Status quo: KI in der Software-entwicklung

Wo stehen wir aktuell?
Wie verändert KI die Entwicklung von Software?
Welche Chancen, Risiken und neuen Engpässe entstehen?

Maik Arnold und Johannes Balk

webit! Kundentag · 05.11.2026



Agenda

Rückblick

KI-Einsatz heute

Chancen & Risiken

Fazit



Agenda

Rückblick

KI-Einsatz heute

Chancen & Risiken

Fazit



ChatGPT macht KI massentauglich

- 30. November 2022: ChatGPT erscheint (Basis: GPT-3.5)
- KI wird über Nacht massentauglich
- Ca. 100 Mio. Nutzer nach nur 2 Monaten – der Beginn des Hypes

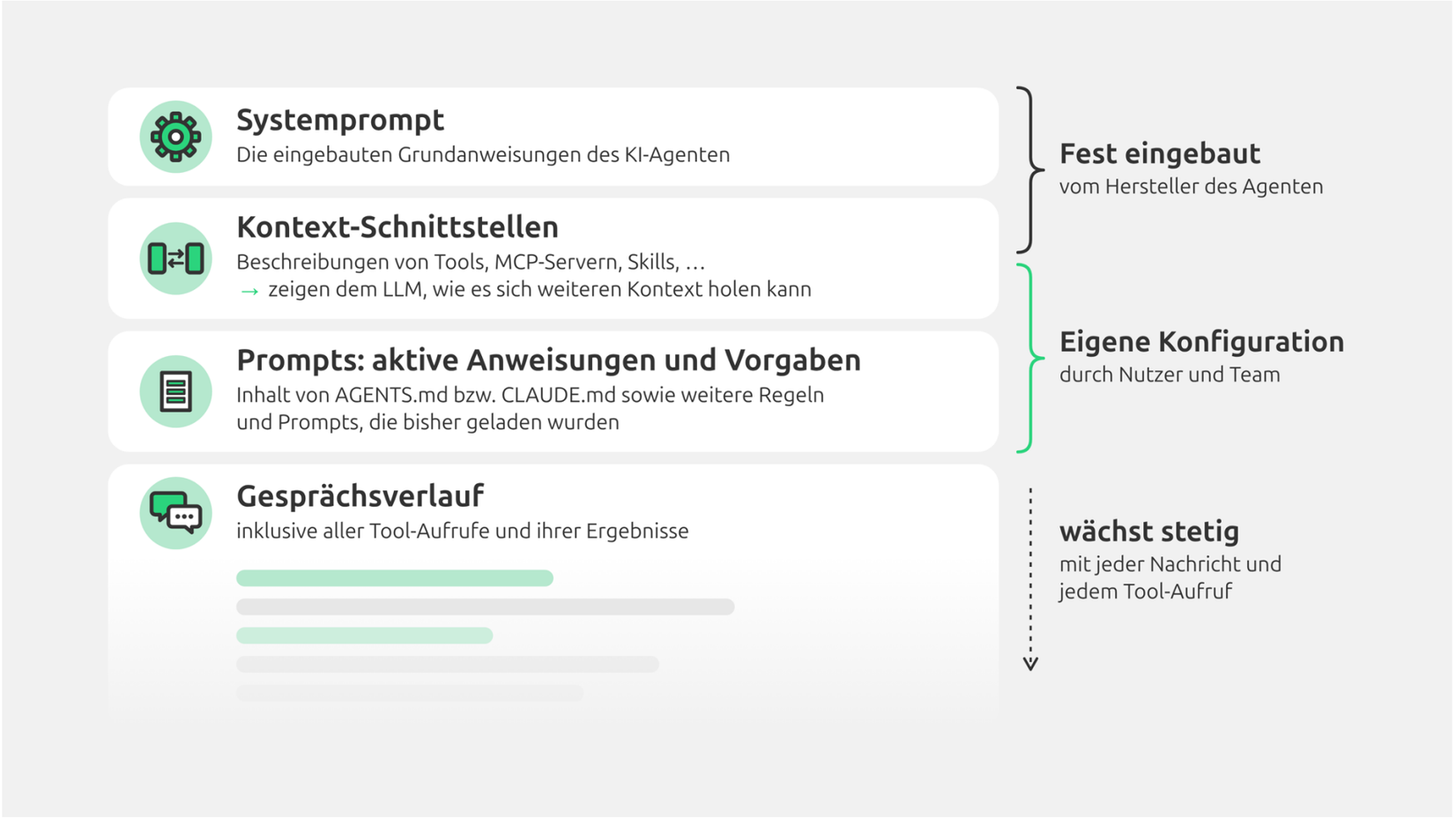
100 Mio. Nutzer

in nur 2 Monaten

Erste Versuche mit KI-generiertem Quellcode

- Damals: reiner Web-Chat – Frage rein, Antwort raus
