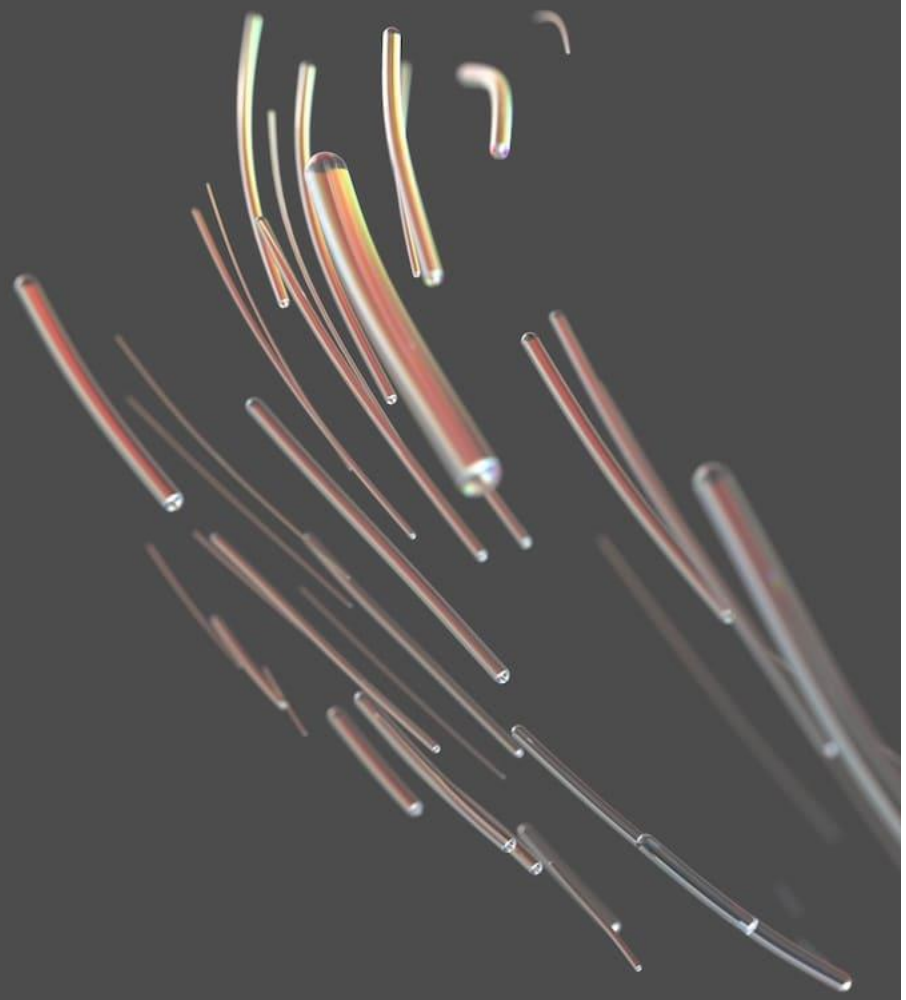


INNOQ Agentic Software Engineering Night #2/ 27. Juli 2026

Slop oder Top

Von Vibe Coding zu AI Aided Software Engineering

Dominik Guhr
Principal Consultant

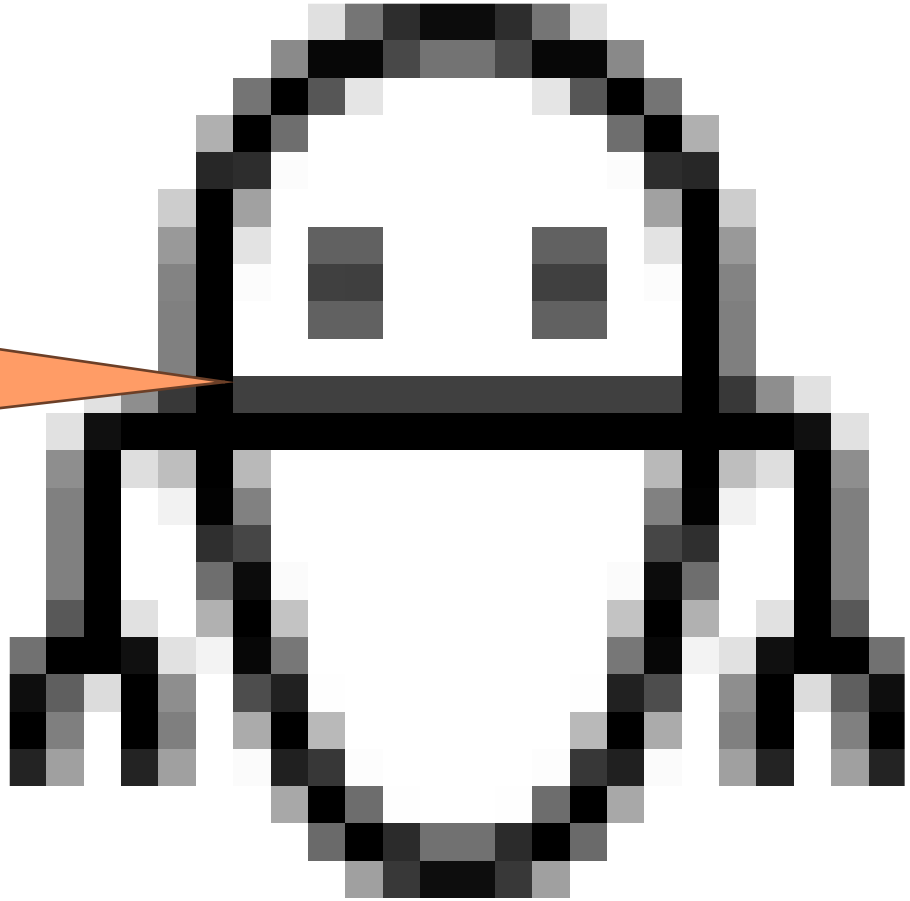
Johannes Seitz
Senior Consultant

INNOQ

Disclaimer:

Wir gehen davon aus, dass ihr langlebige, anpassbare Software bauen wollt.

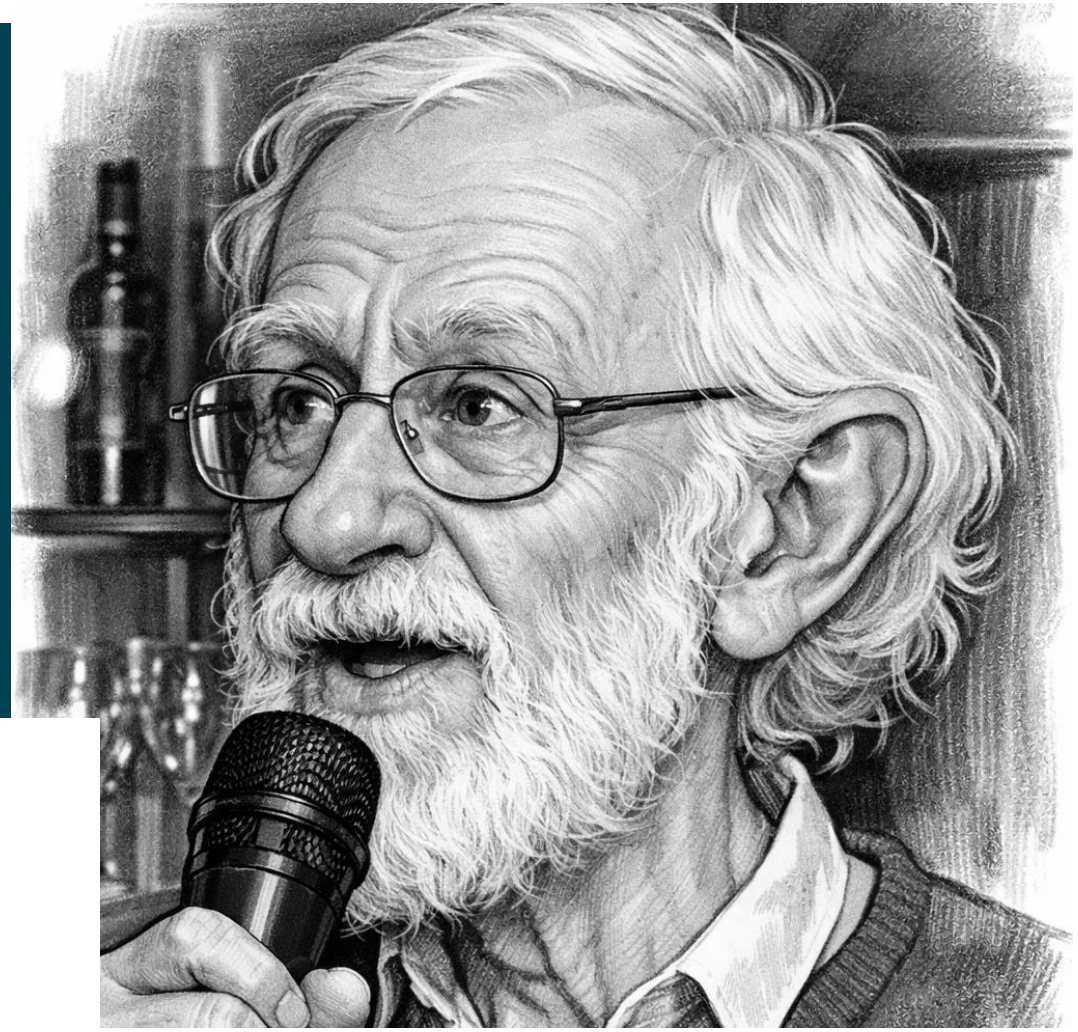
Hi! Ich bin dein Agent. Ich schreibe Code viel schneller, als du es je könntest!



