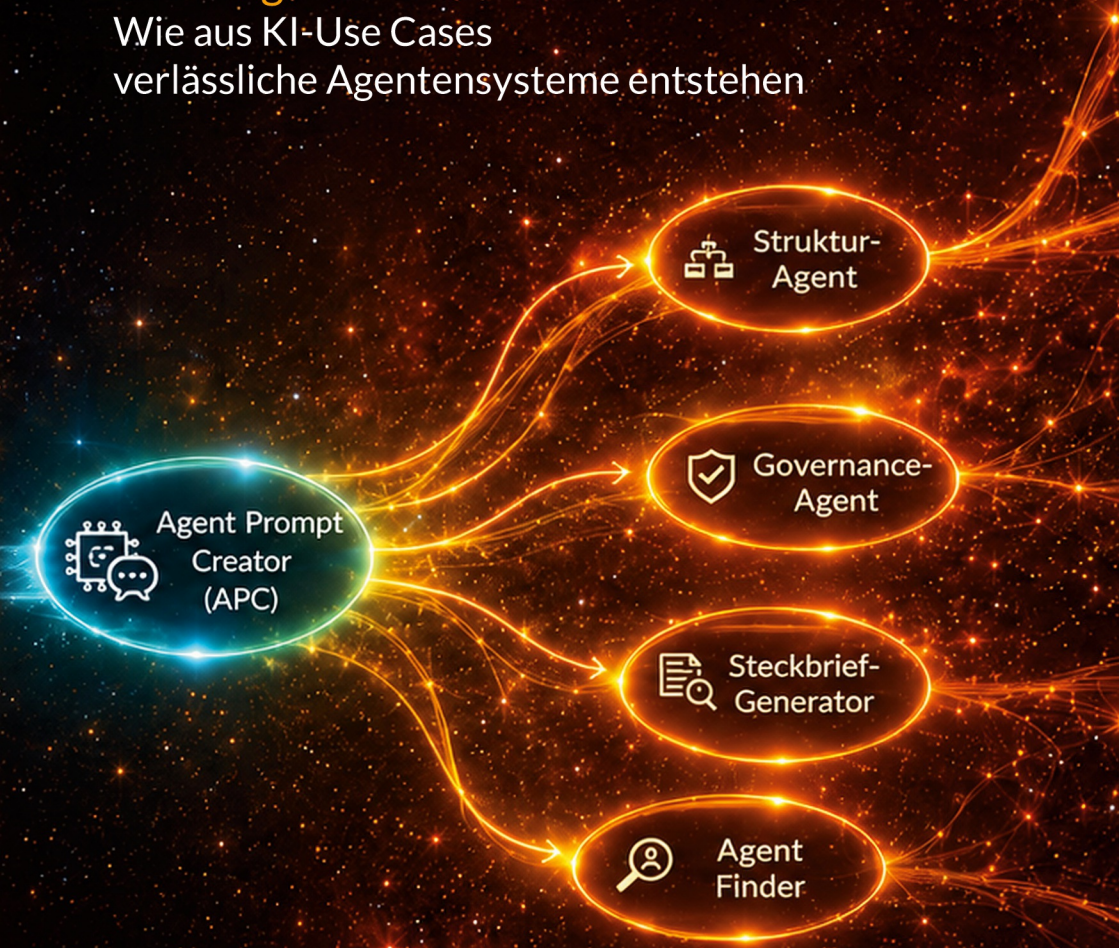
Agent Operating System

Vom Vagen zum Validen -
Wie aus KI-Use Cases
verlässliche Agentensysteme entstehen



Für alle, die mit KI-Agenten
verantwortungsvoll
und sicher umgehen wollen

Christian Timmerhoff

Christian Timmerhoff

AGENT OPERATING SYSTEM

Vom Vagen zum Validen – Wie aus KI-Use Cases verlässliche Agentensysteme entstehen

Für alle, die mit KI-Agenten
verantwortungsvoll
und sicher umgehen wollen.

Christian Timmerhoff

AGENT OPERATING SYSTEM

Vom Vagen zum Validen – Wie aus KI-Use Cases verlässliche Agentensysteme entstehen

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über dnb.dnb.de abrufbar.

Illustration und Medien: Alle Bilder, Podcasts und Videos wurden erstellt mit NotebookLM/ Google, Chat GPT 5.3/ OpenAI und MS Powerpoint/ Microsoft

Herstellung und Verlag:
BoD · Books on Demand GmbH, Überseering 33, 22297 Hamburg

ISBN: 978-3-6963-4628-7

"Wer KI nur fragt, bekommt Texte.
Wer sie strukturiert, bekommt Ergebnisse.“

Inhaltsangabe

Vorwort

Dieses Buch beginnt nicht mit einer Technik.
Es beginnt mit einer Irritation.
Viele, die heute mit KI arbeiten, kennen dieses Gefühl: Es funktioniert – aber es ist nicht mehr immer klar, warum.
Ergebnisse wirken plausibel. Systeme reagieren sinnvoll. Aufgaben werden schneller erledigt. Und doch entsteht im Hintergrund eine Unsicherheit:
Was genau steuert hier eigentlich das Verhalten?
Lange Zeit schien die Antwort einfach zu sein: bessere Prompts führen zu besseren Ergebnissen. Wer präziser formuliert, mehr Kontext gibt und klarere Anweisungen schreibt, bekommt bessere Antworten.
Das stimmt oft. Aber es reicht nicht.
Denn moderne KI-Systeme funktionieren nicht mechanisch. Sie reagieren kontextsensitiv, gewichten Informationen, ergänzen implizite Zusammenhänge und erzeugen Verhalten situativ.
Genau das macht sie leistungsfähig.
Und genau das macht sie schwer steuerbar.
In vielen Organisationen zeigt sich deshalb ein ähnliches Muster: Ergebnisse entstehen, aber sie bleiben nicht stabil. Entscheidungen wirken nachvollziehbar, sind aber nicht zuverlässig reproduzierbar. Systeme liefern, aber sie lassen sich nicht wirklich führen.
Genau hier setzt dieses Buch an.
Es verschiebt den Fokus: weg von der Formulierung einzelner Anfragen – hin zur Gestaltung von Entscheidungsstrukturen. Weg vom Prompt als vermeintlicher Steuerungseinheit – hin zum Agenten als strukturierter Entscheidungsinstanz. Und schließlich hin zu einem übergeordneten Rahmen, in dem Agenten sinnvoll entstehen, zusammenarbeiten, geprüft, gefunden und weiterentwickelt werden können: dem **Agent Operating System (AOS)**.

Dabei ist mir wichtig: Dieses Buch beschreibt kein fertiges technisches Produkt, das bereits alle Möglichkeiten moderner Agentic-AI-Systeme ausschöpft. Es beschreibt einen Weg, wie agentische Ordnungsprinzipien auch dort vorbereitet werden können, wo die technischen Voraussetzungen noch begrenzt sind.
In vielen Unternehmenskontexten lassen sich vor allem Assistenten umsetzen – also Custom-GPT-ähnliche Lösungen mit begrenzter Anzahl an Wissensartefakten, angebundenen Wissensordnern und klar definierten Systemprompts. Oft stehen noch nicht alle Möglichkeiten zur Verfügung, die langfristig denkbar oder wünschenswert wären: tief integrierte Aktionen, vernetzte Workflows, echte Subagents, dynamische Toolketten oder vollständig automatisierte Übergaben zwischen Agenten.

