WAS ICH AUS 9 PRODUKTIONSPROJEKTEN MIT CLAUDE CODE GELERNT HABE

Karsten Silz 10. Juni 2026

- Coding Agents machen produktiver
- Hat die KI recht?
- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen
- Skills automatisieren Workflows
- Subagents schreiben schneller beseren Code
- KI klaut “Code schreiben”, aber noch nicht “KI-Teamleiter”

Warum zuhören?

- Erfahrung von 9 Produktions-Projekten mit Claude Code
- Beschreibe Prinzipien, nicht kurzfristige Tipps & Tricks
- Senior Editor im Java Team von InfoQ
- Folien, Skills & Subagents: https://atra.software/gas

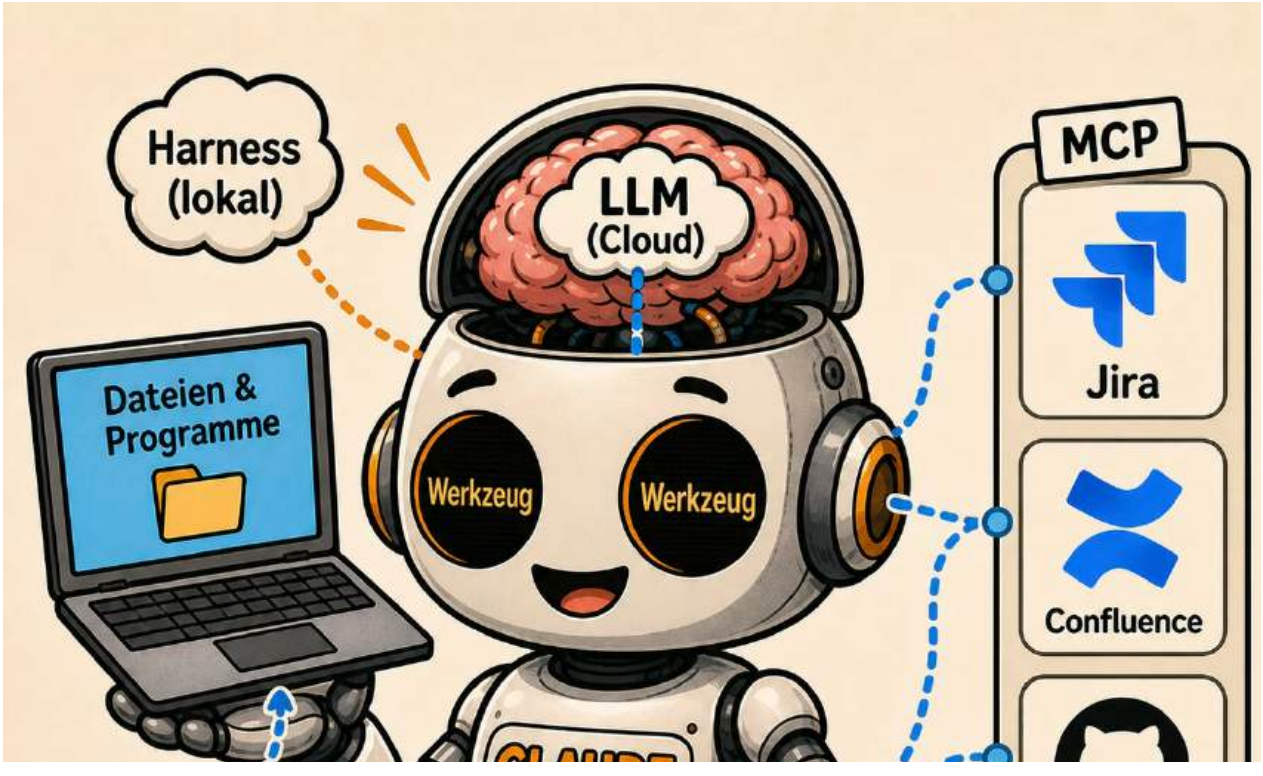
Agenda

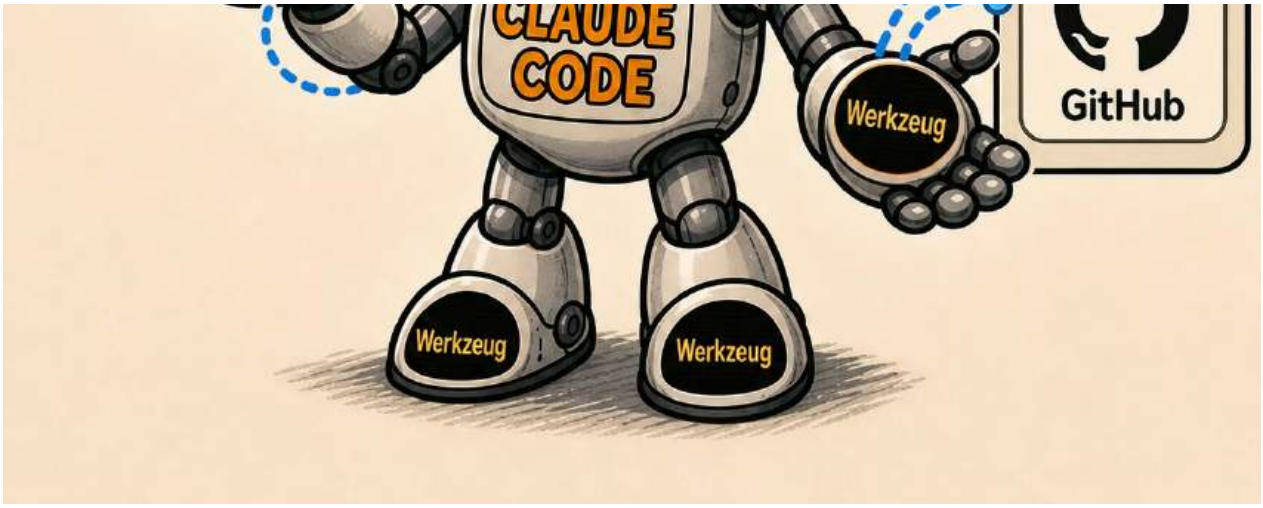
- Was ist ein Coding Agent?
- Meine 9 Projekte
- Produktiver mit Skills & Subagents
- Klaut die KI meinen Job?
- Tipps & Tricks

Was ist ein Coding Agent?

Agent

- KI, die Werkzeuge nutzt
- Agency
  - Agent kommt von “Agency”
  - Bedeutet Handlungsfähigkeit - eigene Handlungen und deren Konsequenzen bewusst steuern und beeinflussen





- Zwei Teile des Coding Agents: LLM & Harness
- Lokale Werkzeuge: Dateien, Programme (vor allem im Terminal), Web-Suche
- Werkzeuge im Netz: Model Context Protocol (MCP)

LLM