Werden Coding Agents Softwareentwickler:innen komplett ersetzen?

Was ist Programmierung?

„Programming properly should be regarded as an activity by which the programmers form or achieve a certain kind of insight, a theory, of the matters at hand. “



Peter Naur (1928-2016†)

Im Essay „Programming as Theory Building (1985)“

Der Fall aus dem Paper…

Gruppe A

Compiler für Sprache L

Läuft auf Computer X

Gruppe B

Soll Compiler für L+M entwickeln

Für Computer Y

Gruppe B erhält von Gruppe A

Vollständige Dokumentation

Kommentierte Quelltexte

(schriftliche) Design-Diskussionen

Persönliche Zusammenarbeit

Ergebnis: Gut

Team A+B: Einigung auf einfache und effektive Lösung

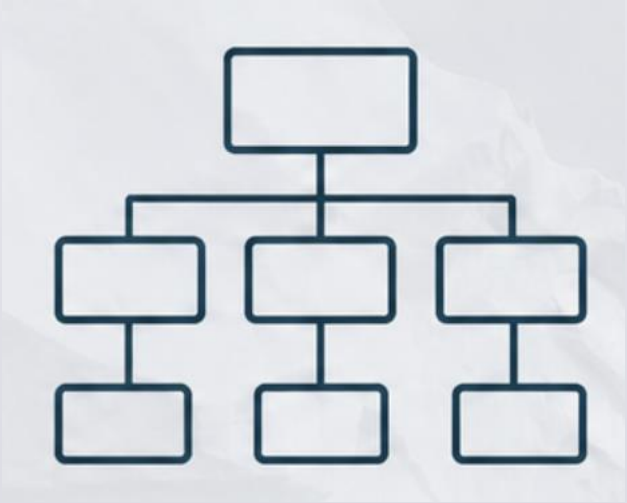
Ein paar Jahre später…

Nach einigen Jahren

Andere Teams übernahmen die Entwicklung des Compilers

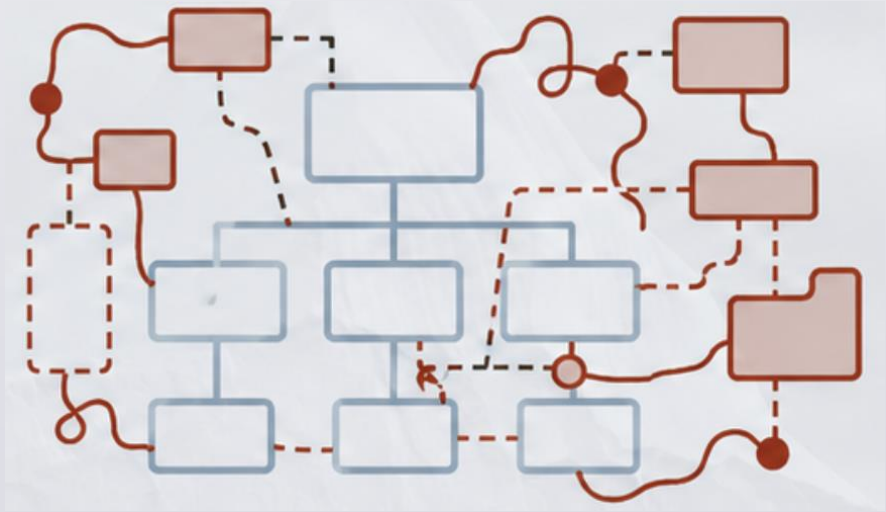
Ohne persönliche Zusammenarbeit mit Gruppe A

Nach ~10 Jahren



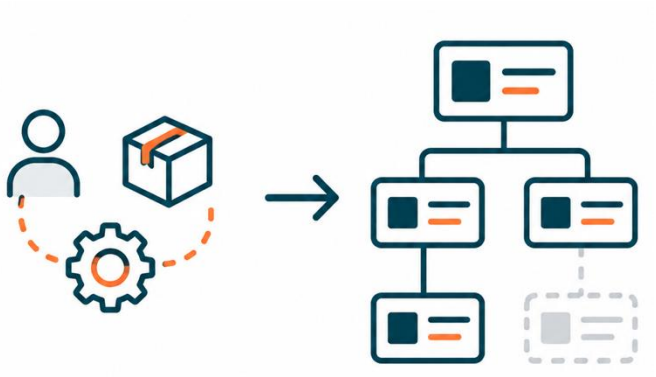
Die ursprüngliche Compilerstruktur war noch erkennbar…

ABER…

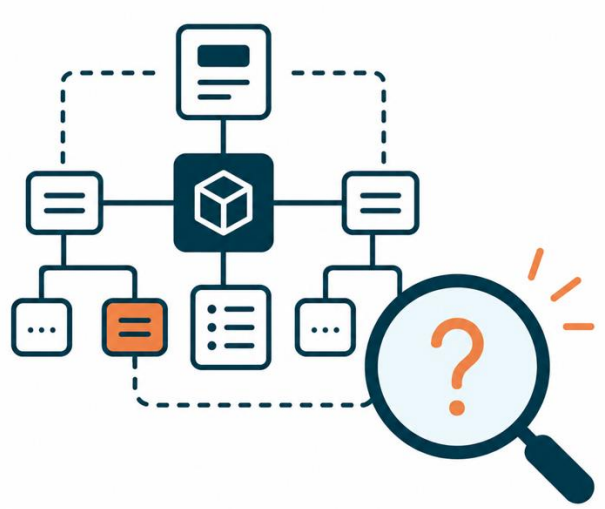


… durch viele amorphe Ergänzungen effektiv unwirksam.

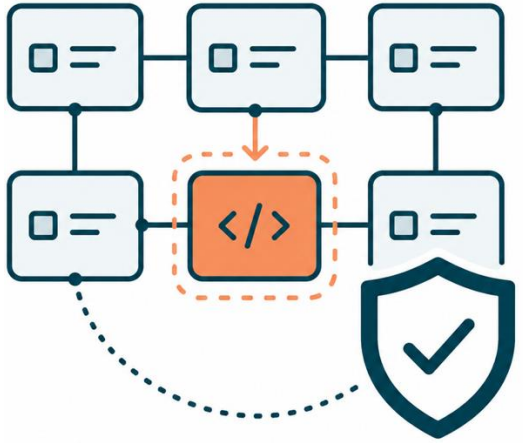
Naur: Hast du eine “Theory”, kannst du:



Erklären, **wie** Konzepte aus der echten Welt in der Software abgebildet sind. **Was** ist mit drin, was wurde (bewusst) ausgelassen?



Erklären, **warum** strukturelle Entscheidungen getroffen wurden.
Warum sieht jeder Teil der Software so aus und ist so strukturiert,
wie er ist?



Erklären, **wie du den Code konstruktiv und nachhaltig abänderst**,
um auf Änderungsanforderungen der echten Welt zu antworten.
Die Änderung spiegelt neue Gegebenheiten wieder, ohne die Integrität des Programms zu zerstören.
Wie ändern wir unsere Software sicher und korrekt?

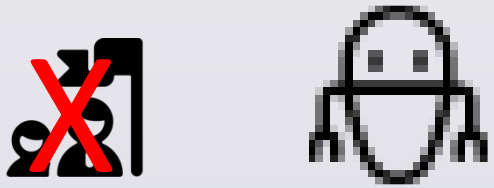
TL;DR

Softwareentwicklung mehr als nur “Schreiben von Programmtext”.

Es ist ein Lernprozess

Und was ist mit GenAI?

2026



Eine KI übernimmt die Entwicklung weitestgehend

Persönliche Gespräche mit Stakeholdern sind nur bedingt möglich (context window, …)

Nach ~10 Jahren

?

Was passiert, wenn Agents den Code schreiben

heise+ Newsticker Security IT & Tech Developer KI Entertainment Wissenschaft Beste

heise online > Künstliche Intelligenz > Künstliche Intelligenz: Vibe-Coding-Dienst Replit löscht Produktionsdatenbank

Künstliche Intelligenz: Vibe-Coding-Dienst Replit löscht Produktionsdatenbank

Security--

Laut einem Replit-User hat der Dienst seine Produktionsdatenbank gelöscht, darüber falsche Aussagen getätigt und Anweisungen ignoriert. Der Hersteller reagiert.

The AI Vampire

Burnout++

Steve Yegge Follow 13 min read · Feb 11, 2026

Vibe coding and databases: the hidden risks of AI-generated database code

Chisom Kanu • 27th April 2026 • 2

Article Recommended About the author

Hidden Risks++

"Vibe coding" with AI can introduce hidden risks in database development, including security vulnerabilities, broken query logic, and data integrity issues.

CEKREM.GITHUB.IO Jesus!? About Recommended Books Podcast (🇳🇴) Blog

Programming as Theory Building: Why Senior Developers Are More Valuable Than Ever

June 26, 2025 7-minute read

How Generative and Agentic AI Shift Concern from Technical Debt to Cognitive Debt

Understanding--

Gunnar Morling • 1.

