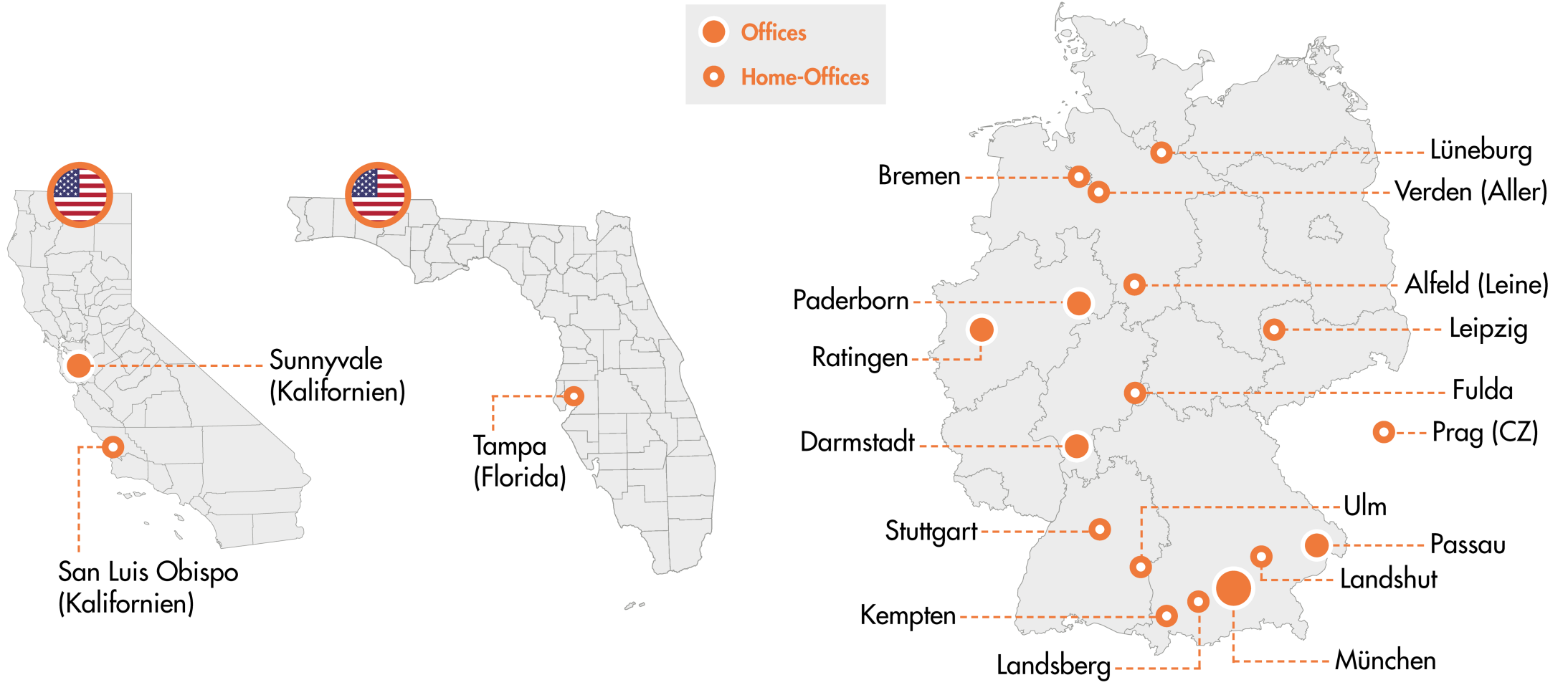
Teamscale by CQSE

Softwarequalität trotz, wegen und mit KI

Wie agentische Entwicklung und verlässliche Softwarequalität zusammenfinden



Dr. Benjamin Hummel



Teamscale

CQSE
Continuous Quality in Software Engineering

Softwarequalität

Softwarequalität

Correctness

Usability



Safety

Security

Accessibility

Reliability

Performance

Scalability

AI GENERATED

Softwarequalität

Correctness

Usability

Safety

Security



Accessibility

Reliability

Performance

Scalability

AI GENERATED



Qualität ist nicht für alle Systeme gleich wichtig

AI GENERATED

“KI”

(Coding Agents)

Was ist ein (Coding) Agent?

Claude Opus 5.5 Claude Code

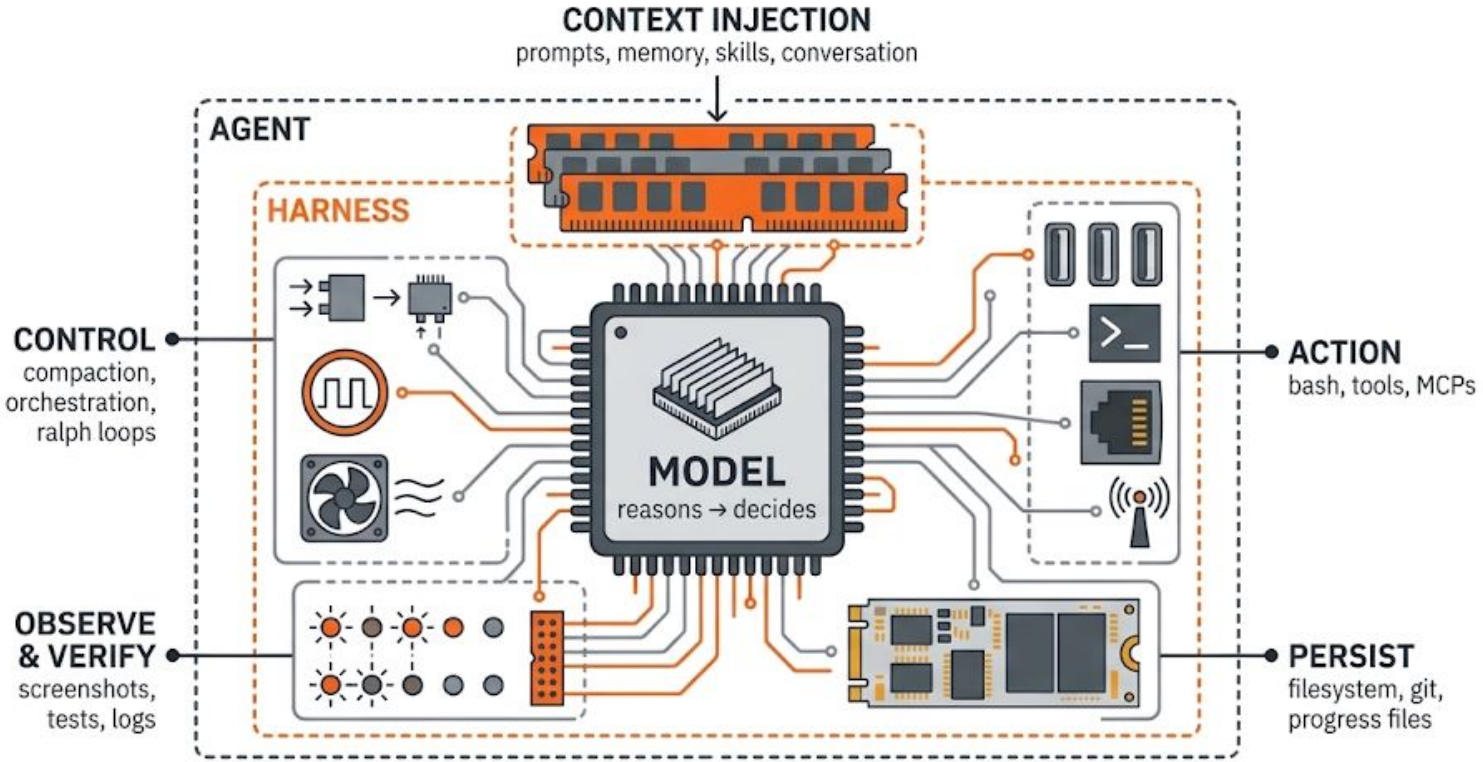
Agent = Model + Harness

Large Language Model (LLM)

```
fun llm (string) → string
```

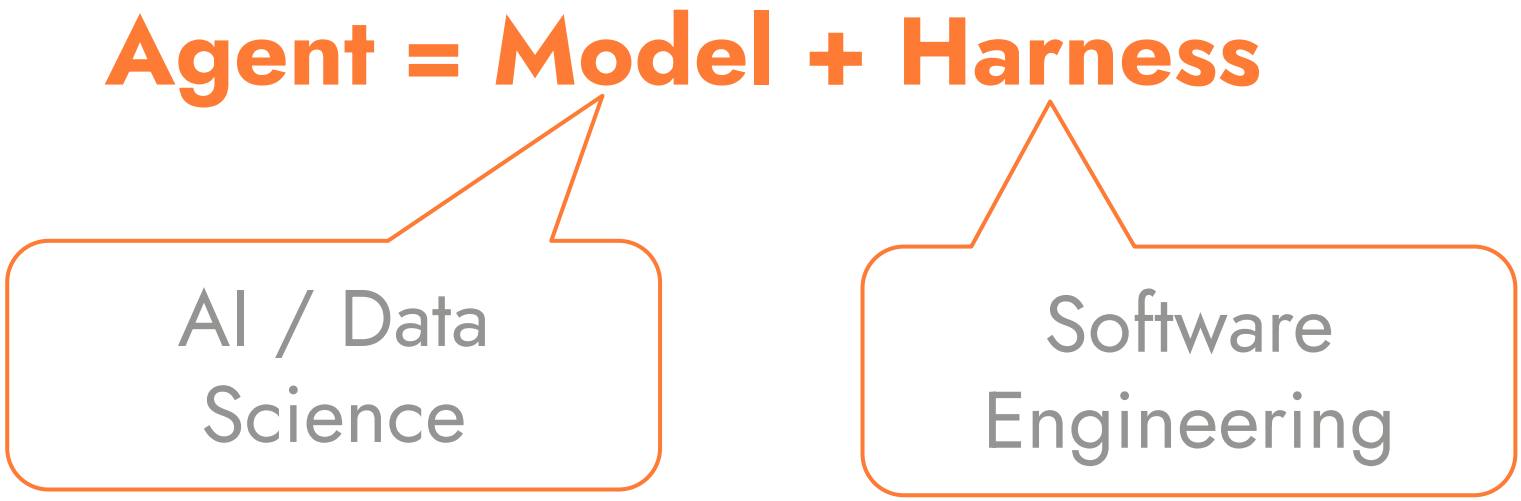
"If you're not the model, you're the harness."