- Text nach Wahrscheinlichkeit
  - Autocomplete Engine für Texterzeugung
- Tokens
  - 1 Token = ungefähr 1 Wort
- Context Window
  - Kurzzeitgedächtnis der LLM
  - Alle Werkzeuge, Skills, Agenten, Dateien, Fragen, Antworten, Eingaben, Ausgaben…
- Komprimierung
  - Context Window voll: Komprimierung von 170-200k Tokens auf 10k

    - Wie Buch mit 300 Seiten (“Effi Briest”) auf 10 Seiten zusammenfassen
    - Möglichst vermeiden, weil die meisten Informationen dann verloren gehen
    - Besonders schlimm: Mehrere Context-Komprimierungen hintereinander

Harness

- UI
    - UI: Terminal oder Graphisch
- Tools & Features
    - Lokale Tools: Dateien lesen, schreiben, durchsuchen, Websuche, Programme starten
    - Skills & Subagents sind Features
- Planung
    - Harness entscheidet oft selbständig, wann sie einen Plan präsentiert oder den Nutzer fragt
- CLAUDE.md
    - Datei automatisch immer im Context Window
    - Wichtige Informationen hier ablegen – oder zu wichtigen Dateien verlinken
- AGENT.md
    - CLAUDE.md nur bei Claude, die meisten anderen Harnesse nennen es AGENT.md

Coding Agent

- KI-EntwicklerTeam
    - PO, Analyst, Programmierer, Tester, Admin, …
- Macht Fehler!

