BO·TT·EGA
IT minds

9 092 sesji deweloperskich
od 44 użytkowników

24.14B zapłaconych tokenów
95% cached

$19 550.06 miesięczny koszt przy użyciu
tylko modeli premium

85% redukcji kosztów poprzez
wybór odpowiedniego modelu

$16 534.12 potencjalnych oszczędności
vs model premium

B = miliard (10^9) tokenów

BRIEFING ZARZĄDCZY

Optymalizacja kosztów agentów kodujących

Analiza danych operacyjnych i prognozy kosztów
Po przejściu **GitHub Copilot** na **rozliczenie według użycia**

Jak zmniejszyć koszty kodowania z AI o 85% bez utraty jakości kodu?

Wyniki z pracy w systemach enterprise — potwierdzone na 9 092 sesjach deweloperskich

Do: Dyrektorzy / Zarząd
Data: 1 lipca 2026
Okres raportowania: Czerwiec 2026 — pełny cykl 30-dniowy
Przygotowane przez: Bottega IT Minds — Transformacje AI

NAJWAŻNIEJSZE

Benchmarki kłamią, faktury niestety nie

Właściwy wybór modelu + harness engineering = 84.2–84.7% oszczędności przy rozliczeniach według użycia tokenów

Niniejszy raport zawiera analizę 9 092 sesji deweloperskich wykonanych przez 44 inżynierów na rzeczywistym kodzie enterprise. Raport zawiera obserwacje, wnioski i rekomendacje istotne przed wdrożeniem AI w całym przedsiębiorstwie.

Dlaczego to ważne: 1 czerwca 2026 Microsoft przeniósł **GitHub Copilot** z subskrypcji ryczałtowej na **rozliczenie według użycia** — osobne stawki za token wejście, wyjście i cached.

Zakres: Analiza obejmuje modele w katalogu GitHub Copilot. Modele open-source (Llama, DeepSeek) na prywatnej infrastrukturze są wykluczone z tego opracowania.

Benchmarki a koszt: Opus osiąga ~11pp lepsze wyniki w benchmarkach (według producentów), kosztując 6.5× więcej niż Mini. Haiku pozostaje w tyle za Mini w 3 z 4 benchmarków, a kosztuje 1.3× więcej. GPT-5.5 to aż 12.8× więcej niż 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×**

SEKCJA 2

Kluczowe wnioski

Modele premium kosztują więcej bez proporcjonalnych zysków

Claude Opus kosztuje 6.5× więcej niż GPT-5.4 Mini, a benchmarki pokazują tylko ~11pp poprawy. Mniejsze modele iterują szybciej, co bezpośrednio przyspiesza tempo developmentu.

Doświadczony użytkownik jest ważniejszy niż model

Mały model z precyzyjnym mapowaniem kodu, dokumentacją architektury i jasnymi specyfikacjami przewyższa duży model bez tego wsparcia.

Caching kontekstu to czynnik kosztów nr 1.

Cached tokeny stanowią *95% całego wolumenu* w sesjach agentów AI. Modele z agresywnymi zniżkami na cache wygrywają na fakturze.

Jedna poprawka użycia narzędzi agenta oszczędza dwucyfrowe kwoty

Polecenia Bash odpowiadają za 41.8% kosztów narzędzi — z powodu zalewania okna kontekstowego obszernymi logami. Oszczędności rzędu kilkudziesięciu dolarów miesięcznie per użytkownik.

← Źródła benchmarków: benchmarklist.com, Anthropic (evals.report), OpenAI / DataCamp. Wynik Opus 4.8 Terminal-Bench 2.1 z allthings.how. Mnożnik kosztów na podstawie rzeczywistego miesięcznego użycia (24.14B tokenów).

JAK TO DZIAŁA

Trzy kategorie tokenów

Przykład: GPT-5.4 Mini — Wejście $0.75/M • Cache $0.075/M • Wyjście $4.50/M
Jedna sesja: 100K Wejścia + 900K Cache + 50K Wyjścia =
(0.1 × $0.75) + (0.9 × $0.075) + (0.05 × $4.50) = $0.37

Każda interakcja AI zużywa tokeny w trzech kategoriach, rozliczane za milion (M) tokenów:

Wejście
Prompty, kod odczytany z repozytorium i instrukcje wysłane *do* modelu.

Cache
Wykorzystany kontekst z pamięci cache. Rozliczany z *~90% zniżką* vs. standardowe wejście.