https://www.oreilly.com/radar/agent-harness-engineering/

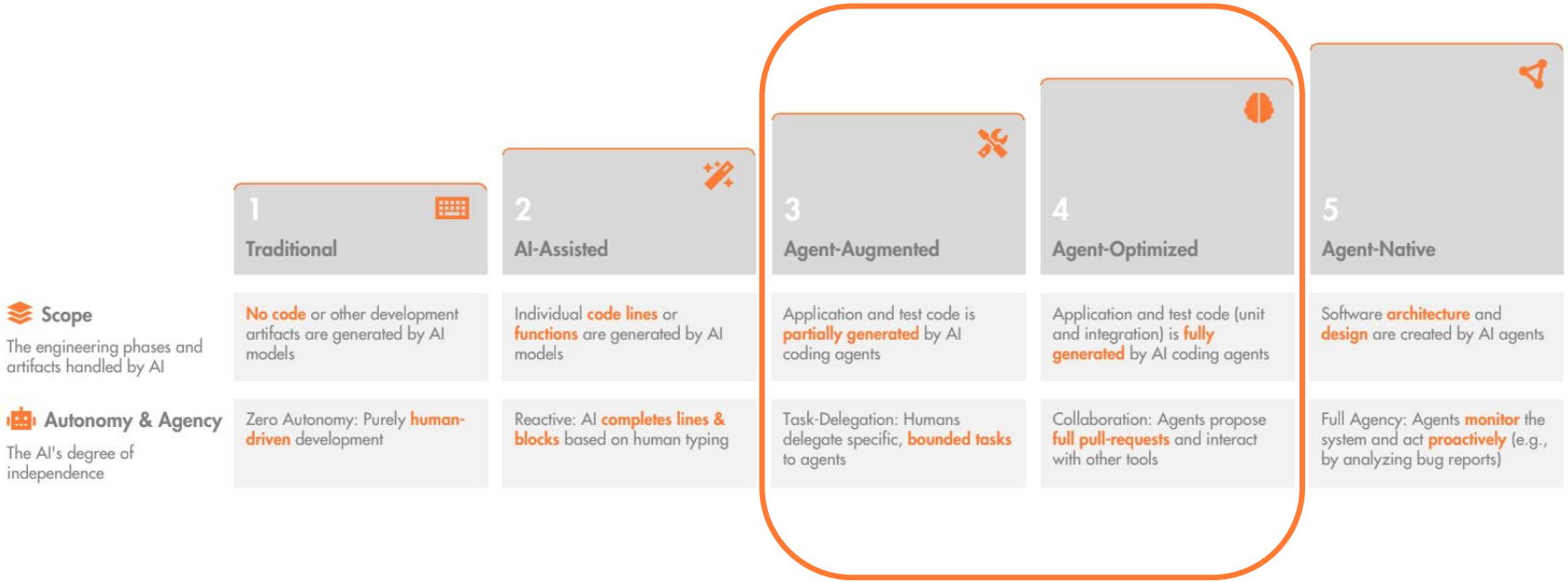


Based on https://www.oreilly.com/radar/agent-harness-engineering/

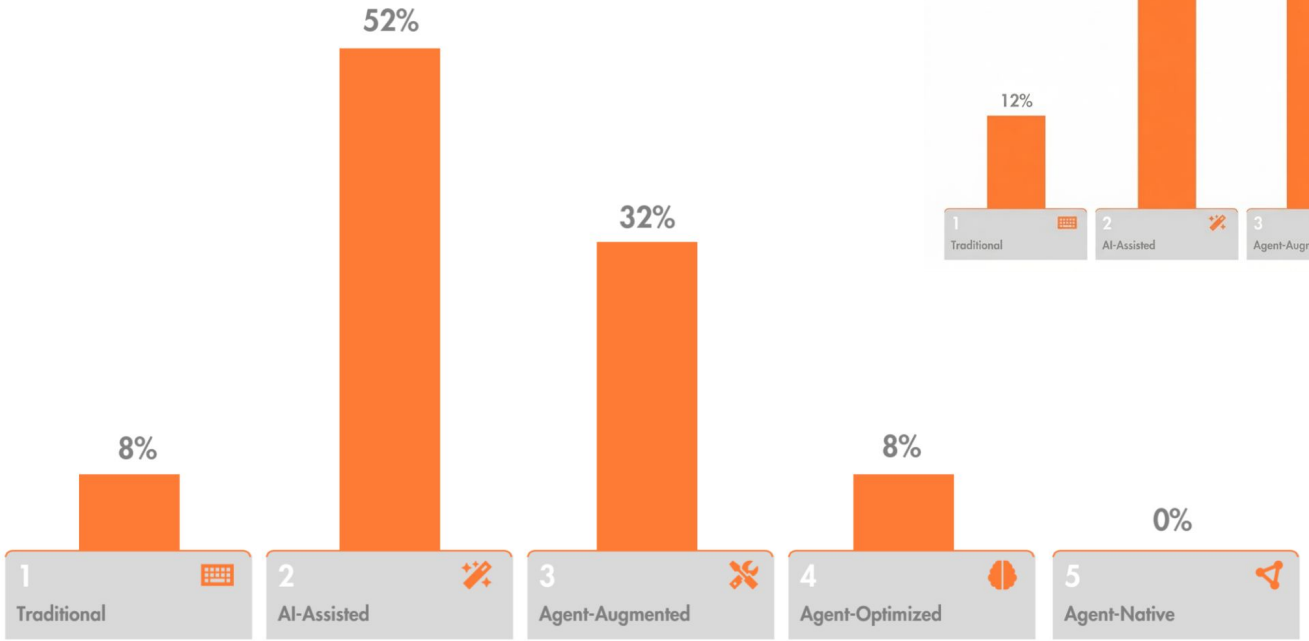
Was ist ein (Coding) Agent?

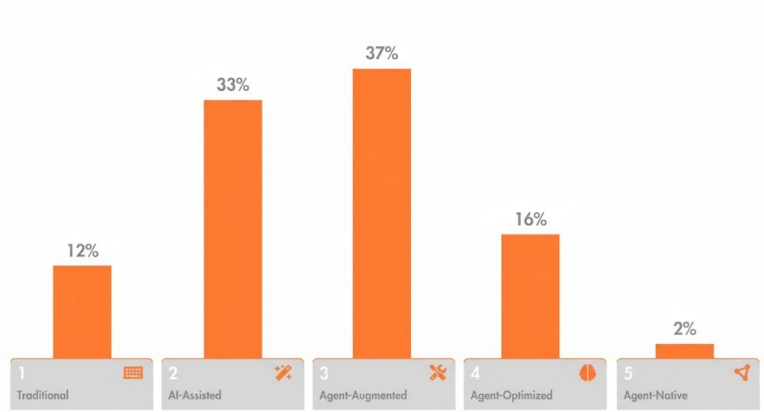


Agentic Development Maturity Model



Wo stehen unsere Kunden





Wie verändern Coding Agents die Softwareentwicklung?

Coding Agents sind nichtdeterministisch

$ claude -p "Output a python program for calculating the factorial"

```python
def factorial(n: int) -> int:  1 usage  new *
    """Return n! for a non-negative integer n."""
    if not isinstance(n, int) or n < 0:
        raise ValueError("n must be a non-negative integer")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result


if __name__ == "__main__":
    try:
        n = int(input("Enter a non-negative integer: "))
        print(f"{n}! = {factorial(n)}")
    except ValueError as e:
        print(f"Invalid input: {e}")
```

```python
def factorial(n: int) -> int:  1 usage  new *
    """Return n! iteratively."""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result


def factorial_recursive(n: int) -> int:  1 usage  new *
    """Return n! recursively (limited by Python's recursion depth)."""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    return 1 if n <= 1 else n * factorial_recursive(n - 1)

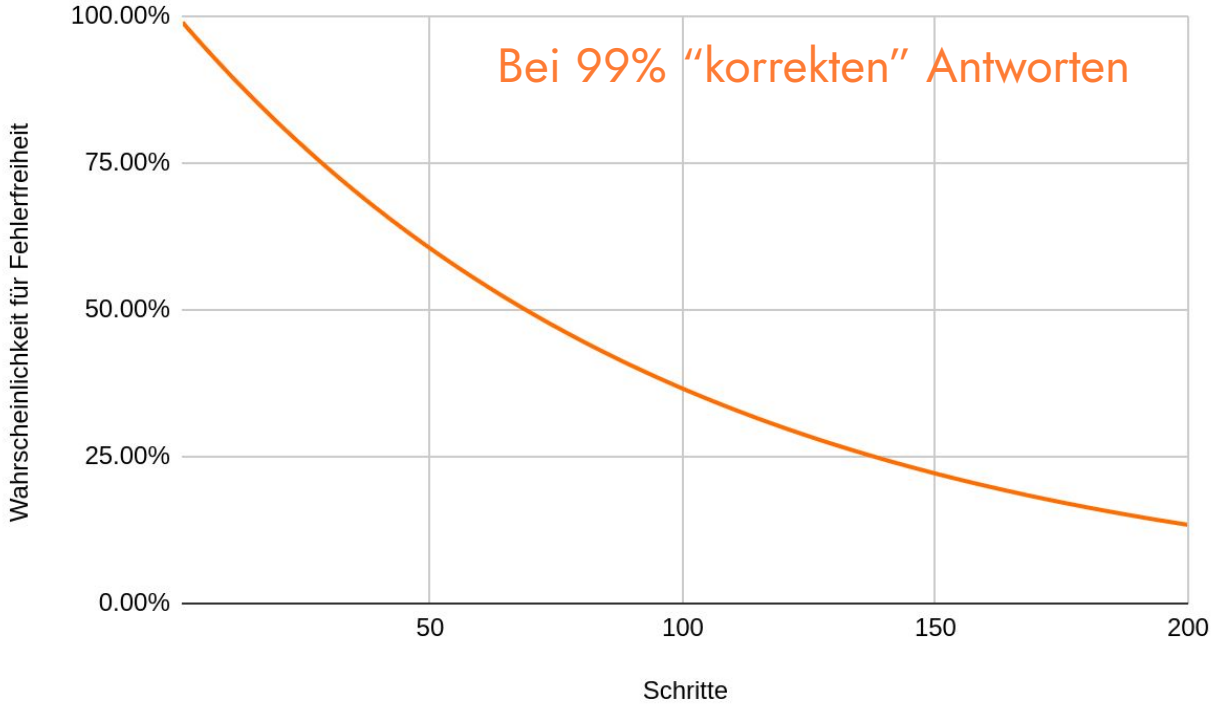

if __name__ == "__main__":
    try:
        n = int(input("Enter a non-negative integer: "))
        print(f"{n}! = {factorial(n)}")
    except ValueError as e:
        print(f"Error: {e}")
```