Technologist @ Confluent · Ex-lead of Debezium · Creator of Hardwood, …

1 Tag(e) • Bearbeitet •

Shower thought this morning: AI agents both lower and raise the bar for contributing to open source, and the latter part deserves more attention.

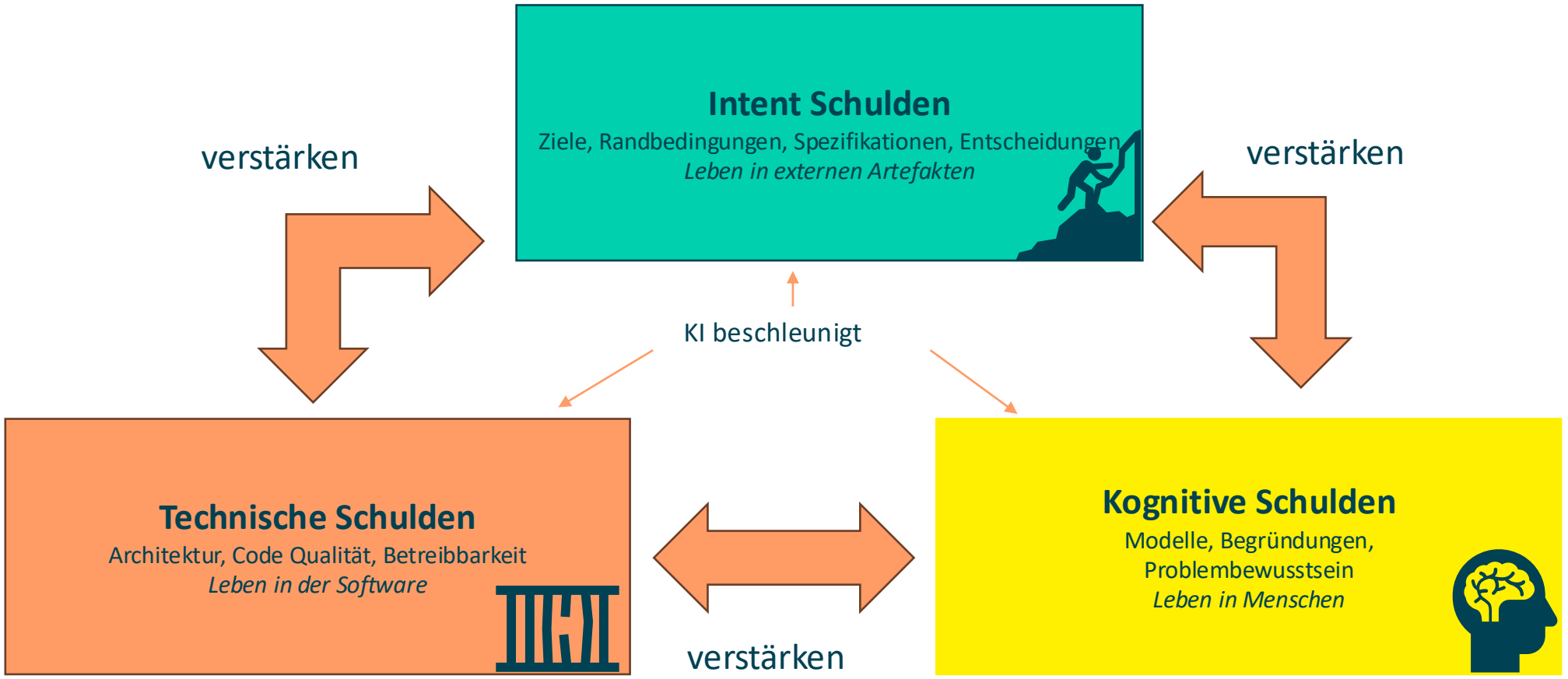
Sure, LLMs make it easier to land your first PR on an unfamiliar OSS project. That part is well-covered.

There's a flip side though. A focused team using agentic tools well can move at a pace that's hard for outside contributors to track. If a project is landing well-considered 5K-LoC PRs that reshape architecture every week, anyone a step removed has to work just to understand the current state of the system. They can point an AI at the diff, yes. But not everyone has the time (or the tokens!) to keep re-onboarding themselves every few days.

It's the old pattern of the one teammate who rewrote half the system overnight while nobody was looking, except ten times over and on a public repo. Good for the roadmap, harder for collaboration.

We talk a lot about AI's effect on maintainers and on code quality. I feel the effect on the contributor pipeline--who can still realistically follow along, and at what cost--is underdiscussed.

Margaret-Anne Storey: Triple Debt Model



From Technical Debt to Cognitive and Intent Debt: Rethinking Software Health in the Age of AI
By Margaret-Anne Storey https://arxiv.org/pdf/2603.22106

Kognitive Schulden

Kognitive Schulden

- „Leben“ in Menschen
- Beschränken wie eine Gruppe Menschen ein IT-System verstehen
- Häufen sich, wenn gemeinsames Verständnis schneller erodiert als es erlangt wird

https://arxiv.org/pdf/2603.22106

Indizien für
“Kognitive Schulden”

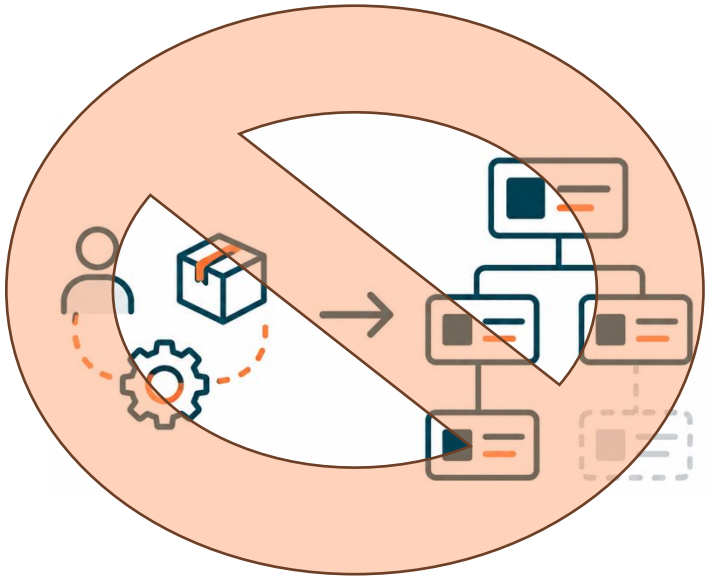
Indizien für kognitive Schulden

- Widerstand gegen Veränderung
- Unerwartete Ergebnisse
- Langsames und unberechenbares Onboarding
- Es ist unklar wen man zu Themen befragen kann
- Niedriger Bus-Faktor
- Erschöpfung beim Versuch die Software zu verstehen

