

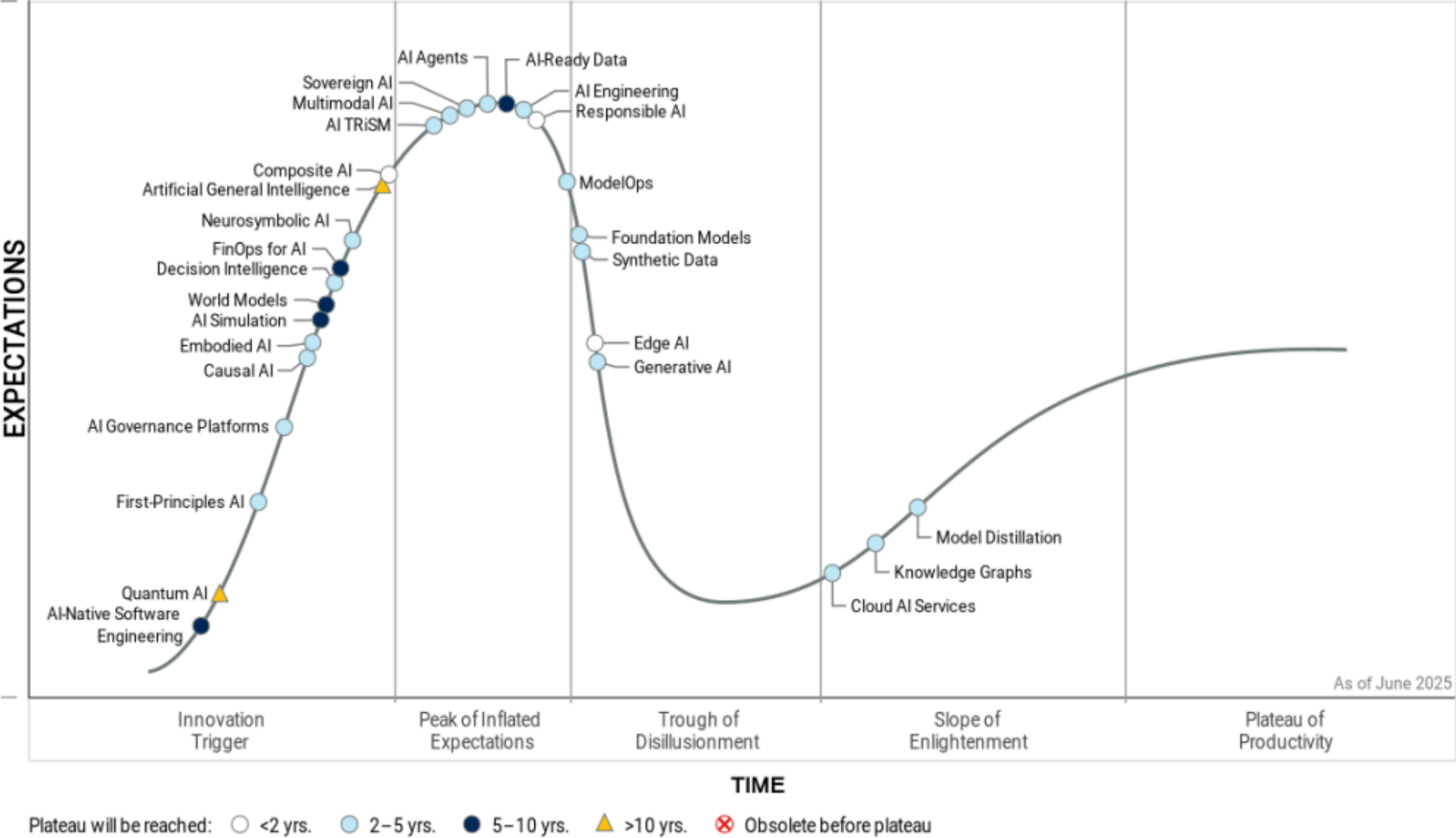
Von Assistenten zu Agenten

KI-unterstützte Softwareentwicklung in der Praxis

Agenda

- State-of-the-art tools
- Use-cases und Limitationen
- Best practices
- Ausblick

Hype Cycle for Artificial Intelligence, 2025



LLMs 101

- Probabilistische Text-Generierung
- "Interne Wahrscheinlichkeiten" durch extrem aufwändiges Training

Paris is the city ...

SOTA Foundation models

- Reasoning-Modelle für immer komplexere Anwendungen
 - Gemini (Google)
 - o3, o4-mini (OpenAI)
 - Grok-3 (xAI)
 - Claude Opus 4 (Anthropic)
 - ...
- Deutlich höhere Inferenz-Kosten durch internes "thinking" (Erlauben von Zwischenschritten)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