Wyjście
Generowany kod, wnioskowanie i analiza napisane *przez* model. Najwyższa stawka.

DANE MIESIĘCZNE

Metryki użytkowników

Rzeczywisty wolumen od 44 użytkowników przez 30 dni, z projekcjami kosztów dla różnych modeli.

95%
cached tokenów w płatnym wolumenie

85%
oszczędności przy Mini vs. Opus ($3 015.94 vs. $19 550.06)

0.5pp
zmienności w oszczędnościach między użytkownikami

Użytkownik	Sesje	Wiadomości	Wejście	Cache	Wyjście	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
Razem	9 092	371 281	1 056.81M	22 969.92M	111.24M	$3 015.94	$3 910.01	$19 550.06

Stopa oszczędności (Mini vs. Opus) różni się tylko o 0.5punktu procentowego między użytkownikami — od 84.2% do 84.7%. Ceny modeli, a nie zachowanie użytkowników, determinują koszty.

SEKCJA 5

Struktura kosztów narzędzi

Agenci kodujący AI wchodzą w interakcje z bazą kodu, dokumentacją, internetem i zadaniami poprzez **narzędzia** — atomowe możliwości, takie jak uruchamianie poleceń terminalowych (bash), czytanie plików (read), wprowadzanie zmian w kodzie (edit), czy tworzenie nowych plików (write).

Każde wywołanie narzędzia zużywa:

- **Tokeny wyjścia** — model generuje **samo wywołanie narzędzia** (np. który plik czytać, które polecenie bash uruchomić z jakimi parametrami).
- **Tokeny wejścia** — definicje narzędzi i parametrów oraz **wyniki narzędzi** przekazane z powrotem do modelu, takie jak wyjście terminala z bash lub zawartość pliku z read.
- **Cachowane tokeny** — tokeny wejścia i wyjścia z poprzednich wywołań narzędzi w tej samej sesji są wysyłane ponownie jako kontekst w **każdej** kolejnej rundzie. Długość sesji bezpośrednio zwielokrotnia koszt.

Poniższa tabela pokazuje, które narzędzia konsumowały najwięcej tokenów.

23%
całkowitych kosztów generuje samo bash

41.8%
kosztów narzędzi pochodzi z jednej komendy: bash

$454.63
miesięczny koszt Claude Opus za samo bash

Narzędzie	Aktywność	Tokeny	Udział	GPT-5.4 Mini	Haiku 4.5	Claude Opus
bash	Skrypty, buildy, testy	2 643.18M	41.8%	$278.18	$363.70	$1 818.51
read	Skanowanie plików	923.74M	20.2%	$134.10	$175.06	$875.30
edit	Zmiany w kodzie	776.96M	12.9%	$85.85	$110.62	$553.10
write	Nowe pliki	349.78M	9.6%	$63.69	$77.84	$389.19
todowrite	Śledzenie zadań	209.36M	4.3%	$28.44	$37.04	$185.20
Inne	20+ narzędzi niskiego wpływu	382.20M	11.2%	$74.49	$96.00	$480.00
Razem		5 285.21M	100%	$664.76	$860.26	$4 301.31

Tabela w oparciu o dane od 3 użytkowników — brak danych dla pozostałych.

SEKCJA 6

Porównanie kosztów modeli

Miesięczny koszt tego samego obciążenia dla modeli w katalogu Copilot (stawki z czerwca 2026).

Dostawca / Model	Wejście $/M	Cache $/M	Wyjście $/M	Koszt miesięczny
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

Nasz wybór: GPT-5.4 Mini — solidne wnioskowanie, doskonałe wywoływanie narzędzi i podążanie za instrukcjami, zwięzła komunikacja i najlepsza szybkość spośród testowanych modeli.

Warty przetestowania: Kimi K2.7 Code — model doskonały do kodowania w rozsądnej cenie

