

# Taming the Junior

IN THE AGE OF AI – A SMALL MANUAL



inovex



# Hallo,

## Ich bin Lisa

Full Stack Developer bei inovex

☀️ B.Sc. Angewandte Informatik, DHBW Karlsruhe

📖 M.Sc. Software Engineering, University of Amsterdam

🎓 Dozentin für Web Engineering (Frontend) und Python, DHBW

🐕 Außerhalb der IT: Alles was mit Hunden zu tun hat

 [elisabeth.kletsco@inovex.de](mailto:elisabeth.kletsco@inovex.de)

 [@elisabeth-kl](#)



## 25 Jahre Wachstum durch Innovation und erfolgreiche Projekte.



- › Gründung 1999
- › Finanziell und inhaltlich unabhängig
- › Umsatz 2025: 52 Mio. €
- › 500 Mitarbeitende
- › 8 innerdeutsche Standorte mit Hauptsitz in Karlsruhe



in vielen verschiedenen Branchen



von Scale-ups und KMUs bis zu Konzernen



Wir skalieren mit den Anforderungen unserer Kunden



Talent Pipelines

Stärken & Schwächen  
von LLMs

Onboarding-Ideen

## Über was sprechen wir heute?

Wer ist ein Junior?

Cognitive Debt



*Beobachtungen aus der Praxis  
Keine finale Lösung*

**Wer hat von Euch mit Juniors zu tun?**

Chaotisches  
Debugging-Verhalten

Geringe Selbständigkeit

Lösung einfacher  
Aufgaben

## Wer sind diese „Juniors“

< 3-5 Jahre  
Berufserfahrung

Fehlende mentale  
Modelle

*Keine einheitliche Definition*

**Warum habe ich ein ❤️ für Juniors?**

Warum ist mir das Thema so wichtig?

## Anteil von Frauen in Tech-Berufen in Europa könnte weiter zurückgehen



Frauen in IT-Berufen ausgebremst

## Diskriminierung von Frauen in der IT-Branche steigt

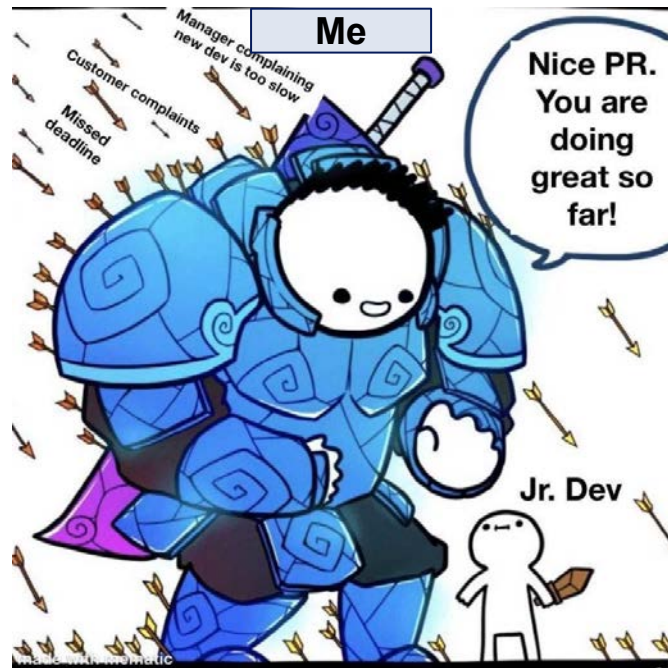
Das Patriarchat prägt Vorurteile, KI verstärkt sie. Beides sind Gründe, warum Frauen es in der Informationstechnik weiterhin schwer haben.

Eine neue Studie zeigt: Vor allem das Klima am Arbeitsplatz treibt Frauen aus Tech-Berufen; andere Faktoren spielen eine deutlich geringere Rolle.



Ich (18-21) wie ich darauf warte,  
dass mich jemand ernst nimmt.

Warum ist mir das Thema so wichtig?



Meine Vision

**5000 YEARS  
OF ACCUMULATED  
HUMAN  
KNOWLEDGE AND KNOWHOW**

**CHATGPT**

imgflip.com



„Wir brauchen **keine Juniors**, wir haben AI!“



*Wunschdenken: Software ist doch nur ein bisschen  
Programmieren und es funktioniert*

Observieren

Programmieren

Debugging

## Aus was besteht Software Engineering\*\*?

\*\*unter anderem

Requirements  
Engineering

Zusammenarbeit im  
Team (Meetings)

*Viel Technik. Aber auch viel Mensch*

Muster erkennen

Durchschnittliche  
Lösungen

Amplifizieren

**Was können LLMs gut 💪 ?**

Syntaktisch valider Code  
bei klar spezifizierten,  
isolierten Aufgaben

Repetitive Aufgaben

# Durchschnittliche Lösungen

We found that the Brain-only group exhibited strong variability in how participants approached essay writing across most topics. In contrast, the LLM group produced statistically homogeneous essays within each topic, showing significantly less deviation compared to the other groups. The Search Engine group was likely, at least in part, influenced by the content that

[1] <https://arxiv.org/pdf/2506.08872v1>

[2] Einordnung als Artikel: <https://time.com/7295195/ai-chatgpt-google-learning-school/>

Implizites Wissen /  
Business-Domäne

Validierung einer  
Lösung

Was können LLMs nicht so gut 🥲?