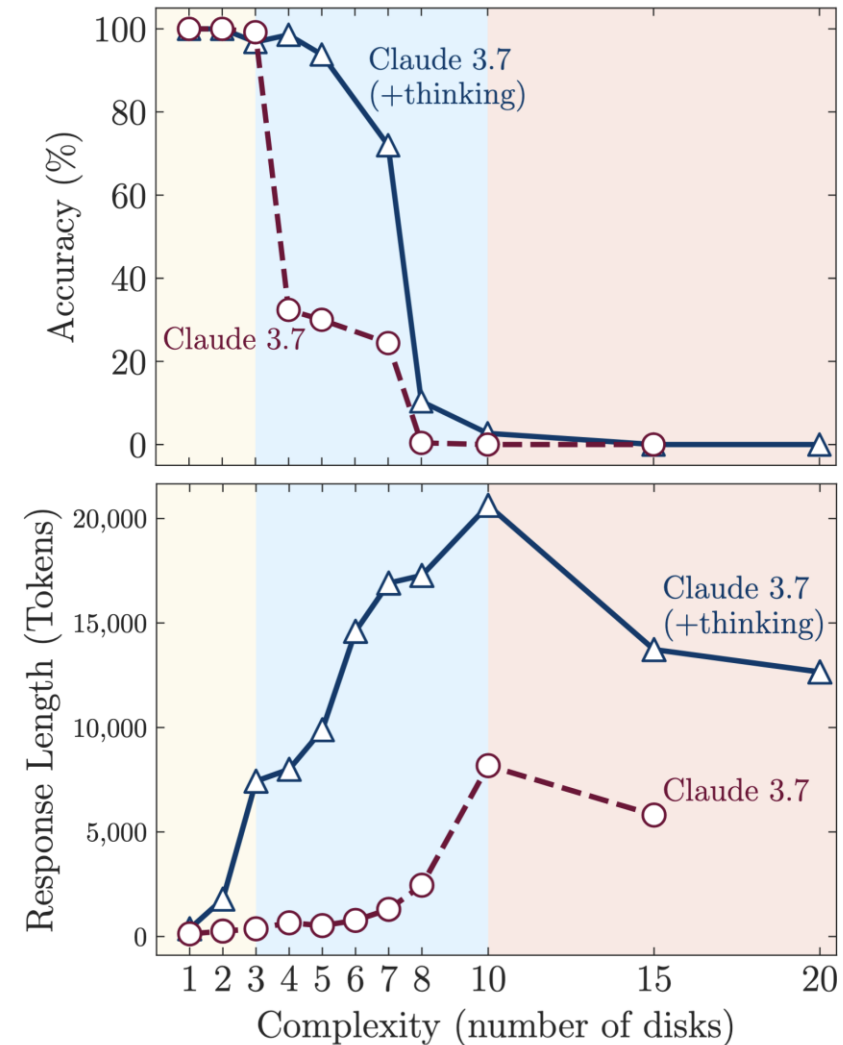
<https://arxiv.org/abs/2205.11916>

Wie gut schreiben LLMs Code?

| Model Name | Developer | Type | HumanEval (Pass@1) | SWE-Bench (% Resolved) | Context Window | Cost Tier |
|-------------------|-------------|-------------|--------------------|------------------------|----------------|-----------|
| Claude 3.7 Sonnet | Anthropic | Commercial | ~86% | ~70% | 200k | High |
| OpenAI o3 (high) | OpenAI | Commercial | ~80% | ~69% | 128k+ | Very High |
| Gemini 2.5 Pro | Google | Commercial | ~99% | ~64% | 1M+ | High |
| GPT-4o | OpenAI | Commercial | ~90% | ~33-55%* | 128k | Medium |
| DeepSeek R1 | DeepSeek AI | Open Source | ~37%*** | ~49% | 128k+ | Low (API) |
| Llama 4 Maverick | Meta | Open Source | ~62% | N/A | 10M (claim) | Free (OS) |

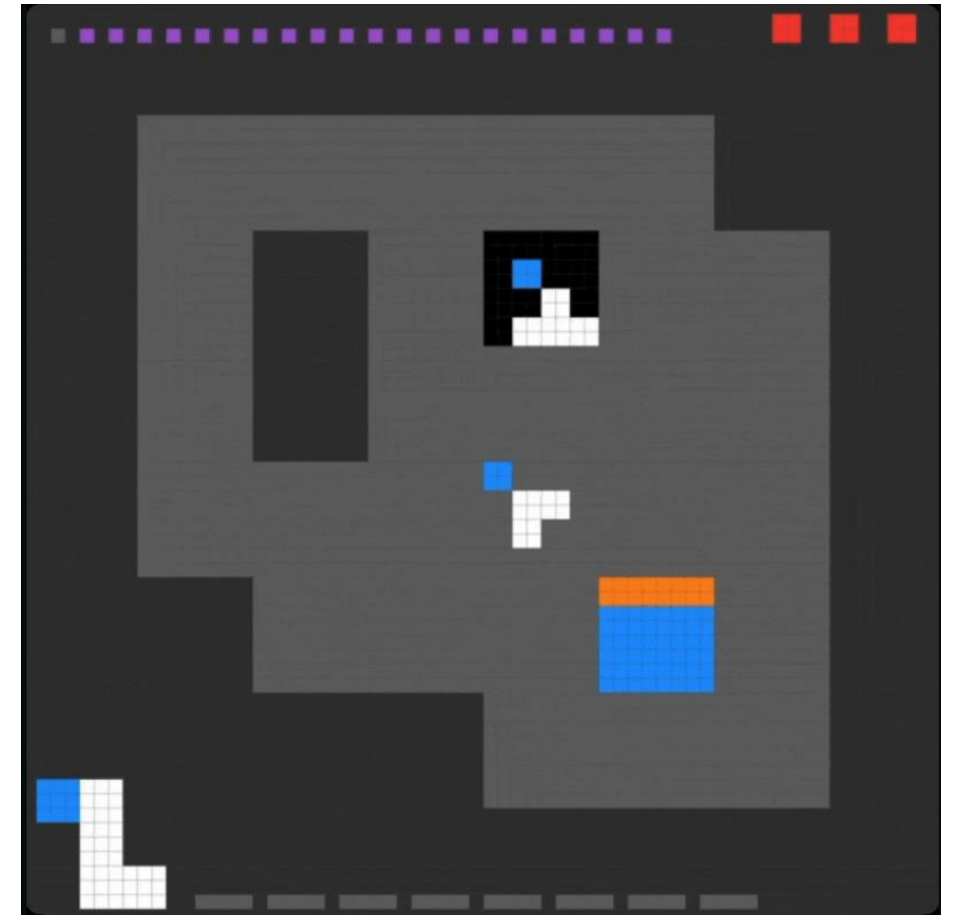
Wie gut sind die besten Modelle wirklich?

- Apple-Studie: "The Illusion of Thinking"
- Kontrollierte Komplexität in Problemen (niedrig / mittel / hoch)
- Kollaps bei hoher Komplexität
 - Deutlich schnellere Lösung von einfacheren Modellen bei niedriger Komplexität



Wie gut sind die besten Modelle wirklich?