SEKCJA 7

Rekomendowane akcje

- **Standaryzacja do korzystnie wycenionych modeli**
  Przestaw profile developerskie na GPT-5.4 Mini lub Claude Haiku 4.5. Oba dostarczają kod klasy produkcyjnej, zapewniając natychmiastową redukcję kosztów o 84.2–84.7% względem modeli premium.
- **Wdrożenie optymalizacji narzędzia Bash**
  Zleć zespołowi DevEx implementację przekierowania wyjścia Basha i filtrowania, aby agent otrzymywał tylko krytyczne sygnały — błędy builda, nieudane testy, wyjątki runtime — zanim wynik wróci do pętli agenta.
- **Obowiązkowe blueprinty i subagenci**
  Wdróż standardowe blueprinty AGENTS.md z wydajnym indeksowaniem repozytorium. Instrukcje delegacji eksploracji kodu do subagentów, chroniąc głównego agenta przed zanieczyszczeniem kontekstu.
- **Lokalny feedback kosztów w czasie rzeczywistym**
  Zintegruj wskaźniki kosztów bezpośrednio w lokalnym IDE lub terminalu dewelopera. Ta psychologiczna odpowiedzialność za koszt zachęca do świadomego użycia, precyzyjniejszego instruowania, restartów sesji i samodzielnej optymalizacji.
- **Centralne zarządzanie i korelacja z zadaniami**
  Wdróż scentralizowaną telemetrię agregującą wydatki na tokeny i mapującą je na zadania. Ilościowe określenie kosztu AI na dostarczoną funkcjonalność zapewnia przejrzystość instytucjonalną i dane bazowe do bezpiecznego dalszego skalowania użycia AI.

BO·TT·EGA
IT minds

Model nie jest wąskim gardłem.
Jest nim harness.

Skargi deweloperów na małe modele odzwierciedlają realne doświadczenie — ale wynikają z **braku harness engineering-u**, a nie z możliwości modelu. Przy zbyt wielu stopniach swobody każdy model halucynuje, dla małych modeli zjawisko jest bardziej wyraźne.

Co decyduje o jakości w enterprise AI?

Gdy model musi samodzielnie zgadywać "jak coś zaimplementować w naszej bazie kodu" zamiast podążać za precyzyjnymi instrukcjami, halucynuje. *Harness engineering* — instrukcje dla agentów, skille, narzędzia CLI, blueprinty architektury, precyzyjne specyfikacje biznesowe, deterministyczne bramki jakościowe — eliminuje zgadywanie. Model wykonuje standardowe procedury i wzorce. *To* jest mnożnik jakości.

SEKCJA 8

O badaniu

Przez 30 dni 44 użytkowników pracowało w różnych systemach, bazach kodu i przy użyciu różnych narzędzi i licencji (GitHub Copilot, OpenAI Codex, Claude Code, OpenCode Zen/Go), przełączając modele w miarę optymalizacji kosztów i jakości.

3 użytkowników pracowało głównie z GPT 5.4 Mini, 4 zadeklarowało pracę głównie na modelach open source, a wielu innych korzystało z Claude Sonnet i Claude Opus oraz rodziny GPT.

Użytkownicy wykonywali zróżnicowane zadania obejmujące pełny cykl życia oprogramowania:

Specyfikacja
Pisanie specyfikacji z dokumentów biznesowych i transkryptów spotkań

Eksploracja legacy codebase
Nawigacja i zrozumienie 1M+ LOC w monolitycznych systemach

Fullstack feature work
Implementacja złożonych funkcji w legacy kodzie

Nowe mikroserwisy
Budowa usług ze złożoną logiką biznesową od podstaw

Testy automatyczne
Od testów jednostkowych po pełne zestawy e2e

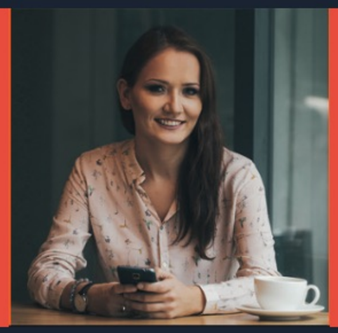
Harness engineering
Tworzenie instrukcji dla agentów, skilli, narzędzi CLI, blueprintów architektury, precyzyjnych specyfikacji biznesowych, deterministycznych bram jakości

Nasi **Klienci**

Jako inżynierowie, za najważniejszy wskaźnik naszego sukcesu uważamy rosnącą liczbę powracających klientów.

OBSZARY KOMPETENCJI:

Fintech, e-commerce, telco, motoryzacja, bankowość mobilna, systemy ERP, systemy CMS, systemy zarządzania danymi, start-upy, wojsko, energetyka, półprzewodniki, przemysł 4.0



Interesuje Cię, co Bottega może dla Ciebie zrobić? **Skontaktuj się z nami.**

Iwona Sobótka — Koordynatorka Konsulting: +48 500 286 698
Biuro: +48 81 473 29 44, contact@bottega.com.pl