Debugging / Kausales  
Schließen

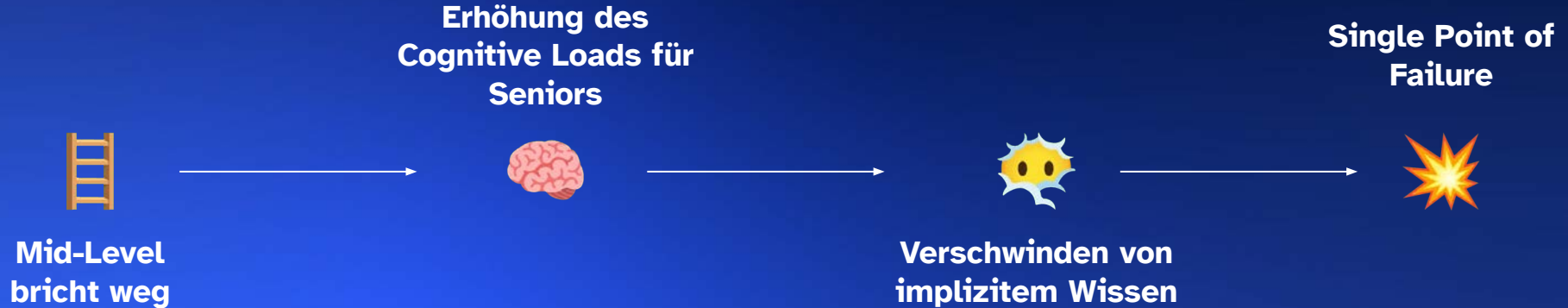
Verantwortung  
übernehmen



*LLMs are statistical pattern matchers, not thinkers.*

**Okay, wir stellen trotzdem keine Juniors mehr ein.**

# Potenzielle Konsequenzen



Es gibt jedoch einen Fall, wann Juniors  
**ersetzbar** werden

Wenn sie nur als ein **menschlicher Proxy für die  
KI** fungieren

*Weniger Coding,  
mehr Orchestrierung*

*Review als ein  
Core Skill*

AI **verändert** die Rolle des Software Engineers

*Systemverständnis  
unerlässlich*

*Site Reliability  
im Fokus*

*Urteilsvermögen*

*Kritisches Denken*

# AI's best coding agent fails 60% of the time — and the data backs it up

AI coding agents stumble on private codebases: Real-SWE benchmark tests models on proprietary code, and the top scorer still failed over 60% of the time.

Sep 14th, 2026 6:22pm by [Amanda Caswell](#)

*Komplexen Business  
Kontext verstehen*

*Angepasste Kommunikation*

*Urteilsvermögen*

*Kritisches Denken*

Wenn wir Juniors jedoch in den Skills fördern,  
**die nur ein Mensch hat**, so werden sie (aktuell)  
unersetzbar.

*Komplexen Business  
Kontext verstehen*

*Angepasste Kommunikation*

**Vielleicht sollten wir doch Juniors einstellen**

„Wie arbeite ich einen Junior **nachhaltig** ein?“



*Bedürfnisse der Gegenwart zu befriedigen, ohne die  
Möglichkeiten zukünftiger Generationen zu gefährden.*

**Das Ziel:** Kein menschlicher Proxy für ein LLM,  
sondern ein unabhängig arbeitendes  
Team-Mitglied



# Vorschlag aus der Praxis: 3 Phasen Onboarding

Vokabular kennenlernen,  
Code lesen, Komponenten  
benennen, Konventionen  
kennenlernen.



## 1. No AI

Funktionen, Control Flow,  
Services verstehen,, Erst die  
eigene Position bilden, dann  
fragen.



## 2. AI als Tutor

Agentic Output kritisch lesen  
und steuern.



