AI Agents – Grundlagen, Architektur und Praxis

Agentische KI-Systeme verstehen, entwickeln und kontrolliert einsetzen

01 INTRO

AI Agents erweitern Sprachmodelle um die Fähigkeit, selbstständig Arbeitsschritte zu planen, Werkzeuge aufzurufen, Informationen aus externen Systemen zu beschaffen und auf Ergebnisse ihrer Aktionen zu reagieren. Aus einem Modell, das Antworten erzeugt, wird damit ein System, das innerhalb definierter Grenzen Aufgaben bearbeiten und Aktionen ausführen kann.

Der Kurs vermittelt die grundlegenden Konzepte, Architekturen und Engineering-Prinzipien agentischer KI-Systeme. Die Teilnehmer entwickeln eigene Agenten, statten sie mit Werkzeugen und externem Wissen aus und untersuchen unterschiedliche Formen von Planung, Zustandsverwaltung und Zusammenarbeit mehrerer Agenten. Ebenso wichtig sind die Grenzen agentischer Autonomie: Berechtigungen, technische Guardrails, Beobachtbarkeit und kontrollierte Eskalation werden deshalb von Beginn an als Bestandteil der Architektur behandelt.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Grundkenntnisse generativer KI und praktische Programmiererfahrung, vorzugsweise mit Python

02 TRAININGSPROGRAMM

Inhalte

Vom Sprachmodell zum Agenten

- Sprachmodell und AI Agent
- Antworten und Handlungen
- Ziel und Aufgabe
- Kontext
- Zustand
- Werkzeuge
- Umgebung
- Feedback
- mehrstufige Bearbeitung

Was macht ein System agentisch?

- Wahrnehmen
- Entscheiden
- Handeln
- Ergebnis beobachten
- nächsten Schritt bestimmen
- Iteration
- Abschlussbedingungen
- unterschiedliche Grade von Autonomie

Agent oder Workflow?

- deterministische Workflows
- LLM-gestützte Workflows
- agentische Entscheidungen
- feste und dynamische Abläufe
- kontrollierbare Prozessketten
- flexible Agenten
- Kosten zusätzlicher Autonomie
- einfachste geeignete Architektur wählen

Grundarchitektur eines Agenten

- Benutzer oder aufrufendes System
- Modell
- System Instructions
- Kontext
- Zustand
- Tool Layer
- externe Systeme
- Kontroll- und Sicherheitsschicht
- Ergebnis und Rückmeldung

Agent Loop

- Aufgabe entgegennehmen
- aktuellen Zustand bewerten
- nächsten Schritt bestimmen
- Tool auswählen
- Aktion ausführen
- Ergebnis verarbeiten
- Plan anpassen
- Abschluss oder Fortsetzung

Aufgaben und Ziele formulieren

- Zielzustand
- Randbedingungen
- verfügbare Informationen
- erlaubte Aktionen
- Erfolgskriterien
- Nicht-Ziele
- Abbruchbedingungen
- Eskalationsbedingungen

Planung

- direkte Tool-Nutzung
- explizite Planung
- Plan-and-Execute
- schrittweise Planung
- Planänderungen
- Abhängigkeiten
- Zwischenziele
- wann Planung mehr Aufwand als Nutzen erzeugt

Tools

- Fähigkeiten eines Agenten erweitern
- Funktionen als Werkzeuge
- Tool-Beschreibungen
- Parameter
- strukturierte Ein- und Ausgaben
- Fehler
- Seiteneffekte
- Tool-Auswahl durch das Modell

Gute Agent Tools entwickeln

- kleine klar definierte Operationen
- eindeutige Semantik
- Validierung
- verständliche Fehlerzustände
- deterministische Komponenten
- idempotente Operationen
- minimale Berechtigungen
- gefährliche Universalwerkzeuge vermeiden

Function Calling und strukturierte Ausgaben

- Tool Calls
- strukturierte Parameter
- Schemas
- Validierung
- strukturierte Modellantworten
- fehlerhafte Tool Calls
- Retry
- Modell und Anwendung klar trennen

