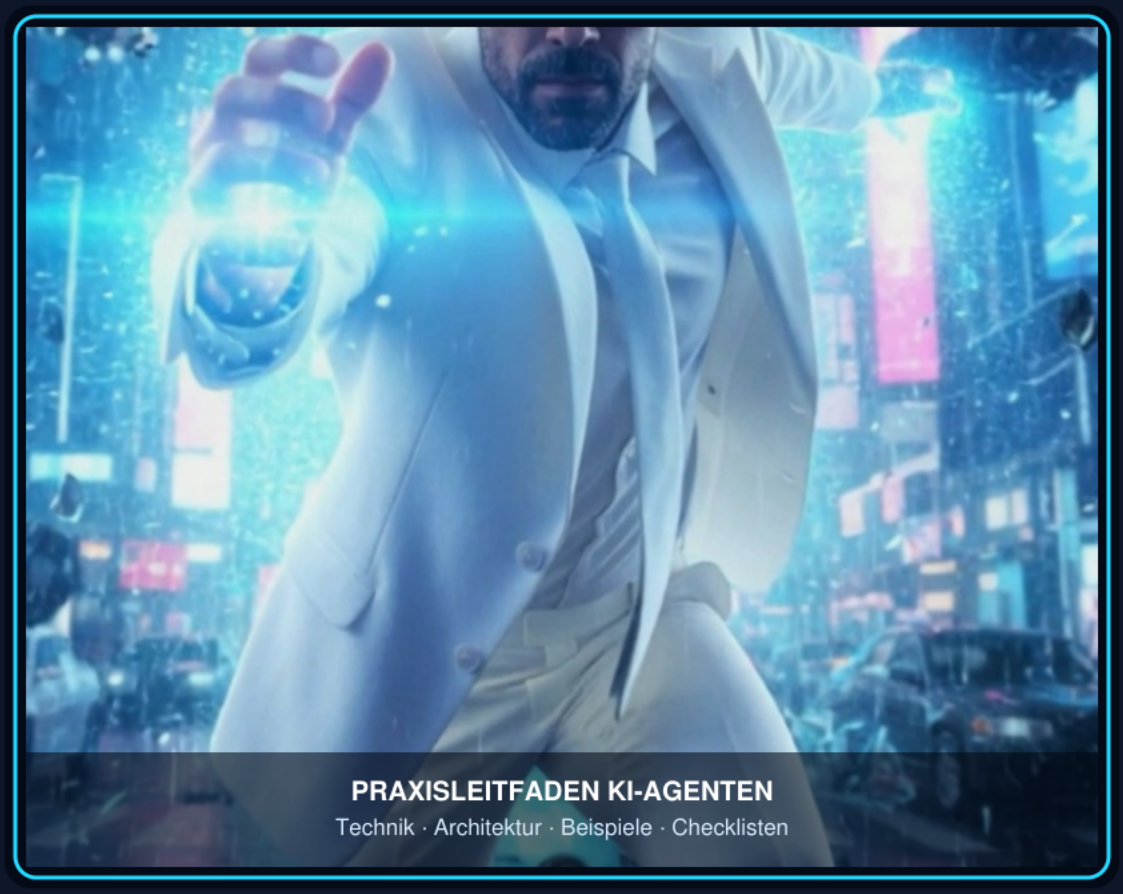
WIE BAUE ICH EINEN KI-AGENTEN?

Von der Idee zum eigenen, handlungsfähigen KI-Agenten



Autor: Maik Heidemann

Ein praxisnahes Einsteiger-E-Book mit Technik, Beispielen, Architektur und Checklisten

Inhalt

1. Was ist ein KI-Agent?

Der Begriff Agent klingt kompliziert, beschreibt aber im Kern eine einfache Idee: Die KI bekommt nicht nur eine Frage, sondern eine Aufgabe. Sie darf anhand ihrer Regeln entscheiden, welche Informationen sie benoetigt, welches Werkzeug sie einsetzen muss und wann die Aufgabe als abgeschlossen gilt.

