KI-Produktivität in der Softwareentwicklung 2026: Der Forschungsstand

Von **Thomas Faß** · 16. September 2026 · aktualisiert 16. September 2026

KI-Produktivität in der Softwareentwicklung bezeichnet den messbaren Zeit- und Durchsatzgewinn, den Entwicklungsteams durch KI-Werkzeuge erzielen. In drei randomisierten Feldexperimenten bei Microsoft, Accenture und einem Fortune-100-Unternehmen mit insgesamt 4.867 Entwicklern lag der Gewinn bei 26,08 Prozent mehr abgeschlossenen Aufgaben (Management Science, Februar 2026). Der Effekt verteilt sich ungleich: Weniger erfahrene Entwickler steigerten ihre Ausgabe um 21 bis 40 Prozent, erfahrene Entwickler um 7 bis 16 Prozent. In einer separaten randomisierten Studie mit 16 erfahrenen Open-Source-Entwicklern in ihren eigenen Repositories arbeiteten diese mit KI-Werkzeugen 19 Prozent langsamer (METR, Juli 2025). Der Gewinn entsteht also dort, wo Aufgaben ausgeführt werden, und nicht dort, wo über sie entschieden wird.

Das Wichtigste in Kürze

- Der Gewinn ist gemessen, nicht behauptet: Drei randomisierte Feldexperimente mit 4.867 Entwicklern ergeben 26,08 Prozent mehr abgeschlossene Aufgaben (Standardfehler 10,3 Prozent; Management Science, 27.02.2026).
- Mit einem Vorgehen, das um die KI herum gebaut ist, liegen die Werte höher: In unserer laufenden eigenen Messung liegt der Median bei 19,5× über acht Datenpunkte, die Spannbreite bei 15× bis 37× — je nach Aufgabenart. Für Arbeit mit unklarer Spezifikation liegt kein Messwert vor; dafür nennen wir keinen Faktor.
- Er verteilt sich ungleich: Weniger erfahrene Entwickler gewinnen 21 bis 40 Prozent, erfahrene 7 bis 16 Prozent (gleiche Studie).
- Am oberen Ende kippt der Effekt: 16 erfahrene Entwickler in ihren eigenen Repositories brauchten mit KI-Werkzeugen 19 Prozent länger — und glaubten hinterher, 20 Prozent schneller gewesen zu sein (METR, 10.07.2025).
- Die Qualität leidet nicht automatisch: In den Feldexperimenten gab es keinen negativen Effekt auf die Build-Erfolgsquote, aber 38,38 Prozent mehr Kompilierungsvorgänge — ein Hinweis auf mehr Versuch und Irrtum.
- Der Engpass wandert: KI-Einsatz zeigt einen positiven Zusammenhang mit dem Auslieferungsdurchsatz und weiterhin einen negativen mit der Auslieferungsstabilität (DORA, 23.09.2025, rund 5.000 Fachleute).
- Die Prüfkosten sind der Grund: 66 Prozent der Entwickler nennen als größte Frustration Lösungen, die „fast richtig, aber nicht ganz“ sind; 45,2 Prozent halten das Debuggen von KI-Code für zeitaufwendiger als bei menschlichem Code (Stack Overflow, über 49.000 Antworten aus 177 Ländern, 2025).
- Die Folge für die Projektplanung: KI in die Ausführung, Senior-Prüfung ins Gate, und vor dem Start eine Zahl festlegen, an der der Gewinn später nachrechenbar ist.

Was die Studien messen — und was nicht

Zur KI-Produktivität in der Softwareentwicklung liegen seit 2023 kontrollierte Studien vor. Sie widersprechen sich scheinbar. Tatsächlich messen sie verschiedene Situationen, und genau darin liegt die verwertbare Erkenntnis.

Größe	Wert	Quelle
Abgeschlossene Aufgaben, drei Feldexperimente, 4.867 Entwickler	**+26,08 %** (SE 10,3 %)	Cui et al., *Management Science*, 27.02.2026
davon weniger erfahrene Entwickler	**+21 bis 40 %**	ebd.
davon erfahrene Entwickler	**+7 bis 16 %**	ebd.
Build-Erfolgsquote	**kein negativer Effekt**	ebd.

