

INNOQ TECHNOLOGY DAY 2026

Agenten als neue Stakeholder

Was Coding Agents von unserer Architektur verlangen

INNOQ

Tammo van Lessen
Sven Johann

Wir haben einen neuen Kollegen



schreibt Code auf dem Niveau eines soliden **Seniors**

trifft Designentscheidungen auf dem Niveau eines **Juniors**

kennt erstaunlich viele Sprachen, Technologien und Bibliotheken

aber: **erste Stunde** im Team

will viel zu sehr gefallen

lebt sehr im Hier und Jetzt

kommt nie in die Kaffeeküche

entscheidet ADRs ohne Intuition und ohne die Vorgeschichte zu kennen

wird nie müde

je mehr er liest, redet und schreibt, desto **teurer** wird er

Ein Coding Agent.

Und jede Session ist sein erster Tag.

"... the onboarding materials for an LLM that's having a permanently recurring first day on the team."

Pete Hodgson, Mai 2025

Hodgson 2025 ↗

ROBUST



Wir haben Architekturen jahrzehntelang an die kognitiven Grenzen von Menschen angepasst. Was passiert, wenn der Codeschreibende ganz andere Grenzen hat?

Wir schlagen Pflöcke ein

ROBUST	gut belegt oder breiter Konsens
UMSTRITTEN	Belege oder Fachleute widersprechen sich
OFFEN	plausibel, aber niemand hat es gemessen

Heute kann noch niemand sagen, wie wir in fünf Jahren Software entwickeln werden.

Wir reden über Teams in größeren Organisationen, in denen Menschen und Agenten zusammenarbeiten.

1. Wer ist der neue Kollege?
2. **Was braucht wer?**
3. Wo wohnt das gemeinsame Wissen?
4. Wer steht dafür gerade?
5. Was ist neu?

Teil 2

Was braucht wer?

Wir sortieren in vier Töpfe

👱 🤖

Beide

brauchen Mensch und Agent

🤖

Nur Agent

braucht nur der Agent

⚡

Konflikt

hilft dem einen, stört den anderen

👱

Nur Mensch

braucht nur der Mensch

Was er mitbringt, was wir ihm geben

WISSENSART	WOHER	INHALT	ARCHITEKTURHEBEL
Parametrisch	bringt er mit	Sprachen, Standards, Frameworks, Muster	bekannte Idiome statt Eigenbauten
Arbeitskontext	stellt er sich pro Aufgabe zusammen	Aufgabe, betroffene Komponenten, Code	Kohäsion, Kopplung, Änderungslokalität
Projektwissen	**müssen wir ihm geben**	Domäne, Grenzen, ADRs, Konventionen	auffindbares, präzises Wissen
Verfahrenswissen	**müssen wir ihm geben**	Abläufe, Prüf- und Reparaturschritte	Skills, Guides, Sensors

BEIDE | NUR AGENT | KONFLIKT | NUR MENSCH

Der größte Topf: beide

WISSENSART	WAS	WARUM BEIDE
Parametrisch	Standards und Konventionen	für uns Vertrautheit, für ihn Trainingswissen
Arbeitskontext	Kohäsion, lose Kopplung, Sichten wie die Bausteinsicht Level 1 und 2	entlasten unseren Kopf und halten seinen Kontext klein
Projektwissen	Ubiquitous Language, ADRs als Leitplanke, auffindbares und aktuelles Projektwissen, der Intent	Intent ist *“externalized rationale that developers and AI agents need to work safely with code”*. Dass er fehlt, merkt nur der Mensch.
Verfahren	Guides und Sensors: Typen, Linter, ArchUnit	nutzen wir seit Jahren. Beim Agenten werden sie für *uns* zur Pflicht.