Model Context Protocol

- Zweck von MCP
- MCP Server und Client
- Tools
- Resources
- zusätzliche Kontextquellen
- vorhandene MCP-Server anbinden
- eigene Systeme verfügbar machen
- MCP als Integrationsschicht

Eigene MCP-Tools

- Tool-Grenzen bestimmen
- Tool-Beschreibungen
- Parameter und Rückgabewerte
- Fehlerbehandlung
- Berechtigungen
- Toolsets strukturieren
- agentenfreundliche Schnittstellen
- MCP-Server testen

Externe Daten und Systeme

- APIs
- Datenbanken
- Dateisysteme
- Suchsysteme
- Unternehmensanwendungen
- Web
- interne Services
- unterschiedliche Vertrauensbereiche

Zustand

- zustandslose Modellaufrufe
- Agentenzustand
- aktueller Auftrag
- bisherige Aktionen
- Zwischenergebnisse
- offene Arbeitsschritte
- Entscheidungen
- persistenter Zustand

Memory

- Kontext und Memory unterscheiden
- kurzfristiger Zustand
- längerfristiges Wissen
- Conversation Memory
- episodische Informationen
- Fakten und Präferenzen
- Speicherung
- Aktualisierung und Löschung
- falsche oder veraltete Erinnerungen

Externes Wissen

- Modellwissen und Anwendungswissen
- Dokumente
- Datenbanken
- Suchsysteme
- Retrieval
- relevante Informationen auswählen
- Quellen
- Aktualität

RAG im Agentenkontext

- Retrieval als Agentenfähigkeit
- Suche gezielt auslösen
- Query formulieren
- Ergebnisse bewerten
- weitere Suche entscheiden
- Quellen kombinieren
- Retrieval und Aktion verbinden
- RAG nicht mit Agentenarchitektur verwechseln

Kontextmanagement

- begrenztes Kontextfenster
- relevante Informationen auswählen
- Tool-Ergebnisse
- Gesprächsverlauf
- Dokumente
- Zwischenergebnisse
- Zusammenfassungen
- Kontext nicht unkontrolliert wachsen lassen

Ein einzelner Agent oder mehrere?

- Aufgabenkomplexität
- Spezialisierung
- Kontexttrennung
- unterschiedliche Toolsets
- parallele Arbeit
- zusätzliche Kommunikation
- Koordinationsaufwand
- Multi-Agent nicht als Selbstzweck

Subagents

- Hauptagent und spezialisierte Agenten
- Aufgaben delegieren
- begrenzter Kontext
- begrenztes Toolset
- Recherche-Agent
- Analyse-Agent
- Ausführungs-Agent
- Ergebnisse zurückgeben

Multi-Agent-Architekturen

- koordinierender Agent
- spezialisierte Rollen
- Übergabe von Aufgaben
- parallele Bearbeitung
- gemeinsame und getrennte Zustände
- Konflikte
- Ergebnisaggregation
- Fehlerfortpflanzung

Orchestrierung

- zentrale Orchestrierung
- dezentrale Zusammenarbeit
- Routing
- Delegation
- Übergaben
- Ereignisse
- Workflow Engine und Agent kombinieren
- deterministische Steuerung dort einsetzen, wo sie möglich ist

Human in the Loop

- Mensch als Informationsquelle
- Mensch als Entscheider
- Freigaben
- Rückfragen
- Eskalationen
- Unsicherheit
- kritische Aktionen
- unnötige Unterbrechungen vermeiden

Autonomiegrade

- Agent macht Vorschlag
- Agent plant
- Agent führt nach Freigabe aus
- ausgewählte Aktionen autonom
- weitgehend autonome Bearbeitung
- Autonomie abhängig vom Risiko
- Aufgaben- statt Systemfreigaben
- minimale notwendige Autonomie

Berechtigungen

- Identität des Agenten
- Lesen
- Schreiben
- externe Aktionen
- Netzwerkzugriff
- Credentials
- Secrets
- Rollen und Rechte
- Least Privilege

Trust Boundaries

- Modell
- Agent Runtime
- Tools
- externe Daten
- Benutzer
- andere Agenten
- externe Systeme
- vertrauenswürdige und nicht vertrauenswürdige Komponenten

