GI Fachgruppentreffen 2024

Integrating LLMs into Software Engineering: AI as a Component

Lukas Netz
Software Engineering
RWTH Aachen

se-rwth.de

AI4SE | SE Software Engineering | RWTH AACHEN UNIVERSITY

About me

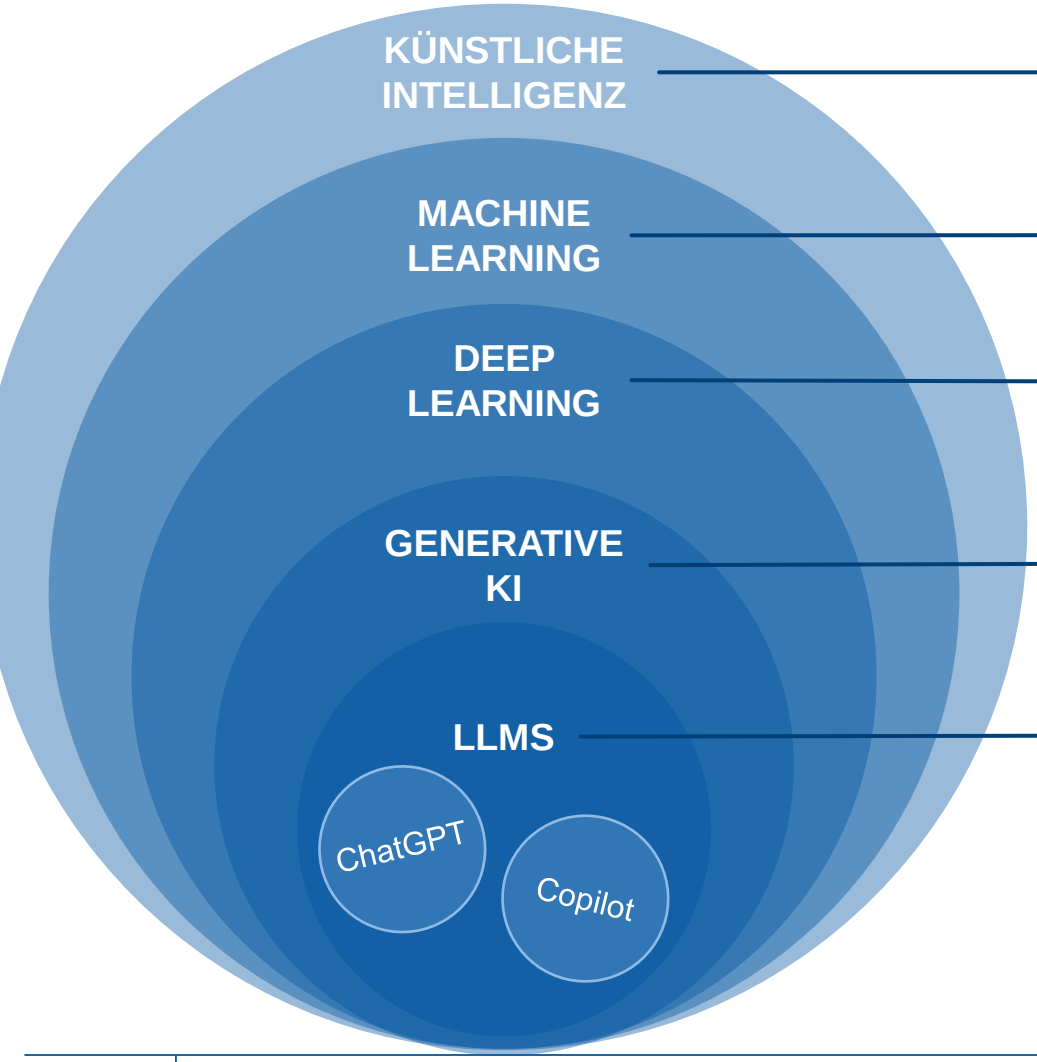
Lukas Netz
Lehrstuhl für Software Engineering RWTH Aachen
Software Engineer | AI-Driven SE | MDD | Low-Code

Teamlead AI4SE
KI-Getriebene Modellsynthese
KI in der Lehre
KI im Software Engineering

PhD:
Model-Driven Method for the Development of Full-Size Web-Based Information Systems
(Erstellung und validierung von Textutellen-Klassendiagrammen auf Grundlage von Domänenbeschreibungen)

se-rwth.de/staff/Lukas.Netz
linkedin.com/in/lukas-netz

Sammelbegriff KI



Forschungsgebiet, dass sich mit der Entwicklung **intelligenter Maschinen** befasst, die wie Menschen arbeiten und reagieren.

Statistische Algorithmen, die aus Daten lernen und auf ungesehene Daten verallgemeinern können.

Teilgebiet von Machine Learning, dass **Neuronale Netze** nutzt, um komplexe Muster in Daten zu erkennen.

Eine Art von KI, die in der Lage ist neue **Inhalte zu erzeugen,** indem sie bestehende Muster und Daten analysiert (genAI).

Große Spachmodelle sind fortschrittliche KI-Modelle, die **große Mengen an Textdaten** verarbeiten, um menschenähnliche Texte zu generieren oder zu verstehen.

➡ Large Language Models

Large Language Models

LLMs: **große Sprachmodelle**

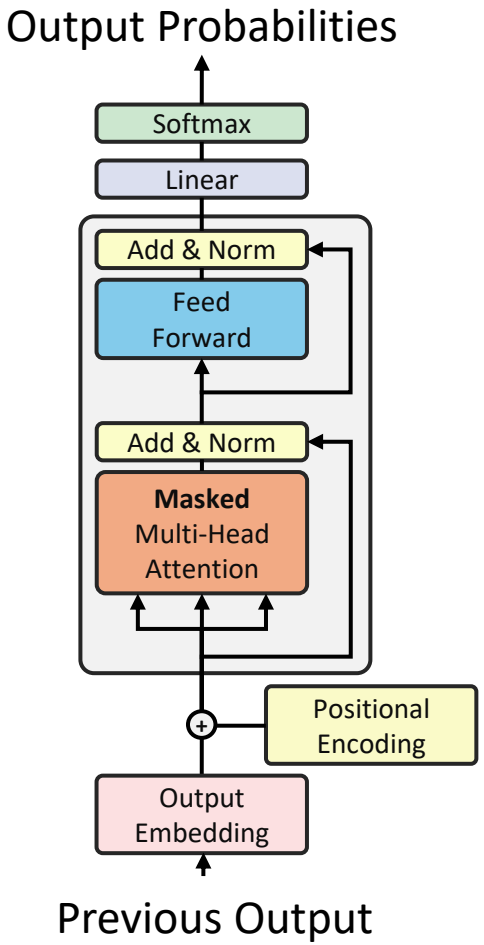
Leistungsstrake Modelle, die darauf ausgelegt sind Menschliche Sprache zu verstehen und zu generieren.

Sie können Text analysieren und verstehen, **kohärente** Antworten generieren und **sprachbezogene** Aufgaben ausführen

- Sentiment-Analyse
- Übersetzungen
- Informationsabfragen
- …

Was zeichnet LLMs aus ?

- **Transformer** Architektur (parallele Verarbeitung, self-attention)
- **Trainingsdaten**: Extrem Große und vielfältige Texte (15 Billionen Worte)
- **Generalisierbarkeit** und vielfältige Einsetzbarkeit (Allgemeiner Trainingsdatensatz)



Gemini (google)

Apple Intelligence

Open AI

Meta AI (Llama)

Claude Anthropic

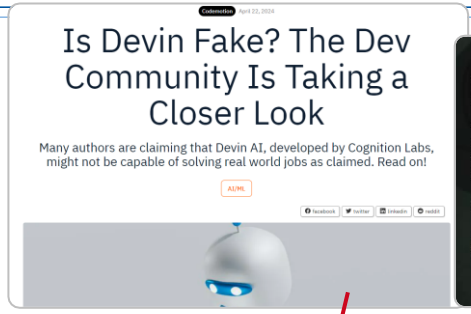
KI-Hype

Wild-West KI

- Publikationen auf Arxiv
- KI-Welt zu schnelllebig für Konferenz-Publikationen ?
- Starke Fluktuation in den Angebotenen Software- und Hardwarelösungen im Bereich KI
- Unklarer Stand der Technik

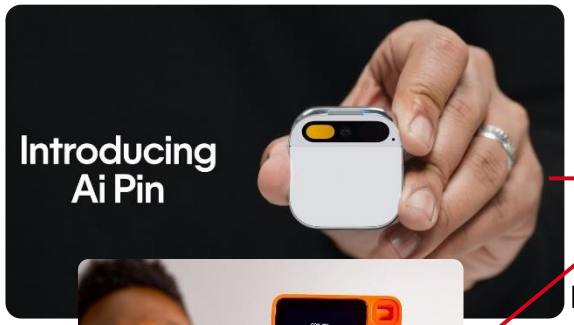
Generalisierbarkeit und Reproduzierbarkeit von Forschungsergebnissen

[3]



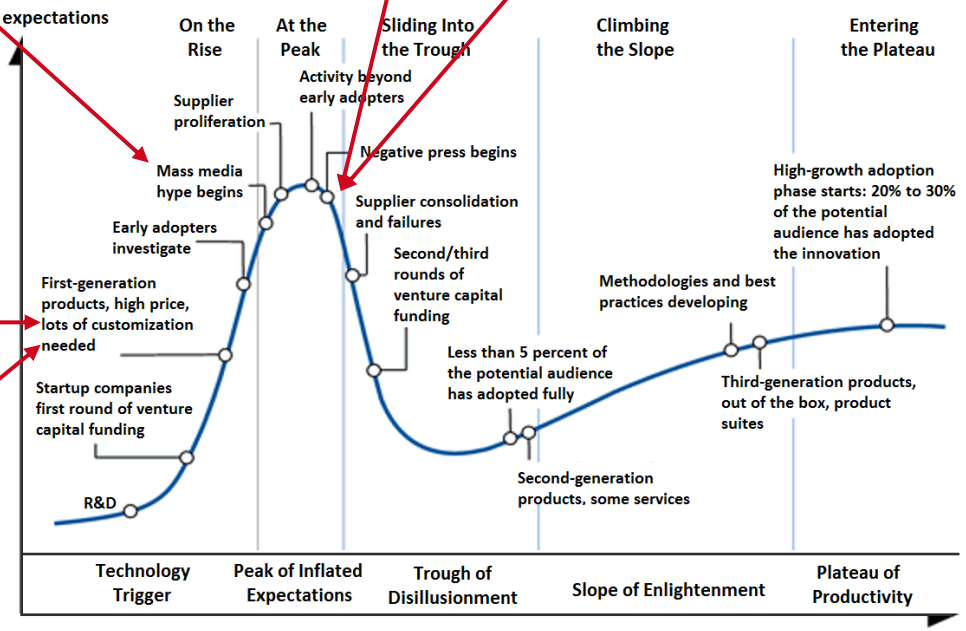
[4]

[5]



[2]

[6]



General Hype Cycle for Technology

[2] https://humane.com/aipin
[3] https://www.geeksforgeeks.org/devin-ai-worlds-first-ai-software-engineer/
[4] https://www.codemotion.com/magazine/ai-ml/is-devin-fake/
[5] https://www.zeniteq.com/blog/devins-demo-as-the-first-ai-software-engineer-was-faked
[6] Rabbit R1: Barely Reviewable https://www.youtube.com/watch?v=ddTV12hErTc

LLMs im Software Development Live Cycle (SDLC)

