

INNOQ TECHNOLOGY DAY 2025 / 20.11.2025

Erst agil, dann agentisch



DANIEL WESTHEIDE
SENIOR CONSULTANT

Was AI-Anbieter versprechen

- 10 x schnellere Softwareentwicklung
- AI-Agents als Junior Developer
- Nie mehr selbst lästigen Boilerplate-Code schreiben

Faros Report 2025

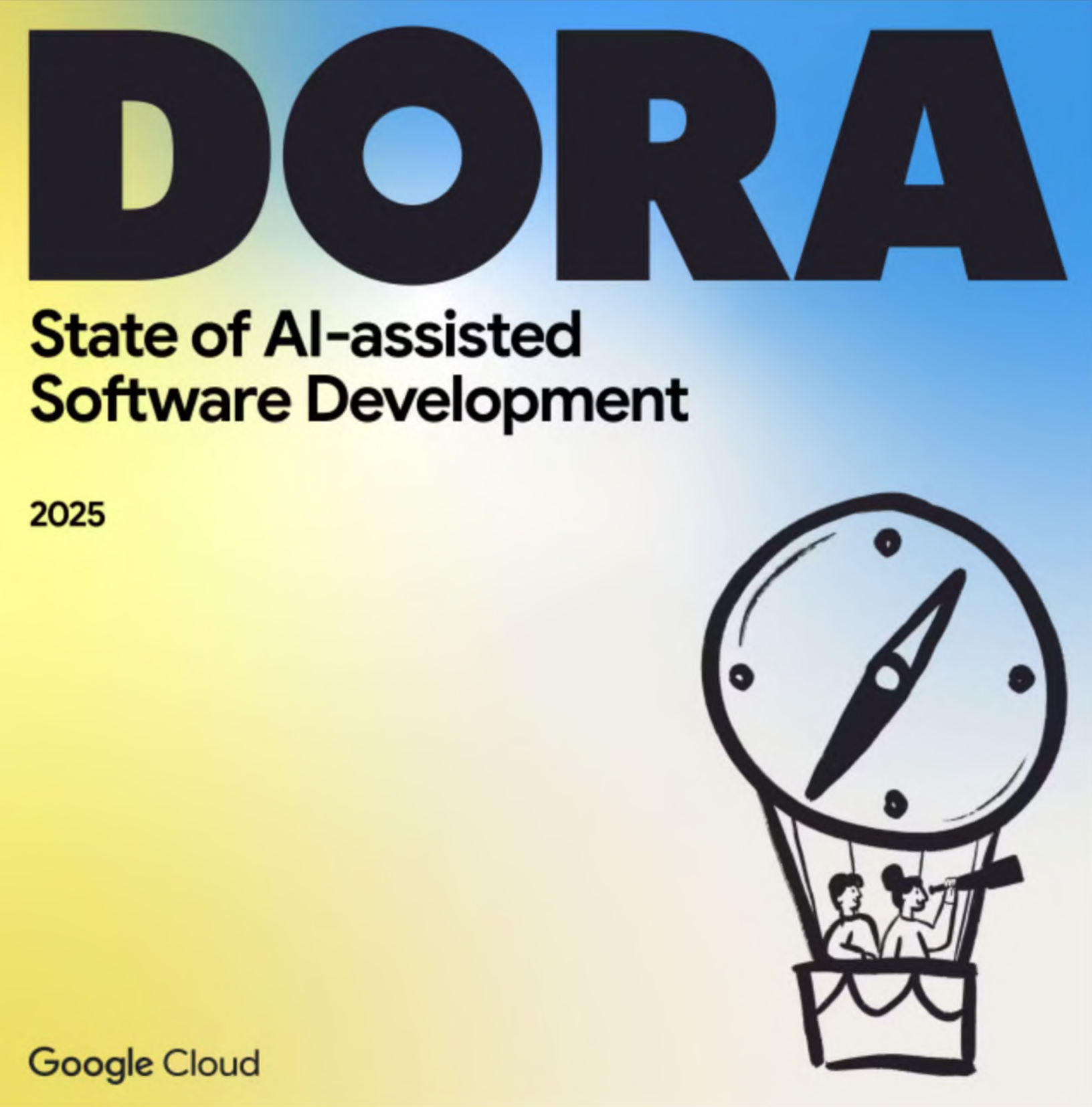
- Entwickler:innen mit AI
 - schließen 21% mehr Aufgaben ab
 - mergen 98% mehr Pull Requests
- Aber:
 - keine kürzeren Lead Times
 - 9% mehr Bugs
 - keine Verbesserungen von Metriken auf Unternehmensebene