Technische Guardrails

- Regeln außerhalb des Sprachmodells
- Eingabevalidierung
- Tool Policies
- Allow- und Deny-Listen
- Wrapper
- Gateways
- Hooks
- Limits
- Freigabemechanismen

Prompt Injection

- direkte Prompt Injection
- indirekte Prompt Injection
- externe Dokumente
- Webseiten
- E-Mails
- Tool-Ergebnisse
- Daten nicht mit Instruktionen verwechseln
- Instruktionen allein sind keine Sicherheitsgrenze

Agenten und untrusted Content

- externe Inhalte kennzeichnen
- Herkunft berücksichtigen
- Aktionen nicht unmittelbar aus Daten ableiten
- kritische Informationen verifizieren
- Tool-Aufrufe begrenzen
- Datenfluss kontrollieren
- Output Encoding und Validierung
- Defense in Depth

Fehlerbehandlung

- Tool nicht verfügbar
- ungültige Parameter
- Timeout
- externe Fehler
- widersprüchliche Informationen
- fehlgeschlagene Aktionen
- Retry
- Eskalation

Schleifen und Abbruchbedingungen

- Agent erreicht Ziel nicht
- wiederholte Aktionen
- wechselnde Lösungsversuche
- maximale Schritte
- Zeitlimits
- Kostenlimits
- Fortschritt erkennen
- kontrollierter Abbruch

Verifikation

- Agent behauptet Erfolg
- tatsächlichen Zustand prüfen
- deterministische Prüfungen
- Regeln und Invarianten
- externe Referenzen
- zweite Prüfung
- Ergebnis und Behauptung unterscheiden
- Verifikation als eigener Prozessschritt

Observability

- Agentenlauf nachvollziehen
- Modellaufrufe
- Tool Calls
- Parameter
- Ergebnisse
- Zustandsänderungen
- Fehler
- Laufzeiten und Kosten

Auditierbarkeit

- wer hat den Agenten gestartet?
- welche Aufgabe wurde gestellt?
- welche Informationen wurden verwendet?
- welche Aktionen wurden ausgeführt?
- welche externen Systeme wurden verändert?
- welche Freigaben erfolgten?
- welches Ergebnis entstand?
- relevante Informationen dauerhaft protokollieren

Qualität agentischer Systeme

- Erfolgsquote
- fachliche Richtigkeit
- Tool-Auswahl
- unnötige Aktionen
- Robustheit
- Laufzeit
- Kosten
- reproduzierbare Evaluationsszenarien

Agenten testen

- einzelne Tools testen
- Tool-Auswahl testen
- kontrollierte Umgebungen
- simulierte externe Systeme
- Erfolgs- und Fehlerpfade
- unerwartete Eingaben
- wiederholte Ausführung
- Regressionstests

Evaluation

- repräsentative Aufgaben
- erwartete Ergebnisse
- erlaubte Aktionen
- unerlaubte Aktionen
- Erfolgsmetriken
- Kostenmetriken
- menschliche Bewertung
- Änderungen an Modell oder Agent gegen Benchmark prüfen

Kosten und Performance

- Anzahl der Modellaufrufe
- Modellwahl
- Kontextgröße
- Tool-Aufrufe
- Retries
- parallele Agenten
- Latenz
- Kosten und Nutzen messen

Agenten in Anwendungen integrieren

- synchrone Aufrufe
- Hintergrundaufgaben
- APIs
- Queues
- Events
- Benutzeroberflächen
- Status und Fortschritt
- Abbruchmöglichkeiten

Produktionsarchitektur

- Agent Runtime
- Model Provider
- Tool Layer
- State Store
- Knowledge Layer
- Policy Layer
- Observability
- externe Systeme
- klare Verantwortlichkeiten zwischen den Komponenten

Typische Einsatzfelder

- Recherche und Informationsaufbereitung
- Dokumentenverarbeitung
- Support
- Datenanalyse
- Prozessunterstützung
- administrative Aufgaben
- Software Engineering
- domänenspezifische Assistenten