Trotzdem beginnt die eigentliche Veränderung nicht erst dort.
Denn auch ein Assistent ist nicht automatisch nur ein langer Prompt. Er kann als strukturierter Arbeitsraum gestaltet werden: mit Rollenlogik, Wissensordnung, Entscheidungsregeln, Grenzen, Prüfmechanismen und klar beschriebenen Übergaben. Das bedeutet nicht, dass ein solcher Assistent bereits ein vollautonomes agentisches System wäre. Aber es bedeutet, dass man ihn nicht mehr nur als Eingabemaske behandeln muss.
Das Agent Operating System wird hier deshalb bewusst unter minimalen Voraussetzungen gedacht. Nicht als große technische Plattform, die erst dann relevant wird, wenn alle Werkzeuge freigeschaltet sind, sondern als Systemlogik, die bereits dort beginnt, wo KI-Lösungen nicht mehr nur formuliert, sondern gestaltet, geprüft, dokumentiert und verantwortet werden.
Dabei geht es nicht darum, KI künstlich komplizierter zu machen. Es geht um das Gegenteil: um Klarheit.
Denn je stärker KI in Arbeitsprozesse eingebunden wird, desto weniger reicht es, einzelne gute Ergebnisse zu erzeugen. Entscheidend wird, ob Verhalten nachvollziehbar bleibt.

Ob Übergänge funktionieren. Ob Verantwortung geklärt ist. Ob Grenzen sichtbar sind. Und ob ein System auch dann stabil bleibt, wenn es genutzt, erweitert und verändert wird.
Deshalb ist dieses Buch auch ein Versuch.
Ein Versuch, unter realen Bedingungen eine AOS-Logik aufzubauen – nicht unter idealen Laborbedingungen. Mit den Mitteln, die heute verfügbar sind: Assistenten, Wissensartefakten, Wissensordnern, Systemprompts, manuellen Übergaben und klaren Prozessrollen. Und zugleich mit dem Blick auf das, was später möglich wird: Tools, Aktionen, Workflows, Subagents, Automatisierung und echte technische Orchestrierung.

Dieses Buch behauptet deshalb nicht, dass sich KI-Verhalten bereits vollständig kontrollieren lässt. Unter den aktuellen Voraussetzungen bleibt vieles promptbasiert, manuell und interpretativ. Regeln können formuliert, aber nicht in jedem Fall technisch erzwungen werden. Übergaben können beschrieben, aber nicht automatisch garantiert werden. Governance kann vorbereitet, aber nicht vollständig automatisiert werden.

Gerade darin liegt jedoch der praktische Ausgangspunkt. Wenn technische Möglichkeiten noch begrenzt sind, wird Struktur umso wichtiger. Das AOS ist in dieser frühen Form kein fertiges Betriebssystem, sondern eine methodische Disziplin: Es hilft, Assistenten nicht als lange Prompts zu behandeln, sondern als prüfbare, begrenzte und weiterentwickelbare Systembausteine.
Der entscheidende Gedanke lautet: Man muss nicht warten, bis alle technischen Möglichkeiten vorhanden sind, um systemisch zu denken. Man kann bereits heute beginnen, KI-Lösungen nicht als isolierte Promptlösungen zu behandeln, sondern als Bestandteile einer wachsenden Agentenlandschaft.
Dieses Buch ist aus praktischer Arbeit entstanden – aus vielen gebauten, getesteten, verworfenen und neu strukturierten Agenten.

Am Anfang war vieles noch Handarbeit: Prompts ergänzen, Rollen präzisieren, Regeln nachschärfen, Ausgaben stabilisieren. Mit der Zeit wurde daraus ein wiederkehrendes Muster.
Gleichzeitig fließt in dieses Buch eine zweite Perspektive ein:
meine berufliche Erfahrung aus der systemischen Prozessberatung. Dort beginnt gute Gestaltung selten mit der Frage, welches Werkzeug verfügbar ist. Prozesse werden nicht vom Instrument her geplant, sondern vom gewünschten Ergebnis, vom Nutzungskontext und vom Kundenbedarf aus gedacht.
Gestaltung beginnt mit deshalb mit anderen Fragen:
Was soll am Ende für den Kunden besser funktionieren?
Welcher Prozess soll stabiler, klarer oder entlastender werden?
Welche Wirkung soll entstehen?
Welche Schnittstellen müssen stimmen?
Und was muss von hinten her gedacht werden, damit vorne nicht nur Aktivität, sondern tatsächlich Nutzen entsteht?

Diese Perspektive prägt auch meinen Blick auf KI-Agenten. Ein Agent wird nicht dadurch sinnvoll, dass er technisch möglich ist. Er wird sinnvoll, wenn er in einen Prozess passt, einen Bedarf sauber adressiert, Übergänge verbessert, Verantwortung klärt und für die Nutzerinnen und Nutzer einen nachvollziehbaren Mehrwert erzeugt.
Deshalb denke ich Agenten nicht (mehr) zuerst vom Prompt her. Und auch nicht zuerst vom Modell. Ich denke sie vom gewünschten Ergebnis, vom Kunden, vom Prozess und vom späteren Einsatz her. Erst dann stellt sich die Frage, welche Rolle KI darin übernehmen soll. Aus diesem Zusammenspiel entstand die zentrale Frage dieses Buches:
Wie baut man Agenten nicht nur so, dass sie funktionieren – sondern so, dass sie Teil eines stabilen Systems werden?
Die Erfahrungen aus über 400 erst gebastelten, später dann konstruierten und getesteten Agenten fließen in dieses Buch ein. Nicht

als Sammlung fertiger Rezepte, sondern als Grundlage für ein strukturiertes Denken: vom Prompt zum Agenten, vom Agenten zum System und vom System zur Verantwortung. Ich liefere keine schnellen Tricks oder eine Abkürzung, die alle Probleme löst.
Ich behaupte auch nicht, dass ein Assistent bereits dasselbe ist wie ein vollautonomes agentisches System. Das wäre ungenau. Aber ich behaupte: Wenn ein Assistent mit klarer Rolle, expliziter Entscheidungslogik, geordnetem Wissen, definierten Grenzen, Governance-Anbindung und prüfbaren Ausgaben gebaut wird, dann lässt sich eine Vorform agentischer Ordnung abbilden.
Nicht, weil der Assistent technisch autonom wäre. Sondern weil sein Verhalten nicht mehr allein aus einer einzelnen Eingabe entstehen soll. Es soll an einer Struktur ausgerichtet werden.
Und genau diese Struktur ist der Anfang des AOS.

Ich möchte ein Verständnis dafür schaffen, warum KI-Systeme funktionieren – oder eben nicht. Warum gute Prompts wichtig sind, aber nicht ausreichen. Warum Agenten Struktur brauchen. Warum Orchestrierung nicht erst mit automatisierten Workflows beginnt. Und warum robuste Systeme nicht zufällig entstehen.
Denn gute Ergebnisse kann man erzeugen.
Stabile Systeme muss man gestalten.