Ein klassischer Chatbot reagiert auf eine Nachricht. Ein Agent arbeitet zielorientierter. Er kann beispielsweise zuerst Daten lesen, danach einen Rechenschritt durchfuehren, anschliessend eine Datei erzeugen und am Ende das Ergebnis erklaeren.

Baustein	Einfache Erklaerung
Ziel	Was soll erreicht werden?
Modell	Welche KI verarbeitet Sprache und entscheidet?
Anweisungen	Welche Rolle, Regeln und Grenzen gelten?
Werkzeuge	Welche externen Aktionen darf die KI ausfuehren?
Wissen	Welche Fachinformationen stehen zur Verfuegung?
Memory	Welche Informationen darf der Agent ueber mehrere Schritte behalten?
Zustand	Wo steht der Agent gerade im Prozess?
Kontrolle	Wann muss ein Mensch pruefen oder freigeben?

Ein einfaches Bild
Stell dir einen Agenten wie einen sehr schnellen Mitarbeiter vor. Das Sprachmodell ist sein Denkwerkzeug, die Tools sind seine Haende, die Knowledge Base ist sein Fachordner und die Regeln sind sein Arbeitsauftrag.

Was macht einen Agenten zum Agenten?

- Er verfolgt ein Ziel statt nur eine einzelne Nachricht zu beantworten.
- Er kann mehrere Schritte nacheinander ausfuehren.
- Er kann Werkzeuge benutzen.
- Er kann Zwischenergebnisse auswerten und darauf reagieren.

Was ein Agent nicht ist

Ein Agent ist keine mystische neue Art von Intelligenz. In den meisten praktischen Systemen ist er eine Softwarearchitektur rund um ein Sprachmodell. Genau deshalb laesst sich ein Agent planen, testen, versionieren und kontrollieren.

2. Was kann ein Agent?

Die Moeglichkeiten eines Agenten werden weniger durch den Chat selbst bestimmt als durch die Werkzeuge und Datenquellen, die du ihm gibst. Ein Agent kann nur das tun, was sein System technisch erreichen kann.

Aufgabe	Beispiel	Was der Agent dafuer braucht
Fragen beantworten	Kundenfrage erklaeren	Modell + Fachwissen
Rechnen	Materialmenge berechnen	Regelwerk + Rechner-Tool
Dateien erzeugen	PDF oder Bericht erstellen	Daten + Dateitool
Daten zuordnen	Anfrage zu Angebot zuordnen	Datenzugriff + Regeln
E-Mails verarbeiten	Postfach sortieren	E-Mail-Tool + Freigaberegeln
Bestand ueberwachen	Lagerbestand pruefen	Datenbank/API + Schwellenwerte
Aktionen ausloesen	Nachbestellung vorbereiten	API + Sicherheitsfreigabe
Bilder analysieren	Baustellenfoto auswerten	Vision-Modell + klare Zonen

Der wichtigste Unterschied: Ausfuehren statt nur Antworten

Nehmen wir eine einfache Frage: "Wie viele Fliesen brauche ich?" Ein Chatbot antwortet vielleicht mit einer Berechnung. Ein Agent kann zusaetzlich die Raumflaeche aus vorhandenen Daten lesen, einen Verschnittfaktor anwenden, einen Lagerbestand pruefen und daraus eine Bestellliste vorbereiten. Genau dort entsteht der praktische Mehrwert.

> **Grenze**
> Je mehr ein Agent ausfuehren darf, desto wichtiger werden Berechtigungen, Protokollierung und Freigaben. Ein Agent sollte nicht automatisch alles duerfen, nur weil er technisch darauf zugreifen kann.

3. Chatbot, Agent oder Automatisierung?

Diese Begriffe werden oft verwechselt. Eine gute Architektur entsteht aber erst, wenn du sie auseinanderhaeltst.