The AI Productivity Paradox: AI Coding Assistants Increase Developer Output, But Not Company Productivity

What Data from 10,000
Developers Reveals about
Impact, Barriers, and the
Path Forward

FAROS

Insights from Faros AI's analysis of 1,255
teams across leading software organizations
July 2025

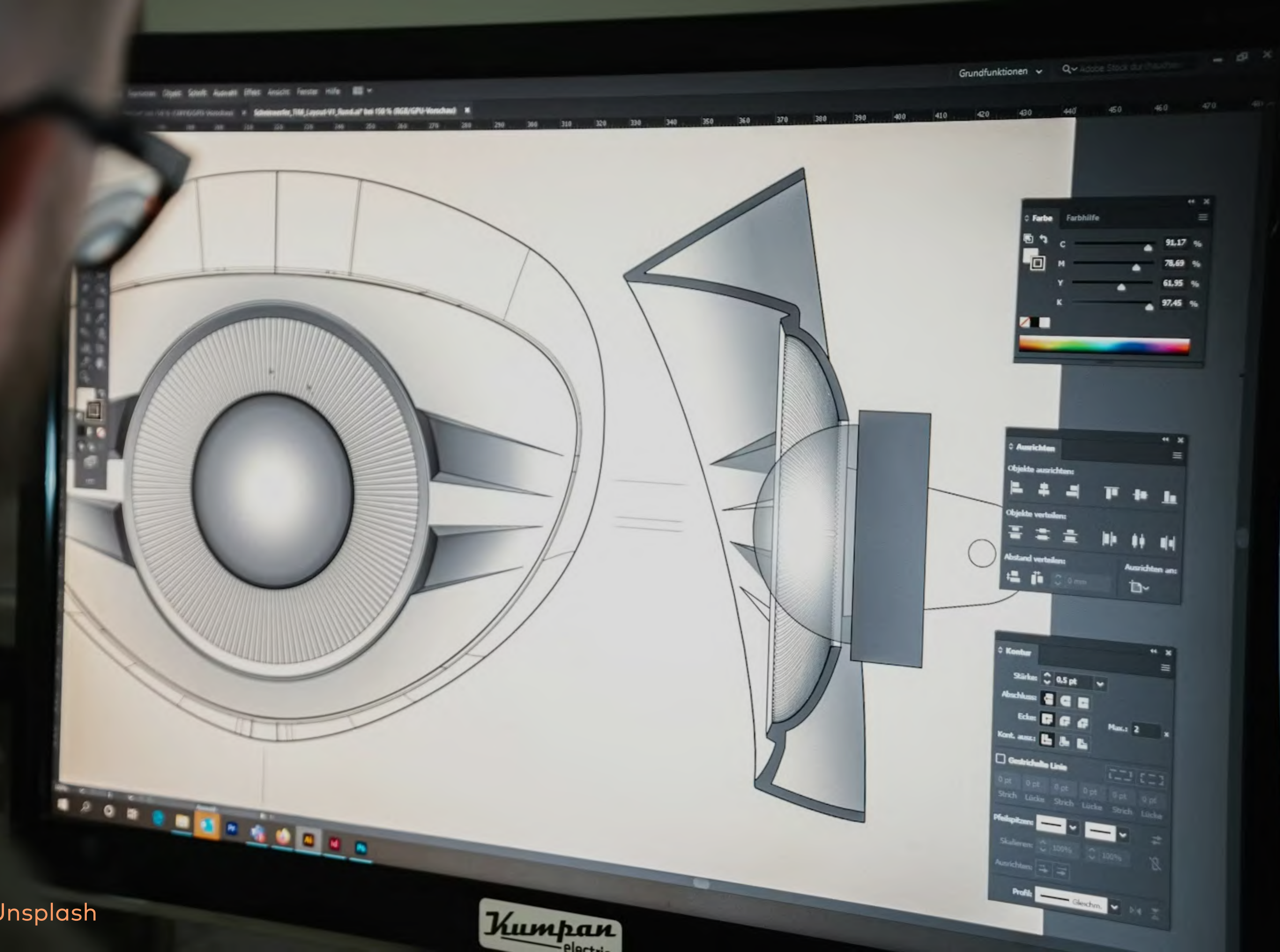


DORA Report 2025

- höherer Durchsatz bei Teams, die AI nutzen
- aber auch größere Instabilität

Warum?

**Wir
müssen
reden.
Über
CAD.**



Phase 0: Pre-CAD

- Ingenieur:innen und technische Zeichner:innen im gleichen Raum
- Kollaborativ und iterativ
- Kurze Feedback-Schleifen

Phase 1: 1960er bis frühe 1980er

- CAD auf Großrechnern in eigenen klimatisierten Räumen
- teuer, langsam und schwierig zu erlernen
- Neue Rolle des CAD Operators
- zentralisiert, eigene Abteilung
- Silos und Hand-offs
- schlechtere Kommunikation
- langsamere Feedback-Schleifen
- sinkende Qualität der Designs