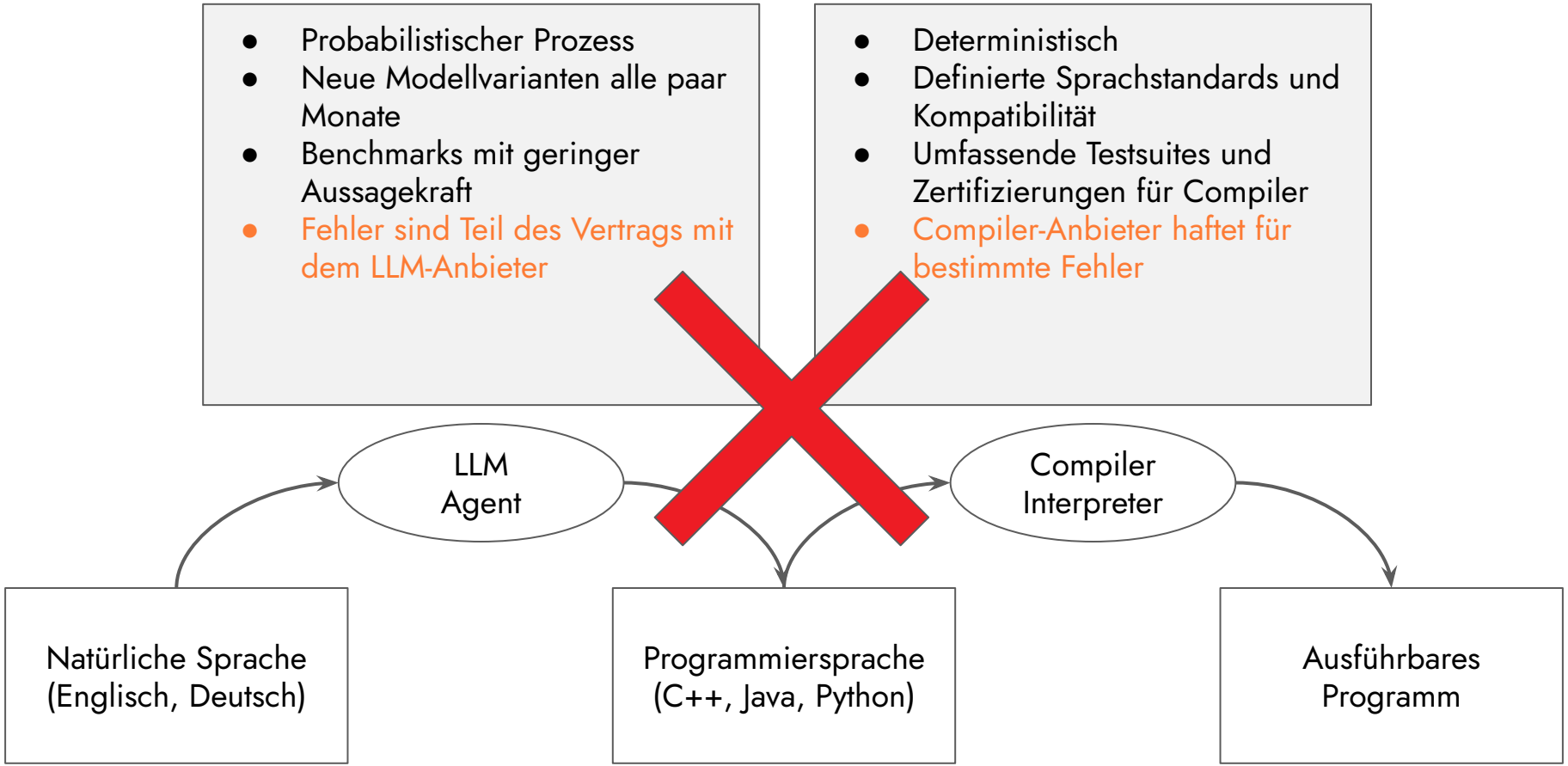
Reliance on Outputs and Actions. Artificial intelligence and large language models are frontier technologies that are still improving in accuracy, reliability and safety. When you use our Services, you acknowledge and agree:

1. Outputs may not always be accurate and may contain material inaccuracies even if they appear accurate because of their level of detail or specificity.
2. Actions may not be error free or operate as you intended.
3. You should not rely on any Outputs or Actions without independently confirming their accuracy.
4. The Services and any Outputs may not reflect correct, current, or complete information.
5. Outputs may contain content that is inconsistent with Anthropic's views.

https://www.anthropic.com/legal/consumer-terms

Fehlerfreiheit von Multi-Step-Agenten




Probabilistischer Prozess
Neue Modellvarianten alle paar Monate
Benchmarks mit geringer Aussagekraft
Fehler sind Teil des Vertrags mit dem LLM-Anbieter
Deterministisch
Definierte Sprachstandards und Kompatibilität
Umfassende Testsuites und Zertifizierungen für Compiler
Compiler-Anbieter haftet für bestimmte Fehler
LLM Agent
Compiler Interpreter
Natürliche Sprache (Englisch, Deutsch)
Programmiersprache (C++, Java, Python)
Ausführbares Programm

Wir produzieren massiv mehr Code

FARDS

AI ENGINEERING REPORT | 2026

THE ACCELERATION WHIPLASH

Engineering throughput is up. However, bugs, incidents, and rework are rising faster.

Analysis of two years of data from 22,000 developers on the real-world impact of AI adoption.



OUTPUT IS UP. NOT ALL OF IT STICKS.

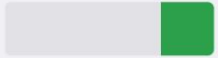
Impacts of high AI adoption on engineering throughput

Epic
A sizable feature or capability, hundreds of tasks



+66.2%
Epics completed per developer

Task
Individual work item, may span multiple PRs



+33.7%
Task throughput per developer

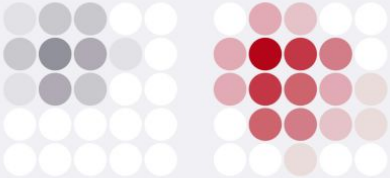
Within this, tasks with an associated PR grew +210% per team. Code-specific work accelerated far faster than overall task volume.

CODE IS GETTING LARGER AND MORE COMPLEX

Impacts of high AI adoption on code complexity

AI ADOPTION

LOW HIGH



+59.7%
files edited per PR

AI ADOPTION

LOW HIGH



+51.3%
average PR size

DEVELOPER FOOTPRINT/MONTH

+149.9%
files touched per developer

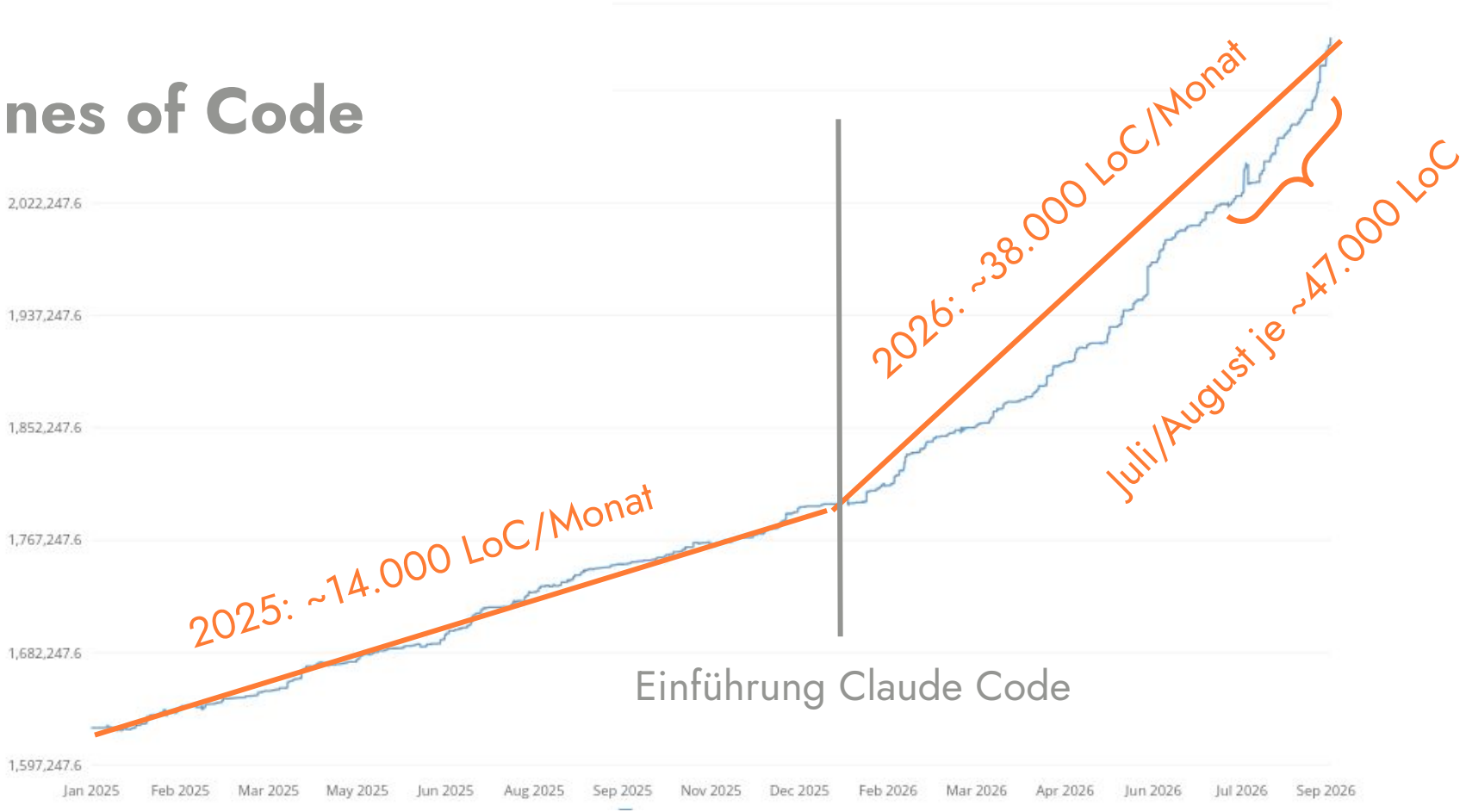
+11.7%
repositiories touched per developer

