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

Warum zuhören?

- 9 Projekte
- Prinzipien
- InfoQ
  - Erfahrung von 9 Produktions-Projekten mit Claude Code
  - Beschreibe Prinzipien, nicht kurzfristige Tipps & Tricks
  - Senior Editor im Java Team von InfoQ

Demo

Agenda

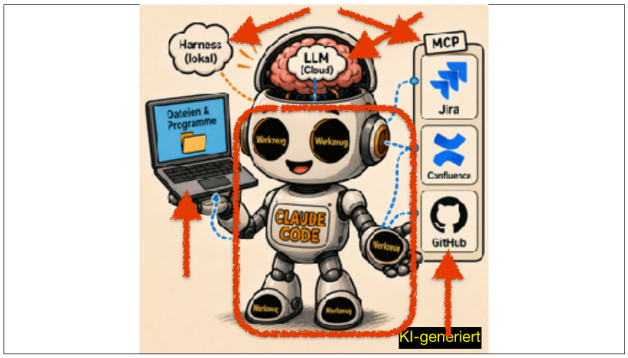
- Was ist ein Coding Agent?
- Meine 9 Projekte
- Produktiver mit Skills & Subagenten
- Klaut die KI meinen Job?
- Tipps & Tricks

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
- Tokens
    - Autocomplete Engine für Texterzeugung
    - 1 Token = ungefähr 1 Wort
- Context Window
- Komprimierung
    - Kurzzeitgedächtnis der LLM
    - Alle Werkzeuge, Skills, Agenten, Dateien, Fragen, Antworten, Eingaben, Ausgaben…
    - 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
- Tools & Features
- Planung
    - UI: Terminal oder Graphisch
    - Lokale Tools
    - Dateien lesen, schreiben, durchsuchen
    - Websuche
    - Programme starten
    - Skills & Subagents sind Features
    - Harness entscheidet oft selbständig, wann sie einen Plan präsentiert oder den Nutzer fragt
- CLAUDE.md
- AGENT.md
    - CLAUDE.md nur bei Claude, die meisten anderen Harnesse nennen es AGENT.md
    - Datei automatisch immer im Context Window
    - Wichtige Informationen hier ablegen – oder zu wichtigen Dateien verlinken

Coding Agent

- KI-EntwicklerTeam
- Macht Fehler
  - PO, Analyst, Programmierer, Tester, Admin, …
- Anthropic
  - Haiku
  - Sonnet
  - Opus
  - Claude Code
  - Claude Desktop
- OpenAI
  - GPT Spark
  - GPT
  - GPT Pro
  - Codex CLI
  - 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
- Google
  - Gemini Flash
  - Gemini
  - Gemini Pro

    - Antigravity CLI
    - Antigravity

- Microsoft

    - MAI-Code-1-Flash
    - MAI-Thinking-1
    - GitHub Copilot CLI
    - GitHub Copilot App

- 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
    - 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

    - ~~Werkzeuge & Kontext~~
    - Pläne prüfen
    - “Erfolg” lehren
    - Werkzeuge & Context
        - 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
        - Wie testen – und was?
        - Wie Produktion überwachen?

- Nicht-deterministisches
- - Software ist deterministisch: Gleiche Eingaben = gleiche Ausgaben
    - LLMs sind nicht-deterministisch: Gleiche Eingabe = manchmal andere Ausgabe
- vergessliches
- - 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 machen die Back-End- und Front-End-Tests

        ~~Front-End-Tests~~ mit Playwright MCP – Browser Steuerung

    - 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, umt 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 & Subagenten

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, wie /plan-and-do
  - Vom Ticket/freier Aufgabe zum PR
  - Teil der Harness (Claude Code / Claude Enterprise), nicht der LLM Außerdem Code Review & PR DEMO
- Branch => PRD => Plan => Code => Tests => Review => PR
- Prompts + Skripts/ Programme
- Global oder Projekt

    - 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 Subagenten

Subagents

- KI-Spezialisten
- Gleiche LLM
    - Teil der Harness (Claude Code / Claude Enterprise), nicht der LLM
    - Eher Personas
    - Direkt im Agent
- 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

    - Reviewer Subagents
        - ba-reviewer db-reviewer ui-reviewer be-reviewer be-test-reviewer fe-reviewer fe-test-reviewer

ZusammenFassung

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

Klaut die KI meinen Job?

Mein Job?

- Business: Business Value
    - Neue Features oder Bug Fixes
- Entwickler?
    - Code schreiben: 😠 Coding Agents
        - Coding Agent nimmt den Teil des Jobs weg, der Spaß macht
    - Computer Dinge tun lassen: ❤ Coding Agents
        - Coding Agent erlaubt mehr Aufgaben

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
    - Tempo des Fortschritts skaliert mit KI-Ausgaben
    - Einfacher zu skalieren als “Neue Entwickler einstellen”

Trends

- Meine Annahme:
- Coding Agents schreiben gesamten Code für EnterpriseAnwendungen…
- …aber nicht für kritische Anwendungen…
- …und brauchen aber menschlichen Team-Leiter

- Was, wenn die KI viel weniger Fehler macht?

- Nur 10% der Entwickler gebraucht… …aber auch 10x mehr Software gebaut?
- Werden Coding Agents teurer?
    - Börsengelistete Unternehmen müssen profitabel werden - und OpenAI und Anthropic wollen an die Börse
    - Außerdem: Preiserhöhung durch Oligopol
    - Aber Microsoft will billiger sein (“ein Zehntel”)
    - Lokale Open-Source-Modelle vielleicht auch bald gut genug

- > *0% Wahrscheinlichkeit: Ich habe 2029 keinen Job mehr*

ZusammenFassung

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

Tipps und Tricks

Wo hängt’s?

- Verhaut kleine Sachen
- Bleibt hängen
- Macht kaputt, was schon mal lief
    - Gegenmittel
        - Spezifikationen der Applikationen pflegen
        - Pläne vergangener Aufgaben aufheben
        - Coding Agent darauf hinweisen

Hat die KI recht?

- Coding Agents schreiben Tests
- 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.
- Spec-anchored
    - Spec-anchored: The spec is kept even after the task is complete, to continue using it for evolution and maintenance of the respective feature
- Spec-as-source
    - 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.

Welche Aufgaben?

- “Meine Kollegen hassen KI!”
  - Mein Beileid
- Bugs fixen
  - Hoffentlich hassen sie Bug-Fixen mehr
  - 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

Advantage: Back End

- Back-End Devs bauen Front End 👍
  - Einfacher, weil Front End leichter zu prüfen: Funktioniert es?
  - Fehler haben geringere Auswirkung
- Front-End Devs bauen Back End 👎
  - Schwere, weil Back End schwerer zu prüfen: Funktioniert es?
  - Fehler haben schwere Auswirkungen

Zurückhaltung

- Unendliche Möglichkeiten
- Sinnvoll?
- Wartung!
    - KI kann fast alles bauen, was wir wollen
    - Ist es auch sinnvoll?
    - Verwirrt es Kunden?
    - Macht es die Anwendung wackeliger?
    - 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

    - KI macht zu viele Fehler und kann technische Entscheidungen nicht beurteilen
    - Aber die allermeisten Änderungen sind klein genug, dass sie keine Programmierer-Beurteilung brauchen

Zusammenfassung

- Coding Agents machen produktiver
- Hat die KI recht?
- Nicht-deterministisches, vergessliches Junior-Team – schreiben &

testen lassen

- Skills automatisieren Workflows
- Subagents schreiben schneller besseren Code
- KI klaut “Code schreiben”, aber noch nicht “KI-Teamleiter”