Hersteller	Anthropic	OpenAI	Google	Microsoft
–	–	–	–	–
LLM	Haiku	GPT Spark	Gemini Flash	MAI-Code-1-Flash
	Sonnet	GPT	Gemini	MAI-Thinking-1
	Opus	GPT Pro	Gemini Pro	
	Fable		Antigravity CLI	GitHub Copilot CLI
Harness	Claude Code	Codex CLI	Antigravity	GitHub Copilot App
	Claude Desktop	Codex		

- Claude Code, Codex CLI, Antigravity CLI, GitHub Copilot CLI: Terminal-Tools
- Claude Desktop: Graphisches Tool mit Cowork - “Claude Code für Nicht-Entwickler” - und eingeschränktem Claude Code
- Codex, Antigravity, GitHub Copilot App: Graphisches Tool
- Anthropic Fable: https://www.anthropic.com/news/claude-fable-5-mythos-5
- Migration von Gemini CLI zu Antigravity CLI: https:// /an-important-update-transitioning-gemini-cli-to- antigravity-cli/
- Neue Microsoft-LLMs: https://microsoft.ai/news/building-a-hillclimbing-machine-launching-seven-new- mai-models/
- GitHub Copilot Updates: https://github.blog/news-insights/product-

news/github-copilot-app-the-agent- native-desktop-experience/

- Claude, Codex & Antigravity: Gut genug
  - Benchmark: SWE-bench - https://www.swebench.com/
  - Ergebnisse aktueller LLMs sind dort nicht mehr - werden nur noch vom Hersteller veröffentlicht
  - Aktuelle LLMs sind alle über 75%

Hat die KI recht?

- Lösung bekannt: 👍
  - Meist bei Features, manchmal bei Bugs
  - Dann wissen wir, ob die KI recht hat
- Lösung unbekannt: 👎
  - Meist bei Bugs oder Migrationen, manchmal bei Features
  - Dann wissen wir nicht, ob die KI recht hat
- “AI makes people in the know more powerful”
  - Aus einem Interview mit Benedict Evans - https://www.ben-evans.com
  - Nur wenn ich den Code lesen kann, weiß ich, ob er richtig ist

Wie behandeln?

- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen
- Team
  - Ich bin KI-Teamleiter, nicht Programmier
  - Als Chef bin ich schuld, wenn’s schiefgeht

    - Meine Produktivität hängt davon ab, wie viele Agenten ich gleichzeitig wie gut steuern kann
    - Der Erfinder von Claude Code hat immer 5 Claude-Code-Instanzen parallel laufen

- Junior: Onboarding
    - Junior-Entwickler brauchen Onboarding
    - Werkzeuge & Kontext
        - Zugriff auf meine Werkzeuge (Tickets, Wiki, Repos, …)
        - Domäne: Was ist das Fachgebiet?
        - Spezifikation: Was macht Anwendung wie?
        - Wie bauen, testen, deployen und überwachen?
        - Was immer tun - und was nie? Pläne prüfen
    - Pläne prüfen
        - Plan für kleinere Aufgaben
        - Spezifikation + Plan für größere Aufgaben “Erfolg” lehren
    - “Erfolg” lehren
        - Wie testen – und was?
        - Wie Produktion überwachen?

- Nicht-deterministisch
    - Software ist deterministisch: Gleiche Eingaben = gleiche Ausgaben
    - LLMs sind nicht-deterministisch: Gleiche Eingabe = manchmal andere Ausgabe

- Vergesslich
    - Coding Agents lernen nicht von ihrer Arbeit und erinnern sich nicht - sehen Code beim Start zum ersten Mal
    - Claude Code Subagents erinnern sich manchmal - MEMORY.md und *-pattern.md

- schreiben lassen
    - Wer soll all die Spezifikationen, Pläne und Dokumente schreiben?
    - Coding Agent schreibt, Mensch prüft
- testen lassen
    - Coding Agents führen die Back-End- und Front-End-Tests
    - Damit wissen sie, wann sie fertig sind
    - Front-End-Tests gern mit Playwright MCP - Browser-Steuerung (auch headless)

ZusammenFassung

- LLM & Harness, die Werkzeuge nutzt
- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen

Meine 9 Projekte

- Cloud Microservices
    - Produktions-Anwendung für sehr großen DAX-Konzern
- Cat-Sitter SaaS
    - Für Teams, die Katzen in Kundenwohnungen betreuen
    - 130 Anwender auf nativer, mobiler iOS- & Android-Anwendung im UK App Store
    - 30 Anwender auf Web-Anwendung für Manager
- Marketing-Seiten