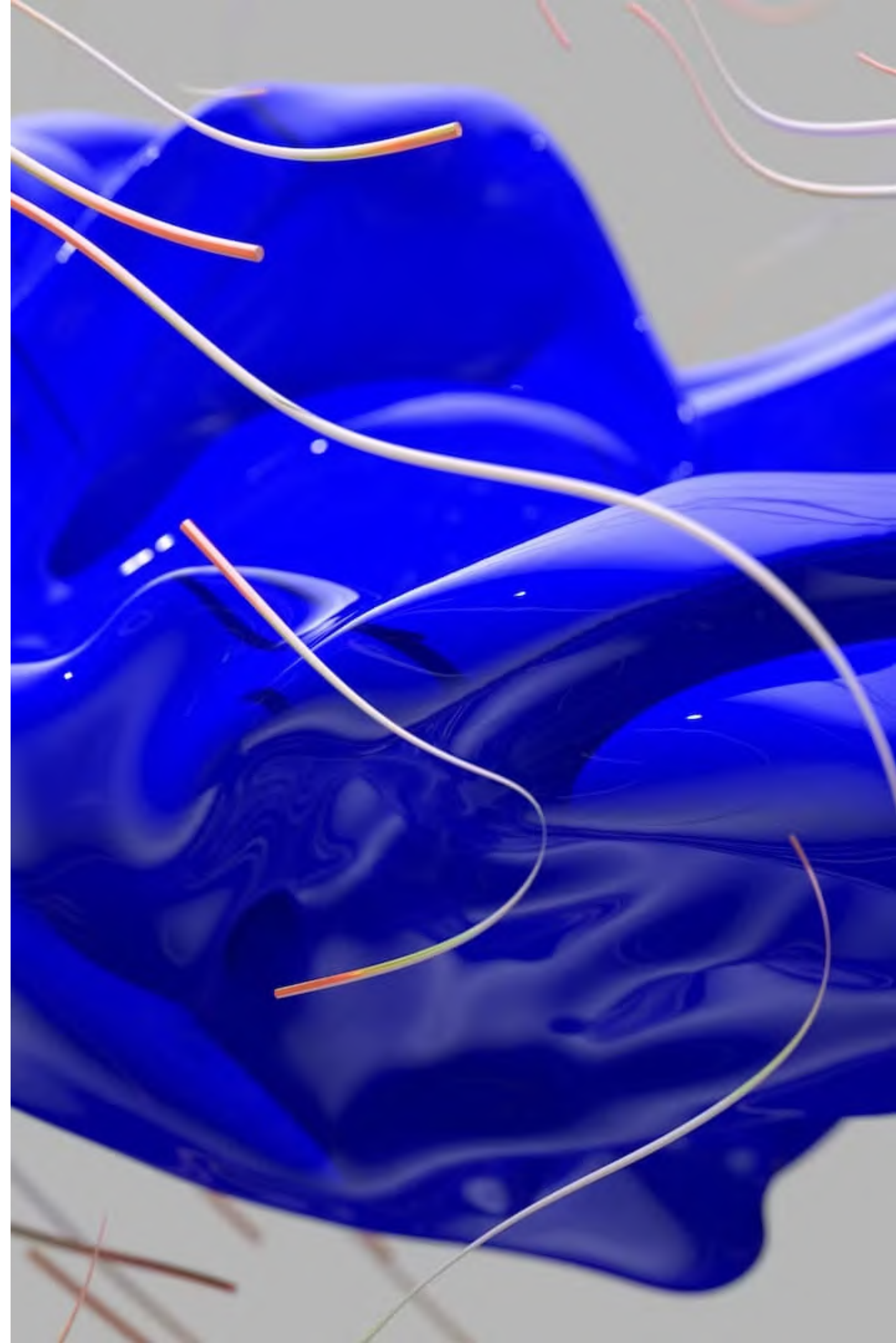
Phase 2: 1980er bis 1990er

- CAD wird zum Standardgut auf Desktop-PCs
- Preisverfall von Software und Hardware
- Ingenieure nutzen CAD-Software selbst
- Rolle des CAD Operators verschwindet
- keine Hand-offs mehr
- schnelle Feedback-Schleifen wieder hergestellt

**Ok, Daniel, aber
was hat das mit