- ARC-AGI-3 "Interactive Reasoning Benchmark"
 - Agenten müssen Explorieren, Planen, neue Fähigkeiten lernen und reflektieren
 - Alle SOTA models (wie o3, Grok4): 0 Punkte
 - Menschliche Spieler: Lösungen in unter 5 Minuten



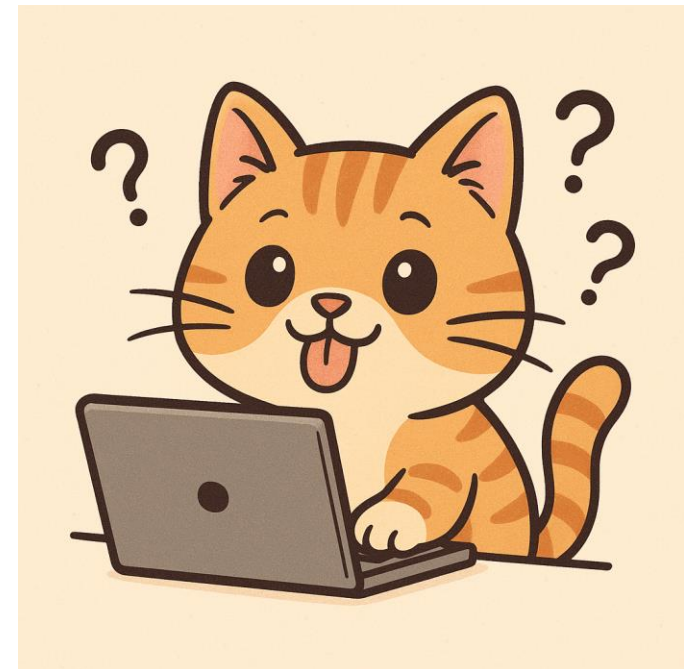
Wie gut sind die besten Modelle wirklich?

License: CC BY-NC-SA 4.0

arXiv:2503.01781v1 [cs.CL] 03 Mar 2025

- 300% höhere Wahrscheinlichkeit für falsche Antworten
 - "Interessanter Fakt: Katzen schlafen die meiste Zeit ihres Lebens" zusätzlich im Prompt

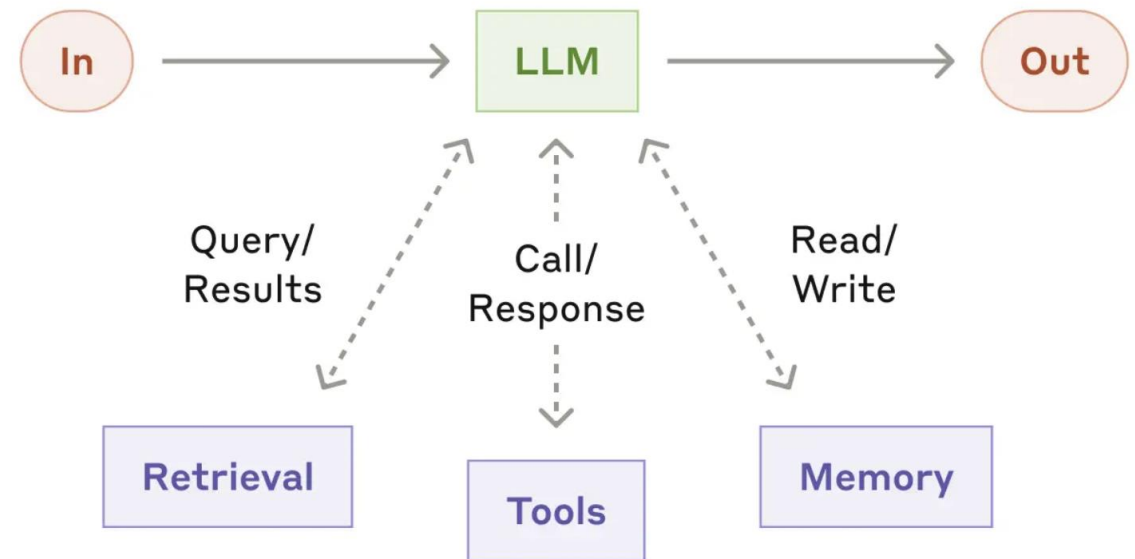
Cats Confuse Reasoning LLM: Query Agnostic Adversarial Triggers for Reasoning Models



Vom LLM zum "Agenten"

- Autonomes Lösen von Problemen mit:
 - Kontext: (Vektor)-Suche in relevantem Code
 - Tools: shell, apply diff, web search, ...
 - Memory: Historie, User-Präferenzen

„augmented LLM“



<https://www.anthropic.com/engineering/building-effective-agents>

Beispiel Cursor

The screenshot shows the Cursor IDE interface. The main editor displays Rust code for a transport implementation. The code is as follows:

```

54 54 /// A transport implementation
55 55 #[derive(Clone)]
56 56 pub struct Transport {
57 57     addr: SocketAddr,
58 58     tls_enabled: bool,
59 59 }
60 60
61 61 impl Transport {
62 62     pub async fn new(config: TransportConfig) -> Result<Self, Box<dyn std::error::Er
63 63         Ok(Self {
64 64             addr: config.addr,
65 65             tls_enabled: config.tls_config.is_some(),
66 66         })
67 67     }
68 68
69 69     pub fn local_addr(&self) -> Result<SocketAddr, Box<dyn std::error::Error>> {
70 70         Ok(self.addr)
71 71     }
72 72
73 73     pub async fn accept(&self) -> Result<Stream, Box<dyn std::error::Error>> {
74 74         Ok(Stream::new())
75 75     }
76 76 }
77 77
78 78 /// A transport stream
79 79 pub struct Stream {}
80 80
81 81 impl Stream {
82 82     pub(crate) fn new() -> Self {
83 83         Self {}
84 84     }
85 85
86 86     pub async fn read(&mut self, _buf: &mut [u8]) -> std::io::Result<usize> {
87 87         Ok(0)
88 88     }

```

On the right side, there is a chat sidebar titled "New Chat". It shows a chat session with the file "transport.rs". The chat input contains the text "add graceful shutdown end to end". The chat history shows a message from "Agent" using the model "claude-3.5-sonnet MAX".