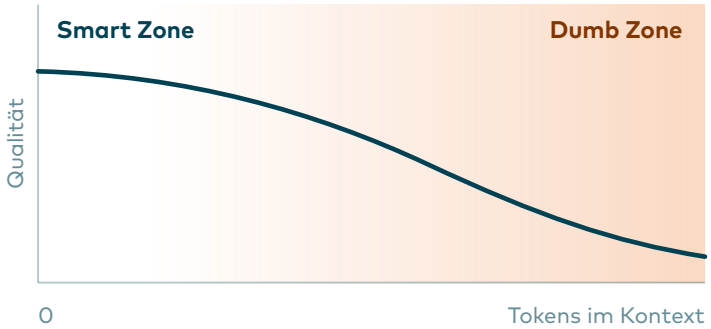
Storey 2026 ↗ Böckeler 2026 ↗

ROBUST

BEIDE | **NUR AGENT** | KONFLIKT | NUR MENSCH

Was wirklich nur der Agent braucht

Kontextbudget



schematisch, keine Messkurve; wo die Grenze liegt, hängt von Modell und Aufgabe ab

Mit jedem Token wird er schleichend schlechter, ohne es zu merken. Es zählen die **absoluten Tokens**, nicht der Füllstand. Jeder Schritt schickt den ganzen Kontext neu, die Kosten wachsen mit.

Jeder Tag ist Tag 1



Leonard in *Memento* hat kein Kurzzeitgedächtnis und lebt von Notizen, Polaroids und Tattoos. Unsere Zettel heißen AGENTS.md, Skills und ADRs.

Horthy 2025 ↗ Weekly Dev's Brew 2026 ↗ Chroma 2025 ↗

Richtung **ROBUST** Schwelle **UMSTRITTEN**

BEIDE | NUR AGENT | **KONFLIKT** | NUR MENSCH

Konflikt: Wie viel Doku?

👨 Wir wollen **Erklärungen**.

🤖 Ihn kosten sie **Tokens**.

DIE STUDIEN WIDERSPRECHEN SICH

Gloaguen u. a. 2026

Kontextdateien bringen keinen signifikanten Erfolgsgewinn, kosten aber im Schnitt über 20 % mehr.

≠

Lulla u. a. 2026

Mit AGENTS.md arbeitet Codex schneller und mit weniger Output-Tokens, bei gleicher Erledigungsquote.

Warum viel Inhalt schaden kann: Die Datei wird immer geladen. Und vieles darin sagen Harness und Systemprompt schon.

Gloaguen u. a. 2026 ↗ Lulla u. a. 2026 ↗

UMSTRITTEN

BEIDE | NUR AGENT | KONFLIKT | **NUR MENSCH**

Was nur wir Menschen brauchen

🧭 Das Warum im ADR

um eine Entscheidung zu akzeptieren und sie sich zu merken. Der Agent braucht es nur, wenn er sie neu bewerten soll.

🎨 Style Guides und Formatierung

soweit kein Formatter sie erzwingt

🧠 Das mentale Modell des Systems

Fehlt es, wächst die **Cognitive Debt**. Wie Technical Debt, nur nicht im Code, sondern in unseren Köpfen: Das Team versteht nicht mehr, was es da betreibt.

⚖️ Verantwortung

Die trägt der Agent nicht. Tragen kann sie nur, wer das mentale Modell hat und den Intent kennt. Fehlt der Intent, wächst die **Intent Debt**.

Storey 2026 ↗

🟢 **ROBUST** | 🟡 Warum **UMSTRITTEN**

Fast alles, was der neue Kollege braucht, hätten wir auch gebraucht.

Neu sind das Kontextbudget und der ewige erste Tag.

Wir merken, wenn uns Wissen fehlt, und fragen nach.

Er fragt nicht. Er füllt die Lücke.