System	Typischer Ablauf	Staerke
Chatbot	Frage -> Antwort	Kommunikation
Automatisierung	Wenn X passiert -> tue Y	Zuverlaessige Wiederholung
KI-Agent	Ziel -> beurteilen -> Werkzeug waehlen -> Ergebnis pruefen	Flexible Problembearbeitung
Multi-Agent-System	Aufgabe -> Spezialagenten -> Koordination -> Ergebnis	Arbeitsteilung

Ein guter Agent ersetzt eine feste Automatisierung nicht automatisch. Wenn eine Aufgabe immer identisch ist, ist eine klassische Automatisierung oft einfacher und robuster. Ein Agent ist besonders stark, wenn unstrukturierte Informationen und wechselnde Situationen verarbeitet werden muessen.

4. Aus welchen Bausteinen besteht ein Agent?

Bevor du programmierst, zeichne den Agenten als System. Genau dieser Schritt spart spaeter viel Arbeit.

Baustein	Frage, die du beantworten musst	Beispiel
Rolle	Wer ist der Agent?	Bauexperte fuer Angebotspruefung
Ziel	Was soll er erreichen?	Plausibles Angebot vorbereiten
Modell	Welche KI wird genutzt?	Ein passendes Sprachmodell
Systemregeln	Was darf er, was nicht?	Keine Preisfreigabe ohne Menschen
Tools	Welche Aktionen kann er ausfuehren?	Datenbank, PDF, Mail
Wissensquellen	Welche Fakten darf er verwenden?	Handwerkerwissen + Firmenregeln
Memory	Was muss er behalten?	Aktueller Auftrag + Kontext
Review	Wann braucht er einen Menschen?	Unsicherheit oder hohe Auswirkung
Logging	Was wird gespeichert?	Aktion, Ergebnis, Fehler, Zeit

Der Agent als Schichtenmodell

```
ZIEL
  |
  v
ROLLE + REGELN
  |
  v
KI-MODELL <---- WISSEN / KNOWLEDGE BASE
  |
  +----> TOOL A: Daten lesen
  +----> TOOL B: Rechnen
  +----> TOOL C: PDF erzeugen
  +----> TOOL D: E-Mail vorbereiten
  |
  v
PRUEFUNG / FREIGABE
  |
  v
ERGEBNIS + LOG
```

5. Wie baue ich einen Agenten - Schritt fuer Schritt?

Jetzt bauen wir gedanklich den ersten Agenten. Du kannst diese Reihenfolge fuer kleine und grosse Projekte verwenden.

Schritt 1 - Eine einzige Hauptaufgabe definieren

Formuliere nicht "Der Agent soll alles koennen". Formuliere ein klares Ergebnis, zum Beispiel: "Pruefe eine eingehende Anfrage auf Vollstaendigkeit und bereite die fehlenden Informationen als Fragen vor."

Schritt 2 - Erfolg messbar machen

Lege fest, woran du erkennst, dass der Agent gut gearbeitet hat. Beispiel: Alle Pflichtinformationen erkannt, keine erfundenen Werte, offene Fragen klar aufgelistet.

Schritt 3 - Rolle und Grenzen schreiben

Schritt 4 - Wissensquellen festlegen

Entscheide, welche Dokumente, Datenbanken oder Regeln als Quelle dienen.

Schritt 5 - Tools festlegen

Gib nur die Werkzeuge frei, die fuer die Aufgabe wirklich gebraucht werden.

Schritt 6 - Den Agenten-Loop bauen

Der Agent muss Anfrage -> Plan -> Tool -> Ergebnis -> Entscheidung -> naechster Schritt abarbeiten koennen.

Schritt 7 - Fehlerpfade bauen

Definiere, was passiert bei fehlenden Daten, widerspruechlichen Daten oder Toolfehlern.

Schritt 8 - Testfaelle erstellen

Baue normale, schwierige, fehlerhafte und missverstaendliche Testfaelle.

Schritt 9 - Menschliche Freigabe definieren

Kritische Aktionen brauchen eine Freigabegrenze.

Schritt 10 - Erst danach produktiv schalten

Zuerst beobachten, dann kontrolliert freigeben, dann skalieren.

