

WAS ICH AUS 9 PRODUKTIONSPROJEKTEN MIT CLAUDE CODE GELERNT HABE

Karsten Silz 9. Juli 2026

[Folien, Repo mit Skills & Subagents](#)

Warum zuhören?

- Hat die KI recht?
- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen
- Skills automatisieren Workflows und Best Practices
- Subagents schreiben schneller besseren Code
- Gut: effizienter, besser: effektiver

Warum zuhören?

- 9 Projekte
 - Erfahrung von 9 Produktions-Projekten mit Claude Code
- Prinzipien
 - Beschreibe Prinzipien, nicht kurzfristige Tipps & Tricks
- InfoQ
 - Senior Editor im Java Team von InfoQ: <https://www.infoq.com/profile/Karsten-Silz/#allActivity>

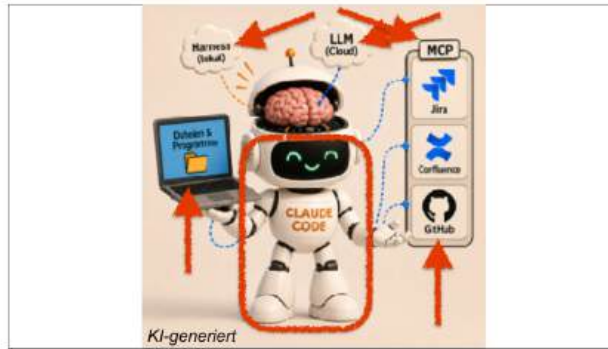
Agenda

- Was ist ein Coding Agent?
- Meine 9 Projekte
- Skills & Subagents
- Gut: effizienter, besser: effektiver
- 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: Zugriff über 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

- AGENT.md
 - CLAUDE.md nur bei Claude Code, andere Harnesse nennen es AGENT.md
 - Datei automatisch immer im Context Window
 - Wichtige Informationen hier ablegen – oder zu wichtigen Dateien verlinken

Coding Agent

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

Anthropic	OpenAI	Google	Microsoft
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
Claude Code	Codex CLI	Antigravity	GitHub Copilot App
Claude Desktop	Codex		

- Claude Code, Codex CLI: Terminal-Tools
- Claude Desktop: Graphisches Tool mit Cowork - "Claude Code für Nicht-Entwickler" - und eingeschränktem Claude Code
- Codex: Graphisches Tool
- Anthropic Fable: <https://www.anthropic.com/news/claude-fable-5-mythos-5>
- Antigravity CLI, GitHub Copilot CLI: Terminal Tools
- Antigravity, GitHub Copilot App: Graphische Tools
- Migration von Gemini CLI zu Antigravity CLI: <https://developers.googleblog.com/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
 - Gut genug für Enterprise-Anwendung
 - Vielleicht nicht für Nischen
 - Embedded Software (Medizin-Geräte, Fahrzeuge) oder Finanz-Algorithmen (Kreditprüfung)
 - 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?

- KI prüft KI
 - Mit gleicher LLM (wie z.B. Subagents)
 - Oder andere LLM (wie z.B. Codex prüft Opus)
- Mensch prüft KI
 - Plan prüfen
 - Code Review
- 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
- Junio
 - 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
 - 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 - siehe `.claude/agent-memory`
- Schreiben lassen
 - Wer soll all die Spezifikationen, Pläne und Dokumente schreiben?
 - Coding Agent schreibt und verbessert später – Mensch prüft
- Testen lassen
 - Coding Agent schreibt Back-End- und Front-End-Tests und verbessert später – Mensch prüft
 - Coding Agent führt alle Tests aus - Front-End-Tests gern mit Playwright MCP - Browser-Steuerung (auch headless)
- KI nervt?
- Junior-Test!
 - Hätte ein JuniorEntwickler das geschafft?
 - Am ersten Tag, allein, nur mit diesen Unterlagen?
 - Nein = Ihr Fehler, nicht KI!

Zusammenfassung

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

Meine 9 Projekte

- Cloud Microservices
 - Cloud Microservices
 - Produktions-Anwendung für DAX-Konzern
- Cat-Sitter SaaS
 - 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
 - Marketing-Seiten
 - Ferienwohnung-Seite & Nachhilfe-Seite mit Buchungsfunktion
 - Eine Freelancer-Seite mit Vortrags-Seite in GmbH-Seite kombiniert - und Links funktionierten noch
- Buchhaltungs-Helfer
 - Buchhaltungs-Helfer
 - Extrahiert Rechnungsdaten aus PDF mit lokaler LLM
 - Kombiniert Rechnungen mit Kontoauszügen
 - Lädt Rechnungen in Buchhaltungsprogramm
- Webseiten-Analyse

- 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 einer Quell-Version zu bauen
 - Hugo & Astro: statische Webseiten-Generatoren
 - Netlify & Cloudflare: Content Delivery Networks (CDN)
- Bugs & Features
- Troubleshooting
- Spring Boot 2 => 3 => 4
- Angular 13 ... 20
- 👍
 - Nur Webseiten-Analyse war Overkill: bloss weil man es machen kann, sollte man es nicht unbedingt machen