Mein Dank gilt an dieser Stelle Deborah Kohl, mit der ich den ersten lauffähigen MVP des Use Case Beraters entwickeln durfte, an Kristina Klemm, die uns die Idee dazu lieferte.
Und nun:
Lasst uns nicht nur basteln.
Lasst uns bauen.

Christian Timmerhoff
Juni 2026

Einleitung

Die meisten KI-Initiativen beginnen mit einer einfachen Hoffnung: Wenn wir die richtigen Prompts schreiben, werden die Ergebnisse besser.
Und am Anfang stimmt das oft. Ein klarer Prompt liefert bessere Antworten als ein unklarer. Mehr Kontext hilft. Struktur verbessert die Ausgabe. Beispiele machen Ergebnisse greifbarer. Doch irgendwann stößt dieser Ansatz an eine Grenze.
Nicht, weil Prompts unwichtig wären. Sondern weil sie nicht mehr ausreichen, sobald KI nicht nur einzelne Antworten erzeugt, sondern Aufgaben strukturiert bearbeitet, Entscheidungen vorbereitet oder in Prozesse eingebunden wird.
Dann verändert sich die eigentliche Frage.
Nicht mehr:
„Wie formuliere ich die perfekte Anfrage?“
Sondern:
„Wie entsteht verlässlicheres Verhalten in einem KI-System?“

Genau um diese Frage geht es in diesem Buch.
Dabei wird KI nicht zuerst als Tool betrachtet, das möglichst viele Aufgaben übernehmen soll. Der Ausgangspunkt ist ein anderer: Welche Wirkung soll in einem Prozess entstehen? Welche Aufgabe muss wirklich gelöst werden? Welche Entscheidung soll vorbereitet, welche Arbeit entlastet, welche Schnittstelle stabilisiert werden?
Erst von dort aus wird sinnvoll, über Prompts, Assistenten, Agenten, Wissensartefakte, Governance oder Orchestrierung zu sprechen. Denn KI wird nicht dadurch besser, dass man sie früher in den Prozess bringt. Sie wird dann nützlich, wenn klar ist, welche Rolle sie im Prozess übernehmen soll.

Vom Prompt zum System

In vielen Organisationen zeigt sich ein ähnliches Muster.
Erste KI-Anwendungen funktionieren. Prompts liefern brauchbare Ergebnisse. Teams bauen eigene Lösungen. Erste Assistenten oder Agenten entstehen. Es wird getestet, verbessert und ausgerollt.
Das wirkt wie Fortschritt. Und das ist es auch.

Aber mit der Zeit entstehen neue Probleme:
Ähnliche Lösungen werden mehrfach gebaut. Agenten verhalten sich bei vergleichbaren Aufgaben unterschiedlich. Ergebnisse wirken plausibel, sind aber schwer nachvollziehbar. Unklarheiten werden still ergänzt. Übergaben zwischen Schritten funktionieren nicht sauber. Und irgendwann ist nicht mehr eindeutig, warum ein System tut, was es tut.

Das Problem liegt dann nicht unbedingt in der Technologie.
Es liegt oft darin, dass niemand das System wirklich führt.
Alle arbeiten mit KI. Viele verbessern einzelne Lösungen. Aber kaum jemand gestaltet den Zusammenhang.
Oder anders gesagt:
Alle arbeiten im System. Aber niemand arbeitet am System.

Genau an diesem Punkt beginnt die Verschiebung, die dieses Buch beschreibt: weg vom einzelnen Prompt, weg von der isolierten Lösung, hin zu einem Systemverständnis. Nicht jede Aufgabe braucht einen Agenten. Nicht jeder Agent braucht maximale Komplexität. Und nicht jede KI-Lösung wird besser, nur weil sie technisch aufwendiger gebaut wird.
Entscheidend ist, ob die Lösung zum Bedarf, zum Prozess, zum Risiko und zum gewünschten Ergebnis passt.

Warum gute Ergebnisse nicht reichen

Ein gutes Ergebnis ist wichtig. Aber es ist noch kein stabiles System. Ein einzelner Prompt kann funktionieren. Ein einzelner Agent kann überzeugen. Ein erster Test kann beeindruckend sein.
Doch Stabilität zeigt sich nicht im besten Einzelfall. Sie zeigt sich daran, ob ein System unter vergleichbaren Bedingungen nachvollziehbar, konsistent und verantwortbar arbeitet.
Genau hier beginnt der Unterschied zwischen Prompting und Agent Design.

Prompting optimiert Eingaben.
Agent Design gestaltet Verhalten.

Ein Agent Operating System ordnet das Zusammenspiel.
Dieses Zusammenspiel muss nicht von Anfang an technisch vollständig automatisiert sein. In frühen Umsetzungen kann es auch durch klare Prozessrollen, manuelle Übergaben, standardisierte Steckbriefe, Governance-Prüfungen und bewusst gestaltete Wissensräume entstehen.
Entscheidend ist nicht, ob bereits jedes technische Element verfügbar ist. Entscheidend ist, ob die Systemlogik erkennbar wird.
Diese Systemlogik ist am Anfang nicht gleichbedeutend mit technischer Durchsetzung. Sie ist zunächst eine Soll-Ordnung: ein beschriebener Rahmen, an dem Verhalten ausgerichtet und später geprüft werden kann.
Das ist ein nüchterner, aber wichtiger Punkt. Ein Systemprompt, ein Wissensartefakt oder eine definierte Entscheidungslogik erzwingen noch kein fehlerfreies Verhalten. Sie machen aber sichtbar, woran Verhalten gemessen werden kann. Ohne einen solchen Rahmen bleibt oft nur die Frage, ob ein Ergebnis gut oder schlecht war. Mit einem Rahmen lässt sich zusätzlich prüfen, ob der Agent seine Rolle, seine Grenzen, seine Prioritäten und seine Stopp-Regeln eingehalten hat.

Dieses Buch zeigt diesen Weg Schritt für Schritt.
Es beginnt mit der Grenze des klassischen Prompt-Denkens. Dann wird sichtbar, was ein Agent eigentlich ist: keine magische autonome Einheit, sondern eine strukturierte Entscheidungsinstanz innerhalb eines Systems.
Von dort aus geht es weiter zur Frage, wie solche Agenten gebaut werden: mit klaren Rollen, Regeln, Entscheidungslogiken, Informationsgrundlagen, Übergaben, Orchestrierung und Governance.
Und schließlich führt der Weg zum Agent Operating System: einem Rahmen, in dem Agenten nicht nur entstehen, sondern gefunden, geprüft, genutzt, verändert und weiterentwickelt werden können.