Ein minimalistischer Agent - vereinfacht

```
def agent(task):
    context = load_relevant_knowledge(task)
    plan = model.decide(task, context, tools)

    while not plan.finished:
        result = run_tool(plan.tool, plan.arguments)
        plan = model.decide_next_step(task, context, result)

    return plan.final_answer
```

Der Code ist bewusst vereinfacht. Das Grundprinzip ist aber wichtig: Der Agent bekommt eine Aufgabe, ruft Wissen ab, entscheidet einen Schritt, benutzt ein Werkzeug, bewertet das Ergebnis und faehrt fort, bis die Aufgabe erledigt oder eine Freigabe erforderlich ist.

Was in einen guten System-Prompt gehoert

- Rolle und Fachgebiet
- Ziele und Prioritaeten
- Was der Agent niemals behaupten oder tun darf
- Wann nachgefragt werden muss
- Wann ein Tool verwendet werden soll
- Welche Aktionen eine menschliche Freigabe brauchen
- Wie Unsicherheit und Fehler gemeldet werden

Regel fuer Einsteiger
Ein kurzer, klarer Auftrag plus saubere Werkzeuge ist meistens besser als ein riesiger Prompt, der jede denkbare Situation beschreiben will.

6. Werkzeuge und Function Calling

Ein Tool ist eine definierte Funktion, die der Agent aufrufen darf. Das Modell selbst veraendert nicht automatisch deine Datenbank oder verschickt eine E-Mail. Die Anwendung stellt ihm eine kontrollierte Schnittstelle zur Verfuegung.

Ein Tool sollte einen klaren Namen, eine klare Beschreibung und ein strukturiertes Eingabeschema haben. Je genauer dieses Schema ist, desto weniger Interpretationsfehler entstehen.

```
Tool: generate_3d_room_model
Zweck: Erzeuge ein 3D-Modell aus vier Wänden.
Eingabe:
- wall_id: Zahl 1-4
- length_m: Zahl
- height_m: Zahl
- area_m2: optional / ableitbar
Regel:
- Genau vier Wände erforderlich.
- Fehlende Werte -> Rueckfrage.
- Unplausible Werte -> nicht ausfuehren, melden.
```

Eine gute Tool-Definition ist fast wie eine Arbeitsanweisung fuer eine Fachkraft. Sie sagt nicht nur, was das Werkzeug heisst, sondern auch, wann es benutzt werden darf und welche Daten erforderlich sind.

Tool-Design: lieber klein und sicher

- Lesende Tools getrennt von schreibenden Tools behandeln.
- Riskante Aktionen mit expliziter Freigabe absichern.
- Parameter validieren, bevor die Funktion ausgefuehrt wird.
- Ergebnisse strukturiert zurueckgeben, nicht nur als langen Text.
- Jede Tool-Nutzung protokollieren.

7. Wissen, Memory und Knowledge Base

Hier passieren viele konzeptionelle Fehler. Wissen ist nicht dasselbe wie Memory, und beides ist nicht dasselbe wie der aktuelle Chat-Kontext.

Begriff	Wofuer?	Beispiel
Kontext	Aktuelle Unterhaltung	Die letzten Nachrichten
Memory	Gezielt gemerkte Informationen	Bevorzugte Arbeitsweise
Knowledge Base	Gepruefte Fachinformationen	Regelwerk, Handbuch, Normen, interne Regeln
Datenbank	Operative Unternehmensdaten	Kunden, Auftraege, Lagerbestand
Log	Nachvollziehbarkeit	Welche Aktion wurde wann ausgefuehrt?

Retrieval statt Prompt aufblasen

Eine wachsende Sammlung von Regeln sollte nicht jedes Mal komplett in den System-Prompt kopiert werden. Besser ist ein Abrufmechanismus: Der Agent sucht gezielt die relevanten Regeln und bekommt nur diese in den aktuellen Kontext.

```
Anfrage
 -> relevante Begriffe erkennen
 -> Knowledge Base durchsuchen
 -> passende Regeln holen
 -> Antwort / Aktion mit Quellenbezug erzeugen
```

Versionierte Regeln

Wenn Wissen veraendert wird, sollte die neue Fassung eine Version bekommen. So bleibt nachvollziehbar, was wann gegolten hat und warum eine spaetere Version eingefuehrt wurde.