Größe	Wert	Quelle
Kompilierungsvorgänge	**+38,38 %** (SE 12,55 %)	ebd.
Einzelaufgabe HTTP-Server in JavaScript, kontrolliertes Experiment	**55,8 % schneller**	Peng et al., arXiv 2302.06590, 13.02.2023
Erfahrene Entwickler in eigenen Repositories, 246 Aufgaben	**19 % langsamer**	METR, 10.07.2025
deren Selbsteinschätzung nach dem Versuch	**20 % schneller**	ebd.
Auslieferungsdurchsatz	**positiver Zusammenhang**	DORA, 23.09.2025
Auslieferungsstabilität	**negativer Zusammenhang**	ebd.
„fast richtig, aber nicht ganz“ als größte Frustration	**66 %**	Stack Overflow Developer Survey 2025
Debuggen von KI-Code zeitaufwendiger	**45,2 %**	ebd.
hohes Vertrauen in die Genauigkeit von KI-Werkzeugen	**3,1 %**	ebd.
Bauzeit je Story, KI-nativ gegen klassische Schätzung — Median über 8 Messpunkte	**19,5×** (Spannbreite 15 bis 37)	InnoShore, laufende eigene Messung seit Juli 2026
davon Security- und Härtungsarbeiten im Bestand	**17,5×** (4 Punkte)	ebd.
davon Frontend-Aufbau nach fertiger Spezifikation	**18×** (1 Punkt)	ebd.
davon gemischter Wartungsstapel	**19×** (1 Punkt)	ebd.
davon neue Fachlogik mit hoher Vertragsdichte	**27× und 37×** (2 Punkte)	ebd.
Arbeit mit unklarer Spezifikation und subjektiver Iteration	**nicht gemessen**	ebd.

Plus 26 Prozent bei 4.867 Entwicklern — die belastbarste Zahl

Die Studie mit der größten Stichprobe und dem stärksten Design erschien im Februar 2026 in *Management Science*. Kevin Zheyuan Cui, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng und Tobias Salz haben drei randomisierte kontrollierte Studien ausgewertet, die die beteiligten Unternehmen im normalen Geschäftsbetrieb durchgeführt hatten: **Microsoft mit 1.746 Entwicklern, Accenture mit 320 und ein anonymes Fortune-100-Unternehmen mit 3.054.** Ein zufällig ausgewählter Teil der Entwickler erhielt

Zugang zu einem KI-Assistenten mit Code-Vervollständigung, der Rest nicht. Die Experimente liefen zwei bis acht Monate.

Das Ergebnis im Wortlaut der Autoren: *„when data are combined across three experiments and 4,867 developers, our analysis reveals a 26.08% increase (standard error: 10.3%) in completed tasks among developers using the AI tool."*

Zwei Dinge sind an dieser Zahl wichtig. Gezählt wurden **abgeschlossene Aufgaben**, nicht Einschätzungen der Beteiligten. Und der Standardfehler von 10,3 Prozent gehört dazugesagt: Die einzelnen Experimente waren verrauscht, der Effekt zeigt sich erst in der Zusammenfassung.

55,8 Prozent bei einer isolierten Aufgabe — die Obergrenze

Die früheste kontrollierte Studie stammt von Sida Peng, Eirini Kalliamvakou, Peter Cihon und Mert Demirer, eingereicht am 13. Februar 2023. Die Aufgabe war eng gefasst: einen **HTTP-Server in JavaScript** implementieren, so schnell wie möglich. Die Gruppe mit KI-Zugang war **55,8 Prozent schneller**.

Diese Zahl wird häufig als allgemeiner Produktivitätsgewinn zitiert. Das ist sie nicht. Sie ist der Wert für eine abgeschlossene, gut dokumentierte Standardaufgabe ohne Altlasten, ohne Domänenwissen und ohne Abstimmung — also für die günstigste denkbare Bedingung. Bemerkenswert ist der Nebensatz der Autoren: *„Observed heterogenous effects show promise for AI pair programmers to help people transition into software development careers."* Die Heterogenität nach Erfahrung war also schon 2023 sichtbar.

Minus 19 Prozent bei Experten im eigenen Code — die Gegenprobe

Im Juli 2025 veröffentlichte METR eine randomisierte kontrollierte Studie, die in die andere Richtung zeigt. Sechzehn erfahrene Open-Source-Entwickler arbeiteten an **246 Aufgaben** in Repositories, zu denen sie seit Jahren beitragen — im Schnitt über 22.000 Sterne und über eine Million Codezeilen. Je Aufgabe wurde zufällig entschieden, ob KI-Werkzeuge erlaubt waren.