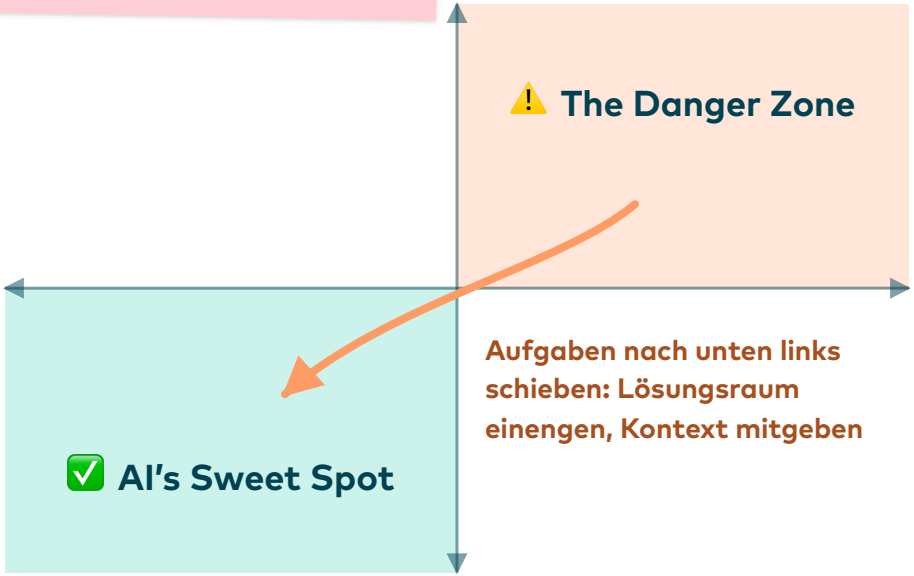
Also muss das Wissen dort liegen, wo er es findet.

ROBUST

Teil 3

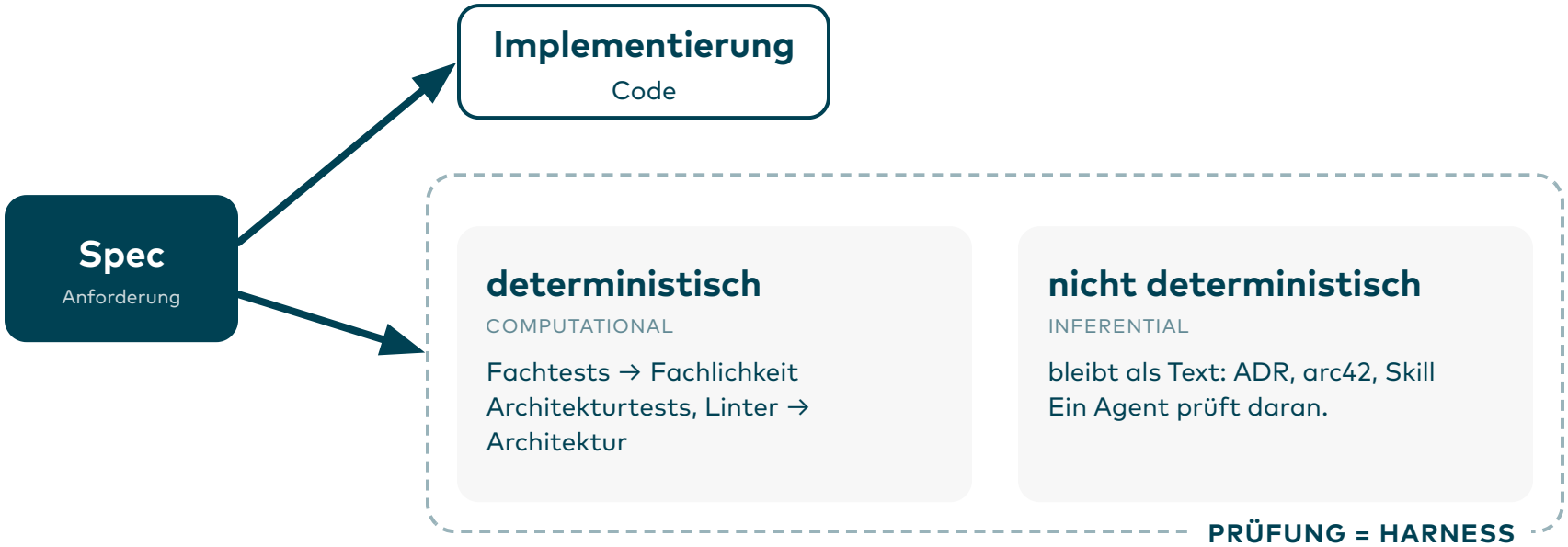
Wo wohnt das gemeinsame Wissen?