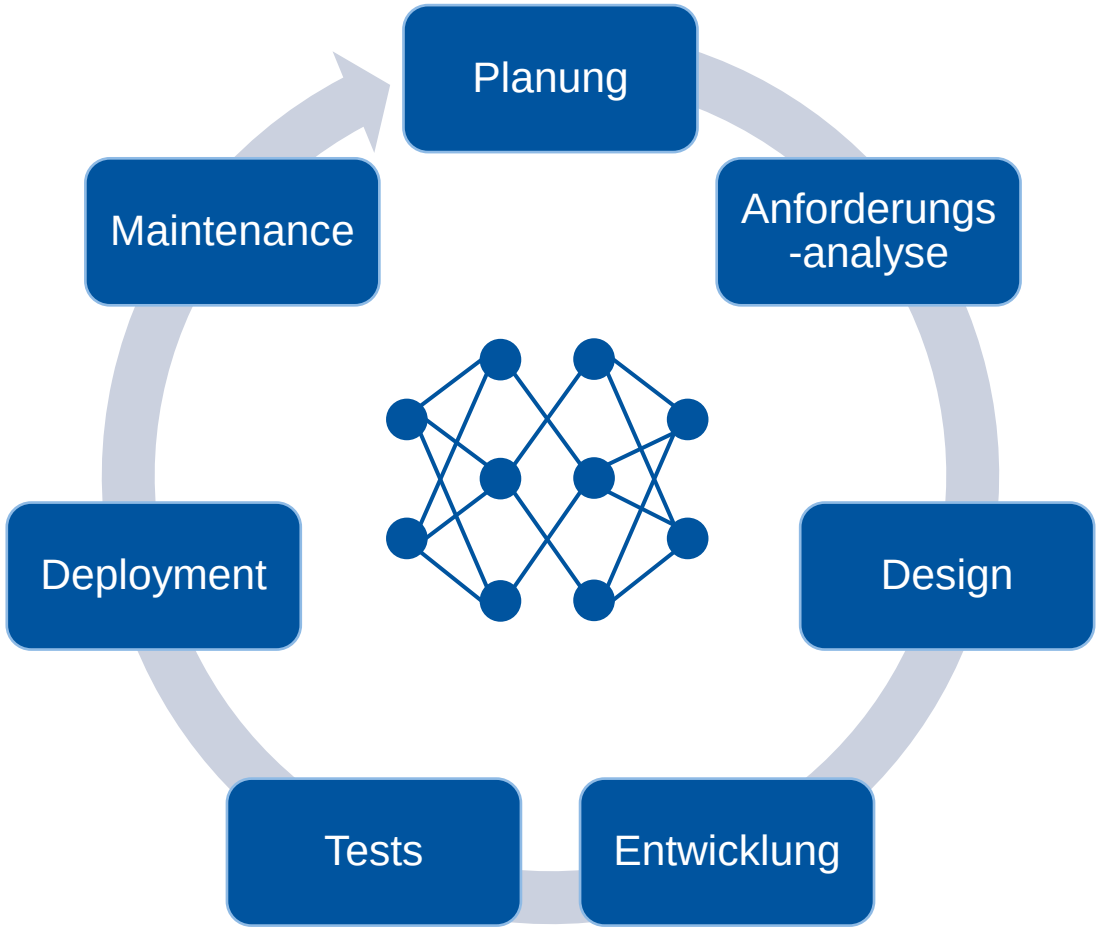
KI in der Software Entwicklung

−

- **Unüberprüfbare** Ergebnisse (Black Box)
- Ungenauigkeit
- Fehleranfälligkeit
- Mittelmäßige Umsetzungen vs. gute Umsetzungen

+

- Befähigung von Amateuren komplexe Entwicklungsprozesse zu bewältigen
- Zeitersparnis und Effizienzsteigerung
- Werkzeuggedanke



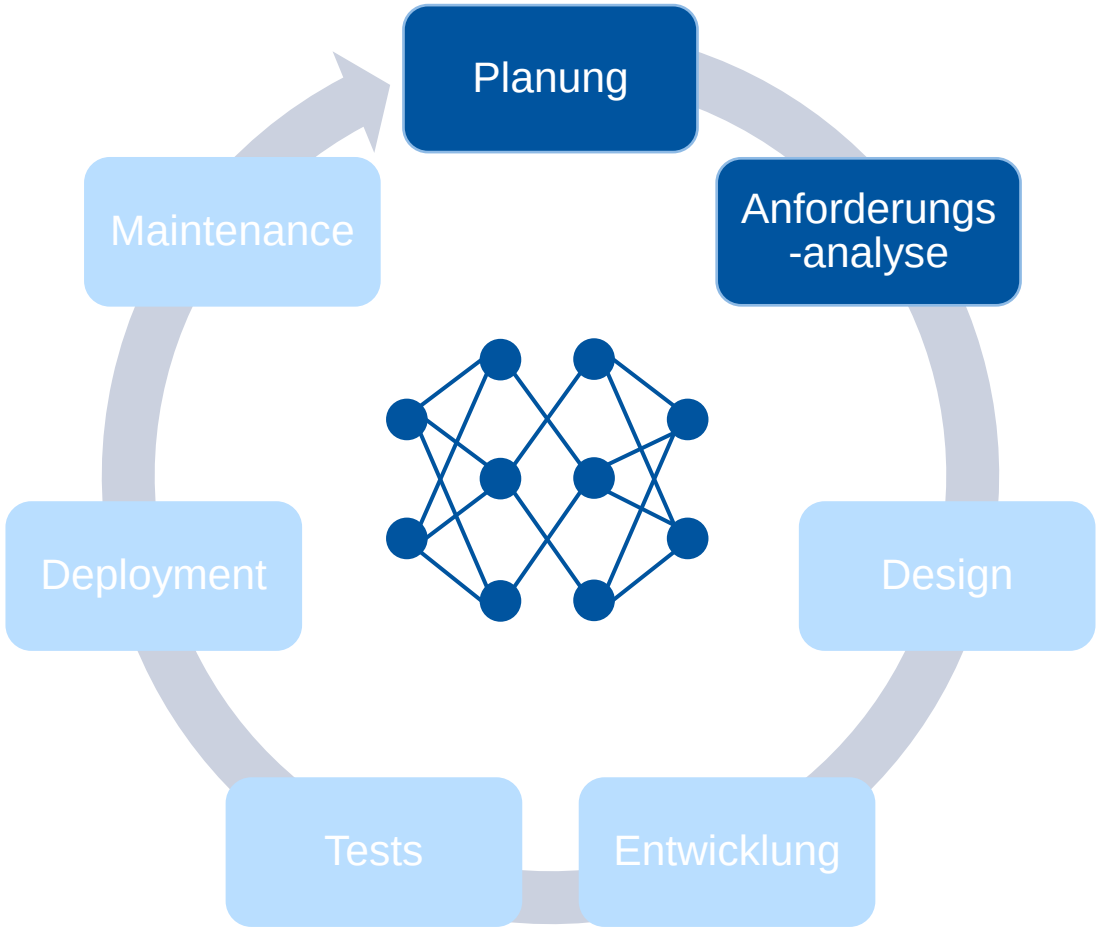
KI-Anwendungsbeispiele: Planung & Anforderungsanalyse

Beispiele zur Verwendung von LLMs in
Planung und Anforderungsanalyse

Überbrückung : Domänenexperte & Software Entwickler

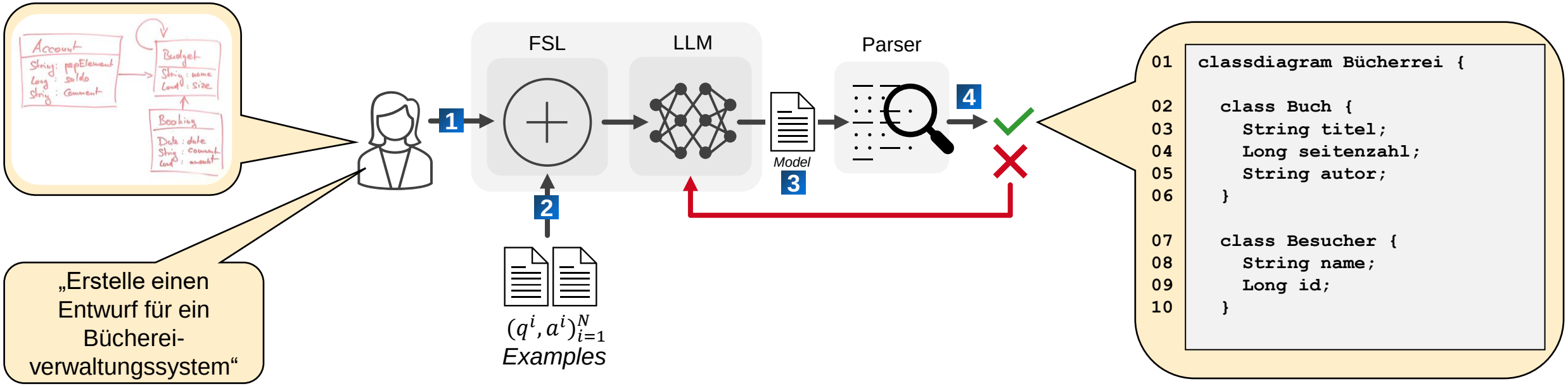
- Formalisierung von Anforderungen:
  - Transformation von Informellen Anforderungen von Domänenexperten in eine Modellierungssprache

- Modelle können analysiert und verifiziert werden
  - SMT-Solving
  - Erkennung von Wiedersprüchen & Duplikaten
  - Abgleich mit Dokumentationen und Regelwerken



LLM4CD Architektur

1. Anwender kann in Freitext seinen Anwendungsfall beschreiben.
2. Prompt wird erweitert.
3. LLM produziert Text
4. Parser entscheidet ob Modell weiterverwendet werden kann



LLM-Basierte Modellsynthese

- **Pre-training / Fine-tuning**
  - Sehr **ressourcen-intensiv**, spezielle Hardware notwendig
  - Benötigt Trainingsdaten. Große Mengen, gute Qualität.
  - Statischer Ansatz. Kann nicht auf andere Modellierungssprachen übertragen werden

- **Instructions** (Erläuterungen)
  - Regeln einer Modellierungssprache (Grammatik) kann einem LLM, ‚erklärt' werden.
  - Limitiert durch Länge und Komplexität der Erläuterungen
  - Zu große/ komplexe Grammatiken sind ‚unerklärbar'

- **Few-Shot Learning**
  - Pragmatischste Ansatz
  - Reihe von Fragen-Antwort-Paaren, um die Konzepte der Grammatik zu vermitteln
  - Ermöglicht In-Kontext-Training für komplexere Grammatiken.