Die zentrale These

Die zentrale These dieses Buches lautet:
Tragfähige KI-Systeme entstehen nicht durch bessere Formulierungen allein, sondern durch klare Entscheidungsstrukturen.
Das bedeutet nicht, dass Prompts unwichtig sind.
Im Gegenteil: Gute Prompts bleiben wichtig. Aber sie sind nicht die eigentliche Steuerungsebene, wenn ein System verlässlich arbeiten soll. Entscheidend ist, was hinter dem Prompt liegt:

- Welche Rolle übernimmt der Agent?
- Welche Regeln gelten?
- Welche Informationen dürfen verwendet werden?
- Welche Prioritäten greifen bei Konflikten?
- Wann muss nachgefragt werden?
- Wann darf entschieden werden?
- Wann muss gestoppt oder eskaliert werden?
- Wie werden Ergebnisse übergeben?
- Wer entscheidet, ob ein Agent genutzt werden darf?
- Und wie bleibt das System anschlussfähig, wenn es wächst?

Diese Fragen sind der Kern des Buches.
Sie zeigen zugleich, warum der Begriff „Agent Operating System“ hier nicht als fertige Softwarelösung verstanden wird. Gemeint ist ein Ordnungsrahmen für die Entstehung, Nutzung, Prüfung und Weiterentwicklung von Agenten. Ein AOS beschreibt nicht nur, dass Agenten existieren. Es beschreibt, wie sie in einem Zusammenhang sinnvoll funktionieren sollen.

Für wen dieses Buch geschrieben ist

Dieses Buch richtet sich an Menschen, die mit KI arbeiten und merken, dass reine Prompt-Optimierung nicht mehr reicht.
Es richtet sich an Entscheiderinnen und Entscheider, die verstehen wollen, warum einzelne KI-Lösungen nicht automatisch skalieren.
Es richtet sich an Prompt Designerinnen und Prompt Designer, die nicht nur bessere Agenten bauen, sondern deren Wirkung verantworten wollen bzw. müssen.
Es richtet sich an Entwicklerinnen, Entwickler und Prompt Engineers, die erkennen, dass technische Umsetzung ohne Systemlogik instabil bleibt.
Und es richtet sich an Fach- und Geschäftsbereiche, die KI nicht nur ausprobieren, sondern sinnvoll, nachvollziehbar und dauerhaft nutzbar machen wollen.
Man braucht dafür nicht jede technische Debatte im Detail zu kennen. Entscheidend ist ein anderes Verständnis: KI ist nicht nur ein Werkzeug, das auf Eingaben reagiert. KI wird in Organisationen zunehmend Teil von Arbeitsabläufen, Entscheidungen und Systemen. Und genau deshalb muss sie auch so gestaltet werden.

Wie dieses Buch aufgebaut ist

Das Buch folgt einem klaren Weg.
Teil 1 beschreibt den Paradigmenwechsel: weg vom reinen Prompt Engineering, hin zum agent-first-Denken.

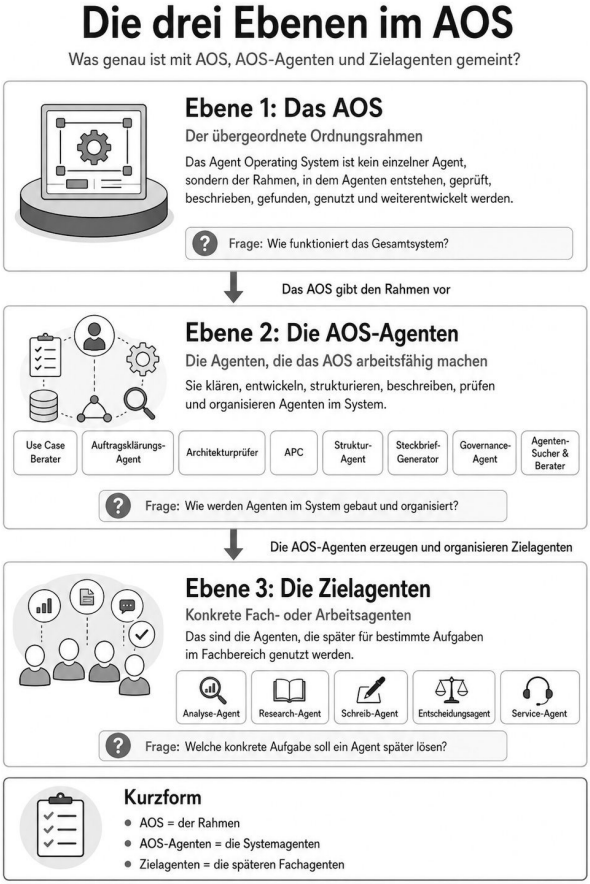
Du lernst, warum Prompts Verhalten beeinflussen, aber nicht verlässlich steuern – und warum Agenten als strukturierte Entscheidungsinstanzen verstanden werden müssen.

Teil 2 führt vom Agenten zum System. Hier geht es um Entscheidungslogik, Informationsgrundlagen, Systembausteine, Übergaben, Orchestrierung und Governance. Dieser Teil legt die theoretische und methodische Grundlage.

Teil 3 zeigt die AOS-Agenten in der Praxis. Gemeint sind hier vor allem die Agenten, mit denen das AOS selbst arbeitsfähig wird: der Use Case Berater, der Auftragsklärungs-Agent, der Architekturprüfer, der Agent Prompt Creator, der Struktur-Agent, der Steckbrief-Generator, der Governance-Agent und der Agenten-Sucher & Berater. Sie sind nicht einfach Beispiele für beliebige KI-Agenten, sondern bilden die operative Kette, mit der aus vagen Bedarfen belastbare, prüfbare und auffindbare Agenten entstehen.
Wichtig ist deshalb eine Unterscheidung, die sich durch das gesamte Buch zieht: Das AOS ist der übergeordnete Rahmen. Die AOS-Agenten sind die Agenten, die diesen Rahmen praktisch herstellen und betreiben. Zielagenten sind die konkreten Fach- oder Arbeitsagenten, die mit Hilfe dieses Systems entstehen (siehe auch die Grafik auf Seite 18). Wo Verwechslungsgefahr besteht, wird die jeweilige Ebene ausdrücklich benannt.

Teil 4 erweitert diese Perspektive auf das Agent Operating System selbst. Denn gute AOS-Agenten und gute Zielagenten reichen nicht aus, wenn Nutzung, Modellwechsel, Skalierung und Veränderung das System mit der Zeit verschieben. Hier geht es darum, wie eine Agentenlandschaft dauerhaft geführt werden kann – zunächst mit den verfügbaren Mitteln, später auch mit erweiterten Möglichkeiten wie Tools, Aktionen, Workflows, Subagents und stärker automatisierter Orchestrierung.

Damit die folgenden Kapitel leichter einzuordnen sind, hilft eine übersichtliche Darstellung:

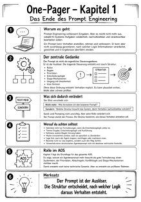


Diese drei Ebenen laufen im Buch immer wieder zusammen. Entscheidend ist jedoch, sie gedanklich getrennt zu halten: Das AOS organisiert das System, die AOS-Agenten betreiben die Entwicklungs- und Prüfprozesse, und die Zielagenten lösen später konkrete Aufgaben.