Der Befund wörtlich: *„Surprisingly, we find that when developers use AI tools, they take 19% longer than without—AI makes them slower."* Und die Selbsteinschätzung: *„developers expected AI to speed them up by 24%, and even after experiencing the slowdown, they still believed AI had sped them up by 20%."*

Die Autoren schränken selbst ein: *„We do not claim that our developers or repositories represent a majority or plurality of software development work."* Sechzehn Personen sind eine kleine Stichprobe, und ein Selektionseffekt ist möglich. Was die Studie trotzdem leistet, ist zweierlei. Sie zeigt eine Situation, in der das Werkzeug nichts bringt. Und sie zeigt, dass die Betroffenen das nicht merken.

Durchsatz steigt, Stabilität sinkt — die Systemwirkung

Der DORA-Report vom 23. September 2025 beruht auf Antworten von knapp 5.000 Fachleuten und über 100 Stunden qualitativer Erhebung. **90 Prozent der Befragten nutzen KI bei der Arbeit, über 80 Prozent halten sich für produktiver, 30 Prozent haben wenig oder kein Vertrauen in den erzeugten Code.**

Für die Lieferfähigkeit lautet der Befund wörtlich: *„Unlike last year, we observe a positive relationship between AI adoption on both software delivery throughput and product performance. However, AI adoption does continue to have a negative relationship with software delivery stability."*

Das ist die organisatorische Übersetzung der Einzelbefunde: Es wird mehr und schneller ausgeliefert, und es geht häufiger etwas schief.

Drei Prozent hohes Vertrauen — die Prüfkosten

Die Stack Overflow Developer Survey 2025 mit über 49.000 Antworten aus 177 Ländern liefert die Erklärung, warum der Gewinn bei Erfahrenen schrumpft. **84 Prozent** nutzen KI-Werkzeuge oder planen es, **59,8 Prozent** stehen ihnen positiv gegenüber — aber nur **3,1 Prozent** haben hohes Vertrauen in die Genauigkeit, und **46 Prozent** misstrauen ihr aktiv.

Der entscheidende Wert ist die genannte Hauptfrustration: **66 Prozent** nennen Lösungen, die „fast richtig, aber nicht ganz" sind. Und **45,2 Prozent** halten das Debuggen von KI-erzeugtem Code für zeitaufwendiger als bei menschlich geschriebenem.

Eine Lösung, die fast richtig ist, kostet mehr Prüfzeit als eine, die offensichtlich falsch ist. Wer die Fehler schnell findet, verliert wenig. Wer sie sucht, verliert den Gewinn.

Was wir selbst messen — und wie

Die Werte oben stammen aus fremden Erhebungen. Seit Juli 2026 messen wir zusätzlich im eigenen Betrieb, nach einem schriftlich festgelegten Rahmen. Die Zahlen liegen deutlich höher als in den Studien, und der Grund dafür ist derselbe, den dieser Artikel die ganze Zeit beschreibt: Gemessen wird ein anderes Vorgehen, nicht ein anderes Werkzeug.

Die Messlogik in einem Satz: Faktor = geschätzter klassischer Aufwand ÷ tatsächlicher Aufwand im KI-nativen Vorgehen. Die Einheit ist die einzelne User Story. Ein Senior schätzt vor Baubeginn die klassischen Personenstunden („rein menschlich, ohne KI-Assistenz") und friert die Schätzung ein; der tatsächliche Aufwand wird gemessen. Aggregiert wird über den **Median**, nicht über den Mittelwert — ein Mittelwert würde von den Ausreißern nach oben dominiert.

Der ausgewiesene Aufwand ist der Systemfaktor, nicht die Maschinenzeit. In den Ist-Wert geht die menschliche Zeit mit ein: Aufgabenstellung formulieren, Prüfrunden, Verbuchung. Bei einem Wartungsstapel aus fünf Teilaufgaben etwa lag die reine Ausführung bei 37 Minuten; mit rund 45 Minuten menschlicher Zeit ergibt das 1,37 Stunden — und aus einem Rohfaktor von 42 wird der belastbare Wert 19. Wir rechnen mit dem zweiten.

Stand der Messung: acht Datenpunkte, Median **19,5×**, Spannbreite **15× bis 37×**. Das arithmetische Mittel liegt bei 21,6, das geometrische bei 20,7. Über die Rohwerte ohne Systembereinigung käme man auf 36 — diese Zahl verwenden wir nicht, weil zwei Ausreißer sie tragen.