- Erste Berührungspunkte mit Quellcode-Generierung
- Noch sehr rudimentär – komplexere Programmlogik nicht umsetzbar

Was ist ein Kontextfenster?



GitHub Copilot bringt wenig Effizienzgewinn

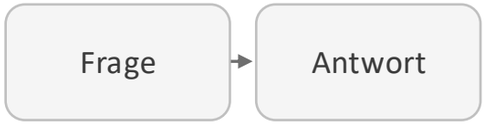
- Einsatz von GitHub Copilot: KI vervollständigt Quellcode beim Tippen
- KI sieht nur einen kleinen Ausschnitt des Projekts
- Ergebnisse durchwachsen – mitunter sogar störend und hinderlich



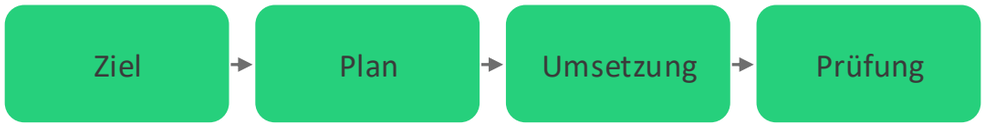
Erste brauchbare KI-Coding-Agenten

- Erste nutzbare KI-Coding-Agenten erscheinen – allen voran Claude Code
- Der Fokus verschiebt sich vom „Chatten“ zum „Handeln“
- KI erfasst noch nicht alles – aber komplexere Geschäftslogik wird machbar
- Der Begriff Vibe Coding etabliert sich