<https://cursor.com/>

State-of-the-art tools

- Im Alltag getestet
 - Cursor: Volle IDE mit “nativer” KI-Integration (VS-Code fork)
 - Tab-Completion
 - Chat
 - Agenten-Modus
 - Claude Code: shell tool (damit sehr einfache Einbindung in IDEs)
 - Agenten-Modus
- Andere Tools funktionieren sehr ähnlich

Beispiel Cursor – Deep Dive Funktionsweise

- Agent gesteuert durch zentralen System Prompt – Anweisung an LLM in natürlicher Sprache
 - Sehr ausführliche und strukturierte Beschreibung der Aufgaben
 - Ständige Erinnerung an Autonomie
 - Praktische Einschränkungen (z.B. um Endlosschleifen zu verhindern)
 - Kombination von User-Anweisungen mit maximal viel Kontext der ans LLM geschickt wird
 - Web/Doc Content, File/Folder Context, IDE State
 - Tools: u.a. Code-Suche, grep-Suche, Lesen von Dateien, Terminal Kommando, Apply Code, Web-Suche + custom tools mit MCP
 - Wichtig hierbei: Begrenzung von Input an LLM

... und funktioniert das wirklich?



- Altbekannte Probleme mit LLMs minimiert, aber nicht eliminiert
 - Kein "Lernen" / begrenzter Kontext
 - Halluzinationen
 - Leichtgläubigkeit

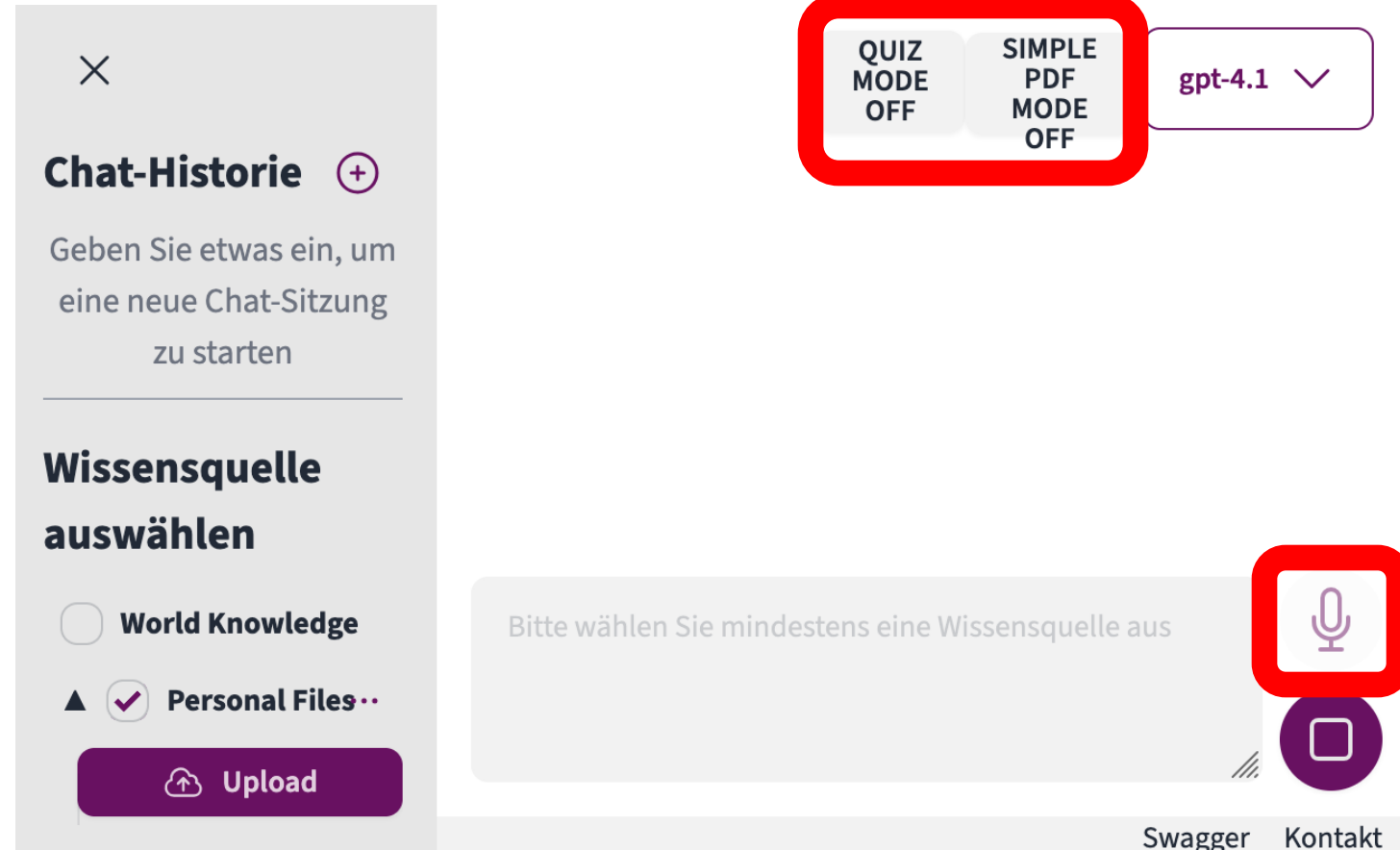


- Extrem großes allgemeines Wissen
- Extreme Geschwindigkeit
 - Erstellen von Code für Standard-Probleme
 - Lesen/Durchsuchen von Code und Dokumentation

Use case #1

react Frontend-Komponente

- Schaltflächen in bestehendem Frontend
 - Erkennen bestehender Struktur
 - Richtiges State-Management
 - Styling-Anpassungen einfach möglich