Die Spreizung ist die eigentliche Information, nicht der Median. Nach Aufgabenart getrennt:

Aufgabenart	Faktor	Messpunkte
Security- und Härtungsarbeiten im Bestand	Median 17,5×	4
Frontend-Aufbau nach fertiger Spezifikation	18×	1

Aufgabenart	Faktor	Messpunkte
Gemischter Wartungsstapel, fragmentiert	19×	1
Neue Fachlogik mit hoher Vertragsdichte	27× und 37×	2
Unklare Spezifikation, subjektive Iteration	**nicht gemessen**	0

Der Befund, der uns selbst überrascht hat: Der Faktor **steigt** mit der Vertragsdichte — je mehr bestehende Garantien eingehalten werden müssen, desto größer der Vorsprung. Und die Kontextwechsel-Steuer, die einen Menschen bei fünf verschiedenen Codebasis-Ecken 20 bis 30 Prozent kostet, ist für die Maschine praktisch null. Was Menschen langsam macht, ist für die KI billig.

Wo wir nichts sagen. Die letzte Zeile der Tabelle ist die wichtigste. Arbeit mit unklarer Spezifikation, wandernden Anforderungen und subjektiver Iteration — Designentscheidungen unterwegs, Abstimmungsrunden über Geschmacksfragen — haben wir **null Mal gemessen**. Das ist der vermutete Einbruchspunkt, und solange er ungemessen ist, behaupten wir dafür keinen Faktor. Wer Zahlen für diese Art Arbeit nennt, hat sie in der Regel nicht erhoben.

Was die Zahlen nicht bedeuten. Sie gelten für die Bauzeit einer Story, nicht für die Dauer eines Vorhabens. Anforderungsklärung davor und Abnahme danach laufen unverändert und menschlich. Wie stark der Vorsprung auf die Projektdauer durchschlägt, hängt am Anteil der Bauzeit am Gesamtvorhaben — bei einem Neubau ein anderer als bei einer Erweiterung im gewachsenen System. Diese Rechnung machen wir im Erstgespräch auf, bevor ein Angebot entsteht.

Was am Verfahren offen ist, und zwar ausdrücklich. Der Vergleichsmaßstab ist eine Schätzung, keine parallele Referenzumsetzung desselben Vorhabens. Von den acht Schätzungen waren zwei vor dem Bau eingefroren, die übrigen retrospektiv erhoben. Schätzer und Ausführender sind bisher nicht unabhängig voneinander; eine unabhängige menschliche Vergleichsschätzung steht als nächster Härtegrad an. Der Abstand zwischen gefühltem und gemessenem Faktor — bei fremden Studien die größte Fehlerquelle — ist in unserer Reihe noch nicht erhoben. Und das Zeitfenster, in dem Fehler nach der Abnahme auffallen könnten, ist bei den jüngsten Punkten kurz.

Deshalb steht hier eine laufende Messung und kein Ergebnis. Unser eigener Rahmen sieht eine externe Aussage erst ab drei bis fünf abgeschlossenen Projekten und nur mit Segment-Angabe vor. Diese Schwelle ist nicht erreicht. Die Zahlen stehen hier als offengelegter Zwischenstand, nicht als Werbeaussage — und die Rohdaten geben wir heraus, wenn jemand sie sehen will.

Ein Messpunkt ist diese Website. Die Lösungsseite zu KI-nativer Softwareentwicklung ist Datenpunkt 007: klassisch geschätzt 13 Stunden, tatsächlich 43 Minuten bis zur grünen CI-Prüfung, Faktor 18. Der Qualitätsteil dieses Punktes ist noch offen.

Quellen & weiterführende Studien

- Cui, K. Z.; Demirer, M.; Jaffe, S.; Musolff, L.; Peng, S.; Salz, T. (2026): „The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers." *Management Science*, veröffentlicht 27.02.2026. https://pubsonline.informs.org/doi/10.1287/mnsc.2025.00535
- Arbeitsfassung derselben Studie (Februar 2025, 28 Seiten) mit den Teilergebnissen nach Erfahrungsstufe: https://economics.mit.edu/sites/default/files/inline-files/draft_copilot_experiments.pdf
- Peng, S.; Kalliamvakou, E.; Cihon, P.; Demirer, M. (2023): „The Impact of AI on Developer Productivity: Evidence from GitHub Copilot." arXiv:2302.06590, eingereicht 13.02.2023. https://arxiv.org/abs/2302.06590
- METR (2025): „Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity", 10.07.2025. https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/
- DORA / Google Cloud (2025): „State of AI-assisted Software Development", 23.09.2025. https://cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report
- Stack Overflow (2025): „Developer Survey 2025", KI-Abschnitt. https://survey.stackoverflow.co/2025/

Warum der Gewinn ungleich verteilt ist

Fünf Treiber erklären, warum dieselbe Technologie in einem Team 40 Prozent bringt und im nächsten nichts.