AI zu tun?**






Gleiches Muster, nur schneller

- **CAD Operator:** ca. 30 Jahre von der Einführung bis zum Verschwinden
- **Prompt Engineer:** ca. 3 Jahre vom Aufkommen bis die Rolle absorbiert wurde



Technologie verändert Organisationen

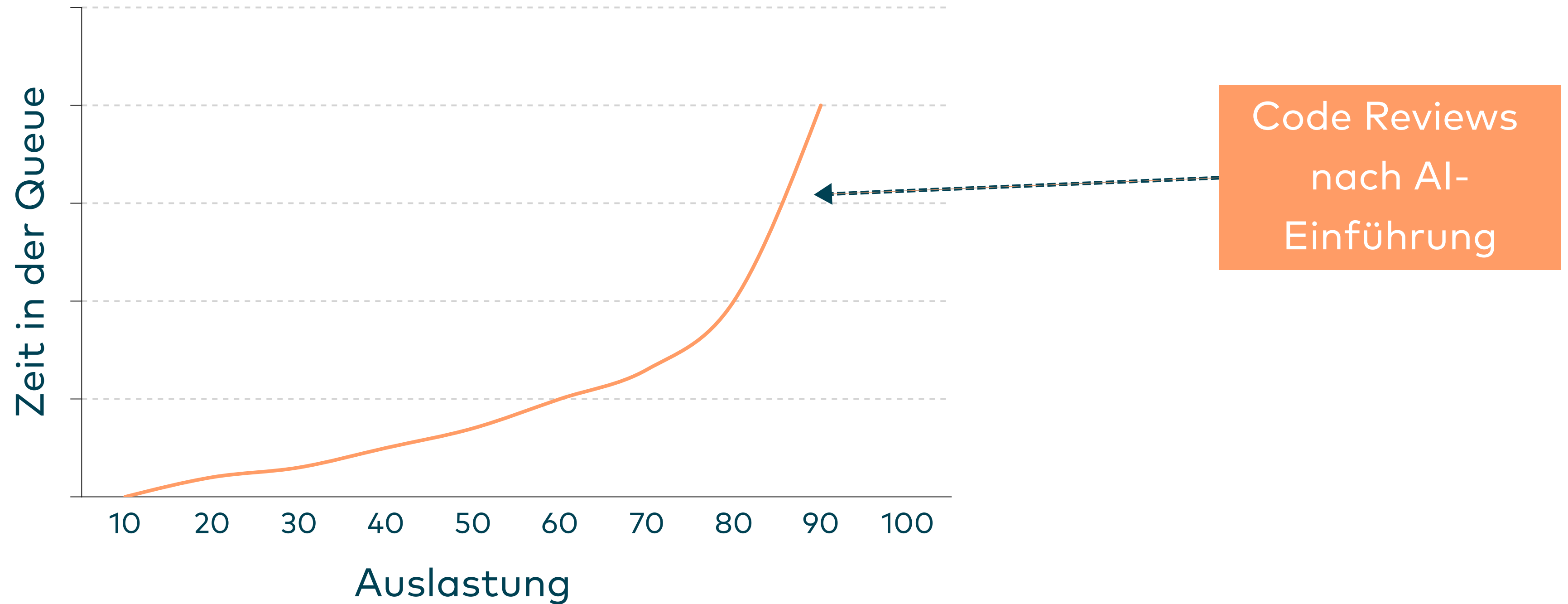
Neue Technologien führen unweigerlich zu Veränderungen in:

-  Organisationsstrukturen
-  Prozessen
-  Rollen und Karrierewegen
-  Machtdynamiken
-  Wissensverteilung

Gestaltungsspielraum: Wir können diese Transformation aktiv gestalten

Warum AI ohne Fundament scheitert

Queuing Theory



BLOG-POST

Context Engineering: Umgang mit der Komplexität von KI-generiertem Code

Kleine Änderungen waren schon immer eine gute Idee

24. Oktober 2025

KI-Tools steigern die Produktivität von Entwicklern beim Schreiben von Code, können jedoch Code-Reviewer mit zu umfangreichen Änderungen überlasten. Dieser Artikel zeigt praktische Strategien zur Kontextverwaltung in der KI-unterstützten Entwicklung, die Code verständlich, Reviews handhabbar und die Produktivität von Teams tatsächlich verbessern. Kleine Aufgabenbereiche waren schon immer gute Praxis – mit KI sind sie unerlässlich.

10 Minuten Lesedauer



JOY
HERON

Faros Report 2025

Abgeschlossene Aufgaben	+21%
Gemergte Pull-Requests	+98%
Code-Review-Dauer	+91%
Pull-Request-Größe	+154%

Kein Effekt auf Organisationsebene

The AI Productivity Paradox: AI Coding Assistants Increase Developer Output, But Not Company Productivity

What Data from 10,000
Developers Reveals about
Impact, Barriers, and the
Path Forward

FAROS

Insights from Faros AI's analysis of 1,255
teams across leading software organizations
July 2025

DX AI-assisted engineering

Q4 impact report

- Obwohl AI-Nutzung rapide steigt, stagniert der (wahrgenommene) Produktivitätszuwachs
- Hypothese: Um weiter profitieren zu können, sind Änderungen der Organisation und Prozesse notwendig

Weitere Queues und Bottlenecks

Product Owner

- AI-augmentierte Entwickler:innen implementieren in gleicher Zeit deutlich mehr Features
- Product Owner müssen
 - mehr **Features testen und abnehmen** — die Auslastung steigt, die Wartezeit vervielfacht sich
 - **schneller User Stories ausarbeiten** und Anforderungen vom Fachbereich sammeln
- Lead Time wird bestimmt durch den Schritt mit der höchsten Auslastung
- Product Owner und Fachbereich werden zum Bottleneck





Zentralisierte Architekturarbeit

- AI-augmentierte Entwickler:innen implementieren schneller
- Im gleichen Zeitraum mehr architektonisch relevante Entscheidungen notwendig
- Architektin oder Architektur-Review-Board wird zum Bottleneck
- Warte-Zeit in der der Architekturentscheidungs-Schlange steigt um ein Vielfaches

Software Delivery

- langsame CI/CD-Pipelines
 - monolithische Anwendung
 - langsame Test-Suite
 - fehlende Parallelisierung
 - Redundanzen
- manuelle Genehmigungen
- große, seltene Deployments
- Teams entwickeln mit AI mehr Features in gleicher Zeit
- Wartezeit in der Build- und Deployment-Queue steigt exponentiell

Abnehmende Qualität

- DORA 2025:
 -  Durchsatz
 -  Stabilität
- Faros AI 2025:
 -  Bugs
 -  Größe von Pull Requests

"[..]while teams are adapting for speed, their underlying systems have not yet evolved to safely manage AI-accelerated development" (DORA Report 2025)

LGTM! 
Ship it! 