Feld	Beispiel
Regel-ID	REG-027
Version	v2
Titel	Nettoflaeche bei mehreren Oeffnungen
Inhalt	Gepruefte Rechenregel

Feld	Beispiel
Freigegeben von	Maik Heidemann
Datum	01.09.2026
Status	Aktiv

Wichtig
Eine Knowledge Base sollte fuer fachliche Wahrheit stehen. Ungepruefte Chat-Aussagen oder widerspruechliche Nutzerbeispiele gehoeren nicht ungefiltert hinein.

8. Planung, Entscheidungen und Agenten-Loop

Ein Agent arbeitet normalerweise nicht in einem einzigen Schritt. Er braucht einen wiederholbaren Loop.

```
1. Aufgabe verstehen
2. Relevantes Wissen holen
3. Naechsten sinnvollen Schritt bestimmen
4. Tool aufrufen oder Rueckfrage stellen
5. Ergebnis pruefen
6. Entscheiden: weiter / fertig / Mensch
7. Ergebnis ausgeben
8. Aktion protokollieren
```

Warum der Loop wichtig ist

Ohne Loop bleibt die KI oft bei einer plausiblen Textantwort stehen. Mit Loop entsteht Handlungsfaehigkeit. Gleichzeitig entsteht aber ein neues Risiko: Endlosschleifen oder falsche Aktionen. Deshalb braucht der Loop Maximalzahlen, Abbruchbedingungen und klare Freigaben.

Kontrolle	Beispiel
Maximale Schritte	z.B. 8 Tool-Aufrufe pro Aufgabe
Timeout	Werkzeug darf nicht unendlich warten
Validierung	Eingaben muessen Schema erfuellen
Abbruch	Widerspruch oder fehlende Information
Freigabe	Kauf, E-Mail-Versand, Loeschen
Rollback	Aenderung rueckgaengig machen, wenn moeglich

9. Testen, Fehlerbehandlung und Sicherheit

Ein Agent ist erst dann produktionsbereit, wenn du nicht nur die guten Faelle, sondern auch die schlechten Faelle kennst.

Die vier Testgruppen

Testgruppe	Was wird geprueft?
Normal	Funktioniert der Standardfall?
Grenzfall	Was passiert bei sehr kleinen/grossen oder ungewoehnlichen Werten?
Fehlerfall	Was passiert bei fehlenden, falschen oder widerspruechlichen Daten?
Missbrauch	Was passiert bei unzulaessigen oder manipulativen Anweisungen?

Objektive Pruefung statt Selbstvertrauen der KI

Eine KI sollte nicht allein entscheiden duerfen, wie sicher ihre eigene Antwort ist. Besser sind nachvollziehbare Kriterien: Sind Informationen vorhanden? Gibt es Widersprueche? Gibt es eine bekannte Regel? Ist das Ergebnis plausibel? Gibt es historische Fehler? Ist der Fall eindeutig? Wird menschliche Entscheidung benoetigt?

Lernfaelle aus echten Fehlern

Wenn ein Agent in der Praxis scheitert, sollte daraus nicht automatisch ein neues "Training" entstehen. Sinnvoller ist ein kontrollierter Lernfall: Problem sichern, Ursache analysieren, Regel oder Wissenseintrag formulieren, menschlich pruefen, testen und erst dann freigeben.

```
Produktionsfall
  -> Problem erkannt
  -> Lernfall erzeugen
  -> Analyse
  -> Regel / Wissen entwerfen
  -> menschliche Freigabe
  -> Testfaelle
  -> Version aktivieren
  -> Agent nutzt neue Version
```

Sicherheitsgrenzen

- Keine geheimen Zugangsdaten in Prompts oder Chatnachrichten.
- Rechte nach dem Prinzip "so wenig wie noetig" vergeben.
- Schreibende Aktionen von lesenden Aktionen trennen.
- Kritische Aktionen immer protokollieren.
- Kunden- und Firmendaten nur fuer den vorgesehenen Zweck verwenden.
- Bei Unsicherheit lieber stoppen oder nachfragen als erfinden.