Skills & Subagents

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
- PRD, Plan, wie weit, Dateien behalten, Review
 - Checkpoints, die manuelle Zustimmung erfordern
- Kleine Skills: "Baue Entity" oder "Baue Formular"
 - Ich benutze diese bisher nicht
- Prompts + Skripts/ Programme
 - Prompts: nicht-deterministisch
 - Skripts/Programme: deterministisch
- Global oder Projekt
 - Global: in ~/.claude/skills
 - Projekt: in .claude/skills
 - Siehe Dokumentation <https://github.com/atra-consulting/coding-with-ai-lab/blob/solution-jfs-2026/docs/SKILLS.md>
- /do-semiautomatically
 - Promptet plan-and-do
 - Überschreibt Checkpoints, aktualisiert Ticket und stellt Rückfrage über Tickets
- CLAUDE.md

- Zum Beispiel: Test-Kommando und Liste der Subagents
- Expliziter Aufruf
 - Zum Beispiel über /plan-and-do
- Claude Code nimmt sie auch automatisch
 - Wenn er anhand der Skill-Beschreibung denkt, dass es passt

Skill Engineering

- /plan-and-do Write a skill that does...
 - Warum /plan-and-do?
 - Nutzt Subagenten, hat Reviews, macht Git Branch & PR
- Subagenten
 - /plan-and-do nutzt automatisch Subagents: skill-writer und skill-reviewer
 - Diese Subagents helfen beim Skills schreiben: skill-writer & skill-reviewer
- skill-creator
 - Außerdem empfohlen: 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 z.B. 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

- ba: Business Analyst
- be: Back End

- fe: Front End
- Schneller besseren Code
 - Eigenes Context Window
 - Laufen parallel
 - Regeln anpassbar
 - Mehr Regeln als in CLAUDE.md
 - Höhere Wahrscheinlichkeit, dass Regeln beachtet werde
 - Subagents lernen
 - Siehe .claude/agent-memory
 - Reviewer Subagents

ba-writer	admin	ba-reviewer	
db-coder	ui-designer	db-reviewer	ui-reviewer
be-coder	be-test-coder	be-reviewer	be-test-reviewer
fe-coder	be-test-coder	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
- Expliziter Aufruf in /plan-and-do
 - CLAUDE.md hat "Agents"-Sektion mit Liste der Subagents
- Claude Code nimmt sie auch automatisch
 - Wenn er anhand der Subagent-Beschreibung denkt, dass es passt

Subagent Engineering

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

Warum?

- Recht universal
 - Subagents passen auf die meisten Projekte
- Skills global
 - Skills bilden weitverbreitete Workflows ab
- Subagents anpassen

- Subagents können gut angepasst werden

Zusammenfassung

- Skills automatisieren Workflows und Best Practices
- Subagents schreiben schneller besseren Code

Gut: effizienter, besser: effektiver

Produktivität

- Coding Agents kosten Geld
- Höhere Produktivität bezahlt
- Wie produktiver?
- Effizienter
 - Bestehende Prozesse optimieren & automatisieren
 - "Die Dinge richtig machen"
 - Einfach
 - Nur Schritte bestehender Prozesse automatisieren
 - Produktivitätszuwachs: beschränkt
 - Siehe Demo
 - Mehrere Tickets gleichzeitig
 - Pro Ticket weniger zu tun, deswegen mehrere
 - Task Switching + viele Entscheidungen = Stress
 - Die meisten Menschen sind nicht gut im Multitasking
 - Entscheidungen: Entwickler schreiben nicht mehr Code, nur noch Review
- Effektiver
 - Neue Prozesse erfinden, dann automatisieren
 - "Die richtigen Dinge machen"
 - Schwer
 - Erst neuen Prozess bestimmen
 - Dann Team von Sinnhaftigkeit des Prozesses überzeugen
 - Produktivitätszuwachs: optimal
 - Kann ideal auf KI zugeschnitten werden
 - Coding Agent kann in Cloud laufen
 - Zum Beispiel über GitHub Actions
 - Siehe <https://github.com/atra-consulting/coding-with-ai-lab/blob/solution-jfs-2026/.github/workflows/d-o-semi-automatic.yml>
 - Achtung: Anthropic Max Plan oft nicht zugelassen hier – und API Billing sehr teuer für Privatleute
 - Kommunikation über Tickets
 - Coding Agent hinterlässt Kommentare und liest sie auch
 - KI "fehlerfrei" bei Ausführung oder Planung – wie?

Hat die KI recht?