    - Ferienwohnung-Seite & Nachhilfe-Seite mit Buchungsfunktion

    - Eine Freelancer-Seite mit Vortrags-Seite in GmbH-Seite kombiniert - und Links funktionierten noch

- Buchhaltungs-Helfer
    - Extrahiert Rechnungsdaten aus PDF mit lokaler LLM
    - Kombiniert Rechnungen mit Kontoauszügen
    - Lädt Rechnungen in Buchhaltungsprogramm
- Webseiten-Analyse
    - Findet Vortrags-Seiten, die keine Folien haben
    - Extrahiert Sprecher-Daten mit lokaler LLM
    - Baut E-Mail für Sprecher
- Java, Kotlin, Spring Boot, Postgres, LLMs, Angular, Thymeleaf, Flutter, Hugo, Astro, Netlify, Cloudflare
    - Angular & Thymeleaf: Front-End Frameworks
    - Flutter: Framework, um zwei native, mobile Anwendungen aus einem Quell-Version zu bauen
    - Hugo & Astro: statische Webseiten-Generatoren
    - Netlify & Cloudflare: Content Delivery Networks (CDN)
- Bugs & Features
- Troubleshooting
- Git => Cloudflare
- Spring Boot 2 => 3 => 4
- Angular 13 … 20
- 👍
    - Nur eine App war Overkill: bloß weil man es machen kann, sollte man es nicht unbedingt machen

Produktiver mit Skills & Subagents

Produktivität

- Bestehende Prozesse automatisieren = Mehr Effizienz
  - Bestehende Prozesse optimieren
  - “Die Dinge richtig machen”
  - Einfacher
- Bugs: KI schreibt oft Tickets & Code
  - Manuelle Prozesse ersetzen => schneller
- Neue Prozesse automatisieren = Mehr Effektivität
  - Prozesse verändern
  - “Die richtigen Dinge machen”
  - Schwerer
- Bugs: KI schreibt alle Tickets & ganzen Code…
- …und findet durch mehr Datenquellen mehr Bugs!
  - Automatische Alarme
  - Datenbanken-Logs: vermelden langsame Abfragen & ineffiziente Strukturen
  - Garbage Collection Logs: Protokolliert Speicherverbrauch von unseren Back-End-Services
  - JFR: Java Flight Recorder - die “24-Stunden Black Box” der JVM
- Coding Agents: Analyse 👍
  - Kann Zusammenhänge über alle Logs & Alarme hinweg herstellen

Custom Skills

- Automatisierte Workflows
- /plan-and-do
  - Vom Ticket/freier Aufgabe zum PR
  - Teil der Harness, nicht der LLM
- Branch => (PRD =>) Plan => Code => Tests => Review => PR
  - PRD: Product Requirements Document
- Prompts + Skripts/ Programme
  - Prompts: nicht-deterministisch
  - Skripts/Programme: deterministisch
- Global: in ~/.claude/skills
- Projekt: in .claude/skills
- /review
  - Reviewet Pull Requests oder lokalen Code
  - Stand-alone und als Teil von /plan-and-do
- CLAUDE.md
  - Zum Beispiel: Test-Kommando und Liste der Subagents

Skill Engineering

- /plan-and-do Write a skill that does…
  - Nutzt Subagenten, hat Reviews, macht Git Branch & PR
- skill-writer
- skill-reviewer
  - /plan-and-do nutzt automatisch Subagents

    - Eigene Subagents, wenn sie häufig Skills schreiben: skill-writer & skill-reviewer

- skill-creator
    - Skill von Anthropic: https://claude.com/plugins/skill-creator
- /plan-and-do Change skill so…
    - Ich editiere Skills nicht direkt
- Nichtdeterministisch?
    - LLMs sind so
- 👍
    - Skills sind ein gutes Gerüst für Nondeterminismus
- 👎
    - Nur selten machen sie Mist: /plan-and-do ignoriert dann Prompts und läuft einfach durch

Subagents

- KI-Spezialisten
    - Analyst, Front-End-Entwickler, Back-End-Entwickler, Tester, Admin, ..
    - Teil der Harness nicht der LLM
- Gleiche LLM
    - Eher Personas
    - Konfiguration direkt im Subagent

. . .

