI / DataCamp. Opus 4.8 Terminal-Bench 2.1 result from allthings.how. Cost multiplier based on actual monthly usage (24.14B tokens).

HOW IT WORKS

Three Token Categories

Example: GPT-5.4 Mini — Input $0.75/M • Cached $0.075/M • Output $4.50/M

One session: 100K Input + 900K Cached + 50K Output =

(0.1 × $0.75) + (0.9 × $0.075) + (0.05 × $4.50) = $0.37

Every AI interaction consumes tokens in three categories, billed per million (M) tokens:

Input

Prompts, repo context, and instructions sent *to* the model.

Cached Input

Reused context from memory cache. Billed at *~90% discount* vs. standard input.

Output

Code, reasoning, and analysis written *by* the model. Highest rate.

MONTHLY DATA

User Metrics

Actual volume from 44 users over 30 days, with cost projections across model tiers.

95%
cached tokens in paid volume

85%
savings with Mini vs. Opus ($3 015.94 vs. $19 550.06)

0.5pp
variability in savings across users

User	Sessions	Messages	Input	Cached	Output	GPT-5.4 Mini	Haiku 4.5	Claude Opus
User 1	244	9 489	46.40M	644.00M	2.20M	$93.00	$121.80	$609.00
User 2	483	19 537	45.30M	1 184.80M	6.20M	$150.73	$194.78	$973.90
User 3	67	2 246	7.20M	105.40M	0.60M	$16.00	$20.74	$103.70
User 4	89	3 080	33.10M	151.40M	1.23M	$41.72	$54.39	$271.95
User 5	104	3 975	9.68M	306.63M	1.05M	$34.98	$45.59	$227.96
User 6	217	8 734	20.83M	734.60M	2.20M	$80.61	$105.28	$526.40
User 7	333	13 360	30.44M	749.16M	3.24M	$93.61	$121.57	$607.83
User 8	18	703	1.60M	38.30M	0.40M	$5.88	$7.44	$37.18
User 9	116	4 910	11.24M	262.21M	1.62M	$35.38	$45.55	$227.77
…								
User 44	466	20 584	44.90M	1 286.50M	6.40M	$158.96	$205.55	$1 027.75
Total	9 092	371 281	1 056.81M	22 969.92M	111.24M	$3 015.94	$3 910.01	$19 550.06

Savings rate (Mini vs. Opus) varies by only 0.5 percentage points across all users—from 84.2% to 84.7%. Model pricing, not usage behavior, drives the costs.

SECTION 5

Tool Cost Breakdown

AI coding agents interact with codebase, documentation, internet or tasks through **tools** — atomic capabilities like running terminal commands (bash), reading files (read), making code changes (edit), or creating new files (write).

Each tool call consumes:

- **Output tokens** — the model generating the **next tool call itself** (e.g. which file to read, which bash command to run with parameters).
- **Input tokens** — tool definitions, parameters, and most importantly **tool results** fed back to the model, such as terminal output from bash or file content from read.
- **Cached tokens** — input and output tokens from previous tool calls in same session are re-sent as context in **every** subsequent round. Session length directly compounds cost.

The table below breaks down where your AI budget actually goes.

23%
of total costs is generated by bash alone

41.8%
of tool costs come from one utility: bash

$454.63
Claude Opus monthly cost for bash alone

Tool	Activity	Tokens	Share	GPT-5.4 Mini	Haiku 4.5	Claude Opus
bash	Scripts, builds, tests	2 643.18M	41.8%	$278.18	$363.70	$1 818.51
read	File scanning	923.74M	20.2%	$134.10	$175.06	$875.30
edit	Code changes	776.96M	12.9%	$85.85	$110.62	$553.10
write	New files	349.78M	9.6%	$63.69	$77.84	$389.19
todowrite	Task tracking	209.36M	4.3%	$28.44	$37.04	$185.20
Other	20+ low-impact tools	382.20M	11.2%	$74.49	$96.00	$480.00
Total		5 285.21M	100%	$664.76	$860.26	$4 301.31

Table based on data from 3 users — no data for remaining users.

SECTION 6

Model Cost Comparison

Monthly cost for the same workload across models in the Copilot catalog (June 2026 rates).

