WAS ICH AUS 9 PRODUKTIONSPROJEKTEN MIT CLAUDE CODE GELERNT HABE

Karsten Silz 5. November 2026



- Gehört zu Blog-Beitrag https://atra.software/wj2
- Hat die KI recht?
- Universalgenies als neue Kollegen – sehr vergesslich, wankelmütig
    - In
- "SDD ’26" nutzen
- "SDD ’27" prüfen
- 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
- Folien, Repo mit Skills & Subagents
- https://atra.software/wj2

Agenda

- Was ist ein Coding Agent?
- Meine 9 Projekte
- User Harness & SDD
- 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

Struktur



- 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



- 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



- Werkzeuge im Netz: Zugriff über Model Context Protocol (MCP)
- Skills automatisieren Aufgaben
- Subagenten schreiben schneller besseren Code
- Markdown-Dateien dokumentieren Anwendung, Kontext & Werkzeuge



Coding Agents

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, & GitHub Copilot: 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?

- Macht Fehler!

- 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?

- Universalgenies als neue Kollegen – sehr vergesslich, wankelmütig
  - Ich bin KI-Teamleiter, nicht Programmierer
  - 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
  - Coding Agents können alles in unseren Projekten: Programmierer, Project Owners, Analysten, Tester, Admins, …
  - Coding Agents sind bessere Programmierer als wir - nicht micromanagen!

- Universalgenies?



- Universalgenies als neue Kollegen – sehr vergesslich, wankelmütig
  - Coding Agents können alles in unseren Projekten: Programmierer, Project Owners, Analysten, Tester, Admins, …
  - Coding Agents sind bessere Programmierer als wir - nicht micromanagen!
- Onboarding
  - Neue Kollegen brauchen Onboarding
- Werkzeuge & Kontext
  - 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?
- Universalgenies als neue Kollegen – sehr vergesslich, wankelmütig
  - Software ist deterministisch: Gleiche Eingaben = gleiche Ausgaben
  - LLMs sind nicht-deterministisch: Gleiche Eingabe = manchmal andere Ausgabe
  - 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
- KI nervt?
- Neuer-Kollege-Test!
- Hätte ein neuer Kollege das geschafft?
- Am ersten Tag, allein, nur mit diesen Unterlagen?
- Nein?
- Ihr Fehler – nicht KI-Fehler!

Zusammenfassung

- LLM & Harnesse, die Werkzeuge nutzen
- Universalgenies als neue Kollegen – sehr vergesslich, wankelmütig

Meine 9 Projekte

- Cloud Microservices
  - Produktions-Anwendung für 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
- Ferienwohnungs-SaaS
- Marketing-Seiten
  - Ferienwohnungs-Seite & Nachhilfe-Seite mit Buchungsfunktion
  - Eine Freelancer-Seite mit Vortrags-Seite in GmbH-Seite
- Schulungs-Anwendung
  - Schulungs- und Demo-Anwendung Node.js + User Harness
  - Skills
  - Subagents
  - Applikations-Dokumentation als MD
  - Übungsaufgaben
  - War zuerst Java + Angular, jetzt Node.js + Angular
- 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
  - kombiniert - und Links funktionierten noch
- Keynote => PDF
  - Damit wurde dieses Handout generiert
  - PDF-Export mit Sprechernotizen in Keynote
  - PDF => HTML
  - HTML => PDF

- 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
- Spring Boot 2 - 3 - 4 - 4.1
- Angular 13 … 22
- 👍
  - Nur eine App war Overkill: bloß weil man es machen kann, sollte man es nicht unbedingt machen