Aufbau der Kapitel im Buch:
Jedes Kapitel folgt einem wiederkehrenden Aufbau. Das hilft dir, dich schnell zu orientieren und die Inhalte so zu nutzen, wie es für dich gerade sinnvoll ist. Du kannst ein Kapitel vollständig von Anfang bis Ende lesen. Du kannst aber auch gezielt in einzelne Elemente springen, wenn du etwas wiederholen, anwenden oder vertiefen möchtest.

Haupttext des Kapitels
Am Anfang steht der Haupttext des Kapitels. Hier wird das jeweilige Thema Schritt für Schritt aufgebaut und verständlich erklärt. Du erfährst, worum es geht, warum das Thema wichtig ist und welche Rolle es im Gesamtzusammenhang des Agent Operating Systems spielt. Dieser Teil ist der rote Faden des Kapitels. Wenn du ein Thema wirklich verstehen willst, solltest du hier beginnen.

Mini-Szenarien
Danach folgen Mini-Szenarien aus der Praxis. Sie zeigen in kurzen, konkreten Situationen, wo typische Probleme entstehen, welche Missverständnisse auftreten können und worauf es im Einsatz wirklich ankommt. Die Szenarien sind keine theoretischen Beispiele, sondern kleine Prüfstellen: Sie machen sichtbar, ob ein Prinzip im Alltag tatsächlich trägt.



One Pager
Der One Pager fasst die wichtigsten Aussagen des Kapitels kompakt zusammen. Er ist dafür gedacht, zentrale Punkte schnell zu wiederholen, später noch einmal nachzuschlagen oder das Kapitel in wenigen Minuten wieder präsent zu haben. Wenn du das Kapitel bereits

gelesen hast, reicht oft ein Blick auf den One Pager, um die Kernaussagen wieder einzuordnen.

Praxistipps

Die Praxistipps übertragen die Kapitelinhalte in die praktische Anwendung. Sie wirken manchmal wie konkrete Anweisungen für Prompt Designer, gemeint sind sie aber als Orientierungshilfen: Worauf solltest du achten? Welche Fehler solltest du vermeiden? Welche Fragen helfen dir, einen Agenten klarer, stabiler oder verantwortbarer zu gestalten? Dieser Abschnitt ist besonders nützlich, wenn du selbst Agenten erstellst, prüfst oder weiterentwickelst.

QR-Codes / Zusatzmaterial

Alle Kapitel enthalten zusätzliche QR-Codes. Dahinter findest du ergänzende Inhalte:

Podcast	Diskussion zum Kapitel (13-25 min. lang)
Video	Kompaktere Zusammenfassung mit visueller Unterstützung (7-12 min. lang)
FAQ	Fragen und Antworten, die Du Dir vielleicht auch beim Lesen gestellt hast (ca. 20)
Schaubild/ Grafik	Umfassende Darstellung der AOS-Agenten

Diese Materialien sind nicht zwingend notwendig, um das Kapitel zu verstehen. Sie ersetzen den Text aber auch nicht. Sie bieten dir weitere Zugänge, wenn du ein Thema lieber hörst, audio-visuell nachvollziehen oder gezielt vertiefen möchtest.

Öffne den QR-Code einfach mit dem Scanner deines Handys (Tipp: wenn die QR-Codes sehr nah beieinander liegen, lege die Linse der Handykamera auf den Code und führe die Kamera langsam nach oben).

Du kannst das Buch also auf unterschiedliche Weise nutzen:

- als durchgehende Einführung,
- als Nachschlagewerk,
- als Arbeitsbuch für die Praxisbegleitung beim Aufbau eigener Agentenstrukturen.

Der wiederkehrende Kapitelaufbau sorgt dafür, dass du nicht jedes Mal neu suchen musst, sondern schnell erkennst, wo Erklärung, Beispiel, Zusammenfassung, Anwendung und Vertiefung zu finden sind.

Der rote Faden

Dieses Buch liefert keine Sammlung schneller Tricks.
Es geht nicht darum, noch einen besseren Prompt zu schreiben.
Es geht nicht darum, möglichst viele Agenten zu bauen.
Und es geht nicht darum, KI möglichst schnell überall einzusetzen.
Es geht darum, KI so zu gestalten, dass sie nachvollziehbar, verantwortbar und verlässlicher arbeitet.
Der rote Faden ist deshalb einfach: Erst verstehen, warum Prompting an Grenzen stößt. Dann erkennen, was Agenten wirklich leisten. Dann lernen, wie Agenten systematisch entstehen. Dann begreifen, warum daraus ein Betriebssystem für Agenten werden muss.
Die entscheidende Frage ist am Ende nicht:
„Sind deine Agenten gut?“
Sondern:
„Bleibt dein System gut, wenn es genutzt wird?“

TEIL 1: DER PARADIGMENWECHSEL

Kapitel 1 – Das Ende des Prompt Engineering

Du schreibst einen Prompt, der richtig gut funktioniert.
Klare Struktur, gutes Ergebnis, alles passt.
Also nutzt du ihn wieder.
Und wieder.
Und irgendwann merkst du:
Er funktioniert… aber nicht mehr gleich.
Mal besser, mal schlechter.
Mal klar, mal irgendwie daneben.

Also machst du das, was naheliegt:
Du optimierst ihn.
Präziser. Länger. Durchdachter.

Und trotzdem bleibt ein Rest Unsicherheit.

Warum fühlt sich Prompting so logisch an –
und verhält sich trotzdem nicht verlässlich?

Podcast: 18:34

Video: 10:07

Das Ende des Prompt Engineering

Prompt Engineering – oder präziser: Prompt Design – wirkt auf den ersten Blick wie eine Disziplin, die sich systematisch verbessern lässt.

Präzisere Formulierungen, klarere Rollen und detailliertere Anweisungen führen oft tatsächlich zu besseren Ergebnissen. Dieses Bild ist intuitiv – und genau deshalb so wirkmächtig.

Es basiert jedoch auf einer Annahme, die in der Praxis nur begrenzt trägt: Dass sich Verhalten direkt und zuverlässig über einzelne Eingaben steuern lässt.
Genau an diesem Punkt beginnt das eigentliche Problem.
Prompting beeinflusst Verhalten. Aber es definiert Verhalten nicht belastbar.

Ein Prompt kann eine Richtung vorgeben. Er kann Kontext liefern, Erwartungen beschreiben und Ergebnisse verbessern. Aber er legt nicht zuverlässig fest, nach welchen Regeln ein System Informationen verarbeitet, gewichtet und in Entscheidungen überführt.

Solange es nur um einfache Einzelaufgaben geht, fällt das kaum auf. Eine Übersetzung, eine kurze Zusammenfassung, eine einfache Formulierung – hier reicht ein guter Prompt oft aus.

Doch sobald Aufgaben mehrdeutig werden, mehrere Informationen zusammenkommen oder Ergebnisse nachvollziehbar und reproduzierbar sein müssen, zeigt sich die Grenze.
Dann reicht es nicht mehr, eine bessere Anweisung zu schreiben.
Dann muss klar sein, wodurch Verhalten überhaupt gesteuert wird.