–	–
ba-writer	admin
db-coder	ui-designer
be-coder	be-test-coder
fe-coder	fe-test-coder
- Abküzungen

- ba: Business Analyst
- be: Back End
- fe: Front End
- Schneller besseren Code
- 1. Eigenes Context Window
- 1. Laufen parallel
- 1. Regeln anpassbar
- 1. Reviewer Subagents
  2. Mehr Regeln als in CLAUDE.md
  3. Höhere Wahrscheinlichkeit, dass Regeln beachtet werde

–	–
ba-reviewer	
db-reviewer	ui-reviewer
be-reviewer	be-test-reviewer
fe-reviewer	fe-test-reviewer
- Reviewer Subagents haben eigenes Context Window und sehen Code zum ersten Mal
- Nichtdeterministisch: Wählen vielleicht anderen Ansatz als Coder-Subagents
- Im Ergebnis finden sie mehr Fehler als Coding Agent allein

Agent Engineering

- Erstellt in Claude
    - /agents, dann “New Agent”
- Direkt editieren
- Nichtdeterministisch?
    - LLMs sind so
- 👍
    - Reviewer-Agents habe andere, frische Perspektive

ZusammenFassung

- Coding Agents machen produktiver
- Skills automatisieren Workflows
- Subagents schreiben schneller beseren Code

Klaut die KI meinen Job?

Mein Job?

- Business: Business Value
    - Neue Features oder Bug Fixes
    - Egal, ob das Menschen oder KI macht
- Entwickler?

    - Code schreiben
        - 😠 Coding Agents
        - Coding Agent nimmt den Teil des Jobs weg, der Spaß macht
    - Computer Dinge tun lassen
        - ❤ Coding Agents
        - Coding Agents ermöglichen mehr davon

Was können Coding Agents?

- Für mich: Fast alles – mein Entwickler-Team
- Für Sie vielleicht: Baut zu viel Mist
- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen
    - Zuerst mal die Coding Agents richtig behandeln 😀
- Software schreibt Software
    - Die profitabelste KI-Anwendung mit dem größten Markt
    - Für OpenAI & Anthropic die beste Einnahmequelle
    - Google & Microsoft wollen sich nicht abhängen lassen
- Software schreibt Software schreibt Software
    - Coding Agents schreiben sich selbst und arbeiten an LLM
    - Unglaubliches Entwicklungstempo - Claude Code hat täglich einen neuen Release!
    - Einfacher zu skalieren als “Neue Entwickler einstellen”

Trends

Trends

- Meine Annahme
- Coding Agents schreiben 95% des Codes für EnterpriseAnwendungen…
- …aber nicht für kritische Anwendungen…
    - Geräte- und Fahrzeug-Steuerung, Medizintechnik, Finanzalgorithmen, regulierte Bereiche
- …und brauchen aber menschlichen Team-Leiter
    - Coding Agents machen noch zu viele Fehler

- Was, wenn die KI viel weniger Fehler macht?
- Nur 10% der Entwickler gebraucht…
- …aber auch 10x mehr Software gebaut?
    - Dann würde die Zahl der Entwickler gleich bleiben

- Buchhalter & Auditoren, in % US-Beschäftigten
  - Aus https://www.ben-evans.com/presentations, May 2026, Slide 55
  - Angenommene Produktivitatssteigerung von 100x 1910-2000
  - Dann 2000 insgesamt 1400x so viel “Buchhaltung & Audit” wie 1900
- Werden Coding Agents teurer?
  - Börsengelistete Unternehmen müssen profitabel werden - und OpenAI und Anthropic wollen an die Börse
  - Außerdem: Preiserhöhung durch Oligopol
  - Hoffnungsschimmer für “Team Human”?
  - Aber Microsoft will billiger sein (“ein Zehntel”)
  - Lokale Open-Source-Modelle vielleicht auch bald gut genug - dann wären Coding Agents “kostenlos”

- > *0% Wahrscheinlichkeit: Ich habe 2029 keinen Job mehr in der Software-Entwicklung*

ZusammenFassung

- KI klaut “Code schreiben”, aber noch nicht “KI-Teamleiter”

Tipps und Tricks

Wo hängt’s?