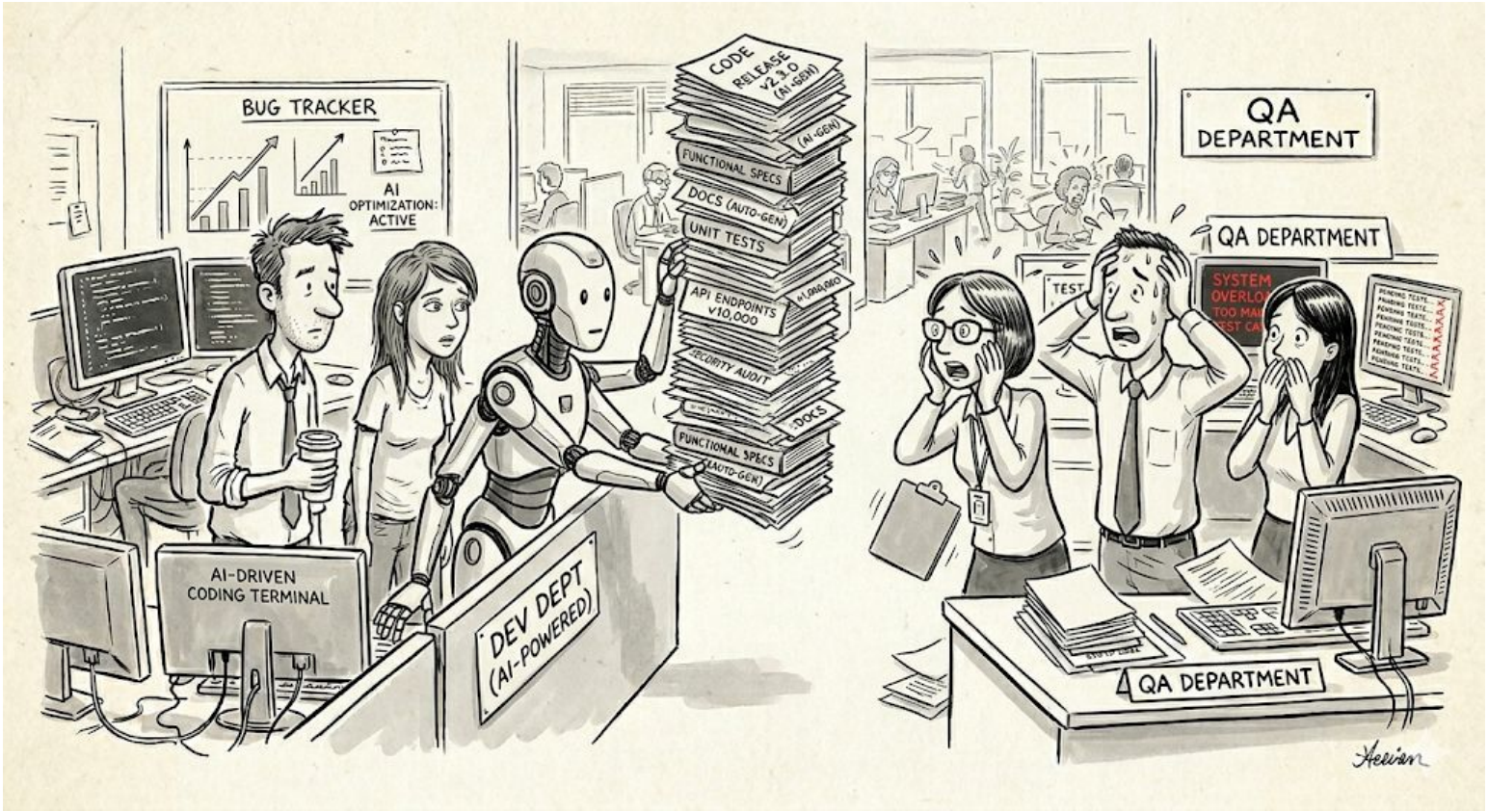
Telemetry dataset size: 22,000 developers across 4,000 teams

AI Engineering Report 2026 | The Acceleration Whiplash

FARDS

Lines of Code

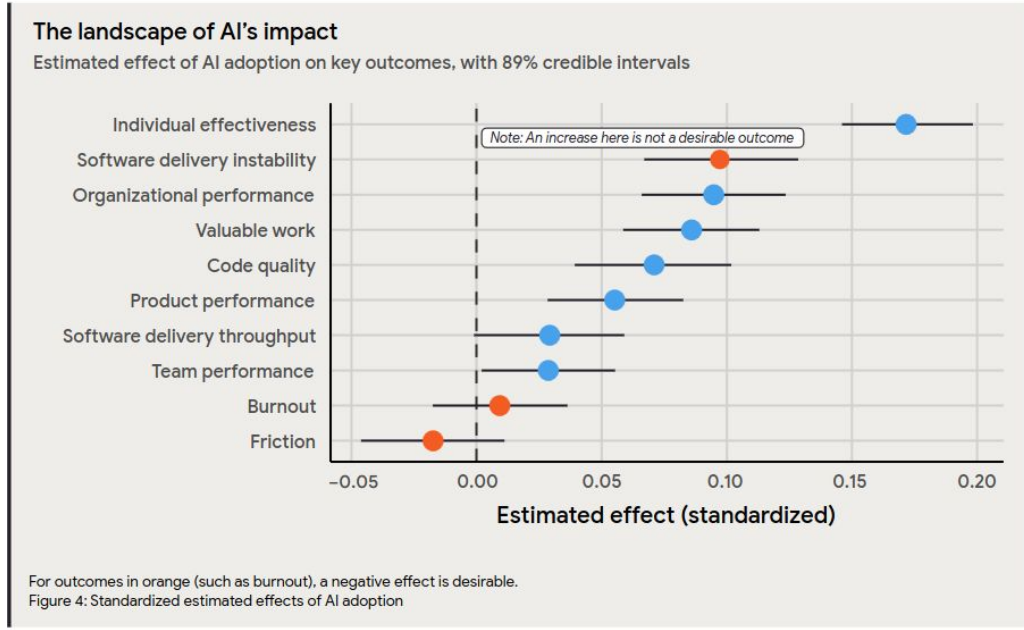



BUG TRACKER
AI OPTIMIZATION: ACTIVE
AI-DRIVEN CODING TERMINAL
DEV DEPT (AI-POWERED)
CODE RELEASE v2.3.0 (AI-GEN)
(AI-GEN)
FUNCTIONAL SPECS
DOCS (AUTO-GEN)
UNIT TESTS
API ENDPOINTS v10,000
SECURITY AUDIT
DOCS
FUNCTIONAL SPECS (AUTO-GEN)
QA DEPARTMENT
QA DEPARTMENT
QA DEPARTMENT

The ROI of AI-assisted Software Development

Google Cloud





For outcomes in orange (such as burnout), a negative effect is desirable.
Figure 4: Standardized estimated effects of AI adoption

Spielt Softwarequalität überhaupt noch eine Rolle?

Kundensichtbare Qualität bleibt wichtig

Correctness

Usability



Safety

Security

Accessibility

Reliability

Performance

Scalability

AI GENERATED

Was ist mit der **Codequalität**?

Correctness

Usability

Safety

Security



Accessibility

Reliability

Performance

Scalability

AI GENERATED

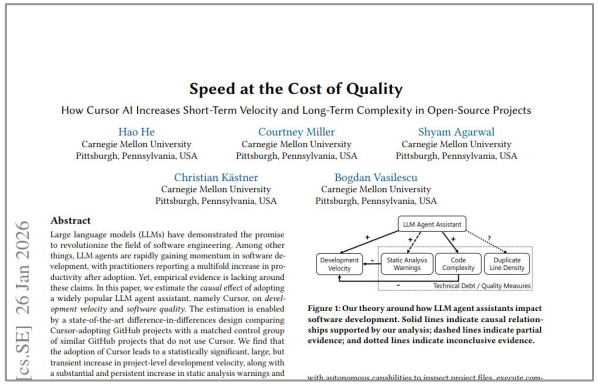
Was ist mit der **Codequalität**?

"KI schreibt schon heute besseren Code als Menschen."



"Der Code muss ja nur noch von KI verstanden werden, also spielt die Qualität keine Rolle."

Was wissen wir **heute** schon dazu?



Speed at the Cost of Quality

How Cursor AI Increases Short-Term Velocity and Long-Term Complexity in Open-Source Projects

Hao He, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
Courtney Miller, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
Shyam Agarwal, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
Christian Kästner, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
Bogdan Vasilescu, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

[cs.SE] 26 Jan 2026

Abstract

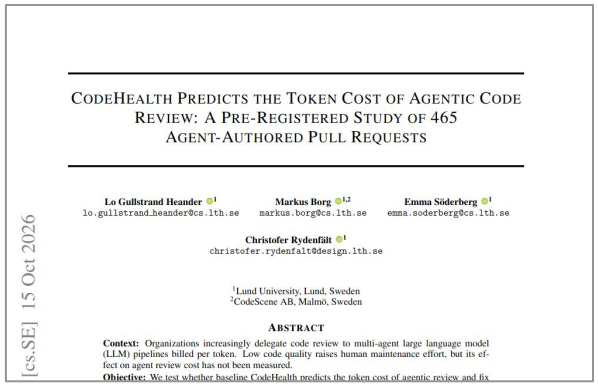
Large language models (LLMs) have demonstrated the promise to revolutionize the field of software engineering. Among other things, LLM agents are rapidly gaining momentum in software development, with practitioners reporting a multifold increase in productivity after adoption. Yet, empirical evidence is lacking around these claims. In this paper, we estimate the causal effect of adopting a widely popular LLM agent assistant, namely Cursor, on *development velocity* and *software quality*. The estimation is enabled by a state-of-the-art difference-in-differences design comparing Cursor-adopting GitHub projects with a matched control group of similar GitHub projects that do not use Cursor. We find that the adoption of Cursor leads to a statistically significant, large, but transient increase in project-level development velocity, along with a substantial and persistent increase in static analysis warnings and

Figure 1: Our theory around how LLM agent assistants impact software development. Solid lines indicate causal relationships supported by our analysis; dashed lines indicate partial evidence; and dotted lines indicate inconclusive evidence.

with autonomous capabilities to inspect project files, execute com-

Ungesteuerter Einsatz von KI-gestützter Entwicklung führt zu schlechterem Code (mehr Warnungen aus statischer Analyse, mehr Duplikation, höhere Komplexität).

He et at.: *Speed at the Cost of Quality.* MSR 2026



CodeHealth Predicts the Token Cost of Agentic Code Review: A Pre-Registered Study of 465 Agent-Authored Pull Requests

Lo Gullstrand Heander [1], lo.gullstrand_heander@cs.lth.se
Markus Borg [1,2], markus.borg@cs.lth.se
Emma Söderberg [1], emma.soderberg@cs.lth.se
Christofer Rydenfält [1], christofer.rydenfalt@design.lth.se

[1]Land University, Lund, Sweden
[2]CodeScene AB, Malmö, Sweden

[cs.SE] 15 Oct 2026

Abstract

Context: Organizations increasingly delegate code review to multi-agent large language model (LLM) pipelines billed per token. Low code quality raises human maintenance effort, but its effect on agent review cost has not been measured.
Objective: We test whether baseline CodeHealth predicts the token cost of agentic review and fix

"Schlechter" Code erhöht den Bedarf an Tokens (Kosten, Dauer) von Code Review durch Agenten.

Gullstrand Heander et at.:
CodeHealth Predicts the Token Cost of Agentic Code Review.

Was ist mit der **Codequalität**?

"KI schreibt schon heute besseren Code als Menschen."