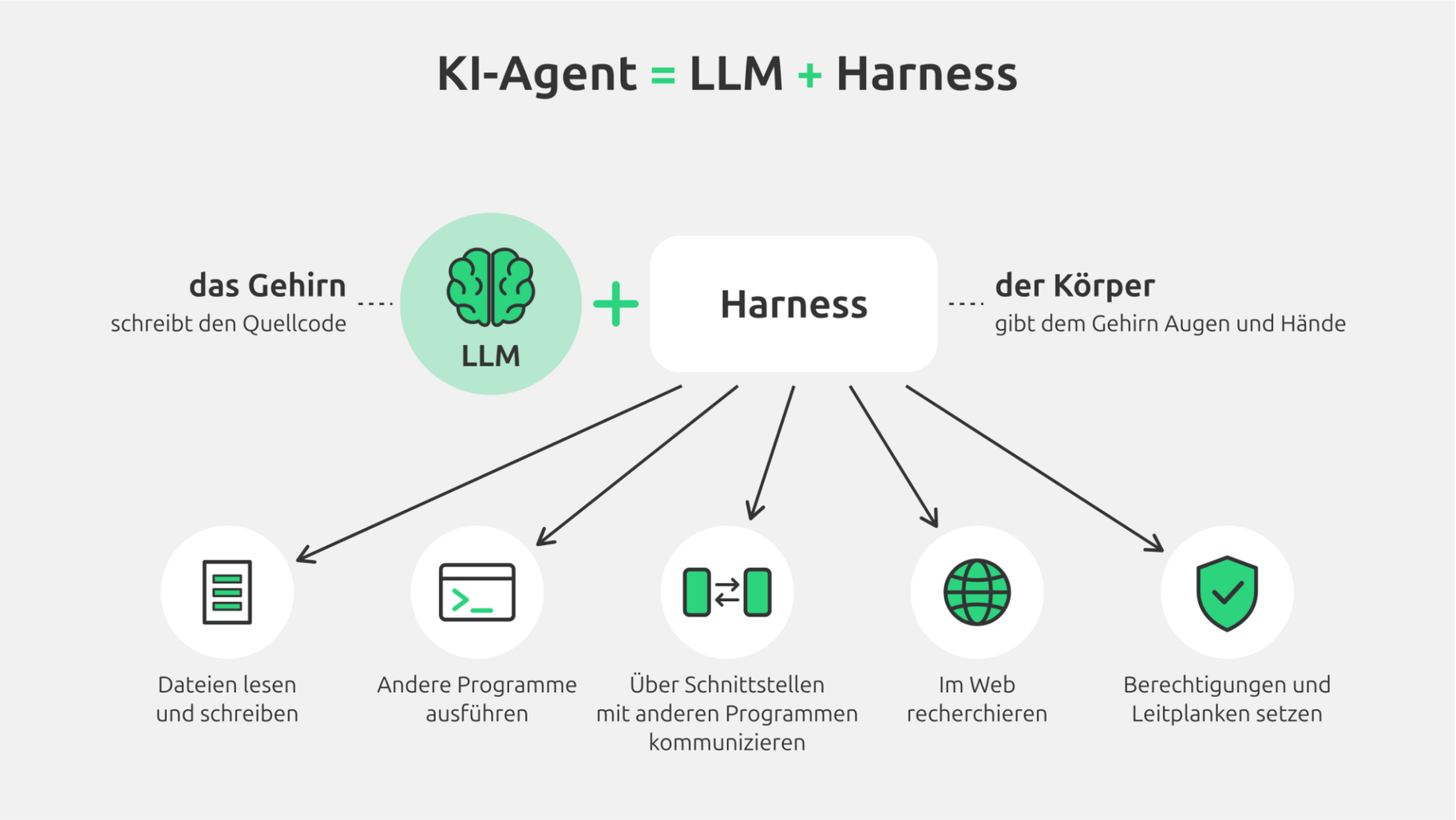
Chat



KI-Agent



Was genau ist ein KI-Agent?



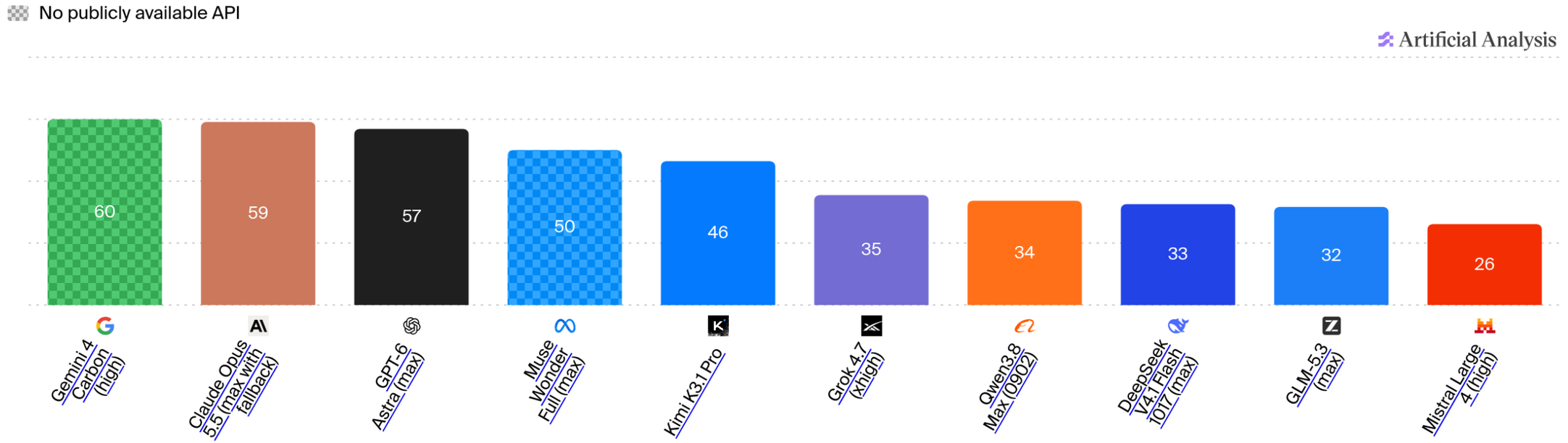
KI-Agenten lösen komplexe Aufgaben – auch in großen Softwareprojekten