- Code-Qualität erhöhen
- Mehr Tests
- Feedback Loops
- Mehr Daten

1. Code-Qualität erhöhen

- Subagents
- Anwendungs- & Domänen-Doku
- Skills
 - write-ticket
 - plan-and-do
 - review
 - update-claude-files
 - Deckt grundlegende Tätigkeiten ab
- Spec-driven Development
 - Pläne der KI prüfen (wie plan-and-do)
 - Verschiedene Methoden
 - BMad - Traceability: <https://www.bmadcode.com/>
 - GitHub Spec Kit: <https://github.github.com/spec-kit/>
 -
 - Kategorisierung nach <https://martinfowler.com/articles/exploring-gen-ai/sdd-3-tools.html>
 - 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)
 - 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

2. Mehr Tests

- KI-erzeugte, automatische Tests
- Mit /review checken

3. Feedback Loops

- KI testet Ergebnisse selbst
- Kennt "Erfolg"
 - Siehe Junior-Onboarding
- Anwendungs- & Domänen-Doku und Subagenten verbessern
 - Während Arbeit an Aufgabe/Ticket: /update-claude-files - plan-and-do ruft das automatisch auf

- Periodisch: Coding Agent Bugs, Pull-Request-Kommentare, Support-Chat & Tickets, ... auswerten lassen
- Y-Combinator: "Queryable Company"
 - Y-Combinator: Startup-Factory von Microsoft-Mitgründer Paul Allen
 - Siehe <https://www.youtube.com/watch?v=EN7frwQlbKc>
 - Make org legible to AI
 - Alles aufnehmen (Meetings & Gespräche)
 - Every action creates an artifact than can selfimprove
 - Agenten nutzen, die an der Kommunikation teilhaben
 - One-shot internal dashboards for every company ops
 - Für alle Kategorien - Umsatz, Kosten, Engineering, Mitarbeite
 - Beispiel: Engineering Agent
 - Tickets, Slack/Teams Channels, KundenFeedback, High-LevelPläne, Vertriebs-Anrufe, Transkripte Dailies
 - Darauf – und noch mehr – hat der Agent Zugriff

4. Mehr Daten

- KI sehr gut in Datenanalyse
 - Alle Daten "in einen Topf werfen" und KI auswerten lassen
- App-Fehler, App-Log, Java Garbage Collection, Java Flight Recorder, SystemAlarmer, Container Runtime, DB-Logs, Slow Queries, ...
 - Alles "in einen Topf werfen" und von KI analysieren lassen
 - App Fehler bei mir: <https://sentry.io>
 - System-Alarmer bei mir: <https://newrelic.com/>
- Schneller FehlerUrsache finden
 - Weil KI Korrelationen entdecken kann
- Mehr Fehler beheben
 - Weil es schneller geht

Klaut die KI meinen Job?

Mein Job?

- Business: Business Value
 - Neue Features oder Bug Fixes
 - Egal, ob das Menschen oder KI macht
- Warum Entwickler?
 - Siehe <https://blog.lmorchard.com/2026/03/11/grief-and-the-ai-split/>
- 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 😊
 - Junior-Test!
 - Hätte ein Junior-Entwickler das geschafft?
 - Am ersten Tag, allein, nur mit diesen Unterlagen?
- 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" - wenn man GPUs kriegt

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 viel Fehler
- Zum Beispiel 2029: KI macht meist alles allein
 - Nur noch "Entwickler auf Abruf", wie Monteure in vollautomatisierter Fabrik
- 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 Produktivitätssteigerung von 100x 1910-2000
 - Dann im Jahre 2000 insgesamt 1400x so viel "Buchhaltung & Audit" wie 1900
- Werden Coding Agents teurer?
 - Börsengelistede 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"
- Wahrscheinlichkeit größer als 0%: Ich habe 2029 keinen Job mehr in der Software-Entwicklung

Zusammenfassung

- KI klagt "Code schreiben", aber noch nicht "KI-Teamleiter"

Tipps und Tricks

Wo hängt's?

- Fertig! Nö
- Verhaut kleine Sachen
 - Gegenmittel
 - Dem Coding Agent Erfolg beibringen
 - Agent testet selbst Zielerreichung - Closed Loop
- 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?

- 1 neue Technologie
 - Alte Projektleiter-Regel: 1 Wunder erlaubt pro Projekt
 - Analog dazu: nur 1 neue Technologie pro Projekt
 - 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

Welche Aufgaben für KI?

- Bug Fixing & Migrationen
- 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...
- ...aber das meiste
- Noch nicht in PROD!
 - KI macht zu viele Fehler und kann technische Entscheidungen nicht beurteilen
 - Aber die allermeisten Änderungen sind klein genug, dass sie keine Programmierer-Beurteilung brauchen
 - Achtung: In PROD wird alles vermutlich schwerer

Zusammenfassung

- Hat die KI recht?
- Nicht-deterministisches, vergessliches Junior-Team – schreiben & testen lassen
- Skills automatisieren Workflows und Best Practices
- Subagents schreiben schneller besseren Code
- Gut: effizienter, besser: effektiver

[Folien, Repo mit Skills & Subagents](#)