"Der Code muss ja nur noch von KI verstanden werden, also spielt die Qualität keine Rolle."

?

"Wenn die Modelle noch besser werden, brauchen wir Code als Zwischenformat gar nicht mehr."

"Ich brauche eh nur noch die Spezifikation und KI kann mir daraus jederzeit die Software neu erstellen."

Wir haben (noch) keine **Langzeiterfahrungen**



Wie erreicht der Agent konsistent hohe Qualität

(auch um mit Nichtdeterminismus umgehen zu können)

Wie kann ich die Ausgabe**qualität** beeinflussen?

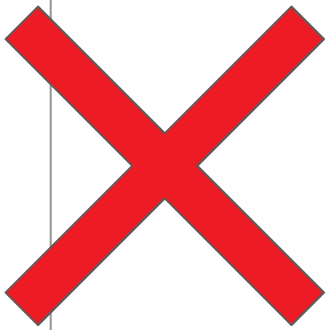
AGENTS.md

```
[…]

You are the best programmer in the world.

Write code of highest quality, technical
excellence and without bugs.

DO NOT HALLUCINATE. MAKE NO MISTAKES!!!
```



Wie kann ich die Ausgabe**qualität** beeinflussen?

AGENTS.md

```
[…]

Follow the following rules …

- Keep classes and methods short (max.
classes, max. 40 lines for methods)
- Make sure that every REST endpoint has a permissi
check and validates all inputs
- …
```

Das werden schnell sehr viele Regeln.
Und woher weiß ich, dass der Agent sich an die Regeln gehalten hat?

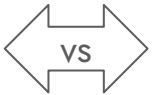
Wie machen wir es denn bei **Kompilierbarkeit**?

AGENTS.md

[…]

You know all code syntax by heart.

Make sure you generate only valid code that compiles without errors.



AGENTS.md

[…]

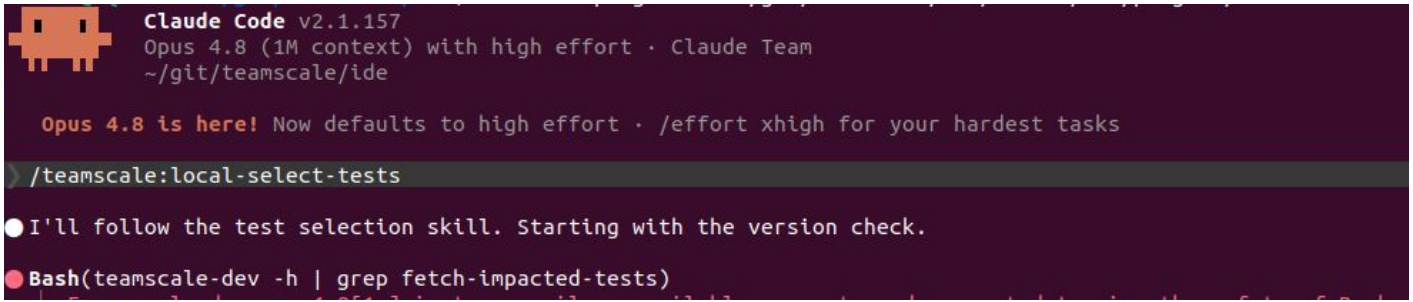
At the end of code generation, run the compiler and resolve all compilation errors and warnings.

Tools als Teil des Harness

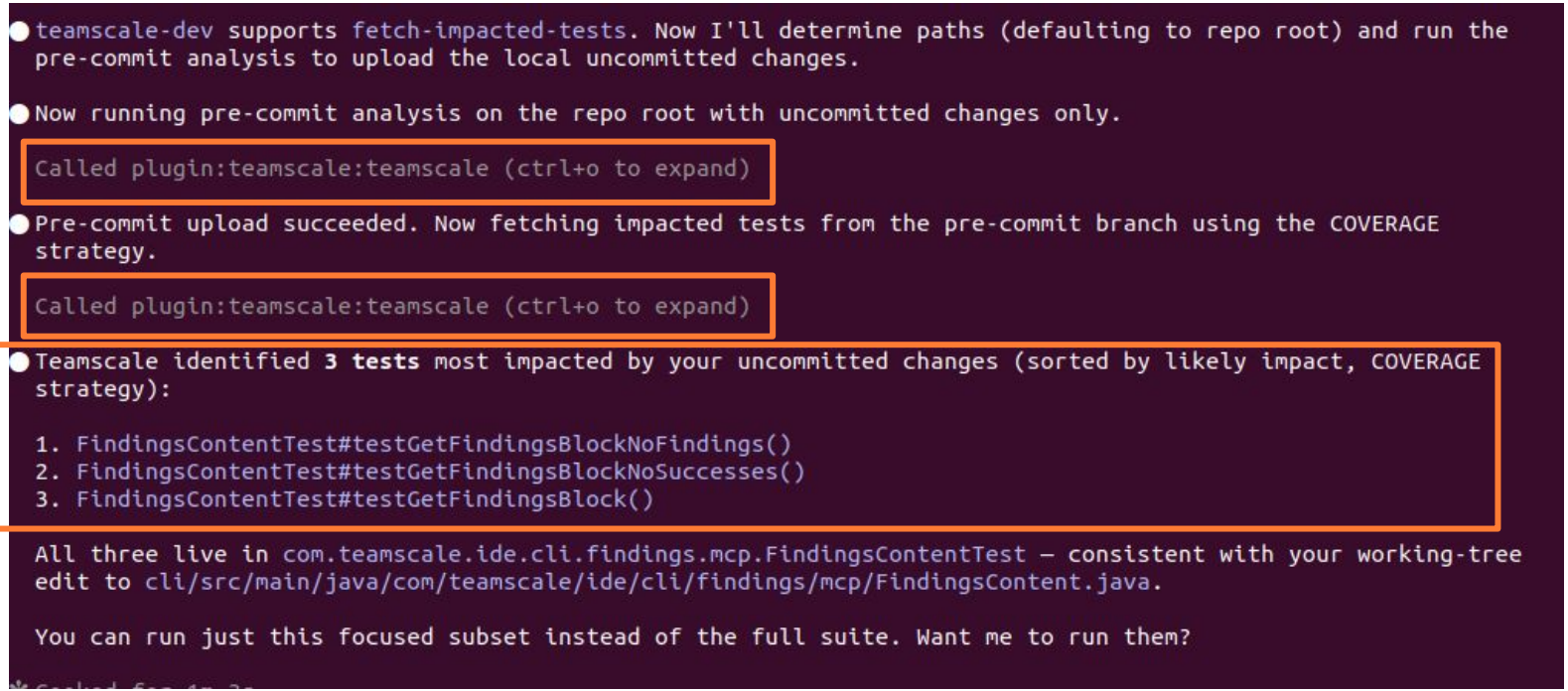
- Schnelle Ausführung, um Agenten nicht auszubremsen
- Deterministisch, damit Agenten sich nicht “im Kreis drehen”
- Konsistent, damit der Agent nicht Code generiert, der dann in der CI scheitert

Beispiele: Compiler, Linter, automatisierte Tests, etc.

Beispiel: Test-Selektion mit Claude Code



Beispiel: Test-Selektion mit Claude Code



Beispiel: Test-Selektion mit Claude Code

```
● Bash(./gradlew :cli:test --tests "com.teamscale.ide.cli.findings.mcp.FindingsContentTest" 2>&1 | tail -30)
  └ com.teamscale.ide.cli.findings.mcp.FindingsContentTest

      Test testGetFindingsBlock() FAILED
    … +30 lines (ctrl+o to expand)
  └ (timeout 10m)
  └ Allowed by auto mode classifier

● Teamscale's test selection was spot-on — all 3 impacted tests ran, and one failed, exposing a real problem in
  your uncommitted edit:

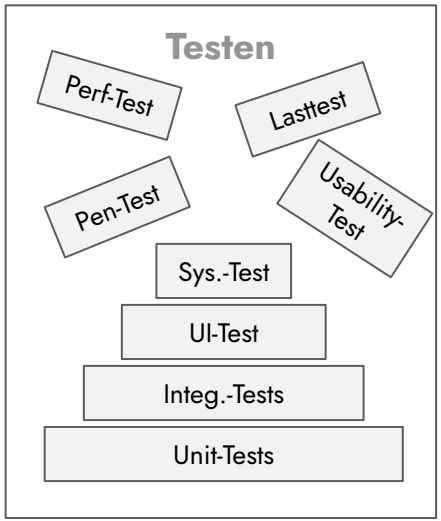
  | Test | Result |
  |---|---|
  | testGetFindingsBlockNoFindings() | ✅ PASSED |
  | testGetFindingsBlockNoSuccesses() | ✅ PASSED |
  | testGetFindingsBlock() | ❌ FAILED |

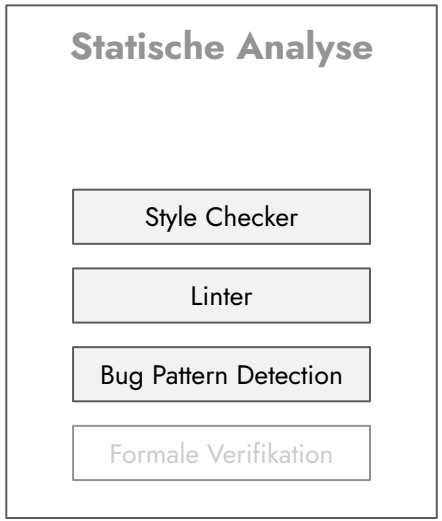
  The failure:
  java.util.MissingFormatArgumentException: Format specifier '%s'
      at FindingsContentTest.java:44
```

Welche Stellschrauben habe ich in der **Qualitätssicherung**?