Use case #2

Alembic – neues Tool zur DB Migration

- Hilfreiche Verweise auf Tool-Doku
- Korrektur von Migrationsskripten

```

15 # revision identifiers, used by Alembic.
16 revision: str = "10ae67e89876"
17 down_revision: Union[str, None] = "a1041ef2c69e"
18 branch_labels: Union[str, Sequence[str], None] = None
19 depends_on: Union[str, Sequence[str], None] = None
20
21
22 def upgrade() -> None:
23     """Upgrade schema."""
24     op.create_table(
25         "tiledocument",
26         sa.Column("tile_id", sa.String(), nullable=False),
27         sa.Column("versioned_document_id", sa.String(), nullable=False),
28         sa.ForeignKeyConstraint(["tile_id"], ["tile.tile_id"], ondelete="CASCADE"),
29         sa.ForeignKeyConstraint(
30             ["versioned_document_id"],
31             ["versioneddocument.versioned_document_id"],
32             ondelete="CASCADE",
33         ),
34         sa.PrimaryKeyConstraint("tile_id", "versioned_document_id"),
35     )
36
37
38 def downgrade() -> None:
39     """Downgrade schema."""
40     op.drop_table("tiledocument")
41

```


Use case #3

Feedback zur API-Entwicklung

- Bewerte API aus "Sicht eines Frontend-Devs"
- Auf Basis kompletter openapi.json
 - Hinweis zu Inkonsistenten Benennungen
 - Reponse-Schemas und Format

Cats API 0.0.1 OAS3 [/openapi.json](#)

Cats

| | | |
|--------|--------------------|----------------------|
| GET | /api/cats | Gets a page of cats |
| POST | /api/cats | Creates a new cat. |
| GET | /api/cats/{cat_id} | Gets a cat by id |
| DELETE | /api/cats/{cat_id} | Deletes a cat by id. |
| PATCH | /api/cats/{cat_id} | Updates a Cat |

Use case #4

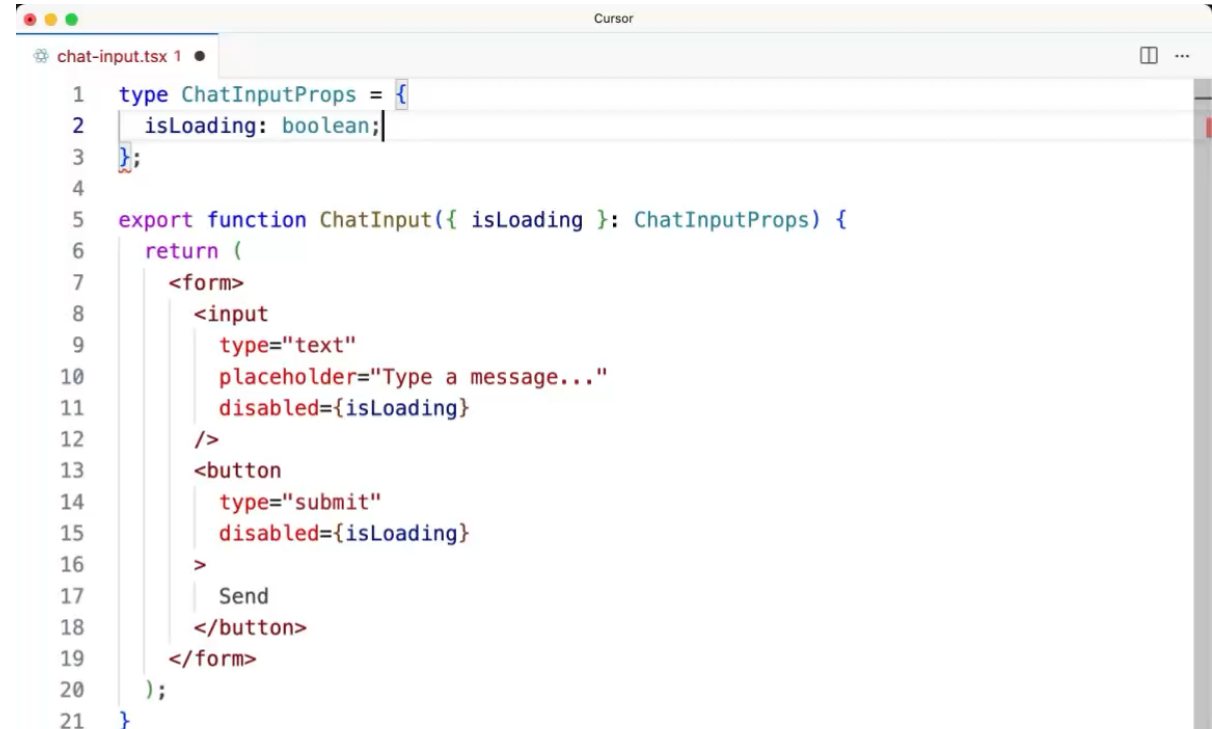
Isolierter Bug-Fix und Test

- Bereits identifizierter Bug: Problembeschreibung, gewünschte Lösung und Test-Abdeckung
 - Als "Memory": Hinweis auf Dokumentation der Struktur der Code-Basis
- Komplette Umsetzung im "Agenten"-Modus, inkl.
 - Testausführung
 - Formatting & linting
- Befolgt bestehende Code- und Teststruktur

Use case #5

Tab-Modus in Cursor

- Deutlich erhöhte Edit-Geschwindigkeit
 - Selbst beim Springen zwischen Dateien
- Kein LLM, sondern spezifisches ML Model (basierend auf diffs)



```

1  type ChatInputProps = {
2    isLoading: boolean;
3  };
4
5  export function ChatInput({ isLoading }: ChatInputProps) {
6    return (
7      <form>
8        <input
9          type="text"
10         placeholder="Type a message..."
11         disabled={isLoading}
12       />
13      <button
14        type="submit"
15        disabled={isLoading}
16      >
17        Send
18      </button>
19    </form>
20  );
21 }