Die Prompt P_{FSL} die die Antwort a zur Frage q liefert:

$$P_{FSL}: (a|q, (q^i, a^i)_{i=1}^N)$$

q^1= *"Erstelle ein Modell für ein Büchereiverwaltungssystem"*

a^1=

```
01  classdiagram Bücherrei {

02    class Buch {
03      String titel;
04      Long seitenzahl;
05      String autor;
06    }

07    class Besucher {
08      String name;
09      Long id;
10    }
```

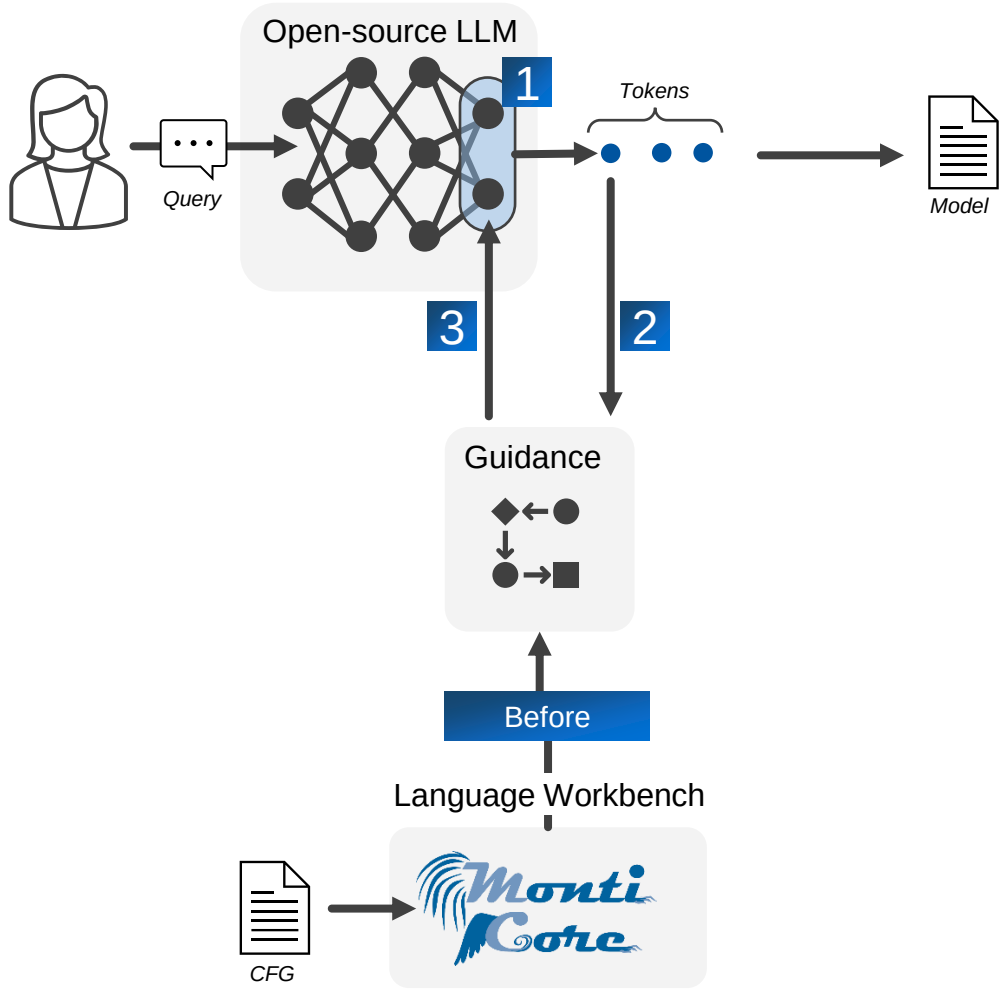
Grammar Masking / Constrained decoding

1 Inference Layer
Verteilungsfunktion im Inferenz-Layer vom LLM. LLM gibt einen Wortbaustein mit der höchsten Gewichtung auf Grundlage vorausgehender Wortbaustein zurück.

2 Aus Grammatik abgeleiteter Automat
Wir können Wortbausteine nutzen, um durch einen Automaten zu iterieren der durch die Grammatik definiert wird.

3 Filter
Bevor das LLM einen Wortbaustein ausgibt. Kann mit dem Automaten geprüft werden, ob dieser Baustein zu inkorrekter Syntax führen würde. In diesem Fall wird der nächstbeste Baustein als Kandidat gewählt

MontiCore Grammar as Input
Beliebige MontiCore-Grammatiken (.mc4) können verarbeitet werden



Beschränkung von LLMs-Ausgaben auf Modellierungssprachen

Verwendung von **Constrained Decoding** zur Optimierung der Formalisierung durch LLMs

1000 Modellierungsaufgaben (Klausuraufgabe 3. Semester)
Erstellung von Klassendiagrammen in CD4A Modellierungssprache

- Starke Verbesserung im Prozentsatz parsender Modelle
- Verwendung ‚kleiner' lokaler LLMs anstelle von Closed-Source APIs zu OpenAI
- Optimierung zu Lasten der Laufzeit

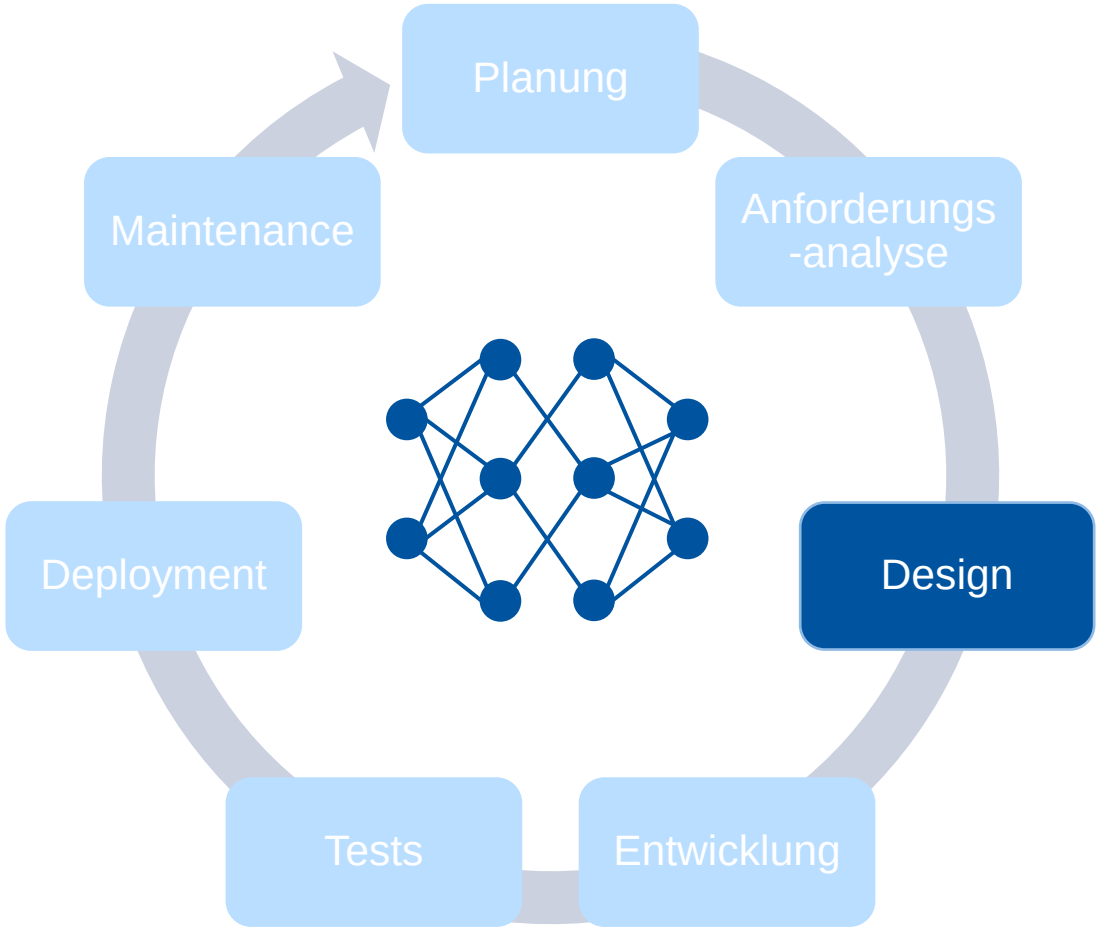
	Unconstrained	Constrained
CD4A (1000 examples)		
Llama3 8B 4-Bit		
Time (s)	5.71	74.10
Parsed (%)	416/991 (41.97%)	918/991 (92.63%)
Phi3 Mini 4-Bit		
Time (s)	4.63	138.29
Parsed (%)	357/976 (36.57%)	849/976 (86.98%)
Gemma 7B 4-Bit		
Time (s)	2.46	54.20
Parsed (%)	2/658 (0.003%)	612/658 (93.00%)
Mistral 7B 3-Bit		
Time (s)	4.89	73.59
Parsed (%)	190/905 (20.99%)	836/905 (92.37%)
GPT-4o (100 examples)		
Time (s)	8.77	N/A
Parsed (%)	76/100 (76.00%)	N/A

Anwendungsbeispiele : Design

KI-Unterstützung im Software Design

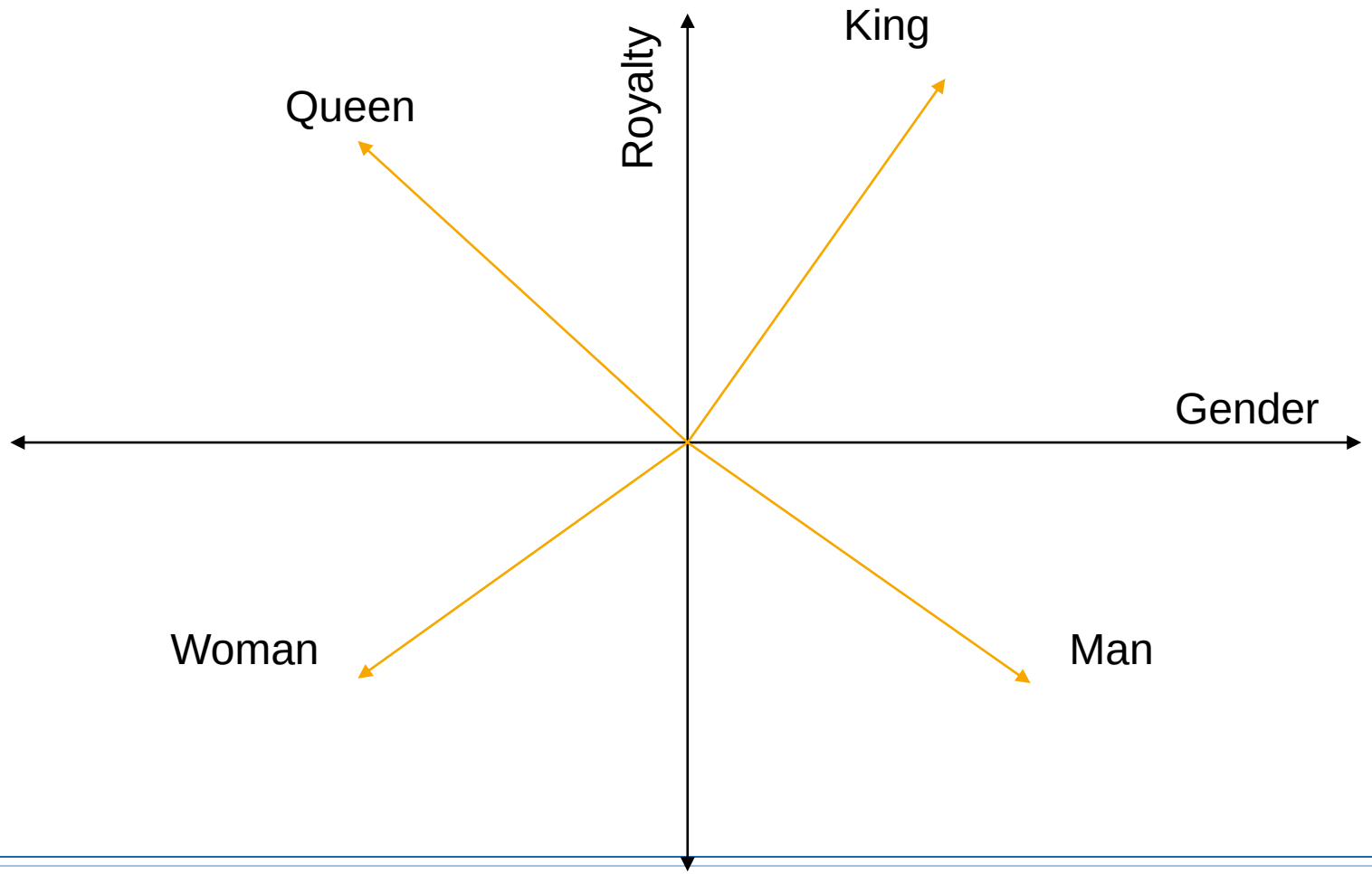
Einhaltung von Unternehmensrichtlinien & Normen , Best-Practices in der Designphase