Der grundlegende Denkfehler: Prompt = Steuerung

Der zentrale Denkfehler besteht darin, den Prompt für die eigentliche Steuerung zu halten.

Moderne KI-Systeme folgen einzelnen Instruktionen nicht linear und eindeutig. Ihr Verhalten entsteht aus dem Zusammenspiel von Eingabe, Kontext, Systemvorgaben und interner Verarbeitung. Genau darin liegt der Widerspruch:

Man versucht, ein System über ein Mittel zu steuern, das Verhalten zwar beeinflusst, aber nicht vollständig und nicht robust festlegt.

Ein Prompt ist deshalb keine hinreichende Steuerungseinheit für stabiles Systemverhalten. Er kann Verhalten anstoßen. Er kann es rahmen. Er kann es in eine Richtung lenken. Aber er bestimmt nicht verlässlich, nach welchen Regeln dieses Verhalten entsteht.

Wenn diese Regeln nicht explizit definiert sind, entsteht Verhalten implizit. Und implizit entstehendes Verhalten ist nur begrenzt steuerbar. Es wirkt oft plausibel, bleibt aber schwer überprüfbar, schwer reproduzierbar und schwer begründbar.

Daraus folgt das **erste Prinzip:**

Verhalten darf nicht nur implizit im Sprachmodell entstehen. Es muss auf expliziten Regeln beruhen.

Genau hier zeigt sich die strukturelle Grenze von Prompt Engineering. Ein Prompt enthält für sich genommen keine verbindliche Struktur, die Verhalten stabil erzeugt. Jede Eingabe wirkt im

Zusammenspiel mit Kontext und Systemzustand. Was dabei wie Flexibilität aussieht, ist häufig fehlende Systembindung.

Die eigentliche Frage lautet deshalb nicht mehr:
„Wie muss ich den Prompt formulieren?“
Sondern: „Wodurch entsteht im System verbindliches Verhalten?“

Formulierung ist nicht die Stabilisierungsebene

Formulierung ist wichtig. Aber sie ist nicht die primäre Stabilisierungsebene. Solange Verhalten hauptsächlich über sprachliche Feinsteuerung beeinflusst wird, bleibt es abhängig von Kontext, impliziten Annahmen und nicht vollständig kontrollierbaren Variationen.

Stabilität entsteht erst dort, wo Regeln, Prioritäten, Grenzen und der Umgang mit Unsicherheit explizit definiert sind.
Das ist der erste Schritt weg vom reinen Prompt-Denken.
Nicht die schönere Formulierung macht ein System stabiler, sondern die klarere Struktur dahinter.
Fehlt diese Grundlage, können Prompts zwar plausible Ergebnisse erzeugen. Sie garantieren aber nicht, dass unter vergleichbaren Bedingungen auch vergleichbare Entscheidungen entstehen.

Kleine Veränderungen im Input oder Kontext können dann zu deutlich anderen Ergebnissen führen. Nicht, weil das System „falsch“ arbeitet, sondern weil keine stabile Entscheidungslogik greift.

Daraus ergibt sich das **zweite Prinzip:**

Unter vergleichbaren Bedingungen müssen Systeme zu hinreichend konsistenten und erklärbaren Ergebnissen kommen.

Ohne dieses Prinzip gibt es keine belastbare Steuerung.
Dann bleibt nur Beobachtung: Man schaut, was herauskommt, bewertet das Ergebnis und optimiert nachträglich. Aber das ist kein robustes Systemdesign. Das ist Nachjustieren am Symptom.

Die Vermischung von Ebenen

Ein weiteres Problem des Prompting liegt in der Vermischung von Ebenen. In einem einzelnen Prompt werden häufig Ziel, Logik und Ausführung gleichzeitig formuliert.

- Man beschreibt, was erreicht werden soll.
- Man gibt Regeln vor.
- Man legt das Format fest.
- Man ergänzt Beispiele.
- Man versucht, Ausnahmen zu berücksichtigen.
- Und manchmal schreibt man auch gleich noch hinein, wie das System denken soll.

Das kann kurzfristig funktionieren. Aber es macht Weiterentwicklung schwer. Denn wenn sich das Ergebnis verändert, bleibt unklar, woran es liegt.

- Lag es an einer anderen Zielsetzung?
- An einer veränderten Entscheidungslogik?
- An einer anderen Formulierung?
- An fehlendem Kontext?
- Oder an der Art der Ausführung?

Solange alles in einem Prompt vermischt ist, wirken Änderungen ungerichtet auf das Gesamtsystem zurück.

Ein belastbarer Systemansatz trennt diese Ebenen sauber:

- Eingabe
- Entscheidungslogik
- Ausführung

Die Eingabe stößt den Prozess an. Die Entscheidungslogik legt fest, wie mit der Eingabe umzugehen ist. Die Ausführung erzeugt das Ergebnis. Erst diese Trennung macht Verhalten besser strukturierbar und gezielter prüfbar. Sie sorgt dafür, dass Änderungen nicht mehr ungerichtet auf das gesamte System zurückwirken, sondern nachvollziehbarer vorgenommen, überprüft und bewertet werden können.
Daraus folgt das **dritte Prinzip:**

Keine Entscheidung ohne definierte Entscheidungsgrundlage.

Sobald ein System Entscheidungen trifft, ohne dass klar ist, nach welchen Regeln dies geschieht, verliert es seine Nachvollziehbarkeit. Ergebnisse dürfen nicht nur plausibel wirken. Sie müssen auf einer Grundlage beruhen, die sich benennen, prüfen und begründen lässt.

Warum getrennte Module allein nicht reichen

Die Trennung von Eingabe, Entscheidungslogik und Ausführung ist ein wichtiger Schritt. Sie verhindert, dass Ziel, Kontext, Regeln, Format und Stil in einem einzigen Anweisungstext ineinanderlaufen. Für sich genommen reicht diese Trennung jedoch noch nicht aus.

Denn auch ein sauber gegliederter Agentenprompt bleibt zunächst ein Text, der vom Modell als Gesamtkontext verarbeitet wird. Das Modell liest nicht nur den Abschnitt, der gerade relevant erscheint.

Es verarbeitet den gesamten Rahmen und muss daraus ableiten, welche Anweisung in der konkreten Situation Vorrang hat.
Damit entsteht ein neues Risiko.

Wenn die einzelnen Module lediglich nebeneinanderstehen, aber nicht zueinander geordnet sind, entscheidet das Modell weiterhin situativ, welche Hinweise es stärker berücksichtigt. Dann ist zwar die Oberfläche strukturierter, die Steuerung aber noch nicht verbindlich. Aus einem unübersichtlichen Prompt wird dann nur ein besser gegliederter Prompt – aber noch kein belastbares System.

Deshalb braucht ein Agentenprompt mehr als Module. Er braucht eine interne Verarbeitungsordnung.

Diese Verarbeitungsordnung legt fest, welche Regeln immer gelten, welche Regeln Vorrang vor anderen haben, in welcher Reihenfolge gearbeitet wird, welche Module unter welchen Bedingungen aktiviert werden und was bei Konflikten geschieht. Sie definiert außerdem, wann der Agent nicht weitermachen darf, weil eine Entscheidungsgrundlage fehlt oder Unsicherheit nicht aufgelöst werden kann.