User Harness & SDD

Skills

- Prompts für LLM + Skripts/Programme
  - Prompts: nicht-deterministisch
  - Skripts/Programme: deterministisch
- Automatisieren Aufgaben
  - Aufgaben können klein sein ("Baue neue DB Entity") oder groß ("Vom Ticket zum PR")
- claude --debug
  - Eigene Skills im Debug-Modus abarbeiten, zweiter Coding Agent findet Fehler in Echtzeit
- Coding Agent Skills testen lassen
  - Geht, habe ich noch nicht probiert
- 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
- /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
- /do-semiautomatically
  - /do-semi-automatically
  - Promptet plan-and-do
  - Überschreibt Checkpoints, aktualisiert Ticket und stellt Rückfrage über Tickets
- CLAUDE.md
  - Zum Beispiel: Test-Kommando und Liste der Subagents

Skill Engineering

- /plan-and-do Write a skill that does…
  - Warum /plan-and-do?
  - Nutzt Subagenten, hat Reviews, macht Git Branch & PR
- skill-writer
- skill-reviewer
- skill-creator
  - /plan-and-do nutzt automatisch Subagents
  - Diese Subagents helfen beim Skills schreiben: skill-writer & skill-reviewer
  - 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
- 1. Eigenes Context Window
- 2. Laufen parallel
- 3. Regeln anpassbar
  - Mehr Regeln als in CLAUDE.md
  - Höhere Wahrscheinlichkeit, dass Regeln beachtet werden
- 4. Subagents lernen
  - Siehe .claude/agent-memory
- 5. 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

Subagent Engineering

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

"SDD ’26" nutzen

- Spec-Driven Development
- KI macht Plan
- Mensch gibt frei
- KI schreibt Code



- Superpowers Spec Kit Get Shit Done OpenSpec BMAD

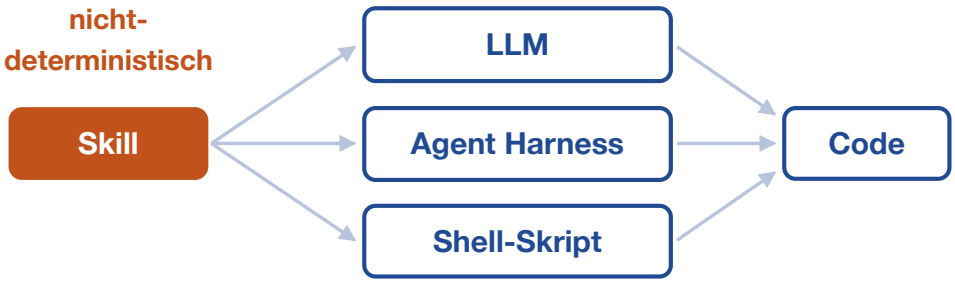


- Nutzen!
- Noch kein SDD-Framework?

- KI fragen!
  - Input
  - Ihre Prozess-Dokumentation
  - Ihre Tickets
  - Ihr Code
  - Ihre PRs
  - Beispiel-Prompt für LLM der Klasse Opus 5.5 oder besser: "Analysiere unseren Software-Entwicklungs-Prozess anhand von… Dann analysiere alle öffentlich verfügbaren SDD-Frameworks und sage mir, welches am besten auf unseren Entwicklungs-Prozess passt. Begründe jede Entscheidung und sage, warum du dich für ein Framework entschieden hast. Gib mir zwei Alternativen und begründe, warum du dich dagegen entschieden hast. Dann sage mir, wie wir das SDD-Framework anpassen und einführen können"
- Probleme?

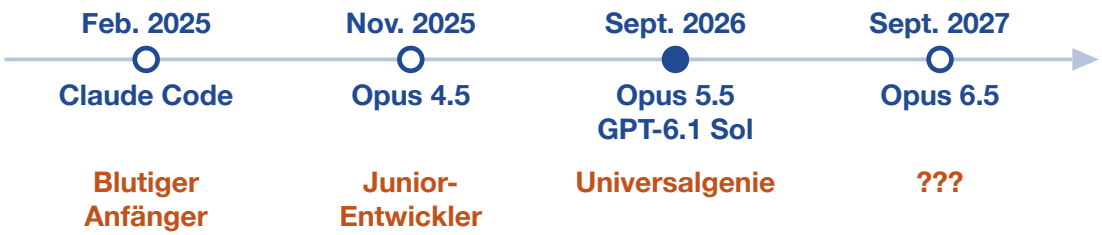


- Ansatz für "KI = Junior-Entwickler"
- Große, detaillierte Skills – nicht deterministisch!