Wann kein Agent sinnvoll ist

- vollständig deterministische Aufgaben
- einfache API-Verkettungen
- bekannte Workflows
- hohe Anforderungen an deterministisches Verhalten
- unnötige Modellentscheidungen
- zu hohe Aktionsrisiken
- fehlende Verifikationsmöglichkeiten
- klassische Software als bessere Lösung

Einen Agenten entwerfen

- Anwendungsfall auswählen
- Ziel definieren
- Umgebung bestimmen
- benötigte Informationen
- benötigte Tools
- Zustand und Memory
- Autonomiegrad
- Sicherheits- und Qualitätsgrenzen

Vom Prototyp zum belastbaren System

- einfachen Agent Loop aufbauen
- Tools schrittweise ergänzen
- Fehlerfälle berücksichtigen
- Berechtigungen begrenzen
- Verifikation ergänzen
- Observability hinzufügen
- Evaluation aufbauen
- erst danach Autonomie erweitern

Praktische Übungen

- einfachen Agenten implementieren
- eigenes Tool anbinden
- strukturierte Tool-Aufrufe validieren
- externes Wissen integrieren
- Zustand über mehrere Schritte verwalten
- MCP-Server anbinden
- Agent mit mehreren Tools entwickeln
- Human-in-the-Loop-Freigabe einbauen
- Prompt-Injection-Szenario untersuchen
- Guardrail außerhalb des Modells implementieren
- Agentenlauf protokollieren und analysieren
- Agent mit einem definierten Aufgabensatz evaluieren
- vollständige Agentenarchitektur für einen Praxisfall entwerfen

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit Entwicklungsumgebung und Zugang zu einem aktuellen Sprachmodell

> **Inhouse-Schulungen**
> Interne APIs, Datenquellen, MCP-Server oder typische Geschäftsprozesse können als Grundlage für die praktischen Agenten verwendet werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Der Kurs richtet sich an Softwareentwickler, Softwarearchitekten, AI Engineers und technische Consultants, die agentische KI-Systeme verstehen, entwickeln oder in bestehende Anwendungen und Prozesse integrieren möchten.

Er eignet sich sowohl für Teilnehmer, die erste eigene Agenten entwickeln wollen, als auch für Teams, die bereits mit agentischen Frameworks oder Produkten experimentieren und die zugrunde liegenden Architektur- und Engineering-Prinzipien systematisch verstehen möchten.

> **Voraussetzungen**
> Praktische Programmiererfahrung wird vorausgesetzt; die Übungen können vorzugsweise mit Python durchgeführt werden. Grundkenntnisse über generative KI und Sprachmodelle werden erwartet. Spezielle Kenntnisse eines Agentenframeworks oder eines bestimmten Modellanbieters sind nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- AI Agents von Sprachmodellen, Assistenten und klassischen Workflows abgrenzen
- den Aufbau eines agentischen Systems erklären
- geeignete Anwendungsfälle für Agenten identifizieren
- Agent Loops und Planungsmechanismen entwickeln
- Tools für Agenten entwerfen und implementieren

- strukturierte Tool-Aufrufe validieren
- MCP zur Integration externer Fähigkeiten einsetzen
- Zustand, Kontext und Memory unterscheiden und gestalten
- externes Wissen und Retrieval integrieren
- Single- und Multi-Agent-Architekturen einordnen
- Human-in-the-Loop und unterschiedliche Autonomiegrade gestalten
- Berechtigungen und Trust Boundaries definieren
- technische Guardrails außerhalb des Modells implementieren
- Prompt-Injection- und Tool-Risiken berücksichtigen
- Fehler- und Abbruchstrategien entwickeln
- Agenten beobachtbar und auditierbar machen
- agentische Systeme systematisch testen und evaluieren
- Kosten und Performance berücksichtigen
- einen Agenten vom Prototyp zu einer belastbaren Architektur weiterentwickeln

06 FAQ

Häufige Fragen

Was unterscheidet diesen Kurs von Agentic Software Development?