Wann der neue Kollege gut ist



nach Pete Hodgson: Constraint-Context Matrix, 2025 ↗

Das Wissen wandert



“If you can tool it, tool it. If you can’t, prompt it.”

aus unserem Training: req42 → arc42 → CLAUDE.md-Zeile → ArchUnit-Test

Böckeler 2026: Guides und Sensors ↗

ROBUST

Die Spec darf sterben

Spec

Werkzeug für die Discovery

… und darf weg

überführen

Fachtests
Abnahmekriterien als ausführbare Tests

Glossar
die Begriffe der Domäne

ADRs
die Entscheidungen und ihr Warum

Code
und der Harness

Böckeler nennt das **spec-first**. Den bewussten Schritt **Überführen** ergänzen wir, nah an *Specification by Example*. Die Gegenposition heißt **spec-anchored**: Die Spec bleibt und wird gepflegt.

Böckeler 2025 ↗ Adzic 2011 ↗

UMSTRITTEN

Das ADR nicht

ADR 0017

Wir machen OIDC und prüfen in jedem Request die Rolle im Token.

Linter-Regel: schwer zu bauen

Agent mit dem ADR in der Hand: **prüft das leicht**

ADR 0002

~~**Wir benutzen Git und Spring Boot.**~~

Steht längst im Code.

Kostet Tokens und sagt ihm nichts Neues.

Die Kunst ist zu erkennen, **wann ein ADR nötig ist**.

ROBUST

Monkey see, monkey do

359/360

Läufe, in denen der Agent dem Muster im Repo folgt statt seiner eigenen Vorliebe. Getestet in Repos, in denen alle Endpunkte die Gegenvariante nutzen.

Zhu u. a., Preprint, 28.10.2026

> *“Codex replicates patterns that already exist in the repository—even uneven or suboptimal ones.”*

Ryan Lopopolo, OpenAI, 11.02.2026

Mit der Zeit wird die Codebasis selbst zum Guide.

Zhu u. a. 2026 ↗ Lopopolo 2026 ↗

ROBUST

Der Agent macht Housekeeping

Index statt Volltext im Dauerkontext

```
# AGENTS.md
## ADRs, Volltext in docs/adr/
0002 Git und Spring Boot
0011 Module nach Fachlichkeit
0017 OIDC, Rolle in jedem Request
…
```

“Like a well-organized manual that starts with a table of contents”

Anthropic über Progressive Disclosure

Der Index gehört **flach in die AGENTS.md**. Hinter einem Skill hat er ihn bei Vercel in 56 % der Fälle nie aufgerufen.

Zhang u. a. 2025 ↗ Vercel 2026 ↗ He u. a. 2026 ↗

Veraltete Doku legt er zur Freigabe vor

🤖 docs: arc42-Konzept Persistenz an den Code angleichen

Das Konzept beschreibt noch JPA, der Code nutzt seit drei Monaten jOOQ.

wartet auf Freigabe

Er zieht die Doku nach, die wir liegen lassen.
Wir geben seine Vorschläge frei.

OFFEN

Skills helfen, wenn Fachleute sie schreiben

+16,6
Prozentpunkte

✅ **kuratiert:** Fachleute schreiben aus Erfahrung, fokussiert

SkillsBench

–8 bis –11,5
Prozentpunkte

✖ **generiert:** der Agent schreibt sie aus eigenem Wissen

SkillsBench

92 von 98
ohne signifikanten Effekt

✖ **gekauft:** öffentliche Marketplace-Skills

SWE-Skills-Bench

Wir entscheiden, wofür: dort, wo er sich verlaufen hat.

Geholfen haben "procedures that the agent would otherwise omit".

✅ Wie schneiden wir bei uns ein Release?

✖ Wie baut man eine Spring-Anwendung?