- Neue Spitzenmodelle: Claude Opus 5.5, GPT-6 Astra (OpenAI)
- Komplexe Aufgaben auch in sehr großen Softwareprojekten lösbar
- KI-Agenten prüfen ihre Arbeit selbst – z. B. mit automatischen Tests
- Modelle sind gut genug – der Fortschritt kommt jetzt über den Harness

Aktuelle Sprachmodelle

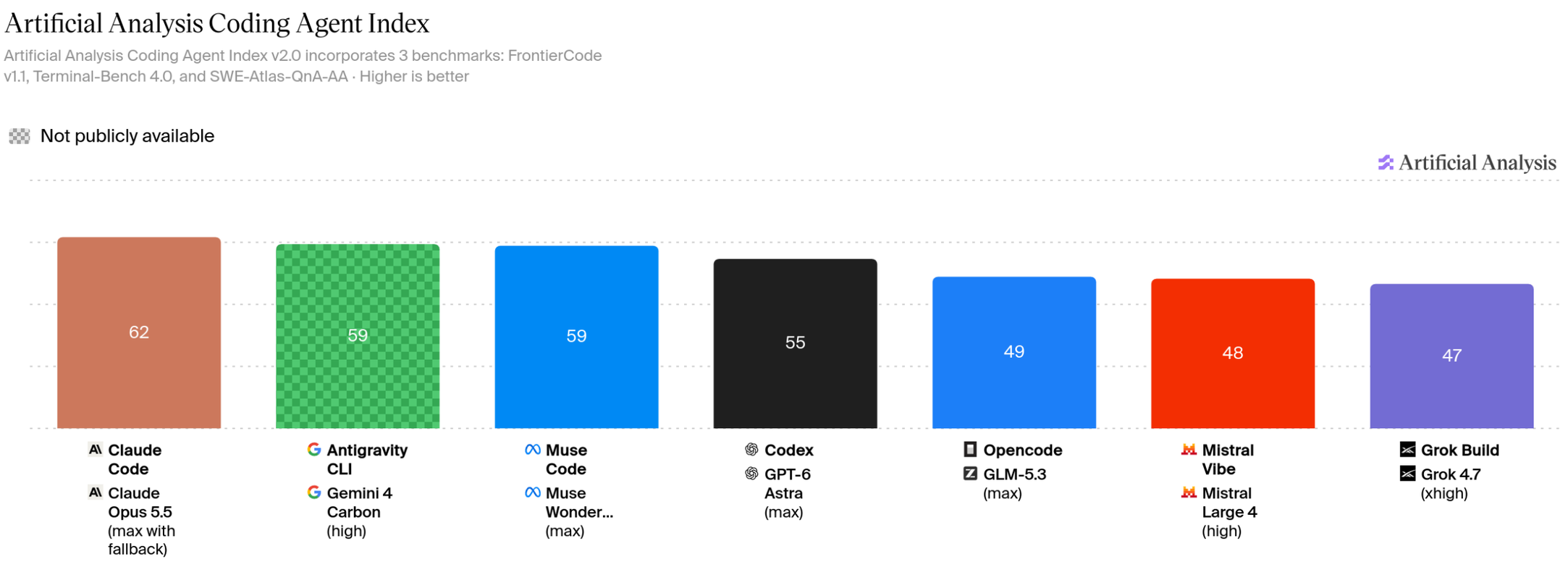
Artificial Analysis Intelligence Index

Artificial Analysis Intelligence Index v5.0.1 incorporates 9 evaluations: AA-Briefcase v1.1, AutomationBench-AA, Terminal-Bench 4.0, FrontierCode 2.0, AA-Omniscience v1.1, GDP.pdf, HLE-Diamond, CritPt v2 (Hard Subset), and Terminal-Bench-Science 0.1



Quelle: Artificial Analysis, Intelligence Index v5.0.1

Aktuelle KI-Agenten



Quelle: Artificial Analysis, Coding Agent Index v2.0

Agenda

Rückblick

KI-Einsatz heute

Chancen & Risiken

Fazit



Mögliche Einsatzfelder



Anforderungsanalyse und Konzeption unterstützen



Quellcode schreiben



Fehler finden und beheben



Quellcode- und Sicherheits-prüfungen unterstützen



Dokumentation erstellen (z. B. für Schnittstellen)