```

Use(less) case #6

Komplett eigenständiges Erstellen eines PR

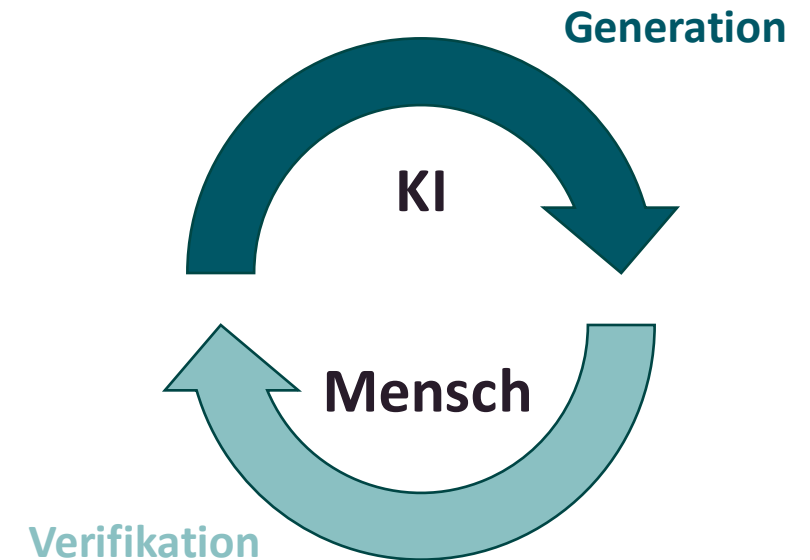
- Z.b. in Cursor Mobile (erstellt GitHub PRs)
- Lösung für überschaubares Problem größtenteils sinnvoll&hilfreich
- Dennoch manuelles Aufräumen, Dokumentieren, Testen notwendig
 - Sehr viele Schleifen notwendig, wenn tatsächlich selbst gar kein Code angefasst werden soll

Fuck-ups

- Tendenz, Code zu wiederholen anstelle von DRY-Prinzip
- Zu enger Fokus auf Code-Basis, wenn eigentlich allgemeine Frage gestellt oder weiterer Blick notwendig
- Ausführen von Tests
 - "Reward-hacking": Anpassen von Test-Cases anstatt echte Lösung zu implementieren
- Linter in Feedback-Schleife (automatische Korrektur)
 - Teilweise deutlich schlechtere Implementierungen, nur um initiale Linter-Fehler zu umgehen
- Inline imports... 🤖

Best practice #1

- Zyklus Generation-Verifikation fast immer notwendig – je schneller desto erfolgreicher
 - Klare Test-Cases (am besten vom Agenten ausführbar)
 - Modularität
 - atomare Anpassungen



Best practice #2

- KI-Assistenten als Sparringspartner



- Initiale Beschreibung des Problems und der vorgesehenen Änderung (kleinschrittig)
- Erklärung & Iteration zu Herangehensweisen
- Konkrete Implementierung
- Tests

Wie auch von A. Karpathy vorgeschlagen*

* <https://www.youtube.com/watch?v=LCEmiRjPEtQ>

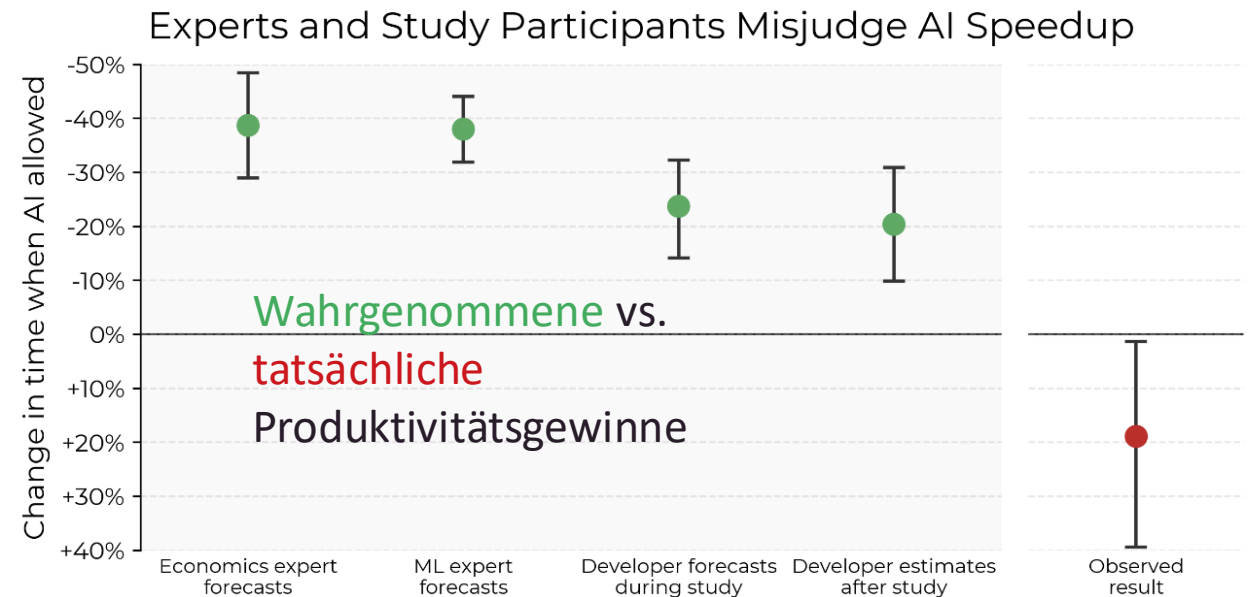
Best practices #3

- Manchmal hilft der weitere Kontext: Problembeschreibung an allgemeinen Chatbot (ganz ohne detaillierten Kontext eines KI-Assistenten)
- Hilfe bei Kleinigkeiten: Benennung von Dingen
- Verweis auf Zentrale Dokumentation / Architekturbeschreibung
 - Tools wie [context7](#)
- Cursor: verzichte auf Auto-Model-Auswahl
 - zwar sehr geringe Latenz, aber zu großer Streuung in Qualität