## 3. AI als Co-Autor



# Meine 3 Pfeiler für ein Onboarding

Erleben von  
Selbstbestimmung



**Autonomie**

Erleben von  
Selbstwirksamkeit



**Kompetenz**

Erleben von Zugehörigkeit  
und Akzeptanz



**Zugehörigkeit**



# Meine Toolbox

**Ticket-Labeling**

**Mix & Match**

**Non-Code Tasks**

**Culture over  
Curriculum**

**Expliziter  
Wissenstransfer**

**Pair Programming  
& Fixing**

**Bewusstsein für  
Comprehension Debt**

# Ticket-Labelling

#beginner-friendly

#good-first-issue

#challenging

- Einstiegspunkte im Ticket
- Ansprechpersonen für Cross-Team-Services
- Dokumentations-Verlinkungen
- Routine-Aufgaben
- Ticket erfordert wenig Domänen-Wissen oder lässt gut in einen bestimmten Domänen Bereich einarbeiten



Stärkung der Autonomie, Erlernen des Vokabulars

# Mix & Match

Große Features

Kleine Features

Refactorings

Neue Features

Coding-Tasks

Non Coding Tasks

Bug Fixes

Testing



Stärkung der Kompetenz, Erkennen der eigenen Stärken und Schwächen

# Non-Coding Themen zeigen

Debugging

Wissensmanagement

Umgang mit internem  
Tooling

Shortcuts

Gleichzeitig als  
Doku-Chance sehen 😊

Alles, was Euch hilft,  
frustfreier & effizienter zu  
arbeiten

Meetings mit anderen  
Kolleg\*innen

Juniors sichtbar machen

# Culture over Curriculum

Fehler werden als Learnings gesehen, nicht als Belastung

Es gibt keine dummen Fragen – auch keine zur  
Aufgabenstellung selbst

Feedback geht in beide Richtungen



Stärkung des Selbstbewusstseins & Zusammengehörigkeitsgefühls

# Expliziter Wissenstransfer

Wie funktioniert X?

Warum machen wir das  
so?

Wissenslücken?

- Feste Termine im Kalender – werden nur im Notfall verschoben
- Eine Person bereitet vor: Vokabular, Sequenzdiagramm, Schnittstellen zu anderen Teams etc.
- Fragen können geklärt werden
- Ergebnis fließt direkt in die Doku
- Als Regeltermin (z.B. Security-Tuesday) oder im On-Site-Workshop → Verbindlichkeit



Erlernen des Vokabulars, Überführung von implizitem Wissen in explizites

# Pair Programming & Fixing

Show one

Do one

Teach one

**Warum der Aufwand?**



Technical  
Debt



Comprehension  
Debt

**Comprehension Debt:** Die Lücke zwischen dem Code, der im System steht, und dem Verständnis, das im Team davon existiert.

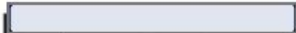


*Der Code funktioniert. Das mentale Modell dazu fehlt.  
Doch wie entsteht diese eigentlich?*

|  Aug 21

Then why do you specify 

---

|  21

Because I did it with Gemini 🏃

# Nutzungsarten: Cognitive Offloading vs Surrender

- Abgabe des *Wie* an die KI, behalten jedoch das *Was*
- Conceptual Questions
- Überprüfung, ob das Ergebnis das ist, was wir wollen

- Übernahme von KI-Antworten, ohne zu Hinterfragen
- Aufnahme von Verständnis-Schulden, bis es zur Gewohnheit wird
- Schleichender Prozess

Durchsatz ist sichtbar,  
Verständnis nicht

**Illusion of Explanatory Depth**

Borrowed Certainty

## Warum sind Developer dafür so anfällig?

Die Oberflächen-Signale  
sehen gut aus

Akkumulation

*Es ist bequem.  
Surrender und Verständnis fühlen sich im ersten Moment  
gleich an.*

Müdigkeit ist ein Stopp-Signal

Woher kommt die eigene  
Sicherheit?

“Kann ich das noch ohne KI?”

## Was kann man dagegen tun?

Erwartung bilden, bevor  
man den Output liest

Einheit des Reviews ist die  
Einheit des Verstehens



*Selbstreflexion  
Den Agent als 2. Developer im Raum sehen, nicht den  
einzigsten*

# Reverse Review

Fragen stellen

Zusammenhänge  
analysieren

Probleme identifizieren



Wissenslücken finden

Was gefällt Dir?

Was gefällt Dir nicht?

**Wer Tokens fürs LLM ausgibt,  
spart Tokens fürs Gehirn**

## **In a nutshell: Why AI (currently) won't do the job**

- Software Engineering ist nicht nur bisschen Programmieren
- AI wird gut darin sein, Routine-Aufgaben zu lösen. Die Nachfrage nach Seniors bleibt stabil, doch woher sollen diese kommen?
- Während AI das reine Code produzieren übernehmen kann, so wird sie ohne Kontext und Anleitung nutzlos sein
- → Wir benötigen Menschen, welche die Domäne verstehen, Lösungen designen und die AI navigieren, um solche Lösungen effektiv umzusetzen.

## **In a nutshell: Taming the Junior? → Guiding**

- AI kann das Onboarding ergänzen, jedoch nicht ersetzen
- Phasen ohne AI, AI als Tutor, AI als Co-Autor
- Autonomie, Kompetenz, Zugehörigkeit fördern
- **Comprehension Debt ernst nehmen**