10. Ein echter Praxis-Agent fuer BauAssistent Pro

Jetzt machen wir aus der Theorie einen konkreten Agenten. Stell dir einen Agenten vor, der Baustellenaufnahme und Angebotsvorbereitung unterstuetzt.

Rolle

"Du bist ein fachlicher Assistenz-Agent fuer BauAssistent Pro. Du hilfst bei der strukturierten Aufnahme von Baustelleninformationen, dem Erkennen fehlender Angaben und der Vorbereitung von Angebotsdaten. Du erfindest keine Werte."

Faehigkeiten

- Sprachaufnahmen in strukturierte Angaben ueberfuehren.
- Baustellenfotos anhand definierter Bereiche auswerten.
- Raeume, Masse und Leistungen zuordnen.
- Materialien gegen einen vorhandenen Stamm abgleichen.
- Fehlende Pflichtinformationen erkennen.
- Entwuerfe fuer Angebot oder Rechnung vorbereiten.
- Bei unklaren Angaben gezielt nachfragen.

Beispiel fuer einen Auftrag

```
Nutzer:
"Im Bad 3,2 m lang, 2,4 m breit, Deckenhoehe 2,5 m.
 Zwei Fenster, eine Tuer. Fliesen 30x60. Angebot vorbereiten."

Agent:
1. Raumdaten erfassen
2. Flaeche berechnen
3. Oeffnungen separat pruefen
4. Material suchen
5. Fehlende Preisinformation erkennen
6. Angebot als Entwurf vorbereiten
7. Vor PDF-Ausgabe auf Freigabe warten
```

Warum ein Spezialagent besser sein kann

Ein allgemeines Modell kann viele Themen bedienen. Ein spezialisierter Agent kennt dagegen die genaue Arbeitsweise deiner Anwendung, die Begrifflichkeit und die Grenzen des Systems. Seine Qualitaet entsteht aus Rolle + Tools + Wissen + Tests - nicht allein aus dem Modellnamen.

Die Verbindung zu KIAMSE OS

Eine Zentrale wie KIAMSE OS kann Agenten verwalten, schulen, vergleichen und neue Wissensregeln kontrolliert freigeben. BauAssistent Pro ist dann die Produktionsumgebung, in der die Agenten echte Arbeitsaufgaben bearbeiten.

```
KIAMSE OS
 -> Agenten definieren
 -> Schulungen
 -> Wissensversionen
 -> Tests
 -> Freigaben
        |
        v
BAUASSISTENT PRO
 -> reale Aufgaben
 -> Nutzerfeedback
 -> Problemfaelle
        |
        v
KIAMSE OS
 -> Lernfall pruefen
 -> Verbesserte Regel
 -> Neue Agentenversion
```

11. Mehrere Agenten: Das Agenten-Team

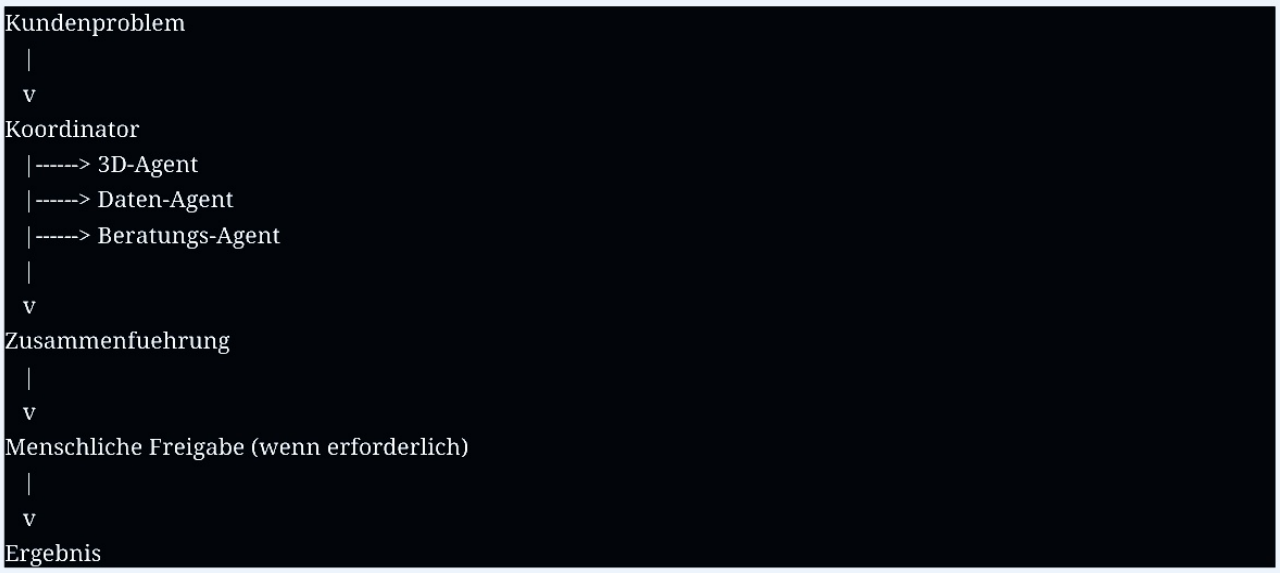
Wenn Aufgaben sehr unterschiedlich werden, kann ein Team aus Spezialisten sinnvoller sein als