Softwareentwicklungszyklus mit KI-Coding-Agenten



1	2	3	4	5
Anforderung klären	Spezifikation: genau beschreiben, was gebaut werden soll	KI-Agent setzt um – mit hinterlegten Projektregeln und Fachwissen	Automatische Tests prüfen das Ergebnis	Entwickler prüft und gibt frei (Quellcode-Review)

Mensch

KI-Agent und Automatik

KI schreibt den Quellcode

- In einigen Projekten wird der Großteil des Quellcodes von KI-Agenten geschrieben
- Der Engpass liegt jetzt bei:
  - Abstimmung der Anforderungen
  - Technischen Spezifikationen
  - Qualitätssicherung (Quellcode-Review, Tests)

Wie kann der Effizienzgewinn ermittelt werden?

Was keine Metrik ist



Codezeilen
Mehr Code heißt nicht mehr Nutzen



Verbrauchte Token
Messen den Aufwand der KI, nicht das Ergebnis



Gefühlte Steigerung
Subjektiv und nicht überprüfbar



Geschätzter vs. benötigter Aufwand
Die Schätzung rechnet die Umsetzung mit KI bereits ein



Unser Ansatz

Features pro Monat

Wir zählen die umgesetzten Features pro Monat und vergleichen die Werte vor und nach der Einführung von KI.

Auch diese Messung ist nicht 100 % genau: Jedes Feature wird nur einmal umgesetzt, nie parallel mit und ohne KI.

Wie viel Effizienzgewinn ist realistisch?

Realistisch:

+20–40 %

mehr umgesetzte Features in gleicher Zeit

- Unsere Erfahrung bei webit! – bis zu 40 % mehr umgesetzte Features in gleicher Zeit
- Microsoft (2026) – +24 % umgesetzte Änderungen, bei täglicher Nutzung +50 %
- NBER (2026) – +30 % veröffentlichte Versionen

Quelle: Microsoft, arXiv 2607.01418 (2026); NBER Working Paper 35275 (2026)

Wie erreichen wir diesen Effizienzgewinn?

Was wird wichtiger

- Spezifikationen der Anforderungen werden noch wichtiger
- Quellcode-Reviews werden wichtiger
- Richtlinien werden wichtiger
- Wenn mehr Tickets umgesetzt werden, müssen auch mehr Tickets spezifiziert, getestet und abgenommen werden

Wie erreichen wir diesen Effizienzgewinn?

Unsere Empfehlung

- Kürzere Release-Zyklen, häufiger ausliefern
- Neue Funktionen per Schalter freigeben (Feature-Flags)
- Einsatz von automatisierten Tests, Analysetools, Sicherheitsscans: Die besten Ergebnisse liefern KI-Agenten dann, wenn sie sich selbst überprüfen können

KI-Coding-Agenten optimal nutzen

Skills sind wiederverwendbare Anweisungen und Arbeitsabläufe, die der KI-Agent in seinen Kontext lädt. Sie geben vor, wie er bei einer Aufgabe vorgehen soll.

Beispiel: Skill „Implement Feature“

1	2	3	4	5	6
Verstehen	**Klären**	**Planen**	**Freigeben**	**Umsetzen**	**Prüfen**
Anforderung und Fachwissen lesen	Rückfragen bei Unklarheiten	Umsetzungsplan erstellen	Team akzeptiert den Plan	Tests zuerst, Code danach	Sicherheitsscan des Codes