Li u. a. 2026 (SkillsBench) ↗ Han u. a. 2026 (SWE-Skills-Bench) ↗

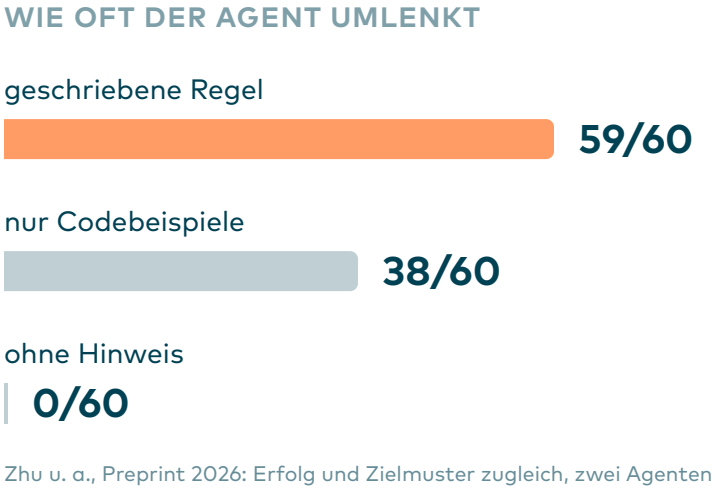
ROBUST

Die Gegenrichtung: der Camp Leader

1. Er kopiert Muster, also auch die schlechten.
2. Der Camp Leader geht regelmäßig über den Platz und findet das Silberpapier.
3. Er schreibt die Regel in den Harness:

```
Business-Logik gehört nie in den UI-Layer.
```