Agent	Spezialisierung	Typische Aufgaben
Agent 1	Beratung	Analyse, Zweitmeinung, Entscheidungen
Agent 2	Software	Architektur, Code, Tests
Agent 3	Daten	Zuordnung, Strukturierung, Dokumente
Agent 4	Automatisierung	E-Mail, Termine, Nachbestellungen
Agent 5	3D / Planung	Raumdaten, Geometrie, Plausibilitaet
Agent 6	Qualitaet	Pruefung, Regeln, Fehlererkennung
Agent 7	Schulung	Trainingsaufgaben und Auswertung
Agent 8	Forschung	neue Methoden, Tools, Experimente

Der grosse Vorteil von Spezialisierung

Ein Agent muss nicht in allem der Beste sein. Er sollte in seiner Spezialrichtung besonders gut sein. Das macht die Bewertung einfacher und ermoeglicht klare Verantwortlichkeiten.

Agenten gemeinsam arbeiten lassen



12. Betrieb, Versionierung und Weiterentwicklung

Ein Agent ist kein Projekt, das man einmal baut und danach nie wieder anfasst. Modelle, Anforderungen, Tools und echte Nutzerfaelle veraendern sich. Deshalb braucht ein Agent Betrieb wie jede andere Software.

Versioniere mindestens diese Dinge

- Agentenrolle und Systemanweisungen
- Tool-Schemas
- Knowledge-Base-Regeln
- Testfaelle
- Freigabestatus
- Konfigurationen und Berechtigungen

Ein einfacher Freigabeprozess

15. Entwurf erstellen
16. Automatische Tests ausfuehren
17. Fehlerfaelle pruefen
18. Menschliche Freigabe
19. Neue Version aktivieren
20. Alte Version archivieren
21. Nach Produktion beobachten

Metriken, die wirklich helfen

Metrik	Warum wichtig?
Erfolgsquote	Wie oft wurde das Ziel erreicht?
Fehlerquote	Wie oft musste korrigiert werden?
Rueckfragen	Wie oft fehlen Informationen?
Tool-Fehler	Wie oft schlagen externe Aktionen fehl?
Kosten	Wie viel Modell- und Infrastrukturaufwand entsteht?
Zeit	Wie lange dauert eine Aufgabe?
Freigabequote	Wie oft wird ein Ergebnis menschlich korrigiert?