Erst dadurch entsteht der Unterschied zwischen einer bloßen Sammlung von Anweisungen und einem stabileren Agentendesign. Ein langer Prompt sagt dem Modell, was es alles beachten soll. Ein Agentendesign legt zusätzlich fest, wie diese Anforderungen zueinander stehen.

Dazu braucht der Agentenprompt einen Systemkern. Dieser Systemkern wirkt wie eine innere Verfassung des Agenten. Er definiert Prioritäten, Ablaufregeln, Aktivierungsbedingungen, Konfliktregeln und Stopp-Mechanismen.

Ohne diesen Systemkern bleiben Eingabe-, Entscheidungs- und Ausführungslogik nur einzelne Abschnitte. Mit ihm werden sie zu einem geordneten Verarbeitungssystem, dessen Regeln zumindest sichtbar, anwendbar und überprüfbar werden.
In der Praxis bedeutet das: Sicherheits- und Governance-Regeln müssen Vorrang vor Stilwünschen haben. Die Entscheidungslogik muss Vorrang vor dem Ausgabeformat haben. Der definierte Arbeitsprozess darf nicht übersprungen werden, nur weil der Nutzer eine schnelle Antwort verlangt. Unsicherheit darf nicht geglättet werden, nur damit das Ergebnis vollständiger wirkt. Und wenn eine Entscheidung nicht belastbar getroffen werden kann, muss der Agent dies markieren, nachfragen oder den Prozess stoppen.

Damit wird ein Agentenprompt nicht dadurch verlässlicher, dass er länger oder detaillierter ist. Er wird robuster, weil er eine Ordnung enthält, nach der seine eigenen Bestandteile verarbeitet werden.

Unsicherheit ist kein Formulierungsproblem

Ein besonders kritischer Punkt ist der Umgang mit Unsicherheit.
Prompting behandelt Unsicherheit häufig nicht als Zustand, sondern als Lücke, die das System möglichst unauffällig schließt.

Fehlende Informationen werden ergänzt. Widersprüche werden geglättet. Unklare Begriffe werden interpretiert. Offene Punkte werden stillschweigend entschieden. Das Ergebnis wirkt dann vollständig und sauber.

Genau darin liegt das Risiko. Denn sprachliche Glätte kann verdecken, dass die Grundlage unsicher ist.
Unsicherheit ist aber kein Fehler. Sie ist ein Zustand, der sichtbar behandelt werden muss.

Wenn entscheidungsrelevante Informationen fehlen, widersprüchlich sind oder nicht ausreichend abgesichert werden können, darf keine implizite Ergänzung an ihre Stelle treten.

Daraus folgt das **vierte Prinzip:**

Unsicherheit muss sichtbar bleiben.
Das kann bedeuten:
- Der Prozess stoppt.
- Eine Rückfrage wird gestellt.
- Eine Annahme wird klar markiert.
- Ein Konflikt wird benannt.
- Ein Risiko wird eskaliert.

Entscheidend ist nicht, dass das System immer sofort eine Antwort liefert. Entscheidend ist, dass es erkennt, wann eine Antwort nicht belastbar wäre.
Genau dieser Punkt unterscheidet plausible Ergebnisse von verlässlicheren Ergebnissen.

Die Grenze des Agentenprompts

Trotzdem bleibt eine wichtige Einschränkung bestehen: Auch ein gut strukturierter Agentenprompt erzeugt keine absolute Verhaltensgarantie.

Er kann Prioritäten setzen, Abläufe definieren, Konflikte regeln und Stopp-Mechanismen beschreiben. Aber er hebt die grundsätzliche Funktionsweise eines Sprachmodells nicht auf.

Das Modell verarbeitet weiterhin Kontext, gewichtet Hinweise und wendet Regeln in einer konkreten Situation an. Dabei kann es

Regeln übergewichten, untergewichten, falsch anwenden oder im Einzelfall übergehen.
Deshalb darf ein Agentenprompt nicht mit technischer Deterministik verwechselt werden.
Seine Stärke liegt nicht darin, Fehlverhalten unmöglich zu machen. Seine Stärke liegt darin, Verhalten strukturierter, sichtbarer und prüfbarer zu machen. Ohne definierte Regeln bleibt unklar, wovon ein Ergebnis überhaupt abweicht. Mit Systemkern, Entscheidungslogik und Stopp-Regeln lässt sich dagegen prüfen, ob der Agent seinen eigenen Rahmen eingehalten hat.

Ein unstrukturierter Prompt kann meist nur nach **Ergebnisqualität** bewertet werden: War die Antwort gut oder schlecht?

Ein strukturierter Agentenprompt kann zusätzlich nach **Prozessqualität** bewertet werden:
Hat der Agent die richtige Reihenfolge eingehalten?
Hat er Unsicherheit markiert?
Hat er eine Entscheidung auf definierter Grundlage getroffen?
Hat er seine Grenzen beachtet?

Damit verschiebt sich die Kontrolle. Nicht nur das Ergebnis wird bewertet, sondern auch der Weg dorthin.
Das macht Agentendesign nicht perfekt. Aber es macht Abweichungen erkennbarer. Und genau darin liegt der praktische Fortschritt gegenüber reinem Prompt Engineering.

Warum Prompt Engineering strukturell an Grenzen stößt

Spätestens hier wird deutlich, warum Prompt Engineering als primäres Steuerungsmodell an Grenzen stößt.

Es operiert auf der Ebene einzelner Eingaben und versucht, Verhalten über Formulierungen zu beeinflussen.

Ein systematischer Ansatz arbeitet dagegen auf der Ebene von Regeln, Entscheidungswegen und Prioritäten. Er definiert nicht nur, was eingegeben wird, sondern unter welchen Bedingungen Verhalten entsteht.
Damit verändert sich auch die Rolle des Prompts.
Der Prompt ist nicht mehr der Ort der eigentlichen Steuerung. Er ist der Einstiegspunkt in ein bereits strukturierteres System. Er liefert Input. Er startet den Prozess. Er gibt Richtung.

Aber er bestimmt nicht allein, wie dieser Input bewertet, verarbeitet und in ein Ergebnis überführt wird. Die Steuerung liegt in der Struktur dahinter.

Genau daraus entsteht die Notwendigkeit einer anderen Gestaltungsebene: einer Struktur, in der Regeln, Prioritäten, Entscheidungswege und Grenzen explizit gebündelt sind.

Wie stark dieser Unterschied ist, zeigt sich in typischen Situationen aus der Praxis.

Sichtbare Systemeffekte

1. Gleicher Prompt, unterschiedliche Ergebnisse
Zwei Nutzer verwenden denselben Prompt mit identischer Zielbeschreibung.
Die Ergebnisse unterscheiden sich dennoch deutlich: Einmal entsteht eine klare, strukturierte Antwort, ein anderes Mal eine oberflächliche Zusammenfassung mit anderem Schwerpunkt.
Die Ursache liegt nicht nur in der Formulierung des Prompts. Sie liegt vor allem darin, dass keine explizite Logik definiert ist, die festlegt, wie Informationen gewichtet, priorisiert und verarbeitet werden.
Der Prompt ist gleich. Der Verarbeitungsweg ist es nicht.