- Viel Wissen ist bereits vorhanden, Aufwand für Anwender / Entwickler das **Wissen** zu **suchen** ist zu Hoch
- Modelle & Entwürfe können semantisch **gegen Richtlinien** und Normen geprüft werden.
- Vorschläge können aus bestehender Dokumentation heraus erstellt werden (**RAG**)



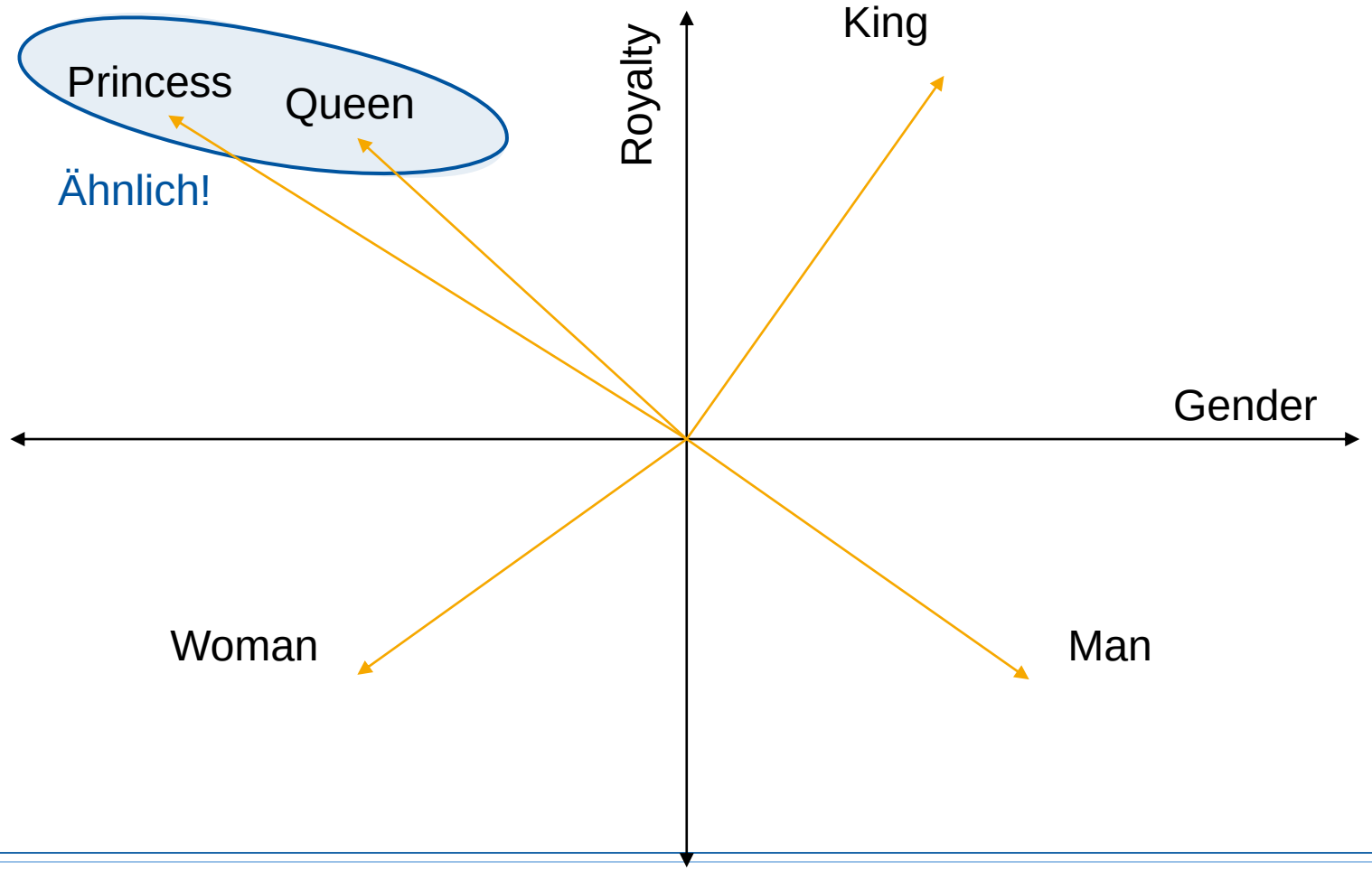
Word Embeddings, Einfaches Beispiel

King	$\begin{bmatrix} +1 \\ +1 \end{bmatrix}$
Queen	$\begin{bmatrix} -1 \\ +1 \end{bmatrix}$
Man	$\begin{bmatrix} +1 \\ -1 \end{bmatrix}$
Woman	$\begin{bmatrix} -1 \\ -1 \end{bmatrix}$



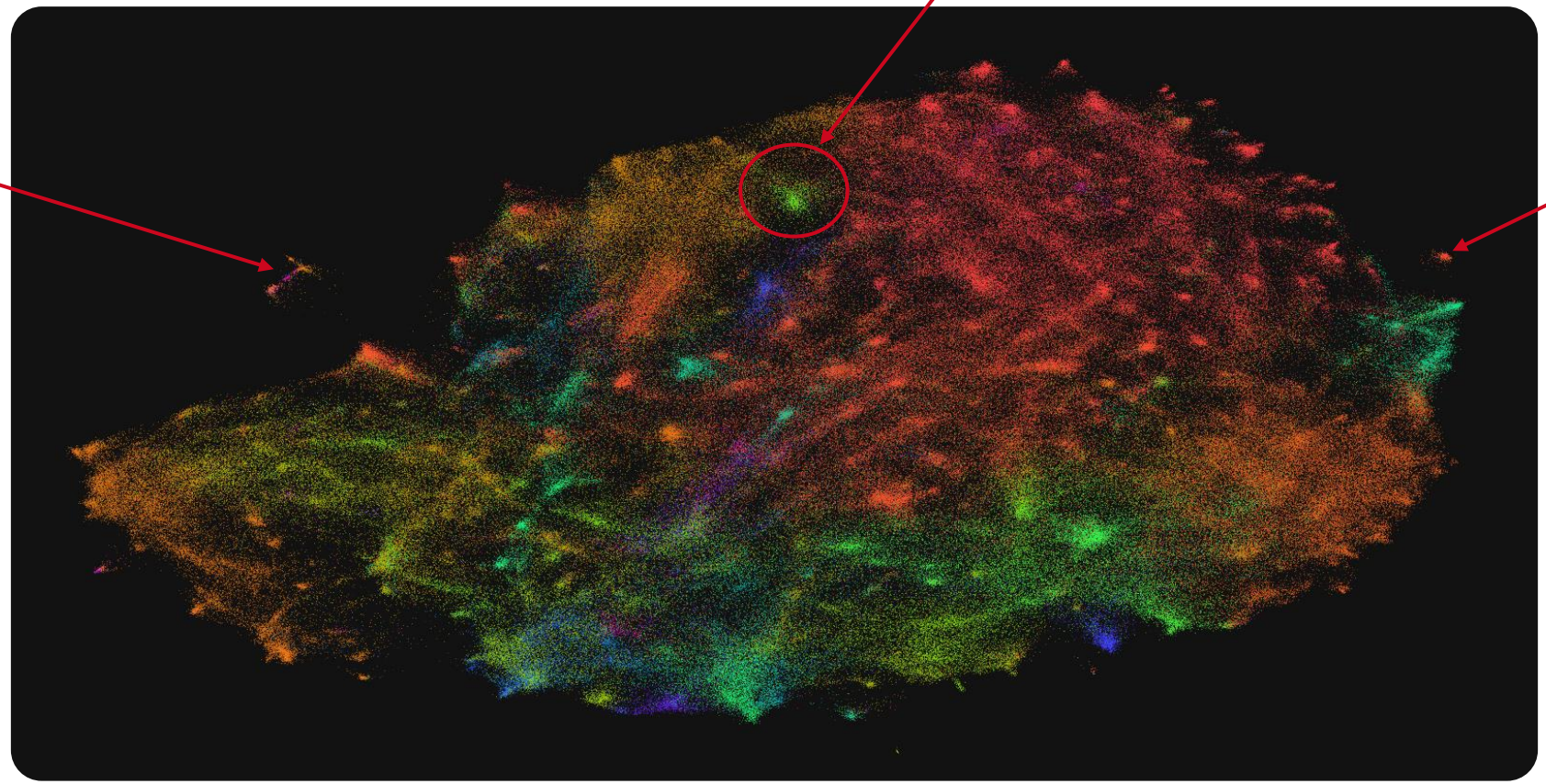
Word Embeddings, Simple Example

King	$\begin{bmatrix}+1\\+1\end{bmatrix}$
Queen	$\begin{bmatrix}-1\\+1\end{bmatrix}$
Man	$\begin{bmatrix}+1\\-1\end{bmatrix}$
Woman	$\begin{bmatrix}-1\\-1\end{bmatrix}$
Princess	$\begin{bmatrix}-1.1\\+1.2\end{bmatrix}$



Semantische Datenanalyse

Textembedding-3 : 3072 Dimensionen
Erkennung von Clustern in Textuellen Daten

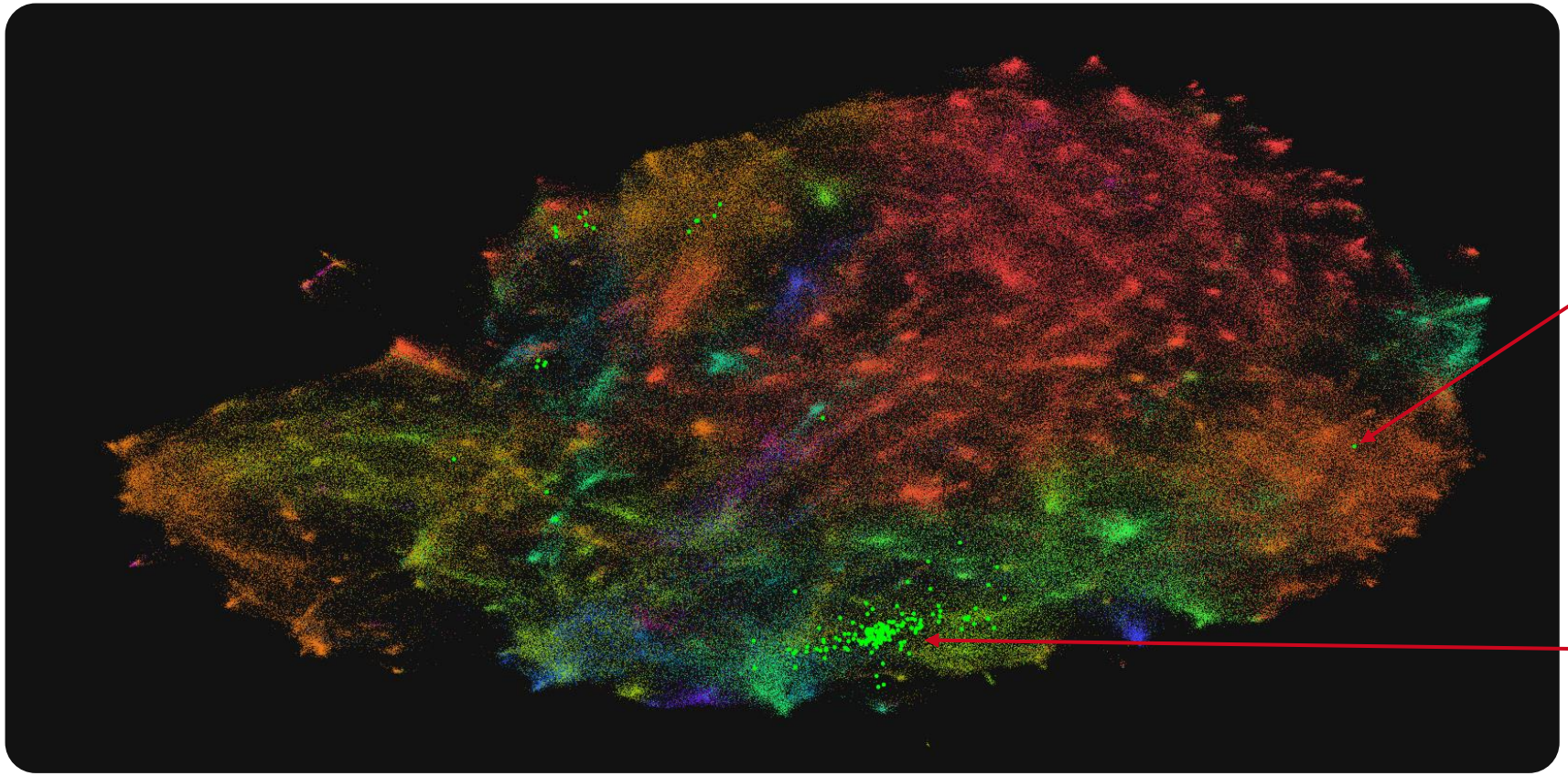


[7] https://atlas.uslu.tech/?map=cs

Semantische Projektion in 2D aller Informatik-Paper auf arxiv.org (485772 Paper) [7]