4. Der Agent räumt nach der Scout Rule auf. OpenAI nennt das *garbage collection*.

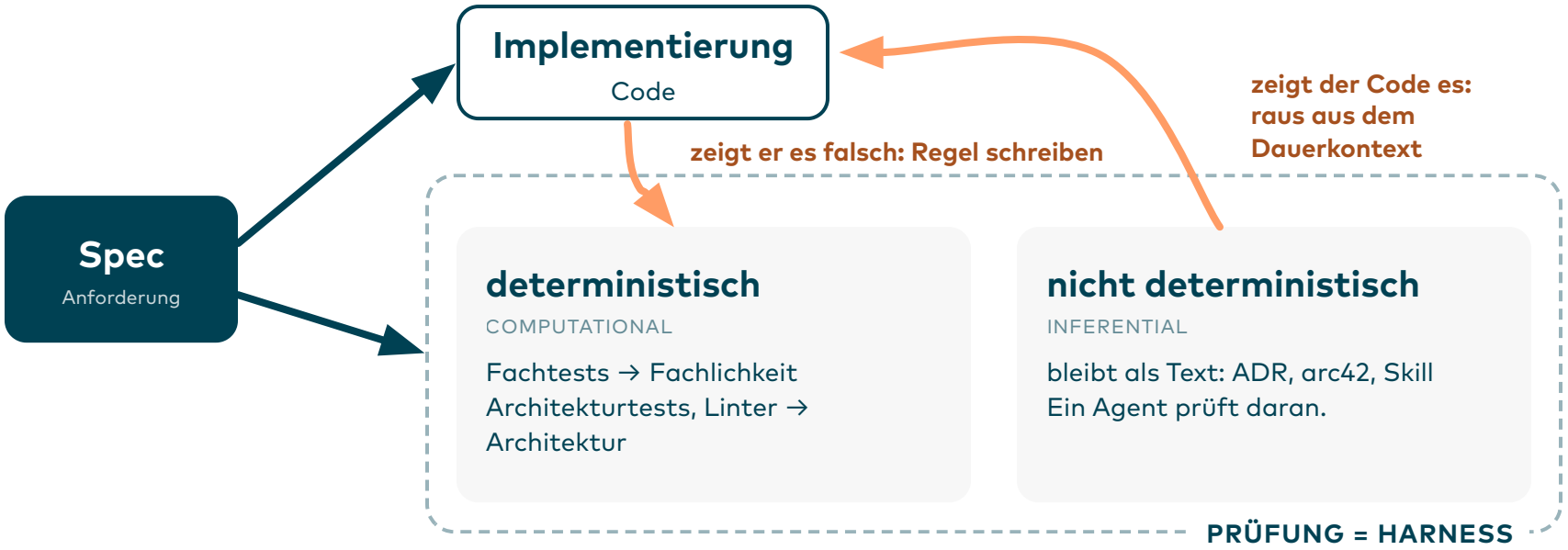


Zhu u. a., Preprint 2026: Erfolg und Zielmuster zugleich, zwei Agenten

Zhu u. a. 2026 ↗ Böckeler 2026 ↗

Prinzip **ROBUST** Wirkung **OFFEN**

In beide Richtungen



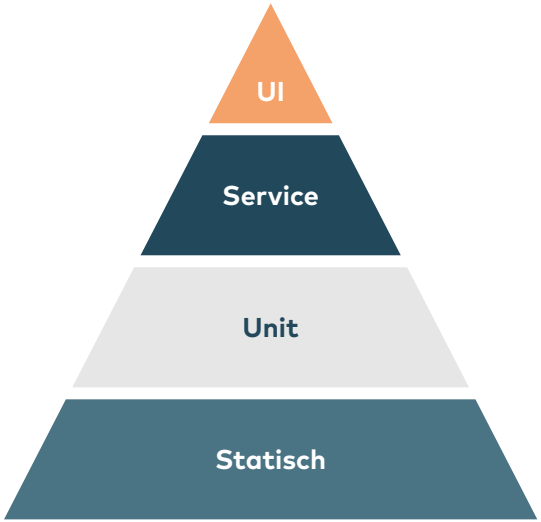
Was der Code zeigt, müssen wir nicht mehr sagen.
Was der Code falsch zeigt, schon.

"Success is silent, failure is loud."

Er beschwert sich nicht. Also braucht es eine Alarmanlage.

Wo die Alarmanlage hängt

welche Tests ↓



wo sie laufen →

Agent-Hook *Sekunden*	**Git-Hook** *Sekunden*	**CI** *Minuten*	**Produktion** *Stunden*
zu langsam	*zu langsam*	● alle Journeys	● Smoke-Test
zu langsam	*zu langsam*	● alle	*zu spät*
● betroffene Tests	● alle	● alle	*zu spät*
● geänderte Dateien, Autofix	● alle	● alle	*zu spät*

← billig, der Agent korrigiert sich selbst

ein Mensch findet es →

Alonso u. a. 2026 ↗ Zhao u. a. 2026 ↗

● Prinzip **ROBUST** ○ beim Agenten **OFFEN**

Teil 4

Wer steht dafür gerade?

Das Leitmotiv

Er beschwert sich nicht.

📉 In der Dumb Zone wird er dümmer, **ohne es zu merken**.

🕳️ Fehlt der Intent, **füllt er die Lücke**.

⚖️ Bei einem Trade-off **entscheidet er einfach**.

🔧 „Hier ist nichts zu tun" sagt er selten. Bei Bug-Reports, die im Code längst behoben sind, schlägt er in **35 bis 65 %** der Fälle trotzdem Änderungen vor.

Wer merkt es?

Gloaguen u. a. 2026 ↗

Die Landkarte, nicht jede Zeile

> **Licht aus, Juli 2025:** Niemand liest mehr den Code. Drei Monate später ist die Seite down, und keiner kennt den Code, den er reparieren muss. Dex Horthy, HumanLayer

Programmieren ist Theoriebildung.
Peter Naur, 1985

Geht die Theorie verloren, wächst die Cognitive Debt. Dem Slop fehlt das mentale Modell.

Wer dafür geradesteht, muss im Zweifel **jede Änderung verstehen können**. Wie eine Geschäftsführerin, die ihre Zahlen kennt.

Die Landkarte setzt **konzeptionelle Integrität** voraus. Ohne sie gibt es keine Karte, nur Code, der irgendwo herumschwimmt.



Code im Sweet Spot braucht kaum Review, Code in der Danger Zone ein enges.

Ein Review-Agent kann melden, wenn eine Änderung eine **heikle Stelle** berührt.

Naur 1985 ↗ Guhr, Seitz 2026 ↗ Horthy 2026 ↗

Naur **ROBUST** Reviewtiefe **UMSTRITTEN**

Das Agentic Trio

🧭 **Produkt** 🎨 **UX** 🛠️ **Engineering** bauen gemeinsam mit 🤖 **Agent**

VORHER · FEEDFORWARD

Der Intent sitzt mit am Tisch.

Anforderung, Abnahme und Umsetzung ohne Übergabe.

HINTERHER · FEEDBACK

Die Fachseite prüft die Fachlichkeit, die Entwicklerin die Technik.

Ihre Rolle: **From Producer to Steward**. Sie hält das mentale Modell des Systems.