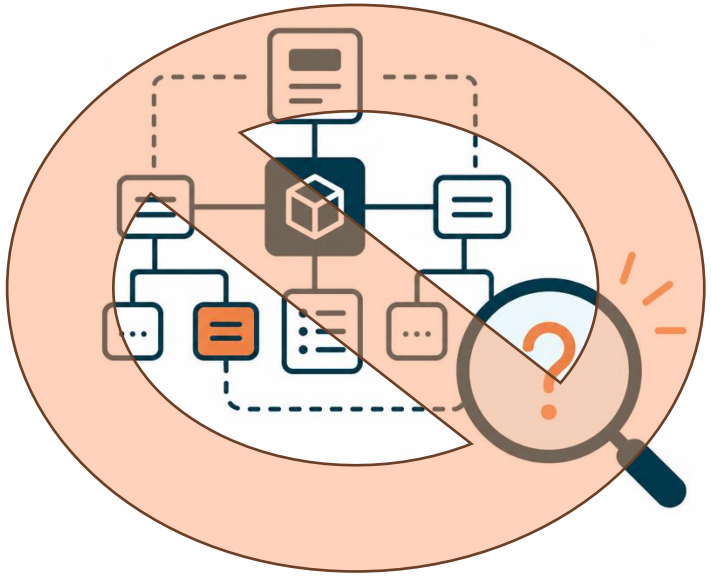
https://arxiv.org/pdf/2603.22106

KI und kognitive Schulden

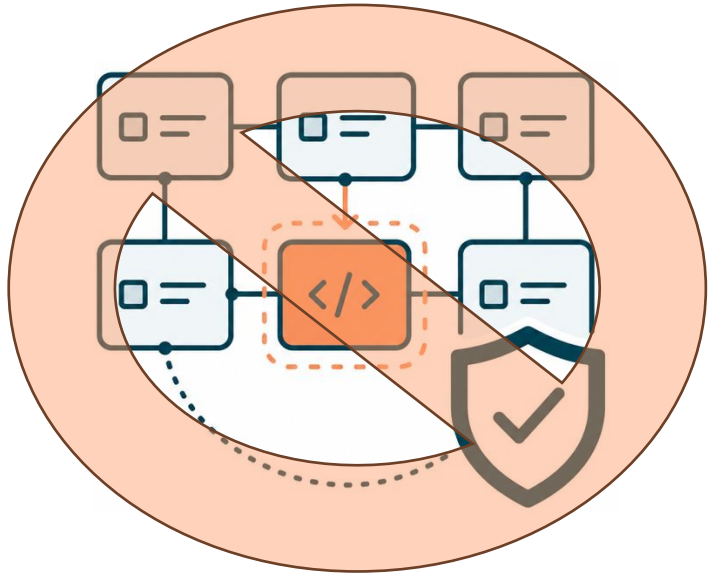
Die Software wird zur “Black Box“



Realwelt-Konzepte <-> Software
Mapping unklar

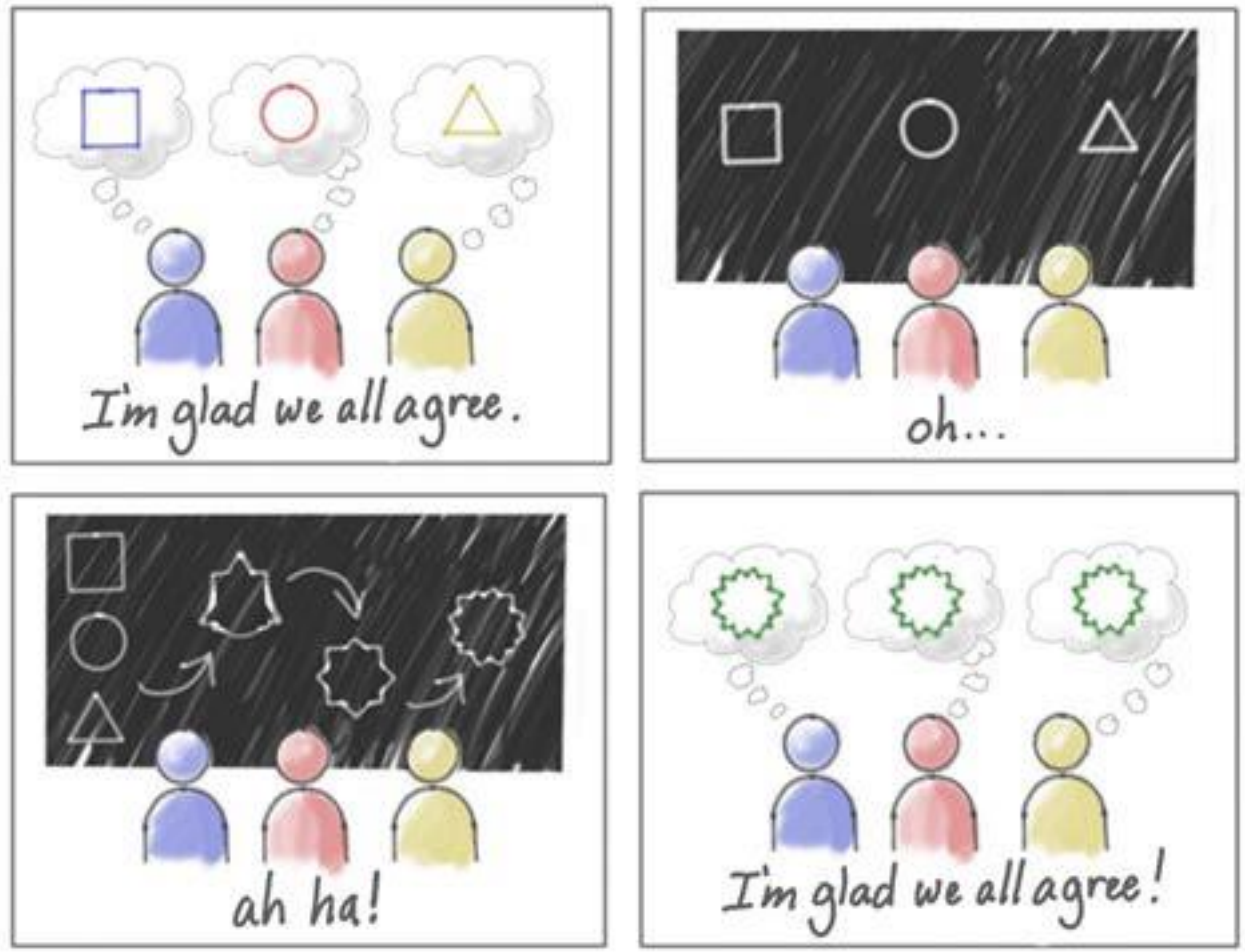


Entscheidungen können
nicht erklärt werden



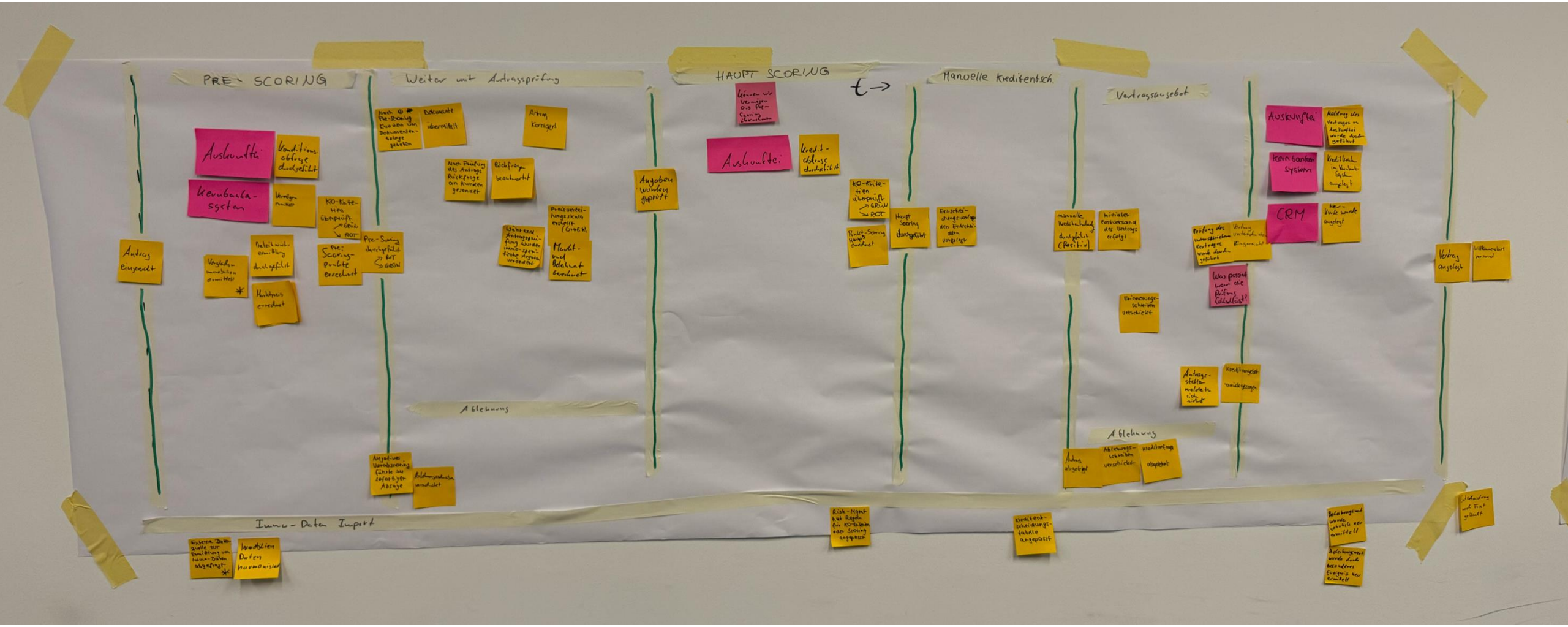
Code kann nicht konstruktiv
und nachhaltig geändert werden

Maßnahmen gegen “Kognitive Schulden”



Grafik nach Jeff Patton – User Story Mapping

Event Storming



Introducing Event Storming – Alberto Brandolini

„When designing software for systems with complex logic, typing code will never become a bottleneck. […]
A developer's job is to problem solve […]
In the end, working code is ultimately the result of learning and understanding the domain.“

Nick Tune, Scott Millett

In “Patterns, Principles and Practices of Domain-driven Design“

Nicht das Modell ist das Ziel –
es ist der Weg dorthin!

Prompt-Techniken

- Kritisches Sparring statt „einfach machen“
  - **Beispiel**: „do not simply affirm my statements [...] Your goal is to be an intellectual sparring partner, not just an agreeable assistant. “
- Bei Unklarheit Rückfragen einfordern
  - **Beispiel: „**Ask questions if something is not clear and you need to make a choice. Don't choose randomly if it's important for what we're doing**“**
- Mentales Modell aufbauen statt black box erzeugen
  - **Beispiel**: „*Work in very small steps and hand them over to me for review*.“

Gist mit Vorlage →

Habitat thinking

RUSS MILES

Softwareentwicklung ist nicht Produktion, sondern Aufbau eines Ökosystems

Code = Lebensraum

Code ist nicht das Endprodukt, sondern der Raum, in dem Teams, Stakeholder und Nutzer:innen sich aufhalten.

KI = Mitbewohner

Keine Fließband-Maschine, sondern eine andere Intelligenz, mit der wir diesen Lebensraum teilen.

Grundidee

„Lebensraum pflegen“ statt „Fabrik optimieren“ – es zählt die Gesundheit des Habitats, nicht die Output-Geschwindigkeit.

Wie müssen wir mit der KI interagieren, damit unser gemeinsames Verständnis über das Habitat nicht verloren geht?