Treiber 1 — Der Aufgabentyp entscheidet mehr als das Werkzeug

Die 55,8 Prozent aus dem Copilot-Experiment gelten für eine klar umrissene Standardaufgabe. Die 19 Prozent Verlangsamung bei METR gelten für Arbeit an gewachsenem Code mit Architekturentscheidungen. Zwischen diesen Polen liegt jedes reale Vorhaben. Wer einen Prozentsatz für „KI-Produktivität" nennt, ohne den Aufgabentyp zu nennen, nennt keine Zahl.

Treiber 2 — Wer den Code kennt, hat weniger zu gewinnen

Der Effekt bei Cui et al. fällt bei kürzerer Betriebszugehörigkeit und in Junior-Positionen deutlich an, bei langer Zugehörigkeit und Senior-Positionen kaum. Die METR-Teilnehmer arbeiteten an Repositories, die sie seit Jahren kennen. Ein Vorschlag, der einem Neuen fünf Minuten Nachschlagen erspart, erspart dem Alteingesessenen nichts — er muss ihn trotzdem lesen.

Treiber 3 — „Fast richtig" ist die teuerste Fehlerklasse

Mit 66 Prozent ist das die am häufigsten genannte Frustration in der größten Entwicklerbefragung. Ein offensichtlich falscher Vorschlag wird verworfen und kostet Sekunden. Ein fast richtiger wird übernommen, läuft durch den Test, fällt im Betrieb auf und kostet Tage. Die 45,2 Prozent, die das Debuggen von KI-Code für zeitaufwendiger halten, beschreiben dieselbe Erfahrung.

Treiber 4 — Mehr Versuch und Irrtum verschiebt Aufwand nach hinten

Die 38,38 Prozent mehr Kompilierungsvorgänge bei unveränderter Build-Erfolgsquote sind ein doppelter Befund. Die Qualität hält — und der Weg dorthin führt über mehr Durchläufe. Die Autoren selbst vermuten, Entwickler könnten *„engage in more trial-and-error coding"*. Wo Durchläufe teuer sind, etwa bei langen Integrationstests oder knapper Testumgebung, frisst das den Gewinn.

Treiber 5 — Die Prüfkapazität wächst nicht mit

Der DORA-Befund zur sinkenden Auslieferungsstabilität ist die Summe der vier Treiber darüber. Wenn die Erzeugung schneller wird und die Prüfung gleich bleibt, steigt der Anteil dessen, was ungeprüft durchgeht. Das entscheidet sich an der Kapazitätsverteilung im Team, nicht am Werkzeug.

Quellen & weiterführende Studien

- Zu den Effekten nach Erfahrungsstufe und zu den Kompilierungsvorgängen: Cui et al. (2026), Arbeitsfassung Februar 2025, Abschnitt zur Heterogenität. https://economics.mit.edu/sites/default/files/inline-files/draft_copilot_experiments.pdf
- Zur Fehlerklasse „fast richtig" und zum Debugging-Aufwand: Stack Overflow Developer Survey 2025. https://survey.stackoverflow.co/2025/
- Zur Auslieferungsstabilität: DORA (2025). https://cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report

Wo der Gewinn anfällt — die Arbeitsschritte im Detail

Neubau auf grüner Wiese — der günstigste Fall

Hier liegt die Obergrenze von 55,8 Prozent aus dem Copilot-Experiment: bekannte Muster, keine Altlasten, klare Akzeptanzkriterien. Ein neuer Dienst, eine Schnittstelle, ein Datenmodell nach Vorgabe.

Wiederkehrende Bausteine — der verlässlichste Fall

Formulare, Validierungen, Datenübernahmen, Testdaten, Migrationsskripte. Kein Architekturrisiko, hoher Wiederholungsanteil. Das ist der Bereich, in dem die 26 Prozent aus den Feldexperimenten entstehen, weil die Zahl abgeschlossener Aufgaben zählt und nicht ihre Schwierigkeit.

Tests und Dokumentation — der unterschätzte Fall

Beides wird traditionell zu spät und zu knapp gemacht. Beides ist gut automatisierbar, und beides senkt die Prüfkosten aus Treiber 3. Wer KI hier einsetzt, verbessert die Bedingungen für den Einsatz an anderer Stelle.

Arbeit an gewachsenem Code — der teure Fall

Hier hat METR minus 19 Prozent gemessen. Domänenwissen, implizite Abhängigkeiten, Architekturentscheidungen mit Folgen. Wer diese Arbeit an KI delegiert, verlagert Aufwand in die Prüfung — er spart ihn nicht.