(um mit viel mehr Code umgehen zu können)

Etablierte Verfahren in der Qualitätssicherung



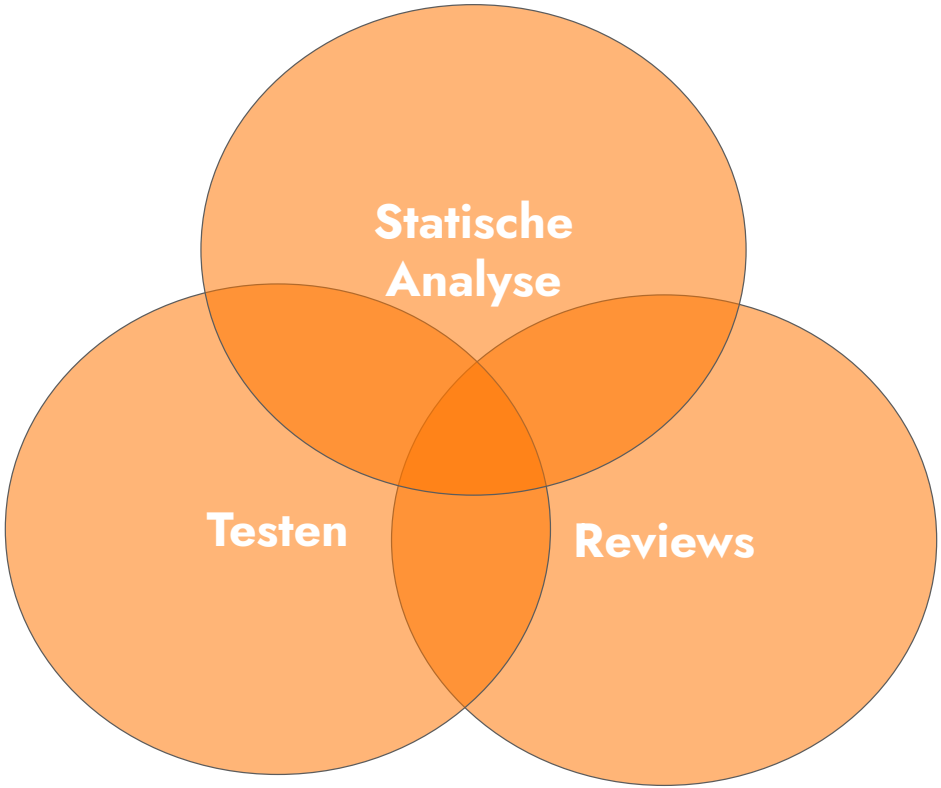




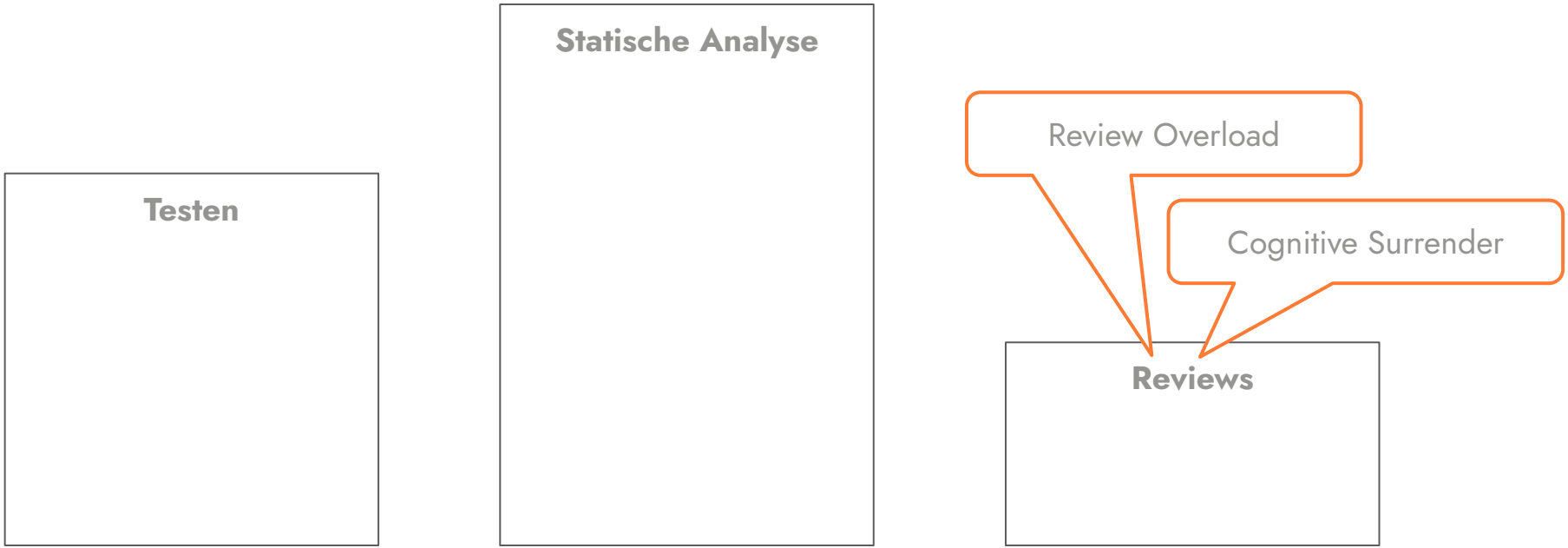
Etablierte Verfahren in der Qualitätssicherung



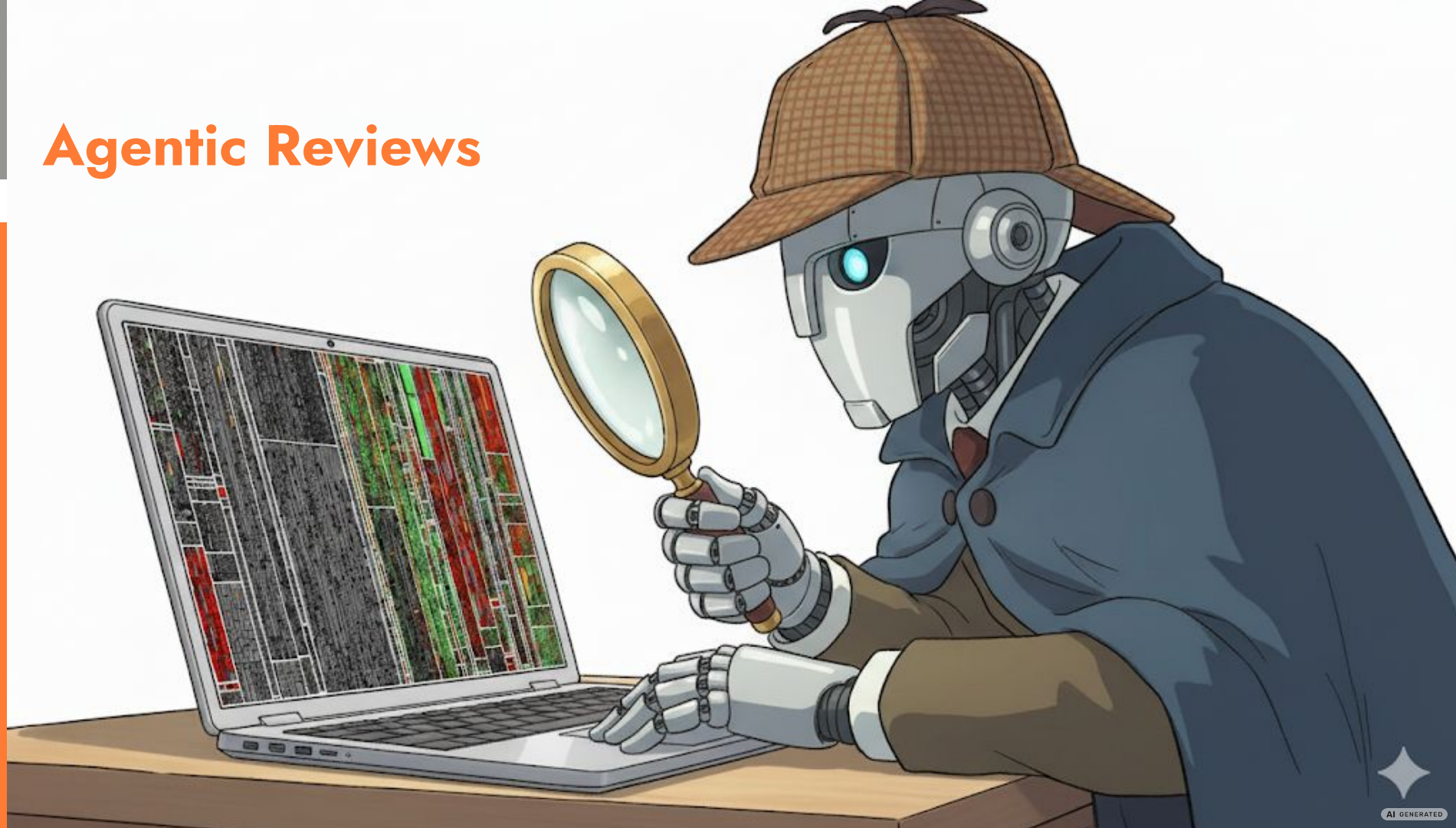
Alle Verfahren finden verschiedene Probleme



Skalierbarkeit



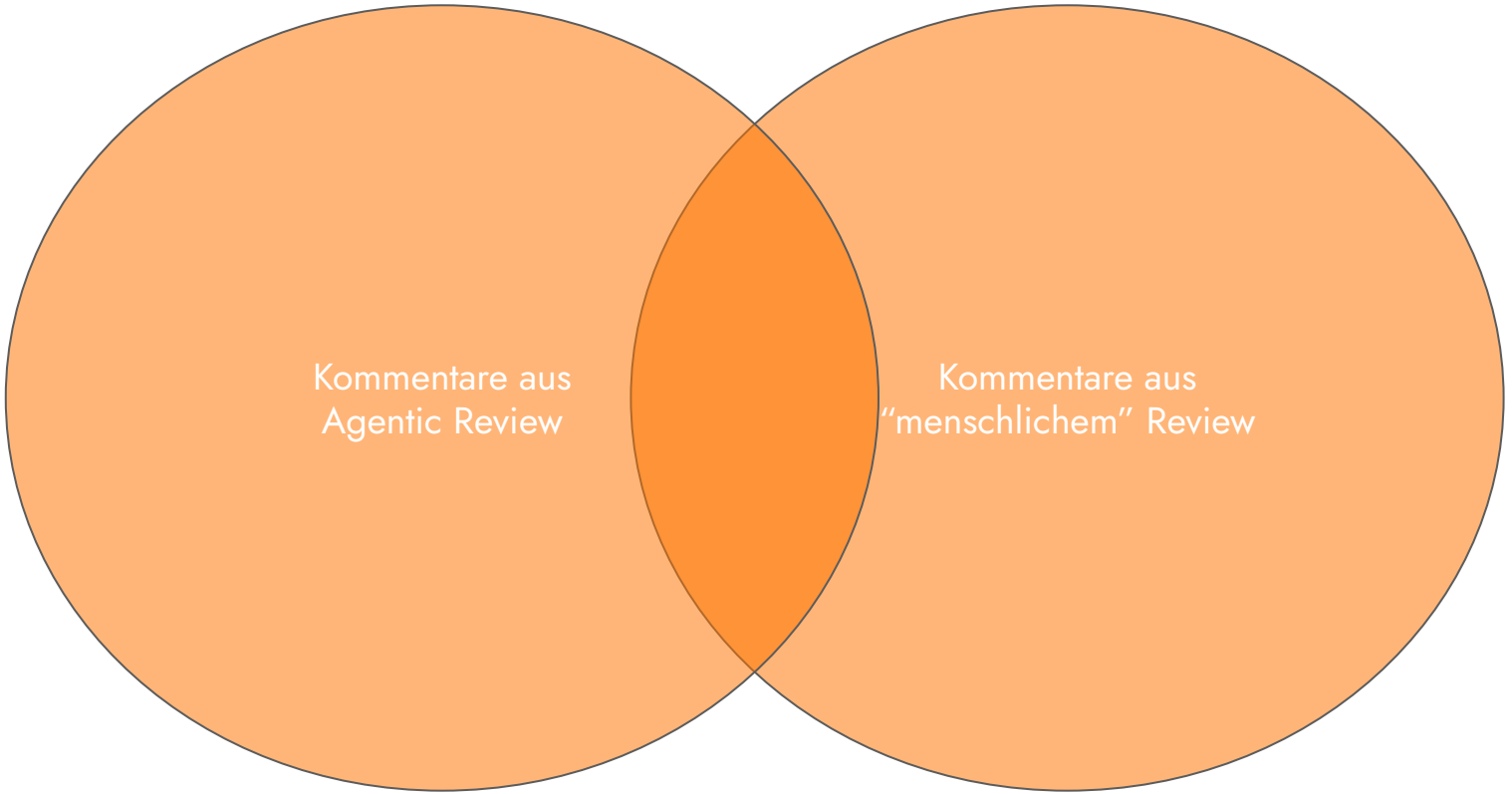
Agentic Reviews



Agentic Reviews als 4. Säule

Testen	Statische Analyse	Agentic Reviews	Reviews
		Annahmequote der Kommentare: 36% - 75%	Annahmequote der Kommentare: 65% - 93%