Je genauer, desto besser

Präzise Vorgaben liefern bessere Ergebnisse

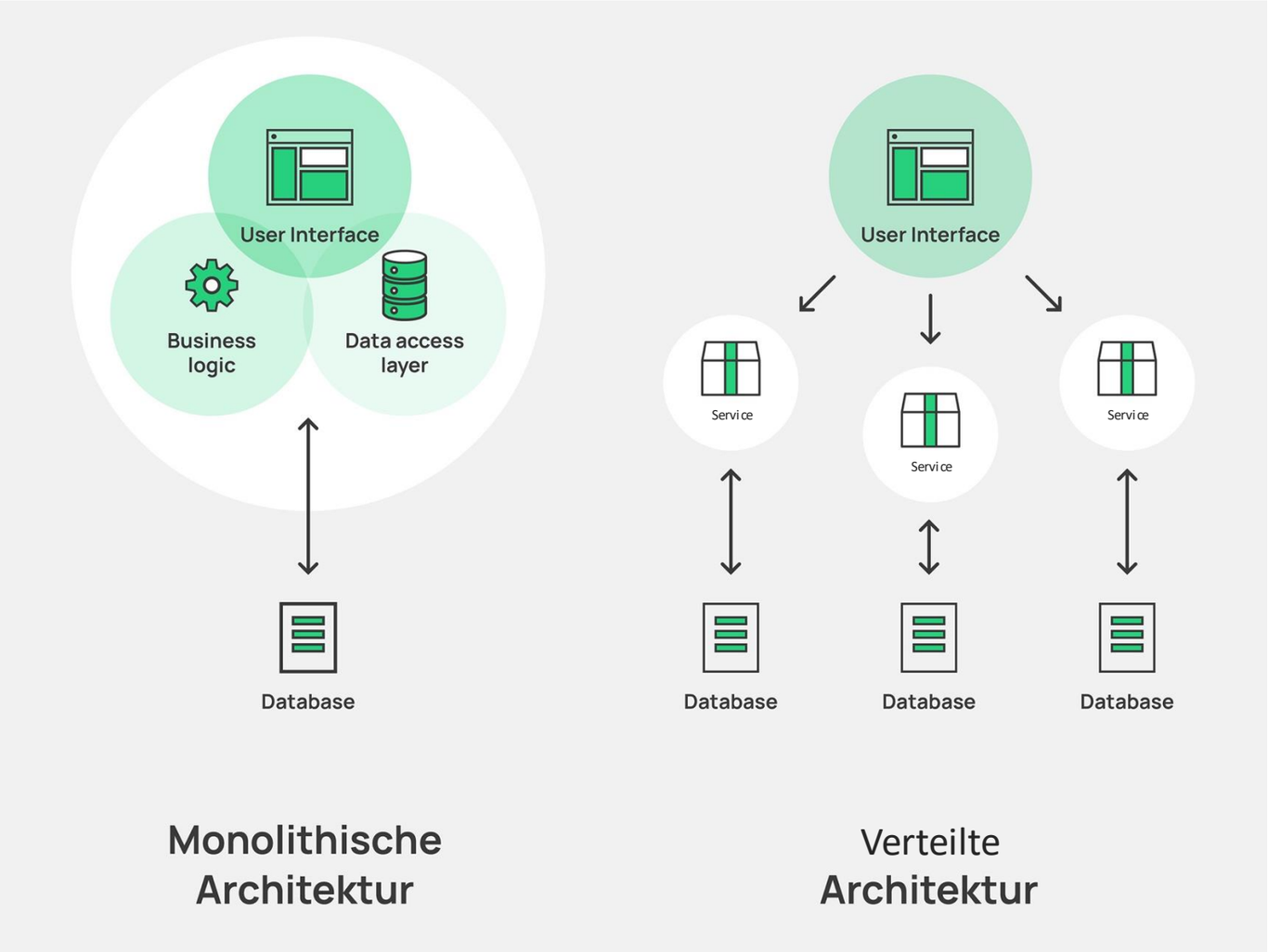
Nicht nur für Code

Auch für Texte, Meeting-Protokolle oder Tickets

Nur aus sicheren Quellen

Fremde Skills aus dem Internet vorher prüfen

Softwarearchitektur in Zeiten von KI



Wie beeinflusst die KI die Architektur-Entscheidung?

Vor KI

Rahmenbedingungen bestimmen die Architektur

Skalierbarkeit, Ausfallsicherheit, Teamstruktur und Sicherheit gaben den Ausschlag

Einführung von KI

Eine neue Frage kommt hinzu

Mit welcher Architektur entwickeln KI-Agenten am effizientesten Software?