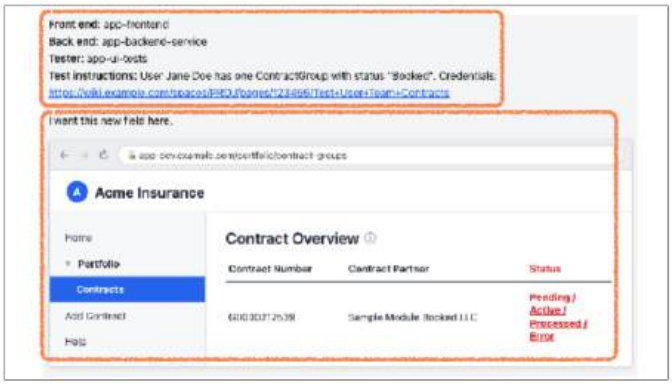
"SDD '27"

- Ansatz für "KI = Universalgenie"
- 1 Jahr: Junior-Entwickler zu Universalgenie



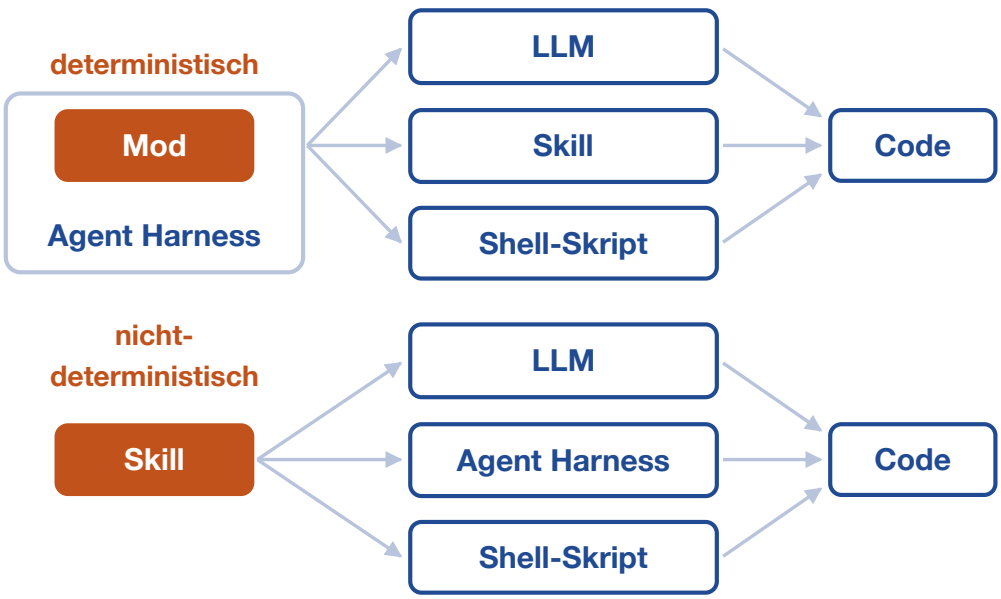
- Drei Projekte
  - Gerade erst gestartet
  - Klappen jetzt vielleicht noch nicht
  - Kann mir vorstellen, dass das 2027 Mainstream ist
- Projekt 1: "KI-Redaktion"
- Noch kein SDD-Framework?
- KI fragen!
  - Input
  - Ihre Prozess-Dokumentation
  - Ihre Tickets
  - Ihr Code
  - Ihre PRs
  - Beispiel-Prompt für LLM der Klasse Opus 5.5 oder besser: "Analysiere unseren Software-Entwicklungs-Prozess anhand von… Dann analysiere alle öffentlich verfügbaren SDD-Frameworks und sage mir, welches am besten auf unseren Entwicklungs-Prozess passt. Begründe jede Entscheidung und sage, warum du dich für ein Framework entschieden hast. Gib mir zwei Alternativen und begründe, warum du dich dagegen entschieden hast. Dann sage mir, wie wir das SDD-Framework anpassen und einführen können"
- Noch kein SDD-Framework?
- KI fragen!
  - KI halluziniert
  - KI kennt nicht ihre Prozesse, Richtlinien, oder allgemeine Regelungen
- Hat die KI recht?
- Zeitungs-Redaktion
- Richtige Artikel
- Recherche Fakt-Prüfung Schreiben Editor Rechtsabteilung
- KI-Redaktion
  - Skills, wie /write-doc und /review-doc
  - Viele Subagents
- author
  - Ein Subagent…
- researcher source-scout coherence-checker realist architecture-fit-reviewer copy-editor tool-evaluator referee process-expert contrarian change-advisor arbiter implementer-reviewer owner-stand-in regulatory-reviewer colleague-stand-in security-reviewer policy-auditor
  - Fakten & Stimmigkeit
- "Findet im Internet & Intranet relevante Infos"
  - So suchen die Subagenten Informationen
- Dokumente
- Fakten
- Annahmen