“Nobody else in the trio can take that on.”

Weil am Tisch alle über alle Belange entscheiden können, wird der Engpass kleiner.

Westheide, Walther 2026 ↗

OFFEN

Wer führt, wer prüft?

	FEEDFORWARD (GUIDES)	FEEDBACK (SENSORS)
Computational	Typen, Templates, Konventionen im Code	Linter, ArchUnit, Fachtests
Inferential	ADRs, Skills, arc42 im Kontext	Review-Agent gegen ADRs und heikle Stellen
Human	Trio: Intent, Entscheidungshilfen	Trio-Review: fachlich und technisch

+441,5 %

Median-Review-Zeit in Phasen hoher KI-Nutzung. Faros 2026, Korrelation, Herstellerangabe.

Je mehr die oberen Zeilen tragen, desto weniger muss die unterste leisten.

30 min

Abstimmung vorher sparen Stunden Review. Horthy 2026, Erfahrungswert.

Was vorher geklärt ist, muss im Review niemand mehr entdecken.

Böckeler 2026 ↗ Faros 2026 ↗ Horthy 2026 ↗

Schema **ROBUST** Balance **UMSTRITTEN**

Teil 5

Was ist neu?

Bekanntes Prinzip, neue Bedeutung

PRINZIP	NEU FÜR DEN AGENTEN	EINORDNUNG
Kohäsion, lose Kopplung	kleiner Kontext, begrenzter Änderungsradius	ROBUST
Ubiquitous Language	grep-bar, weniger Übersetzung	ROBUST
Standards	parametrisches Wissen nutzen	ROBUST
ADRs	Regeln, gegen die ein Review-Agent prüfen kann	ROBUST
Progressive Disclosure	Index statt Volltext im Dauerkontext: AGENTS.md, ADRs, Skills	OFFEN
Tests und Architekturregeln	Alarmanlage, weil er sich nicht beschwert	ROBUST
Verfahrenswissen	Skills, aber nur, was er sonst auslässt	ROBUST
Codequalität	im kleinen Repo Kosten-, im großen Qualitätshebel	UMSTRITTEN
Retro, Scout Rule	Camp Leader schreibt Regeln, der Agent räumt auf	OFFEN

Die Prinzipien sind alt. **Neu ist, warum sie jetzt mehr zählen.**

Je besser die Modelle, desto egaler die Architektur?

Dafür

Bei Sonnet 4.5 und Claude Code hat Codequalität keinen signifikanten Effekt darauf, ob KI-Refactorings die Tests brechen.

Auf sauberem Code ändert sich die Erfolgsquote nicht (Sonnet 4.6 in Claude Code).

Dagegen

Dieselbe Studie: Auf sauberem Code braucht er 7 bis 8 % weniger Tokens und kehrt 34 % seltener zu bearbeiteten Dateien zurück.

In einem großen Repo fällt die Erfolgsquote von 54 auf 24 % (Blog, nicht begutachtet).

Erweitern Agenten ihre eigene Lösung, steigt die strukturelle Erosion in 77 % der Läufe.

Ein Grund: Trainiert wird auf grüne Tests. Die Rechnung für schlechte Architektur kommt erst Monate später.

“A small repository with a good model tolerates almost anything. A large one with a stretched model tolerates nothing.”

Borg u. a. 2026 ↗ Trivedi, Schmitt 2026 ↗ Aguilar 2026 ↗ Orlanski u. a. 2026 ↗ Horthy 2026 ↗

UMSTRITTEN

Was hält

Er weiß viel, aber nicht unser Projekt.	→	Kennt ihn: Was bringt er mit, was müssen wir ihm geben?
Er hat keine Intuition und fragt nicht.	→	Die Intuition müssen wir haben. Was sich prüfen lässt, prüft die Alarmanlage.
Er kopiert, was er vorfindet.	→	Der Platz muss sauber sein, und jedes Wissen an seinem Ort.