Intent-Schulden

Intent Schulden

- Leben in (non-code) Artefakten
- Häufen sich, wenn Ziele und Constraints schnell unklar werden
- Verhindern, dass Systeme weiterhin das tun, wofür sie gebaut wurden
- Limitieren, wie Menschen und Agenten Systeme weiterentwickeln können

https://arxiv.org/pdf/2603.22106

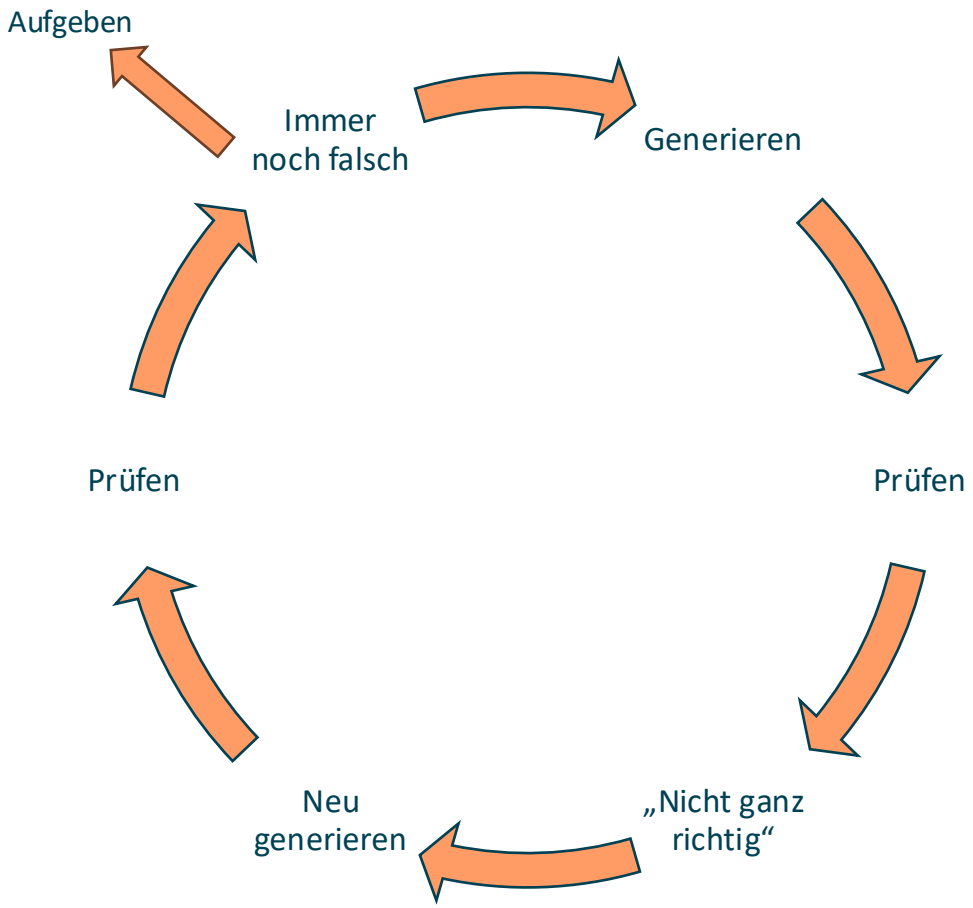
Indizien für “Intent-Schulden”

Indizien für Intent-Schulden

- Nichts ist niedergeschrieben, Anforderungen und Randbedinungen werden vergessen
- Änderungen passen nicht zusammen, niemand weiß warum
- Agenten haben Probleme Entscheidungen zu treffen, sie treffen falsche, aber plausible Entscheidungen

https://arxiv.org/pdf/2603.22106

Frustrationsschleife (nach Rahul Garg)



- KI erzeugt Lösungen, die nicht zur bestehenden Architektur passen
- Entwickler verbringen viel Zeit mit Nacharbeit der generierten Ausgabe
- Früh im Gespräch etablierter Kontext geht im Lauf der Sitzung verloren
- Die Qualität schwankt je nachdem, welches Teammitglied promptet

Quelle: https://martinfowler.com/articles/reduce-friction-ai/

Maßnahmen gegen “Intent-Schulden”

Write it down!

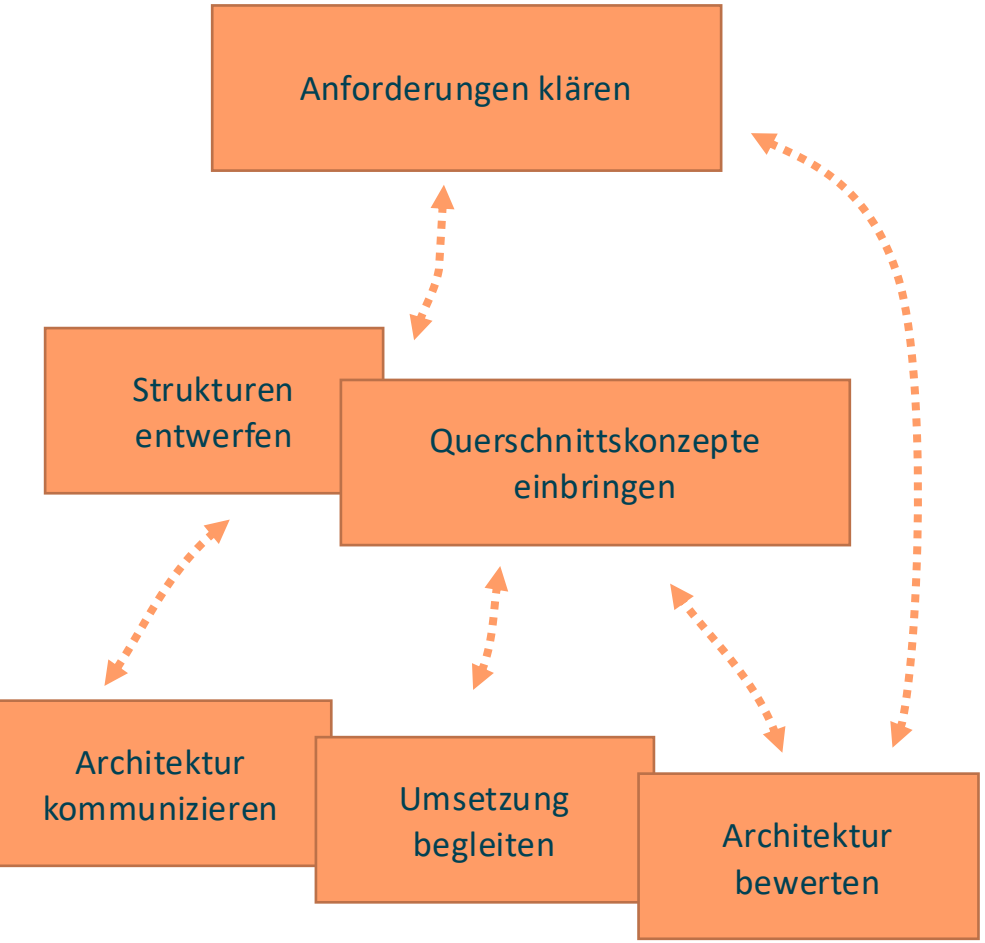
Qualitätsziele

Randbedingungen

Architekturentscheidungen (ADRs)