Dieser Kurs behandelt AI Agents als allgemeine Systemarchitektur. Die Beispiele können aus unterschiedlichen Anwendungsgebieten stammen; behandelt werden Agent Loops, Tools, Zustand, Memory, Retrieval, MCP, Multi-Agent-Systeme, Sicherheit, Evaluation und Produktionsarchitektur.

Agentic Software Development konzentriert sich ausschließlich auf den Einsatz solcher Konzepte in der Softwareentwicklung: Coding Agents, Repositories, Entwicklungswerkzeuge, Git, Tests und Quality Gates stehen dort im Mittelpunkt.

Was unterscheidet einen Agenten von einem normalen LLM-Chat?

Ein Chat erzeugt zunächst Antworten auf Eingaben. Ein Agent kann darüber hinaus innerhalb eines mehrstufigen Ablaufs seinen nächsten Arbeitsschritt bestimmen, Werkzeuge verwenden, deren Ergebnisse auswerten und weitere Aktionen auslösen.

Muss ein Agent autonom arbeiten?

Nein. Autonomie ist kein Selbstzweck. Je nach Aufgabe kann ein Agent lediglich Vorschläge machen, einzelne Aktionen nach Freigabe durchführen oder innerhalb klarer Grenzen selbstständig arbeiten.

Brauche ich ein Agentenframework?

Nein. Gerade für das Verständnis der grundlegenden Mechanismen ist es hilfreich, zunächst einen einfachen Agent Loop und die Tool-Anbindung selbst zu entwickeln. Frameworks können anschließend dort eingesetzt werden, wo sie tatsächlich einen Vorteil bieten.

Wird MCP behandelt?

Ja. MCP wird als standardisierte Möglichkeit behandelt, Agenten mit Werkzeugen und zusätzlichen Informationsquellen zu verbinden. Dabei geht es sowohl um die Nutzung als auch um die Gestaltung geeigneter Tool-Schnittstellen.

Wird RAG behandelt?

Ja, aber aus Sicht eines Agentensystems: Retrieval ist eine mögliche Fähigkeit eines Agenten. Aufbau und Optimierung komplexer RAG-Systeme sind nicht Schwerpunkt dieses Kurses.

Werden Multi-Agent-Systeme behandelt?

Ja. Der Kurs behandelt Subagents, spezialisierte Rollen, Orchestrierung und parallele Bearbeitung. Ebenso wichtig ist die Frage, wann mehrere Agenten gegenüber einem einzelnen Agenten überhaupt einen Vorteil bieten.

Wird Sicherheit praktisch behandelt?

Ja. Berechtigungen, Trust Boundaries, Prompt Injection, Tool Policies und technische Guardrails sind Bestandteile der Übungen. Sicherheitsgrenzen sollen dabei nicht ausschließlich durch Instruktionen an das Sprachmodell entstehen.

Wird ein bestimmtes LLM vorausgesetzt?

Nein. Die vermittelten Konzepte sind grundsätzlich modell- und anbieterunabhängig. Für die praktischen Übungen wird ein geeignetes aktuelles Modell verwendet.

Was unterscheidet den Kurs von einer Framework-Schulung?

Der Kurs vermittelt Agent Engineering und keine Framework-API. Frameworks ändern sich vergleichsweise schnell; Agent Loops, Tool-Schnittstellen, Zustandsmanagement, Trust Boundaries, Verifikation und Evaluation bleiben dagegen grundlegende Architekturthemen.

Warum dauert der Kurs drei Tage?

Der Kurs umfasst sowohl die Entwicklung eines funktionsfähigen Agenten als auch die wesentlichen Architekturthemen für einen kontrollierten Einsatz: Tools, Zustand, MCP, Multi-Agent-Konzepte, Sicherheit, Verifikation und Evaluation. Drei Tage ermöglichen, diese Themen praktisch zu erarbeiten, ohne in die Spezialisierungen der weiterführenden Kurse auszuweichen.

Kann der Kurs an einen konkreten Anwendungsfall angepasst werden?

Ja. Bei Inhouse-Schulungen können beispielsweise interne APIs, Datenquellen, MCP-Server oder typische Geschäftsprozesse als Grundlage für die praktischen Agenten verwendet werden.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/ai-agents/