    - Alles in Git versioniert, verlinkt und geschützt
- Wahrheitsfindung
    - Stimmt dieses Whitepaper? Hat mein Kollege recht?
- Analyse
    - Wie unterscheiden sich die SDD-Frameworks?
    - Regelmäßige Überprüfung: Welche neuen Richtlinien betreffen unsere Abteilung oder unsere Anwendung?
- Handlungsoptionen
    - Welches SDD-Framework passt am besten zu uns?
- Funktioniert
- Dauert lange
- Akzeptanz?
    - Komplexe Aufgaben können stundenlang laufen
    - Anzahl der "KI-Dateien" in Git überfordert Kollegen
    - Tipp: "mostly-for-ai" für "KI-Dateien", "for-humans-and-ai" für Menschen
- Projekt 2: "Der Chef programmiert"
- Input vom PO/PM
- Full-Stack
- Mehrere Coding Agents

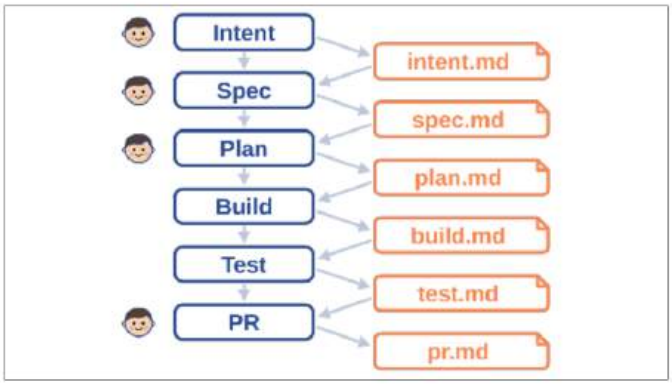


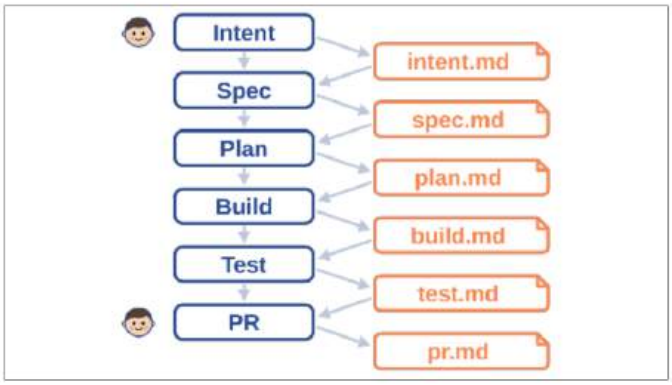
- Skill für Claude-Code-Kommunikation
    - Claude-Code-Instanzen können miteinander reden
- Analyse
    - Erkennt anhand der URL, welcher Front-End-Teil beteiligt ist
    - Daraus dann Back-End-Code identifizieren
- Mehrere Coding Agents
- Front End
- Back End
- Automatischer Test