Querschnittliche Konzepte

arc42: Technische Intent Doku



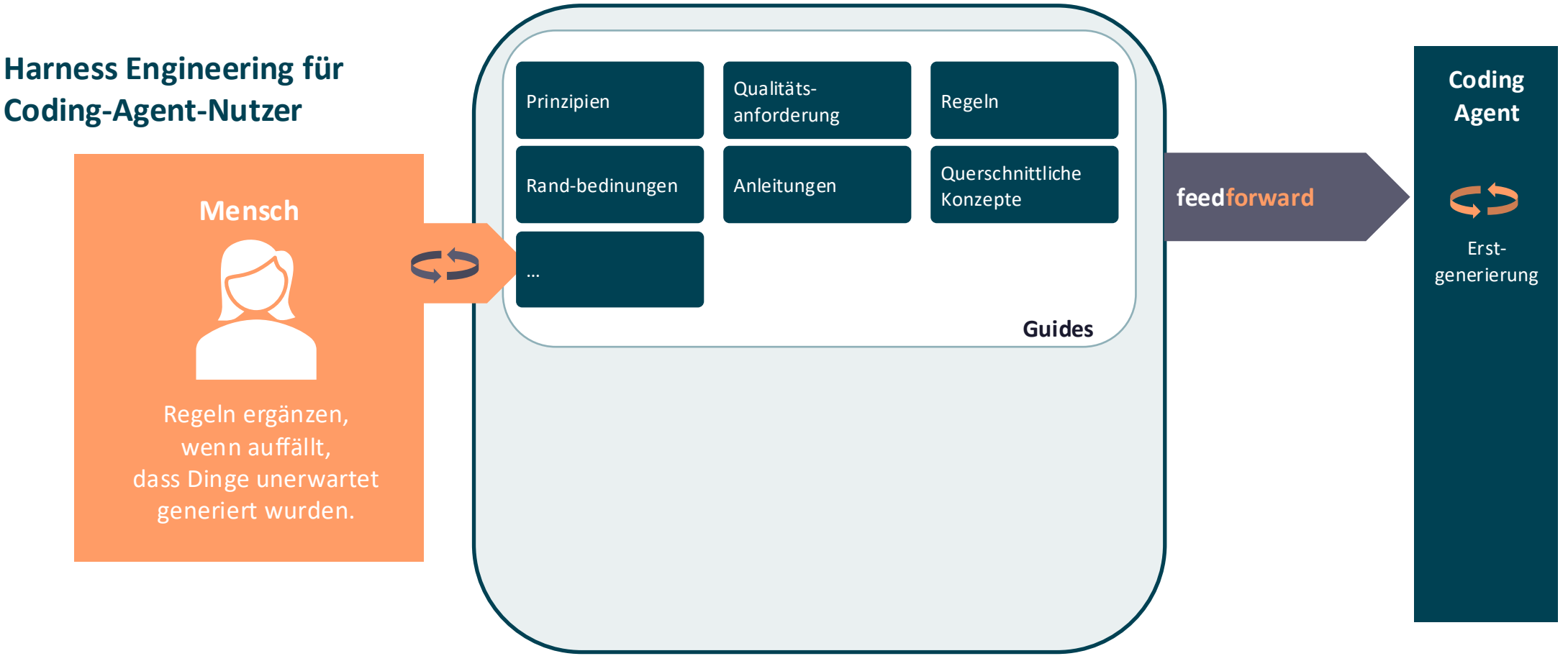
arc42

1. Introduction & Goals Fundamental requirements, esp. quality goals	**7. Deployment View** Hardware, infrastructure & deployment
2. Constraints Regulations and external constraints	**8. Crosscutting Concepts** Cross-cutting topics, often very technical and detailed
3. Context & Scope External systems & interfaces	
4. Solution Strategy Core ideas and solution approaches	**9. Architectural Decisions** Important decisions (not described elsewhere)
5. Building Block View Structure of source code, modularization (hierarchical)	**10. Quality Requirements** Quality tree, quality scenarios
	11. Risks & Technical Debt Known problems and risks
6. Runtime View Important runtime scenarios	**12. Glossary** Important and specific terms ("ubiquitous language")

Nicht nur Menschen- sondern auch agentlesbar
z. B. als .md im docs/ Verzeichnis.

https://arc42.org/

Intent bewahren durch Harness Engineering



Nach einem Modell von Birgitta Böckler:
https://martinfowler.com/articles/harness-engineering.html

Technische Schulden

Technische Schulden

- Leben im Code
- Häufen sich, wenn Implementierungsentscheidungen zulasten der Änderbarkeit und Wartbarkeit gehen
- Limitieren, wie Systeme sich ändern können

https://arxiv.org/pdf/2603.22106

Indizien für
“Technische Schulden”

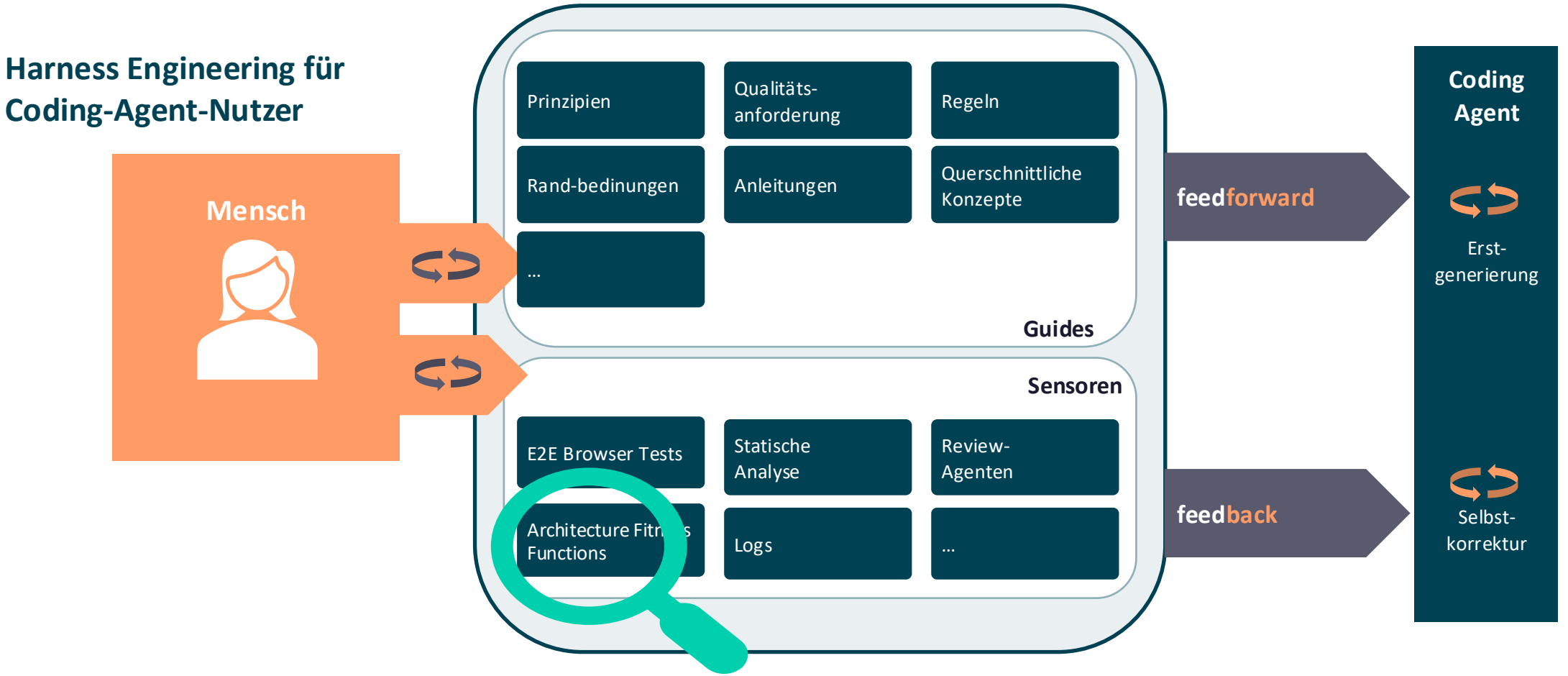
Indizien für technische Schulden

- Spaghetti Code
- Cyclomatic Complexity
- Lange Vorlaufzeit für Änderungen
- Hohe Change Failure Rate
- …

→Das Konzept ist verstanden.

Maßnahmen gegen “Technische Schulden”

Konzeptionelle Integrität durch Harness Engineering



Nach einem Modell von Birgitta Böckler:
https://martinfowler.com/articles/harness-engineering.html

Architecture Fitness Functions

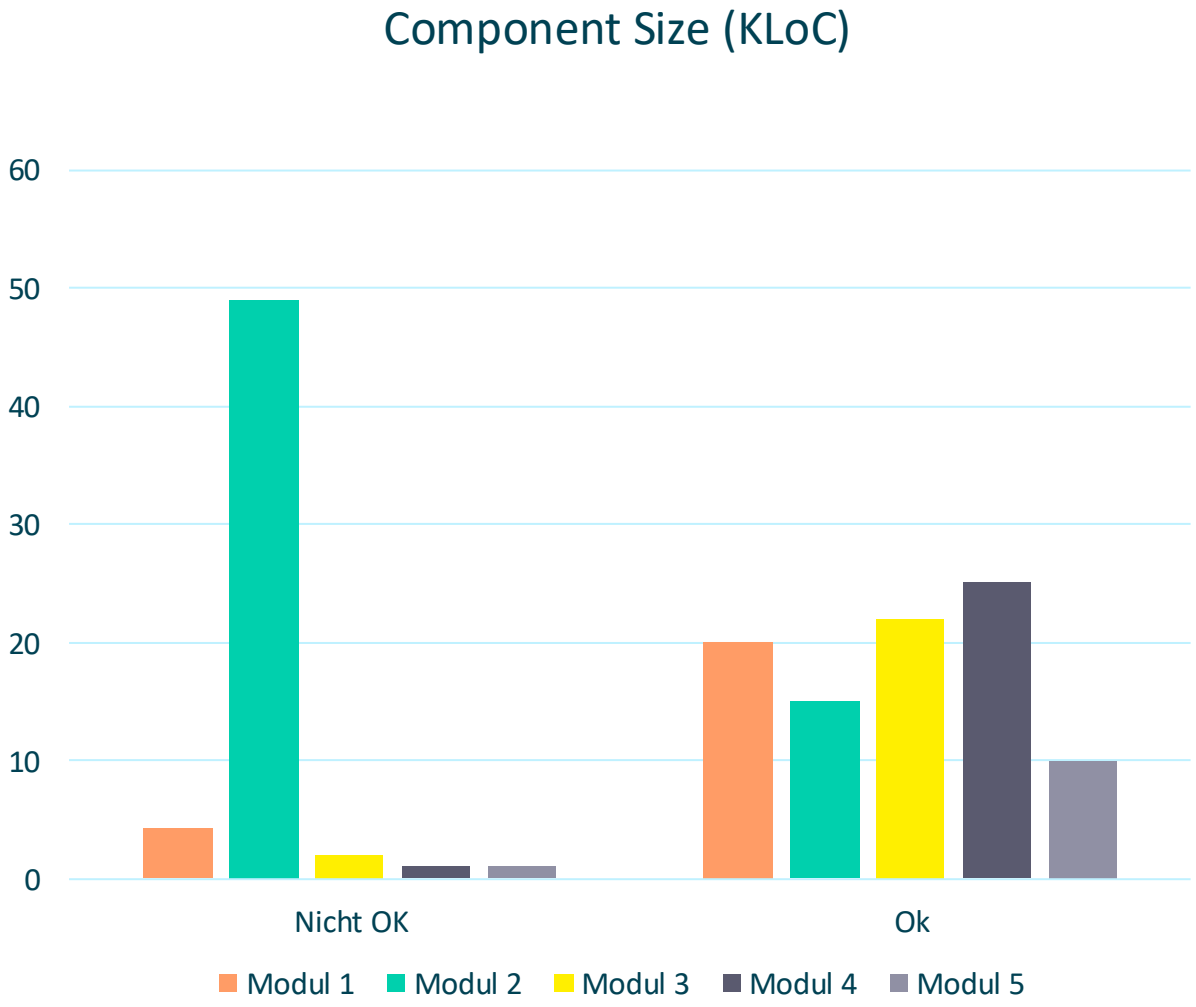
Example 5-3. Pseudocode for maintaining component size based on number of standard deviations

```
# Walk the directory structure, creating namespaces for each complete path
LIST component_list = identify_components(root_directory)

# Walk through all of the source code to accumulate total statements and number
# of statements per component
SET total_statements TO 0
MAP component_size_map
FOREACH component IN component_list {
  num_statements = accumulate_statements(component)
  ADD num_statements TO total_statements
  ADD component,num_statements TO component_size_map
}

# Calculate the standard deviation
SET square_diff_sum TO 0
num_components = get_num_entries(component_list)
mean = total_statements / num_components
FOREACH component,size IN component_size_map {
  diff = size - mean
  ADD square(diff) TO square_diff_sum
}
std_dev = square_root(square_diff_sum / (num_components - 1))

# For each component calculate the number of standard deviations from the
# mean. Send an alert if greater than 3
FOREACH component,size IN component_size_map {
  diff_from_mean = absolute_value(size - mean);
  num_std_devs = diff_from_mean / std_dev
  IF num_std_devs > 3 {
    send_alert(component, num_std_devs)
  }
}
```