Provider / Model	Input $/M	Cache $/M	Output $/M	Monthly Cost
OpenAI GPT-5.4 mini	**$0.75**	**$0.075**	**$4.50**	**$3 015.94**
OpenAI GPT-5 mini	$0.25	$0.025	$2.00	$1 060.93
OpenAI GPT-5.3-Codex	$1.75	$0.175	$14.00	$7 426.54
OpenAI GPT-5.4	$2.50	$0.25	$15.00	$10 053.13
OpenAI GPT-5.4 nano	$0.20	$0.02	$1.25	$809.81
OpenAI GPT-5.4 (long)	$5.00	$0.50	$22.50	$19 271.96
OpenAI GPT-5.5	$5.00	$0.50	$30.00	$20 106.27
OpenAI GPT-5.5 (long)	$10.00	$1.00	$45.00	$38 543.91
Claude Haiku 4.5	**$1.00**	**$0.10**	**$5.00**	**$3 910.01**
Claude Sonnet 4.6	$3.00	$0.30	$15.00	$11 730.04
Claude Opus 4.8	**$5.00**	**$0.50**	**$25.00**	**$19 550.06**
Claude Sonnet 5	$2.00	$0.20	$10.00	$7 820.02
Claude Opus 4.8 (fast)	$10.00	$1.00	$50.00	$39 100.12
Claude Fable 5	$10.00	$1.00	$50.00	$39 100.12
Google Gemini 2.5 Pro	$1.25	$0.125	$10.00	$5 304.67
Google Gemini 3 Flash	$0.50	$0.05	$3.00	$2 010.63
Google Gemini 3.1 Pro	$2.00	$0.20	$12.00	$8 042.51
Google Gemini 3.1 Pro (long)	$4.00	$0.40	$18.00	$15 417.57
Google Gemini 3.5 Flash	$1.50	$0.15	$9.00	$6 031.88
GitHub Raptor mini	$0.25	$0.025	$2.00	$1 060.93
GitHub Kimi K2.7 Code	$0.95	$0.19	$4.00	$5 813.22
Microsoft MAI-Code-1-Flash	$0.75	$0.075	$4.50	$3 015.94

Our model of choice: GPT-5.4 Mini — solid reasoning, excellent tool calling and instruction following, concise communication, and the best speed among tested models.

Worthy of testing: Kimi K2.7 Code — overall a great coding model at a reasonable price

SECTION 7

What to Do Next

- **Standardize on Cost-Effective Tiers**
  Transition corporate developer profiles to GPT-5.4 Mini or Claude Haiku 4.5. Both deliver production-grade code while securing an immediate 84.2% to 84.7% cost reduction relative to premium models.
- **Deploy Bash Tool Optimisations**
  Instruct the DevEx team to inject Bash output redirection and output filtering giving agent critical signals like build fail / failing tests / app runtime exceptions depending on command semantic, before output re-enters the agent loop.
- **Mandate Blueprints & Subagents**
  Implement standard AGENTS.md blueprints with efficient repository indexing. Mandate that all codebase exploration be delegated to isolated subagents, protecting the primary agent from context bloat.
- **Real-Time Local Cost Feedback**
  Integrate real-time cost-visibility metrics directly into the developer's local IDE or terminal. This psychological accountability encourages deliberate usage, tighter scoping, and self-driven optimization.
- **Central Governance & Task Correlation**
  Implement centralized telemetry to aggregate token expenditures and map them to tracking tickets. Quantifying the precise AI cost-per-feature drives institutional transparency and provides the baseline data needed to safely scale autonomous operations.

BOTTEGA
IT minds

The model is not the bottleneck.
The harness is.

Developer complaints about small models reflect real experience — but they stem from **missing harness engineering**, not model capability. Given too many degrees of freedom, any model hallucinates.

What determines quality in enterprise AI?

When a model must independently explore "how to do this in our codebase" instead of following precise instructions, it hallucinates. *Harness engineering* — agent instructions, skills, CLI tools, architecture blueprints, precise business specifications, deterministic quality gates — eliminates the guesswork. The model executes standard procedures and patterns. *That* is the multiplier.

SECTION 8

Study Overview

Over 30 days, 44 users worked across different systems, codebases, various harnesses and licences (GitHub Copilot, OpenAI Codex, Claude Code, OpenCode Zen/Go), switching models as they optimized for cost and quality.

3 users worked mostly with GPT 5.4 Mini, 4 users declared working mostly using open source models, many others declared Claude Sonnet and Claude Opus or GPT family.

Users performed diverse tasks spanning the full development lifecycle:

Specification
Writing specs from business documents and meeting transcripts

Legacy Codebase Exploration
Navigating and understanding 1M+ LOC legacy monoliths

Fullstack Feature Work
Implementing complex features inside legacy code

New Microservices
Building services with intricate business logic from scratch

Automated Testing
From unit tests to complete system e2e suites

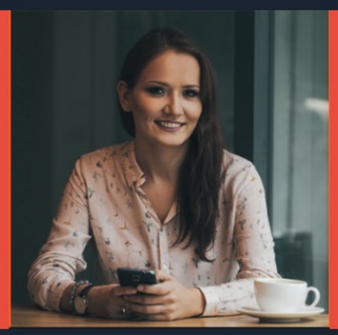
Harness Engineering
Developing agent instructions, skills, CLI tools, architecture blueprints, precise business specifications, deterministic quality gates

Our **Clients**

As engineers, we consider our growing number of returning clients to be **the most important indicator of our success.**

MARKET COVERAGE:

Fintech, e-commerce, telco, automotive, mobile banking, enterprise resource planning, content management systems, data management systems, start-ups, automotive, military, energy, semiconductor manufacturing, industry 4.0



Interested in finding out more about what Bottega can do for you? **Contact us.**

Iwona Sobótka — Consulting Coordinator: +48 500 286 698
Office: +48 81 473 29 44, contact@bottega.com.pl