WORAUF WIR ALS STEWARD ACHTEN

Technical Debt im Code

Intent Debt in den Artefakten

Cognitive Debt in unseren Köpfen

Intent und Cognitive Debt bemerkt der Agent nicht. **Wenn er uns Zeit freimacht, gehört sie hierhin.**

Storey 2026 ↗

ROBUST

Ein Rezept dafür: das Agentic Trio



agentictrio.org

Daniel Westheide, Fabian Walther

Jede Session ist sein erster Tag.

Der neue Kollege bleibt neu.

Und er bekommt ständig Updates. **Jedes kann die Pflöcke verschieben.**

Wir haben Pflöcke eingeschlagen.
Probiert sie aus und sagt uns, wo sie nicht halten.

Referenzen

TEIL 1

- Pete Hodgson: Why Your AI Coding Assistant Keeps Doing It Wrong, and How To Fix It. 2025
- Rich Hickey: 'AI' is Not a Tool. 2026

TEIL 2

- Margaret-Anne Storey: From Technical Debt to Cognitive and Intent Debt. 2026
- Dex Horthy: No Vibes Allowed: Solving Hard Problems in Complex Codebases. 2025
- AI That Works: Episode 51, No Vibes Allowed March Edition. 2026
- The Weekly Dev's Brew: #21, Dex Horthy. 2026
- Hong, Troynikov, Huber: Context Rot. 2025
- Thibaud Gloaguen u. a.: Evaluating AGENTS.md. 2026
- Jai Lal Lulla u. a.: On the Impact of AGENTS.md Files on the Efficiency of AI Coding Agents. 2026
- Birgitta Böckeler: TDD inside the agent loop. 2026

TEIL 3

- Birgitta Böckeler: Understanding Spec-Driven-Development. 2025
- Gojko Adzic: Specification by Example. 2011
- Birgitta Böckeler: Harness engineering for coding agent users. 2026
- Birgitta Böckeler: Harness Engineering, first thoughts. 2026
- Ryan Lopopolo: Harness engineering: leveraging Codex in an agent-first world. 2026
- Haichao Zhu u. a.: Precedent or Intent? What Coding Agents Follow When Evolving Existing Code. 2026
- Barry Zhang, Keith Lazuka, Mahesh Murag: Equipping agents for the real world with Agent Skills. 2025
- Xiangyi Li u. a.: SkillsBench. 2026
- Tingxu Han u. a.: SWE-Skills-Bench. 2026

Referenzen

TEIL 3, FORTSETZUNG

- Jude Gao (Vercel): AGENTS.md outperforms skills in our agent evals. 2026
- Yifeng He u. a.: Is Progressive Disclosure All You Need for Long-Context Agents? 2026
- Alonso, Yovine, Braberman: TDAD: Test-Driven Agentic Development. 2026
- Zhao u. a.: SpecBench: Measuring Reward Hacking in Long-Horizon Coding Agents. 2026

TEIL 4

- Thibaud Gloaguen u. a.: Coding Agents Don't Know When to Act. 2026
- Peter Naur: Programming as Theory Building. 1985
- Dominik Guhr, Johannes Seitz: Slop oder Top. 2026
- Daniel Westheide, Fabian Walther: Agentic Trio, 12 Thesen. 2026
- Daniel Westheide: Agentic Trio. INNOQ-Webinar. 2026
- Faros AI: AI Engineering Report 2026, The Acceleration Whiplash. 2026
- Dex Horthy: Why Software Factories Fail. 2026

TEIL 5

- Markus Borg u. a.: Code for Machines, Not Just Humans. 2026
- Priyansh Trivedi, Olivier Schmitt: Does Code Cleanliness Affect Coding Agents? 2026
- Javier Aguilar: Bad Code Doesn't Hurt an Agent. Until the Repository Is Large. 2026
- Gabriel Orlanski u. a.: SlopCodeBench. 2026

Danke! Fragen?

INNOQ
www.innoq.com

Was hier offen bleibt, sammeln wir im Chat und beantworten es per Slack oder Blogpost.



Tammo van Lessen
Principal Consultant

LinkedIn



Sven Johann
Senior Consultant

LinkedIn

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
(+49) 2173 3366 0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180 E
63067 Offenbach

Kreuzstr. 16
80331 München

c/o betahaus Hamburg
Eifflerstraße 43
22769 Hamburg

Spichernstraße 44
50672 Köln