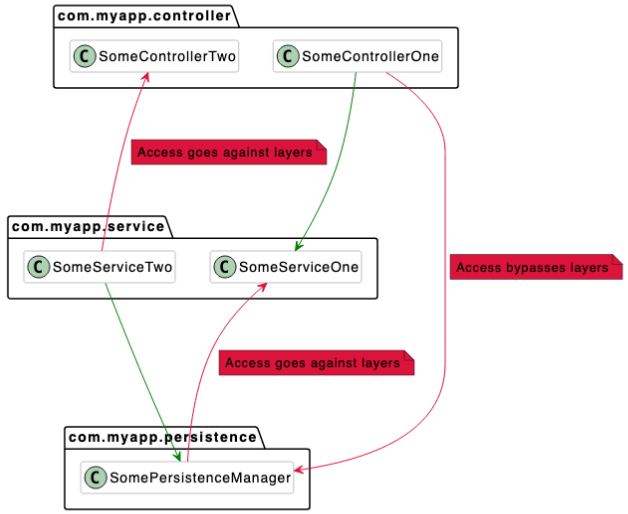
Beispiel aus “Software Architecture: The Hard Parts”

https://www.oreilly.com/library/view/software-architecture-the/9781492086888/

ArchUnit

Arch-Go

Layer Checks




```
layeredArchitecture()
    .consideringAllDependencies()
    .layer("Controller").definedBy("..controller..")
    .layer("Service").definedBy("..service..")
    .layer("Persistence").definedBy("..persistence..")

    .whereLayer("Controller").mayNotBeAccessedByAnyLayer()
    .whereLayer("Service").mayOnlyBeAccessedByLayers("Controller")
    .whereLayer("Persistence").mayOnlyBeAccessedByLayers("Service")
```

File arch-go.yml

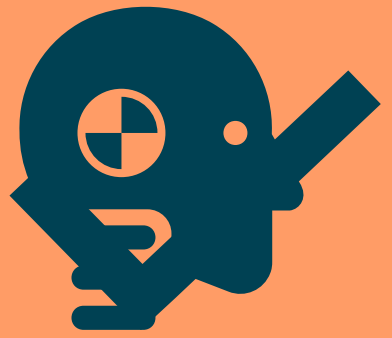
```yaml
version: 1
threshold:
  compliance: 100
  coverage: 100
dependenciesRules:
  - package: "**.impl.*"
    shouldOnlyDependsOn:
      internal:
          - "**.foo.*"
          - "*.bar.*"
    shouldNotDependsOn:
      internal: ["**.model.**"]
  - package: "**.utils.**"
    shouldOnlyDependsOn:
      - "**.model.**"
  - package: "**.foobar.**"
    shouldOnlyDependsOn:
      external:
        - "gopkg.in/yaml.v3"
  - package: "**.example.**"
    shouldNotDependsOn:
      external:
        - "github.com/foobar/example-module"
contentsRules:
  - package: "**.impl.model"
    shouldNotContainInterfaces: true
```

https://www.archunit.org/

… sowie viele weitere Open Source Tools …

https://github.com/arch-go/arch-go

Es gibt noch mehr:
“Verifizierungs-Schulden”

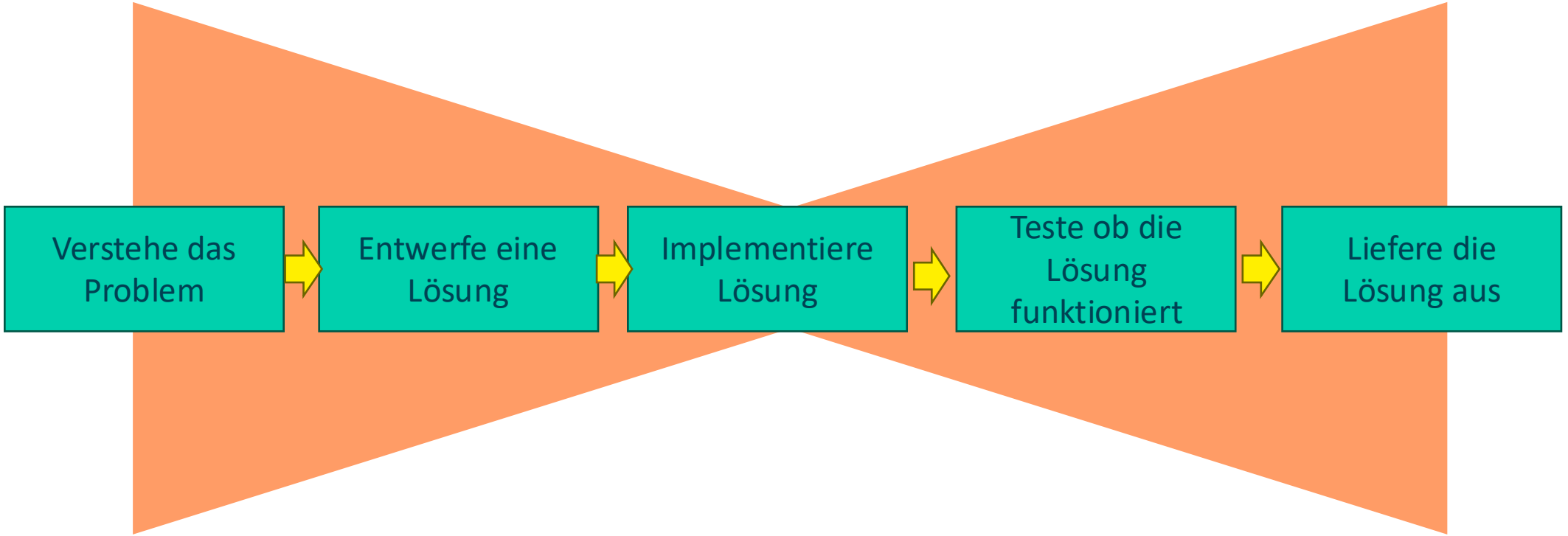


Wenn du denkst, die Geschwindigkeit in der Code geschrieben werden kann ist das Problem, hast du eigentlich tieferliegende Probleme.

- Andrew Murphy

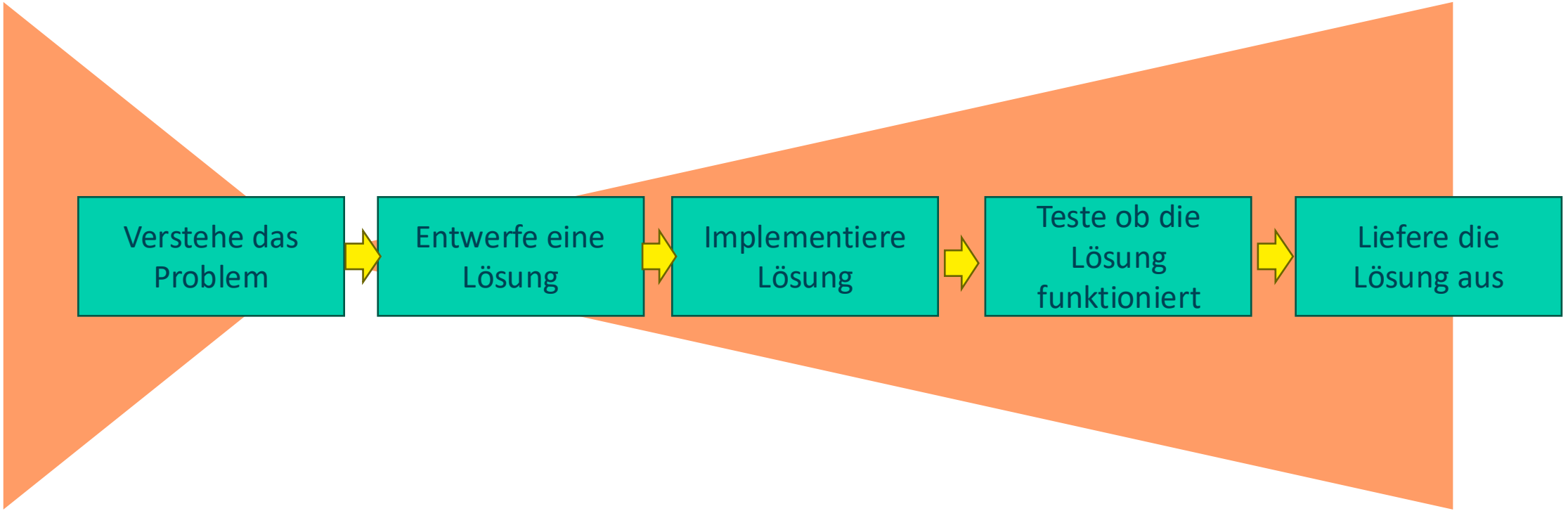
https://andrewmurphy.io/blog/if-you-thought-the-speed-of-writing-code-was-your-problem-you-have-bigger-problems