- Hat bei mir beinahe geklappt
  - Unser Jira MCP Server konnte keine Dateien herunterladen - und ich hatte die Login-Informationen für den Test vergessen
- Funktioniert bei Kollegen
  - Allerdings nicht mit Screenshots
- Projekt 3: "SDD Next"
- Nur in Claude Code
- Hoffentlich bald in anderen Agent Harnesses
- KI = Universalgenie
- Ziel
  - Wie guten Entwickler behandeln
- Erfolg
- Test
- /goal
  - /goal ist Kommando von Claude Code: Ziel setzen, Claude Code verfolgt es dann
- Schlanker Skill
  - Höchstens ein paar hundert Zeilen
- Mehr Determinismus
- Bessere Developer Experience
- Claude Code Mod
  - Mehr Determinismus
  - Bessere Developer Experience
  - https://code.claude.com/docs/en/plugins/mods/overview



- Eigenes SDD-Framework bauen (?)
- Vertrag für mehrere Implementierungen (?)
  - Das ist Software - und Software bauen ist billig





Zusammenfassung

- "SDD ’26" nutzen: Skills & Subagents
- "SDD ’27" evaluieren: KI-Redaktion, Chef programmiert, "SDD Next"

Gut: effizienter, besser: effektiver

Produktivität

- Wie produktiver?
- Effizienter
- Bestehende Prozesse
  - 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
    - Neue Prozesse erfinden, dann automatisieren
    - "Die richtigen Dinge machen"
- Schwer
    - Erst neuen Prozess bestimmen
    - Dann Team von Sinnhaftigkeit des Prozesses überzeugen
- Produktivitätszuwachs: optimal
    - Aber kann ideal auf KI zugeschnitten werden
- 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
- 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/
    - ….
- 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, mitgegründet von Paul Graham
    - Siehe https://www.youtube.com/watch?v=EN7frwQlbKc
- Make org legible to AI
    - Alles aufnehmen (Meetings & Gespräche)
- Every action creates an artifact that can self-improve
    - Agenten nutzen, die an der Kommunikation teilhaben
- One-shot internal dashboards for every company ops
    - Für alle Kategorien - Umsatz, Kosten, Engineering, Mitarbeiter
- Beispiel: Engineering Agent
- Tickets, Slack/Teams Channels, Kunden-Feedback, High-Level-Pläne, 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, System-Alarme, Container Runtime, DB-Logs, Slow Queries, …
  - Alles "in einen Topf werfen" und von KI analysieren lassen
  - App Fehler bei mir: https://sentry.io
  - System-Alarme bei mir: https://newrelic.com/
- Schneller Fehler-Ursache 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 Enterprise-Anwendungen…
- …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



- 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?

    - 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"



- Wahrscheinlichkeit größer als 0%: 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
- 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-Stack-Entwickler

- Back-End Entwickler bauen Front End
- 👎
  - Schwieriger, weil Front End schwerer zu prüfen: Hat es den richtigen Stil?
  - Fehler haben geringere Auswirkung
- Front-End Entwickler bauen Back End
- 👍
  - Einfacher, weil Back End leichter zu prüfen: Sind die richtigen Daten da? Keine Fehler im Log?
  - 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
  - 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: Nach einiger Zeit in PROD wird alles vermutlich schwerer

Zusammenfassung

- Hat die KI recht?
- Universalgenies als neue Kollegen – sehr vergesslich, wankelmütig
  - In
- "SDD ’26" nutzen
- "SDD ’27" prüfen
- Gut: effizienter, besser: effektiver



- Folien, Repo mit Skills & Subagents
- https://atra.software/wj2