Semantische Datenanalyse

Publikationen unseres Lehrstuhls (Prof. Rumpe)



Semantische Projektion in 2D aller Informatik-Paper auf arxiv.org (485772 Paper) [7]

[7] https://atlas.uslu.tech/?map=cs

Interaktion mit LLMs auf Grundlage von Daten außerhalb des Trainings-Korpus

Retrieval Augmented Generation (RAG)

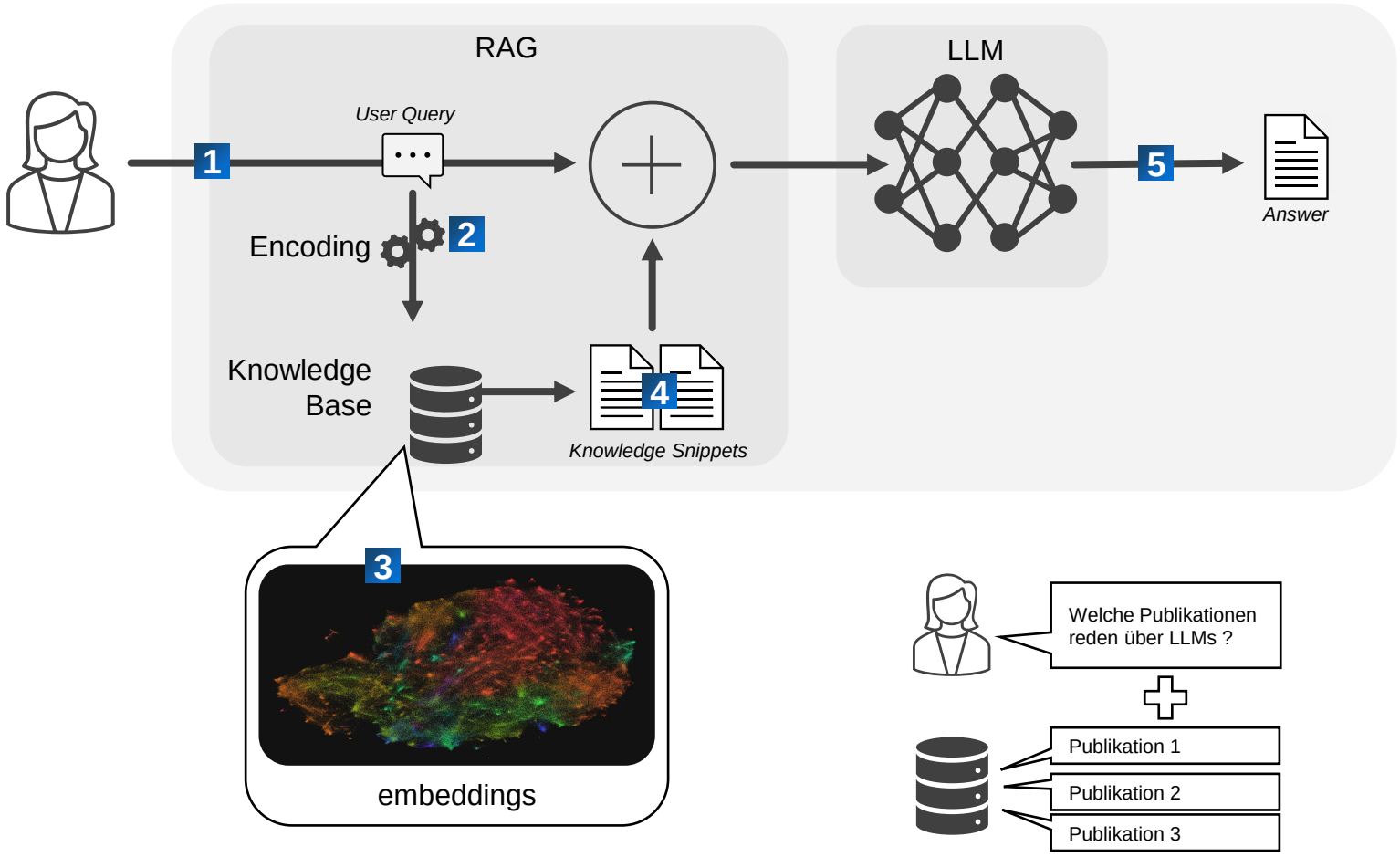
1 Anwender fordert Wissen an

2 Nutzeranfrage wird in einen semantischen Vektor encoded (Embedding)

3 Vektor wird mit embedded Einträgen aus der Datenbank Verglichen

4 Embedding wird verwendet, um Ähnliche Vektoren zu finden, die passenden Einträge werden dem LLM zu der Frage mitgegeben

5 **LLM** erzeugt Antwort mit erweitertem Wissen.

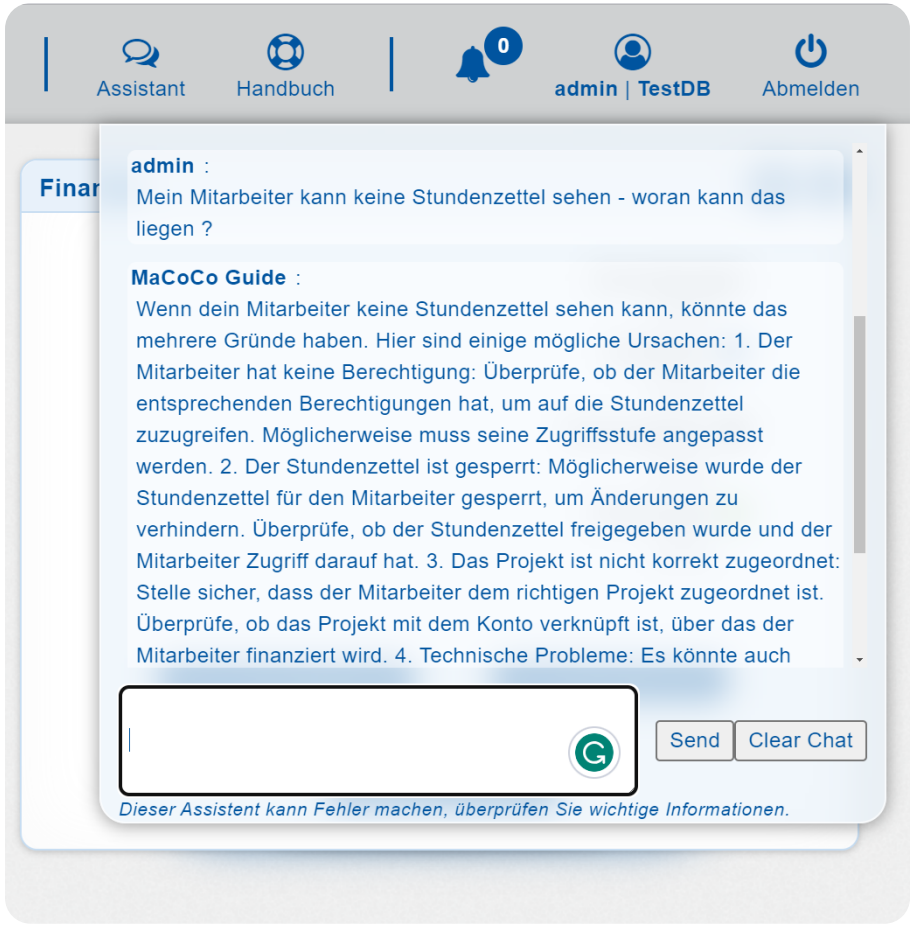


Dokumentation mit LLMs Anwendern verfügbar machen

Einbindung von Dokumentation in Anwendungen

Integration von Guidelines und Tutorials in Informationssystem mit dem RAG-Ansatz

- Nutzer werden können auf **ihrem Level** mit dem Wissen interagieren
- Nutzer haben weniger Scheu **‘dumme’ Fragen** zu stellen.
- **Analysedaten** zeigen an welchen Stellen in Dokumentation Lücken sind und welche stellen ggf. in Einweisungen oder der Dokumentation ausgebessert werden sollten.



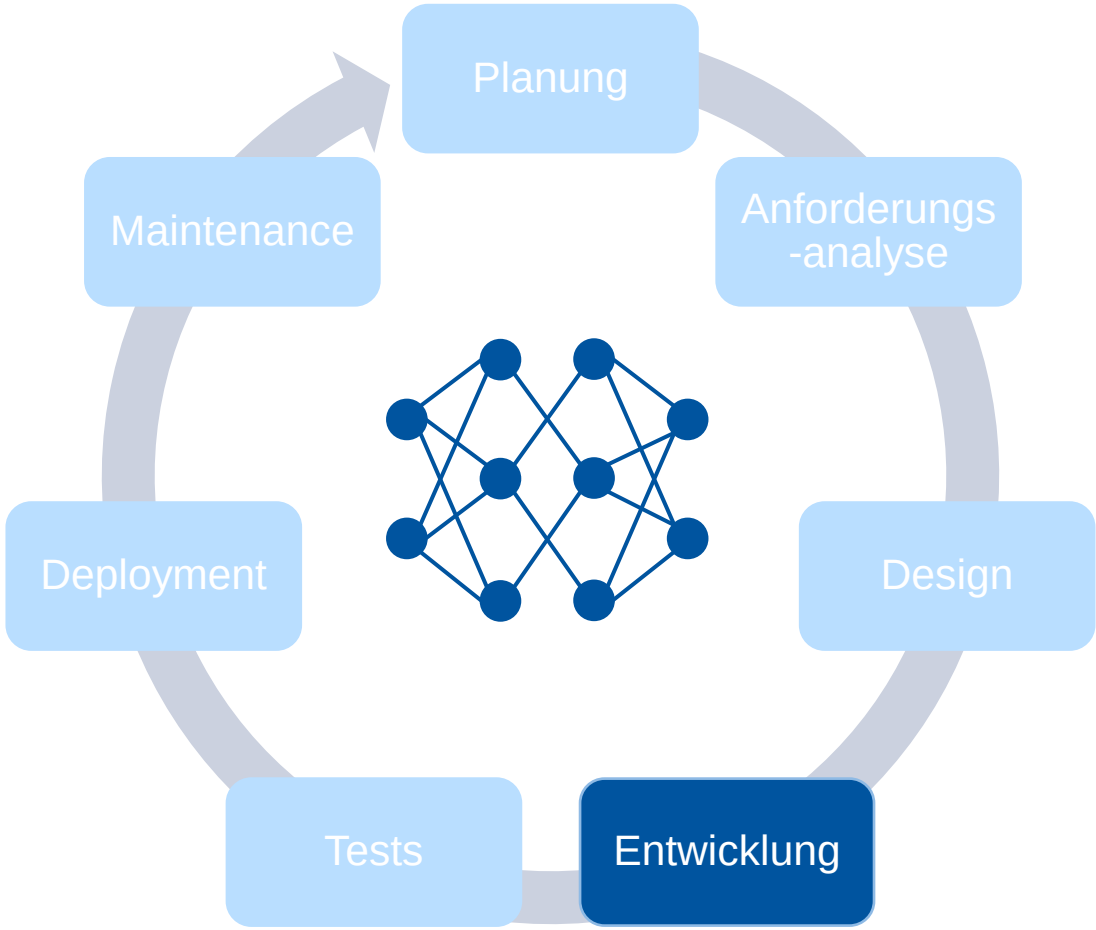
Einsatz in der Lehre zu Vorlesungsspezifischen Wissen