Das gemeinsame Muster

- AI erhöht lokale Geschwindigkeit
- Entscheidungen bleiben zentralisiert
- Queues werden immer länger
- Qualität nimmt ab

Erst agil, dann agentisch!

Was ich mit agil NICHT meine

- Daily Standups
- Sprints
- Story Points
- Planning Poker
- Sprint Reviews
- Refinement Sessions
- jegliche reglementierte Prozesse oder Rituale

Was ich mit agil meine

Agiles Manifesto

Selbstorganisation,
Embracing Change,
Sustainable Pace

DevOps

Blameless Postmortems,
Psychologische Sicherheit,
Continuous Delivery

Ökonomische Frameworks

Cost of Delay, Lifecycle
Profits, Queuing Theory
(Reinertsen)

Product Thinking

Schnelles User
Feedback,
Hypothesen
getriebene
Entwicklung

End-to-end-Ownership

Fast Feedback
Flow
dezentrale Entscheidungen
sichere Experimente

Organisationsdesign

Conway's Law, Team
Topologies (Stream-aligned
Teams)

Fünf Grundlagen für agentische AI

Schnelle Feedback-Schleifen

AI-Agents ermöglichen deutlich schnellere Feature-Entwicklung.
Feedback muss genauso schnell kommen.

AI-Entwicklung + Schnelles Feedback:

- Mehr Experimente pro Zeiteinheit
- Schneller lernen, was funktioniert
- Weniger Zeit mit falschen Lösungen

AI-Entwicklung + Langsames Feedback:

- Viel Code generiert, aber spät getestet
- Risiko multipliziert

Schnelle Feedback-Schleifen

Drei kritische Dimensionen:

1. Schnell ausliefern
 - Schnelle CI/CD-Pipeline, Testpyramide, Feature Flags
2. Schnell technisches Feedback
 - Security Scans, Performance-Checks
 - Probleme umgehend sehen
3. Schnell fachliches Feedback
 - Monitoring, Analytics, Observability, A/B-Tests
 - Schnell von echten Usern lernen

Kleine Batch-Größen

- **Schnelleres Feedback:** früher lernen, Kosten für Korrekturen minimieren
- **Reduzierte Variabilität:** z.B. vorhersagbare Review-Zeiten statt "30 Min oder 3 Stunden?"
- **Geringeres Risiko:** kleinere Deployments, kleinerer Blast Radius
- **Besserer Flow:** kürzere Wartezeiten in der Queue

Mit AI Agents besonders wichtig, da:

- sie dazu verleiten, sehr große Batches generieren zu lassen
- die kognitiven Fähigkeiten der Reviewer begrenzt sind

Aber: Fachliche Abnahme von großen Batches behindert weiter den Flow!

Dezentrale Entscheidungen

Das Problem:

- Teams mit AI Agents erzeugen viel mehr Code, Features, Optionen, die Entscheidungen erfordern
- Traditionell müssen viele Entscheidungen eskaliert werden
 - Architecture Board
 - fachliche Abnahme
 - Security Review
- Wartezeit in Queues erhöht sich drastisch

Dezentrale Entscheidungen

Die Lösung: Entscheidungen lokalisieren

Ökonomische Frameworks (Reinertsen)

- Teams verstehen Impact auf Lifecycle Profit
- Wägen ab: Cost of Delay, Cost of Defects, Development und Operating Costs

Architecture Advice Process (Harmel-Law):

- Team holt Rat (nicht Genehmigung) von Stakeholdern und Expert:innen
- entscheidet selbst, dokumentiert ADR

Dezentrale Entscheidungen

Voraussetzungen

Stream-aligned Teams brauchen für echte Autonomie:

- Ownership über Value Stream
- Fachliche, technische und ökonomische Kompetenz
- CD-Pipeline, Feature Flags, Monitoring

→ Können eigenständig entscheiden *und* umsetzen

Ohne Autonomie: AI generiert schnell → Team muss eskalieren → Bottleneck

Mit Autonomie: AI generiert schnell → Team entscheidet schnell → Flow

Psychologische Sicherheit

Teams eskalieren wegen

- "unsichtbarer Elektroäune" (Reinertsen)
 - Grenzen durch Trial-and-Error ausloten
 - Eskalation als Default aus Unsicherheit
- Angst vor Verantwortung und Bestrafung

Die Lösung:

- Grenzen explizit machen
- Fehler als Chancen zu Lernen

Fehlt beides, führt AI-induzierte Geschwindigkeit zu Eskalations-Lawine



HÆTTA
DANGER-ACHTUNG
FARE-PERICOLO

Was müsst IHR optimieren?

Das toxische Narrativ:

- "Als Top-Entwickler:in von heute bist du morgen nur noch Durchschnitt!"
- "Codest du noch oder spezifizierst du schon?"
- "Wie, du hast nur drei Agenten gleichzeitig laufen?"

Die Realität: Unterschiedliche ökonomische Kontexte

- Startup: Cost of Delay dominiert
- Medizingeräte: Cost of Defects dominiert, Geschwindigkeit nachrangig

Was ist *eure* dominierende Kostengröße?

Die Schlüsselfrage

- Habt ihr berechnet, ob AI Agents *euren* Lifecycle Profit verbessern, basierend auf *eurer* Kostenstruktur?
- Was ist die Cost of Delay in eurem Geschäftskontext?
- Was ist die Cost of Defects?
- Wie lange muss euer Code leben und wartbar bleiben?

AI anders nutzen

- Sokratisch, nicht produktiv:
 - Rubber Duck / Sparring Partner
 - Besseres Denken statt schnellerer Output
- AI-basierte Security-Analysen

Photo by [Felipe Pérez Lamana](#) on [Unsplash](#)



Zwei Wege Agents einzuführen

Agents einführen und hoffen:

- Längere Queues
- Qualitätsverschlechterung
- Keine messbaren Vorteile auf Unternehmensebene

Erst Grundlagen, dann Agents:

- Stream-aligned Teams
- Schnelle Feedback-Schleifen und kleine Batches
- Psychologische Sicherheit
- Dezentrale Entscheidungen

AI als Verstärker

Agents einführen und hoffen:

- noch mehr Chaos
- noch langsamer
- noch höhere kognitive Last
- noch mehr Bugs

Erst Grundlagen, dann Agents:

- schneller
- höhere Qualität

Erst agil, dann agentisch

Baut die Fundamente, die Teams befähigen, autonom

- wahrzunehmen
- zu entscheiden
- zu handeln

Dann verstärken AI Agents diese Fähigkeit.

Der Weg ist das Ziel

- Die Transformation ist der eigentliche Wert
- AI ist der Weckruf
- Ohne Fundamente macht AI alles schlimmer.
- Habt ihr den Weg beschritten, ist AI eine Option.

Photo by [Jonas Degener](#) on [Unsplash](#)



Referenzen

- Faros Report 2025 / The AI Productivity Paradox: <https://www.faros.ai/ai-productivity-paradox>
- 2025 DORA State of AI-assisted Software Development Report: <https://cloud.google.com/resources/content/2025-dora-ai-assisted-software-development-report>
- DX AI-assisted engineering Q4 impact report: <https://getdx.com/report/ai-assisted-engineering-q4-impact-report/>
- Reinertsen, D.G. (1997). Managing the Design Factory: A Product Developer's Toolkit. Free Press
- Reinertsen, D.G. (2009). The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas Pub

Danke! Fragen?



Daniel Westheide

daniel.westheide@innoq.com

<https://www.linkedin.com/in/danielwestheide/>

<https://mastodon.social/@kaffeecoder>

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 333660

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Hermannstr. 13
20095 Hamburg

Erftstr. 15-17
50672 Köln

Königstorggraben 11
90402 Nürnberg