2. Unsicherheit wird nicht sichtbar gemacht
Ein Prompt beschreibt eine Aufgabe nur teilweise. Wichtige Informationen fehlen oder bleiben unklar.
Statt diese Lücken offen zu kennzeichnen, ergänzt das System sie implizit. Das Ergebnis wirkt vollständig und schlüssig, basiert aber auf Annahmen, die weder geprüft noch transparent gemacht wurden.

Ohne klare Regeln zum Umgang mit Unsicherheit entsteht ein scheinbar valides Ergebnis, das in Wirklichkeit auf unsicherer Grundlage steht.

3. Konflikte werden implizit aufgelöst
In einer Analyse werden mehrere Quellen mit widersprüchlichen Aussagen verarbeitet. Statt den Konflikt offenzulegen, bevorzugt das System implizit eine der beiden Varianten. Welche Quelle sich durchsetzt und warum, bleibt intransparent.

Das Ergebnis wirkt konsistent. Aber seine Grundlage ist nicht nachvollziehbar.
Ohne definierte Prioritäten oder Entscheidungsregeln entsteht eine Antwort, die plausibel klingt, aber nicht belastbar begründet ist.

Verdichtung

Diese drei Situationen folgen demselben Muster:
Verhalten entsteht implizit – und bleibt deshalb nur begrenzt steuerbar. Der Paradigmenwechsel lässt sich deshalb präzise zusammenfassen: Prompt Engineering optimiert Eingaben.
Systemdesign strukturiert Verhalten über Regeln, Logik und Prioritäten. Genau darin liegt die eigentliche Verschiebung.
Es geht nicht mehr nur darum, bessere Antworten zu formulieren. Es geht darum, Systeme zu gestalten, die Entscheidungen nachvollziehbar, reproduzierbar und verantwortbar vorbereiten können.
Die Stabilisierung entsteht jedoch nicht aus der Formulierung allein, sondern aus der Logik, die hinter der Eingabe arbeitet.

KAPITEL 1

Das Ende des Prompt Engineering

WAHRER EINFLUSS ENTSTEHT NICHT DURCH BESSERE PROMPTS – SONDERN DURCH BESSERE SYSTEME.

Wir haben lange geglaubt, bessere Prompts führen zu besseren Ergebnissen.

Die Praxis zeigt: Das reicht nicht.
Prompts sind ein Auslöser, keine Steuerung.

Stabilität entsteht nicht durch Formulierung – sondern durch Struktur.

Der nächste Schritt heißt: Agent Design.

KERNGEDANKE

Prompts beeinflussen das Verhalten, sie steuern es nicht.

Stabilität entsteht durch Struktur, nicht durch bessere Formulierung.

Wer Verhalten verlässlich steuern will, muss Systeme gestalten – nicht Prompts perfektionieren.

WARUM PROMPT ENGINEERING AN STRUKTURELLE GRENZEN STÖSST

PROMPT = AUSLÖSER
Der Prompt startet einen Prozess. Er legt aber nicht die Logik fest, nach der entscheiden wird.

EBENEN VERMISCHT
Ziel, Logik und Ausführung werden in einem Prompt vermischt. Das verhindert verlässliches Verhalten.

KONTEXT VERÄNDERT ALLES
Schon kleine Änderungen im Input führen zu anderen Ergebnissen.

UNSICHERHEIT WIRD VERDECKT
Fehlende oder widersprüchliche Informationen werden oft still aufgefüllt.

ZUFALL BLEIBT
Selbst bei identischem Prompt entstehen unterschiedliche Ergebnisse.

SYSTEMEFFEKTE ENTSTEHEN
Ineffizienzen, Fehlerfolgen und fehlende Nachvollziehbarkeit sind die Folge.

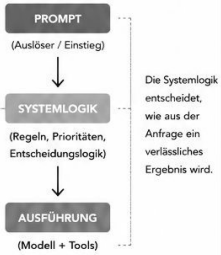
DER ZENTRALE DENKFEHLER

Annahme: Bessere Prompts → bessere Ergebnisse.
Realität: Bessere Strukturen → bessere und stabilere Ergebnisse.

WAS STATTDESSEN ENTSCHEIDEND IST

- **Klare Rolle und Ziel**
  Der Agent weiß, wofür er gedacht ist.
- **Explizite Entscheidungslogik**
  Regeln, Prioritäten und Kriterien sind festgelegt.
- **Strukturierte Informationsgrundlagen**
  Relevante Informationen sind priorisiert und begrenzt.
- **Umgang mit Unsicherheit**
  Rückfragen, Kennzeichnung oder Stopp statt stiller Annahmen.
- **Trennung der Ebenen**
  Eingabe, Entscheidungslogik und Ausführung sind sauber getrennt.
- **Stabile Übergaben**
  Zustände werden strukturiert übergeben, nicht nur als Text.

VOM PROMPT ZUR SYSTEMSTEUERUNG



RESTAURANT-ANALOGIE

1. BESTELLUNG (PROMPT)
Du sagst, was du möchtest. Das ist der Auslöser.

2. KÜCHENLOGIK (SYSTEM)
Rezepte, Standards und Abläufe entscheiden, was wirklich passiert.

3. GERICHT (ERGEBNIS)
Das Ergebnis entsteht durch das System – nicht durch die Formulierung der Bestellung.

VERDICHTUNG

Prompts werden auch in Zukunft wichtig bleiben – als Einstiegspunkt.
Aber Stabilität, Qualität und Verantwortung entstehen nicht durch bessere Prompts, sondern durch besseres Agent Design.

ÜBERGANG ZU KAPITEL 2

Nun klären wir, was ein Agent wirklich ist und wie sich sein innerer Aufbau von herkömmlichen Assistenten unterscheidet.

Verhalten lässt sich nicht allein durch bessere Formulierungen tragfähiger machen. Wenn Ergebnisse schwanken, liegt das Problem oft nicht primär im Prompt, sondern in der fehlenden oder zu schwachen Struktur dahinter.

Für deine Praxis heißt das:

- Optimiere nicht nur den Prompt, wenn das eigentliche Problem in der Entscheidungslogik liegt.
- Definiere klare Regeln, Prioritäten und Entscheidungswege.
- Trenne Eingabe, Entscheidungslogik und Ausführung.
- Ordne diese Ebenen durch einen Systemkern, damit sie nicht nur nebeneinanderstehen.
- Mache Unsicherheit sichtbar, statt sie sprachlich zu überdecken.
- Lege offen, wie Konflikte behandelt und Entscheidungen begründet werden.
- Verwechsle einen strukturierten Agentenprompt nicht mit absoluter technischer Kontrolle.
- Der wichtigste Wechsel ist dabei nicht technisch, sondern gedanklich: Du steuerst nicht primär über Sprache. Du steuerst über Struktur.

Kurz gesagt:
Der Prompt ist der Auslöser.
Die Struktur bestimmt, nach welcher Logik daraus Verhalten entsteht.