Entscheidungen über Architektur und Schnittstellen — der falsche Fall

Für diesen Bereich gibt es keine Studie mit einem positiven Effekt. Die Entscheidung, welche Systeme verbunden werden, welche Daten wohin fließen und welche Verpflichtung ein Vertrag daraus macht, ist eine Urteilsfrage. Sie fällt in den Bereich, in dem der gemessene Gewinn bei 7 bis 16 Prozent liegt — und dort auch nur für die Ausführung, nicht für das Urteil.

Vier Vorgehensmodelle im Direktvergleich

Modell 1 — Rein menschliche Entwicklung

Kein Werkzeugrisiko, kein Prüfmehraufwand, keine Geschwindigkeitsgewinne. Als Referenzpunkt tragfähig, als Entscheidung 2026 nur noch dort begründbar, wo Vorgaben den KI-Einsatz ausschließen.

Modell 2 — KI-Werkzeuge ohne geänderten Prozess

Der Regelfall in der Breite: 84 Prozent nutzen oder planen KI-Werkzeuge, während der Prozess unverändert bleibt. Ergebnis ist der DORA-Befund — mehr Durchsatz bei sinkender Stabilität. Der Gewinn ist real und wird an anderer Stelle teilweise wieder ausgegeben.

Modell 3 — KI-Einsatz mit definierten Prüfpunkten

Der KI-Anteil wird dort maximiert, wo die Studien Gewinne zeigen, und jede Freigabe läuft über eine Senior-Prüfung. Die Prüfkapazität wird als eigene Größe geplant, nicht als Restposten. Dieses Modell nimmt die 26 Prozent mit und begrenzt die Stabilitätswirkung.

Modell 4 — Prompt-getriebene Entwicklung ohne Prüfpfad („Vibe Coding“)

Für Prototypen und Machbarkeitsnachweise ein legitimer und schneller Weg. Für Systeme im Betrieb fehlt alles, was die Stabilitätsfrage beantwortet: Testabdeckung, Nachvollziehbarkeit, Verantwortlichkeit für die Freigabe. Ein Prototyp in zwei Tagen sagt nichts über die Zeit bis zum Betrieb.

Zehn Hebel für CIOs und Entwicklungsleitende

1. **Eine Baseline erheben, bevor das erste Werkzeug ausgerollt wird.** Durchlaufzeit je Vorgang, Fehlerquote nach Freigabe, Anteil zurückgerollter Änderungen. Einmal messen, aufschreiben, wegheften. Ohne diesen Wert ist jede spätere Aussage über den Gewinn eine Selbsteinschätzung — und die lag bei METR um 39 Prozentpunkte neben dem gemessenen Wert.
2. **Den Gewinn nach Aufgabentyp getrennt messen.** Ein Mischwert über alle Vorhaben verdeckt genau die Verteilung, aus der sich die Steuerung ergibt.
3. **Die KI dort ansetzen, wo die Studien Gewinne zeigen.** Neubau, wiederkehrende Bausteine, Tests, Dokumentation.
4. **Prüfkapazität als eigene Planungsgröße führen.** Wenn die Erzeugung um ein Viertel schneller wird, braucht die Prüfung mehr Zeit, nicht gleich viel.
5. **Freigaben an benannte Personen binden.** Die Verantwortung für einen freigegebenen Stand lässt sich nicht an ein Werkzeug delegieren.

6. **Senior-Zeit auf Urteil umwidmen, nicht auf Tempo.** Bei 7 bis 16 Prozent Ausführungsgewinn ist die teuerste Stunde im Team schlecht in der Erzeugung investiert und gut in Architektur, Review und Abnahme.
7. **Junior-Einsatz neu bewerten.** Der größte gemessene Effekt liegt bei kurzer Betriebszugehörigkeit. Das verändert die Rechnung für Teamzusammensetzung und Einarbeitung.
8. **Testabdeckung vor dem Tempo erhöhen.** Sie ist die Bedingung dafür, dass mehr Durchläufe nicht zu mehr Störungen führen.
9. **Regeln vor Agenten.** Bevor ein System eigenständig handelt, müssen Datenzugriff, Systemgrenzen und Freigaberegeln festliegen.
10. **Nach sechs Monaten gegen die Baseline rechnen — mit dem Ergebnis, das herauskommt.** Ein Gewinn von 15 Prozent, der belegt ist, trägt weiter als ein Faktor, den niemand prüfen kann.

Wie InnoShore das anders macht

Unser Modell für KI-native Softwareentwicklung folgt der Verteilung, die die Studien zeigen. Die Anforderungen entstehen im Gespräch mit Menschen, im Bau wird der KI-Anteil maximiert, und vor jeder Freigabe steht ein definierter Prüfpunkt mit Senior-Steuerung in Deutschland.