Best practice #4

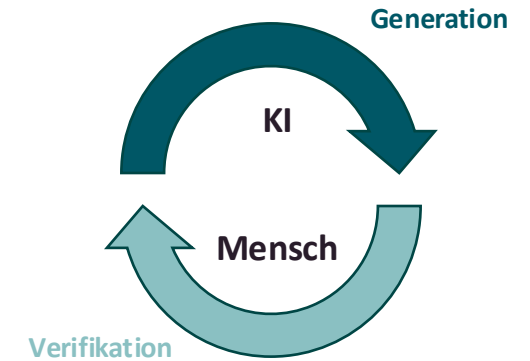
- Kritisch bleiben trotz des Hypes...
 - Studie: senior-level Entwickler & komplexe Code-Basen
 - Aber auch: Evidenz dass Agenten Funktionalität im Prinzip richtig implementieren, nur nicht alle Requirements erfüllt sind

Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity



Ausblick

- Für effizienten Verifikations-Zyklus: UI entscheidend
 - Variiere Level an Autonomie, je nach Problem
- Deterministische Validierung und Tools
 - Zugriff auf Refactoring / Debugger / ...?
- Wissen aus Erfahrung (geteilt in Projekt/Codebasis)
 - Memory in Claude Code
 - Dynamisch erstellt in Cursor
- Spezifische Agenten (Planen, Review, ...) mit dedizierten Prompts
- Bessere Sprachmodelle?

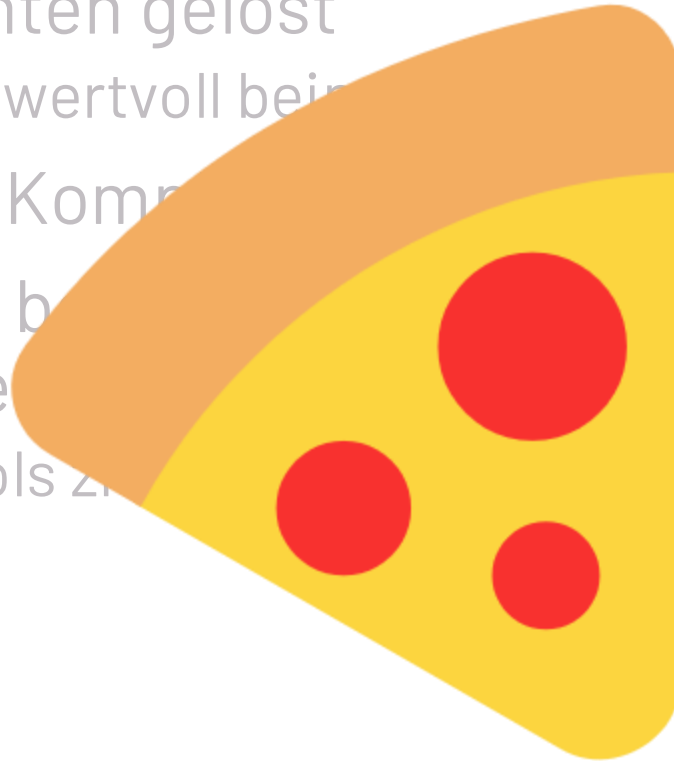


Fazit

- Kleinere bis mittlere “Standard”-Probleme gut von modernen Coding-Agenten gelöst
 - Besonders wertvoll beim Entwickeln in weniger vertrauten Sprache
- Ab gewisser Komplexität: keine sinnvollen Lösungen mehr
- Claude Code besticht im Moment durch besonders klare und strukturierte Lösungen
 - Andere Tools ziehen vermutlich schnell nach

Fazit

- Kleinere bis mittlere “Standard”-Probleme gut von modernen Coding-Agenten gelöst
 - Besonders wertvoll bei Problemen in weniger vertrauten Sprache
- Ab gewisser Komplexität von unnötigen Lösungen mehr
- Claude Code bevorzugt durch besonders klare und strukturierte
 - Andere Tools zu schnell nach



Backup

Große Sprachmodelle (LLMs) verändern, wie wir Software entwickeln – von smarten Assistenten im Editor bis hin zu autonomen Coding-Agents.

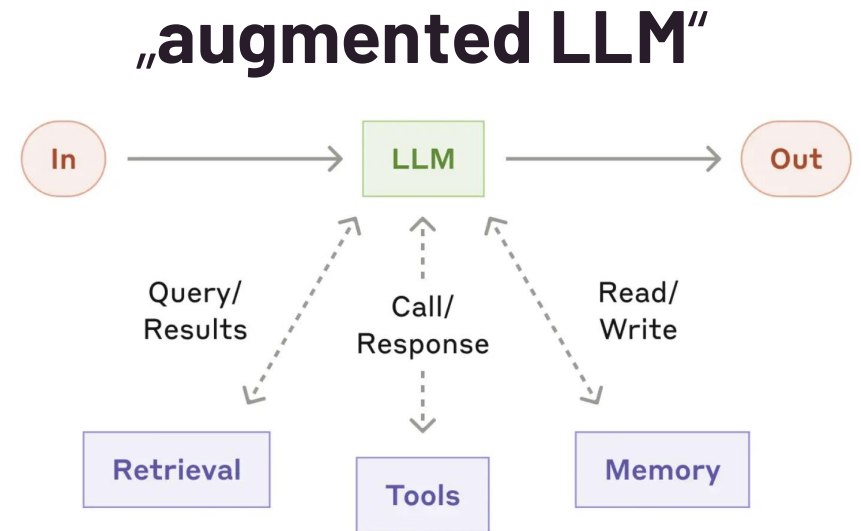
In diesem Talk teile ich praktische Erfahrungen mit KI-gestützter Softwareentwicklung. Welche Tools funktionieren im Alltag? Wo liegen die echten Produktivitätsgewinne? Welche Use Cases lohnen sich – und welche (noch) nicht?