13. Typische Fehler und deren Loesung

Problem	Typische Ursache	Bessere Loesung
Agent antwortet zu allgemein	Rolle zu unscharf	Ziel + Fachbereich + Beispiele konkretisieren
Agent erfindet Fakten	Quelle nicht zwingend	Quellenprioritaet + Rueckfrage-Regel
Agent nutzt Tool zu oft	Kein Stopkriterium	Maximale Schritte + klare Tool-Beschreibung
Agent tut etwas Gefaehrliches	Zu viele Rechte	Least Privilege + Freigaben
Wissen wird widerspruechlich	Ungepflegte Quellen	Versionierte Knowledge Base
Prompt wird riesig	Alles wird hineinkopiert	Retrieval / gezielter Wissensabruf
Fehler sind nicht reproduzierbar	Keine Logs	Strukturiertes Logging
Neues Wissen macht alte Faelle schlechter	Keine Regressionstests	Feste Test-Suite pro Skill

Der Klassiker: "Ich gebe der KI einfach mehr Prompt"

Mehr Text ist nicht automatisch mehr Kompetenz. Ein Agent wird nicht stabiler, wenn Regeln, Beispiele und Sonderfaelle unkontrolliert wachsen. Qualitaet entsteht durch klare Grenzen, gute Werkzeuge, abrufbares Wissen, Tests und einen sauberen Entscheidungsprozess.

14. Checklisten und Glossar

Agent-Checkliste vor dem ersten Test

- [ ] Hauptaufgabe in einem Satz formuliert
- [ ] Erfolgskriterien festgelegt
- [ ] Rolle und Grenzen beschrieben
- [ ] Wissensquellen bestimmt
- [ ] Tools definiert
- [ ] Schreibende Aktionen abgesichert
- [ ] Fehlerpfade definiert
- [ ] Menschliche Freigabe festgelegt

Agent-Checkliste vor Produktion

- [ ] Normale Faelle bestanden
- [ ] Grenzfaelle bestanden
- [ ] Fehlerfaelle kontrolliert
- [ ] Missbrauchstests bestanden
- [ ] Keine geheimen Daten im Prompt
- [ ] Berechtigungen minimiert
- [ ] Notfall-/Abbruchweg vorhanden
- [ ] Version markiert und dokumentiert
- [ ] Rueckgaengige Aktivierung alter Version moeglich

Glossar

Begriff	Kurz erklaert
Agent	Software-System, das ein Ziel verfolgt und dafuer Modell, Wissen und Werkzeuge kombiniert.
System Prompt	Grundlegende Rolle, Regeln und Prioritaeten des Agenten.
Tool	Kontrollierte Funktion, die der Agent ausfuehren darf.
Function Calling	Strukturierter Mechanismus, mit dem ein Modell ein Tool anfordert.
Memory	Gezielt gespeicherte Informationen fuer spaetere Schritte.
Knowledge Base	Gepruefte Wissenssammlung, die der Agent gezielt abrufen kann.
Retrieval	Gezieltes Abrufen relevanter Wissensinformationen.
State	Aktueller Zustand eines laufenden Prozesses.
Logging	Nachvollziehbare Aufzeichnung von Aktionen und Ergebnissen.
Human-in-the-loop	Menschliche Freigabe innerhalb eines automatisierten Ablaufs.
Multi-Agent-System	Mehrere spezialisierte Agenten, die gemeinsam Aufgaben bearbeiten.

Begriff	Kurz erklaert
	nicht verschlechtern.

Der einfache Bauplan zum Mitnehmen

```
1. Ziel definieren
2. Erfolg messen
3. Rolle schreiben
4. Wissen organisieren
5. Tools bauen
6. Agenten-Loop bauen
7. Fehler und Stopkriterien definieren
8. Tests schreiben
9. Menschliche Freigaben einbauen
10. Versionieren und beobachten
11. Aus echten Fehlern Lernfaelle machen
12. Nur gepruefteS Wissen zurueckgeben
```

Schlussgedanke
Der wichtigste Schritt beim Bau eines Agenten ist nicht die Wahl des spektakulaersten Modells. Es ist die saubere Definition von Aufgabe, Wissen, Werkzeugen, Grenzen und Tests. Wer diese Architektur beherrscht, kann spaeter Modelle austauschen, Agenten spezialisieren und ganze Agenten-Teams aufbauen.

Ueber den Autor

Maik Heidemann beschaeftigt sich mit dem praktischen Einsatz von KI in Software, Automatisierung und Handwerksprozessen. Dieses E-Book verfolgt dabei einen bewusst praxisnahen Ansatz: weniger Magie, mehr nachvollziehbare Architektur.

Maik Heidemann