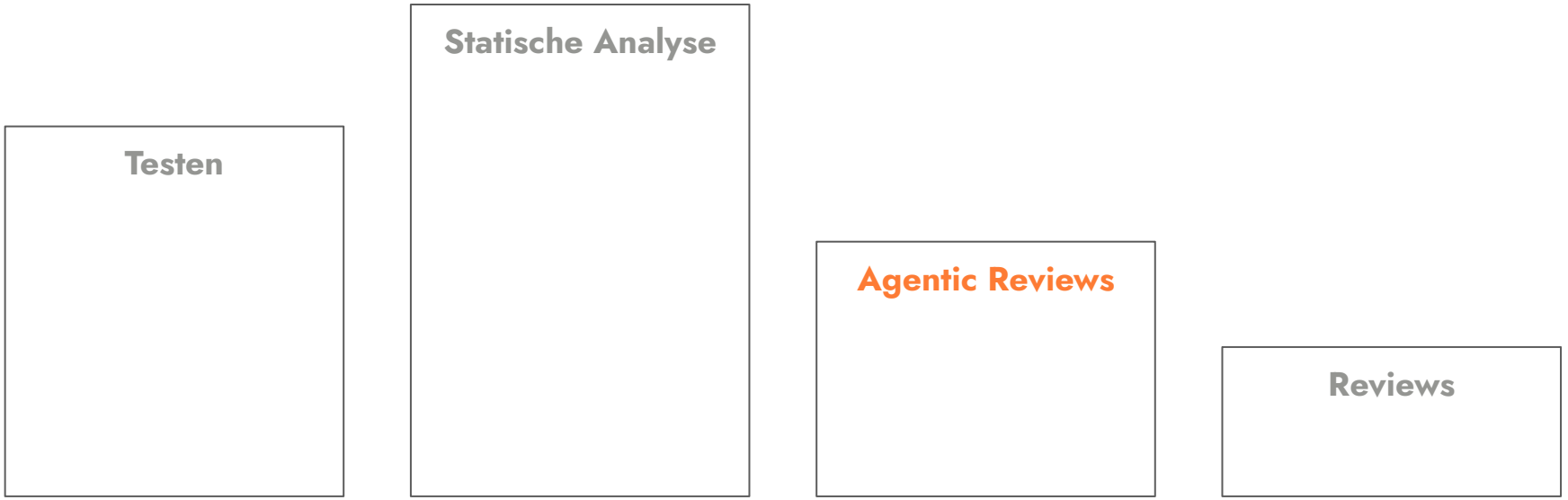
Beller et al.: Modern Code Reviews in Open-Source Projects: Which Problems Do They Fix? MSR 2014
Hong et al.: Is Agentic Code Review Helpful? Mining Developers' Feedback to CodeRabbit Reviews in the Wild. arXiv:2607.03316
Own experiments at CQSE/Teamscale


Kommentare aus
Agentic Review
Kommentare aus
“menschlichem” Review

Was bedeutet das für die **Praxis**?

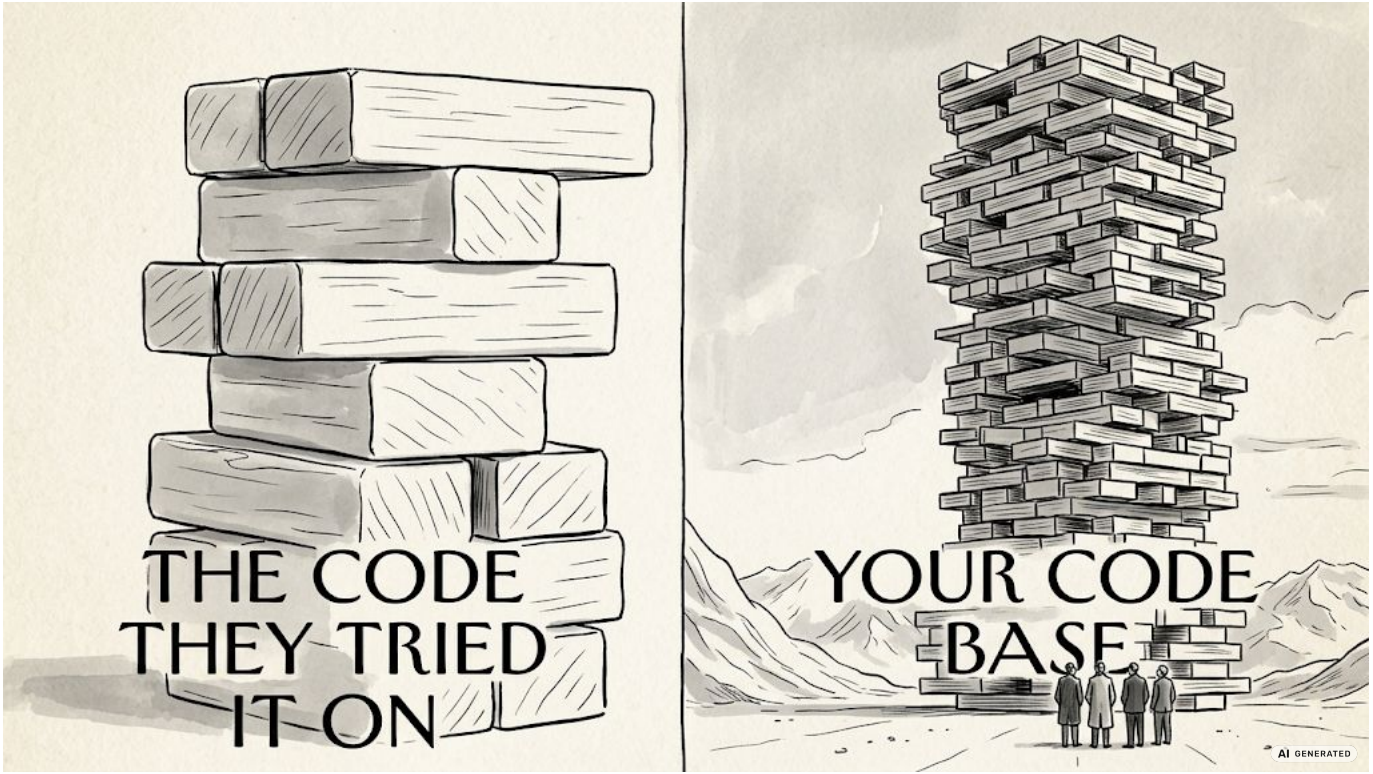
- Agentische Reviews haben eine schlechtere Annahmequote der Kommentare als “menschliche” Reviews
  ⇒ Bewertung durch Menschen bleibt wichtig
- Beide Ansätze finden Probleme, die der jeweils andere Ansatz nicht findet
  ⇒ Kein Ersatz, sondern eine Ergänzung
- Spezialisierte Review Agenten finden mehr relevante Kommentare als generische
  ⇒ Tailoring auf Projektumgebung und Benchmarking

Agentic Reviews als 4. Säule



Wie behalte ich die geistige Lufthoheit?

(weil die Verantwortung am Ende beim Menschen liegt)


THE CODE THEY TRIED IT ON
YOUR CODE BASE
AI GENERATED

Messen und Bewerten

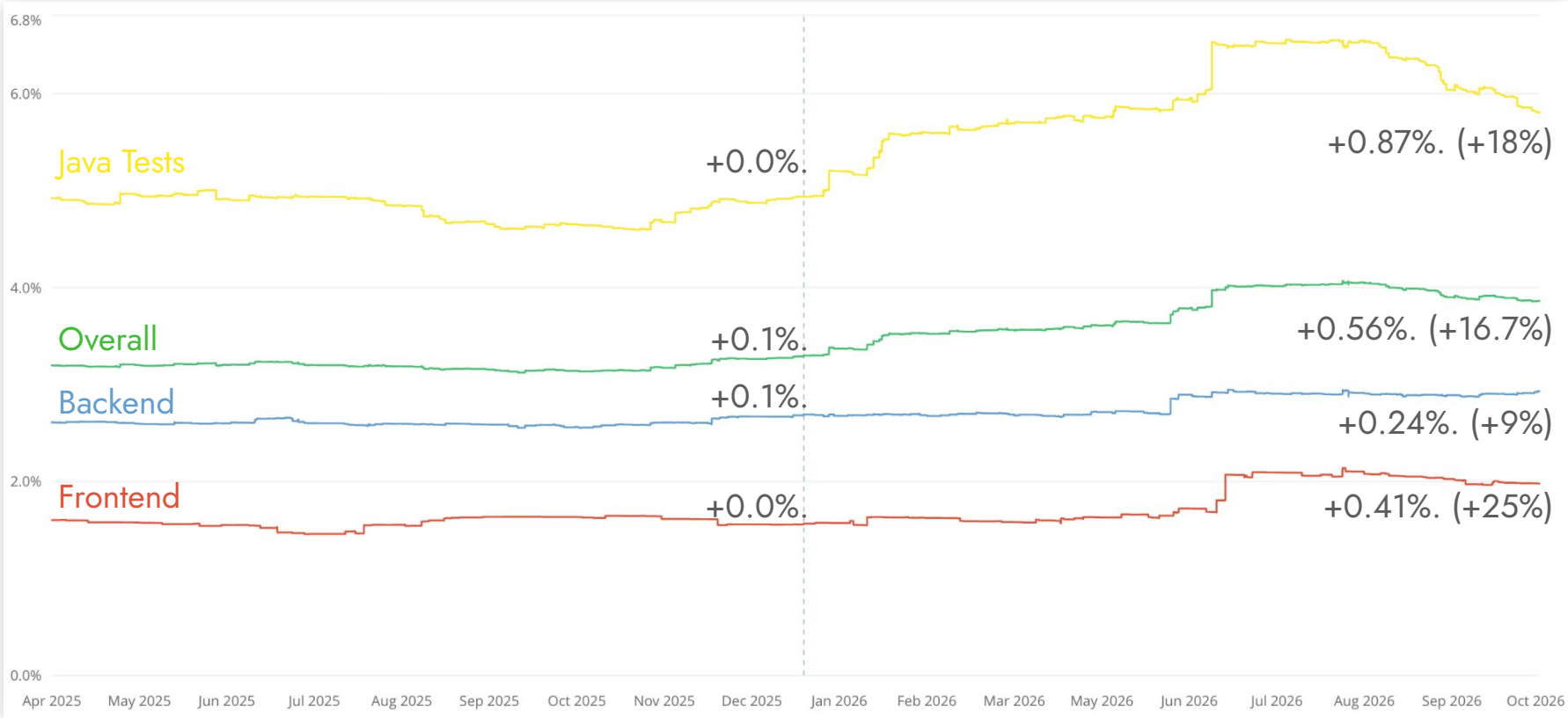
- Definition von Kernmetriken und Schwellwerten
- Beobachtung über die Zeit
- Drill-Down auf einzelne Bereiche oder Commits