Der Gewinn wird gegen eine eingefrorene Schätzung gemessen und nach Aufgabenart getrennt ausgewiesen, statt als ein Wert über alles. Dazu gehört, dass wir die Grenzen der eigenen Messung mitveröffentlichen — welche Aufgabenart noch keinen Wert hat, wie viele Schätzungen vorab eingefroren waren und ab wann wir eine Aussage überhaupt als belastbar behandeln. Der Mess-Rahmen steht schriftlich und ist auf Anfrage einsehbar.

Was dazugehört, ist der Compliance-Baukasten: Datenzugriff, Nachvollziehbarkeit und Freigaberegeln werden pro Mandat konfiguriert. Bei einem Vorgehen, das den Anteil maschinell erzeugten Codes bewusst erhöht, ist die Frage, wer was freigegeben hat, keine Formalie.

Häufige Fragen

Was bedeutet KI-Produktivität in der Softwareentwicklung?

KI-Produktivität in der Softwareentwicklung ist der messbare Zeit- und Durchsatzgewinn, den Entwicklungsteams durch den Einsatz von KI-Werkzeugen erzielen. Gemessen wird sie an abgeschlossenen Aufgaben, Durchlaufzeiten oder Auslieferungsfrequenz — nicht an der Selbsteinschätzung der Beteiligten.

Wie hoch ist der Produktivitätsgewinn durch KI in der Entwicklung?

In drei randomisierten Feldexperimenten mit 4.867 Entwicklern lag er bei 26,08 Prozent mehr abgeschlossenen Aufgaben (Management Science, Februar 2026). Bei einer isolierten Standardaufgabe wurden 55,8 Prozent Zeitgewinn gemessen (Peng et al., 2023). Bei erfahrenen Entwicklern in eigenen Repositories wurde eine Verlangsamung um 19 Prozent gemessen (METR, 2025).

Warum widersprechen sich die Studien?

Sie messen verschiedene Situationen. Der Gewinn hängt vom Aufgabentyp und von der Repository-Kenntnis ab. Standardaufgaben ohne Altlasten liegen am oberen Ende, Arbeit an gewachsenem Code am unteren.

Profitieren erfahrene Entwickler weniger?

Ja. In den Feldexperimenten steigerten weniger erfahrene Entwickler ihre Ausgabe um 21 bis 40 Prozent, erfahrene um 7 bis 16 Prozent.

Verschlechtert KI die Codequalität?

Die Feldexperimente fanden keinen negativen Effekt auf die Build-Erfolgsquote, aber 38,38 Prozent mehr Kompilierungsvorgänge. Auf Systemebene zeigt der DORA-Report 2025 einen weiterhin negativen Zusammenhang zwischen KI-Einsatz und Auslieferungsstabilität.

Was ist die häufigste Schwierigkeit im Alltag?

Lösungen, die fast richtig sind. 66 Prozent der Entwickler nennen das als größte Frustration, 45,2 Prozent halten das Debuggen von KI-Code für zeitaufwendiger als bei menschlichem Code.

Wie viele Entwickler vertrauen KI-Werkzeugen?

3,1 Prozent geben hohes Vertrauen in die Genauigkeit an, 46 Prozent misstrauen ihr aktiv — bei gleichzeitig 84 Prozent, die die Werkzeuge nutzen oder nutzen wollen (Stack Overflow 2025).

Was heißt KI-nativ im Unterschied zu KI-gestützt?

KI-gestützt bezeichnet den Einsatz von KI-Werkzeugen in einem unveränderten Prozess. KI-nativ bezeichnet ein Vorgehen, dessen Ablauf, Rollen und Prüfpunkte um den KI-Einsatz herum gebaut sind.

Kann man den Gewinn im eigenen Unternehmen messen?

Ja, wenn vor dem Start eine Baseline erhoben wird. Geeignet sind Durchlaufzeit je Vorgang, Fehlerquote nach Freigabe und Anteil zurückgerollter Änderungen — Größen, die in den meisten Organisationen bereits erfasst werden.

Warum genügt die Einschätzung der Entwickler nicht?

Weil sie in der bestverfügbaren Messung falsch war. Die METR-Teilnehmer erwarteten 24 Prozent Beschleunigung, erlebten 19 Prozent Verlangsamung und glaubten anschließend an 20 Prozent Beschleunigung.

Ist Vibe Coding dasselbe wie KI-native Entwicklung?