Erste KI-Agenten

Kleine Kontextfenster: Microservices empfohlen

Agenten erfassten nur wenig Code auf einmal, kleine Dienste passten dazu

Neuere KI-Agenten

Größere Kontextfenster: auch Monolithen funktionieren

Agenten arbeiten nun auch in monolithischen Architekturen effizient

Heute

KI arbeitet in beiden Architekturen effizient

Die Entscheidung fällt weiterhin anhand der bisherigen Fachlichkeit

Agenda

Rückblick

KI-Einsatz heute

Chancen & Risiken

Fazit



Schnellere Feature-Entwicklung

Risiken

- Quellcode-Generierung wird schneller und damit auch die Feature-Entwicklung
  - Risiko: Es entstehen Features, die niemand braucht

  → Inkonsistenzen in der Anwendung

  → Anwendungen werden schwerer wartbar

  → Mehr Quellcode kann auch zu mehr Sicherheitsproblemen führen

Unsere Empfehlung

- Product Owner bewertet jedes Feature nach Nutzen
- Regelmäßig aufräumen: Ungenutztes entfernen

Sicherheitslücken im KI-generierten Quellcode

Risiken

- KI-Quellcode funktioniert meist – ist aber nicht automatisch sicher
- Mehr Quellcode in kürzerer Zeit → mehr zu prüfen

Unsere Empfehlung

- Automatische Sicherheitsprüfungen bei jeder Änderung
- Menschliches Review bleibt Pflicht – KI-Review nur zusätzlich

Abhängigkeit von der KI – Kompetenzverlust

Risiken

- Entwickler verlieren das tiefere Verständnis für den Quellcode
- Fachwissen und Domänenwissen gehen verloren
- Junior-Entwickler erlernen nicht die notwendigen Programmierkenntnisse, um die Ergebnisse der KI bewerten zu können

Unsere Empfehlung

- Quellcode-Review weiterhin durch Menschen
- Domänenwissen gezielt im Team halten
- Ausbildung der Junior-Entwickler muss angepasst & erweitert werden

Abhängigkeit vom KI-Anbieter (Vendor Lock-in)

Risiken

- Abhängigkeit von einem externen KI-Unternehmen
- Preise und Konditionen bestimmt der Anbieter

Unsere Bewertung

- Aktuell geringes Risiko – starker Wettbewerb
- Wechsel zu anderen KI-Agenten ist relativ leicht möglich

KI-Kosten

Risiken

- Kosten bei Abrechnung nach Verbrauch können außer Kontrolle geraten
- Preis pro Aufgabe sinkt aktuell, aber Gesamtausgaben steigen durch mehr Nutzung von KI
- KI-Anbieter erhöhen die Preise

Unsere Empfehlung

- Bei Abrechnung nach Verbrauch Kostenlimits setzen
- Passendes Modell je Aufgabe verwenden
- Erst prüfen, ob der Einsatz von KI nötig ist oder ob ein klassisches Programm ausreichend ist
- Open-Weights-KI-Modelle auf eigener Infrastruktur betreiben (lokal oder in der Cloud)

Datenschutz & Datensicherheit

Risiken

- Quellcode und Daten werden bei externen KI-Anbietern verarbeitet
- KI-Agenten können Daten anderer Bereiche des Entwicklerrechners abgreifen und versenden
- Werden Passwörter oder andere geheime Daten an die KI übertragen, müssen diese sofort geändert werden

Unsere Empfehlung

- Open-Weights-KI-Modelle auf eigener Infrastruktur betreiben (lokal oder in der Cloud)
- KI-Agenten in einer Sandbox ausführen, isoliert vom restlichen System
- API-Keys, Passwörter und Tokens verschlüsselt ablegen, ohne Zugriff für den KI-Agenten

Agenda

Rückblick

KI-Einsatz heute

Chancen & Risiken

Fazit



Unser Fazit: KI-Softwareentwicklung 2026

- Qualität des generierten KI-Quellcodes wird immer besser
- Effizienzsteigerungen sind möglich, wenn die Rahmenbedingungen korrekt angewendet werden
- Product Owner: schnellere Entwicklung → mehr Anforderungen und Abnahmen
- Entwickler: vom Quellcode-Schreiben hin zum Spezifizieren und Steuern von KI-Agenten – ein echtes Umdenken

> Agentisches Arbeiten ist die Zukunft der Softwareentwicklung

Danke fürs Zuhören!



Kontakt

Bärensteiner Straße 30

01277 Dresden

Telefon 0351 46766 - 0

Telefax 0351 46766 - 66

info@webit.de / www.webit.de

webit!

Haben Sie noch Fragen?



Kontakt

Bärensteiner Straße 30

01277 Dresden

Telefon 0351 46766 - 0

Telefax 0351 46766 - 66

info@webit.de / www.webit.de

webit!