Entwicklung

Implementierung von Software mit LLMs

Coding Assistants

- GitHub Copilot, codium, CodeWhisperer …
- Implementieren nicht die gesamte Anwendung nur Teile davon
- Sehr gut in **populären Sprachen**,
  schlechter in Sprachen zu denen es wenige open-source Projekte gibt Automobil-branche: OTX, CAPL
- **DRY-Problematik** („Don‘t Repeat Yourself“)
  - Limitierung in Kontext der Berücksichtigt werden kann
  - Bereits ex. Implementierungen sind dem LLM ggf. unbekannt.
  - Für jede Anfrage wird eine NEUE Lösung implementiert



Programmieren mit LLMs

Verwendung von LLMs um Source Cope zu Produzieren

Zeitersparnis **Acceleration & Exploration** [8]

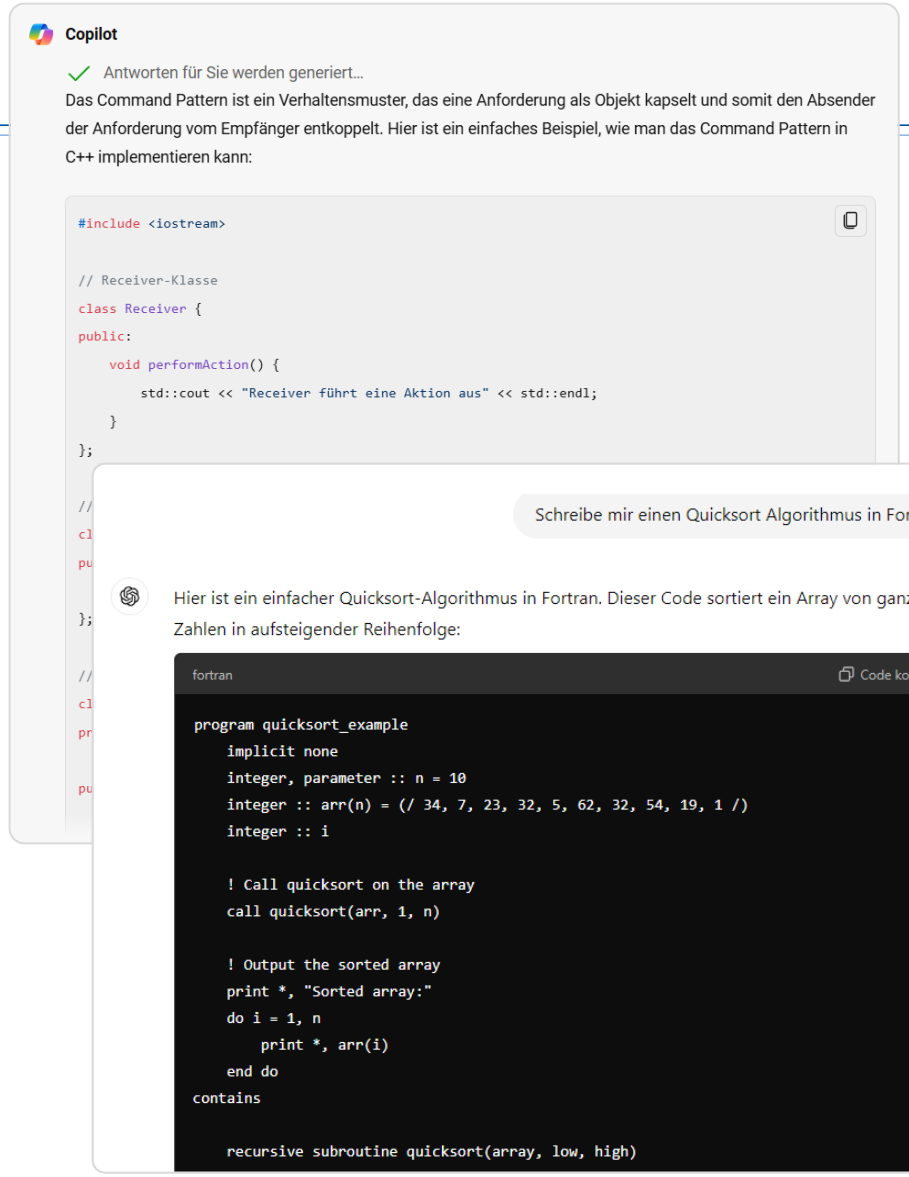
- IDE-Integration von LLMs
- Lösung **anpassen** ggf. schneller als Lösung **komplett neu** zu definieren.

LLM-Trainingsdaten unterliegen **keiner Code-Review**

- StackOverflow: Antwort mit den meisten Upvotes ist nicht immer die Korrekte/ Effizienteste
- Mittelmäßige / Ineffiziente Implementierungen
- Sicherheitslücken

Überprüfung / **Validierung** aller Ausgaben eines LLMs zwingend notwendig

- Anwendung bekannter QA-Methoden:
- Code Review
- Automatisierte Tests
- Test Driven Development



[8] Barke, S., James, M.B. and Polikarpova, N., 2023. Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, *7*(OOPSLA1), pp.85-111.

Tests erstellen mit LLMs

Beispiel zur Verwendung von LLMs zum Testen von Software

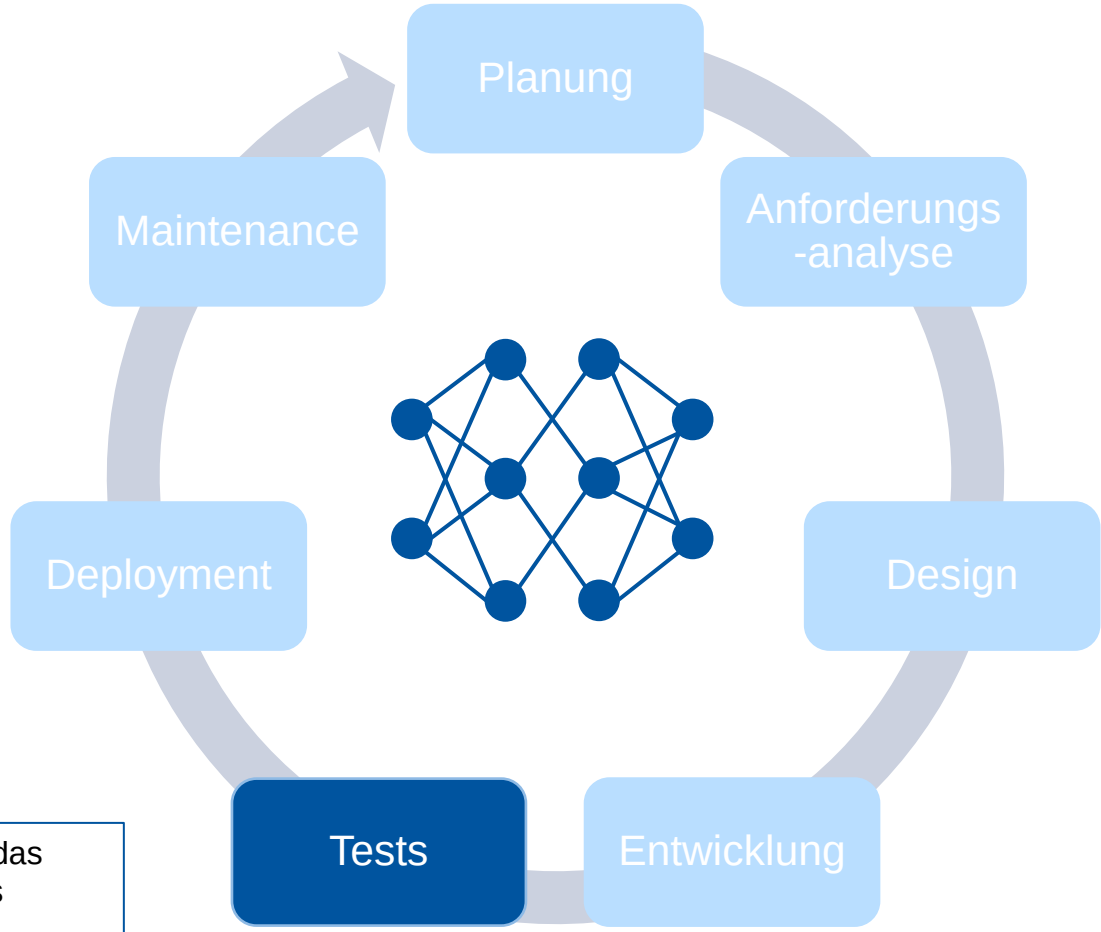
Tests können ebenfalls mit CodingAssistants erstellt werden.

Synthetische Testdaten

- LLMs können sehr gut verwendet werden, um User-Input zu simulieren
- Test-szenarien Erstellung
- Ausnutzung von Kreativ-Schöpferischen Eigenschaften des LLMs

Erstellung von Tests

- Keine Garantie der Vollständigkeit von generierten Tests
- Keine Garantie von Abdeckung von Randfällen
- Keine Garantie von Richtigkeit der Tests



LLM-Basierte Tests ersetzen nicht Handgeschriebene Tests und sollten nicht das alleinige verfahren darstellen um Code zu validieren der ebenfalls durch LLMs erzeugt wurde