Nein. Prompt-getriebene Entwicklung ohne Prüfpfad eignet sich für Prototypen. Für Systeme im Betrieb fehlen Testabdeckung, Nachvollziehbarkeit und eine benannte Verantwortung für die Freigabe.

Was ändert sich für die Teamzusammensetzung?

Der größte gemessene Effekt liegt bei kurzer Betriebszugehörigkeit. Senior-Zeit ist deshalb in Architektur, Review und Abnahme besser investiert als in die Erzeugung.

Welche Rolle spielt der AI Act?

Für die Entwicklung mit KI-Werkzeugen entstehen Pflichten vor allem aus der Rolle als Betreiber oder Anbieter des entstehenden Systems, nicht aus dem Werkzeugeinsatz selbst. Die Einordnung gehört in die Anforderungsphase.

Wie groß ist der Gewinn bei InnoShore-Projekten?

In der laufenden eigenen Messung liegt der Median bei 19,5 über acht Datenpunkte seit Juli 2026, die Spannbreite bei 15 bis 37 je nach Aufgabenart. Gemessen wird die Bauzeit einer Story gegen die eingefrorene klassische Schätzung derselben Story, aggregiert über den Median. Der ausgewiesene Wert enthält die menschliche Zeit für Aufgabenstellung, Prüfung und Verbuchung.

Warum liegen diese Werte so weit über den Studienwerten?

Weil sie etwas anderes messen. Die Studien messen KI-Werkzeuge in einem unveränderten Prozess. Ein KI-natives Vorgehen ändert Ablauf, Rollen und Prüfpunkte. Der Unterschied zwischen beiden Zahlen ist der Unterschied zwischen KI-gestützt und KI-nativ.

Für welche Arbeit gibt es keinen Messwert?

Für Arbeit mit unklarer Spezifikation, wandernden Anforderungen und subjektiver Iteration. Diese Aufgabenart ist in unserer Reihe null Mal gemessen, und wir nennen dafür keinen Faktor.

Ist die eigene Messung schon extern belastbar?

Nein, und das steht im eigenen Mess-Rahmen. Eine externe Aussage ist dort erst ab drei bis fünf abgeschlossenen Projekten und nur mit Segment-Angabe vorgesehen. Die Zahlen hier sind ein offengelegter Zwischenstand. Von acht Schätzungen waren zwei vor dem Bau eingefroren, die übrigen retrospektiv; eine unabhängige Vergleichsschätzung steht als nächster Härtegrad an.

Wie lange hält die aktuelle Studienlage?

Die hier ausgewerteten Studien stammen aus 2023, 2025 und 2026. Beide Leitquellen werden fortgeschrieben. Dieser Artikel wird bei neuen Erhebungen aktualisiert; das Aktualisierungsdatum oben zeigt den Stand.

Thematische Vertiefungen

KI-native Softwareentwicklung für den Mittelstand

KI-Agenten im Unternehmenseinsatz

Enterprise AI einführen

Quellen

- Cui, K. Z.; Demirer, M.; Jaffe, S.; Musolff, L.; Peng, S.; Salz, T.: „The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers." Management Science, 27.02.2026. Drei randomisierte kontrollierte Studien bei Microsoft (1.746 Entwickler), Accenture (320) und einem anonymen Fortune-100-Unternehmen (3.054), Laufzeit zwei bis acht Monate. – https://pubsonline.informs.org/doi/10.1287/mnsc.2025.00535
- Dieselbe Studie, Arbeitsfassung Februar 2025, 28 Seiten — Fundstelle für die Effekte nach Erfahrungsstufe und für die Angaben zur Build-Erfolgsquote. – https://economics.mit.edu/sites/default/files/inline-files/draft_copilot_experiments.pdf
- Peng, S.; Kalliamvakou, E.; Cihon, P.; Demirer, M.: „The Impact of AI on Developer Productivity: Evidence from GitHub Copilot." arXiv:2302.06590, eingereicht 13.02.2023. Kontrolliertes Experiment, Aufgabe: HTTP-Server in JavaScript. – https://arxiv.org/abs/2302.06590
- METR: „Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity", 10.07.2025. Randomisierte kontrollierte Studie, 16 erfahrene Entwickler, 246 Aufgaben in eigenen Repositories. – https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/
- DORA / Google Cloud: „State of AI-assisted Software Development", 23.09.2025. Knapp 5.000 Fachleute weltweit, über 100 Stunden qualitative Erhebung. – https://cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report
- Stack Overflow: „Developer Survey 2025". Über 49.000 Antworten aus 177 Ländern. – https://survey.stackoverflow.co/2025/