Beispiel: Entwicklung der Metriken

Quality Indicator (QI) ▲	Value	Delta	Trend
Clone Coverage	✓ 3.9%	+0.56%	↘
File Size Assessment		+0.1% -0.5% +0.3%	↗
Findings Density	4.9	-0.14	↗
Method Length Assessment		-0.2% -0.6% +0.8%	↗
Nesting Depth Assessment		+0.0% -0.1% +0.1%	↗
Source Lines of Code	1.5M	+275612	→

Seit Januar 2026

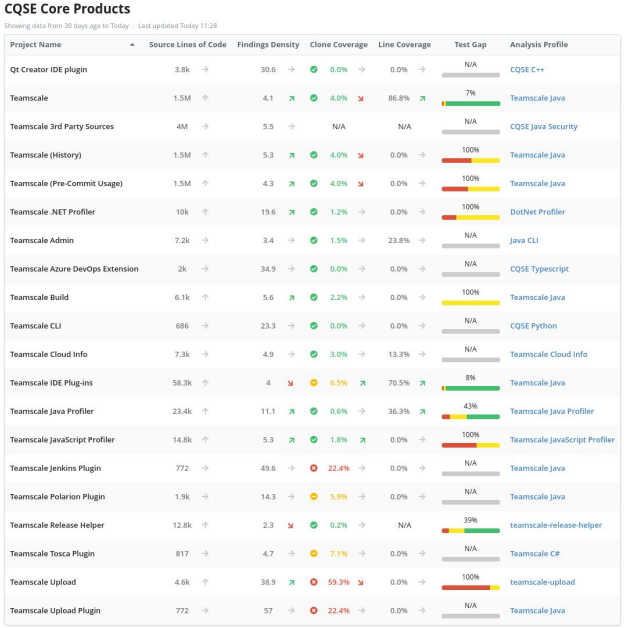
Beispiel: Clone Coverage




YOUR SOFTWARE LANDSCAPE

Software Intelligence

- Sammeln und Aggregieren von relevanten Daten aus dem Entwicklungsprozess
- Anpassbare automatische Analyse der Daten
- Zielgerichtete Bereitstellung von Analyseergebnissen (z.B. im Pull Request)
- Ableitung und Priorisierung von Maßnahmen auf Basis der Analyseergebnisse



CQSE Core Products

Showing data from 30 days ago to Today · Last updated Today 11:28

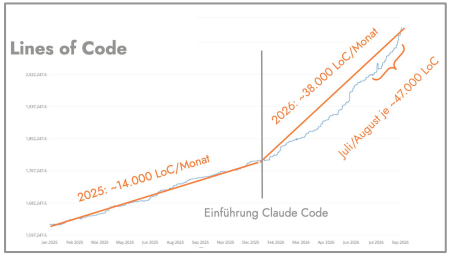
Project Name	Source Lines of Code	Findings Density	Clone Coverage	Line Coverage	Test Gap	Analysis Profile
Qt Creator IDE plugin	3.8k	30.6	0.0%	0.0%	N/A	CQSE C++
Teamscale	1.5M	4.1	4.0%	86.8%	7%	Teamscale Java
Teamscale 3rd Party Sources	4M	5.5	N/A	N/A	N/A	CQSE Java Security
Teamscale (History)	1.5M	5.3	4.0%	0.0%	100%	Teamscale Java
Teamscale (Pre-Commit Usage)	1.5M	4.3	4.0%	0.0%	100%	Teamscale Java
Teamscale .NET Profiler	10k	19.6	1.2%	0.0%	100%	DotNet Profiler
Teamscale Admin	7.2k	3.4	1.5%	23.8%	N/A	Java CLI
Teamscale Azure DevOps Extension	2k	34.9	0.0%	0.0%	N/A	CQSE Typescript
Teamscale Build	6.1k	5.6	2.2%	0.0%	100%	Teamscale Java
Teamscale CLI	686	23.3	0.0%	0.0%	N/A	CQSE Python
Teamscale Cloud Info	7.3k	4.9	3.0%	13.3%	N/A	Teamscale Cloud Info
Teamscale IDE Plug-ins	58.3k	4	6.5%	70.5%	8%	Teamscale Java
Teamscale Java Profiler	23.4k	11.1	0.6%	36.3%	43%	Teamscale Java Profiler
Teamscale JavaScript Profiler	14.8k	5.3	1.8%	0.0%	100%	Teamscale JavaScript Profiler
Teamscale Jenkins Plugin	772	49.6	22.4%	0.0%	N/A	Teamscale Java
Teamscale Polarion Plugin	1.9k	14.3	5.9%	0.0%	N/A	Teamscale Java
Teamscale Release Helper	12.8k	2.3	0.2%	N/A	39%	teamscale-release-helper
Teamscale Tosca Plugin	817	4.7	7.1%	0.0%	N/A	Teamscale C#
Teamscale Upload	4.6k	38.9	59.3%	0.0%	100%	teamscale-upload
Teamscale Upload Plugin	772	57	22.4%	0.0%	N/A	Teamscale Java

Zusammenfassung

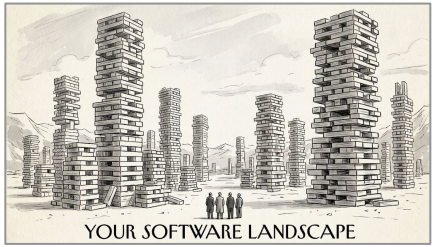
Qualität des Agenten-Codes



Überlastung der QA



Geistige Lufthoheit trotz KI



Integration in den Harness

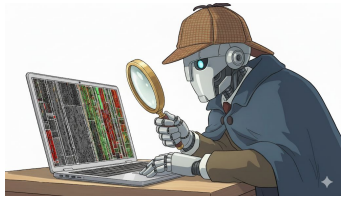
```
Bash(./gradlew :cli:test --tests "com.teamscale.ide.cli.findings.mcp.FindingsContentTest" 2>&1 | tail -30)
  com.teamscale.ide.cli.findings.mcp.FindingsContentTest
    Test testGetFindingsBlock() FAILED
  (timeout 10m)
  Allowed by auto mode classifier

Teamscale's test selection was spot-on — all 3 impacted tests ran, and one failed, exposing a real problem in
your uncommitted edit:
```

Test	Result
testGetFindingsBlockNoFindings()	✅ PASSED
testGetFindingsBlockNoSuccesses()	✅ PASSED
testGetFindingsBlock()	❌ FAILED

```
The failure:
java.util.MissingFormatArgumentException: Format specifier '%s'
    at FindingsContentTest.java:44
```

Agentic Reviews als Ergänzung



Software Intelligence

Quality Indicator (QI)	Value	Delta Trend
Clone Coverage	3.9%	+0.56%
File Size Assessment		
Findings Density	4.9	-0.14
Method Length Assessment		
Nesting Depth Assessment		
Source Lines of Code	1.5M	+275612

Teamscale

Integration in den Harness

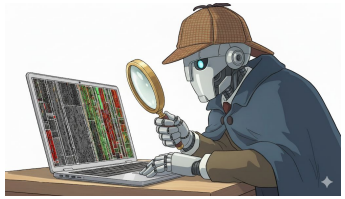
```
Bash(./gradlew :cli:test --tests "com.teamscale.ide.cli.findings.mcp.FindingsContentTest" 2>&1 | tail -30)
    com.teamscale.ide.cli.findings.mcp.FindingsContentTest
        Test testGetFindingsBlock() FAILED
    (timeout 10m)
    Allowed by auto mode classifier

● Teamscale's test selection was spot-on — all 3 impacted tests ran, and one failed, exposing a real problem in
  your uncommitted edit:

| Test | Result |
|---|---|
| testGetFindingsBlockNoFindings() | ✅ PASSED |
| testGetFindingsBlockNoSuccesses() | ✅ PASSED |
| testGetFindingsBlock() | ❌ FAILED |

The failure:
java.util.MissingFormatArgumentException: Format specifier '%s'
    at FindingsContentTest.java:44
```

Agentic Reviews als Ergänzung



Software Intelligence

Quality Indicator (QI)	Value	Delta	Trend
Clone Coverage	3.9%	+0.56%	↘
File Size Assessment		-0.1% -0.5% +0.3%	↗
Findings Density	4.9	-0.14	↗
Method Length Assessment		-0.2% -0.6% +0.8%	↗
Nesting Depth Assessment		+0.6% -0.1% +0.1%	↗
Source Lines of Code	1.5M	+275612	↑

Teamscale by CQSE

Vielen Dank für die Aufmerksamkeit!

Teamscale by CQSE

Test Gap Analysis

»We ship too many defects, and AI is making it worse!«

Wednesday
December 2, 2026
10:30 AM to 12:00 PM CET
online and free – with recording

Register now: tmscl.me/tga2612-sit2611





Ramona Kühn

Fabian Streitel

Teamscale by CQSE

Raus aus dem Elfenbeinturm

Wie Teams zu sinnvollen Coding-Guidelines kommen, die alle anwenden können

Mittwoch
17. Februar, 2027
10:30 bis 12:00 Uhr
online und kostenlos – mit Aufzeichnung

Jetzt anmelden: tmscl.me/sit272-sit2611





Maik Heene
parcIT

Johannes Veihelmann
Teamscale