Unterstützung im Deployment

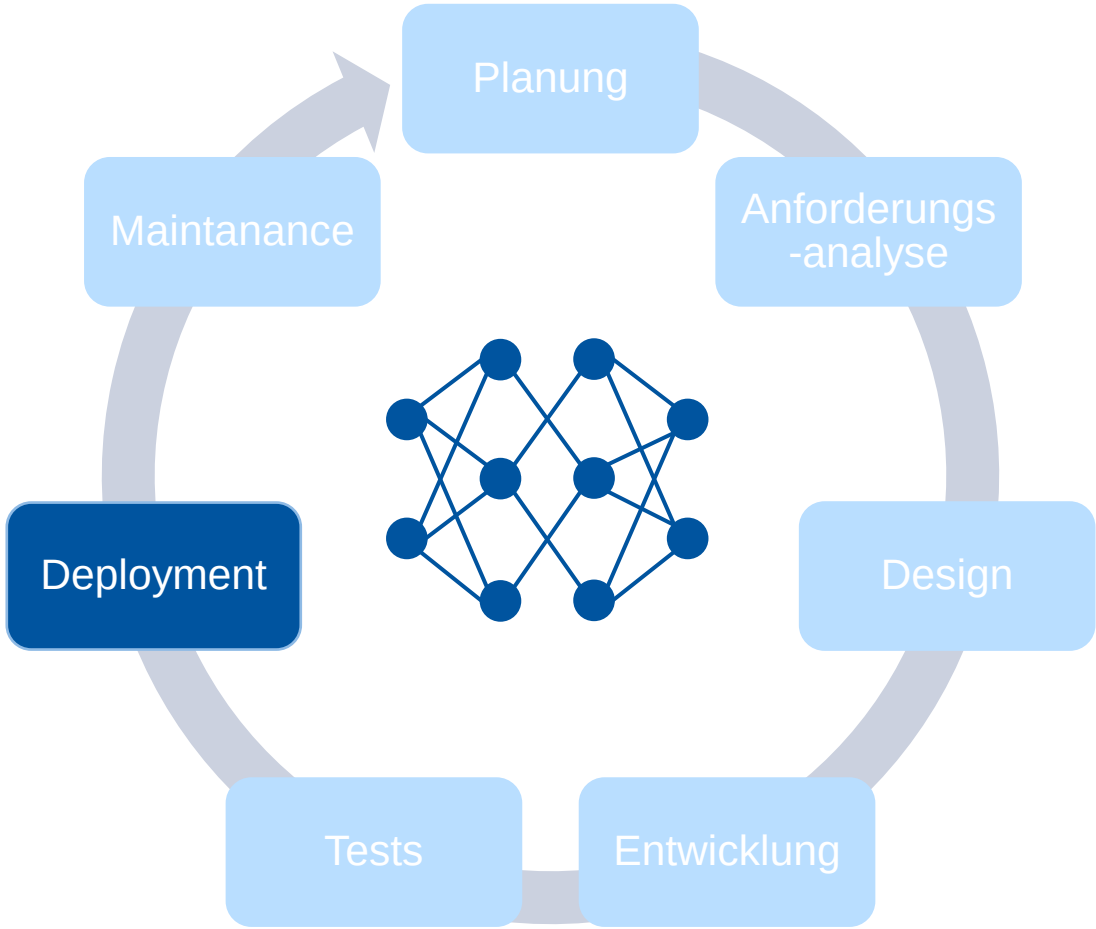
Einsatz von LLMs im Deployment

Automatische Erzeugung von Deployment-Skripten

Besser: Anpassung von existierenden Skripten

- Yaml und conf-Dateien
- CI-Pipeline Definition
- Cron jobs

LLMs sind sehr gut in der **Anpassung von Konfigurationen** zu bestimmten Systemen und Frameworks wenn Passende Dokumentation öffentlich verfügbar ist.



Sicherheitsrelevante Aspekte beachten!
Ergebnisse und Anpassungen Prüfen

Keine Passwörter an das LLM senden …

Unterstützung in Maintenance

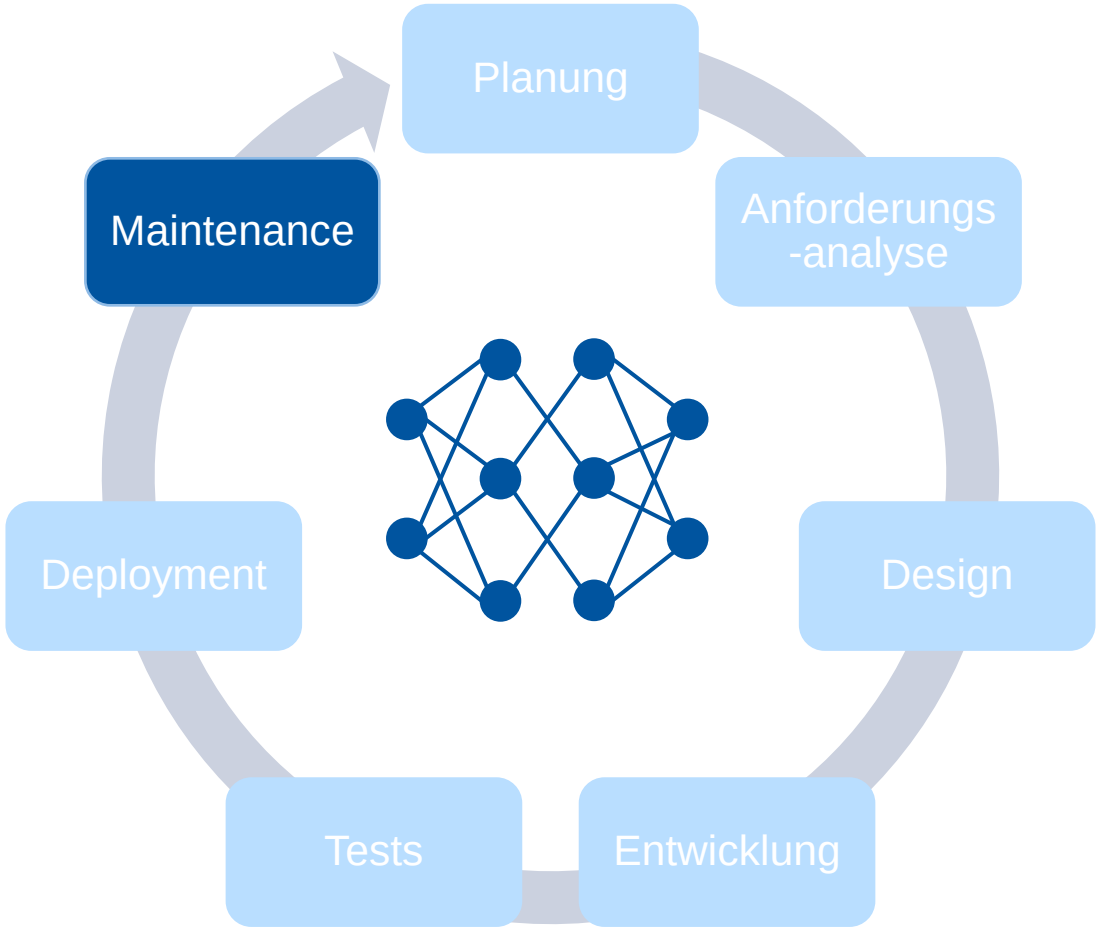
Beispiele für den Einsatz von LLMs in der Wartung und Instandhaltung von Software-Projekten

Automatisierte Code Dokumentation
- Generierung von Kommentaren
  ```
  String getName() // gets name
  ```

- Generierung von Dokumentation
- Erläuterung des zwecks bestimmter code Abschnitte

Erkennung von Anomalien und Optimierung von Leistung
- LLMs können bad-practices erkennen und Optimierungen empfehlen
- Code Smells (SonarCube)

Code-Diff und Änderungsverfolgung
- LLMs können Änderungen analysieren und deren Auswirkungen erläutern
- Semantisches Durchsuchen von Historien



Verwendung von Embeddings um Code zu durchsuchen

Semantische Codeanalyse

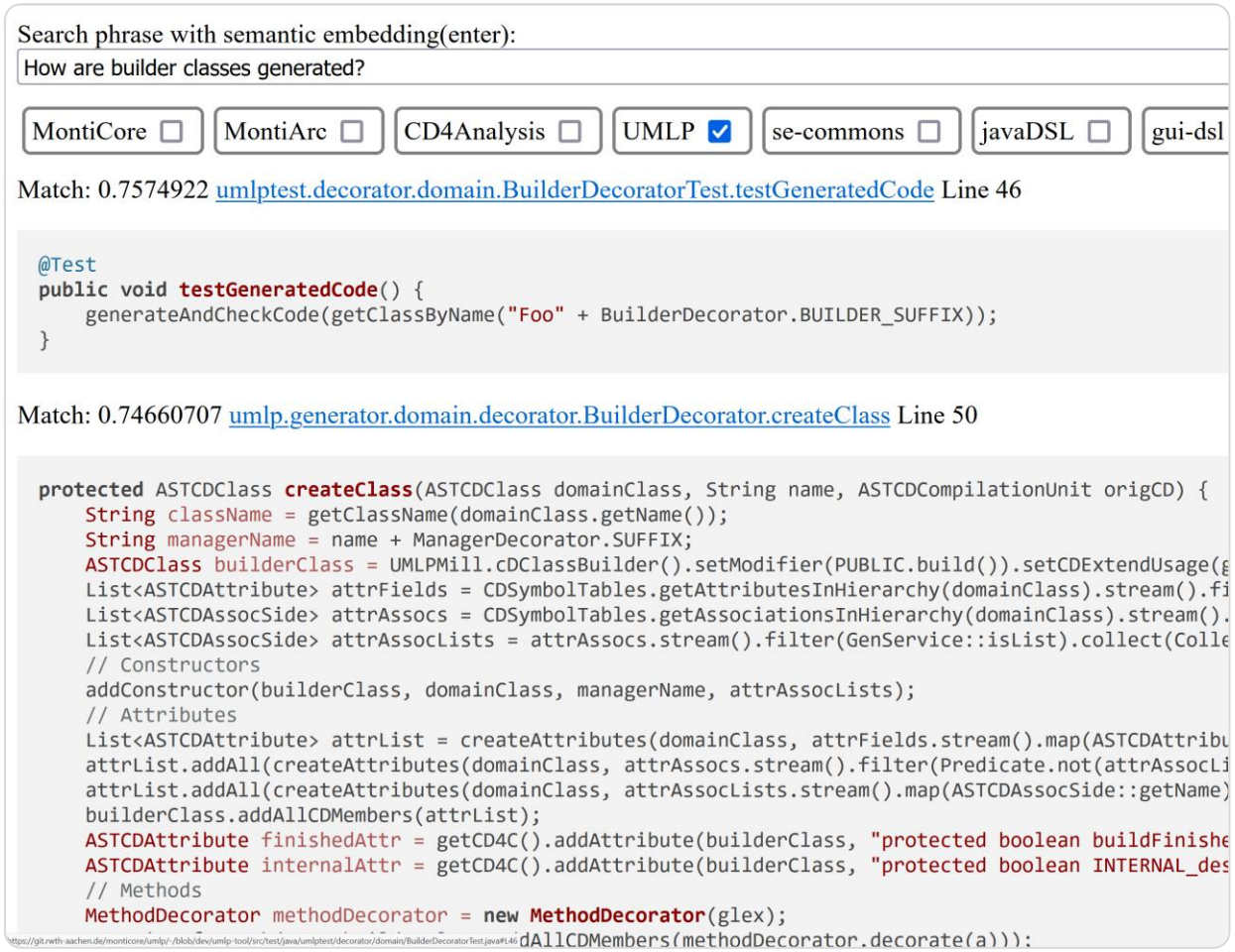
Auffinden von relevanten Code-Abschnitten über eine Semantische Suche (keine Stichwortsuche)

Wo wird was im Code umgesetzt ?