Shifting Bottlenecks



Shifting Bottlenecks

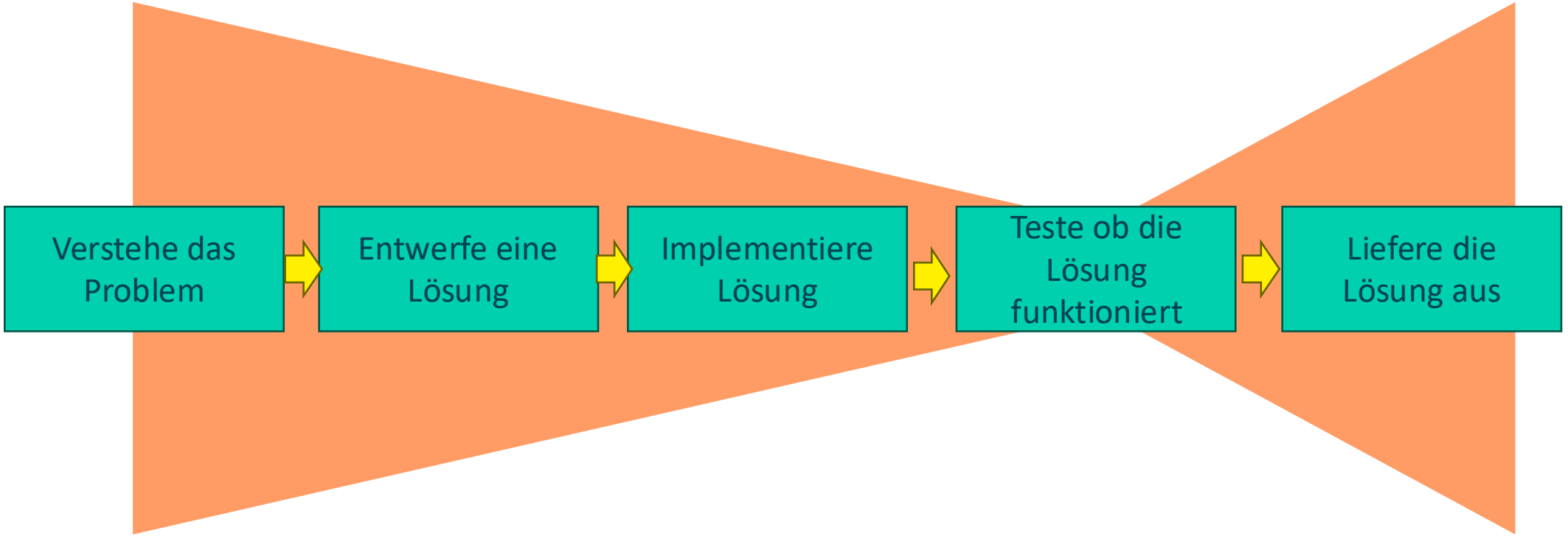
Szenario:
Eine Firma mit hohen technischen und/oder kognitiven Schulden



https://www.innoq.com/de/blog/2025/07/erst-agil-dann-agentisch/

Shifting Bottlenecks

Szenario:
Firma mit gut funktionierendem Build-Measure-Learn Loop, der ständig neue Ideen für Funktionen generiert.



Most important mesaure of progress:
Working Software

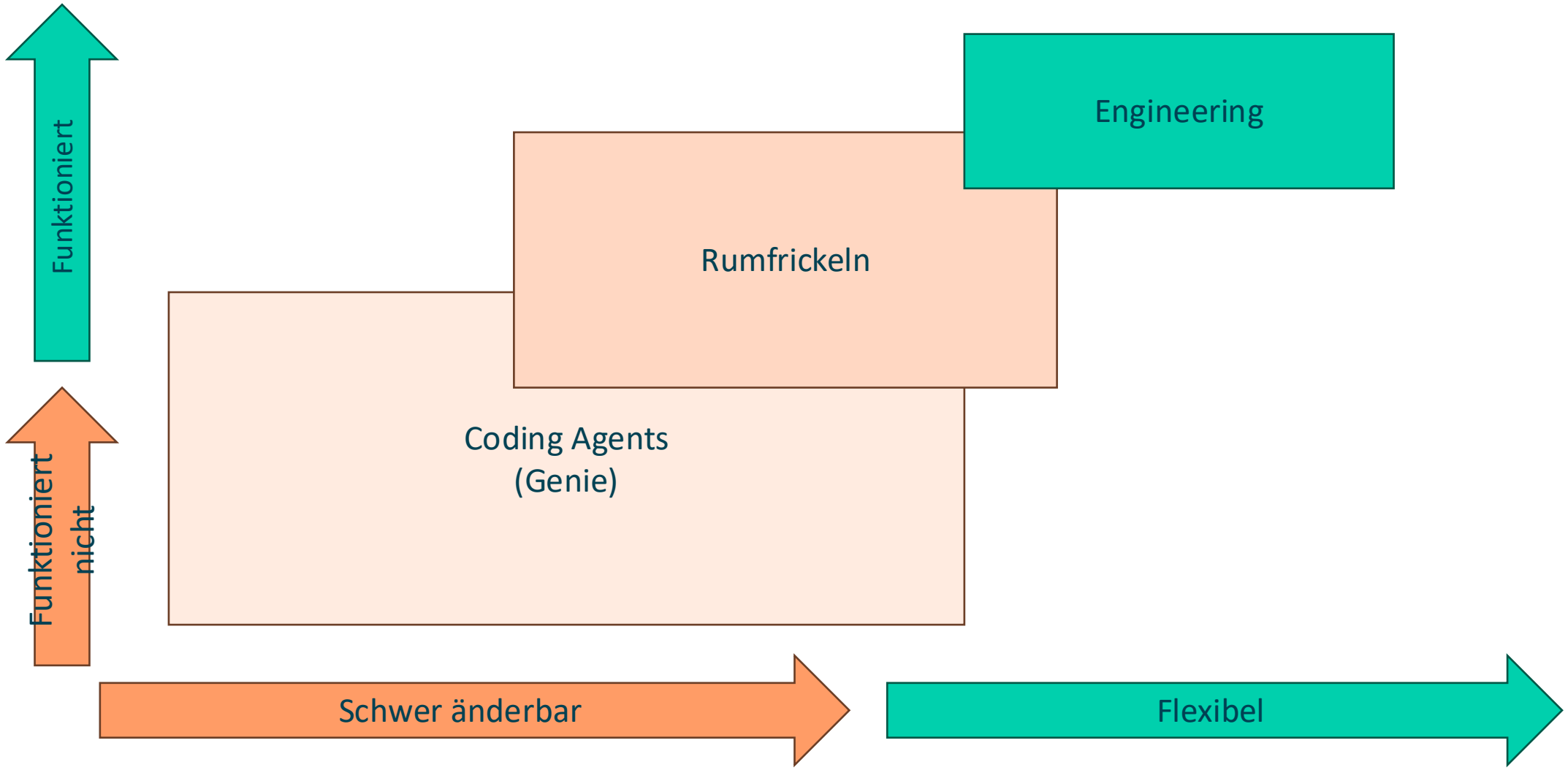
Fazit

GenAI-produzierter Code der Schulden jeder Art verstärkt

(Kognitiv, Intent, Technisch)

Bild: https://mastodon.social/deck/@Dany@hsnl.social/115847527651095264

Kent Beck‘s Genie-Tarpit



https://tidyfirst.substack.com/p/genie-tarpit

Menschen schließen die Verständnislücke

You generate it, you own it.

⇒ Investiert in Harness- und Habitat Engineering.

⇒ Nutze das LLM als Lernhilfe nicht als “Arbeitssklaven"

⇒ Verstehe Code und Fachdomäne

⇒ Verstehe deine Technologien um Entscheidungen des Agenten hinterfragen zu können.

Verliert ihr die Fähigkeit, KI-Output qualitativ einzuschätzen habt ihr verloren.

Danke! Fragen?

Dominik Guhr
dominik.guhr@innoq.com

Johannes Seitz
johannes.seitz@innoq.com

INNOQ
www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Wendenstraße 130
20573 Hamburg

Spichernstrasse 44
50672 Köln