- Fertig! Nö
    - Gegenmittel
        - Dem Coding Agent Erfolg beibringen
        - Agent testet selbst Zielerreichung - Closed Loop
- Verhaut kleine Sachen
- Bleibt hängen
- Macht kaputt, was schon mal lief
    - Gegenmittel
        - Pläne vergangener Aufgaben aufheben
        - Spezifikationen der Applikationen pflegen
        - Coding Agent darauf hinweisen

Hat die KI recht?

- Coding Agents/ Subagents reviewen Coding Agents
    - Eine KI prüft die andere

- Mensch prüft
    - Zum Beispiel PR Reviews
- 1 neue Technologie
    - Alte Projektleiter-Regel: 1 Wunder erlaubt pro Projekt
    - Deswegen nur 1 neue Technologie pro Projekt empfohlen - Sie müssen ja prüfen können, ob es stimmt

Umgang

- Wie mit Menschen reden
    - Trainingsmaterial waren unzählige Gespräche
- Offene Fragen
    - Nicht “Ist es nicht so, dass…”, sondern “Was sind Lösungen?”
- Nachfragen
    - “Erklär mir…” und “Warum hast Du…” als Nachfragen

Spec-Driven Development

- Spec-first
    - A well thought-out spec is written first, and then used in the AI-assisted development workflow for the task at hand (Pläne wegwerfen)
- Spec-anchored
    - The spec is kept even after the task is complete, to continue using it for evolution and maintenance of the respective feature (Pläne & Anwendungs-Specs behalten)

(Pläne & Anwendungs-Specs behalten)

- Spec-as-source
    - The spec is the main source file over time, and only the spec is edited by the human, the human never touches the code (klingt wie Model-Driven Architecture - hat nicht funktioniert)
- Specs auf Subagents zuschneiden
    - Idealerweise nur eine Spec-Datei pro Agent

Skill Engineering

- Ziel: 3+ Coding Agents laufen parallel
    - Der Erfinder von Claude Code hat immer 5 Instanzen parallel laufen
- Skills: Infos am Anfang, laufen dann unbeaufsichtigt
    - /plan-and-do fragt viel am Anfang
    - Läuft dann durch bis zur PR-ERstellenung
- claude –dangerouslyskip-permissions
    - Schaltet die meisten Nachfragen ab
    - Klappt bei mir
    - Vielleicht nicht bei Ihnen

Welche Aufgaben für KI?

- Ideal für Coding Agents, weil die meisten Entwickler das nicht mögen
- Bug Fixing

    - Selbst wenn KI scheitert: Kann meist nicht schlimmer werden - sind ja schon Bugs da

- Migrationen
    - Spring Boot 2 - 3 und 2x 3-4
        - OpenRewrite Recipes
        - Spring Boot Migration Guide
    - Angular 13 - 20 (jede einzelne Version)
    - Jeweils Tests & Builds fixen

Wir sind alle Full-StackEntwickler

- Back-End Entwickler bauen Front End: 👍
    - Einfacher, weil Front End leichter zu prüfen: Funktioniert es?
    - Fehler haben geringere Auswirkung
- Front-End Entwickler bauen Back End: 👎
    - Schwerer, weil Back End schwerer zu prüfen: Funktioniert es? Wird es richtig gespeichert?
    - Fehler haben schwere Auswirkungen

Zurückhaltung

- Unendliche Möglichkeiten
    - KI kann fast alles bauen, was wir wollen
- Sinnvoll?
    - Ist es auch sinnvoll?

    - Verwirrt es Kunden?
    - Macht es die Anwendung wackeliger?
- Wartung!
    - Muss auch gewartet und aktualisiert werden
    - KI kann Teile wieder kaputt machen

PO/PM programmiert?

- Meine Erfahrung
    - Arbeit an SaaS mit meiner Frau: Buchungssystem plus Content Management System
- Nicht alles allein…
    - KI macht zu viele Fehler und kann technische Entscheidungen nicht beurteilen
- …aber das meiste
    - Aber die allermeisten Änderungen sind klein genug, dass sie keine Programmierer-Beurteilung brauchen
- Noch nicht in PROD!

Zusammenfassung

- Coding Agents machen produktiver

- Hat die KI recht?
- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen
- Skills automatisieren Workflows
- Subagents schreiben schneller beseren Code
- KI klaut “Code schreiben”, aber noch nicht “KI-Teamleiter”

- Folien, Skills & Subagents: https://atra.software/gas