Frage

Ergebnis 1

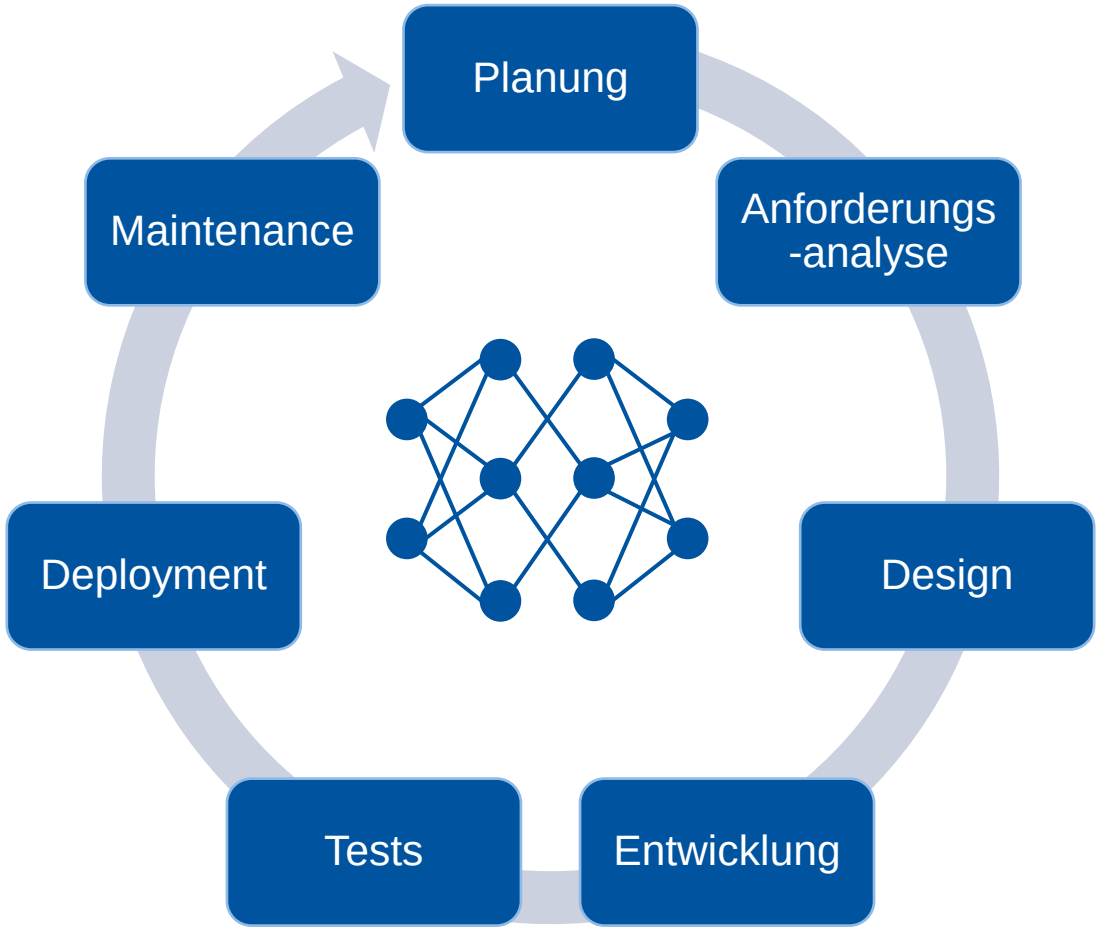
Ergebnis 2



Verwendung von KI in der Software Entwicklung

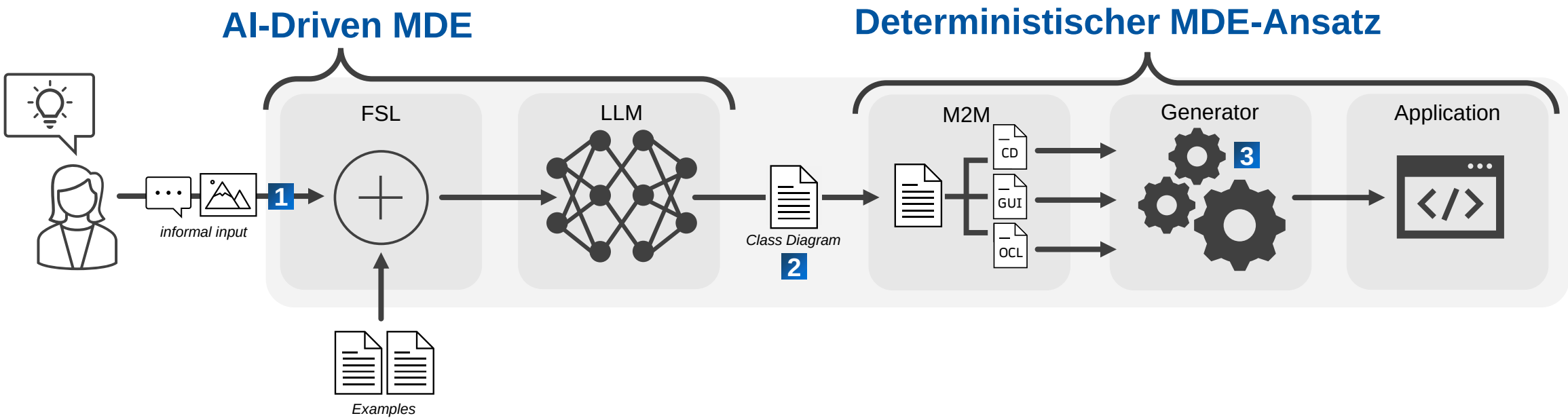
KI im Software Development Life Cycle

- KI kann einen **Entwickler noch nicht ersetzen** und sollte eher als **Werkzeug**, welches den Entwicklungsprozess beschleunigt betrachtet werden.
- Aufgaben die KI enthalten brauchen **Prüfung und Validierung**
- KI kann nicht '**blind**' verwendet werden.



Beispiel : LLMs zur Generierung von Informationssystemen

1. Nutzer kann informell seine Anwendungsdomäne beschreiben
2. Aus Domänenbeschreibungen werden Klassendiagramme abgeleitet
3. Generator erstellt aus Modellen ein vollwertiges Informationssystem



(1) Michael, Judith, et al. "Generating digital twin cockpits for parameter management in the engineering of wind turbines." *Modellierung 2022* (2022).
(2) Gerasimov, Arkadii, et al. "Agile generator-based GUI modeling for information systems." *Modelling to Program: Second International Workshop, M2P 2020, Lappeenranta, Finland"*
(3) Buschhaus, Constantin, et al. "Lessons learned from applying model-driven engineering in 5 domains: The success story of the MontiGem generator framework." *Science of Computer Programming* 232 (2024)

Sicherheitsrisiko LLM

- LLMs können überzeugt/ bestochen und überredet werden, ihre Konfiguration preis zu geben und **nicht vorgesehen Ausgaben** zu produzieren!
  - Unethische Aussagen
  - Für den Anwender Schadhafte Aussagen (zB. „Lösche den Ordner System32“)

- LLMs können hierdurch **interne Daten preisgeben!**
  - Integration von LLMs in Email-Versandprozesse
  - Anbindung von LLMs an Datenbanken

- LLMs selbst können ein **Angriffsziel** sein
  - Nachtrainierte LLMs enthalten Unternehmensdaten und sind damit ein beliebtes Ziel

- Manipulation von LLMs durch **Zeitlich-Beschränkte** Manipulation von **Trainingsdaten**
  - Wenn Zeitpunkt des Trainings bekannt ist, können Trainingsdaten (z.B. Wikipedia) gezielt angepasst werden, um LLM zu manipulieren
  - Coding Assistant →Verwendung von unsicheren Frameworks
  - Beratung → Empfehlung bestimmter Produkte

Software Entwicklung mit LLMs an der Universität

Lehrstühle müssen viele Hürden überwinden

Bürokratische Hürden

- Datenschutz
- Komplizierte Abrechnungen / Einkauf
- Kostenmanagement auf Individueller Anwenderbasis
- Hardwareeinkauf
  - Eigene Server
  - Rechenzentren
  - Cloudanbieter

Technologiehürden

- Schnelllebige Frameworks und Trends
- Unklarer Stand der Dinge
- Mangelnde Expertisen unter den Studierenden

LLM-Cafe

Austauschforum zu LLMs

Möglichkeit den Umgang mit LLMs im Universitätsalltag

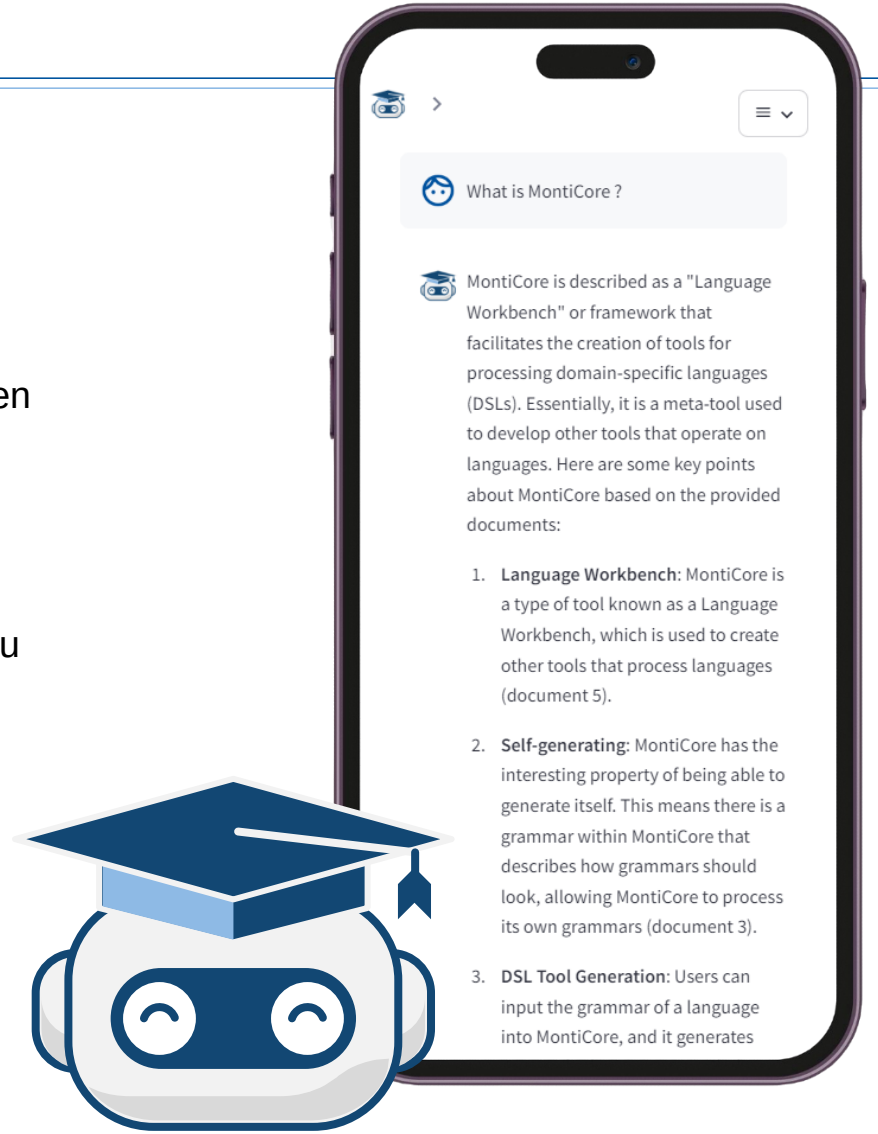
- Umgang mit dem Einsatz von LLMs für **Hausarbeiten/ Abschlussarbeiten**
- Bewältigung **Bürokratischer Hürden**
- **Erfahrungsaustausch** zu Frameworks, Methodiken und Modellen und deren Einsatz in der Forschung und Lehre
- Interdisziplinärer Austausch zu möglichen **Anwendungsfällen** – wo kann man LLMs gut einsetzen Wo nicht

LLM Cafe 10.10.24

LLM-Lernplatform

Entwicklung einer RWTH-weiten KI-Basierten Lehrplattform

- Dozenten und Wissenschaftliche Mitarbeiter reichern LLM mit **Vorlesungsinhalten** an
- Dozenten und Wissenschaftliche Mitarbeiter können **Werkzeuge und Modi** implementieren und zur Verfügung stellen
  - Simulation von Mündlichen Prüfungen
  - Modell-Generator
  - WolframAlpha Integration
- Studierende und Wissenschaftler können Plattform nutzen, um sich fortzubilden und um zu lernen
  - Vorlesungsspezifische Fragen
  - Durchsuchen von Publikationen des Lehrstuhls



THANK YOU, LET‘S TALK!