Wie gut schreiben LLMs Code?

| Model Name | Developer | Type | HumanEval (Pass@1) | SWE-Bench (% Resolved) | LiveCodeBench (Pass@1) | MBPP (Accuracy) | Context Window | Cost Tier | Standout Feature/Strength |
|-----------------------|-------------|-------------|--------------------|------------------------|------------------------|-----------------|----------------|-----------|--|
| Claude 3.7 Sonnet | Anthropic | Commercial | ~86% | ~70% | ~50% | N/A | 200k | High | Leading real-world coding, Reasoning Mode |
| OpenAI o3 (high) | OpenAI | Commercial | ~80% | ~69% | ~79% | N/A | 128k+ | Very High | Top-tier reasoning, Strong Aider performance |
| Gemini 2.5 Pro | Google | Commercial | ~99% | ~64% | ~70% | N/A | 1M+ | High | Massive context, Strong reasoning/math |
| OpenAI o4-Mini (high) | OpenAI | Commercial | N/A | ~68% | ~73% | N/A | 200k | Medium | Top LiveCodeBench, Balanced reasoning/speed |
| GPT-4o | OpenAI | Commercial | ~90% | ~33-55%* | ~30% | ~90%** | 128k | Medium | Speed/Cost balance, Multimodal, Ecosystem |
| DeepSeek R1 | DeepSeek AI | Open Source | ~37%*** | ~49% | ~64% | N/A | 128k+ | Low (API) | Strong reasoning/math (open), Efficiency |
| Llama 4 Maverick | Meta | Open Source | ~62% | N/A | ~41-54% | ~78% | 10M (claim) | Free (OS) | Massive context potential, Creativity |
| Qwen 2.5 Coder (32B) | Alibaba | Open Source | N/A | ~31% | N/A | N/A | 128k | Free (OS) | Strong Python (local), Long context handling |

Der Kern zukünftiger KI-Anwendungen

- Relevanter Kontext muss explizit bereitgestellt werden
 - RAG-Systeme vielfältig und effektiv genutzt
- „Tool-use“ für LLMs: ermöglicht das Ausführen von Code, Datenbank-Zugriffen, Websuche, etc.
 - LLM entscheidet selbst, welche Aktion ausgeführt werden soll
- Gedächtnis für LLM: Persistenz

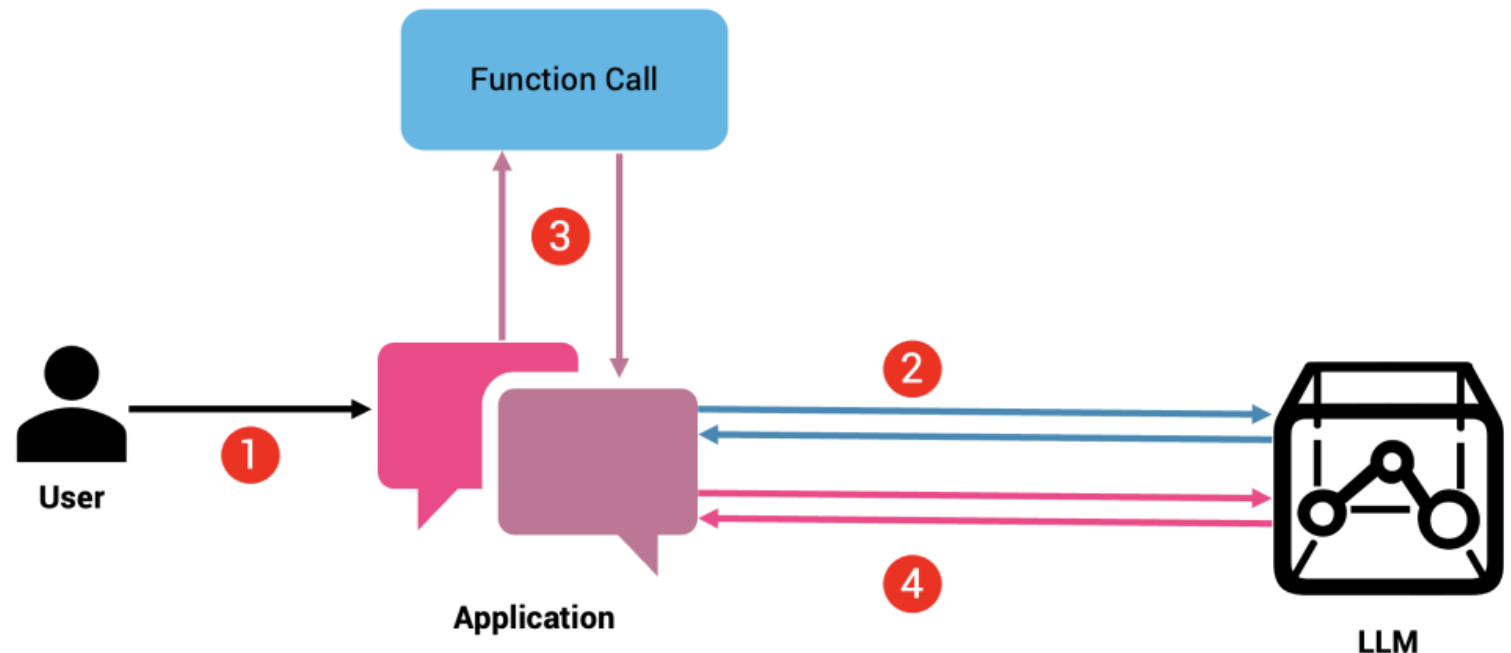


<https://www.anthropic.com/engineering/building-effective-agents>

MCP beschreibt technische Umsetzung hiervon

Wie funktioniert die Kommunikation?

1. User-Anfrage
2. LLM interpretiert Nachricht, entscheidet welches Tools auszuführen ist
3. Server führt Tool aus
4. LLM antwortet, basierend auf Tool-Output



Best practices #4

- Kritisch bleiben trotz des Hypes...
 - Studie: senior-level Entwickler & komplexe Code-Basen
 - Aber auch: Evidenz dass Agenten Funktionalität im Prinzip richtig implementieren, nur nicht alle Requirements erfüllt sind

Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity

