andrena
OBJECTS

KI-Coding ist nicht perfekt…

aber sehr nützlich

andrena
OBJECTS

Wer sind wir?

Peter Wegner

peter.wegner@andrena.de

LinkedIn

GitHub

andrena
OBJECTS

Wer sind wir?

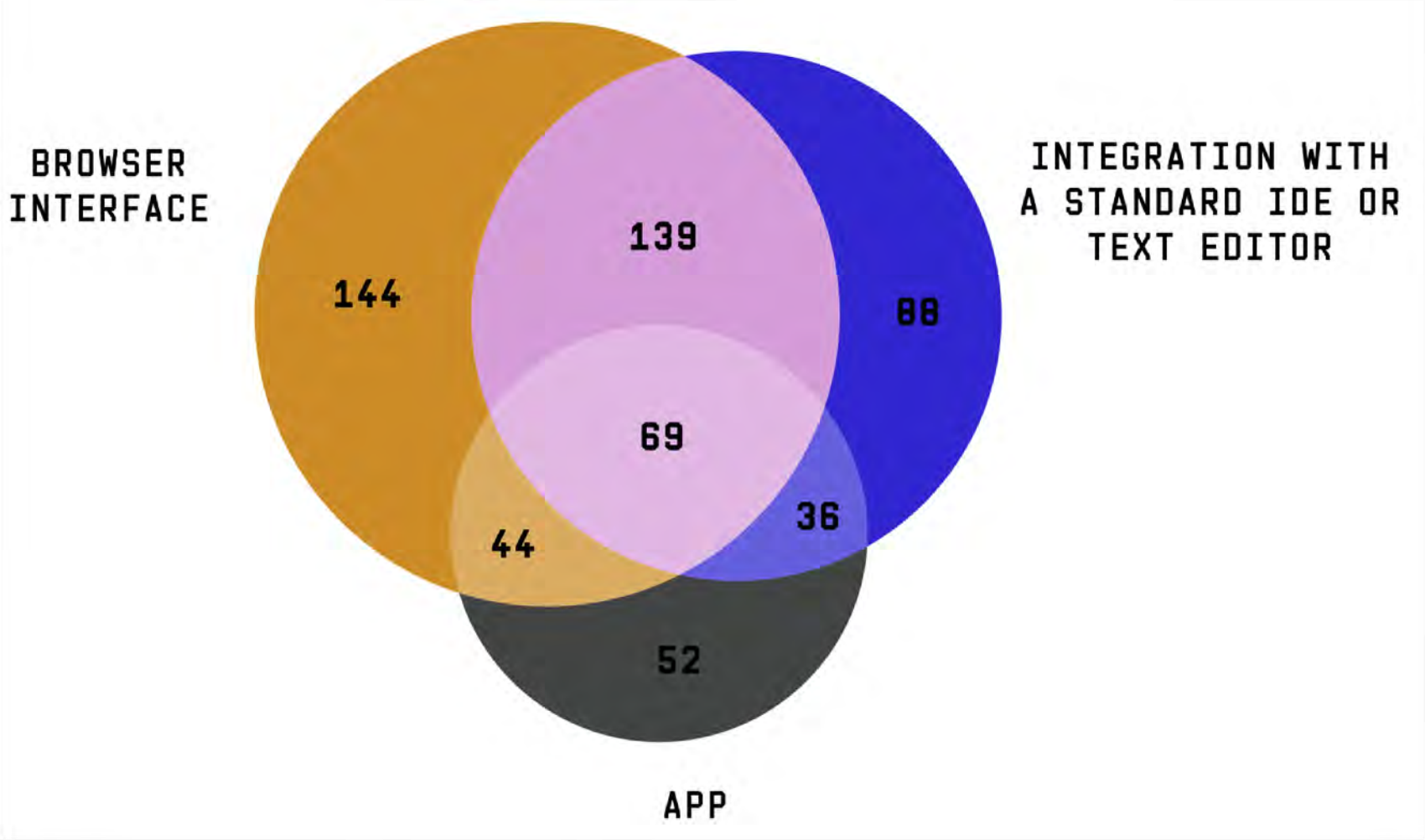
Jacques Huss

jacques.huss@andrena.de

LinkedIn

GitHub

andrena
OBJECTS



https://www.wired.com/story/how-software-engineers-coders-actually-use-ai/

andrena
OBJECTS

Zwei Aussagen:

"Ich muss kaum noch Code selbst schreiben! Die KI nimmt mir super viel Arbeit ab. Vibe-Coding ftw!"

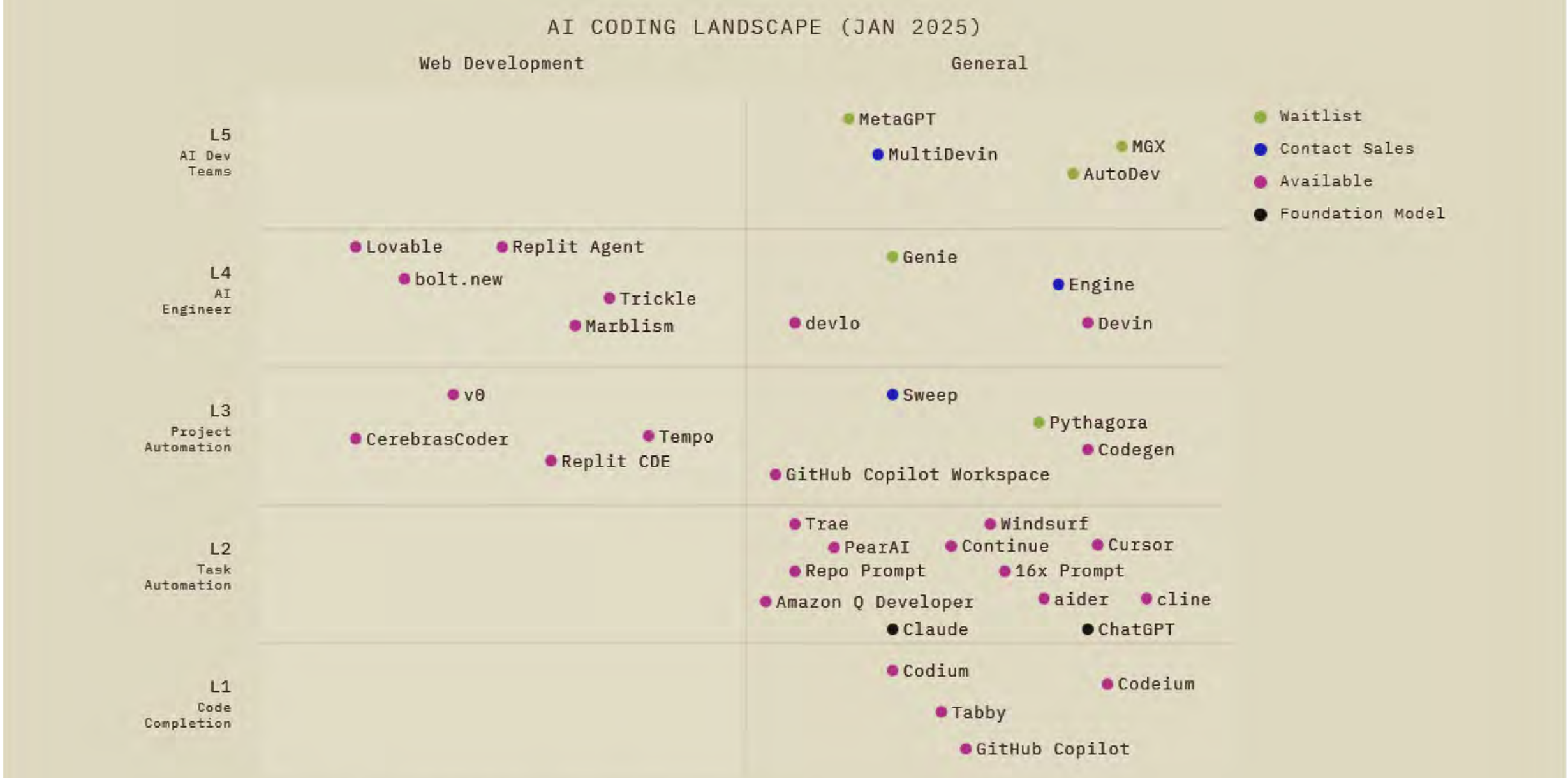
"Die KI kann gar nichts, den Code hätte ich selbst viel schneller und sauberer geschrieben!"

Wer hat Recht?

andrena
OBJECTS

Beide!

- KI-Tooling hat enormes Potenzial zur Produktivitätssteigerung
- KI-Tooling macht immer noch Anfängerfehler
- **Ziel:** Schwächen ausgleichen, Stärken nutzen



Note: Item positions within quadrants are arbitrary.

Created by Zhu Liang (16x Prompt) using d3 and Cursor. Source code

Read my blog post for more details on the levels (L1-L5) and AI tools included in this visualization. You might also be interested in my first impressions of Devin.

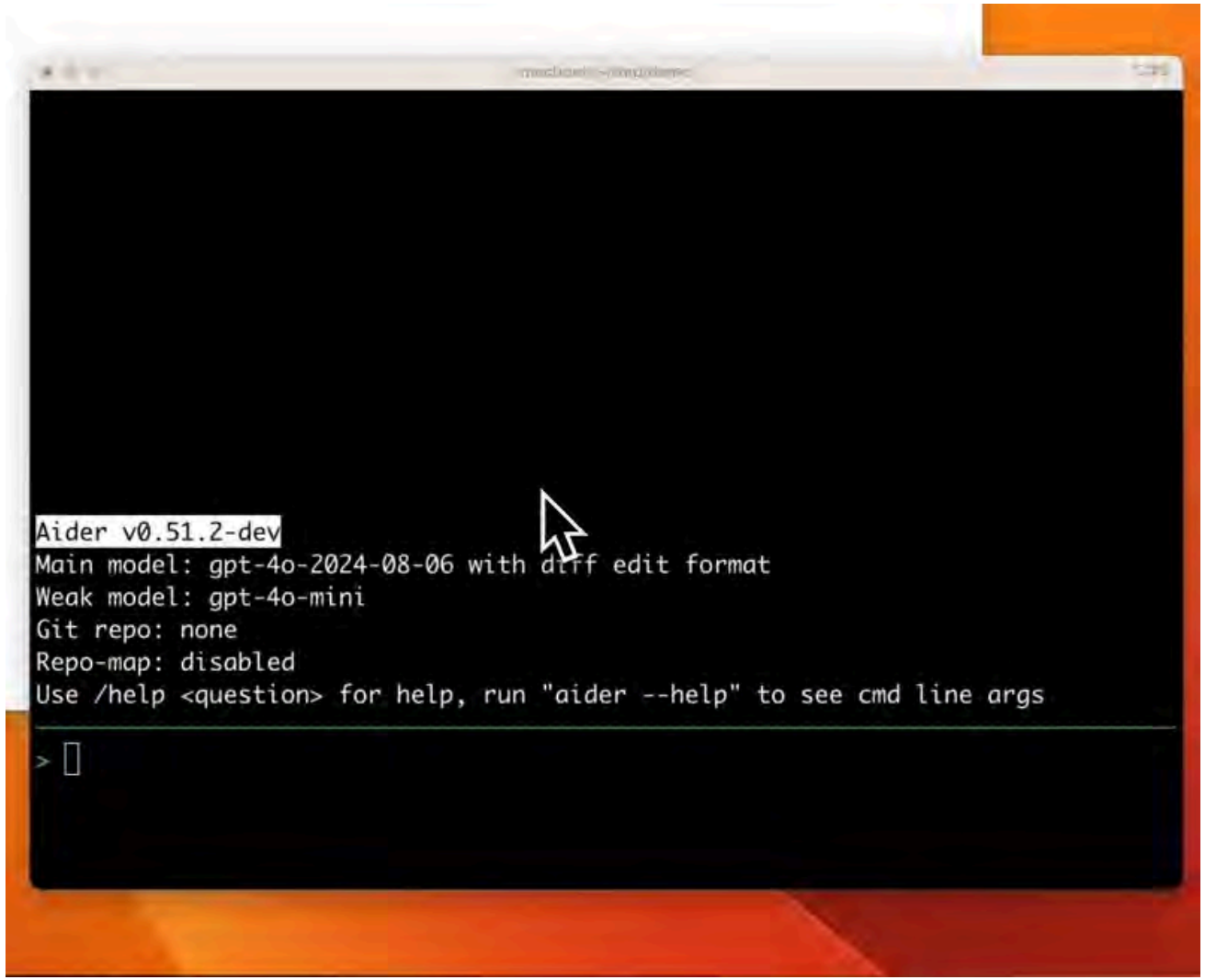
DEMO

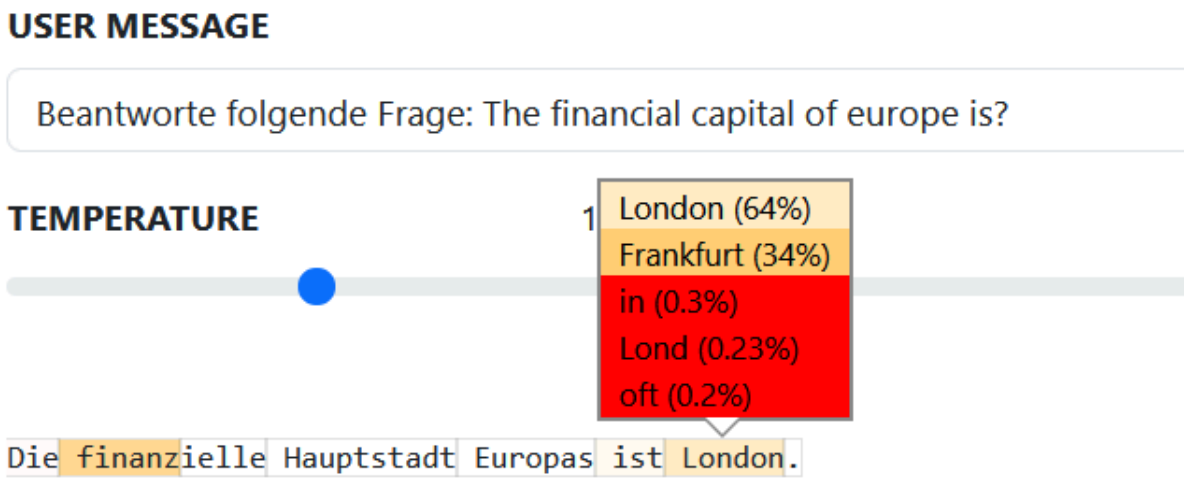
Level 4

- bolt.new

Level 2

- Coding aider





Warum machen LLM's Fehler?

- LLM's werden mit dem Ziel trainiert den nächsten passenden Token zu finden
- Plausibel =/= wahr
- Richtige Antworten sind situationsabhängig

andrena OBJECTS

Kontext ist sehr wichtig

- gesamte Codebasis ist in der Regel zu viel
- LLMs haben sich seit GPT3.5 Ende 2022 enorm verbessert
- seit Mai 2024 wirklich brauchbare Kontextlängen möglich (GPT-4o 128k Token Kontext)
- Worauf es aber dennoch ankommt: Fiction Livebench
- Knowledge Cutoff, durch Nachtuning wird nachgebessert

Konsequenz: Nur notwendigen Code mitgeben, der alle erwünschen Muster enthält und auch nur diese (d.h. nichts was gar nicht mit dem Problem zu tun hat)

andrena
OBJECTS

DEMO: Muster werden fortgesetzt

Warum "Gutes Prompten" allein oft nicht reicht?

- Weil mein bestehender Code auch Teil des Prompts ist, ggf. sogar ein sehr großer
- LLMs setzen von Natur aus bestehende Muster fort

Chatting with LLMs to fix errors:

User: Error X occurs
Model: Solution A

User: Error X still occurs
Model: Solution B

User: Error X still occurs
Model: Solution C

User: Error X still occurs
Model: Solution A

User: Error X still occurs
Model: Solution B

andrena
OBJECTS

Weniger ist oft mehr (Sonnet 3.5)

```
#### okay so lets take a step back. If the audio is sent for transcription or to Make.com it handles
creating a signed URL perfectly - whats the difference between that one and the documents we have here.
I have now applied the RLS

Ok.

> Tokens: 34k sent, 1.7k received. Cost: $0.12 message, $6.13 session.

#### answer me please

Ok.

> Tokens: 34k sent, 3.0k received. Cost: $0.14 message, $6.27 session.
```

Exkurs: Was wir an Bildgeneratoren lernen können

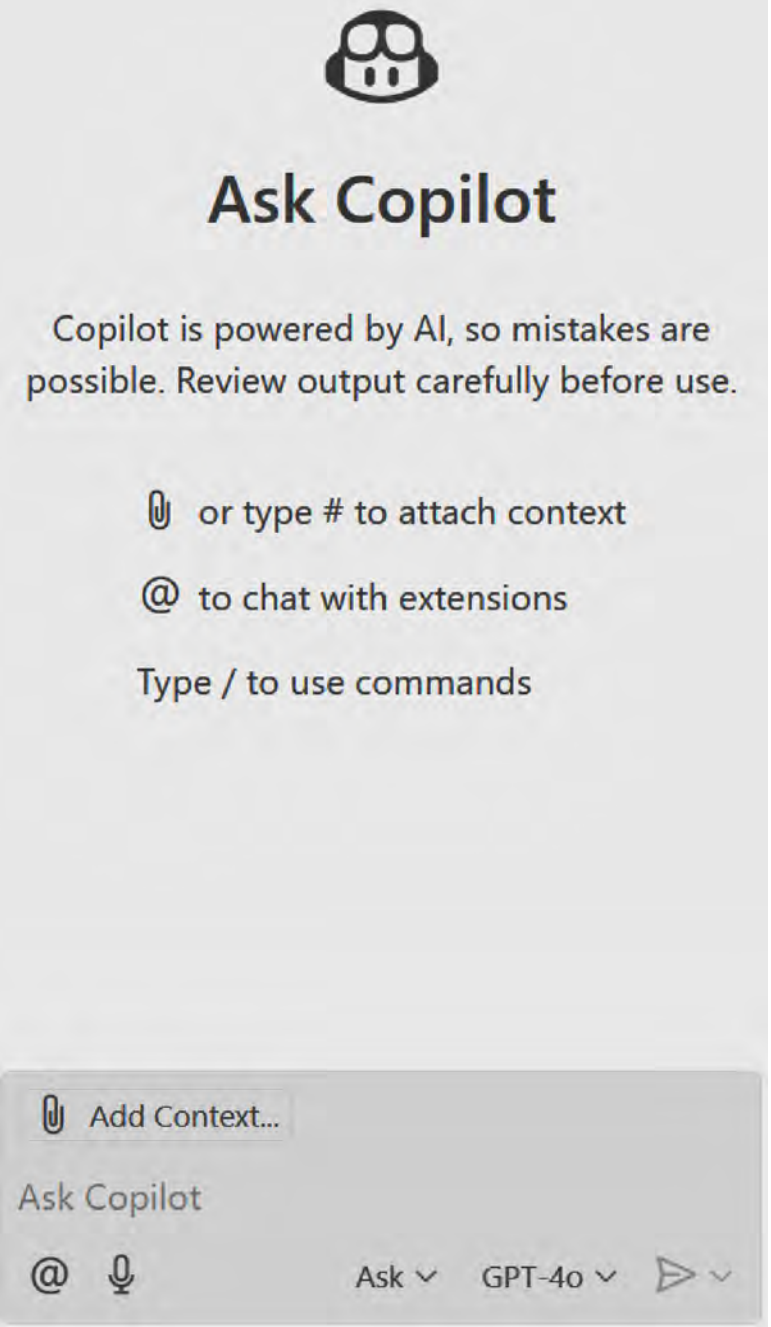
Bildgenerator (Flux) Prompt:

```
car
```

andrena
OBJECTS

Inpainting

Was soll passieren wenn ich auf der linken Bildhälfte "Inpainting" mache?

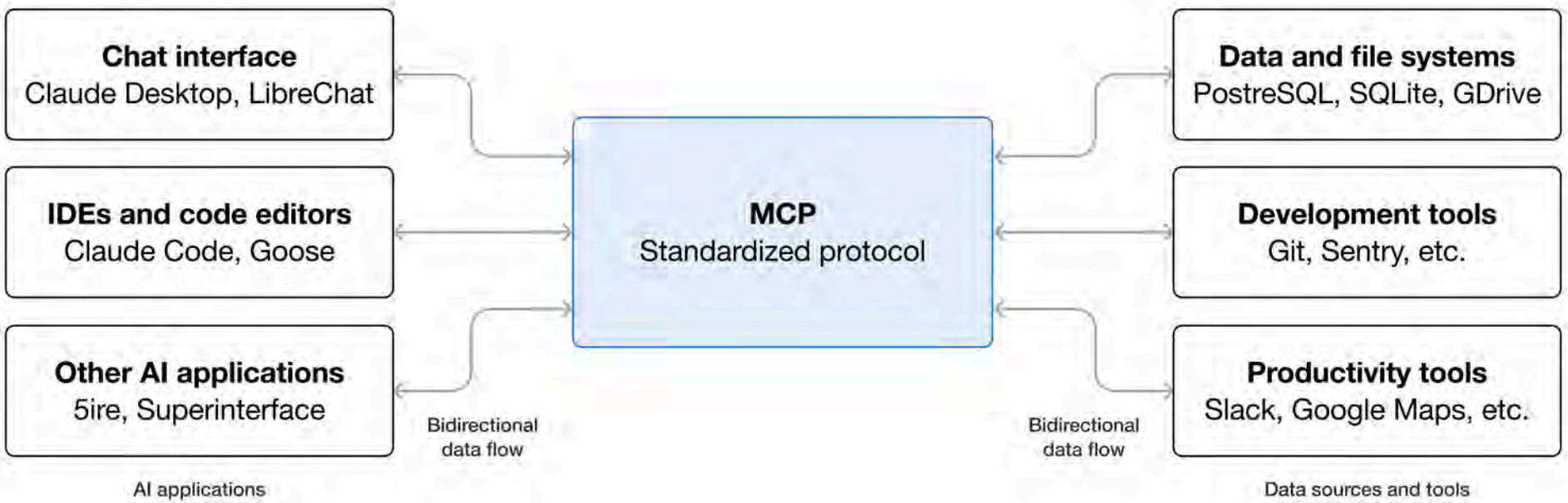


Kontext bereitstellen

- Toolcalling
  - grep, websuche, …
  - Dokumentation
  - Compiler
  - Linter
  - Tests
  - …
- Manuelles nachahmen möglich

andrena
OBJECTS

Model Context Protocol (MCP)



Quelle: https://modelcontextprotocol.io/faqs

andrena
OBJECTS

DEMO: MCP

andrena
OBJECTS

Wie man Kontext verbessert

- Eigenes Wissen in den Prompt integrieren
- Beispiele statt mehr Prompts
- Feedbackloops nutzen oder erstellen

andrena OBJECTS

Tooling - Schwächen und Herausforderungen

- **IDE Tool-Ceiling**: Begrenzte Integration
- **Übliche Refactorings nicht direkt verfügbar**: Rename, Extract Method, Extract Interface, ...
- LLMs bekommen kaum Unterstützung durch übliche Entwicklerwerkzeuge wie **Language Service Protocol**, **Compiler**, **Linter**, ...
- Dadurch **Fehlende Feedback-Loops**: Compiler, Linter, Tests
- Bereits **partielle Dateiänderungen** sind eine Herausforderung

andrena
OBJECTS

Herausforderungen

- **LLM** als austauschbarer Baustein mit individuellen Grenzen
- **Agent vs. Workflow**: Manche Werkzeuge sind sehr agentisch, andere folgen fester Blaupause
- Stand jetzt sind **Agenten ohne Limits** in der Regel (zu) teuer
- Aboservices wie *Github CoPilot*, *Cursor*, *Windsurf*, *Jetbrains Junie* können nicht das volle Potenzial nutzen und müssen **Kosten optimieren**
- **Neue Modelle sind nicht in allem besser**: siehe aider benchmark

andrena OBJECTS

Zusammenfassung

- **Kontext ist entscheidend** für erfolgreiche Zusammenarbeit
- **Code-Qualität beeinflusst** die Effektivität der KI-Unterstützung
- **Neue Routinen** nötig, das Beste aus den Werkzeugen herauszuholen (smarte Kontextauswahl, dem LLM Tools zur Verfügung stellen)
- **Zukünftige Entwicklungen** könnten die Art, wie wir programmieren, grundlegend verändern (Agenten, **Model Context Protocol**, *Agent to Agent Protocol*)

andrena
OBJECTS

Fragen und Diskussion / Feedback

andrena
OBJECTS

Praktische Tipps für effektives AI Coding

andrena
OBJECTS

Builder vs. Mender Mentalität

- **Builder**: Neues erschaffen
- **Mender**: Bestehendes verbessern
- **KI kann beides unterstützen**
- Output ist immer eine Grundlage die ich danach anpassen kann
- aber: Manchmal fängt man lieber von vorne an

andrena OBJECTS

Praktische Tipps für effektives AI Coding

1. Das richtige Werkzeug je nach Problem wählen **Ohne KI vs. Chat vs. Workflow vs Agent** wählen
2. **KI anpassen**: Eigene Regeln und Präferenzen mitteilen oder dokumentieren, Projektstruktur und technische Anforderungen dokumentieren, auch automatisch z.b. mit Cline Memory Bank, `cursorrules` , `clinerules`
3. **Werkzeuge und Workflows bauen**, z.b. Skripte generieren lassen, die z.b. AI Tools wie aider ausführen. Auch MCP kann hier einiges ermöglichen.
    - Ai Scientist
    - Airbnb test migration Enzyme -> React Testing Library

andrena OBJECTS

Praktische Tipps für effektives AI Coding (Fortsetzung)

4. **Ask/Act** Pattern
5. **Chats fokussiert halten**: Ein Problem pro Konversation
6. **Interfaces/Dummys** anlegen
7. **Probleme aufteilen**: Komplexe Aufgaben in kleinere Teile zerlegen
8. **Modularer Code** hilft guten Kontext zu geben
9. **Typisierte** Sprachen verwenden

andrena
OBJECTS

Praktische Tipps für effektives AI Coding (Abschluss)

10. Tests geben **Sicherheit**
11. **Code nicht blind akzeptieren**, immer überprüfen und verstehen
12. **Ownership**/Verantwortung für den Code übernehmen