Medien & Digitale Technologien
University of Applied Sciences St. Pölten

Graph-basierte Systemarchitektur vs. monolithische Systemprompt-Struktur

Eine empirische Evaluation von fachlicher Korrektheit, Halluzinationsverhalten, Entscheidungsstabilität und Nachvollziehbarkeit in LLM-gestützten Validierungs- und Korrektursystemen im Produktionsumfeld

Bachelorarbeit

Ausgeführt zum Zweck der Erlangung des akademischen Grades
Bachelor of Science in Engineering

am Bachelorstudiengang

Smart Engineering of Production Technologies and Processes
an der

Hochschule für angewandte Wissenschaften St. Pölten

von:
Ahmad Alsayad
52318601

Betreuer/in: Dipl.-Ing. **Michael Macher**, BSc, MSc

St.Pölten, 15.09.2026

I

EUDRES²

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass

- ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
- ich mich bei der Erstellung der Arbeit an die Standards guter wissenschaftlicher Praxis gemäß Satzungsteil 05 der Hochschule für Angewandten Wissenschaften St. Pölten gehalten habe.
- ich die vorliegende Arbeit an keiner Hochschule zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt oder veröffentlicht habe.

Über den Einsatz von Hilfsmitteln der generativen Künstlichen Intelligenz wie Chatbots, Bildgeneratoren, Programmieranwendungen, Paraphrasier- oder Übersetzungstools erkläre ich, dass

☐ im Zuge dieser Arbeit kein Hilfsmittel der generativen Künstlichen Intelligenz zum Einsatz gekommen ist.

☒ ich Hilfsmittel der generativen Künstlichen Intelligenz verwendet habe, um die Arbeit Korrektur zu lesen.

☒ ich Hilfsmittel der generativen Künstlichen Intelligenz verwendet habe, um Teile des Inhalts der Arbeit zu erstellen. Ich versichere, dass ich jeden generierten Inhalt mit der Originalquelle zitiert habe. Das genutzte Hilfsmittel der generativen Künstlichen Intelligenz ist an entsprechenden Stellen ausgewiesen.

Durch den Satzungsteil 05 - Gute Wissenschaftliche Praxis der Hochschule für Angewandte Wissenschaften bin ich mir über die Konsequenzen einer wahrheitswidrigen Erklärung bewusst.

Kurzfassung

Der Einsatz von Large Language Models ermöglicht die automatisierte Verarbeitung strukturierter Produktionsdaten, wirft jedoch die Frage auf, unter welchen Bedingungen solchen Systemen operative Entscheidungen anvertraut werden können. Die vorliegende Arbeit untersucht, ob die Architektur eines LLM-gestützten Korrektursystems messbare Unterschiede in der Qualität und Nachvollziehbarkeit seiner Ausgaben bewirkt.

Untersuchungsgegenstand ist ein Vier-Agenten-System auf Basis von Azure OpenAI GPT-4.1, das fehlerhafte JSON-Snapshot-Dateien analysiert, korrigiert und automatisiert zurückschreibt. Die Snapshots stammen aus dem laufenden Betrieb eines ERP-Systems und werden über das Produktionsplanungssystem Smart Planning bereitgestellt. Ein einzelner Snapshot umfasst bis über 200.000 Einträge. Verglichen werden eine monolithische Systemprompt-Struktur und eine graphbasierte Systemarchitektur. Siebzehn Testfälle mit 29 durch kontrollierte Fehlerinjektion definierten erwarteten Änderungen wurden in jeder Bedingung fünfmal ausgeführt, insgesamt 255 Messläufe. Ergänzend wurde eine strukturierte Expertenerhebung mit zwei Fachpersonen durchgeführt.

Die Ergebnisse zeigen in drei der vier untersuchten Dimensionen geringe Unterschiede. Bei der fachlichen Korrektheit tritt die größere Differenz beim Übergang zur modularen Regelbereitstellung auf und gleicht sich bei Betrachtung des tatsächlich hergestellten Endzustands weitgehend an. Beim Halluzinationsverhalten und bei der Entscheidungsstabilität liegen die drei Bedingungen eng beieinander. Über alle 255 Läufe entstand ein einziger neuer Fehler im Zielsystem. Ein deutlicher Unterschied zeigt sich bei der Nachvollziehbarkeit. Allein die graphbasierte Variante erzeugt zusammenhängende Ablauf- und Herkunftsinformationen, die eine Rekonstruktion einzelner Korrekturentscheidungen ermöglichen.

Die Ergebnisse unterstreichen zudem die Notwendigkeit, fachliche Korrektheit und Halluzinationsverhalten getrennt zu erfassen. In jeder Bedingung wichen 15 Änderungsinstanzen vom Referenzwert ab und waren zugleich durch die vorgelegten Daten gedeckt. Die Expertenerhebung lieferte keine vergleichenden Urteile, da beide Teilnehmenden eine belastbare Beurteilung erst nach einer mehrwöchigen Erprobung im laufenden Betrieb für möglich hielten. Diese Einschätzung ist selbst ein Ergebnis und bestimmt die Anlage einer weiterführenden Untersuchung.

Abstract

The use of large language models enables the automated processing of structured production data, but it raises the question of the conditions under which such systems can be entrusted with operational decisions. This thesis examines whether the architecture of an LLM-based correction system produces measurable differences in the quality and traceability of its outputs.

The object of study is a four-agent system based on Azure OpenAI GPT-4.1 that analyses faulty JSON snapshot files, corrects them, and writes them back automatically. The snapshots originate from the live operation of an ERP system and are provided through the production planning system Smart Planning. A single snapshot comprises up to more than 200,000 entries. A monolithic system prompt structure is compared with a graph-based system architecture. The study comprises two architectures and three measurement conditions. Seventeen test cases with 29 expected changes defined through controlled fault injection were executed five times in each condition, resulting in 255 measurement runs. A structured expert survey with two domain practitioners was conducted in addition.

The results show small differences in three of the four dimensions examined. For factual correctness, the larger difference occurs at the transition to modular rule provision and largely levels out when the state actually established in the target system is considered. Hallucination behaviour and decision stability are closely comparable across the three conditions. Across all 255 runs, a single new error was introduced into the target system. A clear difference emerges for traceability. Only the graph-based variant produces coherent execution and provenance information that allows individual correction decisions to be reconstructed.

The results further underline the need to assess factual correctness and hallucination behaviour separately. In each condition, 15 change instances deviated from the reference value while at the same time being supported by the data provided. The expert survey yielded no comparative judgements, as both participants considered a reliable assessment possible only after several weeks of use in live operation. This assessment is itself a finding and shapes the design of any follow-up study.

Inhaltsverzeichnis

Kurzfassung III

Abstract IV

Inhaltsverzeichnis V

1 Einleitung 9
1.1 Einleitung und Problemstellung 9
1.2 Praxiskontext und bestehendes System 10
1.3 Problemstellung 11
1.4 Zielsetzung der Arbeit 14
1.5 Forschungsfrage und Thesen 15
1.5.1 Von drei Problemdimensionen zu vier Messdimensionen 15
1.5.2 Die vier Dimensionen 16
1.5.3 Hauptforschungsfrage 17
1.5.4 Unterfragen (zur Ergänzung/Vertiefung) 17
1.6 Aufbau der Arbeit 18
1.7 Abgrenzung der Arbeit 18

2 Theoretische Grundlagen 21
2.1 Large Language Models und strukturierte Ausgaben 21
2.2 Prompting und Kontextbereitstellung 21
2.3 Monolithische Bereitstellung 22
2.4 Modularisierung und Zerlegung 23
2.5 Graphbasierte Orchestrierung 24
2.6 Halluzinationen und Belegbarkeit 26
2.7 Stabilität und Sensitivität von Modellausgaben 27
2.8 Nachvollziehbarkeit, Faithfulness und Provenienz 27
2.9 Evaluation von LLM-Systemen im industriellen Kontext 28
2.10 Forschungslücke und Zusammenfassung 30

3 Untersuchungsgegenstand: das bestehende System 31
3.1 Projektkontext 31
3.2 Das Zielsystem Smart Planning 32
3.3 Das Vier-Agenten-System 35
3.4 Die Korrekturpipeline 37
3.5 Die Regelbereitstellung 40
3.6 Eigenschaften der Ausgangsarchitektur 42
3.7 Zusammenfassung 43

4 Entwicklung und Implementierung der Vergleichsbedingungen 44

4.1 Entwicklungsverlauf und Ableitung der Vergleichsbedingungen 44
4.2 Die drei Vergleichsbedingungen 45
4.3 Gemeinsame Bestandteile 46
4.4 Regelbereitstellung 46
4.5 Entwurfsentscheidungen der graphbasierten Variante 48
4.6 Der GraphState 49
4.7 Die neun Knoten 51
4.8 Kanten und Iterationslogik 53
4.9 Anwendung, Upload und erneute Validierung 54
4.10 Provenienz 54
4.11 Was nicht verändert wurde 56
4.12 Nachgelagerte Konfigurierbarkeit 56
4.13 Zusammenfassung 58

5 Forschungsdesign und Methodik 59
5.1 Untersuchungsanlage 59
5.2 Vergleichsbedingungen und Vergleichslogik 60
5.3 Kontrollvariablen 61
5.4 Herstellung der Vergleichbarkeit 63
5.5 Testdesign 63
5.6 Ablauf eines Messlaufs 64
5.7 Reproduzierbarkeit 64
5.8 Präzisierung der Forschungsfragen 65
5.9 Nachträglich festgelegte Auswertungsregeln 66
5.10 Expertenerhebung 66
5.11 Validität und Grenzen des Designs 68
5.12 Zusammenfassung 69

6 Operationalisierung und Messinstrumente 70
6.1 Aufbau der Bewertung 70
6.2 Testfallkatalog und Referenzwerte 70
6.2.1 Konstruktion der Fälle 70
6.2.2 Umfang der Referenz 70
6.3 Fachliche Korrektheit 71
6.3.1 Dreistufige Bewertung 71
6.3.2 Bewertungsumfang der Referenz 73
6.3.3 Kennzahl auf Laufebene 74
6.4 Funktionaler Endzustand 74
6.5 Halluzinationsverhalten 74
6.5.1 Vier Kategorien 74
6.5.2 Erhebbarkeit der Kategorien 75

6.6 Folgefehler 76
6.7 Entscheidungsstabilität 77
6.8 Technische Nachvollziehbarkeit 78
6.8.1 Messvorschrift 78
6.8.2 Herkunft der Angaben 79
6.8.3 Grenzen 80
6.9 Expertenerhebung 80
6.9.1 Begründung des Verfahrens 80
6.9.2 Konstruktion des Erhebungsinstruments 81
6.9.3 Ableitung der Fragen 82
6.9.4 Abweichungen vom Rahmenwerk 83
6.9.5 Durchführung und Auswertung 83
6.9.6 Reichweite der Aussagen 84
6.10 Zusammenfassung 84

7 Ergebnisse 85
7.1 Datengrundlage 85
7.2 Fachliche Korrektheit 86
7.3 Funktionaler Endzustand 88
7.4 Halluzinationsverhalten 89
7.5 Folgefehler und Entscheidungsstabilität 89
7.6 Technische Nachvollziehbarkeit 90
7.7 Ergebnisse der Expertenerhebung 91
7.8 Nicht erhobene Kennzahlen 91

8 Diskussion 92
8.1 Fachliche Korrektheit und die Wahl der Messgröße 92
8.2 Die beiden Interventionen 93
8.3 Halluzinationsverhalten 94
8.4 Entscheidungsstabilität 94
8.5 Verhalten im Zielsystem 95
8.6 Nachvollziehbarkeit 95
8.7 Zusammenführung der beiden Evidenzquellen 97
8.8 Bedrohungen der Validität 97
8.9 Beantwortung der Forschungsfrage und der Unterfragen 98

9 Fazit und Ausblick 101
9.1 Zusammenfassung der Untersuchung 101
9.2 Zentrale Befunde 101
9.3 Methodischer Beitrag 102
9.4 Empfehlungen für die Praxis 103
9.5 Reichweite der Ergebnisse 103

9.6 Ausblick 103

Literaturverzeichnis 105

Abbildungsverzeichnis 109

Tabellenverzeichnis 110

Quellcodeverzeichnis 111

Anhang 112

Anhang A: Vollständiger Auszug aus einem Snapshot 112
Anhang B: Vollständiger Auszug aus dem Regelwerk 115
Anhang C: Gegenüberstellung von Regelwerk und Regelkarte 118
Anhang D: Definition des GraphState 119
Anhang E: Auswertungsprotokoll 121
Anhang F: Testfallkatalog und Referenzwerte 127
Anhang G: Erhebungsbogen der Expertenerhebung 129
Anhang H: Transkript und schriftliche Zusammenfassung 135
Anhang I: Nachweis zum Einsatz generativer Künstlicher Intelligenz 138

1 Einleitung

1.1 Einleitung und Problemstellung

Die industrielle Fertigung befindet sich im Zuge der digitalen Transformation in einem tiefgreifenden Wandel. Unter dem Begriff Industry 4.0 werden Produktionsprozesse zunehmend durch vernetzte, datengetriebene Systeme gesteuert, in denen KI-Technologien nicht mehr nur eine analytische Rolle einnehmen, sondern aktiv in operative Entscheidungsprozesse eingebunden sind. Die automatisierte Verarbeitung strukturierter Produktionsdaten, die Unterstützung bei der Fehlererkennung in komplexen Planungssystemen sowie die intelligente Assistenz bei wiederkehrenden, regelbasierten Aufgaben gewinnen dabei zunehmend an Bedeutung – nicht nur als Forschungsgegenstand, sondern als gelebte Praxis in industriellen Projekten. Wie weit dieser Einsatz tatsächlich gediehen ist, relativiert allerdings eine Übersichtsarbeit von Henkel et al. (2026). Der überwiegende Teil der dort ausgewerteten Arbeiten bewegt sich auf dem Reifegrad von Labormustern und Prototypen.

Im Rahmen eines Praxisprojekts bei der CANCOM Austria AG konnte ich unmittelbar erleben, wie komplex und fehleranfällig die Kombination von KI und Produktionsplanung in der Realität sein kann. Das Produktionsplanungssystem des Kunden verwaltet Fertigungsaufträge, Ressourcenbelegungen und Zeitplanungen, die in Form strukturierter JSON-Snapshot-Dateien abgebildet werden. Diese Snapshots spiegeln den aktuellen Planungszustand wider und bilden die Grundlage für operative Entscheidungen auf dem Shopfloor. In der Praxis enthalten sie jedoch regelmäßig Fehler, die durch wiederholtes manuelles Pflegen an den verschiedensten Stellen entlang der Produktionskette entstehen – angefangen bei der Erfassung von Lagerzuständen, über die Eintragung von Mitarbeiterqualifikationen, bis hin zur Auslieferung von Arbeitsaufträgen an Endkunden. Die manuelle Identifikation und Korrektur dieser Fehler war zeitaufwändig, repetitiv und trotz des Fachpersonals fehleranfällig – ein klarer Ansatzpunkt für KI-gestützte Automatisierung.

Large Language Models können strukturierte Daten analysieren, Muster erkennen und Korrekturen begründen, und erste praktische Erfahrungen zeigen eine erhebliche Reduktion manueller Bearbeitungszeiten. Gleichzeitig bringt ihr Einsatz in produktionskritischen Umgebungen ein Risiko mit sich, das ich im Projektverlauf selbst beobachten konnte.

LLMs produzieren unter bestimmten Bedingungen faktisch inkorrekte, aber syntaktisch plausible Ausgaben – in der Forschungsliteratur als Halluzinationen

bezeichnet. Ji et al. (2023) dokumentieren in ihrer systematischen Übersicht, dass dieses Phänomen kein Randproblem ist, sondern ein strukturelles Merkmal aktueller LLM-Architekturen, das insbesondere bei steigender Aufgabenkomplexität und zunehmender Kontextlänge an Bedeutung gewinnt. Huang et al. (2025) bestätigen diesen Befund in einer aktuelleren Übersicht und ordnen ihn nach Ursachen entlang der Trainings- und Inferenzkette. Ergänzend zeigen Wen et al. (2024), dass die Kombination von Retrieval-Mechanismen mit graph-basiertem Reasoning einen vielversprechenden Ansatz darstellt, um sowohl Halluzinationen zu reduzieren als auch die Transparenz von Modellentscheidungen zu verbessern.

Im industriellen Produktionskontext ist dieses Risiko besonders schwerwiegend. Eine syntaktisch valide, aber inhaltlich falsche JSON-Korrektur kann unbemerkt in das ERP-System übernommen werden und dort fehlerhafte Fertigungsaufträge, inkorrekte Ressourcenzuweisungen oder fehlerhafte Terminierungen auslösen – mit direkten operativen Konsequenzen. Dass strukturierte Ausgaben zwar formal gültig, aber im Zielsystem nicht verwendbar sind, führen Henkel et al. (2026) als eigene Problemkategorie industrieller Agentensysteme an.

Hinzu kommen zwei weitere Probleme, die ich in der praktischen Systemarbeit beobachtet habe. Zum einen die mangelnde Nachvollziehbarkeit von Korrekturentscheidungen, die eine effektive Qualitätssicherung erschwert. Zum anderen die mangelnde Robustheit des Systems gegenüber variierenden oder strukturell abweichenden Eingaben, die in heterogenen Produktionsumgebungen alltäglich sind. Diese Unklarheit bei bestimmten Entscheidungen führt dazu, dass die Einhaltung eines Mindestmaßes an Qualität schwer garantierbar erscheint.

Diese drei Problemdimensionen – Halluzinationsrate, Nachvollziehbarkeit und Robustheit – bilden den Ausgangspunkt der vorliegenden Arbeit. Sie beschreiben zunächst Beobachtungen aus dem Projektbetrieb. Ihre Überführung in messbare Größen erforderte im Verlauf der Untersuchung eine Präzisierung, die in Abschnitt 1.5 beschrieben wird.

1.2 Praxiskontext und bestehendes System

Das empirische Fundament dieser Arbeit ist ein Entwicklungsprojekt bei der CANCOM Austria AG in der Abteilung Digital Makers. Konzeption, Architekturentscheidungen und Implementierung des Systems entstanden im Rahmen meines dualen Studiums und waren Gegenstand des vorangegangenen Praxisprojekts.

Für die vorliegende Arbeit dient das System als Untersuchungsgegenstand, nicht als eigenständige Entwicklungsleistung.

Der unmittelbare Zugang zu Systemgrenzen, Entwurfsentscheidungen und beobachtbarem Verhalten ist dabei zugleich eine methodische Einschränkung, die in Kapitel 5 behandelt wird.

Das System schließt eine Lücke in einem Produktionsplanungssystem. Dieses stellt seinen aktuellen Planungszustand als strukturierte JSON-Snapshot-Datei bereit und erkennt Fehler darin über eine eigene Validierungs-Engine, kann sie aber nicht korrigieren. Die Korrektur erfolgte bis zur Einführung des hier untersuchten Systems vollständig manuell durch spezialisiertes Fachpersonal und beanspruchte je Snapshot regelmäßig mehrere Stunden. Das entwickelte Korrektursystem bezieht fehlerhafte Snapshots, analysiert und korrigiert sie mithilfe eines LLM-gestützten Agentensystems und schreibt sie automatisiert zurück.

Fehler in diesem Prozessschritt wirken unmittelbar operativ. Der korrigierte Snapshot ist die Eingangsgröße des eigentlichen Planungslaufs, der daraus Fertigungsaufträge, Ressourcenzuweisungen und Zeitfenster auf dem Shopfloor ableitet. Eine fachlich falsche Korrektur wird damit nicht in einem Bericht sichtbar, sondern in der Fertigung.

Eine ausführliche Beschreibung des Systems, seiner Komponenten und der bestehenden Korrekturpipeline gibt Kapitel 3.

1.3 Problemstellung

Die im Projektverlauf beobachteten Schwächen des monolithischen Ansatzes lassen sich in drei klar abgrenzbare Problemdimensionen gliedern. Diese Probleme sind nicht isoliert zu betrachten – sie bedingen und verstärken sich gegenseitig – und bilden gemeinsam den wissenschaftlichen Kern, den diese Arbeit adressiert.

Problem 1: Halluzinationen bei komplexen Validierungsaufgaben

In der praktischen Arbeit mit dem System zeigte sich ein konsistentes Muster. Bei einfachen, gut strukturierten Snapshots mit wenigen, klar lokalisierbaren Fehlern liefert das System zuverlässige Korrekturen. Bei komplexen Snapshots hingegen – mit vielen verschachtelten Abhängigkeiten, mehreren gleichzeitigen Fehlern oder unvollständigen Referenzen – steigt die Rate fachlich inkorrekter Ausgaben an. Korrekturen, die syntaktisch valide erscheinen, aber inhaltlich falsche Werte oder nicht belegbare Zustandsänderungen enthalten, wurden dabei wiederholt beobachtet.

Besonders ausgeprägt war dieses Verhalten in Situationen, in denen das LLM-Korrekturvorschläge mit Referenzen oder Annahmen generierte, die nicht den dokumentierten Unternehmensregeln entsprachen – auch dann, wenn diese Regeln

explizit im Systemprompt oder im RAG-Kontext hinterlegt waren. Die Messbarkeit dieser Fehlerrate ist dabei selbst eine methodische Herausforderung. Im laufenden Betrieb existiert keine Referenz, gegen die sich eine Korrektur automatisiert prüfen ließe.

Die Ursache liegt in einer bekannten Eigenschaft monolithischer Prompts. Je mehr Validierungsregeln, Kontextinformationen und Korrekturanweisungen gleichzeitig in einem einzigen Kontext verarbeitet werden müssen, desto wahrscheinlicher ist es, dass das Modell relevante Constraints vernachlässigt oder widersprüchliche Regeln fehlerhaft auflöst. Ji et al. (2023) beschreiben dieses Muster als charakteristisch für LLM-Systeme, bei denen Aufgabenkomplexität und Kontextlänge eine kritische Schwelle überschreiten. Huang et al. (2025) ordnen es der Inferenzphase zu und führen es unter anderem auf eine unzureichende Gewichtung entfernter Kontextanteile zurück. Im vorliegenden Anwendungsfall ist dieses Problem besonders kritisch, weil eine fachlich falsche, aber syntaktisch valide JSON-Korrektur von nachgelagerten Validierungsstufen möglicherweise nicht erkannt und unbemerkt in das ERP-System übernommen werden kann.

Problem 2: Fehlende Nachvollziehbarkeit von Korrekturentscheidungen

Ein Problem, das mir im Projektalltag besonders deutlich wurde, ist die fehlende Transparenz der Entscheidungslogik bei fehlerhaften Ausgaben. Wenn das System korrekte Korrekturen liefert, begründet es seine Entscheidungen – da dies als explizite Anforderung im Systemprompt hinterlegt ist. Das eigentliche Problem tritt auf, wenn Korrekturen inhaltlich falsch sind und die mitgelieferte Begründung nicht nachvollziehbar ist oder auf Regelreferenzen verweist, die sich beim näheren Hinsehen als nicht existent oder falsch interpretiert erweisen. Turpin et al. (2023) zeigen für diesen Effekt, dass die von einem Modell ausgegebene Begründung den tatsächlichen Entscheidungsweg systematisch falsch darstellen kann, ohne dass dies an der Begründung selbst erkennbar wäre. In solchen Fällen ist weder als Entwickler noch als Fachperson rekonstruierbar, an welcher Stelle im Systemprompt oder im Entscheidungsprozess der Fehler entstanden ist.

Eine gezielte Verbesserung würde voraussetzen, dass sich der Entscheidungspfad rekonstruieren und die verantwortliche Stelle korrigieren lässt. Da die für eine Korrektur maßgeblichen Informationen gemeinsam in einen einzigen Modellaufruf eingehen, existieren jedoch keine expliziten Zwischenzustände und keine dokumentierten Entscheidungspunkte. Was bleibt, ist eine Anpassung auf Verdacht, ohne dass überprüfbar wäre, ob sie die Ursache trifft.

Hinzu kommen Nebeneffekte an Stellen, die zuvor stabil funktionierten, und auch diese bleiben mangels Zwischenzuständen unbemerkt. Je sorgfältiger das bestehende Verhalten ausdefiniert wurde, desto mehr Angriffspunkte entstehen bei jeder weiteren Änderung.

Besta et al. (2024) zeigen, dass die explizite Modellierung von Reasoning-Schritten als Graph genau hier ansetzt – indem Zwischenergebnisse sichtbar, validierbar und gezielt korrigierbar werden, ohne dass eine Anpassung an einer Stelle unkontrolliert in andere Verarbeitungsschritte ausstrahlt.

Problem 3: Mangelnde Robustheit bei variierenden Eingaben

ERP-Snapshots in produktionsnahen Umgebungen sind selten konsistent strukturiert. Im Projektverlauf begegneten mir regelmäßig Snapshots mit fehlenden Pflichtfeldern, unbekannten Feldwerten, inkonsistenten Referenzen oder strukturellen Abweichungen – Grenzfälle, die das System vor besondere Herausforderungen stellen. Das beobachtete Verhalten in solchen Situationen war instabil: Wiederholte Anfragen mit identischen Eingaben lieferten mitunter unterschiedliche Korrekturen, und stark abweichende Snapshots wurden ohne adäquate Fehlererkennung verarbeitet.

Erschwerend lösen Korrekturen häufig Kaskadeneffekte aus, weil eine behobene Inkonsistenz eine weitere aufdeckt. Das verlängert die Korrekturschleife und reduziert die Vorhersehbarkeit des Gesamtprozesses.

Es ist dabei wichtig, zwischen zwei Arten von Nichtdeterminiertheit zu unterscheiden. Zum einen ist eine gewisse Variabilität in den Formulierungen von LLM-Ausgaben systemimmanent und unvermeidbar – sie liegt in der stochastischen Natur der Modelle begründet. Sclar et al. (2024) zeigen darüber hinaus, dass bereits bedeutungserhaltende Änderungen an der Formatierung eines Prompts die Leistung eines Modells erheblich verschieben können. Instabilität ist damit kein Randphänomen, sondern eine messbare Eigenschaft. Zum anderen gibt es eine inhaltliche Instabilität, bei der dieselben Symptome zu unterschiedlichen fachlichen Korrekturen führen – obwohl die zugrundeliegenden Fehler bekannt, dokumentiert und im Systemprompt beschrieben sind.

Aus betrieblicher Sicht ist die zweite Form der Instabilität nicht hinnehmbar, weil sie eine Nachprüfung jeder einzelnen Korrektur erforderlich macht. Aus dieser Beobachtung ergab sich die Erwartung, dass eine explizite Modellierung von Verarbeitungsschritten und Zustandsübergängen den Entscheidungsraum je Teilschritt enger fasst und dadurch zu konsistenteren Korrekturen führen könnte. Diese Erwartung eliminiert die stochastische Natur des Modells nicht. Ob sie sich bestätigt, ist Gegenstand der Untersuchung und wird in Kapitel 7 geprüft.

Die wissenschaftliche Lücke

Die beschriebenen Probleme sind in der Forschungsliteratur grundsätzlich bekannt. Halluzinationen in LLM-Systemen sind umfassend dokumentiert (Huang et al., 2025; Ji et al., 2023), die Sensitivität von Modellausgaben gegenüber der

Promptgestaltung ist empirisch belegt (Sclar et al., 2024), und die Grenzen modellgenerierter Begründungen sind beschrieben (Turpin et al., 2023).

Wie weit der Einsatz solcher Systeme in industriellen Kontexten tatsächlich gediehen ist, zeigt die bereits genannte Übersichtsarbeit von Henkel et al. (2026). Von 88 ausgewerteten Publikationen bewegen sich 75,0 Prozent auf einem Reifegrad zwischen Labormuster und Prototyp, während nur 9,1 Prozent Evidenz aus einem tatsächlichen Einsatz berichten. Belastbare Aussagen über das Verhalten solcher Systeme unter realen Betriebsbedingungen sind damit insgesamt selten.

Die nächstverwandte Vorarbeit stammt von Wu et al. (2022). Sie weisen nach, dass die Verkettung mehrerer Modellaufrufe zu abgegrenzten Teilaufgaben Transparenz und Kontrollierbarkeit erhöht. Untersucht werden dort allerdings allgemeine Sprachverarbeitungsaufgaben, und die Zielgröße ist die wahrgenommene Transparenz aus einer Nutzerstudie, nicht die Übereinstimmung strukturierter Ausgaben mit einer fachlichen Referenz.

Eine explorative Literaturrecherche im Zeitraum Februar bis Juli 2026 hat keine empirische Arbeit ergeben, die den Einfluss der Verarbeitungsarchitektur auf die Qualität automatisierter Korrekturen strukturierter Planungsdaten unter Nachvollziehbarkeitsanforderungen untersucht. Die Recherche erhebt keinen Anspruch auf Vollständigkeit im Sinne einer systematischen Übersichtsarbeit. Sie stützt die Einschätzung, dass für diesen spezifischen Anwendungsfall belastbare Evidenz fehlt, sowohl hinsichtlich des Architektureinflusses als auch hinsichtlich geeigneter Messverfahren für solche Ausgaben. Diese doppelte Lücke, empirisch und methodisch, zu verkleinern ist das Anliegen der vorliegenden Arbeit.

1.4 Zielsetzung der Arbeit

Aus der beschriebenen Problemstellung und der identifizierten wissenschaftlichen Lücke leiten sich drei klar unterscheidbare Zielsetzungen ab, die gemeinsam den Beitrag dieser Arbeit definieren.

Wissenschaftliches Ziel: empirischer Architekturvergleich

Das primäre Ziel ist ein methodisch kontrollierter, empirisch fundierter Vergleich zwischen dem bestehenden monolithischen System und einer neu konzipierten Variante mit graph-basierter Architektur.

Dieser Vergleich soll auf Basis operationalisierbarer Metriken quantifizierbare Aussagen darüber liefern, ob und in welchem Ausmaß sich die untersuchten Ansätze in den Dimensionen unterscheiden, die in Abschnitt 1.5 hergeleitet werden. Dabei ist ausdrücklich nicht das Ziel, eine allgemeingültige Aussage über LLM-Architekturen zu treffen, sondern einen empirisch fundierten Beitrag für den

spezifischen Kontext produktionskritischer Validierungssysteme mit strukturierten Ausgaben zu leisten.

Methodisches Ziel: Evaluierungsrahmen für strukturierte LLM-Ausgaben

Ein eigenständiges Ziel ist die Entwicklung eines geeigneten Evaluierungsrahmens für komplexe, strukturierte LLM-Ausgaben. Etablierte Frameworks wie RAGAS sind primär für textbasierte Antworten konzipiert und lassen sich nicht ohne Weiteres auf JSON-Korrekturen anwenden (Es et al., 2024).

Wie schwierig die objektive Beurteilung solcher Ausgaben ist, habe ich im Projektverlauf selbst erfahren, insbesondere dort, wo eine Korrektur syntaktisch korrekt und inhaltlich falsch ist. Die Arbeit entwickelt und erprobt daher eine Kombination aus automatisierten Metriken, domänenexpertenbasierter Bewertung und struktureller Validierung. Baltes et al. (2026) formulieren für empirische Studien mit Sprachmodellen inzwischen Berichtspflichten, die unter anderem die Offenlegung von Prompts und Ablaufspuren sowie die Validierung von Modellausgaben gegen menschliches Urteil umfassen. Jacovi & Goldberg (2020) liefern dazu die begriffliche Grundlage, indem sie zwischen einer Begründung unterscheiden, die den tatsächlichen Entscheidungsweg abbildet, und einer, die lediglich plausibel wirkt. Beide Anforderungen prägen den in dieser Arbeit entwickelten Evaluierungsrahmen.

Praktisches Ziel: produktionsnahe Systemarchitektur

Das praktische Ziel ist die prototypische Implementierung einer graph-basierten Variante des bestehenden Systems sowie die Ableitung konkreter Designrichtlinien für den Einsatz strukturierter Systemarchitekturen in industriellen KI-Systemen. Das Ergebnis soll unmittelbar als Grundlage für eine Weiterentwicklung des bestehenden Systems nutzbar sein.

1.5 Forschungsfrage und Thesen

1.5.1 Von drei Problemdimensionen zu vier Messdimensionen

Die in Abschnitt 1.3 beschriebenen Problemdimensionen sind Beobachtungen aus dem Projektbetrieb. Für eine empirische Untersuchung mussten sie in Größen überführt werden, die sich an konkreten Systemausgaben bestimmen lassen. Dabei ergaben sich zwei Präzisierungen.

Die Halluzinationsrate ließ sich nicht als einheitliche Größe führen. Eine Korrektur kann von einem vorab festgelegten Sollwert abweichen und dennoch durch die vorgelegten Daten und Regeln gedeckt sein. In der durchgeführten Messung trat dieser Fall in jeder der drei untersuchten Bedingungen fünfzehn Mal auf.

Beides in einer gemeinsamen Rate zu führen hätte diesen Unterschied verdeckt. Die Dimension wurde deshalb in fachliche Korrektheit und Halluzinationsverhalten aufgeteilt. Hinzu kommt, dass sich von den unterschiedenen Halluzinationskategorien nicht alle über alle Bedingungen hinweg erheben lassen, weshalb der Quotientenbegriff Rate durch den Begriff Verhalten ersetzt wurde.

Aus der Robustheit wurde die Entscheidungsstabilität. Gemessen wurde, wie stabil das System bei wiederholter identischer Eingabe entscheidet. Die Robustheit gegenüber strukturell abweichenden oder gestörten Eingaben wurde nicht untersucht, weil der Testfallkatalog keine kontrollierte Variationsreihe desselben Falls enthält. Die Nachvollziehbarkeit blieb dabei unverändert.

Diese Präzisierung betrifft ausschließlich die Benennung und Gruppierung der Messgrößen. Die Messvorschriften selbst standen vor Beginn der Datenerhebung fest und wurden nicht verändert. Der genaue Zeitpunkt und der Anlass jeder Präzisierung sind in Kapitel 5 dokumentiert, die vollständigen Messvorschriften in Kapitel 6.

1.5.2 Die vier Dimensionen

Fachliche Korrektheit bezeichnet die Übereinstimmung eines Korrekturvorschlags mit einem vorab festgelegten Sollwert. Kleinste Bewertungseinheit ist die einzelne erwartete Änderung und nicht der Testfall. Geprüft wird, ob der Vorschlag die richtige Stelle im Datenbestand adressiert und dort den richtigen Wert setzt.

Halluzinationsverhalten bezeichnet das Auftreten von Korrekturen oder Begründungen, die durch die vorgelegten Daten und Regeln nicht gedeckt sind. Unterschieden werden vier Kategorien, die sich in ihrer Erhebbarkeit unterscheiden. Der Begriff Verhalten ersetzt die ursprünglich vorgesehene Rate, weil ein Quotient über alle vier Kategorien einen Nenner voraussetzen würde, den die Messung nicht durchgängig liefert.

Entscheidungsstabilität bezeichnet die Übereinstimmung der fachlichen Entscheidung bei wiederholter Ausführung desselben Falls mit identischer Eingabe. Gemeint ist die inhaltliche Entscheidung und nicht die sprachliche Formulierung der Begründung. Eingeschlossen ist die Fähigkeit, Unsicherheit auszuweisen, statt eine nicht belegbare Korrektur zu erzwingen.

Nachvollziehbarkeit bezeichnet das Ausmaß, in dem sich aus den erzeugten Artefakten rekonstruieren lässt, welcher Fehler bearbeitet, welche Regelgrundlage verwendet und welcher Datenausschnitt herangezogen wurde. Die Dimension ist graduell zu verstehen und nicht binär. Alle untersuchten Varianten erzeugen maschinell auswertbare Artefakte, sie unterscheiden sich in deren Zusammenhang, Reihenfolge und Herkunftsangabe.

1.5.3 Hauptforschungsfrage

Die zentrale Forschungsfrage dieser Arbeit lautet:

> *Inwiefern unterscheiden sich eine monolithische Systemprompt-Struktur, eine modulare Regelbereitstellung und eine graph-basierte Orchestrierung hinsichtlich fachlicher Korrektheit, Halluzinationsverhalten, Entscheidungsstabilität und Nachvollziehbarkeit bei der automatisierten Validierung und Korrektur strukturierter JSON-Daten in einem produktionskritischen Umfeld?*

Die Frage benennt drei Ansätze, vergleicht aber zwei Architekturen. Untersucht werden die monolithische Systemprompt-Struktur und die graph-basierte Systemarchitektur. Die modulare Regelbereitstellung ist keine dritte Architektur, sondern eine Erweiterung, die sich in beiden Ablaufformen einsetzen lässt. Sie wird als eigene Messbedingung geführt, weil sie andernfalls gemeinsam mit der Ablaufsteuerung verändert würde und ein beobachteter Unterschied dann keiner der beiden Eigenschaften zurechenbar wäre. Die Untersuchung arbeitet damit mit zwei Architekturen und drei Messbedingungen. Kapitel 4 beschreibt die Bedingungen technisch, Kapitel 5 begründet ihre Anordnung.

Aus der Hauptforschungsfrage ergeben sich drei Unterfragen, die jeweils einen spezifischen Aspekt des Vergleichs operationalisieren:

1.5.4 Unterfragen (zur Ergänzung/Vertiefung)

- Welche Unterschiede zeigen sich zwischen den drei untersuchten Ansätzen hinsichtlich fachlicher Korrektheit und evidenzbezogenem Halluzinationsverhalten?

Im Mittelpunkt steht nicht nur, ob sich Unterschiede zeigen, sondern auch, wie sie überhaupt messbar gemacht werden können. Etablierte Evaluierungsframeworks sind für offene Textantworten konzipiert und lassen sich nicht ohne Weiteres auf strukturierte Korrekturen übertragen, die gegen dokumentierte Regelwerke zu prüfen sind. Die Entwicklung eines geeigneten Messansatzes ist deshalb selbst Teil der Untersuchung und zugleich der methodische Beitrag dieser Arbeit.

- Wie stabil reproduzieren die drei Systemvarianten ihre fachlichen Entscheidungen bei wiederholter Ausführung identischer Testfälle, und welche Instabilitätsmuster zeigen sich über die untersuchten Fehlerszenarien?

In der praktischen Systemarbeit führten identische Eingaben mitunter zu unterschiedlichen fachlichen Korrekturen. Sclar et al. (2024) zeigen, dass Modellausgaben bereits auf bedeutungserhaltende Änderungen der Eingabe empfindlich reagieren, und Wang et al. (2023) verwenden die Streuung über mehrere Durchläufe derselben Eingabe als etabliertes Messobjekt. UF2 untersucht, ob die explizite Zerlegung des Verarbeitungsprozesses zu konsistenteren Entscheidungen führt.

Dabei wird zwischen sprachlicher Variabilität, die systemimmanent ist, und inhaltlicher Instabilität unterschieden. Nur die zweite ist Gegenstand der Messung.

- Wie unterscheiden sich die Systemvarianten hinsichtlich der verfügbaren technischen Zustands- und Ablaufinformation sowie der Fehlertoleranz in iterativen Korrekturschleifen, und inwieweit unterstützen diese Informationen die menschliche Fehlerrekonstruktion und -lokalisierung?

Anpassungen am bestehenden System setzten an vermuteten Stellen an, weil sich der Verarbeitungsschritt, in dem ein unerwünschtes Verhalten entstand, nicht bestimmen ließ. UF3 untersucht, welche Ablaufinformation die Varianten jeweils bereitstellen und ob sich damit die Ursache eines fehlerhaften Ergebnisses lokalisieren lässt. Die Frage zerfällt in einen systemseitigen und einen menschlichen Teil. Der systemseitige Teil betrachtet, welche Angaben in den Artefakten überhaupt vorhanden sind. Der menschliche Teil betrachtet, ob eine Fachperson damit tatsächlich zu einem Urteil kommt. Diese Trennung ist notwendig, weil Turpin et al. (2023) zeigen, dass modellgenerierte Begründungen den tatsächlichen Entscheidungsweg falsch darstellen können. Die in dieser Arbeit betrachtete Ablaufinformation wird deshalb vom Programmcode aufgezeichnet und nicht vom Modell berichtet.

1.6 Aufbau der Arbeit

Die Arbeit gliedert sich in neun Kapitel. Kapitel 2 legt die theoretischen Grundlagen zu Sprachmodellen, strukturierten Ausgaben, Modularisierung, graphbasierter Orchestrierung, Halluzinationen, Stabilität und Nachvollziehbarkeit dar und schließt mit der Forschungslücke. Kapitel 3 beschreibt das bestehende System in dem Zustand vor der untersuchten Weiterentwicklung. Kapitel 4 stellt die Entwicklung und Implementierung der drei Vergleichsbedingungen dar. Kapitel 5 beschreibt das Forschungsdesign, die Kontrollbedingungen und den Ablauf der Messung. Kapitel 6 operationalisiert die vier Dimensionen und beschreibt Testfallkatalog, Referenzwerte und Erhebungsinstrumente. Kapitel 7 präsentiert die Ergebnisse. Kapitel 8 diskutiert sie im Rückbezug auf die Forschungsfragen und die Literatur und benennt die Grenzen der Untersuchung. Kapitel 9 fasst die Erkenntnisse zusammen und gibt einen Ausblick.

1.7 Abgrenzung der Arbeit

Der Vergleich beschränkt sich auf zwei Systemarchitekturen, nämlich die bestehende monolithische Struktur und eine graph-basierte Systemarchitektur, im Kontext eines spezifischen Anwendungsfalls: die automatisierte Validierung und Korrektur von JSON-Snapshot-Dateien in einem Produktionsplanungssystem.

Untersucht werden drei Messbedingungen, weil die modulare Regelbereitstellung als eigenständige, von der Ablaufsteuerung unabhängige Erweiterung getrennt geführt wird. Die vierte denkbare Kombination aus vollständigem Regelwerk und graph-basierter Ablaufsteuerung ist technisch wählbar, war aber nicht Teil der Messung und wird in dieser Arbeit nicht als Bedingung geführt. Ausgangspunkt ist das bestehende Vier-Agenten-System aus dem Praxisprojekt. Ziel ist nicht die vollständige Produktivsetzung einer neuen Lösung, sondern die nachvollziehbare Bewertung, ob und in welchen Bereichen eine graph-basierte Struktur Vorteile bietet.

Unter einer graph-basierten Systemarchitektur wird in dieser Arbeit nicht nur die Struktur einzelner Prompts verstanden, sondern die strukturierte Modellierung von Daten, Systemprompts, Teilprompts, Regeln, Beispieldaten, Kontextinformationen, Tool-Ausgaben, Zwischenergebnissen und fachlichen Entscheidungsgrundlagen als gerichteter Verarbeitungsgraph. Da der Begriff „Prompt-Architektur" in der Forschungsliteratur nicht einheitlich verwendet wird – Besta et al. (2024) und Wen et al. (2024) sprechen von graph-basiertem Reasoning und graph-basierter Wissensmodellierung, nicht von Prompt-Architekturen im engeren Sinne, wird er in dieser Arbeit bewusst nicht als etablierter wissenschaftlicher Fachbegriff vorausgesetzt, sondern durch den präziseren Begriff der graph-basierten Systemarchitektur ersetzt und auf den beschriebenen Anwendungskontext eingegrenzt.

Kein Bestandteil dieser Arbeit ist die Grundlagenforschung an Large Language Models selbst. Es wird nicht untersucht, welches Sprachmodell grundsätzlich am leistungsfähigsten ist oder welches Foundation Model für industrielle Anwendungen am besten geeignet wäre. Das zugrundeliegende Modell, Azure OpenAI GPT-4.1, bleibt in beiden Vergleichsvarianten identisch, damit beobachtbare Unterschiede primär auf die Systemstruktur und nicht auf Modellunterschiede zurückgeführt werden können.

Die Arbeit grenzt sich außerdem von einer reinen Prompt-Optimierung ab. Es geht nicht primär darum, einzelne Formulierungen innerhalb eines Prompts zu verbessern. Vielmehr wird untersucht, ob die Zerlegung eines komplexen Validierungs- und Korrekturprozesses in mehrere definierte Verarbeitungsschritte zu besseren und nachvollziehbareren Ergebnissen führt. Der Schwerpunkt liegt damit auf der Systemstruktur und dem Entscheidungsfluss, nicht auf rein sprachlicher Prompt-Formulierung. Nicht untersucht wird die Robustheit gegenüber strukturell abweichenden, gestörten oder unvollständigen Eingaben. Der Testfallkatalog enthält keine kontrollierte Variationsreihe desselben Falls. Gemessen wurde ausschließlich die Stabilität der fachlichen Entscheidung bei wiederholter identischer Eingabe. Der Begriff Robustheit ist in dieser Arbeit entsprechend auf diese Bedeutung eingegrenzt.

Ebenfalls nicht untersucht wird die Wartbarkeit der verglichenen Varianten. Für sie wurde kein Messinstrument entwickelt, und Aussagen über Entwicklungsaufwand,

Änderungsfreundlichkeit oder Fehlersuche im Entwicklungsprozess sind aus den erhobenen Daten nicht ableitbar. Beobachtungen aus dem Projektverlauf werden als solche gekennzeichnet und nicht als Untersuchungsergebnis geführt.

Der entwickelte Evaluierungsrahmen erhebt keinen Anspruch auf universelle Übertragbarkeit. Klassische Validierungsmechanismen prüfen syntaktische Korrektheit, Schemakonformität und Pflichtfelder. Sie beurteilen jedoch nicht, ob eine Korrektur fachlich sinnvoll ist oder Folgefehler auslöst. Technische Validierung wird deshalb mit fachlicher Expertenbewertung und kontrollierten Testfällen kombiniert.

Etablierte Evaluierungsframeworks decken den vorliegenden Gegenstand nicht ab. RAGAS ist für Retrieval-Augmented-Generation-Szenarien konzipiert, in denen eine natürlichsprachliche Antwort gegen einen abgerufenen Kontext bewertet wird (Es et al., 2024). Im untersuchten System entstehen jedoch strukturierte Korrekturvorschläge, Validierungsergebnisse, Werkzeugausgaben und operative Entscheidungen. Für sie existieren dokumentierte Regelwerke und ein Zielsystem, gegen das eine Ausgabe inhaltlich geprüft werden kann. RAGAS wurde in dieser Arbeit deshalb nicht eingesetzt.

An seine Stelle tritt eine Kombination mehrerer Verfahren. Die fachliche Korrektheit wird gegen eine über kontrollierte Fehlerinjektion konstruierte Referenz geprüft. Der Zustand nach einer Korrektur wird durch die Validierungs-Engine des Zielsystems neu berechnet und ausgewertet. Das Halluzinationsverhalten wird anhand vier unterschiedener Kategorien erhoben, deren jeweilige Erhebungsgrenzen ausgewiesen werden. Die Entscheidungsstabilität wird durch wiederholte Ausführung identischer Eingaben bestimmt. Die technische Nachvollziehbarkeit wird an der maschinell aufgezeichneten Ablauf- und Herkunftsinformation gemessen. Ergänzt wird dies durch eine strukturierte Expertenbewertung, die insbesondere den Teil von UF3 abdeckt, der maschinell nicht beantwortbar ist. Diese Kombination folgt den Berichtspflichten, die Baltes et al. (2026) für empirische Studien mit Sprachmodellen formulieren. Kapitel 6 beschreibt die Verfahren im Einzelnen.

Die Untersuchung beschränkt sich auf ausgewählte Fehler- und Testfälle aus dem Smart-Planning-Kontext, etwa leere Pflichtfelder, ungültige Referenzen und inkonsistente Datenbeziehungen. Eine vollständige Abdeckung aller denkbaren Fehlerfälle industrieller Produktionsplanung ist nicht Ziel der Arbeit.

Zusammenfassend untersucht die Arbeit nicht allgemein, ob LLMs in der Produktion sinnvoll eingesetzt werden können, und auch nicht ausschließlich, ob bessere Prompts bessere Antworten erzeugen. Der Fokus liegt auf der Frage, ob sich die untersuchten Ansätze in einem LLM-gestützten Validierungs- und Korrektursystem hinsichtlich fachlicher Korrektheit, Halluzinationsverhalten, Entscheidungsstabilität und Nachvollziehbarkeit messbar unterscheiden.

2 Theoretische Grundlagen

2.1 Large Language Models und strukturierte Ausgaben

Large Language Models (LLMs) sind neuronale Sprachmodelle, die auf Basis massiver Textkorpora trainiert werden und in der Lage sind, natürlichsprachliche Eingaben zu verarbeiten und kontextuell relevante Ausgaben zu erzeugen. Modelle wie GPT-4 von OpenAI basieren auf der Transformer-Architektur (Vaswani et al., 2017), deren Self-Attention-Mechanismus Abhängigkeiten zwischen beliebig weit auseinanderliegenden Textstellen modelliert, ohne sie sequenziell durchlaufen zu müssen. In industriellen Anwendungsfällen werden LLMs zunehmend nicht nur für generative Aufgaben, sondern auch für strukturierte Verarbeitungsprozesse eingesetzt - etwa zur Validierung, Klassifikation oder Korrektur von Daten.

In den hier betrachteten Anwendungsfällen erzeugt ein Modell keine freie Textantwort, sondern eine Ausgabe in einem vorgegebenen Format, etwa ein JSON-Objekt mit festgelegten Feldern. Solche Formatvorgaben sind nicht folgenlos für die Qualität der Antwort. Tam et al. (2024) vergleichen freie und formatgebundene Ausgaben über mehrere Aufgabentypen hinweg und finden einen aufgabenabhängigen Effekt. Bei Aufgaben, die mehrschrittiges Schließen erfordern, sinkt die Leistung unter Formatzwang, und zwar umso deutlicher, je strenger die Vorgabe ist. Bei Klassifikationsaufgaben kehrt sich der Effekt um, weil der eingeschränkte Antwortraum Fehler bei der Auswahl der Antwort reduziert.

Für ein System, das strukturierte Daten prüft und korrigiert, sind beide Aufgabentypen zugleich vorhanden. Die Zuordnung eines Fehlers zu einer Fehlerklasse ist eine Klassifikationsaufgabe, die Bestimmung eines konkreten Korrekturwerts aus mehreren Datenquellen erfordert Schließen. Aus der Literatur lässt sich deshalb nicht ableiten, in welche Richtung sich ein höherer Anteil formatgebundener Ausgaben auf die Gesamtqualität auswirkt.

2.2 Prompting und Kontextbereitstellung

Prompt Engineering bezeichnet die Disziplin der gezielten Gestaltung von Eingaben (Prompts) an ein LLM, um dessen Ausgabeverhalten systematisch zu steuern. Die Qualität und Struktur eines Prompts haben direkten Einfluss auf die Korrektheit, Konsistenz und Nachvollziehbarkeit der Modellantworten.

Grundlegende Techniken umfassen Zero-Shot-Prompting, Few-Shot-Prompting mit Beispielen sowie strukturierte Anweisungsformate. Mit zunehmender Aufgabenkomplexität stoßen einfache Prompting-Strategien jedoch an ihre Grenzen, was zur Entwicklung fortgeschrittener Ansätze wie Chain-of-Thought oder graphbasierter Architekturen geführt hat.

Neben der Formulierung eines Prompts bestimmt auch die Menge des gleichzeitig bereitgestellten Kontexts das Ausgabeverhalten. Ji et al. (2023) beschreiben in ihrer Übersicht, dass faktisch inkorrekte Ausgaben mit steigender Aufgabenkomplexität und zunehmender Kontextlänge häufiger auftreten. Huang et al. (2025) ordnen diesen Effekt der Inferenzphase zu und führen ihn unter anderem darauf zurück, dass die Aufmerksamkeit des Modells sich ungleichmäßig über den Kontext verteilt und entfernt liegende Anteile schwächer gewichtet werden.

Daraus folgt eine Gestaltungsfrage, die für diese Arbeit zentral ist. Wird einem Modell das gesamte verfügbare Regelwissen vorgelegt, unabhängig davon, welcher Teil davon für den vorliegenden Fall einschlägig ist, steigt die Kontextlänge ohne entsprechenden Informationsgewinn. Wird dagegen vorab ausgewählt, welche Regeln relevant sind, verlagert sich ein Teil der Entscheidung aus dem Modellaufruf heraus in einen vorgelagerten Schritt. Beide Varianten werden in dieser Arbeit als eigene Bedingungen untersucht.

2.3 Monolithische Bereitstellung

Ein monolithischer Systemprompt ist ein einzelner, zentraler Prompt, der die gesamte Steuerungslogik eines LLM-basierten Systems in einem Textblock bündelt. Darin werden typischerweise die Rolle des Modells, Verarbeitungsregeln, Validierungslogik, Ausgabeformate und Fehlerbehandlung gemeinsam definiert. Dieser Ansatz ist in der Praxis weit verbreitet, da er konzeptionell einfach ist und keine zusätzliche Infrastruktur erfordert.

Mit steigender Komplexität zeigen sich jedoch charakteristische Schwächen.

Da alle Regeln und Kontextinformationen gemeinsam in einen Aufruf eingehen, wächst die Länge des Prompts unabhängig davon, welcher Teil für den vorliegenden Fall einschlägig ist. Huang et al. (2025) und Ji et al. (2023) beschreiben, dass faktisch inkorrekte Ausgaben mit steigender Kontextlänge häufiger auftreten, und führen dies unter anderem auf die ungleichmäßige Gewichtung entfernt liegender Kontextanteile zurück (siehe Abschnitt 2.2).

Entscheidungen des Modells sind zudem nicht auf einzelne Verarbeitungsschritte zurückführbar, weil der Prozess nach außen als eine Operation erscheint.

Jacovi & Goldberg (2020) fassen Nachvollziehbarkeit als graduelle Eigenschaft, die daran hängt, welche Zwischeninformation überhaupt zugänglich ist. Hinzu kommt eine Beobachtung aus dem Projektbetrieb, für die keine Literaturbelege herangezogen werden: Änderungen an einem Aspekt der Logik wirken sich auf das Gesamtverhalten aus, ohne dass sich die Reichweite der Änderung vorab bestimmen ließe.

2.4 Modularisierung und Zerlegung

Der Gedanke, eine komplexe Aufgabe nicht in einem einzigen Modellaufruf zu bearbeiten, sondern in mehrere abgegrenzte Teilschritte zu zerlegen, ist unabhängig von der konkreten technischen Umsetzung untersucht worden.

Wu et al. (2022) prägen dafür den Begriff der AI Chain. Sie verketten mehrere Modellaufrufe zu jeweils klar umrissenen Teilaufgaben und weisen in einer Nutzerstudie nach, dass die Zerlegung sowohl die Qualität der Ergebnisse als auch die Transparenz und Kontrollierbarkeit aus Sicht der Nutzenden erhöht. Der Effekt entsteht dabei nicht allein durch bessere Ausgaben, sondern dadurch, dass ein fehlerhafter Zwischenschritt überhaupt erst als solcher sichtbar und gezielt korrigierbar wird.

Khot et al. (2023) untersuchen die Zerlegung als Prompting-Strategie. Eine Aufgabe wird in Teilaufgaben aufgeteilt, für die jeweils spezialisierte Anweisungen zuständig sind, und die Ergebnisse werden anschließend zusammengeführt. Der Ansatz erlaubt es, einzelne Teilschritte auszutauschen oder durch nicht modellbasierte Verfahren zu ersetzen, ohne die übrige Kette zu verändern.

Madaan et al. (2023) beschreiben mit Self-Refine ein iteratives Muster, bei dem eine erzeugte Ausgabe bewertet und auf Grundlage dieser Bewertung überarbeitet wird. Die Rückmeldung stammt dort vom Modell selbst. In einem System, das gegen ein Zielsystem arbeitet, kann sie ebenso aus einer externen Prüfung stammen, was die Bewertung von einer Selbsteinschätzung unabhängig macht.

Für die vorliegende Arbeit sind diese Befunde die theoretische Grundlage der Erwartung, dass eine Zerlegung wirksam sein könnte. Sie sind zugleich in zweifacher Hinsicht abzugrenzen. Untersucht werden dort allgemeine Sprachverarbeitungsaufgaben und nicht die Korrektur strukturierter Daten gegen ein dokumentiertes Regelwerk. Und die Zielgrößen sind Aufgabenerfolg und wahrgenommene Transparenz, nicht die Übereinstimmung einer Ausgabe mit einer fachlichen Referenz, das Auftreten nicht belegter Aussagen oder die Stabilität über Wiederholungen. Ob sich die berichteten Vorteile unter diesen anderen Zielgrößen einstellen, ist offen und Gegenstand dieser Untersuchung.

2.5 Graphbasierte Orchestrierung

Der Begriff der graph-basierten Systemarchitektur wird in der Forschungsliteratur nicht einheitlich verwendet. Besta et al. (2024) und Wen et al. (2024) sprechen von graph-basiertem Reasoning und graph-basierter Wissensmodellierung, nicht von Prompt-Architekturen im engeren Sinne. In dieser Arbeit wird der Begriff daher explizit definiert und auf den beschriebenen Anwendungskontext eingegrenzt: Unter einer graph-basierten Systemarchitektur wird nicht nur die Struktur einzelner Prompts verstanden, sondern die strukturierte Modellierung des gesamten Verarbeitungsprozesses als gerichteter Graph – umfassend Prompts, Datenflüsse, Regelreferenzen, Tool-Ausgaben und Zwischenzustände.

Konkret bedeutet das: Die Steuerungslogik wird auf mehrere spezialisierte Knoten verteilt, die über definierte Zustandsübergänge miteinander verbunden sind. Jeder Knoten übernimmt eine klar abgegrenzte Teilaufgabe. Ob ein Knoten dafür einen Modellaufruf benötigt oder deterministisch arbeitet, hängt von der Aufgabe ab und ist keine Eigenschaft der Architektur. Der Gesamtprozess wird als gerichteter Graph modelliert, in dem der Zustand des Systems explizit verwaltet und bei jedem Übergang aktualisiert wird.

Der Ansatz wird in der Literatur mit mehreren strukturellen Eigenschaften begründet. Jeder Knoten sei unabhängig entwickelbar, testbar und austauschbar. Da jeder Verarbeitungsschritt einem definierten Knoten entspreche, ließen sich Entscheidungen auf einzelne Stationen im Graph zurückverfolgen und Fehler auf dem verursachenden Knoten isolieren. Neue Verarbeitungsschritte könnten als zusätzliche Knoten ergänzt werden, ohne bestehende Logik zu verändern. Ob sich diese Eigenschaften in einem produktionsnahen System mit strukturierten Ausgaben tatsächlich in messbaren Unterschieden niederschlagen, ist Gegenstand der vorliegenden Untersuchung.

In der Praxis werden graph-basierte Architekturen durch Frameworks wie LangGraph realisiert, das auf dem LangChain-Ökosystem aufbaut und eine deklarative Definition von Agenten-Workflows als zustandsbehaftete Graphen ermöglicht.

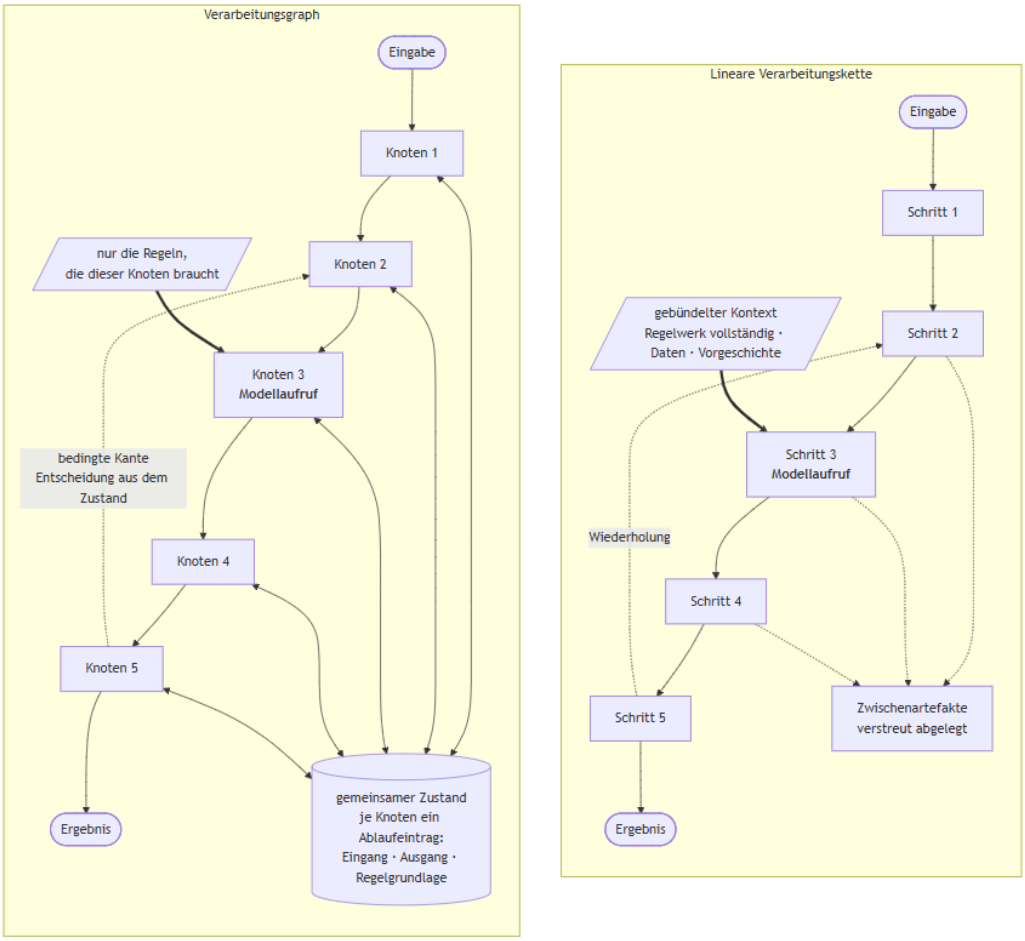


Abbildung 1: Schematische Darstellung einer graph-basierten Architektur: Verarbeitungsschritte als Knoten, Übergänge als Kanten.

Graph-of-Thoughts[1] und verwandte Ansätze

Der Graph-of-Thoughts-Ansatz (GoT) von Besta et al. (2024) stellt eine konzeptuelle Weiterentwicklung gegenüber linearen Reasoning-Methoden wie Chain-of-Thought (CoT) dar. Während CoT das Reasoning eines LLMs als lineare Abfolge von Gedankenschritten modelliert, repräsentiert GoT einzelne Gedanken als Knoten in einem gerichteten Graphen. Abhängigkeiten zwischen Gedanken werden als Kanten modelliert, wodurch Operationen wie die Aggregation, Verfeinerung und Verzweigung von Zwischenergebnissen möglich werden. Besta et al. evaluieren den Ansatz an abstrakten Aufgabenstellungen wie Sortier- und Mengenoperationen sowie an Aufgaben zur Dokumentzusammenführung. Eine Übertragbarkeit auf die Validierung strukturierter Produktionsdaten ist damit nicht belegt und wird in dieser Arbeit nicht vorausgesetzt.

[1] **Graph of Thoughts:** Ansatz, bei dem Denk- oder Verarbeitungsschritte als Knoten eines Graphen und ihre Beziehungen als Kanten dargestellt werden.

Wen et al. (2024) erweitern diesen Ansatz im MindMap-Framework, indem sie externe Knowledge Graphs mit Graph-of-Thoughts-Strukturen verbinden. Sie zeigen, dass die Kombination strukturierten Weltwissens mit graph-basiertem Reasoning die Antwortqualität von LLMs in komplexen Wissensdomänen deutlich steigert - ein Prinzip, das für wissensintensive industrielle Validierungsaufgaben besonders relevant ist.

Der Begriff des Graphen wird in diesen Arbeiten in einer anderen Bedeutung verwendet als in der vorliegenden Untersuchung. Besta et al. (2024) modellieren einzelne Gedankenschritte des Modells als Knoten und erlauben es, diese zu verzweigen, zusammenzuführen und zu verfeinern. Wen et al. (2024) verbinden diese Struktur mit einem externen Wissensgraphen, in dem das Domänenwissen selbst als Knoten und Kanten vorliegt. In beiden Fällen wird der Inhalt der Verarbeitung zum Graphen.

2.6 Halluzinationen und Belegbarkeit

Als Halluzination bezeichnet man in der LLM-Forschung das Phänomen, bei dem ein Modell Ausgaben produziert, die faktisch inkorrekt, widersprüchlich oder nicht durch den gegebenen Kontext gedeckt sind – und dies mit scheinbarer Überzeugung tut. Ji et al. (2023) klassifizieren Halluzinationen in ihrer umfassenden Übersicht in zwei Hauptkategorien: intrinsische Halluzinationen, bei denen die Ausgabe dem gegebenen Kontext widerspricht, und extrinsische Halluzinationen, bei denen die Ausgabe nicht verifizierbar ist.

Für die Bewertung strukturierter Ausgaben ist eine weitere Unterscheidung erforderlich, die in der Halluzinationsliteratur angelegt, aber selten explizit gemacht wird. Eine Ausgabe kann von einem vorab festgelegten Sollwert abweichen und dennoch durch die Daten gedeckt sein, die dem Modell vorlagen. Das ist keine Halluzination im Sinne der oben genannten Definitionen, sondern eine belegte, aber nicht der Referenz entsprechende Entscheidung. Umgekehrt kann eine Ausgabe mit dem Sollwert übereinstimmen und trotzdem auf einer Begründung beruhen, die sich auf nicht vorhandene Regeln stützt.

Fachliche Unrichtigkeit und Halluzination sind damit zwei verschiedene Eigenschaften einer Ausgabe. Werden sie in einer gemeinsamen Kennzahl geführt, geht die Unterscheidung verloren. Die vorliegende Arbeit trennt sie deshalb in zwei Messdimensionen, deren Operationalisierung Kapitel 6 beschreibt.

Ob sich Halluzinationen durch architektonische Maßnahmen verringern lassen, ist eine offene empirische Frage und Gegenstand dieser Untersuchung.

2.7 Stabilität und Sensitivität von Modellausgaben

Sprachmodelle erzeugen ihre Ausgaben durch Abtastung aus einer Wahrscheinlichkeitsverteilung. Eine gewisse Variabilität in der Formulierung ist damit systemimmanent. Für ein System, das Entscheidungen trifft, ist jedoch nicht die sprachliche, sondern die inhaltliche Variabilität von Bedeutung.

Sclar et al. (2024) untersuchen, wie empfindlich Modelle auf Änderungen reagieren, die den Inhalt eines Prompts nicht verändern. Getestet werden Trennzeichen, Groß- und Kleinschreibung, Reihenfolge von Beispielen und ähnliche Oberflächenmerkmale. Über eine Menge solcher bedeutungserhaltenden Varianten hinweg beobachten sie erhebliche Leistungsunterschiede desselben Modells bei derselben Aufgabe. Daraus folgt, dass eine an einer einzelnen Promptvariante gemessene Leistung wenig über das tatsächliche Verhalten aussagt und dass Streuung als eigene Eigenschaft zu berichten ist.

Wang et al. (2023) machen diese Streuung methodisch nutzbar. In ihrem Verfahren wird dieselbe Aufgabe mehrfach bearbeitet und aus den Ergebnissen dasjenige gewählt, das am häufigsten auftritt. Für die vorliegende Arbeit ist weniger das Auswahlverfahren als die zugrundeliegende Annahme relevant, nämlich dass mehrfache Ausführung derselben Eingabe eine messbare Verteilung erzeugt und diese Verteilung selbst informativ ist.

Beide Arbeiten begründen, warum die Stabilität einer Entscheidung als eigene Größe zu erheben ist und nicht als Nebenprodukt der Korrektheit. Sie begründen zugleich, wie: durch wiederholte Ausführung identischer Eingaben und den Vergleich der inhaltlichen Ergebnisse. Nicht abgedeckt ist damit die Frage, wie ein System auf strukturell abweichende Eingaben reagiert. Diese Unterscheidung zwischen Wiederholungsstabilität und Störungsrobustheit wird in Kapitel 6 aufgegriffen.

2.8 Nachvollziehbarkeit, Faithfulness und Provenienz

Ein System, das Entscheidungen begründet, wirkt nachvollziehbar. Ob es das ist, hängt davon ab, ob die Begründung den tatsächlichen Entscheidungsweg abbildet.

Jacovi & Goldberg (2020) trennen diese beiden Eigenschaften begrifflich. Eine Erklärung ist plausibel, wenn sie überzeugend wirkt, und sie ist treu, wenn sie den Prozess korrekt wiedergibt, der zur Ausgabe geführt hat. Beides fällt nicht zusammen.

Eine überzeugende Erklärung kann sogar irreführend sein, wenn sie Vertrauen erzeugt, das der tatsächliche Ablauf nicht rechtfertigt. Sie plädieren daher dafür, Treue graduell danach zu bewerten, in welchem Ausmaß und für welchen Zweck eine Erklärung den tatsächlichen Prozess wiedergibt.

Turpin et al. (2023) belegen die Relevanz dieser Unterscheidung empirisch. Sie zeigen, dass Modelle Begründungen erzeugen, die den tatsächlich ausschlaggebenden Einfluss auf die Antwort nicht nennen, und dass diese Auslassung an der Begründung selbst nicht erkennbar ist. Eine modellgenerierte Begründung ist damit als Beleg für den Entscheidungsweg ungeeignet, unabhängig davon, wie schlüssig sie klingt.

Für ein produktionsnahes System folgt daraus, dass Nachvollziehbarkeit nicht durch modellgenerierte Erläuterungen, sondern durch unabhängig aufgezeichnete Ablaufdaten entsteht. Dazu gehört, welcher Verarbeitungsschritt zu welchem Zeitpunkt ausgeführt wurde, welche Eingaben er erhalten hat und welche Wissensgrundlage dabei tatsächlich geladen war. Diese Herkunftsangabe ist eine Beobachtung des ausführenden Programms und keine Selbstauskunft des Modells.

Diese Unterscheidung zwischen aufgezeichneter und berichteter Information ist für die vorliegende Arbeit tragend. Sie bestimmt, welche Artefakte in Kapitel 4 erzeugt werden, und sie bestimmt in Kapitel 6, was als Beleg für Nachvollziehbarkeit gilt.

2.9 Evaluation von LLM-Systemen im industriellen Kontext

Die Evaluation von LLM-Systemen stellt eine eigenständige methodische Herausforderung dar, die im vorliegenden Anwendungsfall besondere Sorgfalt erfordert. Klassische Evaluierungsframeworks wie RAGAS (Es et al., 2024) sind primär für Retrieval-Augmented-Generation-Szenarien konzipiert, bei denen eine natürlichsprachliche Antwort anhand eines abgerufenen Kontextes bewertet wird.

Für strukturierte JSON-Korrekturvorschläge, Validierungsergebnisse, Tool-Ausgaben und operative Systementscheidungen reichen klassische RAG-Metriken jedoch nicht aus. JSON-Korrekturen können anhand dokumentierter Regelwerke inhaltlich geprüft werden, zugleich aber trotz formaler Gültigkeit fachlich falsch sein oder neue Folgefehler erzeugen, was textbasierte Metriken nicht erfassen. Im laufenden Betrieb existiert keine Referenz, gegen die sich eine Korrektur automatisiert prüfen ließe. Für eine kontrollierte Untersuchung muss eine solche Referenz erst hergestellt werden.

Aus diesem Grund wird in dieser Arbeit ein kombinierter Evaluierungsansatz verfolgt, der technische Validierung und domänenexpertenbasierte Bewertung kombiniert. Die folgende Tabelle gibt einen Überblick über die eingesetzten Methoden, ihre Eignung für den vorliegenden Anwendungsfall und ihre Rolle im Gesamtrahmen.

Für die Anlage einer solchen Untersuchung liegen methodische Vorgaben vor. Runeson & Höst (2009) beschreiben für Fallstudien in der Softwaretechnik die Kombination mehrerer unabhängiger Datenquellen als Qualitätsmerkmal, weil sich Einzelbefunde damit gegenseitig stützen oder relativieren lassen. Baltes et al. (2026) formulieren speziell für empirische Studien mit Sprachmodellen konkrete Berichtspflichten, darunter die Offenlegung von Prompts und Laufzeitspuren, die Angabe von Modellversion und Parametern sowie die Validierung von Modellausgaben gegen menschliches Urteil. Beide prägen das in Kapitel 5 beschriebene Vorgehen.

Bewertungsebene	**Erhobene Dimension**	**Rolle**
Abgleich gegen eine über Fehlerinjektion konstruierte Referenz	fachliche Korrektheit	primär
Erneute Validierung durch die Validierungs-Engine des Zielsystems	funktionaler Endzustand, Folgefehler	primär
Kategorisierung nach vier Halluzinationsarten	Halluzinationsverhalten	primär
Wiederholte Ausführung identischer Eingaben	Entscheidungsstabilität	primär
Maschinell aufgezeichnete Ablauf- und Herkunftsinformation	technische Nachvollziehbarkeit	primär
Strukturierte Expertenbewertung	wahrgenommene Qualität, menschliche Fehlerrekonstruktion	ergänzend

Tabelle 1: Die verwendeten Bewertungsebenen

Für strukturierte Korrekturen, Werkzeugausgaben und operative Entscheidungen greifen textbasierte Metriken nicht. RAGAS wurde deshalb nicht eingesetzt, klassische Metriken wie BLEU oder ROUGE ebenfalls nicht, weil sie Wortüberlappung messen und keine fachliche Korrektheit. Kapitel 6 beschreibt die eingesetzten Verfahren im Einzelnen.

2.10 Forschungslücke und Zusammenfassung

Die in diesem Kapitel dargestellten Arbeiten decken die Bestandteile des untersuchten Problems einzeln ab. Halluzinationen in Sprachmodellen sind umfassend beschrieben und klassifiziert (Huang et al., 2025; Ji et al., 2023). Die Empfindlichkeit von Modellausgaben gegenüber der Eingabegestaltung ist empirisch belegt (Sclar et al., 2024). Die Grenzen modellgenerierter Begründungen sind bekannt (Turpin et al., 2023), ebenso die begriffliche Unterscheidung zwischen plausibler und treuer Erklärung (Jacovi & Goldberg, 2020). Die Wirksamkeit der Zerlegung komplexer Aufgaben in abgegrenzte Teilschritte ist untersucht (Khot et al., 2023; Madaan et al., 2023; Wu et al., 2022), ebenso die explizite Modellierung von Verarbeitungsstrukturen als Graph (Besta et al., 2024; Wen et al., 2024)

Was fehlt, ist die Zusammenführung dieser Befunde an einem industriellen Anwendungsfall mit strukturierten Ausgaben. Die Arbeiten zur Zerlegung untersuchen allgemeine Sprachverarbeitungsaufgaben und erheben Aufgabenerfolg oder wahrgenommene Transparenz. Die Arbeiten zu graphbasierten Strukturen evaluieren an abstrakten Aufgabenstellungen. Keine von ihnen misst, ob eine strukturierte Ausgabe mit einer fachlichen Referenz übereinstimmt, ob sie durch die vorgelegten Daten gedeckt ist, ob sie bei Wiederholung stabil bleibt und ob sich ihr Zustandekommen im Nachhinein rekonstruieren lässt.

Wie selten belastbare Evidenz aus tatsächlichen Einsätzen ist, zeigt die Übersichtsarbeit von Henkel et al. (2026). Von 2.341 gesichteten Publikationen zu foundation-modellbasierten Agenten in der industriellen Automatisierung wurden 88 ausgewertet. Drei Viertel davon bewegen sich auf einem Reifegrad zwischen Labormuster und Prototyp, und nur 9,1 Prozent berichten Evidenz aus einem tatsächlichen Einsatz. Als eigene Problemkategorie führen die Autoren dabei ungültige oder im Zielsystem nicht ausführbare strukturierte Ausgaben.

Daraus ergibt sich die Lücke, die diese Arbeit adressiert. Für die automatisierte Korrektur strukturierter Planungsdaten unter Nachvollziehbarkeitsanforderungen liegt keine empirische Untersuchung zum Einfluss der Verarbeitungsarchitektur vor. Ebenso fehlt ein Messverfahren, das die dafür relevanten Eigenschaften erfasst. Beide Lücken bedingen einander, denn ohne geeignetes Messverfahren lässt sich der Architektureinfluss nicht bestimmen. Die Entwicklung dieses Verfahrens ist deshalb Teil der Untersuchung und nicht nur ihre Voraussetzung.

3 Untersuchungsgegenstand: das bestehende System

Dieses Kapitel beschreibt das System, an dem die Untersuchung durchgeführt wurde, in dem Zustand vor der in Kapitel 4 beschriebenen Weiterentwicklung. Neben der Architektur und den einzelnen Komponenten wird auch der Ablauf der bestehenden Korrekturpipeline dargestellt. Abschließend werden die Eigenschaften der Ausgangsarchitektur zusammengefasst, die für den späteren Vergleich der Systemvarianten relevant sind.

Die Beschreibung des bestehenden Systems bildet die Grundlage für den späteren Architekturvergleich. Sie zeigt, welche Funktionen und Verarbeitungsschritte bereits in der Ausgangsarchitektur vorhanden waren und welche Änderungen erst im Rahmen der in Kapitel 4 beschriebenen Weiterentwicklung vorgenommen wurden. Diese Abgrenzung ist notwendig, damit beobachtete Unterschiede zwischen den untersuchten Varianten den jeweiligen Änderungen nachvollziehbar zugeordnet werden können.

3.1 Projektkontext

Die Arbeit entstand im Rahmen eines dualen Studiums bei der CANCOM Austria AG in der Abteilung Digital Makers. Das untersuchte System wurde dort im Rahmen eines Kundenprojekts konzipiert und implementiert. Das Unternehmen des Auftraggebers wird in dieser Arbeit nicht genannt. Sämtliche verwendeten Daten, Regeltexte und Systemausgaben sind anonymisiert beziehungsweise auf freigegebene Testbestände beschränkt.

Durch meine Mitwirkung an Konzeption, Implementierung und Betrieb des Systems besteht ein direkter Zugang zur technischen Architektur und zum beobachtbaren Systemverhalten. Dadurch können die Systemgrenzen und die für die Untersuchung relevanten Verarbeitungsschritte detailliert nachvollzogen werden. Gleichzeitig ist diese Nähe zum Untersuchungsgegenstand als methodische Einschränkung zu berücksichtigen und wird in Kapitel 5 gesondert behandelt.

3.2 Das Zielsystem Smart Planning

Smart Planning ist ein Produktionsplanungssystem, das als Integrationsschicht zwischen dem ERP-System des Auftraggebers und den operativen Fertigungsprozessen liegt. Es verwaltet Fertigungsaufträge, Ressourcenpläne, Maschinenbelegungen und Zeitplanungen und stellt den jeweils aktuellen Planungszustand als strukturierte JSON-Snapshot-Datei bereit.

Ein einzelner Snapshot bildet den Planungszustand vollständig ab und umfasst zwischen 70.000 und über 200.000 Einträge. Enthalten sind unter anderem Produkte, Maschinenkennungen, Produkttypen, Mitarbeiterverfügbarkeiten und -qualifikationen sowie sämtliche produktionsrelevanten Prozessparameter. Auf dieser Grundlage erzeugt Smart Planning den konkreten Produktionsplan, also die operative Zuordnung von Fertigungsaufträgen, Ressourcen und Zeitfenstern. Die Qualität des Snapshots bestimmt damit unmittelbar die Qualität der Planung, denn fehlerhafte oder inkonsistente Eingangsdaten führen zu fehlerhaften Planungsergebnissen.

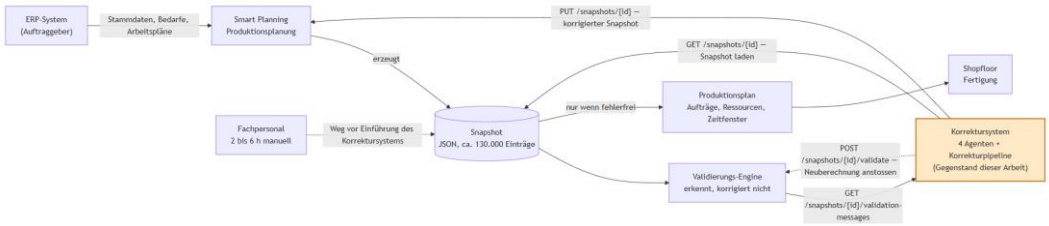


Abbildung 2: Systemkontext des untersuchten Korrektursystems, vom ERP über Smart Planning und den Snapshot bis zur Rückschreibung in den Shopfloor.

Smart Planning verfügt über eine eigene Validierungs-Engine. Sie erkennt Fehler im Snapshot und lokalisiert sie, ist jedoch nicht in der Lage, erkannte Fehler zu korrigieren. Die Korrektur erfolgte bis zur Einführung des hier untersuchten Systems vollständig manuell durch spezialisiertes Fachpersonal und erforderte je Snapshot regelmäßig zwischen zwei und über sechs Stunden.

Eine Eigenschaft der Validierungs-Engine ist für das Verständnis der späteren Verarbeitung wesentlich. Die Validierung wird nicht selbsttätig neu berechnet, wenn ein Snapshot verändert wird. Ein Abruf der Validierungsmeldungen liefert nur den zuletzt berechneten Stand. Eine belastbare Aussage über den Zustand nach einer Korrektur setzt daher voraus, dass die Neuberechnung ausgelöst und ihr Abschluss abgewartet wird. Wie damit im Rahmen der Untersuchung umgegangen wurde, beschreibt Abschnitt 5.4.

```
{
  "articles": [
    {
      "articleId": "100005",
      "articleName": "Produkt A",
      "workPlanId": "SP10        SP01",
      "workItemConfigs": [
        { "workItemKey": "VOAR01", "rampUpTime": 15, "netTimeFactor": 0 },
        { "workItemKey": "HE01",   "rampUpTime": 30, "netTimeFactor": 1 }
      ],
      "relDensityMin": 1.017,
      "relDensityMax": 1.02,
      "hygieneImpactClass": 50,
      "standardPackaging": "71125",
      "minOptimalBatchSize": 200,
      "maxOptimalBatchSize": 26000,
      "departmentId": "20100",
      "departmentName": "Abteilung Ost"
    }
  ],
  "equipment": [
    {
      "equipmentId": "4c350df3-325d-427c-a8bf-5309d3b71910",
      "equipmentKey": "AAR01",
      "name": "Abfuellanlage 01",
      "predecessors": ["MTO02", "MTO01", "MTO04", "MTO05"],
      "throughput": 570,
      "batchSizeMin": 0,
      "batchSizeMax": 200,
      "functions": ["Abfueller"],
      "usageFactor": 1,
      "qualification": "Abfuellen-Q1",
      "workItems": ["ABF01"],
      "departments": ["10100", "10300", "20100", "20200"],
      "wasteVolume": 20,
      "hygieneRelevant": true,
      "lastHygieneImpactClass": 99,
      "defaultDispatcherGroup": "20"
    }
  ],
  "demands": [
    {
```

```
      "demandId": "D100005_001",
      "articleId": "100005",
      "quantity": 98.9,
      "packaging": "71125",
      "successor": "02A11310451014000",
      "dueDate": "2026-05-19T23:59:59Z",
      "dispatcherGroup": "20"
    }
  ]
}
```


Code 1: Anonymisierter Auszug aus einem Snapshot

Ein vollständiger Auszug findet sich in Anhang A.

Eine Validierung einer fehlerhaften Snapshot könnte wie folgt aussehen:


```
[
  {
    "level": "ERROR",
    "message": "[validate_work_plan_ids] Missing work plans for the following IDs: WP_3221"
  },
  {
    "level": "WARNING",
    "message": "[validate_equipment_department_presence] Equipment without departments (Disponent2): ABD01 (c04b2710-d66a-4759-9f3c-8902b1c39083), APU03 (543b2138-b611-4a0f-bac0-7255d57d2009), BGS05 (a90a828b-3857-4fc2-b386-5534f9b093b5)"
  },
  {
    "level": "ERROR",
    "message": "[validate_packaging_equipment_compatibility_references] Invalid packaging equipment compatibility predecessor references: Packaging 70381 references non-existent equipment predecessor: COMPAT_EQUIPMENT_NOT_FOUND"
  },
  {
    "level": "ERROR",
    "message": "[validate_packaging_references] Article 100005 references non-existent packaging: 99998"
  }
]
```


Code 2: Validierung Snapshot

3.3 Das Vier-Agenten-System

Das im Projekt entwickelte Korrektursystem setzt an der beschriebenen Lücke an. Es bezieht fehlerhafte Snapshots aus Smart Planning, analysiert und korrigiert sie mithilfe eines LLM-gestützten Agentensystems und lädt die korrigierten Snapshots automatisiert zurück. Smart Planning ist damit zugleich Datenquelle und Ziel der Korrekturausgabe - das Korrektursystem liegt genau in dem Prozessschritt, der dem eigentlichen Planungslauf vorausgeht.

Die Verarbeitung verteilt sich auf vier Agenten. Als Sprachmodell dient in allen Komponenten Azure OpenAI GPT-4.1. Die Infrastruktur ist in Azure betrieben und über Terraform als Infrastructure as Code beschrieben.

Agent	***Rolle***	***Technische Grundlage***
Orchestrierungs-Agent	Routing, Mehrschrittplanung, Konsolidierung der Rückmeldungen	Azure OpenAI GPT-4.1
RAG-Agent	Wissensabfrage aus Produktionsdokumenten und Validierungsregelwerken	Azure AI Search
Smart-Planning-Agent	Ausführung der Korrekturpipeline und Systemintegration	Werkzeug- und Pipelinesteuerung, Smart-Planning-API
Chat-Agent	Allgemeine Rückfragen, Erklärung von Fehlerklassen, fachliche Einordnung	Azure OpenAI GPT-4.1

Tabelle 2: Die vier Agenten

Der Orchestrator-Agent ist die primäre Schnittstelle zum Endnutzer. Er empfängt Anfragen, analysiert die Nutzerintention, plant die notwendigen Verarbeitungsschritte und koordiniert die Aufgabenverteilung – nicht nach fest programmierten if-then-Regeln, sondern durch kontextsensitive Entscheidungsfindung auf Basis von LLM-Calls mit einem spezialisierten Systemprompt. Die Qualität dieser Koordination – insbesondere die Fähigkeit, mehrstufige Korrekturprozesse zu planen und bei Fehlschlägen alternative Strategien zu wählen – erwies sich im Projektverlauf als eine der zentralen praktischen Herausforderungen.

Der RAG-Agent greift über Azure AI Search [2] auf eine strukturierte Wissensbasis aus Produktionsdokumenten und Validierungsregelwerken zu. Die Korrekturregeln sind dabei in natürlicher Sprache verfasst – nicht als Code oder hard-kodierte if-then-Logik – sodass Fachexperten ohne Programmierkenntnisse die Regeln erstellen und pflegen können. Die Qualität der Retrieval-Ergebnisse hat sich als entscheidender Faktor für die inhaltliche Korrektheit der Systemausgaben erwiesen.

Der Smart-Planning-Agent ist die operative Kernkomponente des Systems. Er verfügt über ein dediziertes Toolset aus zehn spezialisierten Werkzeugen und vier vordefinierten Pipelines für wiederkehrende Workflows. Diese Tools ermöglichen es ihm, Snapshots direkt aus Smart Planning abzuholen, sie mit der systemseitigen Validierungs-Engine zu prüfen, Fehler mithilfe von LLM-Aufrufen zu analysieren, strukturierte Korrekturvorschläge zu generieren, diese auf die Snapshot-Daten anzuwenden und die korrigierten Snapshots schließlich automatisiert zurückzuladen.

Der Chat-Agent verfügt über initiales Trainingswissen zu allgemeinen Produktionsregeln und typischen Fehlermustern. Er übernimmt Aufgaben, die weder spezifisches Retrieval noch eine direkte Systemintegration erfordern – etwa allgemeine Rückfragen zum System, Erklärungen von Fehlerklassen oder die Einordnung von Korrekturen in einen übergeordneten fachlichen Kontext.

Ein fünfter Agent für E-Mail-Verarbeitung existiert im System, ist jedoch nicht Gegenstand dieser Arbeit.

[2] Cloudbasierter Such- und Indexierungsdienst von Microsoft Azure, der unter anderem Volltext-, semantische und vektorbasierte Suche über Wissensbestände ermöglicht.

3.4 Die Korrekturpipeline

Die eigentliche Korrekturlogik liegt in einer Kette eigenständiger Laufzeitskripte. Jedes Skript ist ein eigenständig ausführbares Programm mit eigener Kommandozeilenschnittstelle. Der Smart-Planning-Agent ruft sie als Unterprozesse auf.

#	SCHRITT (SKRIPT)	AUFGABE
1	validate_snapshot.py	Snapshot-Kennung lesen, Validierung über die API abrufen, Ergebnis nach snapshot-validation.json schreiben
2	identify_error_llm.py	Rohfehler analysieren, zu bearbeitenden Fehler auswählen, Suchmodus und Suchwert festlegen, Kontextsuche als Unterprozess anstoßen
3	generate_correction_llm.py	Regeltext, Analyseergebnis und Suchkontext zu einem Prompt bündeln, strukturierten Korrekturvorschlag erzeugen
4	validate_correction_schema_llm.py	Vorschlag gegen das Pydantic-Ausgabeschema prüfen. Bei Verstoß das Modell erneut aufrufen (max_retries=5)
5	apply_correction.py	Geprüften Vorschlag auf snapshot-data.json anwenden, Original zuvor in den Iterationsordner sichern
6	update_snapshot.py	Korrigierten Snapshot per PUT /snapshots/{snapshotId} zurückschreiben
7	validate_snapshot.py	Erneuter Abruf der Validierung nach der Korrektur (derselbe Schritt wie 1)

Tabelle 3: Einige Skripte (Tools) der Korrekturpipeline

Der Ablauf der vollständigen Korrektur folgt einer festen Schrittfolge. Zunächst werden die Validierungsmeldungen des Snapshots abgerufen. Ein Modellaufruf analysiert die Rohfehler, wählt den zu bearbeitenden Fehler aus und legt die Suchstrategie fest. Anschließend wird der relevante Ausschnitt aus dem Snapshot extrahiert. Ein zweiter Modellaufruf erzeugt daraus einen strukturierten Korrekturvorschlag. Ein dritter prüft diesen Vorschlag gegen das Ausgabeschema und fordert bei Verstößen eine korrigierte Fassung an. Der geprüfte Vorschlag wird auf die Snapshot-Daten angewendet, der veränderte Snapshot zurückgeschrieben und die Validierung erneut abgerufen.

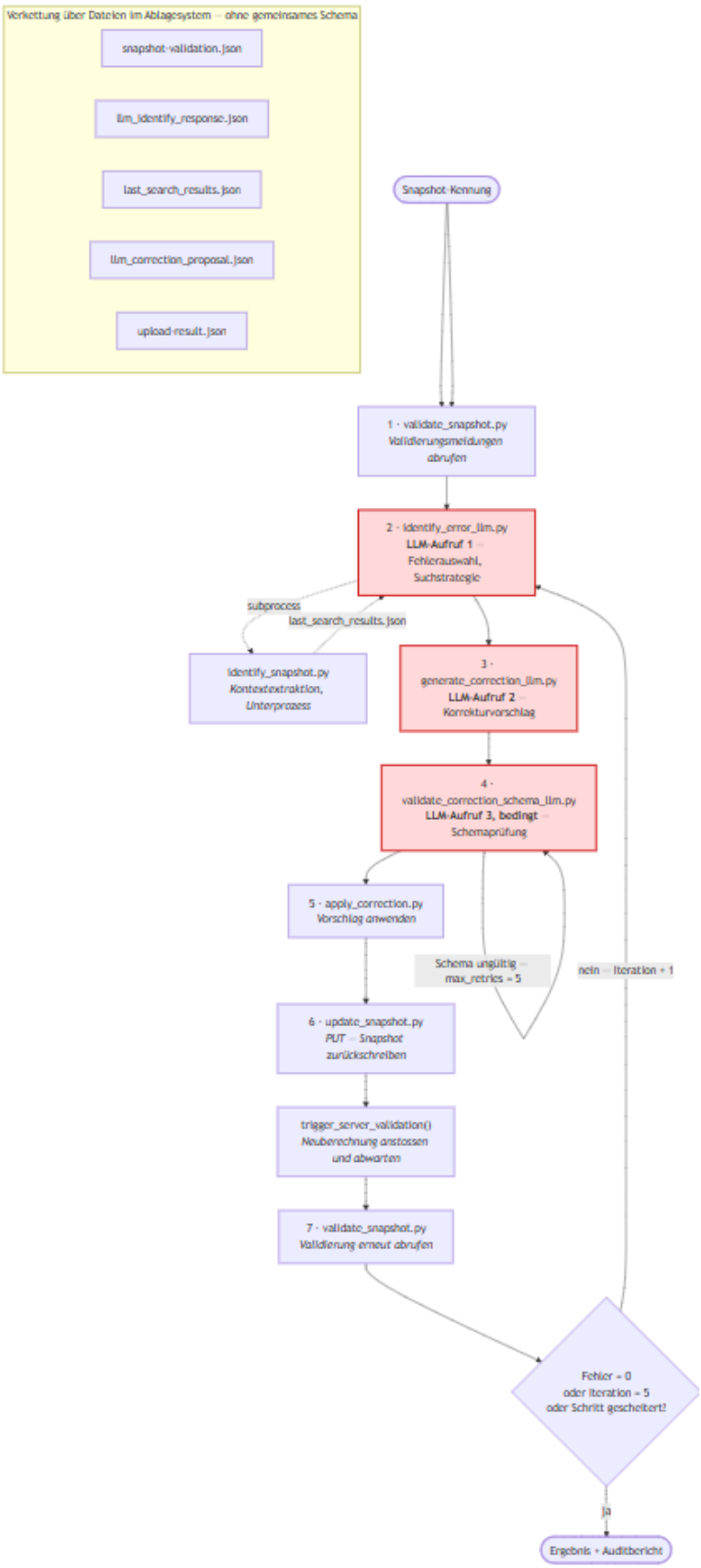


Abbildung 3: Ablauf der Korrekturpipeline von der Snapshot-Aufnahme bis zur Rückschreibung.

Da ein Snapshot mit 70.000 bis über 200.000 Einträgen nicht vollständig an ein Sprachmodell übergeben werden kann, arbeitet die Pipeline mit gezielten Kontextextraktionen. Der extrahierte Ausschnitt umfasst typischerweise 1.000 bis 4.000 Zeilen. Die Extraktion ist damit ein eigener, für die Qualität der nachfolgenden Korrektur bestimmender Verarbeitungsschritt.

Die Art der Datenweitergabe zwischen den einzelnen Skripten ist für den späteren Architekturvergleich relevant. Die Verarbeitungsschritte sind überwiegend über Dateien im Ablagesystem miteinander verbunden. Ein Skript schreibt sein Ergebnis in eine Datei, die anschließend vom nächsten Verarbeitungsschritt eingelesen wird. Auf diese Weise entstehen verschiedene maschinell auswertbare Artefakte, darunter die Ergebnisse der Fehleranalyse, Suchergebnisse, Korrekturvorschläge und Validierungsstände. Diese Informationen liegen jedoch verteilt über mehrere Dateien vor und werden nicht in einem gemeinsamen, strukturierten Ablaufzustand zusammengeführt. Dadurch besteht keine durchgängige maschinelle Verknüpfung zwischen den einzelnen Verarbeitungsschritten, ihren Ein- und Ausgaben sowie der jeweils verwendeten Regelgrundlage.

Der Prozess ist iterativ angelegt. Eine Korrektur deckt häufig eine weitere Inkonsistenz auf, die im vorangegangenen Durchlauf nicht sichtbar war. Die Schleife wiederholt sich daher, bis der Snapshot als fehlerfrei gilt oder eine Obergrenze von fünf Durchläufen erreicht ist. Gesteuert wird sie durch eine Zählschleife im Programmcode des Agenten, die nach jedem Durchlauf die verbliebene Fehlerzahl prüft.

Jede Änderung wird in einer Metadatendatei protokolliert und am Ende in einem Bericht zusammengefasst, der die durchgeführten Korrekturen, ihre Begründungen und die angewendeten Regeln aufführt.

Im praktischen Betrieb lag die Bearbeitungszeit für Snapshots mit drei bis fünf Fehlern typischerweise bei etwa ein bis drei Minuten. Die zuvor notwendige manuelle Korrektur nahm dagegen in vergleichbaren Fällen etwa zwei bis sechs Stunden in Anspruch. Diese Werte stammen aus Beobachtungen im Projektbetrieb und wurden nicht im Rahmen der kontrollierten Evaluation dieser Arbeit erhoben. Sie dienen daher ausschließlich zur Einordnung des praktischen Ausgangsproblems und werden nicht als Untersuchungsergebnis verwendet.

Zwei weitere Bestandteile des Produktivsystems liegen im Verarbeitungspfad und sind für das Verständnis der Messanordnung relevant. Ein Freigabemechanismus kann die Anwendung von Korrekturen an eine menschliche Bestätigung binden. Ein episodisches Gedächtnis speist frühere menschliche Korrekturentscheidungen als Belege in den Korrekturprompt ein und kann den Modellwert bei Objektgleichheit überschreiben.

3.5 Die Regelbereitstellung

Das bestehende System basiert auf monolithischen Systemprompts. Die gesamte Verarbeitungslogik jedes Agenten – Rollendefinition, Validierungsregeln, Korrekturanweisungen, Ausgabeformatierung und Beispielfälle – ist in einem einzigen, umfangreichen Prompt-Dokument gebündelt. Die fachlichen Korrekturregeln umfassen 937 Zeilen und 34.899 Zeichen. Dieser Ansatz war in der initialen Entwicklungsphase pragmatisch und schnell umsetzbar, da er keine vorgelagerte Architekturentscheidung erforderte und eine schnelle Iteration über Prompt Inhalte ermöglichte. Ein Beispiel hierfür:

```
name: density-values
applies_to: [DENSITY_VALUES]
description: Numerische Werte aus array_context ableiten (Gruppenfilter,
Median)
---
# Card: DENSITY_VALUES  (`[validate_density_values]`)
Deriving a numeric field value from array_context — worked examples.
The general principle ("filter to the same group first") lives in
`_core.md`; this card carries the detailed examples.
## KRITISCH: Domain-Intelligence bei array_context
**Evidenzreihenfolge:**
1. Dichte desselben `articleId` aus einem bestätigten gültigen Snapshot
2. Gleicher Artikelstamm beziehungsweise eindeutig gleiche Produktvari-
ante
3. Median mehrerer Artikel mit gleichem `departmentId` und `workPlanId`
Nur positive, numerische Vergleichswerte verwenden. `relDensityMin` muss
nach dem Fix `> 0`
sein. Wenn `relDensityMax` vorhanden ist, zusätzlich `relDensityMin <=
relDensityMax` prüfen,
ohne `relDensityMax` unnötig zu verändern. Bei weniger als zwei ver-
gleichbaren Artikeln und ohne historischen Treffer keine Dichte erfinden.
3. Finde fehlende Sequenznummer
```

Code 3: Auszug aus dem Regelwerk

Ein vollständiger Auszug findet sich in Anhang B.

In der ursprünglichen Fassung des Systems wurde dieses Dokument vollständig und unabhängig vom vorliegenden Fehlerbild in den Korrekturprompt eingesetzt. Das Regelwerk erscheint dabei nicht als Systemnachricht, sondern als Bestandteil der dynamisch aufgebauten Nutzernachricht.

Die Systemnachrichten der drei Modellaufrufe sind kurze, aufgabenspezifische Anweisungen von jeweils einer Zeile.

Der Begriff der monolithischen Systemprompt-Struktur bezieht sich in dieser Arbeit nicht auf eine bestimmte technische Nachrichtenrolle. Gemeint ist vielmehr die gebündelte Bereitstellung aller für die Korrektur relevanten Informationen innerhalb eines Modellaufrufs. Dazu zählen die Rollenanweisung, das vollständige Regelwerk, die Fehlermeldung, das Analyseergebnis, der Suchkontext, das Ausgabeschema und die Formatvorgaben. Ob einzelne Bestandteile technisch als System- oder Nutzernachricht übergeben werden, ist für die Untersuchung nicht ausschlaggebend. Entscheidend ist, dass sie gemeinsam in den Korrekturaufruf eingehen.

Der zentrale Korrekturprompt enthält in dieser Form unter anderem das vollständige Regelwerk, die ursprüngliche Validierungsmeldung, das Ergebnis der Fehleranalyse, den extrahierten Snapshot-Kontext, Vorgaben zur Belegbarkeit von Werten, Regeln für die Bestimmung des Zielpfads, Vorgaben für den Fall einer notwendigen manuellen Prüfung, das Ausgabeschema, die Anforderung einer deutschsprachigen Begründung sowie eine Konfidenzeinstufung.

In einem früheren Weiterentwicklungsstand des Systems wurde die Regelbereitstellung modularisiert. Anstelle des vollständigen Regelwerks wird dabei ein gemeinsamer Kernbestand zusammen mit den Regelkarten geladen, die für den jeweils erkannten Fehlertyp relevant sind. Der ursprüngliche Bereitstellungspfad blieb weiterhin verfügbar und kann über einen Konfigurationsschalter ausgewählt werden. Die kartenbasierte Bereitstellung bildet den aktuellen Standardmodus. Dass beide Varianten weiterhin im System vorhanden sind, ermöglicht ihre getrennte Betrachtung innerhalb dieser Arbeit. Kapitel 4 greift diesen Unterschied im Rahmen der Vergleichsbedingungen erneut auf.

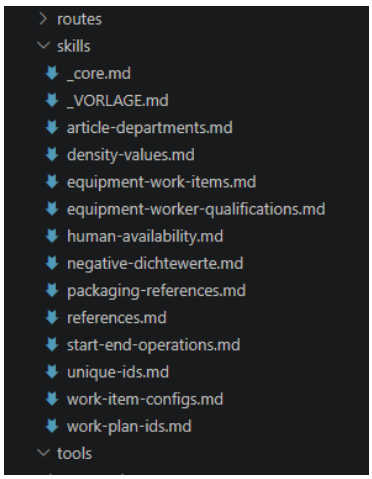


Abbildung 4: Auszug aus dem Regelwerk: Aufbau einer Regelkarte mit Bedingung, Prüflogik und zugeordnetem Validator.

3.6 Eigenschaften der Ausgangsarchitektur

Eigenschaften der Ausgangsarchitektur, aus denen sich die Fragestellung ergibt:

Die Ausgangsarchitektur ist kein einzelner Modellaufruf. Sie besteht bereits aus mehreren eigenständigen Verarbeitungsschritten mit klar abgegrenzten Aufgaben, verfügt über eine Iterationslogik, eine Schemaprüfung mit Wiederholungsversuchen und eine Reihe maschinell auswertbarer Zwischenartefakte. Diese Feststellung ist für die Aussagekraft der gesamten Untersuchung wesentlich. Eine Ausgangsbedingung, die einfacher dargestellt wird, als sie ist, würde den Vergleich entwerten, weil ein beobachteter Unterschied dann ebenso gut auf die Vereinfachung wie auf die Intervention zurückgehen könnte.

Der monolithische Charakter der Ausgangsarchitektur liegt an zwei anderen Stellen:

Erstens werden im Schritt der Korrekturerzeugung mehrere unterschiedliche Informationsbestandteile innerhalb eines einzelnen Modellaufrufs zusammengeführt. Dazu gehören unter anderem das vollständige Regelwerk, die Fehlermeldung, das Analyseergebnis, der extrahierte Suchkontext sowie die Vorgaben für Ausgabeformat und Zielschema. Im bestehenden Ablauf wird dabei nicht strukturiert dokumentiert, welche dieser Informationen für die erzeugte Korrektur herangezogen wurden. Mit zunehmender Aufgaben- und Kontextkomplexität steigt zugleich die Anforderung an das Modell, relevante Informationen aus einem umfangreichen Kontext zuverlässig zu berücksichtigen. (Huang et al., 2025; Ji et al., 2023)

Zweitens wird der Kontrollfluss nicht über einen gemeinsamen strukturierten Zustand abgebildet. Die einzelnen Verarbeitungsschritte erzeugen zwar Zwischenartefakte, diese liegen jedoch verteilt vor und werden nicht zu einem durchgängigen Ablaufzustand zusammengeführt. (Jacovi & Goldberg, 2020) Dadurch ist zwar nachvollziehbar, welche einzelnen Ergebnisse erzeugt wurden, die Abfolge der Verarbeitungsschritte und die jeweils verwendeten Eingaben und Regelgrundlagen müssen für eine nachträgliche Rekonstruktion jedoch aus mehreren Artefakten zusammengesetzt werden.

Aus dieser zweiten Eigenschaft folgt eine praktische Konsequenz, die im Projektverlauf wiederholt zu beobachten war. Zeigt eine Korrektur ein unerwünschtes Verhalten, lässt sich der Verarbeitungsschritt, in dem es entstanden ist, nicht zuverlässig bestimmen. Eine Anpassung setzt daher an einer vermuteten Stelle an, ohne dass überprüfbar wäre, ob sie die Ursache trifft, und ohne dass sich absehen ließe, welche zuvor stabilen Verhaltensweisen sie verändert.

Ein Beispiel aus dem Projektverlauf verdeutlicht die Reichweite. In einem Korrekturvorschlag berief sich die Begründung auf Vergleichsartikel aus einer bestimmten Abteilung, die tatsächlich herangezogenen Artikel stammten aus einer anderen. Der Fehler fiel nur auf, weil eine Person die Angabe nachrechnete. Das System hielt nicht fest, welches Vergleichskollektiv tatsächlich verwendet worden war, und konnte die Abweichung daher weder anzeigen noch verhindern. (Turpin et al., 2023)

Diese beiden Eigenschaften, die Bündelung im Korrekturschritt und die Implizitheit des Kontrollflusses, bilden den Ansatzpunkt der in Kapitel 4 beschriebenen Entwicklung.

3.7 Zusammenfassung

Der Untersuchungsgegenstand ist ein LLM-gestütztes Korrektursystem für strukturierte Planungsdaten, das aus vier Agenten und einer mehrstufigen Korrekturpipeline besteht. Das System ist für den fachlichen Einsatz konzipiert, bei dem Anwender die erzeugten Korrekturvorschläge nachvollziehen und sich auf deren fachliche Qualität verlassen können müssen. Die Pipeline umfasst bereits mehrere abgegrenzte Verarbeitungsschritte, eine Iterationslogik und maschinell auswertbare Zwischenartefakte. Monolithisch ist sie in zwei Hinsichten: zum einen in der ungefilterten Bündelung von Regelwerk, Kontext und Formatvorgaben im Schritt der Korrekturerzeugung, zum anderen in einem impliziten und nicht zusammenhängend dokumentierten Kontrollfluss. Neben der ursprünglichen Form der Regelbereitstellung existiert eine modularisierte Variante, ohne dass der frühere Verarbeitungspfad entfernt wurde. Diese Eigenschaften bilden die Grundlage für den Aufbau der in Kapitel 4 entwickelten Vergleichsbedingungen.

4 Entwicklung und Implementierung der Vergleichsbedingungen

Dieses Kapitel beschreibt, was für die Untersuchung entwickelt wurde. Es beginnt mit dem Entwicklungsverlauf, weil aus ihm die Anordnung der Vergleichsbedingungen folgt, und beschreibt anschließend die graphbasierte Variante in der tatsächlich implementierten Form. Warum die drei Bedingungen in dieser Anordnung verglichen werden, behandelt Kapitel 5 - hier geht es darum, was sie technisch sind.

Die Darstellung beschränkt sich auf das, was zum Verständnis des Unterschieds zwischen den Bedingungen, zur Beurteilung der Vergleichbarkeit und zur grundsätzlichen Wiederholbarkeit der Untersuchung erforderlich ist.

4.1 Entwicklungsverlauf und Ableitung der Vergleichsbedingungen

Das System entstand zunächst in der in Kapitel 3 beschriebenen Form. Das Regelwerk wurde vollständig und unabhängig vom vorliegenden Fehlerbild in den Korrekturprompt eingesetzt, und der Ablauf wurde durch Programmcode im Agenten gesteuert.

Im Verlauf der Weiterentwicklung entstand die modulare Regelbereitstellung. Der Gedanke dahinter ist unabhängig von jeder Ablaufarchitektur. Wenn dem Modell nur die Regeln vorgelegt werden, die für den erkannten Fehlertyp einschlägig sind, sinkt die Menge gleichzeitig zu berücksichtigender Anweisungen, ohne dass fachliche Information verloren geht. Umgesetzt wurde das über einen Konfigurationsschalter, der den bisherigen Pfad erhielt und die kartenbasierte Bereitstellung zur Voreinstellung machte.

Für den geplanten Architekturvergleich ergab sich daraus zunächst eine Anordnung, die ich später verworfen habe. Die graphbasierte Variante wäre gegen den ursprünglichen Zustand gemessen worden, und die modulare Regelbereitstellung wäre dabei als Teil der neuen Variante mitgelaufen, weil sie zum Zeitpunkt der Entwicklung bereits die Standardkonfiguration war. Ein beobachteter Unterschied hätte dann zwei gleichzeitig veränderte Eigenschaften abgebildet, nämlich die Form der Regelbereitstellung und die Form der Ablaufsteuerung. Welche von beiden ihn trägt, wäre aus den Daten nicht bestimmbar gewesen.

Die modulare Regelbereitstellung ist kein notwendiger Bestandteil einer graphbasierten Ablaufsteuerung. Sie ist eine eigenständige Erweiterung, die sich in beiden Ablaufformen einsetzen lässt, und sie kann deshalb als eigene Bedingung geführt werden. Aus dem geplanten Zweiervergleich wurde damit eine Anordnung mit drei Messbedingungen, in der die mittlere genau eine der beiden Eigenschaften isoliert.

Die Untersuchung vergleicht weiterhin zwei Architekturen. Die mittlere Bedingung ist keine dritte Architektur, sondern dieselbe Ablaufsteuerung wie die Ausgangsbedingung bei veränderter Regelbereitstellung. Der Vergleich zwischen der monolithischen Systemprompt-Struktur und der graphbasierten Systemarchitektur bleibt der Hauptvergleich, die mittlere Bedingung macht ihn deutbar.

Erst danach entstand die graphbasierte Variante. Sie überführt die bestehende Verarbeitungskette in eine explizit modellierte Abfolge benannter Knoten mit einem gemeinsamen, typisierten Zustandsobjekt und einer aufgezeichneten Ablaufspur.

4.2 Die drei Vergleichsbedingungen

	ABLAUFSTEUERUNG	REGELBEREITSTELLUNG	BESCHREIBUNG
A	prozedural, impliziter Zustand	vollständiges Regelwerk	Bereitstellungspfad vor Einführung des Konfigurationsschalters
B	prozedural, impliziter Zustand	modulare Regelkarten	Standardkonfiguration des Systems
C	graphbasiert, expliziter Zustand	modulare Regelkarten	die in dieser Arbeit entwickelte Variante

Tabelle 4: Die drei Vergleichsbedingungen

Bedingung A bezeichnet den ursprünglichen Bereitstellungspfad, der im Anwendungscode fest verdrahtet war und weiterhin über einen Konfigurationsschalter auswählbar ist. Sie bildet damit einen erhaltenen Entwicklungsstand des Systems ab und keine eigens für die Untersuchung konstruierte Vereinfachung.

Bedingung B entspricht der Standardkonfiguration des Systems und unterscheidet sich von A ausschließlich in der Regelbereitstellung.

Bedingung C unterscheidet sich von B ausschließlich in der Ablaufsteuerung.

Die vierte denkbare Kombination aus vollständigem Regelwerk und graphbasierter Ablaufsteuerung ist technisch wählbar, war aber nicht Teil der Messung und wird in dieser Arbeit nicht als Bedingung geführt.

4.3 Gemeinsame Bestandteile

Damit ein beobachteter Unterschied auf die jeweils variierte Eigenschaft zurückgeführt werden kann, mussten alle übrigen Bestandteile über die drei Bedingungen hinweg gleich bleiben. Bei der Umsetzung habe ich mich deshalb an eine durchgängige Regel gehalten: bestehende Funktionen werden aufgerufen, nicht nachgebaut. Die graphbasierte Variante verwendet dieselben Funktionen für Fehleranalyse, Kontextextraktion, Korrekturerzeugung, Schemaprüfung, Anwendung, Upload und Validierung wie die prozedurale. Sie ruft sie aus benannten Knoten heraus auf und hält ihre Ergebnisse in einem gemeinsamen Zustand fest.

Identisch sind das Sprachmodell und seine Parametrisierung, die Prompttexte der drei Modellaufrufe, die Kontextextraktion aus dem Snapshot, die Schemaprüfung samt der Zahl zulässiger Wiederholungsversuche, die Obergrenze der fachlichen Iterationen sowie Anwendung, Upload und Auslösung der erneuten Validierung. Die Einzelwerte und ihre Herkunft dokumentiert Abschnitt 5.3.

Eine Wiederholungsschicht ist nicht in allen Bedingungen vorhanden. Die prozedurale Steuerung wiederholt einen gescheiterten Pipelineschritt bis zu zweimal, der Graph kennt keine Knotenwiederholung. Diese Schicht greift bei einem fehlgeschlagenen Unterprozess und nicht bei einem fachlich unzureichenden Ergebnis, sie betrifft also die Prozessstabilität und nicht die Korrekturlogik. In den Messläufen ist kein Lauf als abgebrochen verzeichnet, weshalb sie vermutlich nicht gegriffen hat. Kapitel 8 führt die Asymmetrie als Einschränkung, sie wirkt zugunsten von A und B.

Für einen weiteren Punkt habe ich von der architektonisch saubereren Lösung abgesehen: Der Modellaufruf der Fehlerklassifikation liefert neben der Fehlerauswahl auch einen Vorschlag, welche Regelkarten einschlägig sein könnten. Fachlich gehört dieser Vorschlag in den Schritt der Regelzuordnung, und er ließe sich aus dem Prompt entfernen. Da dieser Prompt in allen drei Bedingungen identisch ist, hätte eine Änderung allein für die graphbasierte Variante einen Unterschied eingeführt, der nicht die Ablaufsteuerung betrifft.

Da Sclar et al. (2024) zeigen, dass bereits bedeutungserhaltende Promptänderungen die Leistung verschieben, wurde die Trennung als Zuständigkeitsänderung umgesetzt und nicht als Promptänderung.

4.4 Regelbereitstellung

Die beiden Bereitstellungsformen unterscheiden sich in Auswahl und Umfang des vorgelegten Regeltexts.

Im vollständigen Modus wird das gesamte Regelwerk in den Korrekturprompt eingesetzt, unabhängig davon, welcher Fehler vorliegt. Der vorgelegte Text ist über alle Fälle hinweg identisch.

Im modularen Modus wird ein gemeinsamer Kernbestand geladen und um diejenigen Regelkarten ergänzt, die dem erkannten Fehlertyp zugeordnet sind. Beide Wege der Auswahl laufen in einer einzigen Auflösungsfunktion zusammen, sodass genau eine Stelle bestimmt, welcher Regeltext dem Modell vorliegt.

> **Eine Gegenüberstellung eines Regelwerkabschnitts mit der entsprechenden Regelkarte findet sich in Anhang C.**

Der modulare Bestand ist größer als der monolithische, der je Prompt vorgelegte Text ist kleiner. Das Aufteilen erzeugt Wiederholung im Bestand und Ersparnis im Einzelfall. Beide Werte beschreiben eine Eigenschaft der Bedingungen und sind kein Ergebnis der Untersuchung.

Die Zuordnung von Fehlertyp zu Regelkarte erfolgt über zwei Wege. Der erste ist deterministisch. Der Validator-Tag aus der Fehlermeldung wird gegen eine Deklaration in der Karte geprüft, und für die abgedeckten Tags ist die Zuordnung eindeutig und konfliktfrei. Der zweite Weg ist es nicht. Der Klassifikationsschritt schlägt aus einem Verzeichnis aller Karten zusätzliche Karten vor, und dieser Vorschlag stammt aus einem Modellaufruf. In den auswertbaren Läufen der Bedingung C schwankte die Zahl geladener Karten deshalb zwischen zwei und vier, obwohl der deterministische Weg je Fehler auf genau eine Karte führt. Eine Karte ohne Tag-Deklaration wurde in einem Lauf ausschließlich über den Modellvorschlag geladen.

Die modulare Bereitstellung ist damit nicht vollständig deterministisch, und der vorgelegte Regeltext ist auch bei identischem Fehlerbild nicht in jedem Lauf identisch. Das ist als Eigenschaft der Bedingung festzuhalten und nicht als Störgröße zu behandeln, es gehört zu dem, was die kartenbasierte Bereitstellung tatsächlich ist.

Nicht alle Validator-Tags, die das Zielsystem ausgeben kann, haben eine zuständige Karte. Tritt ein solcher Fehler auf, wird nur der Kernbestand geladen. In den Messfällen ist dieser Fall nicht eingetreten, alle dort aufgetretenen Tags sind durch eine Karte abgedeckt. Für den Betrieb außerhalb der Messung bleibt es eine Grenze des Kartensystems.

Der Übergang von der vollständigen zur modularen Bereitstellung verändert Auswahl und Umfang des Regeltexts gleichzeitig.

Beide Anteile lassen sich in dieser Anordnung nicht trennen, der Vergleich bildet den Gesamteffekt der Bereitstellungsform ab.

4.5 Entwurfsentscheidungen der graphbasierten Variante

Der Begriff Graph bezeichnet in dieser Arbeit den Programmablauf und nicht das verarbeitete Material. Verarbeitungsschritte sind Knoten, Übergänge zwischen ihnen sind Kanten, während der Snapshot JSON bleibt und die Regelkarten Textdokumente (bzw. MD-Dateien[3]) bleiben. Die Abgrenzung zu wissensgraphbasierten Ansätzen und zu Graph of Thoughts beschreibt Abschnitt 2.5. Sie ist der Grund, warum die dort genannten Referenzimplementierungen hier keine Verwendung finden.

Näher liegt der Ansatz von Wu et al. (2022), die zeigen, dass die Verkettung mehrerer Modellaufrufe zu abgegrenzten Teilaufgaben Transparenz und Kontrollierbarkeit für Nutzende erhöht, und von Khot et al. (2023), die die Zerlegung komplexer Aufgaben in modulare Teilschritte gegenüber der Bearbeitung in einem Zug untersuchen. Beide Arbeiten begründen, warum eine Zerlegung überhaupt eine Wirkung haben sollte. Sie untersuchen allgemeine Sprachverarbeitungsaufgaben und erheben wahrgenommene Transparenz in Nutzerstudien, nicht fachliche Korrektheit strukturierter Ausgaben gegen eine Referenz.

Als Rahmenwerk dient LangGraph in Version 1.2.11, das eine deklarative Definition zustandsbehafteter Abläufe erlaubt und die Zustandsführung übernimmt.

LangGraph ist ein Framework zur Entwicklung zustandsbasierter Anwendungen mit Large Language Models. Es ermöglicht, einzelne Verarbeitungsschritte als Knoten eines Graphen zu modellieren und deren Übergänge über Kanten zu steuern. Dadurch lassen sich auch Verzweigungen, Schleifen, Zustandsübergänge und komplexere Ablaufentscheidungen explizit abbilden.

Verwendet werden davon die Graphdefinition, die bedingten Kanten und die Kompilierung. Die im Rahmenwerk enthaltenen Modellanbindungen, vorgefertigten Agenten und Wiederholungsrichtlinien bleiben ungenutzt, weil die Knoten den bestehenden Modellclient der Laufzeitskripte aufrufen. Andernfalls wären Modell, Temperatur und Schnittstellenversion zwischen den Varianten nur dokumentiert und nicht strukturell gleich.

Eine Voreinstellung des Rahmenwerks musste ich anpassen: Die Obergrenze für die Zahl ausgeführter Knoten liegt standardmäßig bei 25 und wäre bei fünf fachlichen Iterationen über neun Knoten erreicht worden. Der Lauf hätte dann mit einer Ausnahme des Rahmenwerks geendet, also mit einem Abbruchgrund, den die

[3] **Markdown-Dokumente:** Textbasierte Dateien mit einfacher Syntax zur Strukturierung und Formatierung von Inhalten.

prozeduralen Bedingungen nicht kennen und der sich später als Architektureffekt gelesen hätte. Die Grenze ist deshalb aus der Iterationsobergrenze und der Knotenzahl berechnet und liegt bei 55. Die fachliche Obergrenze bleibt allein die Iterationszahl, durchgesetzt vom Bewertungsknoten.

Der Ablauf ist als sequenzielle Kette mit einer Rückkante modelliert und nicht als Suchbaum mit parallelen Zweigen, wie etwa beim Tree-of-Thoughts-Ansatz[4]. Eine Baumsuche hätte die Zahl der Modellaufrufe gegenüber der prozeduralen Bedingung erhöht und damit einen zusätzlichen Einflussfaktor eingeführt. Dadurch wäre nicht mehr ausschließlich die Struktur der Orchestrierung verglichen worden, sondern zugleich auch ein erhöhter Rechenaufwand.

Der Schnitt der Knoten folgt einem einheitlichen Kriterium. Eine Grenze zwischen zwei Verarbeitungsschritten wird dort gezogen, wo ein eigener Fehlermodus beobachtbar wird. Die Zerlegung richtet sich damit nicht am Programmieraufwand aus, sondern daran, welche Fehlerarten sich anschließend voneinander unterscheiden lassen. Vier Knoten sind aus diesem Grund nicht zusammenlegbar, an ihnen wird jeweils eine der in Kapitel 6 definierten Fehlerkategorien beobachtbar.

4.6 Der GraphState

Der gemeinsame Zustand ist ein typisiertes Objekt mit 22 deklarierten Feldern, das die Kette durchläuft. Jeder Knoten liest, was vorherige Knoten hinterlassen haben, und schreibt genau sein eigenes Feld.

Der Zustand führt die Identität und die Metadaten des Laufs, den Fehlerstand vor und nach der Korrektur, je ein Ausgabefeld pro Knoten sowie eine Ablaufspur. Die Ablaufspur enthält für jeden ausgeführten Knoten den Namen, einen Zeitstempel, die Ausführungsdauer sowie je einen Eingangs- und Ausgangsprüfwert.

Zwei Feldschnitte wurden nachträglich gezogen, um Verwechslungen zwischen dem Zustand beim Einstieg und dem Zustand nach der Korrektur auszuschließen und um dem Prüfknoten die vollständige Antworthülle des Modells zugänglich zu machen. Anhang D dokumentiert beide.

Für die Nachvollziehbarkeit ist die Herkunft der Einträge ausschlaggebend. Sie werden vom Programmcode aufgezeichnet und nicht vom Modell berichtet.

[4] **Tree of Thoughts (ToT):** Ansatz, bei dem mehrere mögliche Denk- bzw. Lösungswege als Verzweigungen eines Baums erzeugt, bewertet und bei Bedarf weiterverfolgt oder verworfen werden.

Das Feld der Regelzuordnung hält fest, welche Karten die Ladefunktion geladen hat, das Feld der technischen Prüfung, was der Schemavalidator zurückgegeben hat, das Feld der Anwendung, was die erneute Validierung gemeldet hat.

Diese Unterscheidung ist wichtig, da Turpin et al. (2023) zeigen, dass von Sprachmodellen erzeugte Begründungen den tatsächlichen Entscheidungsprozess nicht zuverlässig widerspiegeln müssen. Eine Ablaufspur, die lediglich auf solchen Selbstauskünften des Modells basiert, wäre daher nur eingeschränkt aussagekräftig. Die hier verwendete Ablaufspur protokolliert dagegen tatsächlich ausgeführte Verarbeitungsschritte und Zustandsübergänge des Systems.

Der Unterschied zur prozeduralen Bedingung ist dabei nicht grundsätzlich, sondern liegt im Umfang der verfügbaren Nachvollziehbarkeit. Auch dort entstehen maschinell auswertbare Artefakte, wie in Abschnitt 3.4 beschrieben. Es fehlen jedoch die Zusammenführung zu einem gemeinsamen Zustand, die festgehaltene Reihenfolge der Verarbeitungsschritte und die Angabe der jeweils verwendeten Regelgrundlage.

Letztere erscheint dort lediglich in einer Bildschirmausgabe und geht mit dem Unterprozess verloren. Jacovi & Goldberg (2020) sprechen sich dafür aus, Nachvollziehbarkeit nicht als binäre Eigenschaft, sondern in unterschiedlichen Ausprägungen zu betrachten.

Dieser Auffassung folgt die vorliegende Arbeit, da sie auch dem beobachteten Unterschied zwischen den untersuchten Bedingungen entspricht.

4.7 Die neun Knoten

#	Knoten	Eingang	Ausgang	LLM -Model-laufruf
1	Eingabeanalyse	Snapshot-Kennung	Aufgabenbeschreibung	nein
2	Fehlerklassifikation	Validierungsmeldungen	ausgewählter Fehler, Suchstrategie, Kartenvorschlag	ja
3	Kontextsuche	Suchstrategie	Kontextausschnitt, verwendetes Vergleichskollektiv	nein
4	Regelzuordnung	Fehlertyp, Kartenvorschlag	geladene Karten, Regeltext, Prüfwert	nein
5	Korrekturerzeugung	Kontext, Regeltext	Korrekturvorschlag mit Begründung	ja
6	Technische Prüfung	Korrekturvorschlag	Schemastatus, Zahl der Wiederholungen	bedingt
7	Anwendung und erneute Validierung	geprüfter Vorschlag	Fehlerbilanz nach der Korrektur	nein
8	Ergebnisbewertung	Fehlerbilanz, Schemastatus	Entscheidung mit Begründung	nein
9	Ausgabe	Endzustand	einheitlich formatiertes Ergebnis	nein

Tabelle 5: Die neun Verarbeitungsknoten

Drei der neun Knoten rufen das Sprachmodell. Es sind dieselben drei Aufrufe, die auch die prozedurale Pipeline ausführt, nämlich Fehleranalyse, Korrekturerzeugung und Schemaprüfung. Die graphbasierte Variante ist damit nicht modellintensiver als die prozedurale, sie macht sichtbar, was zwischen denselben drei Aufrufen geschieht. Der Ausgabeknoten erzeugt das Endergebnis deterministisch aus dem Zustand. Ursprünglich war an dieser Stelle die modellgestützte Erzeugung eines Berichts vorgesehen. Da keine der prozeduralen Pipelines einen solchen Schritt enthält, hätte die graphbasierte Bedingung damit einen vierten Modellaufruf ausgeführt. Der Bericht bleibt als nachgelagerte Funktion des Produktivsystems erhalten und ist nicht Bestandteil der verglichenen Verarbeitung.

Die Aufteilung der Zuständigkeiten zwischen Klassifikation, Kontextsuche und Regelzuordnung folgt der in Abschnitt 4.3 beschriebenen Festlegung. Der Klassifikationsknoten führt die Suche nicht mehr selbst aus, sondern hinterlegt die Suchstrategie im Zustand. Der Suchknoten führt sie aus und hält zusätzlich fest, welches Vergleichskollektiv verwendet wurde. Diese Angabe betrifft unmittelbar den in Abschnitt 3.6 geschilderten Fall, in dem eine Begründung sich auf ein anderes Kollektiv berief als das tatsächlich verwendete.

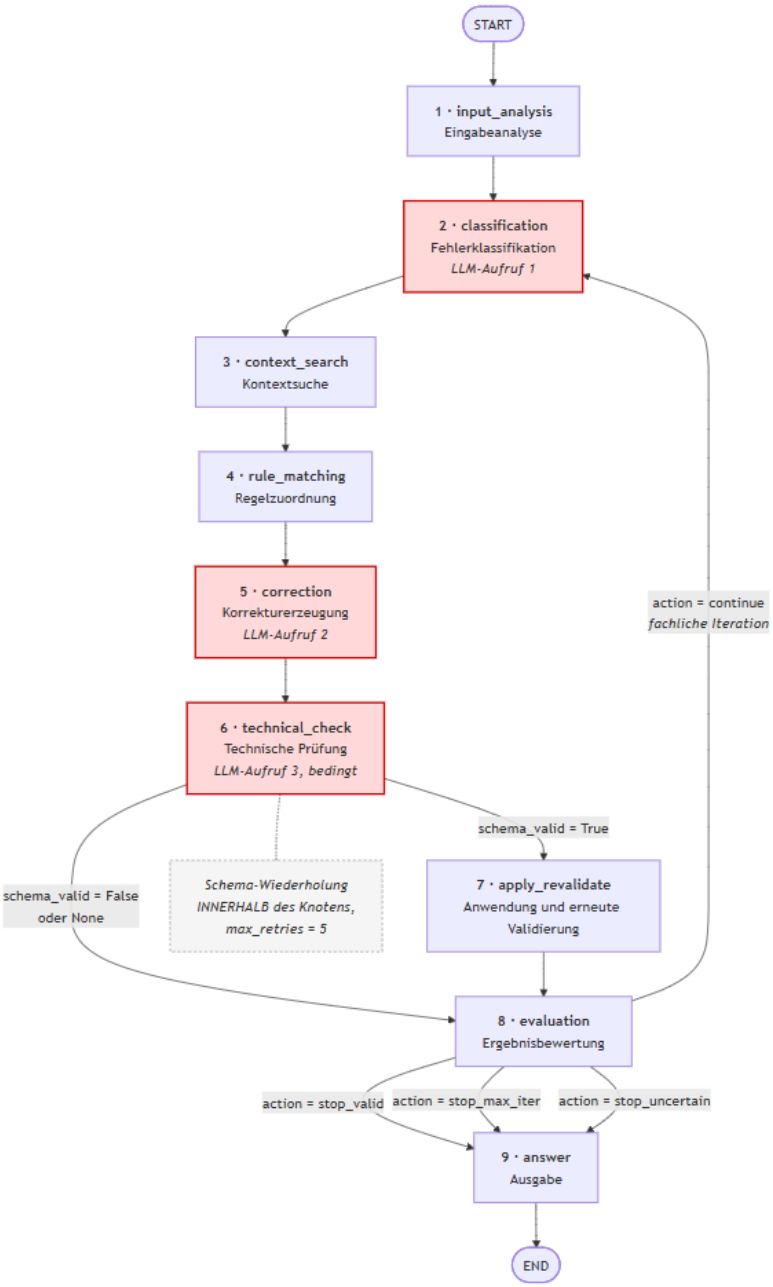


Abbildung 5: In der Untersuchung implementierter LangGraph-Verarbeitungsgraph mit neun Verarbeitungsknoten, Rückkante für weitere fachliche Iterationen und separaten Abschlusszuständen.

4.8 Kanten und Iterationslogik

Der Ablauf verläuft von der Eingabeanalyse über Klassifikation, Kontextsuche, Regelzuordnung und Korrekturerzeugung zur technischen Prüfung.

Nach der technischen Prüfung entscheidet allein der Schemastatus. Ist der Vorschlag gültig, folgt die Anwendung, andernfalls geht der Ablauf ohne Anwendung zur Ergebnisbewertung.

Nach der Ergebnisbewertung liest ein Verteiler ausschließlich das Feld, das der Bewertungsknoten geschrieben hat. Vier Ausgänge sind möglich. Bei fehlerfreiem Snapshot endet der Lauf regulär, andernfalls führt eine Rückkante bei verfügbarer Iteration zur Fehlerklassifikation mit aktualisiertem Validierungsstand. Bei erreichter Iterationsgrenze endet der Lauf mit dem Vermerk, dass eine manuelle Prüfung erforderlich ist. Liegt kein bewertbarer Vorschlag vor oder ist die Schemaprüfung endgültig gescheitert, endet der Lauf unter Ausweisung von Unsicherheit.

Der letzte Ausgang formalisiert ein im Betrieb beobachtetes Verhalten, denn eine ausgewiesene Unsicherheit ist einer erzwungenen, nicht belegbaren Korrektur vorzuziehen.

Die Rückkante entspricht dem Prinzip von Self-Refine[5] nach Madaan et al. (2023), bei dem ein erzeugtes Ergebnis auf Grundlage einer Rückmeldung iterativ überarbeitet wird. Im Unterschied zum ursprünglichen Self-Refine-Ansatz stammt diese Rückmeldung jedoch nicht aus einer Selbsteinschätzung des Modells, sondern aus der erneuten Validierung durch das Zielsystem.

Der Verteiler selbst enthält keine fachliche Logik. Er liest ein Feld, das ein sichtbarer Knoten samt Begründung geschrieben hat, sodass jede Verzweigung in der Ablaufspur nachlesbar ist. Ein Verteiler mit eingebauter Bedingungskette wäre wieder der implizite Kontrollfluss, dessen Auswirkungen die Arbeit untersucht.

Technische und fachliche Wiederholungen sind getrennt. Verstößt ein Vorschlag gegen das Ausgabeschema, wiederholt die Prüffunktion den Modellaufruf innerhalb des Knotens bis zu einer festgelegten Obergrenze. Eine zusätzliche Kante zurück zur Korrekturerzeugung existiert nicht. Sie wäre eine zweite Wiederholungsschicht über der bestehenden und würde die graphbasierte Bedingung in einer Dimension verändern, die nicht Gegenstand des Vergleichs ist.

Die Rückkante zur Klassifikation dient ausschließlich einer neuen fachlichen Iteration nach erneuter Validierung.

---

[5] **Self-Refine:** Ansatz, bei dem ein Sprachmodell seine eigene Ausgabe bewertet, daraus Feedback ableitet und die Antwort iterativ verbessert.

4.9 Anwendung, Upload und erneute Validierung

Der Knoten der Anwendung wendet den geprüften Vorschlag auf die Snapshot-Daten an, schreibt den veränderten Snapshot zurück, löst die Neuberechnung der Validierung aus, wartet deren Abschluss ab und ruft das Ergebnis ab.

Das Abwarten ist nicht nebensächlich. Wie in Abschnitt 3.2 beschrieben, rechnet die Validierung nach einer Änderung nicht selbsttätig neu, ein Abruf ohne vorherige Auslösung liefert deshalb den veralteten Stand. Da die nachfolgende Ergebnisbewertung auf diesem Wert beruht, hängt an ihm sowohl die Entscheidung über eine weitere Iteration als auch die spätere Messung neu entstandener Fehler.

Ein nicht belastbarer Zustand wird nicht als fehlerfrei behandelt. Liefert die Neuberechnung kein Ergebnis, führt die Bewertung auf die Ausweisung von Unsicherheit. Eine fehlende Angabe wird damit nicht als Unbedenklichkeit gelesen.

Für die spätere Auswertung reicht die bloße Fehlerzahl nicht aus, weil eine unveränderte Zahl sowohl bedeuten kann, dass nichts geschehen ist, als auch, dass ein Fehler behoben und ein anderer erzeugt wurde. Der Knoten leitet deshalb zusätzlich ab, welche Fehler behoben wurden, welche verblieben und welche neu entstanden sind. Die Identität eines Fehlers ist dabei eine Näherung aus Prüfkennung und Meldungsinhalt, da die Validierungs-Engine keine robuste Fehlerkennung liefert.

Diese Semantik muss für alle drei Bedingungen gelten und wurde deshalb an der gemeinsam genutzten Stelle hergestellt und nicht im Knoten. Die methodische Begründung und die Reichweite des Eingriffs behandelt Abschnitt 5.4.

4.10 Provenienz

Über die Ablaufspur hinaus hält der Zustand an zwei Stellen fest, worauf sich ein Schritt gestützt hat. Die Regelzuordnung hält fest, welche Karten geladen wurden und welchen Prüfwert der daraus entstandene Regeltext trägt. Die Kontextsuche hält fest, welcher Ausschnitt verwendet wurde und welches Vergleichskollektiv der Auswertung zugrunde lag.

Der geladene Regeltext wird dabei im Zustand weitergereicht und nicht im Korrekturknoten erneut geladen. Andernfalls könnte die Provenienzangabe einen anderen Text ausweisen als den, der dem Modell vorlag. Dass beide übereinstimmen, ist in der Ablaufspur über den Prüfwert nachvollziehbar.

Die Provenienzangaben beantworten eine andere Frage als die Ablaufspur. Die Ablaufspur zeigt, welchen Weg der Prozess genommen hat, die Provenienzangaben zeigen, worauf sich ein einzelner Schritt gestützt hat.

Für die zweite Frage genügt ein Feld im Zustand, eine graphförmige Modellierung des Regelwerks wäre dafür nicht erforderlich.


```
[
  { "node": "input_analysis",   "timestamp_utc": "13:29:12", "duration_ms": 0,
    "input_digest":  { "snapshot_id": "3b3f5177-7f1f-423b-aef7-297ddecd912a" },
    "output_digest": { "quelle": "snapshot-validation.json", "errors": 1, "warnings": 5,
                       "error_tags": ["validate_packaging_references"] } },

  { "node": "classification",   "timestamp_utc": "13:29:13", "duration_ms": 3582,
    "input_digest":  { "iteration": 1, "meldungen": 6 },
    "output_digest": { "tag": "PACKAGING_REFERENCES", "search_mode": "value",
                       "search_value": "70381", "should_investigate": true,
                       "relevant_cards_vorgeschlagen": ["packaging-references.md"],
                       "identify_response_sha256": "4f5537b9…" } },

  { "node": "context_search",   "timestamp_utc": "13:29:16", "duration_ms": 631,
    "output_digest": { "results_count": 12, "results_hash": "6d6b43bc…",
                       "lines_used": ["articles[121].standardPackaging", "articles[189].standardPackaging",
                                      "articles[338].standardPackaging", "packagingEquipmentCompatibility[0].pac
                                      "demands[301].packaging", "demands[302].packaging", "demands[314].packagin
                                      "demands[453].packaging", "demands[455].packaging", "demands[697].packagin
                                      "demands[728].packaging", "demands[1393].packaging"] } },

  { "node": "rule_matching",    "timestamp_utc": "13:29:17", "duration_ms": 43,
    "input_digest":  { "tag": "PACKAGING_REFERENCES", "extra_cards": ["packaging-references.md"] },
    "output_digest": { "rulebook_mode": "cards", "chars": 13564,
                       "cards_loaded": ["_core.md", "packaging-references.md"],
                       "rule_text_hash": "f44761b9…" } },

  { "node": "correction",       "timestamp_utc": "13:29:17", "duration_ms": 7371,
    "input_digest":  { "regeln_von_knoten4": true, "regeln_sha256": "f44761b9…",
                       "regeln_zeichen": 13564, "karten": ["_core.md", "packaging-references.md"],
                       "context_input_sha256": "6d6b43bc…", "context_handoff_ok": true,
                       "identify_input_sha256": "4f5537b9…", "identify_handoff_ok": true },
    "provenienz":    { "response_sha256": "f36f3ce0…" },
    "output_digest": { "action": "update_field",
                       "target_path": "packagingEquipmentCompatibility[0].predecessors",
                       "new_value": [], "value_source": "llm", "confidence_score": 0.475,
                       "blockiert": false } },

  { "node": "technical_check",  "timestamp_utc": "13:29:24", "duration_ms": 0,
    "input_digest":  { "hat_vorschlag": true, "response_sha256_eingang": "f36f3ce0…" },
    "provenienz":    { "proposal_sha256_before": "f36f3ce0…", "proposal_sha256_after": "f36f3ce0…",
                       "retry_hat_vorschlag_geaendert": false },
    "output_digest": { "schema_valid": true, "retries": 0, "fehleranzahl": 0 } },

  { "node": "apply_revalidate", "timestamp_utc": "13:29:24", "duration_ms": 12604,
    "input_digest":  { "proposal_sha256": "f36f3ce0…", "proposal_identisch": true,
                       "proposal_aus_state": true },
    "output_digest": { "applied_ok": true, "uploaded": true, "revalidation_ok": true,
                       "revalidation_job": "fc932acd", "revalidation_waited_s": 7.1,
                       "errors_before": 1, "errors_after": 1,
                       "errors_resolved": 0, "errors_remaining": 1, "errors_new": 0 } },

  { "node": "evaluation",       "timestamp_utc": "13:29:37", "duration_ms": 0,
    "input_digest":  { "errors_after": 1, "iteration": 1, "max_iterations": 5 },
    "output_digest": { "action": "continue",
                       "reasoning": "Noch 1 Fehler nach Iteration 1 (0 behoben, 0 neu); neue FACHLICHE Iteration

  { "node": "classification",   "timestamp_utc": "13:29:37", "duration_ms": 3192,
    "input_digest":  { "iteration": 2, "meldungen": 6 },
```


Abbildung 6: Gekürzter Auszug einer Ablaufspur aus einem Messlauf der Bedingung C, mit Knotenfolge und Herkunftsangabe der geladenen Regeln.

4.11 Was nicht verändert wurde

Nicht verändert wurden der Orchestrierungs-Agent, der RAG-Agent und der Chat-Agent, die Prompttexte der drei Modellaufrufe, die Kontextextraktion, die Schemaprüffunktion samt ihrer Wiederholungsgrenze, das Regelwerk in beiden Bereitstellungsformen, die Zuordnung von Fehlertypen zu Regelkarten sowie die Ablage der Laufzeitartefakte auf Dateiebene. Die graphbasierte Variante schreibt dieselben Dateien wie die prozedurale und legt ihren Zustand zusätzlich ab.

Diese Aufzählung grenzt ein, worauf ein beobachteter Unterschied überhaupt zurückgehen kann.

Eine Eigenschaft, die beide Varianten teilen, gehört dabei benannt. Beide erzwingen an mehreren Stellen eine strukturierte Ausgabe. Tam et al. (2024) berichten, dass Formatvorgaben die Leistung bei Reasoning-Aufgaben senken, bei Klassifikationsaufgaben mit strukturierter Ausgabe dagegen erhöhen können, die Wirkung also aufgabenabhängig ist. Die graphbasierte Variante erzwingt Struktur an mehr Übergängen als die prozedurale, während zugleich mehrere ihrer Knoten klassifikationsnah arbeiten. In welche Richtung sich das auswirkt, ist aus der Literatur nicht ableitbar und wird in Kapitel 8 aufgegriffen.

4.12 Nachgelagerte Konfigurierbarkeit

Nach Abschluss und Einfrieren der quantitativen Hauptmessung wurden zwei Erweiterungen ergänzt.

Die beiden Konfigurationsdimensionen wurden über die Weboberfläche zugänglich gemacht, sodass Regelbereitstellung und Ablaufsteuerung je Sitzung wählbar sind. Ergänzt wurde außerdem ein Schalter für die menschliche Freigabe, der keine Architekturdimension darstellt.

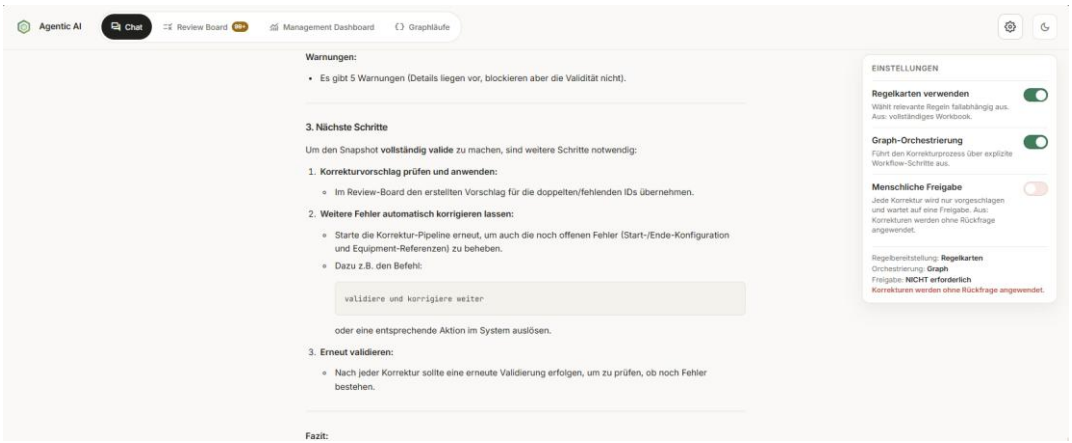


Abbildung 7: Konfigurationsansicht des Systems mit den zur Laufzeit einstellbaren Parametern.

Hinzu kam eine Ansicht, die einen abgelegten Graphzustand als Ablauf darstellt, mit den durchlaufenen Knoten, der Dauer je Knoten, den fachlichen Iterationen und den Ein- und Ausgangsangaben des jeweils gewählten Schritts. Sie liest die abgelegten Zustände und erzeugt keine. Für die prozeduralen Bedingungen existiert keine Entsprechung, dieselben Angaben wären dort aus mehreren Laufzeitartefakten zusammenzusetzen, die Regelgrundlage gar nicht. Wen et al. (2024) beschreiben eine vergleichbare Aufbereitung, allerdings für die Darstellung von Wissenspfaden und nicht von Verarbeitungsschritten.

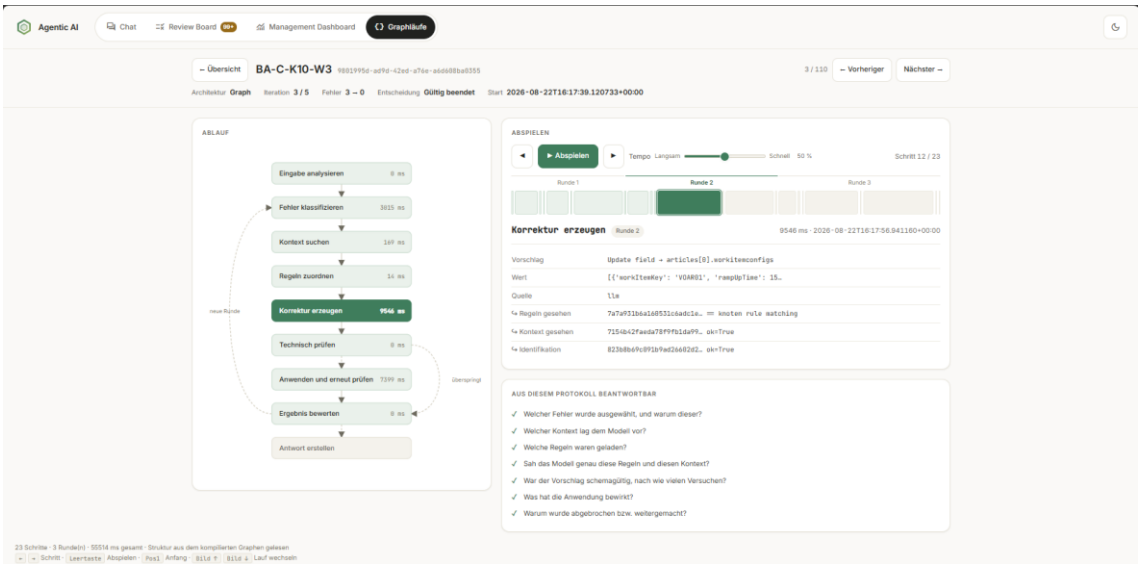


Abbildung 8: Darstellung des Graphzustands in der Benutzeroberfläche während eines laufenden Korrekturvorgangs.

Beide Erweiterungen sind für die Expertenerhebung in Kapitel 6 von Bedeutung, weil sie die Bedingungen vorführbar machen. Die Messläufe wurden nicht über die Oberfläche ausgeführt, sondern über eigenständige Ausführungsprozesse, die ihre Bedingung selbst setzen. Eine in der Oberfläche gewählte Einstellung kann diese Prozesse nicht beeinflussen. Über die Oberfläche erzeugte Ergebnisse sind zudem keine Messwerte, da im interaktiven Betrieb das episodische Gedächtnis und die Freigabepflicht auf ihren Produktivwerten stehen.

4.13 Zusammenfassung

Für die Untersuchung wurden drei Vergleichsbedingungen hergestellt, die sich paarweise in genau einer Eigenschaft unterscheiden. Die Trennung folgt daraus, dass die modulare Regelbereitstellung eine von der Ablaufsteuerung unabhängige Erweiterung ist und deshalb als eigene Bedingung geführt werden muss, damit der Hauptvergleich deutbar bleibt.

Die graphbasierte Variante überführt die bestehende Verarbeitungskette in neun benannte Knoten mit einem typisierten Zustand, einer aufgezeichneten Ablaufspur und Provenienzangaben zu Regelgrundlage und Kontext. Sie führt dieselben drei Modellaufrufe aus wie die prozedurale Variante und verwendet dieselben Funktionen, Prompts und Parameter. Verändert sind die Sichtbarkeit der Zwischenzustände und die Explizitheit der Übergänge, nicht der fachliche Verarbeitungsinhalt.

Zwei Asymmetrien bleiben bestehen und werden in Kapitel 8 als Einschränkung geführt. Die prozeduralen Bedingungen verfügen über eine Wiederholung auf Schrittebene, die der Graph nicht kennt, und die modulare Regelbereitstellung ist durch den Kartenvorschlag des Klassifikationsschritts nicht vollständig deterministisch.

5 Forschungsdesign und Methodik

5.1 Untersuchungsanlage

Die Untersuchung ist als Einzelfallstudie mit eingebettetem Experiment angelegt. Den Rahmen bildet ein produktiv genutztes System in einem realen industriellen Kontext, innerhalb dieses Rahmens wird ein kontrollierter Vergleich mehrerer Systemkonfigurationen durchgeführt.

Diese Anlage folgt der Argumentation von Runeson & Höst (2009), die für Fallstudien in der Softwaretechnik festhalten, dass der Erkenntniswert nicht aus der Verallgemeinerbarkeit einer Stichprobe entsteht, sondern aus der Tiefe der Beschreibung eines Falls und aus der Kombination mehrerer unabhängiger Datenquellen. Beides trifft hier zu. Der Fall ist ein einzelnes System, das über mehrere Monate entwickelt und betrieben wurde, und die Erhebung stützt sich auf eine automatisierte Messreihe und eine Expertenerhebung.

Der experimentelle Kern ist ein kontrolliertes Variationsdesign. Ausgangspunkt ist eine Systemvariante, aus der zwei Eigenschaften nacheinander verändert werden, sodass sich der Beitrag jeder Eigenschaft getrennt bestimmen lässt. Dabei wird jeweils genau ein Bestandteil verändert, während der übrige Aufbau konstant bleibt.

Für die Berichterstattung folgt die Arbeit den Vorgaben, die Baltes et al. (2026) für empirische Studien mit Sprachmodellen formulieren. Betroffen sind insbesondere die Angabe von Modellversion und Parametern, die Offenlegung der verwendeten Prompts und Laufzeitartefakte, das Berichten mehrerer Durchläufe statt eines einzelnen und die Validierung von Modellausgaben gegen menschliches Urteil. Wie die einzelnen Vorgaben umgesetzt sind, ist in den folgenden Abschnitten jeweils angegeben.

Für die Untersuchung wurde ein kontrolliertes Vergleichsdesign gewählt, da die zentralen Fragen nach Korrektheit und Stabilität sowohl eine Referenz als auch wiederholte Durchläufe voraussetzen. Ein ausschließlich qualitatives Vorgehen wäre dafür nicht ausreichend. Auf ein kontrolliertes Laborexperiment mit synthetischen Aufgaben wurde ebenfalls verzichtet, da dadurch ein wesentlicher Bestandteil des Untersuchungsgegenstands verloren ginge: die Arbeit mit realen Datenbeständen im tatsächlichen Zielsystem.

5.2 Vergleichsbedingungen und Vergleichslogik

Verglichen werden die drei in Kapitel 4 beschriebenen Bedingungen.

- Bedingung A verwendet das vollständige Regelwerk bei prozeduraler Ablaufsteuerung
- Bedingung B modulare Regelkarten bei prozeduraler Ablaufsteuerung
- Bedingung C modulare Regelkarten bei graphbasierter Ablaufsteuerung.

Benachbarte Bedingungen unterscheiden sich in genau einer Eigenschaft. Daraus ergeben sich drei Vergleiche mit unterschiedlicher Aussage.

Der Vergleich von A und B isoliert den Beitrag der Regelbereitstellung. Verändert wird ausschließlich, welcher Regeltext dem Modell vorliegt, der Ablauf bleibt identisch.

Der Vergleich von B und C isoliert den Beitrag der Ablaufsteuerung. Verändert wird ausschließlich, wie der Verarbeitungsprozess organisiert und dokumentiert wird, der Regeltext wird auf dieselbe Weise bereitgestellt.

Der Vergleich von A und C ist der Gesamtvergleich und entspricht dem in Titel und Forschungsfrage benannten Gegenüber von monolithischer Systemprompt-Struktur und graphbasierter Systemarchitektur.

Die Notwendigkeit der mittleren Bedingung ergibt sich aus einer Konfundierung, die in Abschnitt 4.1 beschrieben ist. Ohne sie wären in einem einzigen Vergleich zwei Eigenschaften gleichzeitig verändert worden, und ein beobachteter Unterschied wäre keiner von beiden zurechenbar gewesen.

Die vierte mögliche Kombination aus vollständigem Regelwerk und graphbasierter Ablaufsteuerung war technisch verfügbar, wurde jedoch nicht in die Messung einbezogen. Dadurch lässt sich eine mögliche Wechselwirkung zwischen den beiden untersuchten Eigenschaften nicht separat bestimmen. Der Vergleich von A und C bildet daher den gemeinsamen Effekt beider Veränderungen ab, ohne zwischen den jeweiligen Haupteffekten und einer möglichen Wechselwirkung zu unterscheiden.

5.3 Kontrollvariablen

Damit ein beobachteter Unterschied der jeweils variierten Eigenschaft zugerechnet werden kann, wurden alle übrigen Einflussgrößen über die drei Bedingungen hinweg konstant gehalten.

GRÖßE	WERT	HERSTELLUNG
SPRACHMODELL	Azure OpenAI GPT-4.1	gemeinsame Konfiguration
MODELLPARAMETER	Temperatur und Antwortlänge unverändert	gemeinsame Konfiguration
SCHNITTSTELLENVERSION	unverändert über alle Läufe	eingefrorene Umgebung
PROMPTTEXTE	identisch in allen Bedingungen	eine Implementierung, kein Nachbau
KONTEXTEXTRAKTION	identische Funktion	dieselbe Implementierung
SCHEMAPRÜFUNG	identische Funktion, gleiche Wiederholungsgrenze	dieselbe Implementierung
ITERATIONSOBERGRENZE	fünf fachliche Iterationen	gemeinsame Konfiguration
EPISODISCHES GEDÄCHTNIS	abgeschaltet	Umgebungsvariable je Messprozess
MENSCHLICHE FREIGABE	abgeschaltet	Umgebungsvariable je Messprozess
TESTFÄLLE	identischer Katalog	ein Katalog für alle Bedingungen
REFERENZWERTE	identisch	ein Bestand für alle Bedingungen
INFRASTRUKTUR	identische Umgebung	eingefrorener Stand
WIEDERHOLUNGEN	fünf je Fall und Bedingung	Ausführungsplan

Tabelle 6: Einflussgrößen für die Messung

Das episodische Gedächtnis speist frühere menschliche Korrekturentscheidungen als Belege in den Korrekturprompt ein und kann den Modellwert bei Objektgleichheit überschreiben. Im produktiven Betrieb ist das erwünscht, in der Messung hätte es die Bedingungen abhängig von der Reihenfolge der Läufe unterschiedlich beeinflusst. Es war deshalb in allen 255 Läufen abgeschaltet. Damit bilden die Messwerte nicht das Verhalten des Systems im vollen Produktivbetrieb ab, sondern das Verhalten der verglichenen Verarbeitungspfade.

Die menschliche Freigabe war ebenfalls in allen Läufen abgeschaltet, weil andernfalls kein Lauf ohne Eingriff zu Ende gelaufen wäre. Beide Abschaltungen gelten für alle drei Bedingungen gleichermaßen und begünstigen keine von ihnen.

Eine Asymmetrie bleibt bestehen und lässt sich nicht auflösen. Die prozedurale Steuerung wiederholt einen fehlgeschlagenen Pipelineschritt bis zu zweimal, die graphbasierte Variante kennt keine Knotenwiederholung. Diese Schicht greift bei technischen Fehlschlägen und nicht bei fachlich unzureichenden Ergebnissen. In keinem der 255 Läufe ist ein Abbruch verzeichnet, was ein Greifen unwahrscheinlich macht, ohne es auszuschließen. Die Asymmetrie wirkt zugunsten von A und B und wird in Kapitel 8 als Einschränkung geführt.

GRÖßE	WERT
DEPLOYMENT	gpt-4.1
MODELLVERSION	gpt-4.1-2025-04-14
SCHNITTSTELLENVERSION	2025-01-01-preview
TEMPERATUR	**0,3 in beiden gemessenen Modellaufrufen**
ANTWORTFORMAT	response_format={"type": "json_object"}
ANTWORTLÄNGE	**nicht gesetzt** – kein max_tokens; es gilt der Standard des Endpunkts

Tabelle 7: Konfiguration des verwendeten Sprachmodells und der Modellaufrufe

5.4 Herstellung der Vergleichbarkeit

Zwei Eigenschaften des bestehenden Systems hätten die Messung verfälscht und wurden vor der Erhebung an der gemeinsam genutzten Stelle behoben, also für alle drei Bedingungen zugleich.

Die Validierungs-Engine des Zielsystems berechnet nach einer Änderung nicht selbsttätig neu. Ein Abruf ohne vorherige Auslösung liefert den zuvor berechneten Stand, und ein korrigierter Snapshot hätte deshalb als fehlerfrei erscheinen können, ohne es zu sein. Da genau dieser Wert sowohl die Entscheidung über eine weitere Iteration als auch die spätere Auswertung trägt, wurde die Neuberechnung ausgelöst und ihr Abschluss abgewartet, bevor das Ergebnis übernommen wird. Bleibt die Neuberechnung ohne Ergebnis, wird der Zustand nicht als fehlerfrei behandelt, sondern als unbestimmt.

Die zweite Eigenschaft betrifft den Suchkontext. Innerhalb einer mehrfachen Korrektur wurde in einzelnen Fällen ein Kontextausschnitt weiterverwendet, der den Stand vor der letzten Änderung abbildete. Auch das wurde an der gemeinsamen Stelle behoben.

5.5 Testdesign

Der Testfallkatalog umfasst siebzehn Fälle. Jeder Fall entsteht durch kontrollierte Fehlerinjektion in einen fehlerfreien Ausgangssnapshot, wodurch für jeden injizierten Fehler der Sollwert bekannt ist. Die Konstruktion des Katalogs und die Herleitung der Referenzwerte beschreibt Kapitel 6.

Jeder Fall wurde in jeder der drei Bedingungen fünfmal ausgeführt. Daraus ergeben sich 255 Messläufe. (17 x 5 x 3).

Die fünf Wiederholungen dienen zwei Zwecken. Sie sind die Datengrundlage für die Entscheidungsstabilität, und sie verhindern, dass ein Einzelergebnis als Eigenschaft einer Bedingung gelesen wird. Baltes et al. (2026) führen das Berichten mehrerer Durchläufe ausdrücklich als Anforderung, weil ein einzelner Durchlauf bei stochastischen Modellen keine belastbare Aussage trägt.

Die Ausführungsreihenfolge wurde randomisiert. Damit wird ausgeschlossen, dass eine Bedingung systematisch von einem bestimmten Zustand der Umgebung profitiert, etwa von einer geringeren Auslastung des Modellendpunkts zu einer bestimmten Tageszeit. Die verwendete Reihenfolge ist über den Startwert des Zufallsgenerators reproduzierbar.

5.6 Ablauf eines Messlaufs

Ein Messlauf folgt einer festen Abfolge.

Zunächst wird eine frische Kopie des fehlerfreien Ausgangssnapshots erzeugt und in das Zielsystem eingespielt. Anschließend werden die für den jeweiligen Fall vorgesehenen Fehler injiziert, und der Sollzustand wird zusammen mit dem injizierten Zustand festgehalten. Es folgt eine initiale Validierung durch die Validierungs-Engine, deren Ergebnis den Ausgangsfehlerstand des Laufs bildet.

Danach wird die Korrekturverarbeitung in der für den Lauf vorgesehenen Bedingung ausgeführt. Der erzeugte Korrekturvorschlag wird geprüft, angewendet und zurückgeschrieben, die Neuberechnung der Validierung wird ausgelöst und abgewartet. Verbleiben Fehler und ist die Iterationsobergrenze nicht erreicht, beginnt eine weitere Iteration.

Am Ende werden sämtliche Artefakte des Laufs abgelegt, also der injizierte und der korrigierte Snapshot, die Validierungsstände vor und nach der Korrektur, die Korrekturvorschläge mit ihren Begründungen sowie die Laufmetadaten mit Bedingung, Wiederholungsnummer, Zeitstempeln und Konfigurationsschaltern. In Bedingung C kommt der abgelegte Ablaufzustand hinzu.

Die Auswertung erfolgte ausschließlich auf diesen abgelegten Artefakten und nicht während der Ausführung. Damit ist jede Kennzahl in Kapitel 7 bis zu ihrem Lauf und ihren Rohdaten zurückverfolgbar.

5.7 Reproduzierbarkeit

Vor Beginn der Erhebung wurde der Stand von Programmcode, Regelwerk, Testfallkatalog und Ausführungsumgebung festgeschrieben. Zwei Codestände sind dabei zu unterscheiden. Der Produktcode, also die Implementierung der drei Bedingungen, trägt die Kennung 15f2a44. Der Messstand, der zusätzlich den Ausführungsrunner umfasst, trägt die Kennung 2ab9546. Der Unterschied zwischen beiden betrifft ausschließlich die Art, in der der Runner seine Zwischenergebnisse ablegt. Schlüssel, Schema und Reihenfolge der abgelegten Daten sind unverändert, an der Verarbeitung selbst wurde nichts geändert.

Zum eingefrorenen Stand wurde ein Artefakt erzeugt, das für die messrelevanten Dateien jeweils Größe und Prüfsumme festhält.

Die Erhebung selbst fand nach dem Einfrieren statt. Änderungen am System, die danach vorgenommen wurden, insbesondere die in Abschnitt 4.12 beschriebenen Erweiterungen der Oberfläche, sind nicht Teil des Messstands und konnten die Ergebnisse nicht beeinflussen.

5.8 Präzisierung der Forschungsfragen

Forschungsfragen und Messdimensionen wurden im Verlauf der Untersuchung weiter konkretisiert. Um die methodische Nachvollziehbarkeit zu gewährleisten, werden Anlass und Zeitpunkt dieser Präzisierungen transparent dargestellt.

URSPRÜNGLICHE FASSUNG	FINALE FASSUNG	ANLASS
ZWEI VARIANTEN, GRAPH GEGEN MONOLITH	drei Messbedingungen bei zwei Architekturen	Konfundierung von Regelbereitstellung und Ablaufsteuerung
ROBUSTHEIT GEGENÜBER VARIIERENDEN EINGABEN	Entscheidungsstabilität bei wiederholter identischer Eingabe	keine kontrollierte Variationsreihe im Katalog
WARTBARKEIT ALS TEIL VON UF3	entfällt	kein Messinstrument entwickelt
HALLUZINATIONSRATE	fachliche Korrektheit und Halluzinationsverhalten getrennt	Ausgaben, die von der Referenz abweichen und dennoch belegt sind

Tabelle 8: Änderungen und Präzisierungen des Untersuchungsdesigns

Die ersten drei Präzisierungen betreffen den Aufbau der Untersuchung und wurden vor der Datenerhebung entschieden.

Die vierte wurde nach Vorliegen der Auswertung vorgenommen und ist deshalb gesondert einzuordnen. Verändert wurden ausschließlich Benennung und Gruppierung der Messgrößen. Die Erhebungsvorschriften standen zum Zeitpunkt des Einfrierens fest und wurden nicht angepasst, kein Messwert hängt an dieser Präzisierung. Anlass war ein Befund der Auswertung, nämlich dass in jeder Bedingung fünfzehn Instanzen auftraten, die vom Sollwert abwichen und zugleich durch die vorgelegten Daten gedeckt waren. Eine gemeinsame Kennzahl hätte diesen Unterschied verdeckt. Die Präzisierung ist damit selbst ein Ergebnis der Untersuchung und wird in Kapitel 9 als methodischer Beitrag geführt.

5.9 Nachträglich festgelegte Auswertungsregeln

Die Regeln für die Aggregation der Rohdaten, also für die Bildung von Kennzahlen aus den Einzelläufen, wurden nach Abschluss der Datenerhebung schriftlich festgelegt. Das betrifft unter anderem die Behandlung nicht bestimmbarer Werte, die Bezugsgröße für relative Angaben und den Umgang mit Läufen ohne vollständige Artefaktlage.

Diese Reihenfolge weicht vom ursprünglich festgelegten Vorgehen ab und wird daher transparent ausgewiesen. Das daraus entstehende Validitätsrisiko wird durch zwei Umstände begrenzt. Die Regeln wurden festgelegt, bevor die nach Bedingungen getrennten Korrektheitsergebnisse ausgewertet wurden. Zudem wurden sie schriftlich dokumentiert, bevor sie zur Anwendung kamen. Eine gezielte Anpassung der Regeln an bereits bekannte Ergebnisse kann damit ausgeschlossen werden. Eine mögliche unbewusste Beeinflussung durch die zuvor gesichteten Rohdaten lässt sich jedoch nicht vollständig ausschließen.

Die vollständigen Auswertungsregeln sind dokumentiert und liegen im Anhang E bei. Kapitel 6 beschreibt sie inhaltlich.

5.10 Expertenerhebung

Ergänzend zur automatisierten Messung wurde eine strukturierte Expertenbewertung durchgeführt. Sie deckt jene Teilfrage ab, die aus den Artefakten allein nicht beantwortbar ist, nämlich ob eine Fachperson mit der vorhandenen Ablaufinformation die Ursache eines fehlerhaften Ergebnisses tatsächlich lokalisieren kann. Baltes et al. (2026) führen die Validierung von Modellausgaben gegen menschliches Urteil als eigene Berichtspflicht, und Runeson & Höst (2009) begründen die Kombination unabhängiger Datenquellen als Qualitätsmerkmal einer Fallstudie.

Ein Teil der Fragen erhebt Konstrukte, die auch die automatisierte Messung erfasst. Diese Überschneidung ist beabsichtigt. Übereinstimmung zwischen zwei methodisch unabhängigen Zugängen stützt die entwickelte Messvorschrift, eine systematische Abweichung weist auf eine Grenze des gemessenen Konstrukts hin. Zwei Eigenschaften der Erhebung betreffen das Design und sind deshalb hier festzuhalten. Die Bewertung erfolgt nicht verblindet, weil die Teilnehmenden die Varianten selbst umstellen und sich die Varianten in der Oberfläche unterscheiden. Und die Nutzungsdauer war auf wenige Tage begrenzt, wodurch die Erhebung nur Aussagen über kurzzeitige, konzentrierte Nutzung trägt und keine über längerfristigen produktiven Einsatz. Instrument, Fragen, Teilnehmerkreis und Durchführung beschreibt Abschnitt 6.9.

Merkmal	Automatisierte Messung	Expertenerhebung
Datengrundlage	17 Testfälle mit kontrolliert injizierten Fehlern; je Fehler ein bekannter Sollwert. Ausgewertet wird auf den nach dem Lauf abgelegten Artefakten	vorbereitete Datenbestände im laufenden System, genutzt in mehreren Konfigurationen
Zugang	maschineller Abgleich des Ergebnisses gegen den Referenzwert, nach Abschluss der Ausführung	Selbstnutzung durch die Fachperson, anschließend leitfadengestütztes Interview
Erhobene Dimensionen	fachliche Korrektheit und Halluzinationsverhalten (UF1), Entscheidungsstabilität über Wiederholungen (UF2), Verfügbarkeit der Zustands- und Ablaufinformation (UF3, systemseitig)	wahrgenommene fachliche Plausibilität, Nachvollziehbarkeit einer Korrektur, menschliche Fehlerrekonstruktion und -lokalisierung (UF3), Bereitschaft zur Übernahme
Anzahl Beobachtungen	**255 Läufe** – 17 Fälle × 3 Bedingungen × 5 Wiederholungen, davon 85 je Bedingung	**2 Personen**, Nutzung über je etwa zwei Tage; 1 Transkript, 1 Gedächtnisnotiz
Verblindung	entfällt – kein menschliches Urteil im Messpfad. Stattdessen wurden die Aggregationsregeln schriftlich fixiert, bevor sie angewendet wurden	**nein** — die Teilnehmenden stellen die Varianten selbst um und wissen, welche sie beurteilen
Grenze	misst nur, was vorab operationalisiert wurde, zwei der vier Halluzinationskategorien tragen keinen bedingungsübergreifenden Vergleich, das episodische Gedächtnis war abgeschaltet, weshalb keine Aussage über den vollen Produktivbetrieb möglich ist	keine unabhängige Stichprobe (Projektbeteiligte), nur zwei Teilnehmende, die Expositionsdauer von zwei Tagen begrenzt die Aussage auf kurzzeitige Nutzung

Tabelle 9: Gegenüberstellung der automatisierten Messung und der Expertenerhebung

5.11 Validität und Grenzen des Designs

Die interne Validität wird durch die in Abschnitt 5.3 beschriebene Konstanthaltung und das kontrollierte Variationsdesign unterstützt. Eine Einschränkung ergibt sich aus der nicht erhobenen vierten Kombination sowie aus der Wiederholungsschicht der prozeduralen Bedingungen.

Die Konstruktvalidität wird dadurch gestützt, dass für jede Messdimension eine ausformulierte Messvorschrift vorliegt und die fachliche Korrektheit zusätzlich durch einen zweiten Zugang, den funktionalen Endzustand, betrachtet wird. Eine Einschränkung besteht darin, dass zwei der vier Halluzinationskategorien keinen bedingungsübergreifenden Vergleich ermöglichen.

Die externe Validität ist bewusst begrenzt. Untersucht werden ein konkretes System, ein Modell, ein Datenbestand und ein Regelwerk. Eine Übertragung der Ergebnisse auf andere Systeme lässt sich auf Grundlage dieser Untersuchung nicht begründen. Übertragbar ist daher weniger das konkrete Ergebnis als vielmehr das methodische Vorgehen.

Die Objektivität ist dadurch eingeschränkt, dass Entwicklung und Auswertung durch dieselbe Person erfolgten. Zur methodischen Absicherung wurden sämtliche Artefakte vollständig abgelegt, die Auswertungsregeln vor ihrer Anwendung schriftlich festgelegt und spätere Präzisierungen transparent dokumentiert, wie in Abschnitt 5.8 beschrieben. Eine mögliche Beeinflussung der Auswertung kann dadurch reduziert, jedoch nicht vollständig ausgeschlossen werden.

Eine weitere Einschränkung betrifft die zeitliche Reichweite der Untersuchung. Die automatisierte Messung bildet 255 Einzelläufe ab, die innerhalb weniger Stunden durchgeführt wurden und in denen das episodische Gedächtnis deaktiviert war. Die ergänzende Erhebung mit den Teilnehmenden fand innerhalb eines Zeitraums von etwa zwei Tagen statt, wobei die aktive Nutzung nach Angaben der Teilnehmenden jeweils mehrere Stunden umfasste. Beide Zugänge erlauben daher keine Aussage darüber, wie sich das System bei längerfristiger Nutzung im Arbeitskontext verhält. Wiederkehrende Korrekturmuster, die Wirkung des episodischen Gedächtnisses und die Entwicklung von Vertrauen in das System über einen längeren Zeitraum werden durch das gewählte Untersuchungsdesign nicht erfasst. Diese zeitliche Begrenzung bestimmt entsprechend die Reichweite der aus den Ergebnissen ableitbaren Aussagen.

5.12 Zusammenfassung

Die Untersuchung ist eine Fallstudie mit eingebettetem Ablationsexperiment. Drei Bedingungen, die sich paarweise in genau einer Eigenschaft unterscheiden, wurden über siebzehn Testfälle bei je fünf Wiederholungen ausgeführt, woraus 255 Messläufe entstanden. Modell, Parameter, Prompts, Testfälle und Referenzwerte waren über alle Bedingungen konstant, der Codestand wurde vor der Erhebung festgeschrieben, und sämtliche Auswertungen erfolgten auf abgelegten Artefakten.

Vier Punkte begrenzen die Aussagekraft und sind offengelegt. Die vierte Kombination wurde nicht erhoben, eine Wiederholungsschicht existiert nur in zwei der drei Bedingungen, die Aggregationsregeln wurden nach der Erhebung festgelegt, und eine der vier Präzisierungen der Fragestellung erfolgte nach der Auswertung.

Hinzu kommt eine zeitliche Grenze, die beide Erhebungszugänge betrifft und in Abschnitt 5.11 beschrieben ist.

6 Operationalisierung und Messinstrumente

6.1 Aufbau der Bewertung

Kapitel 5 beschreibt, wie die Daten erhoben wurden. Dieses Kapitel beschreibt, wie aus ihnen Kennzahlen entstehen.

Die Bewertung erfolgt auf drei Ebenen. Die einzelne erwartete Änderung trägt die Korrektheitsbewertung, der einzelne Lauf fasst die zugehörigen Änderungen zusammen und ergänzt sie um den Zustand des Datenbestands nach der Korrektur, und der Fall betrachtet die fünf Wiederholungen nebeneinander und trägt damit die Entscheidungsstabilität. Eine vierte Ebene liegt außerhalb der automatisierten Auswertung und ist Gegenstand von Abschnitt 6.9.

6.2 Testfallkatalog und Referenzwerte

6.2.1 Konstruktion der Fälle

Jeder Fall entsteht durch kontrollierte Fehlerinjektion in einen fehlerfreien Referenzsnapshot. Ein Generator verändert dabei gezielt einzelne Werte und protokolliert für jede Veränderung den Pfad, den Wert vor der Injektion und den Wert danach. Damit ist der Sollwert konstruktionsbedingt bekannt, weil er dem Zustand vor der Veränderung entspricht. Das Verfahren entspricht dem Grundgedanken der Mutationsanalyse, bei der bekannte Veränderungen in ein korrektes Artefakt eingebracht werden, um die Erkennungsleistung eines Verfahrens zu bestimmen. Die gewählten Fehlerarten entsprechen den im Projektbetrieb tatsächlich auftretenden Mustern, unter anderem doppelten Kennungen, leeren Pflichtfeldern, ungültigen Referenzen, inkonsistenten Beziehungen zwischen Objekten und Wertebereichsverletzungen.

Testfallkatalog Ein vollständiger Auszug findet sich im Anhang F.

6.2.2 Umfang der Referenz

Über die siebzehn Fälle verteilen sich 29 erwartete Änderungen. Zehn Standardfälle tragen elf Änderungen, sieben Komplexfälle tragen achtzehn. Bei fünf Wiederholungen je Bedingung ergeben sich daraus 145 zu bewertende Änderungsinstanzen je Bedingung.

Die Analyseeinheit für Aussagen über Fehlerarten und Systemverhalten bleibt jedoch die Zahl der Fälle, also siebzehn. Die Wiederholungen stellen keine zusätzlichen Fälle dar. Eine Einbeziehung der Wiederholungen in die Fallzahl würde die Aussagekraft der Untersuchung daher um das Fünffache überschätzen. Aus demselben Grund werden die Ergebnisse in absoluten Zahlen und nicht in Prozentwerten berichtet, da Prozentangaben bei lediglich siebzehn Fällen eine Genauigkeit suggerieren würden, die durch die Datengrundlage nicht gegeben ist.

6.3 Fachliche Korrektheit

6.3.1 Dreistufige Bewertung

Die Bewertung einer erzeugten Korrektur erfolgt in drei strikt getrennten Stufen. Die Trennung ist der methodische Kern dieses Kapitels, weil sie zwei Eigenschaften auseinanderhält, die in einer gemeinsamen Kennzahl verschmelzen würden.

Stufe eins, Pfadzuordnung. Eine erzeugte Korrektur wird einer erwarteten Änderung genau dann zugeordnet, wenn beide Pfadangaben auf dasselbe Element im Datenbestand auflösen. Als Auflösungsgrundlage dient die eingefrorene Katalogdatei des Falls und nicht der jeweils aktuelle Zwischenstand eines Laufs. Das ist notwendig, weil die Grundlage für A, B und C identisch sein muss und unabhängig davon, wie weit ein Lauf gekommen ist. Löst ein Pfad mehrdeutig oder gar nicht auf, gilt die Zuordnung als nicht bestimmbar. Eine ungefähre Zuordnung gibt es nicht. Alle 29 Referenzpfade lösen deterministisch auf.

Stufe zwei, Übereinstimmung mit dem Referenzwert. Erst nach erfolgreicher Zuordnung wird der vorgeschlagene Wert mit dem erwarteten verglichen, und zwar ohne jede Berücksichtigung dessen, worauf sich der Vorschlag stützt. Stimmen beide überein, gilt die Änderung als korrekt, andernfalls als falsch. Liegt kein bewertbarer Wert vor, gilt sie als nicht bestimmbar.

Stufe drei, Belegbarkeit. Unabhängig davon wird geprüft, ob der vorgeschlagene Wert durch die Daten gedeckt ist, die dem Modell vorlagen. Diese Stufe beschreibt Abschnitt 6.5.

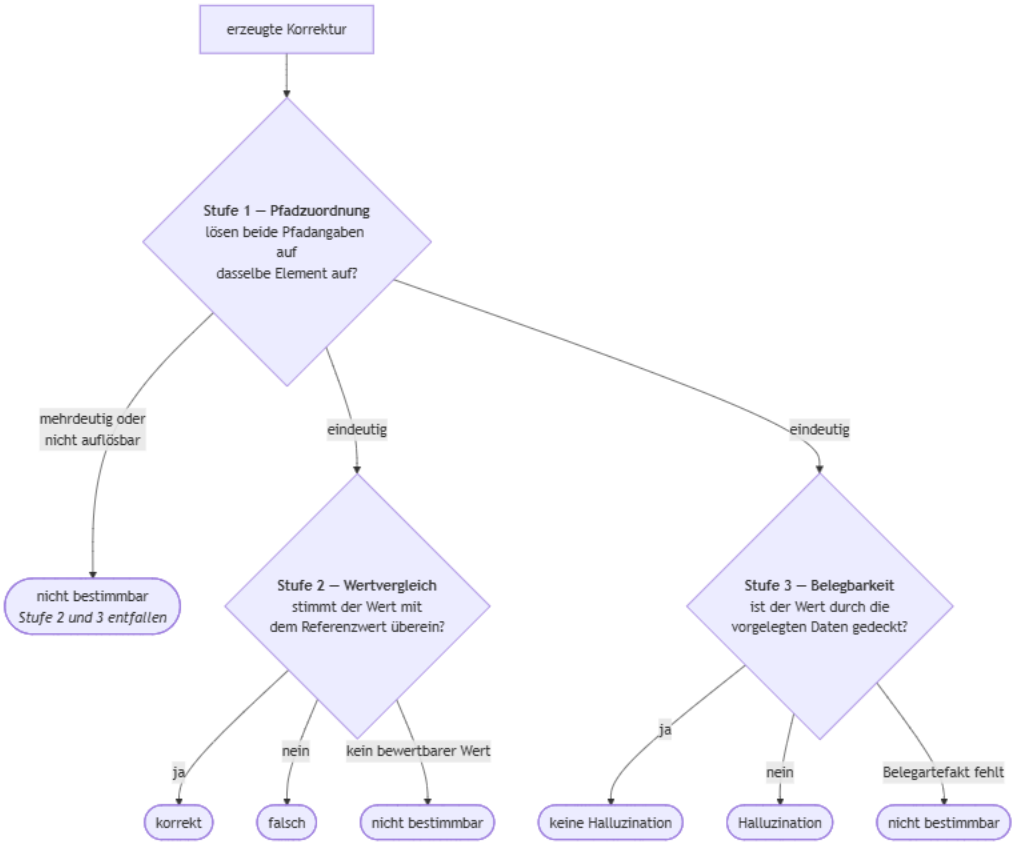


Abbildung 9: Die drei Bewertungsstufen der fachlichen Korrektheit als Ablauf, mit den Ausgängen je Stufe und den Bedingungen für den Übergang zur nächsten.

Warum die Trennung notwendig ist

Die ursprüngliche Fassung des Auswertungsprotokolls las das Ergebnis der Belegbarkeitsprüfung als Korrektheitsurteil. Das ist falsch, und der Fehler wurde vor Einsicht in die nach Bedingungen aufgeschlüsselten Ergebnisse korrigiert.

Die Belegbarkeitsprüfung liefert dieselbe Antwort für zwei sehr verschiedene Fälle, nämlich für einen Wert, der dem Referenzwert entspricht, und für einen Wert, der vom Referenzwert abweicht, aber durch die vorgelegten Daten gestützt wird. Hätte man diese Antwort als Korrektheitsurteil verwendet, wäre ein abweichender Wert als korrekt gezählt worden.

Die folgende Wahrheitstabelle macht die beiden Größen und ihr Verhältnis vollständig sichtbar. Sie wurde vor der Anwendung auf die Messdaten synthetisch geprüft.

Fall	*Übereinstimmung*	*Halluzination*
Referenzwert getroffen, Belege vorhanden	korrekt	nein
Referenzwert verfehlt, Belege stützen den Wert	falsch	nein
Referenzwert verfehlt, Belege stützen nicht	falsch	ja
Referenzwert verfehlt, Belegartefakt fehlt	falsch	nicht bestimmbar
Referenzwert getroffen, Belegartefakt fehlt	korrekt	nicht bestimmbar
kein Wert vorgeschlagen	nicht bestimmbar	nicht bestimmbar
Pfad nicht auflösbar	nicht bestimmbar	nicht bestimmbar

Tabelle 10: Wahrheitstabelle der beiden Größen

Die zweite Zeile ist der Kern. Ein Vorschlag kann vom Referenzwert abweichen und dennoch durch die vorgelegten Daten gedeckt sein. Das ist keine Halluzination im Sinne der in Abschnitt 2.6 dargestellten Definitionen, sondern ein Befund über die Aufbereitung der Belege. Die Verallgemeinerung, dass eine falsche Ausgabe automatisch eine halluzinierte sei, ist damit unzulässig.

Fehlt das Belegartefakt eines Laufs, bleibt das Korrektheitsurteil davon unberührt. Nur die Belegbarkeit ist dann nicht prüfbar. Beide Größen überschreiben einander in keiner Richtung.

6.3.2 Bewertungsumfang der Referenz

Die Referenz führt je Änderung nur Pfad, Vorzustand und Nachzustand. Sie enthält keine Angabe darüber, mit welcher Operation eine Korrektur durchzuführen wäre. Für mehrere Fälle sind zwei verschiedene Operationen gleichermaßen geeignet, etwa das Setzen eines Listenwerts und das Hinzufügen eines Listenelements. Die gewählte Operation wird deshalb nur beschreibend berichtet und nicht als Korrektheitskriterium geführt. Eine erwartete Operation wird nicht nachträglich aus Modellausgaben rekonstruiert.

6.3.3 Kennzahl auf Laufebene

Je Lauf werden sechs Zahlen erfasst, nämlich die Anzahl erwarteter Änderungen, die Anzahl korrekt erfüllter, die Anzahl falsch adressierter, die Anzahl nicht bestimmbarer, die Anzahl nicht adressierter sowie die Anzahl zusätzlicher Korrekturen, die keiner erwarteten Änderung zuzuordnen sind. Alle sechs beziehen sich auf die Übereinstimmung mit dem Referenzwert und nicht auf die Belegbarkeit.

Ergänzend wird ein einzelner Indikator gebildet. Er ist erfüllt, wenn alle erwarteten Änderungen eines Laufs auf dem richtigen Pfad mit dem korrekten Wert erfüllt wurden und keine zusätzliche fachliche Änderung vorgenommen wurde.

Zwei Einschränkungen gehören zu diesem Indikator. Er misst ausschließlich die Übereinstimmung mit der Referenz, nicht die Abwesenheit von Halluzinationen. Ein abweichender, aber belegter Vorschlag erfüllt ihn nie, kann in der Halluzinationsdimension gleichwohl unauffällig sein. Und er ist eine nach der Erhebung festgelegte Aggregationsregel und keine vor dem Einfrieren definierte Einzelwertkategorie. Beides wird bei der Ergebnisdarstellung kenntlich gemacht.

6.4 Funktionaler Endzustand

Der Indikator aus Abschnitt 6.3 bewertet den Weg, also ob die vorgesehene Stelle mit dem vorgesehenen Wert getroffen wurde. Ob der Datenbestand danach aus Sicht des Zielsystems fehlerfrei ist, bewertet er nicht. Beides kann auseinanderfallen. Eine Korrektur kann von der Referenz abweichen und den Fehler dennoch beheben, etwa wenn ein anderer, ebenfalls gültiger Wert gesetzt wird. Umgekehrt kann eine referenzgetreue Korrektur einen neuen Fehler auslösen.

Der Fehlerstand nach erfolgreicher Neuberechnung der Validierung wird deshalb als ergänzende Betrachtung ausgewertet, allerdings als nachträgliche Sensitivitätsanalyse und nicht als Primärkennzahl. Sie prüft, ob der Befund der Primärbewertung von der gewählten Bewertungsperspektive abhängt. Ein fehlerfreier Endzustand allein belegt keine fachliche Korrektheit, weil ein Wert so gesetzt werden kann, dass die Prüfregel nicht mehr greift, ohne dass der zugrunde liegende Sachverhalt richtig abgebildet wäre. Dieser Fall wurde bereits in der Pilotphase beobachtet und war Anlass für die getrennte Führung beider Perspektiven.

6.5 Halluzinationsverhalten

6.5.1 Vier Kategorien

Halluzinationsverhalten bezeichnet das Auftreten von Korrekturen oder Begründungen, die durch die vorgelegten Daten und Regeln nicht gedeckt sind.

Unterschieden werden vier Kategorien, die vor Beginn der Erhebung definiert und im Programmcode festgeschrieben wurden.

#	*Kategorie*	*Erhebung*	*Beobachtungspunkt*
1	fachliche Halluzination, ein Wert ohne Deckung durch die vorgelegten Daten	automatisch, über den Abgleich des Werts mit den im Suchkontext vorhandenen Belegen	Korrekturerzeugung
2	strukturelle Halluzination, ein Verstoß gegen das Ausgabeschema	automatisch, über das Ergebnis der Schemaprüfung	technische Prüfung
3	Regelhalluzination, die Berufung auf eine nicht vorhandene oder falsch wiedergegebene Regel	Abgleich der Behauptung mit dem tatsächlich geladenen Regeltext	Regelzuordnung und Korrekturerzeugung gemeinsam
4	Folgefehlererzeugung, die Korrektur erzeugt einen neuen Fehler	automatisch, über die erneute Validierung	Anwendung

Tabelle 11: Die vier Halluzinationskategorien

Kategorie eins unterscheidet über den Abgleich mit dem extrahierten Kontext einen abweichenden, aber gestützten Wert von einem frei erfundenen. Kategorie drei setzt voraus, dass sowohl die geladenen Regeln als auch die in der Begründung beanspruchten bekannt sind.

6.5.2 Erhebbarkeit der Kategorien

Die vier Kategorien sind nicht gleichermaßen erhebbar, und das ist der Grund, warum der Begriff Halluzinationsrate nicht trägt.

Kategorie eins und Kategorie vier sind über alle drei Bedingungen hinweg erhebbar und tragen den Vergleich.

Kategorie zwei ist nur innerhalb der graphbasierten Bedingung erhebbar. Die Schemaprüfung der prozeduralen Pipeline legt ihr Ergebnis nicht ab, es existiert ausschließlich zur Laufzeit. In der graphbasierten Variante wird es als eigenes Feld im Zustand geführt. Ein bedingungsübergreifender Vergleich ist damit nicht möglich.

Kategorie drei wurde nicht automatisiert erhoben. Die Prüfung setzt voraus, dass die in einer Begründung genannte Regelreferenz maschinell mit dem geladenen Regeltext abgeglichen wird. Ein solches Verfahren wurde nicht implementiert. Die Kategorie bleibt definiert, trägt aber keine Messwerte.

Für Kategorie eins gilt eine weitere Einschränkung. Fehlt das Artefakt mit den Suchergebnissen eines Laufs, lässt sich eine leere Belegmenge nicht von einer fehlenden unterscheiden. In diesem Fall gilt die Kategorie als nicht bestimmbar, mit ausgewiesenem Grund.

Aus diesen Gründen wird jede Angabe zum Halluzinationsverhalten ausschließlich mit ausgewiesenem Nenner berichtet. Ein Quotient über alle vier Kategorien wäre ein Verhältnis mit unbestimmter Bezugsgröße.

Der Umstand, dass zwei von vier Kategorien in den prozeduralen Bedingungen nicht oder nur eingeschränkt erhebbar sind, ist nicht nur eine Messgrenze. Er ist zugleich ein Befund zur Nachvollziehbarkeit und wird in Abschnitt 6.8 wieder aufgegriffen.

6.6 Folgefehler

Folgefehler werden als eigene Dimension geführt und nicht mit der Korrektheit verrechnet.

Erhoben werden je Lauf vier Größen, nämlich die behobenen Fehler, die verbliebenen, die neu entstandenen sowie die Arten der neu entstandenen Fehler. Grundlage ist der Vergleich des Fehlerstands vor der Korrektur mit dem Stand nach erfolgreicher Neuberechnung der Validierung.

Die bloße Fehlerzahl reicht dafür nicht aus. Eine unveränderte Zahl kann bedeuten, dass nichts geschehen ist, oder dass ein Fehler behoben und ein anderer erzeugt wurde.

Die Identität eines Fehlers wird aus der Prüfkennung und dem Inhalt der Meldung gebildet, weil die Validierungs-Engine keine stabile Fehlerkennung vergibt. Das ist eine Näherung und wird als solche berichtet.

Drei Festlegungen gehören dazu. Ein referenzgetreuer Vorschlag kann einen Folgefehler erzeugen. Ein fehlerfreier Endzustand belegt keine Übereinstimmung mit der Referenz. Und ein neu entstandener Fehler ändert die Bewertung des ursprünglichen Vorschlags nicht rückwirkend.

Die Berechnung erfolgt für alle drei Bedingungen über dieselbe gemeinsam genutzte Funktion.

Der in der graphbasierten Bedingung zusätzlich abgelegte Zustand dient ausschließlich der Gegenprobe und nicht als eigene Quelle, weil sonst eine Bedingung nach einem anderen Verfahren bewertet würde als die anderen beiden.

6.7 Entscheidungsstabilität

Entscheidungsstabilität bezeichnet die Übereinstimmung der fachlichen Entscheidung bei wiederholter Ausführung desselben Falls mit identischer Eingabe.

Gemessen wird die inhaltliche Entscheidung und nicht die sprachliche Formulierung. Eine gewisse Variabilität in der Wortwahl ist bei Sprachmodellen systemimmanent und für die Fragestellung ohne Belang. Erheblich ist, ob derselbe Fehler zu unterschiedlichen fachlichen Korrekturen führt. Die theoretische Begründung dafür, dass Streuung über Wiederholungen ein eigenständiges Messobjekt ist, gibt Abschnitt 2.7.

Erhoben werden je Fall und Bedingung, also über 51 Zellen, die fünf Wiederholungen einzeln. Berichtet werden für jede Zelle die Zahl der Wiederholungen mit erfülltem Korrektheitsindikator, die Zahl verschiedener fachlicher Ergebnisse und die aufgetretenen Abbruchzustände.

Die fünf Wiederholungen werden ausdrücklich nicht zu einem abschließenden Ergebnis je Fall verdichtet. Eine Mehrheitsregel würde eine Schwelle einführen, die nachträglich gesetzt wäre und die beobachtete Instabilität gerade wegmitteln würde. Stattdessen werden drei schwellenfreie Klassen gebildet, nämlich Fälle mit einheitlichem Ergebnis in allen fünf Wiederholungen, Fälle mit einheitlichem Ausbleiben und Fälle mit uneinheitlichem Ergebnis. Die Spannweite wird je Fall angegeben. Eingeschlossen in die Definition ist die Fähigkeit, Unsicherheit auszuweisen, statt eine nicht belegbare Korrektur zu erzwingen. Ein Lauf, der eine manuelle Prüfung verlangt, gilt deshalb nicht als Fehlschlag, sondern wird als eigener Zustand geführt.

Die zugehörigen erwarteten Änderungen gelten als nicht adressiert und ihr Wertsurteil als nicht bestimmbar. Bereits zuvor im selben Lauf korrekt durchgeführte Änderungen bleiben davon unberührt.

Nicht erhoben wurde die Reaktion auf strukturell abweichende oder gestörte Eingaben. Der Katalog enthält keine kontrollierte Variationsreihe desselben Falls. Die Unterscheidung zwischen Wiederholungsstabilität und Störungsrobustheit ist damit einseitig aufgelöst, und die Arbeit trifft nur zur ersten eine Aussage.

Fall	Bedingung	erfüllte Wiederholungen	verschiedene Ergebnisse	Zustände	Klasse
I01	A	5 / 5	1	5× fehlerfrei	einheitlich erfüllt
K08	C	0 / 5	1	5× fehlerfrei	einheitlich ausgeblieben
I08	B	3 / 5	2	5× fehlerfrei	uneinheitlich
K07	C	0 / 5	3	5× unsicher	uneinheitlich, 4× Ablaufzustand verloren

Tabelle 12: Erhebungsschema der Entscheidungsstabilität

6.8 Technische Nachvollziehbarkeit

6.8.1 Messvorschrift

Nachvollziehbarkeit wird nicht darüber gemessen, ob eine Bedingung eine Ablaufspur besitzt. Das wäre durch die Konstruktion vorweggenommen und zugleich sachlich falsch, weil auch die prozeduralen Bedingungen maschinell auswertbare Artefakte erzeugen, wie Abschnitt 3.4 beschreibt. Sie liegen nur verstreut vor, ohne gemeinsames Schema, ohne festgehaltene Reihenfolge und ohne Angabe der verwendeten Regelgrundlage.

Der Unterschied ist damit graduell und wird entsprechend erhoben. Jacovi & Goldberg (2020) begründen diese Auffassung, indem sie die Treue einer Erklärung als abstufbare und nicht als binäre Eigenschaft fassen.

Gemessen werden 3 Größen:

Rekonstruierbarkeit. Für sieben festgelegte Fragen wird je Bedingung bestimmt, ob die zur Beantwortung nötigen Angaben aus den abgelegten Artefakten belegbar sind, ob sie nur vermutet werden können oder ob sie fehlen.

Suchaufwand. Gezählt wird, wie viele getrennte Artefakte herangezogen werden müssen, um diese Fragen zu beantworten. Die Größe ist beschreibend und ohne Bewertung.

Verfügbarkeit der Artefakte. Erfasst wird, in welchen Läufen die für eine Rekonstruktion nötigen Artefakte vollständig vorliegen und in welchen nicht.

Nachvollziehbare Information	***A und B***	***C***
bearbeiteter Fehler	belegbar	belegbar
Grund der Priorisierung	nur aus Modellausgabe	nur aus Modellausgabe, aber dem Lauf zuordenbar
verwendeter Suchkontext	belegbar	belegbar
verwendete Regeln	nur indirekt rekonstruierbar	explizit belegbar
Übergabe von Regeln und Daten	nur indirekt rekonstruierbar	explizit belegbar
Ergebnis der Schemaprüfung	überwiegend belegbar	explizit belegbar
Anzahl technischer Wiederholungen	nicht belegbar	belegbar
Zustand vor/nach Anwendung	über mehrere Artefakte rekonstruierbar	explizit belegbar
Grund für Fortsetzung oder Abbruch	nur indirekt rekonstruierbar	explizit belegbar

Tabelle 13: Die sieben Rekonstruktionsfragen

6.8.2 Herkunft der Angaben

Für die Bewertung ist entscheidend, woher eine Angabe stammt. Turpin et al. (2023) zeigen, dass eine vom Modell erzeugte Begründung den tatsächlichen Entscheidungsweg systematisch verfehlen kann, ohne dass dies an der Begründung erkennbar wäre.

Als Beleg für Nachvollziehbarkeit gelten deshalb ausschließlich Angaben, die vom ausführenden Programm aufgezeichnet wurden. Die Begründung, die das Modell einem Korrekturvorschlag beifügt, ist eine Systemausgabe und wird als solche berichtet, sie zählt aber nicht als Nachweis des Entscheidungswegs.

6.8.3 Grenzen

Die Erhebung bestimmt, ob die zur Rekonstruktion nötigen Angaben vorhanden sind. Sie bestimmt nicht, ob ein Mensch mit ihnen zu einem Urteil kommt. Diese zweite Frage ist Gegenstand der Erhebung in Abschnitt 6.9.

Auf eine manuelle Bewertung, welcher Verarbeitungsschritt für ein fehlerhaftes Ergebnis ursächlich war, wurde verzichtet. Da Entwicklung und Auswertung der graphbasierten Variante durch dieselbe Person erfolgten, hätte eine solche Zuordnung das Risiko einer konstruktionsbedingten Verzerrung erhöht.

In einzelnen Läufen der graphbasierten Bedingung liegt der Ablaufzustand nicht vor, in weiteren Läufen aller Bedingungen wurde kein Korrekturvorschlag erzeugt. Beide Gruppen bleiben im Datensatz. Ihre vorhandenen Vorschläge werden nach den Regeln der Abschnitte 6.3 und 6.5 bewertet, ein vollständiger Abschluss der Verarbeitung wird für sie nicht behauptet, und es werden keine Werte ergänzt. Läufe ohne Ablaufzustand tragen ein eigenes Kennzeichen. Die Zahlen berichtet Kapitel 7.

6.9 Expertenerhebung

6.9.1 Begründung des Verfahrens

Drei der vier Messdimensionen lassen sich vollständig automatisiert erheben, die vierte nicht. Ob eine Fachperson mit der vorhandenen Ablauf- und Herkunftsinformation die Ursache eines fehlerhaften Ergebnisses lokalisieren kann, ist keine Eigenschaft der Artefakte, sondern eine Eigenschaft ihres Zusammenspiels mit einem Menschen, der über Domänenwissen verfügt.

Die maschinelle Erhebung aus Abschnitt 6.8 beantwortet nur, ob die zur Rekonstruktion nötigen Angaben vorliegen. Ob jemand mit ihnen zu einem Urteil kommt, ist damit offen. Jacovi & Goldberg (2020) fassen genau diese Eigenschaft als abstufbar und nicht als binär, weshalb die Frage praktisch lautet, in welchem Ausmaß und für welchen Zweck eine Information zur Rekonstruktion taugt.

Für diese Beurteilung ist Erfahrungswissen erforderlich, das nur Personen besitzen, die das Planungssystem, die fachlichen Regeln und die Folgen fehlerhafter Korrekturen kennen. Bogner et al. (2014) bezeichnen diesen Zugang als Experteninterview und unterscheiden den explorativen, den systematisierenden und den theoriegenerierenden Typ. Die vorliegende Erhebung ist überwiegend dem systematisierenden Typ zuzuordnen, weil sie auf Prozess- und Anwendungswissen zielt. Sie enthält zugleich deutungswissensbezogene Anteile, weil mehrere Fragen nach Bewertungen und Wahrnehmungen fragen und nicht nach Sachverhalten.

6.9.2 Konstruktion des Erhebungsinstruments

Die Entwicklung folgt dem fünfphasigen Rahmenwerk von Kallio et al. (2016) und verwendet als vorhandenes Wissen die Forschungsfragen, die Messdimensionen aus Kapitel 5 und die in Kapitel 2 dargestellte Literatur. Die Fragenformulierung folgt dem SPSS-Prinzip nach Helfferich (2011) mit den Schritten Sammeln, Prüfen, Sortieren und Subsumieren. Helfferich begründet es mit dem Problem des Fremdverstehens, also der Gefahr, im eigenen Relevanzsystem gefangen zu bleiben, was hier besonders erheblich ist, weil das untersuchte System von mir selbst konzipiert und implementiert wurde.

Beim Prüfschritt wurden deshalb gezielt solche Formulierungen entfernt, die eine der Varianten sprachlich begünstigen. Die Bezeichnungen A, B und C aus der Messanordnung finden keine Verwendung, weil sie eine Rangfolge nahelegen und in der Anwendung nicht vorkommen. Die Varianten werden stattdessen über ihre fachlichen Merkmale benannt.

Die Skala ist für alle Vergleichsfragen identisch und fünfstufig, mit den Polen bei den beiden verglichenen Varianten und einer inhaltlich besetzten Mitte. Diese Mitte bedeutet, dass kein relevanter Unterschied wahrgenommen wurde, und ist von der Option, eine Frage nicht beurteilen zu können, getrennt. Der Vergleich ist relativ und nicht absolut angelegt, weil bei zwei bis vier Bewertenden absolute Urteile je Variante einen gemeinsamen Maßstab voraussetzen würden, den es nicht gibt.

6.9.3 Ableitung der Fragen

Die Fragen sind deduktiv aus den Forschungsfragen und den Messdimensionen abgeleitet. Die folgende Tabelle weist für jede Frage aus, worauf sie zielt und woher sie stammt.

Frage	*Dimension*	*Bezug*
1 Fachlicher Eindruck	fachliche Korrektheit	UF1
2 Nachvollziehbarkeit	Nachvollziehbarkeit	UF3
3 Fehlerrekonstruktion	Nachvollziehbarkeit	UF3
4 Informationsbereitstellung	Nachvollziehbarkeit	UF3
5 Umgang mit Unsicherheit	Entscheidungsstabilität	UF2
6 Regelbereitstellung	alle	Ablation
7 Orchestrierung	alle	Ablation
8 Wiederholte Ausführung	Entscheidungsstabilität	UF2
9 Praktischer Einsatz	—	Diskussion
10 Gesamtbeurteilung	—	Diskussion

Tabelle 14: Ableitung der Erhebungsfragen

Die Fragen sind deduktiv aus den Forschungsfragen und den Messdimensionen abgeleitet. Die folgende Tabelle weist für jede Frage aus, worauf sie zielt und woher sie stammt.

Der vollständige Wortlaut des Erhebungsbogens findet sich in Anhang G.

Die Fragen eins bis fünf bilden den Gesamtvergleich zwischen der monolithischen Referenz und der graphbasierten Variante ab und entsprechen damit dem in Titel und Hauptforschungsfrage benannten Gegenüber. Die Fragen sechs und sieben zerlegen diesen Vergleich in die beiden Teilinterventionen und spiegeln die Ablationsanordnung aus Abschnitt 5.2 auf der Wahrnehmungsebene. Frage acht verbindet die gemessene Entscheidungsstabilität mit einer Beobachtung durch die Teilnehmenden, die Fragen neun und zehn sind offen gehalten. Helfferich empfiehlt eine abschließende offene Frage ausdrücklich, weil sie den Teilnehmenden erlaubt, für sie relevante Punkte zu ergänzen.

6.9.4 Abweichungen vom Rahmenwerk

Zwei der fünf Phasen nach Kallio et al. wurden nicht vollständig umgesetzt.

Eine Erprobung des Instruments mit einer nicht zur Zielgruppe gehörenden Person fand nicht statt. Gründe waren die geringe Zahl verfügbarer Fachpersonen und der enge Zeitrahmen im laufenden Projektbetrieb. Kallio et al. weisen einer solchen Erprobung die Funktion zu, unklare oder unpassende Fragen vor der Haupterhebung zu erkennen. Diese Absicherung fehlt.

Der Subsumierungsschritt nach Helfferich wurde ohne zusätzliche externe Beteiligung durchgeführt. Helfferich empfiehlt für diesen Schritt ausdrücklich eine Bearbeitung im Team oder zumindest eine Rückmeldung von außen, insbesondere aufgrund des Fremdverstehensproblems. Eine solche zusätzliche Absicherung war im Rahmen der Untersuchung nicht möglich.

6.9.5 Durchführung und Auswertung

Vorgesehen waren zwei bis vier Fachpersonen. Teilgenommen haben zwei Personen aus dem Projekt- und Kundenumfeld, die sowohl das Planungssystem als auch die fachlichen Korrekturregeln kennen. Diese Doppelrolle ist notwendig, weil eine Beurteilung fachlicher Korrektheit ohne Domänenkenntnis nicht möglich ist, und zugleich eine Einschränkung, weil die Teilnehmenden keine unabhängige Stichprobe bilden.

Beide arbeiteten über etwa zwei Tage mit dem System, jeweils mit vorbereiteten Datenbeständen und in mehreren Konfigurationen.

Für jede Variante stand ein eigener Satz von Snapshots bereit. Die Sätze enthalten dieselben Fehlerarten in derselben Anzahl, sind aber verschiedene Datenbestände, weil ein bereits korrigierter Snapshot für die nächste Variante nicht mehr fehlerhaft wäre. Die drei Varianten werden damit auf äquivalenten und nicht auf identischen Eingaben verglichen.

Die Erhebung erfolgte im Gespräch. Von einer der beiden Erhebungen liegt ein Transkript in Anhang H vor, das mit Einwilligung der teilnehmenden Person verwendet wird. Die zweite Erhebung wurde nicht aufgezeichnet und im Anschluss schriftlich zusammengefasst. Diese Zusammenfassung wird in Kapitel 7 entsprechend gekennzeichnet. Der ursprünglich vorgesehene verblindete Teil, bei dem Korrekturvorschläge in einem variantenneutralen Format vorgelegt worden wären, wurde nicht durchgeführt.

Als Auswertungsverfahren war eine inhaltlich strukturierende qualitative Inhaltsanalyse nach Kuckartz und Rädiker (2022) vorgesehen. Sie kam nicht zur Anwendung, weil sie einen Materialumfang voraussetzt, der eine Kategorienbildung und deren Absicherung erlaubt.

Bei zwei Erhebungen, von denen eine nicht aufgezeichnet ist, und bei überwiegend nicht abgegebenen Vergleichsurteilen wäre ein Kategoriensystem eine Form ohne tragende Datengrundlage.

Die Auswertung erfolgt stattdessen als belegte Einzelfalldarstellung. Berichtet werden die tatsächlich abgegebenen Antworten in absoluten Zahlen, die Begründungen für nicht abgegebene Urteile im Wortlaut, soweit ein Beleg vorliegt, und die daraus ableitbaren Aussagen über das Instrument und über die Wahrnehmbarkeit der untersuchten Unterschiede. Kapitel 7 berichtet diese Ergebnisse.

6.9.6 Reichweite der Aussagen

Die Erhebung liefert keine statistisch belastbaren Aussagen. Sie ist eine ergänzende Evidenzquelle und wird als solche berichtet.

Aus einer Präferenz der Teilnehmenden für eine Variante folgt kein Nachweis ihrer Überlegenheit. Zulässig ist die Aussage, dass die verfügbaren Informationen die Beurteilung einer Korrektur unterstützen oder nicht unterstützen. Nicht zulässig ist die Aussage, dass eine Architektur besser sei. Diese Zurückhaltung ist umso wichtiger, als die quantitative Messung für Korrektheit und Stabilität keinen durchgängigen Vorteil zeigt. Gerade darin liegt der Erkenntniswert der Erhebung, denn sie kann klären, ob der Unterschied zwischen den Varianten an anderer Stelle liegt als in der Korrektheit, oder ob er für Fachpersonen überhaupt wahrnehmbar ist.

6.10 Zusammenfassung

Die vier Dimensionen der Forschungsfrage sind damit in Messvorschriften überführt. Die fachliche Korrektheit wird in drei Stufen bestimmt, wobei Pfadzuordnung, Wertübereinstimmung und Belegbarkeit getrennt geführt werden.

Das Halluzinationsverhalten wird in vier Kategorien erhoben, von denen zwei einen Vergleich der Bedingungen tragen.

Die Entscheidungsstabilität ergibt sich aus dem Vergleich der fünf Wiederholungen je Fall und Bedingung, ohne dass eine Schwelle eingeführt wird. Die technische Nachvollziehbarkeit wird an sieben festgelegten Rekonstruktionsfragen bestimmt, deren menschliche Entsprechung Gegenstand der Expertenerhebung ist.

Für jede Kennzahl sind Bezugsgröße und Erhebbarkeit ausgewiesen, sodass in Kapitel 7 sichtbar bleibt, worauf sich eine Zahl stützt.

7 Ergebnisse

7.1 Datengrundlage

Die Hauptmessung umfasste siebzehn Fehlerfälle in drei Bedingungen mit je fünf Wiederholungen, insgesamt 255 Läufe. Der Katalog besteht aus zehn isolierten Fällen mit elf erwarteten Änderungen und sieben kombinierten Fällen mit achtzehn erwarteten Änderungen, zusammen 29. Da jede erwartete Änderung je Bedingung fünfmal beobachtet wurde, ergeben sich 435 Änderungsinstanzen, davon 145 je Bedingung.

Die fachliche Fallzahl beträgt siebzehn. Die 255 Läufe und die 435 Instanzen sind keine unabhängigen Beobachtungen, weil die fünf Wiederholungen je Zelle die Stabilität innerhalb eines Falls abbilden und mehrere Instanzen demselben Fall entstammen. Auf inferenzstatistische Verfahren wird deshalb verzichtet, sämtliche Vergleiche sind deskriptiv.

Modell, Parametrisierung und Schalterstellung waren über alle Läufe identisch, mit dem Antwortmodell gpt-4.1-2025-04-14 bei einer Temperatur von 0,3 und deaktiviertem episodischem Gedächtnis sowie deaktivierter menschlicher Freigabe. Je Bedingung trat über alle 85 Läufe genau eine Schalterkombination auf.

Die Erhebung fand am 22.08.2026 zwischen 11:30 und 16:17 UTC statt. Der finale Datensatz enthält 255 Positionen ohne Lücke oder Dopplung. Der Datensatz ist über seine Prüfsumme eindeutig identifiziert, sämtliche Auswertungsartefakte sind daraus abgeleitet.

7.2 Fachliche Korrektheit

Eine erwartete Änderung gilt als erfüllt, wenn eine erzeugte Korrektur am entsprechenden Element ansetzt und den erwarteten Wert exakt herstellt.

Kennzahl	A	B	C
korrekt erfüllt	54 / 145	66 / 145	61 / 145
falsch adressiert	38	32	30
adressiert, Korrektheit nicht bestimmbar	0	7	8
nicht adressiert	53	40	46
bedingte Korrektheit	54 / 92	66 / 98	61 / 91
Läufe mit Exact Match	26 / 85	33 / 85	30 / 85
Läufe mit zusätzlichen Vorschlägen	28 / 85	24 / 85	26 / 85
Läufe ohne adressierte Änderung	17 / 85	9 / 85	11 / 85

Tabelle 15: Ground-Truth-Erfüllung je Bedingung. Anteil der Änderungsinstanzen, die den Referenzwert an der vorgesehenen Stelle herstellen, bezogen auf 145 erwartete Instanzen.

Die Kategorien nicht adressiert und adressiert bei nicht bestimmbarer Korrektheit werden getrennt geführt. Die erste bezeichnet Änderungen, an denen keine erzeugte Korrektur ansetzte, die zweite solche mit Ansatzpunkt, aber ohne bewertbaren Wert. Über alle Bedingungen entfallen 139 Instanzen auf die erste und 15 auf die zweite Kategorie.

Zum Indikator Exact Match siehe Abschnitt 6.3.

Kennzahl	A → B	B → C	A → C
korrekt erfüllte Änderungen	+12	−5	+7
falsch adressierte Änderungen	−6	−2	−8
nicht adressierte Änderungen	−13	+6	−7
Läufe mit Exact Match	+7	−3	+4

Tabelle 16: Beobachtete Unterschiede zwischen den Bedingungen, getrennt für die Übergänge A→B, B→C und A→C.

Für die bedingte Korrektheit wird keine Differenz ausgewiesen, weil ihre Bezugsgrößen mit 92, 98 und 91 voneinander abweichen.

Die größte Einzeldifferenz tritt beim Übergang von A zu B auf, die Differenzen zwischen B und C fallen kleiner aus und weisen in die entgegengesetzte Richtung.

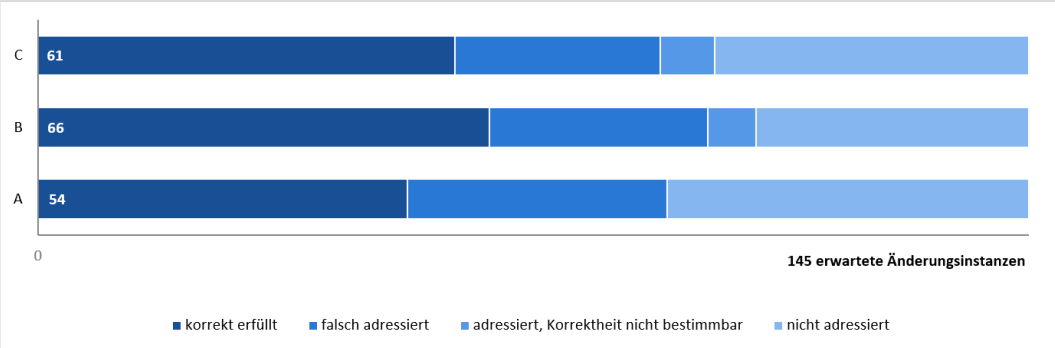


Abbildung 10: Anzahl pfadgenau korrekt erfüllter erwarteter Änderungen in den Bedingungen A, B und C auf einer Gesamtbasis von jeweils 145 Änderungen.

Die Aufschlüsselung nach Fallart zeigt unterschiedliche Muster. In den zehn isolierten Fällen liegen die Werte für Exact Match bei 25, 28 und 25 von je 50 Läufen. In den sieben kombinierten Fällen liegen sie bei 1, 5 und 5 von je 35 Läufen. Bei zehn beziehungsweise sieben Fällen bleibt diese Gegenüberstellung deskriptiv.

7.3 Funktionaler Endzustand

Die primäre Kennzahl verlangt Übereinstimmung auf derselben kanonischen Zielebene. Eine Korrektur, die ein übergeordnetes Element adressiert und dabei den erwarteten Wert herstellt, erfüllt diese Bedingung nicht.

Zur Prüfung, wie stark die berichteten Unterschiede von dieser Festlegung abhängen, wurde ergänzend der hergestellte Endzustand des Snapshots betrachtet. Von 435 Instanzen erfüllten 70 den erwarteten Wert über einen übergeordneten Zielpfad, ohne eine zusätzliche fachliche Änderung zu erzeugen, also rund 16%.

Kennzahl	**A**	**B**	**C**
primär korrekt erfüllt	54	66	61
zusätzlich funktional erfüllt	+28	+19	+23
funktional erfüllt gesamt	82 / 145	85 / 145	84 / 145
erfüllter Wert mit zusätzlichen Änderungen	0	0	0
Functional Exact Match	37 / 85	40 / 85	39 / 85

Tabelle 17: Funktionaler Endzustand je Bedingung. Gegenüberstellung der pfadgenauen Erfüllung und der Erfüllung über einen übergeordneten Zielpfad.

Unter dieser Betrachtung fallen die Unterschiede geringer aus. Für den Übergang von A zu B ergibt sich eine Differenz von drei statt zwölf Instanzen, für B zu C eine von minus eins statt minus fünf, für A zu C eine von zwei statt sieben. Die funktionale Betrachtung ist eine Sensitivitätsanalyse und ersetzt die primäre Kennzahl nicht.

7.4 Halluzinationsverhalten

Kategorie 1 klassifiziert einen Korrekturwert als fachliche Halluzination, wenn er von der Referenz abweicht und durch die bereitgestellte Evidenz nicht gestützt ist. Ein abweichender, aber gestützter Wert gilt nicht als Halluzination.

Kennzahl	A	B	C
ja	22	17	15
nein	69	81	76
nicht bestimmbar	54	47	54
Gesamtbasis	145	145	145
bestimmbare Basis	91	98	91
abweichend und evidenzgestützt	15	15	15

Tabelle 18: Evidenzbezogenes Halluzinationsverhalten je Bedingung, aufgeschlüsselt nach den vier Kategorien aus Tabelle 11.

Beide Bezugsgrößen werden gemeinsam genannt, weil die Zahl der nicht bestimmbaren Instanzen zwischen den Bedingungen variiert.

In vier Fällen wichen die dem Modell vorgelegten Regeltexte zwischen B und C voneinander ab. Werden sie ausgeschlossen, liegen B und C bei jeweils fünfzehn Instanzen mit ja, bei bestimmbaren Basen von 81 und 76. Dasselbe Ergebnis stellt sich beim alleinigen Ausschluss des Falls K07 ein.

Die Zahl der Instanzen, die von der Referenz abweichen und dennoch durch die vorgelegte Evidenz gestützt sind, beträgt in allen drei Bedingungen fünfzehn.

7.5 Folgefehler und Entscheidungsstabilität

Kategorie 4 erfasst neu entstandene Fehler über den Vergleich der Validierungsmeldungen vor der Korrektur mit denen nach abgeschlossener Neuberechnung, für alle drei Bedingungen mit derselben Funktion.

Über alle 255 Läufe wurde ein neu entstandener Fehler beobachtet, in Bedingung A im Fall K06, fünfte Wiederholung. Die Zahl der behobenen Fehler beträgt 120, 117 und 115, die der verbliebenen 0, 7 und 5. In 25 Läufen der Bedingung C wich eine Gegenprobe gegen den abgelegten Graphzustand von der gemeinsamen Berechnung ab, weil der Zustand die Bilanz der letzten Iteration führt und die

gemeinsame Berechnung die des gesamten Laufs. Diese Läufe werden nicht als Folgefehler gezählt.

Eine Zelle aus Fall und Bedingung gilt als entscheidungsstabil, wenn alle fünf Wiederholungen in allen fünf laufbezogenen Kennzahlen übereinstimmen.

Die uneinheitlichen Fälle sind K09 in Bedingung A und I08 in Bedingung B. Stabilität und Korrektheit fallen dabei auseinander. Von den entscheidungsstabilen Zellen lieferten in A neun, in B neun und in C acht in keiner Wiederholung einen Exact Match.

Über alle siebzehn Fälle zeigen fünfzehn dasselbe Exact-Match-Muster in allen drei Bedingungen. Abweichend sind K09 mit einem von fünf in A gegenüber fünf von fünf in B und C sowie I08 mit null von fünf in A, drei von fünf in B und null von fünf in C. Das Muster, bei dem A und B übereinstimmen und C abweicht, tritt in keinem Fall auf.

7.6 Technische Nachvollziehbarkeit

Erfasst wurde die Verfügbarkeit strukturierter Zustands- und Ablaufinformation. Die graphbasierte Bedingung legt je Lauf eine Ablaufspur ab, deren Einträge Knotennamen, Zeitstempel, Ausführungsdauer sowie je einen Eingangs- und Ausgangsprüfwert führen. An den Knoten der Korrekturerzeugung und der Schemaprüfung tritt eine Herkunftsangabe hinzu.

In C lag diese Ablaufspur für 81 von 85 Läufen vor. A und B erzeugen aufgrund ihrer Architektur kein entsprechendes Artefakt, ihre Zwischeninformationen liegen in verstreuten Laufzeitdateien.

In vier Läufen der Bedingung C, sämtlich im Fall K07, fehlt die Ablaufspur. Die von den Laufzeitskripten erzeugten Artefakte blieben in diesen Läufen erhalten, also Identifikationsaufruf und Antwort, Suchergebnisse, Korrekturaufruf und Vorschlag, Snapshot- und Validierungsstände sowie das Upload-Ergebnis. Alle vier Läufe tragen den Vermerk, dass eine manuelle Prüfung erforderlich ist, jeweils mit ausformulierter Begründung. Warum die graphspezifische Zustandsschicht fehlt, ist aus den vorhandenen Artefakten nicht bestimmbar.

Aus der Verfügbarkeit dieser Artefakte wird hier keine Aussage über die menschliche Fehlerrekonstruktion abgeleitet.

7.7 Ergebnisse der Expertenerhebung

An der Erhebung nahmen zwei Fachpersonen aus dem Projekt- und Kundenumfeld teil, die das System über jeweils etwa zwei Tage in mehreren Konfigurationen nutzten.

Beide Teilnehmenden gaben zu den Vergleichsfragen überwiegend keine Urteile ab. Als Begründung nannten beide übereinstimmend, dass die Nutzungsdauer für eine belastbare Unterscheidung zwischen den Varianten nicht ausreichte. Eine der beiden Personen formulierte, dass sich ohne ein produktives Einsatzszenario schwer testen lasse und dass genau dies bereits eine Erkenntnis sei. Beide gaben an, dass eine belastbare Beurteilung eine Erprobung im laufenden Betrieb über mehrere Wochen oder Monate voraussetze.

Eine der Teilnehmenden schlug für einen aussagekräftigeren Vergleich vor, denselben Fehler in mehrere Snapshots einzubringen und unter verschiedenen Einstellungen zu testen. Von dieser Erhebung liegt ein Transkript vor, von der zweiten eine im Anschluss erstellte schriftliche Zusammenfassung.

Aussagen zur menschlichen Fehlerrekonstruktion liegen aus dieser Erhebung nicht vor.

7.8 Nicht erhobene Kennzahlen

Für Kategorie 2, die strukturelle Halluzination, entsteht die erforderliche Provenienz ausschließlich im Graph-Pfad. Sie lag in A und B für keinen der 85 Läufe vor, in C für 81. Ein Vergleich der drei Bedingungen ist für diese Kategorie nicht möglich. Innerhalb von C wurde in achtzig Läufen keine strukturelle Halluzination klassifiziert, in einem eine, in vier war die Klassifikation nicht bestimmbar.

Für Kategorie 3, die Regelhalluzination, war die vorgelagerte Extraktion der in Anspruch genommenen Regeln aus dem Begründungstext für die Hauptmessung nicht spezifiziert. Die Kategorie wurde deshalb nicht automatisiert erhoben. Es liegen für sie keine Messwerte vor, was von einem durchgängig unbestimmten Messergebnis zu unterscheiden ist.

Für Kategorie 1 sind 155 der 435 Instanzen nicht bestimmbar. Für einen Lauf fehlt das Evidenzartefakt, weshalb dort die Halluzinationsklassifikation entfällt, die Korrektheitsbewertung jedoch bestehen bleibt.

Die Einordnung dieser Ergebnisse, die Betrachtung möglicher Ursachen sowie die Beantwortung der Forschungsfrage und der Unterfragen erfolgen in Kapitel 8.

8 Diskussion

Kapitel 7 hat die Messergebnisse beschrieben. Dieses Kapitel ordnet sie ein und beantwortet in Abschnitt 8.9 die Forschungsfrage sowie die drei Unterfragen. Drei Befunde tragen die Diskussion, nämlich an welcher Stelle der Architektur ein Unterschied entsteht, die empirische Trennung zweier bislang vermischter Konstrukte und der Nachweis einer Eigenschaft, die sich über Korrektheitskennzahlen nicht abbilden lässt.

8.1 Fachliche Korrektheit und die Wahl der Messgröße

Die drei Bedingungen unterscheiden sich in der fachlichen Korrektheit, und wie deutlich, hängt davon ab, welche der beiden Betrachtungen zugrunde gelegt wird. In der pfadgenauen Bewertung liegen zwischen der niedrigsten und der höchsten Bedingung zwölf Instanzen, in der funktionalen Betrachtung des Endzustands drei.

Dass beide Betrachtungen vorliegen, ist selbst ein Ergebnis. Siebzig Änderungsinstanzen, also rund sechzehn Prozent, stellten den erwarteten Wert über einen übergeordneten Zielpfad her. In der Primärbewertung zählen sie als nicht adressiert, im Endzustand ist der erwartete Wert gleichwohl vorhanden, und in keiner dieser Instanzen entstand dabei eine zusätzliche fachliche Änderung. Ohne die zweite Betrachtung wäre dieser Umstand unbemerkt geblieben, und der Vergleich der Bedingungen hätte deutlicher ausgesehen, als er ist.

Runeson & Höst (2009) empfehlen für Fallstudien die Betrachtung eines Konstrukts aus mehreren Perspektiven, weil die Konstruktvalidität daran hängt, ob das Gemessene dem entspricht, was erfasst werden sollte. Genau das leistet die Doppelbetrachtung, denn die primäre Kennzahl misst die Zielgenauigkeit einer Korrektur, die funktionale die Zustandsäquivalenz nach ihrer Anwendung.

Die primäre Kennzahl wird nicht rückwirkend ersetzt, weil sie vor Einsicht in die Ergebnisse festgelegt war. Für die Berichterstattung folgt daraus eine übertragbare Empfehlung.

Wer strukturierte Korrekturen gegen eine Referenz bewertet, sollte beide Zahlen nennen, denn die primäre beschreibt, wie präzise korrigiert wurde, die funktionale, ob korrigiert wurde.

8.2 Die beiden Interventionen

Der Übergang von A zu B verändert allein die Bereitstellung des Regelwerks, der Übergang von B zu C allein die Ablaufsteuerung. Die Anordnung erlaubt damit eine Aussage darüber, an welcher Stelle ein Unterschied entsteht.

Er entsteht bei der Regelbereitstellung. Beim Übergang von A zu B wurden zwölf erwartete Änderungen mehr pfadgenau erfüllt, bei funktionaler Betrachtung drei, und die Zahl der nicht adressierten Änderungen sank um dreizehn. Da genau diese Kategorie die auf einem übergeordneten Pfad ansetzenden Korrekturen aufnimmt, betrifft ein erheblicher Teil des Unterschieds die Zielgranularität der Korrekturen. Selektivität und Umfang des Regeltexts ändern sich dabei gemeinsam, weshalb der Vergleich den Gesamteffekt des Kartenkonzepts abbildet und keine Zuschreibung an einen seiner Bestandteile erlaubt.

Der beobachtete Befund wich von der ursprünglichen Erwartung ab. Die Untersuchung war auf die Ablaufsteuerung ausgerichtet, und ich hatte angenommen, dass sich ein Unterschied dort zeigt, wo der Verarbeitungsprozess explizit modelliert wird. Die modulare Regelbereitstellung war ursprünglich gar nicht als eigenständige Intervention vorgesehen, sondern wäre als Standardkonfiguration in der graphbasierten Variante mitgelaufen. Erst die in Abschnitt 4.1 beschriebene Auflösung der Konfundierung machte sie zu einer eigenen Bedingung. Wäre sie es nicht geworden, wäre der beobachtete Unterschied der graphbasierten Orchestrierung zugeschrieben worden. Die Anordnung mit drei Bedingungen hat damit einen Fehlschluss verhindert, der bei einem Zweiervergleich unvermeidlich gewesen wäre.

Beim Übergang von B zu C entsteht kein weiterer Korrektheitsunterschied. Die Primärbewertung liegt um fünf Instanzen niedriger, die funktionale um eine, und der Befund bleibt bestehen, wenn die vier Fälle mit abweichender Regelbasis ausgeschlossen werden. Wu et al. (2022) und Khot et al. (2023) führen die Wirkung einer Zerlegung darauf zurück, dass Teilaufgaben mit jeweils eigenem, fokussiertem Kontext bearbeitet werden. Die graphbasierte Variante macht Kontrollfluss und Zustandsführung explizit, führt dem Modell an den korrekturerzeugenden Stellen jedoch keine zusätzliche fachliche Information zu, die in Bedingung B nicht ebenfalls vorläge. Das beobachtete Ergebnis ist damit mit der Architektur vereinbar und keine Anomalie.

Für die Praxis ist dieser Befund unmittelbar verwertbar. Der erhebliche Aufwand einer graphbasierten Orchestrierung lässt sich nicht über einen Korrektheitsgewinn rechtfertigen, wohl aber über die in Abschnitt 8.5 beschriebene Nachvollziehbarkeit. Wer zuerst an der Ergebnisqualität arbeiten will, setzt an der Regelbereitstellung an, weil sie mit deutlich geringerem Aufwand veränderbar ist.

8.3 Halluzinationsverhalten

Der wichtigste Befund dieser Dimension betrifft nicht den Vergleich der Bedingungen, sondern die Messung selbst.

In jeder der drei Bedingungen wichen fünfzehn Instanzen von der Referenz ab und waren zugleich durch die bereitgestellte Evidenz gestützt. In den vorliegenden Daten fallen die beiden Konstrukte damit auseinander, denn ein Wert kann falsch sein, ohne erfunden zu sein. Dass diese Zahl in allen drei Bedingungen identisch ist, spricht dafür, dass es sich um eine Eigenschaft der Aufgabenstellung handelt und nicht um ein Artefakt einer einzelnen Konfiguration.

Huang et al. (2025) und Ji et al. (2023) fassen Halluzinationen als Ausgaben, die durch den gegebenen Kontext nicht gedeckt sind, und trennen damit die Deckung durch den Kontext von der Übereinstimmung mit einem externen Sollwert. Für die Bewertung strukturierter Ausgaben folgt daraus eine Konsequenz, die in dieser Arbeit gezogen wurde. Eine Auswertung, die beides in einer Kennzahl führt, würde die genannten fünfzehn Instanzen als Halluzinationen zählen und damit einen Befund über die Aufbereitung der bereitgestellten Evidenz als Befund über eine unbelegte Modellausgabe ausgeben.

Für den Vergleich der Bedingungen ist die Aussagekraft begrenzt. Zwei der vier Kategorien tragen einen bedingungsübergreifenden Vergleich, und die zwischen den beiden kartenbasierten Bedingungen beobachtete Differenz von zwei Instanzen entfällt vollständig, sobald ein einzelner Fall ausgeschlossen wird. Die über alle drei Bedingungen beobachtete Folge 22, 17 und 15 ist als Verlauf zu lesen und nicht als nachgewiesener Effekt, weil die Bezugsgrößen voneinander abweichen.

8.4 Entscheidungsstabilität

Alle drei Varianten reproduzierten ihre fachlichen Entscheidungen bei fünfmaliger Ausführung identischer Eingaben in der überwiegenden Mehrzahl der Fälle, mit 14, 15 und 14 von je 17 Zellen. Für ein System, das auf einem stochastischen Sprachmodell aufsetzt und in einem produktionsnahen Umfeld arbeitet, ist das ein tragfähiges Ergebnis. Die im Projektbetrieb beobachtete Instabilität, die den Ausgangspunkt der Arbeit bildete, tritt unter kontrollierten Bedingungen deutlich seltener auf als erwartet.

Ebenso deutlich ist, dass Stabilität und Richtigkeit auseinanderfallen. In A und B erreichten je neun und in C acht der entscheidungsstabilen Zellen in keiner Wiederholung einen Exact Match. Ein System kann eine Entscheidung zuverlässig reproduzieren, ohne dass diese Entscheidung zutrifft. Für den produktiven Einsatz

ist Stabilität deshalb kein Qualitätsnachweis, sondern die Voraussetzung dafür, dass ein Verhalten überhaupt beurteilt und gezielt verändert werden kann. Wer ein instabiles System verbessert, korrigiert ein bewegliches Ziel.

Für einen der instabilen Fälle ließ sich der Wirkungsweg über die aufgezeichnete Provenienz vollständig rekonstruieren, was Abschnitt 8.9 im Rahmen von UF2 ausführt.

Die im Exposé formulierte Ausgangserwartung lautete, dass die explizite Modellierung von Verarbeitungsschritten den Entscheidungsraum je Teilschritt enger fasst und dadurch zu konsistenteren Korrekturen führt. Diese Erwartung wird durch die Daten nicht gestützt, denn C erreicht denselben Wert wie A und liegt numerisch unter B. Zwei Randbefunde weisen in die entgegengesetzte Richtung, nämlich dass C als einzige Bedingung ohne einen Fall mit uneinheitlichem Ergebnis bleibt und dass die maximale Zahl verschiedener Resultate innerhalb einer Zelle dort niedriger liegt. Beide beruhen auf einer Differenz von einem Fall beziehungsweise einer Zelle und sind zu schwach, um die Erwartung zu tragen.

Die Messung betrifft ausschließlich die Stabilität bei identischer Eingabe, also die von Wang et al. (2023) verwendete Streuung über Wiederholungen, und nicht die von Sclar et al. (2024) untersuchte Empfindlichkeit gegenüber variierten Eingaben.

8.5 Verhalten im Zielsystem

Neben dem Vergleich der Bedingungen erlaubt die Messung eine Aussage über das Verhalten des Systems insgesamt, weil jeder Lauf die Korrektur tatsächlich anwendete und das Zielsystem sie neu validierte.

Über alle 255 Läufe entstand genau ein neuer Fehler. Für ein System, das automatisiert in einen produktionsnahen Datenbestand schreibt, ist das der praktisch bedeutsamste Einzelbefund der Untersuchung, denn das Risiko, das den Ausgangspunkt der Arbeit bildete, bestand gerade darin, dass eine syntaktisch valide Korrektur unbemerkt Folgeschäden auslöst. Die Zahl der behobenen Fehler lag in allen drei Bedingungen im Bereich zwischen 115 und 120. Beide Befunde sind an die Bedingungen der Messung gebunden, also an konstruierte Fehlerfälle, abgeschaltetes episodisches Gedächtnis und abgeschaltete Freigabepflicht. Sie tragen keine Aussage über beliebige Produktivdaten, wohl aber über das Verhalten der verglichenen Verarbeitungspfade unter kontrollierten Bedingungen.

8.6 Nachvollziehbarkeit

In dieser Dimension besteht der deutlichste Unterschied zwischen den Bedingungen, und er ist nicht gradueller, sondern grundsätzlicher Natur.

Die graphbasierte Bedingung legt in 81 von 85 Läufen einen zusammenhängenden Ablaufzustand ab. Für jeden ausgeführten Verarbeitungsschritt sind Name, Zeitpunkt, Dauer sowie Ein- und Ausgangsprüfwert festgehalten, an den korrekturerzeugenden Stellen ergänzt um die Angabe, welcher Regeltext und welcher Datenausschnitt tatsächlich verwendet wurden. Damit ist für einen abgeschlossenen Lauf maschinell rekonstruierbar, welche Schritte in welcher Reihenfolge abliefen und worauf sich jeder von ihnen stützte. Die Bedingungen A und B erzeugen konstruktionsbedingt kein entsprechendes Artefakt, ihre Zwischeninformationen liegen verstreut über einzelne Laufzeitartefakte vor, und die verwendete Regelgrundlage erscheint dort ausschließlich in einer Bildschirmausgabe, die mit dem Unterprozess verloren geht.

Entscheidend für die Belastbarkeit dieser Information ist ihre Herkunft. Die Einträge werden vom Programmcode aufgezeichnet und nicht vom Modell berichtet. Turpin et al. (2023) zeigen, dass modellgenerierte Begründungen den tatsächlichen Entscheidungsweg systematisch verfehlen können, weshalb eine aus Selbstauskünften bestehende Spur diesen Nachweis nicht trüge. Jacovi & Goldberg (2020) plädieren dafür, Nachvollziehbarkeit als abstufbare Eigenschaft zu behandeln, und genau diese Abstufung wird hier sichtbar, denn beide Ablaufformen erzeugen auswertbare Artefakte, unterscheiden sich aber in deren Zusammenhang, Reihenfolge und Herkunftsangabe.

Der praktische Wert zeigt sich an einem konkreten Fall. Die in Abschnitt 8.4 genannte Instabilität ließ sich ausschließlich deshalb rekonstruieren, weil die Provenienz festhielt, welche Regelkarten in welcher Wiederholung geladen waren. In den Bedingungen A und B wäre derselbe Fall als unerklärte Streuung im Datensatz verblieben.

In vier Läufen fehlt der Ablaufzustand vollständig. Er zeigt, dass die zusätzliche Information an den erfolgreichen Abschluss des Laufs gebunden war und damit gerade in jenen Fällen fehlte, in denen ein Fehler aufgetreten war. Er war zugleich unmittelbar behebbar. Nach Abschluss der Messung wurde die Ablage auf einen Ausführungsmodus umgestellt, der den Zustand nach jedem Verarbeitungsschritt ausgibt und ihn bei einer Ausnahme als ausdrücklich unvollständig gekennzeichnet ablegt. Diese Änderung ist kein Bestandteil des Architekturvergleichs und geht in keine Kennzahl aus Kapitel 7 ein.

Ob die vorhandene Information die menschliche Fehlersuche unterstützt, ist eine davon zu trennende Frage. Sie sollte die Expertenerhebung beantworten und blieb offen, weil beide Teilnehmenden nach jeweils etwa zwei Tagen Nutzung keine belastbare Unterscheidung zwischen den Varianten treffen konnten. Aus der technischen Verfügbarkeit wird deshalb kein Rückschluss auf den Nutzen für die menschliche Fehlersuche gezogen.

8.7 Zusammenführung der beiden Evidenzquellen

Die Untersuchung stützt sich auf zwei methodisch unabhängige Zugänge, die kontrollierte Messung über 255 Läufe und die Erhebung bei zwei Fachpersonen. Beide kommen zu einem vergleichbaren Bild, denn die Messung zeigt in den ergebnisbezogenen Dimensionen Unterschiede, die sich unter funktionaler Betrachtung auf wenige Instanzen reduzieren, und zwei Personen mit unmittelbarer Projektkenntnis konnten nach jeweils zwei Tagen intensiver Nutzung keine belastbare Unterscheidung treffen. Dass zwei so verschiedene Zugänge zu derselben Einschätzung führen, stützt den zentralen Befund der Arbeit.

Die Übereinstimmung ist mit Vorsicht zu lesen. Zwei Teilnehmende tragen keine belastbare Aussage, und die ausbleibende Beurteilung kann ebenso an der kurzen Nutzungsdauer wie an der Größe der Unterschiede liegen. Beide Erklärungen sind mit den Daten vereinbar.

Eine der Teilnehmenden schlug für einen aussagekräftigeren Vergleich vor, denselben Fehler in mehrere Datenbestände einzubringen und unter verschiedenen Einstellungen zu testen. Dieser Vorschlag entspricht dem Grundprinzip der in Kapitel 5 beschriebenen Anordnung und stützt damit die gewählte Methodik aus externer Perspektive.

8.8 Bedrohungen der Validität

Die fachliche Analyseeinheit umfasst siebzehn Fälle. Die 255 Läufe und 435 Änderungsinstanzen stellen keine unabhängigen Beobachtungen dar. Sämtliche berichteten Unterschiede sind daher deskriptiv zu verstehen. Differenzen von drei, fünf oder zwölf Änderungsinstanzen entsprechen wenigen Fällen

Zwei der vier Halluzinationskategorien tragen keinen Vergleich der Bedingungen. Gemessen wurde jener Teil, für den ein bedingungsübergreifendes Instrument vorlag, und jede Aussage zu dieser Dimension bezieht sich entsprechend auf diesen Teil.

Die in Abschnitt 8.1 beschriebene Eigenschaft der Primärkennzahl betrifft rund sechzehn Prozent aller Änderungsinstanzen und wirkt auf Bedingung A am stärksten. Sie verschiebt insbesondere die Vergleiche, an denen A beteiligt ist.

In vier von siebzehn Fällen unterschieden sich die vorgelegten Regeltexte zwischen B und C. Für die Korrektheit bleibt der Befund bei Ausschluss dieser Fälle bestehen, für das Halluzinationsverhalten entfällt die beobachtete Differenz vollständig.

Beide Erhebungen sind zeitlich eng begrenzt. Die Messung bildet 255 Läufe innerhalb weniger Stunden bei abgeschaltetem episodischem Gedächtnis ab, die Expertenerhebung eine Nutzung über zwei Tage durch zwei Personen aus dem Projektumfeld, ohne Verblindung und mit nur einer Aufzeichnung. Über das Verhalten im längerfristigen produktiven Einsatz trifft die Untersuchung keine Aussage.

Diese Punkte begrenzen die Reichweite der Aussagen, ohne die berichteten Befunde in Frage zu stellen. Sie betreffen alle drei Bedingungen gleichermaßen und verschieben den Vergleich nicht in eine bestimmte Richtung. Der zentrale Befund, dass sich die Ansätze in den ergebnisbezogenen Dimensionen kaum und in der Verfügbarkeit strukturierter Ablaufinformation grundsätzlich unterscheiden, ist von ihnen nicht berührt.

8.9 Beantwortung der Forschungsfrage und der Unterfragen

Hauptforschungsfrage

> *Inwiefern unterscheiden sich eine monolithische Systemprompt-Struktur, eine modulare Regelbereitstellung und eine graph-basierte Orchestrierung hinsichtlich fachlicher Korrektheit, Halluzinationsverhalten, Entscheidungsstabilität und Nachvollziehbarkeit bei der automatisierten Validierung und Korrektur strukturierter JSON-Daten in einem produktionskritischen Umfeld?*

Die drei Ansätze unterscheiden sich in einer der vier Dimensionen grundsätzlich und in den übrigen drei nur geringfügig.

Der grundsätzliche Unterschied liegt bei der Nachvollziehbarkeit. Die graphbasierte Bedingung stellt eine zusammenhängende Ablauf- und Herkunftsinformation bereit, die in den beiden anderen Bedingungen konstruktionsbedingt kein Gegenstück hat und die eine Rekonstruktion einzelner Korrekturentscheidungen erst möglich macht.

In den drei ergebnisbezogenen Dimensionen liegen die Ansätze eng beieinander. Der bei der fachlichen Korrektheit beobachtete Unterschied entsteht beim Übergang zur modularen Regelbereitstellung und gleicht sich unter Betrachtung des hergestellten Endzustands weitgehend an.

Daraus folgt eine für die Praxis verwertbare Aussage. Eine graphbasierte Orchestrierung ist in diesem Anwendungsfall keine Maßnahme zur Verbesserung der Ergebnisqualität, sondern eine Maßnahme zur Herstellung von Nachvollziehbarkeit, und sie ist als solche zu begründen.

UF1

Welche Unterschiede zeigen sich zwischen den drei untersuchten Ansätzen hinsichtlich fachlicher Korrektheit und evidenzbezogenem Halluzinationsverhalten?

Der beobachtete Unterschied entsteht bei der Regelbereitstellung. Der Übergang von der monolithischen zur modularen Form ist mit dem größeren der beiden Unterschiede verbunden, die zusätzliche graphbasierte Orchestrierung mit keinem erkennbaren weiteren Korrektheitsgewinn. Unter Betrachtung des hergestellten Endzustands verringern sich beide Abstände deutlich.

Für das Halluzinationsverhalten trägt der Vergleich der beiden kartenbasierten Bedingungen keine Aussage, und zwei der vier Kategorien lassen sich bedingungsübergreifend nicht erheben. Unabhängig davon liefert diese Dimension den methodisch tragfähigsten Befund der Arbeit. In jeder Bedingung waren fünfzehn Instanzen von der Referenz abweichend und zugleich durch die bereitgestellte Evidenz gestützt. Fachliche Falschheit und Halluzination sind in diesen Daten getrennte Eigenschaften, und eine Bewertung strukturierter Ausgaben, die sie in einer Kennzahl führt, misst nicht, was sie zu messen vorgibt.

UF2

Wie stabil reproduzieren die drei Systemvarianten ihre fachlichen Entscheidungen bei wiederholter Ausführung identischer Testfälle, und welche Instabilitätsmuster zeigen sich über die untersuchten Fehlerszenarien?

Alle drei Varianten reproduzierten ihre Entscheidungen in der überwiegenden Mehrzahl der Fälle, und die verbleibende Instabilität konzentriert sich auf wenige Fälle. Ein Vorteil einer der Varianten ist dabei nicht erkennbar.

Für einen der instabilen Fälle ließ sich der Wirkungsweg rekonstruieren. Die im Identifikationsschritt benannte Auswahl der Regelkarten schwankte zwischen den Wiederholungen und führte über die deterministische Regelauflösung zu unterschiedlichen Regeltexten, weshalb Modellstochastik und Architekturbestandteil hier keine unabhängigen Ursachen sind.

Ein zweiter Befund betrifft das Verhältnis von Stabilität und Richtigkeit. Ein erheblicher Teil der stabilen Zellen lieferte in keiner Wiederholung ein korrektes Ergebnis. Stabilität ist damit kein Qualitätsnachweis, sondern die Voraussetzung dafür, dass ein Verhalten gezielt verbessert werden kann. Diese Aussagen gelten ausschließlich für wiederholte identische Eingaben.

UF3

Wie unterscheiden sich die Systemvarianten hinsichtlich der verfügbaren technischen Zustands- und Ablaufinformation sowie der Fehlertoleranz in

iterativen Korrekturschleifen, und inwieweit unterstützen diese Informationen die menschliche Fehlerrekonstruktion und -lokalisierung?

Bei der technischen Zustands- und Ablaufinformation ist der Unterschied eindeutig. Die graphbasierte Variante stellt in der weit überwiegenden Zahl der Läufe ein Ablaufprotokoll mit Provenienzangaben bereit, für das es in den beiden anderen Varianten kein Gegenstück gibt. Der praktische Wert dieser Eigenschaft ist an einem Fall belegt, dessen Wirkungsweg ausschließlich über diese Angaben rekonstruierbar war.

Zur Fehlertoleranz erlaubt die Untersuchung eine eingeschränkte Aussage, weil Fehlerzustände nicht gezielt herbeigeführt wurden. Die vier technischen Ausnahmen in der graphbasierten Bedingung zeigen, dass die zusätzliche Verarbeitungsschicht einen eigenen Fehlermodus mitbringt, der behebbar war.

Der Teil der Unterfrage, der die menschliche Fehlerrekonstruktion betrifft, bleibt offen. Beide Teilnehmenden der Expertenerhebung gaben an, dass eine belastbare Beurteilung eine Erprobung im laufenden Betrieb über mehrere Wochen oder Monate voraussetzt. Diese Aussage ist selbst ein Ergebnis und bestimmt die Anlage der in Kapitel 9 beschriebenen Anschlussuntersuchung.

9 Fazit und Ausblick

9.1 Zusammenfassung der Untersuchung

Ausgangspunkt der Arbeit war ein Korrektursystem, dessen Verhalten im Projektbetrieb drei wiederkehrende Probleme zeigte, nämlich fachlich unzutreffende Korrekturen bei komplexen Datenbeständen, eine schwer rekonstruierbare Entscheidungslogik und uneinheitliche Ergebnisse bei gleichartigen Eingaben. Aus diesen Beobachtungen entstand die Frage, ob eine explizite Modellierung des Verarbeitungsablaufs messbare Verbesserungen bewirkt.

Zur Beantwortung wurde ein Ablationsdesign mit drei Messbedingungen aufgebaut, die sich paarweise in genau einer Eigenschaft unterscheiden. Siebzehn Testfälle mit insgesamt 29 über kontrollierte Fehlerinjektion konstruierten Referenzwerten wurden in jeder Bedingung fünfmal ausgeführt. Sämtliche Auswertungen erfolgten auf abgelegten Artefakten eines vor der Erhebung eingefrorenen Codestands. Ergänzt wurde die Messung durch eine strukturierte Erhebung bei zwei Fachpersonen aus dem Projekt- und Kundenumfeld.

9.2 Zentrale Befunde

Die fachliche Korrektheit unterscheidet sich zwischen den Bedingungen, und wie deutlich, hängt von der gewählten Betrachtung ab. In der pfadgenauen Bewertung liegt der größere Unterschied beim Übergang von der monolithischen zur modularen Regelbereitstellung, während die zusätzliche graph-basierte Orchestrierung keinen weiteren Gewinn zeigt. Wird der tatsächlich hergestellte Endzustand betrachtet, gleichen sich die drei Ansätze weitgehend an.

Das Halluzinationsverhalten fällt über die Bedingungen hinweg gering unterschiedlich aus. Der zwischen den beiden kartenbasierten Bedingungen beobachtete Abstand entfällt vollständig, sobald ein einzelner Fall ausgeschlossen wird, und trägt damit keine Aussage.

Die Entscheidungsstabilität ist in allen drei Bedingungen hoch und unterscheidet sich zwischen ihnen kaum. Für ein System, das auf einem stochastischen Sprachmodell aufsetzt, ist dieses Ergebnis für sich genommen bemerkenswert. Ebenso deutlich zeigt sich, dass Stabilität und Richtigkeit auseinanderfallen, denn ein erheblicher Teil der stabilen Zellen lieferte in keiner Wiederholung ein korrektes Ergebnis.

Bei der Nachvollziehbarkeit besteht der einzige grundsätzliche Unterschied. Die graph-basierte Variante erzeugt eine zusammenhängende Ablauf- und Herkunftsinformation, für die es in den beiden anderen Bedingungen konstruktionsbedingt kein Gegenstück gibt. Ihr praktischer Wert zeigte sich an einem Fall, dessen Instabilität ausschließlich über die aufgezeichnete Herkunft der geladenen Regeln rekonstruierbar war.

Für den produktiven Einsatz relevant ist außerdem, dass über alle 255 Läufe genau ein neuer Fehler im Zielsystem entstand. Das Risiko, das den Ausgangspunkt der Arbeit bildete, nämlich eine unbemerkt übernommene fehlerhafte Korrektur, trat damit unter den Bedingungen der Messung nahezu nicht auf.

9.3 Methodischer Beitrag

Der wesentliche Beitrag dieser Arbeit liegt in der Messung selbst.

Fachliche Unrichtigkeit und Halluzination sind zwei verschiedene Eigenschaften einer Ausgabe. In jeder der drei Bedingungen wichen fünfzehn Änderungsinstanzen vom Referenzwert ab und waren zugleich durch die vorgelegten Daten gedeckt. Eine Auswertung, die beides in einer Kennzahl führt, würde diese Fälle als Halluzinationen zählen und damit einen Befund über die Aufbereitung der Evidenz als Befund über eine unbelegte Modellausgabe ausgeben. Für die Bewertung strukturierter LLM-Ausgaben folgt daraus, dass beide Konstrukte getrennt zu erheben sind.

Ebenso zeigt die Arbeit, dass die Wahl der Operationalisierung das Ergebnisbild verschiebt. Rund sechzehn Prozent aller Änderungsinstanzen stellten den erwarteten Wert über einen übergeordneten Zielpfad her. Je nachdem, ob die Zielgenauigkeit oder die Zustandsäquivalenz gemessen wird, fallen die Unterschiede zwischen den Bedingungen unterschiedlich deutlich aus. Beide Betrachtungen gehören deshalb gemeinsam berichtet.

Der dritte Beitrag betrifft die Anordnung. Die modulare Regelbereitstellung wurde erst im Verlauf der Konzeption als eigenständige Bedingung geführt, nachdem erkennbar wurde, dass sie sich sonst gemeinsam mit der Ablaufsteuerung verändert hätte. Genau in diesem Übergang liegt der einzige nennenswerte Korrektheitsunterschied. Ein Zweiervergleich hätte ihn gezeigt und der graph-basierten Orchestrierung zugeschrieben. Bei Untersuchungen, in denen mehrere Eigenschaften gemeinsam verändert werden, ist die Zwischenbedingung damit die Voraussetzung dafür, dass ein Ergebnis überhaupt zurechenbar ist.

9.4 Empfehlungen für die Praxis

Aus den Befunden lassen sich drei Empfehlungen ableiten.

Wer die Ergebnisqualität eines vergleichbaren Systems verbessern will, sollte zuerst die Form der Regelbereitstellung betrachten. Sie ist mit erheblich geringerem Aufwand veränderbar als die Ablaufsteuerung und war im untersuchten Fall der Ort, an dem sich überhaupt ein Unterschied zeigte.

Eine graph-basierte Orchestrierung ist als Maßnahme zur Herstellung von Nachvollziehbarkeit zu begründen und nicht als Maßnahme zur Verbesserung der Korrektheit. Wer sie mit einem erwarteten Qualitätsgewinn rechtfertigt, wird diesen Nachweis nach den vorliegenden Daten nicht führen können.

Die Ablage der Ablaufinformation sollte nach jedem Verarbeitungsschritt erfolgen und nicht erst nach erfolgreichem Abschluss eines Laufs. Andernfalls fehlt sie genau in den Fällen, in denen ein Fehler aufgetreten ist, also dort, wo sie gebraucht wird.

9.5 Reichweite der Ergebnisse

Die Aussagen gelten für ein System, ein Sprachmodell, einen Datenbestand und siebzehn über Fehlerinjektion konstruierte Fälle. Die verwendeten Fehlerarten wurden von beiden Teilnehmenden der Expertenerhebung als für den Anwendungskontext relevant bezeichnet, sodass sich die Konstruktion auf die Herbeiführung der Fehler bezieht und nicht auf ihre Art. Eine Übertragung der Ergebnisse auf andere Systeme oder Domänen ist damit gleichwohl nicht begründbar. Übertragbar ist das Vorgehen, nicht das Ergebnis.

Beide Erhebungszugänge erfassen zudem ein enges Zeitfenster. Über das Verhalten des Systems im längerfristigen produktiven Einsatz, in dem sich Korrekturmuster wiederholen und das episodische Gedächtnis wirksam wird, trifft die Untersuchung keine Aussage.

9.6 Ausblick

Der nächste Untersuchungsschritt ergibt sich unmittelbar aus dem Verlauf dieser Arbeit. Beide Teilnehmenden der Expertenerhebung gaben übereinstimmend an, dass eine belastbare Beurteilung der Unterschiede zwischen den Varianten eine Erprobung im laufenden Betrieb über mehrere Wochen oder Monate voraussetzt. Insbesondere Nachvollziehbarkeit und Fehlerrekonstruktion zeigen sich nach dieser Einschätzung erst bei wiederholter Nutzung und bei wechselnden Fehlerbildern.

Eine Längsschnittuntersuchung unter produktiven Bedingungen ist damit keine allgemeine Forschungsempfehlung, sondern eine aus der Praxis begründete Anforderung an das Studiendesign.

Drei weitere Anschlusspunkte ergeben sich aus der Anlage der Messung. Die vierte mögliche Kombination aus vollständigem Regelwerk und graph-basierter Orchestrierung wurde nicht erhoben, weshalb eine Wechselwirkung zwischen den beiden Eigenschaften unbestimmt bleibt. Die Robustheit gegenüber strukturell abweichenden Eingaben wurde nicht untersucht und erfordert einen Testfallkatalog mit kontrollierten Variationsreihen desselben Falls. Und die Kategorie der Regelhalluzination, also die Berufung auf nicht vorhandene oder falsch wiedergegebene Regeln, wurde nicht automatisiert erhoben. Für sie wäre ein maschineller Abgleich zwischen den in einer Begründung beanspruchten und den tatsächlich geladenen Regeln zu entwickeln, wofür die in dieser Arbeit erzeugte Herkunftsinformation bereits die Grundlage bietet.

Literaturverzeichnis

Baltes, S., Angermeir, F., Arora, C., Barón, M. M., Chen, C., Böhme, L., Calefato, F., Ernst, N., Falessi, D., Fitzgerald, B., Fucci, D., He, J., Treude, C., Kalinowski, M., Lambiase, S., Russo, D., Lungu, M., Montes, C. M., Prechelt, L., … Wagner, S. (2026). *Guidelines for Empirical Studies in Software Engineering involving Large Language Models* (arXiv:2508.15503). arXiv. https://doi.org/10.48550/arXiv.2508.15503

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., & Hoefler, T. (2024). Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(16), 17682–17690. https://doi.org/10.1609/aaai.v38i16.29720

Bogner, A., Littig, B., & Menz, W. (2014). *Interviews mit Experten: Eine praxisorientierte Einführung*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-531-19416-5

Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024). RAGAs: Automated Evaluation of Retrieval Augmented Generation. In N. Aletras & O. De Clercq (Hrsg.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (S. 150–158). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.eacl-demo.16

Helfferich, C. (2011). *Die Qualität qualitativer Daten*. VS Verlag für Sozialwissenschaften. https://doi.org/10.1007/978-3-531-92076-4

Henkel, V., Gehlhoff, F., Kube, D., Almutareb, A., Cruz, L., Hellingrath, B., Koch, P., Legat, C., Mohr, F., Oberle, M., Ocker, F., Schoeler, T., Thron, M.,

Töpfer, N. A., Vogt, L., & Xia, Y. (2026). *Foundation-Model-Based Agents in Industrial Automation: Purposes, Capabilities, and Open Challenges* (arXiv:2605.02592). arXiv. https://doi.org/10.48550/arXiv.2605.02592

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2025). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, *43*(2), 1–55. https://doi.org/10.1145/3703155

Jacovi, A., & Goldberg, Y. (2020). Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness? *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4198–4205. https://doi.org/10.18653/v1/2020.acl-main.386

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, *55*(12), 1–38. https://doi.org/10.1145/3571730

Kallio, H., Pietilä, A., Johnson, M., & Kangasniemi, M. (2016). Systematic methodological review: Developing a framework for a qualitative semi-structured interview guide. *Journal of Advanced Nursing*, *72*(12), 2954–2965. https://doi.org/10.1111/jan.13031

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., & Sabharwal, A. (2023). *Decomposed Prompting: A Modular Approach for Solving Complex Tasks* (arXiv:2210.02406). arXiv. https://doi.org/10.48550/arXiv.2210.02406

Kuckartz, U., & Rädiker, S. (2022). *Qualitative Inhaltsanalyse: Methoden, Praxis, Computerunterstützung: Grundlagentexte Methoden* (5. Auflage). Beltz Juventa.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., & Clark, P. (2023). *Self-Refine: Iterative Refinement with Self-Feedback* (arXiv:2303.17651). arXiv. https://doi.org/10.48550/arXiv.2303.17651

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Sclar, M., Choi, Y., Tsvetkov, Y., & Suhr, A. (2024). *Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting* (arXiv:2310.11324). arXiv. https://doi.org/10.48550/arXiv.2310.11324

Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H., & Chen, Y.-N. (2024). Let Me Speak Freely? A Study On The Impact Of Format Restrictions On Large Language Model Performance. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 1218–1236. https://doi.org/10.18653/v1/2024.emnlp-industry.91

Turpin, M., Michael, J., Perez, E., & Bowman, S. R. (2023). *Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting* (arXiv:2305.04388). arXiv. https://doi.org/10.48550/arXiv.2305.04388

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need* (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). *Self-Consistency Improves Chain of Thought Reasoning in Language Models* (arXiv:2203.11171). arXiv. https://doi.org/10.48550/arXiv.2203.11171

Wen, Y., Wang, Z., & Sun, J. (2024). MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models. In L.-W. Ku, A. Martins, & V. Srikumar (Hrsg.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (S. 10370–10388). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.558

Wu, T., Terry, M., & Cai, C. J. (2022). AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. *CHI Conference on Human Factors in Computing Systems*, 1–22. https://doi.org/10.1145/3491102.3517582

Abbildungsverzeichnis

Abbildung 1: Schematische Darstellung einer graph-basierten Architektur: Verarbeitungsschritte als Knoten, Übergänge als Kanten. 25

Abbildung 2: Systemkontext des untersuchten Korrektursystems, vom ERP über Smart Planning und den Snapshot bis zur Rückschreibung in den Shopfloor. 32

Abbildung 3: Ablauf der Korrekturpipeline von der Snapshot-Aufnahme bis zur Rückschreibung. 38

Abbildung 4: Auszug aus dem Regelwerk: Aufbau einer Regelkarte mit Bedingung, Prüflogik und zugeordnetem Validator. 41

Abbildung 5: In der Untersuchung implementierter LangGraph-Verarbeitungsgraph mit neun Verarbeitungsknoten, Rückkante für weitere fachliche Iterationen und separaten Abschlusszuständen. 52

Abbildung 6: Gekürzter Auszug einer Ablaufspur aus einem Messlauf der Bedingung C, mit Knotenfolge und Herkunftsangabe der geladenen Regeln. 55

Abbildung 7: Konfigurationsansicht des Systems mit den zur Laufzeit einstellbaren Parametern. 56

Abbildung 8: Darstellung des Graphzustands in der Benutzeroberfläche während eines laufenden Korrekturvorgangs. 57

Abbildung 9: Die drei Bewertungsstufen der fachlichen Korrektheit als Ablauf, mit den Ausgängen je Stufe und den Bedingungen für den Übergang zur nächsten. 72

Abbildung 10: Anzahl pfadgenau korrekt erfüllter erwarteter Änderungen in den Bedingungen A, B und C auf einer Gesamtbasis von jeweils 145 Änderungen. 87

Tabellenverzeichnis

Tabelle 1: Die verwendeten Bewertungsebenen 29

Tabelle 2: Die vier Agenten 35

Tabelle 3: Einige Skripte (Tools) der Korrekturpipeline 37

Tabelle 4: Die drei Vergleichsbedingungen 45

Tabelle 5: Die neun Verarbeitungsknoten 51

Tabelle 6: Einflussgrößen für die Messung 61

Tabelle 7: Konfiguration des verwendeten Sprachmodells und der Modellaufrufe 62

Tabelle 8: Änderungen und Präzisierungen des Untersuchungsdesigns 65

Tabelle 9: Gegenüberstellung der automatisierten Messung und der Expertenerhebung 67

Tabelle 10: Wahrheitstabelle der beiden Größen 73

Tabelle 11: Die vier Halluzinationskategorien 75

Tabelle 12: Erhebungsschema der Entscheidungsstabilität 78

Tabelle 13: Die sieben Rekonstruktionsfragen 79

Tabelle 14: Ableitung der Erhebungsfragen 82

Tabelle 15: Ground-Truth-Erfüllung je Bedingung. Anteil der Änderungsinstanzen, die den Referenzwert an der vorgesehenen Stelle herstellen, bezogen auf 145 erwartete Instanzen. 86

Tabelle 16: Beobachtete Unterschiede zwischen den Bedingungen, getrennt für die Übergänge A→B, B→C und A→C. 86

Tabelle 17: Funktionaler Endzustand je Bedingung. Gegenüberstellung der pfadgenauen Erfüllung und der Erfüllung über einen übergeordneten Zielpfad. 88

Tabelle 18: Evidenzbezogenes Halluzinationsverhalten je Bedingung, aufgeschlüsselt nach den vier Kategorien aus Tabelle 11 89

Quellcodeverzeichnis

Code 1: Anonymisierter Auszug aus einem Snapshot 34

Code 2: Validierung Snapshot 34

Code 3: Auszug aus dem Regelwerk 40

Anhang

Der Anhang enthält Materialien, die zur Nachvollziehbarkeit und Wiederholbarkeit der Untersuchung erforderlich sind, im Fließtext jedoch den Lesefluss unterbrechen würden. Sämtliche Kunden- und Personennamen sind ersetzt, technische Kennungen sind gekürzt oder pseudonymisiert.

Anhang A: Vollständiger Auszug aus einem Snapshot

Der folgende Auszug zeigt die Struktur eines Snapshots am Beispiel je eines Artikels, einer Ressource und eines Bedarfs. Sämtliche Bezeichnungen sind ersetzt, die Struktur und die Feldnamen entsprechen dem Original. Zwei ERP-Herkunftsfelder wurden entfernt.

```
{
  "articles": [
    {
      "articleId": "100005",
      "articleName": "Produkt A",
      "workPlanId": "SP10        SP01",
      "workItemConfigs": [
        { "workItemKey": "VOAR01", "rampUpTime": 15,   "netTimeFactor": 0
},
        { "workItemKey": "HE01",   "rampUpTime": 120, "netTimeFactor": 0.3
},
        { "workItemKey": "ABF01",  "rampUpTime": 1,    "netTimeFactor": 1
}
        …  (13 Arbeitsgänge insgesamt)
      ],
      "relDensityMin": 1.017,
      "relDensityMax": 1.02,
      "hygieneImpactClass": 50,
      "standardPackaging": "71125",
      "minOptimalBatchSize": 200,
      "maxOptimalBatchSize": 26000,
      "departmentId": "20100",
      "departmentName": "Abteilung Ost"
    }
    …  (422 Artikel)
  ],
```

```
  "workPlans": [
    {
      "workPlanId": "SP10        SP01",
      "workplanName": "Standardprozess",
      "preparationAroma": {
        "workItemKey":   "VOAR01",   "workItemDescription":   "Vorbereiten
Aroma",
        "rampupTimeEquipment": 1, "netTimeEquipment": 1,
        "rampupTimeWorker": 1,    "netTimeWorker": 1
      },
      "manufacturing": {
        "workItemKey": "HE01", "workItemDescription": "Herstellen",
        "rampupTimeEquipment": 1, "netTimeEquipment": 1,
        "rampupTimeWorker": 1,    "netTimeWorker": 1
      }
      …  (14 Prozessschritte: Vorbereiten, Warten, Herstellen,
          Filtern, Qualitätssicherung, Abfüllen)
    }
  ],

  "equipment": [
    {
      "equipmentId": "4c350df3-325d-427c-a8bf-5309d3b71910",
      "equipmentKey": "AAR01",
      "name": "Abfüllanlage 01",
      "workItems": ["ABF01"],
      "predecessors": ["MTO01", "MTO02", "MTO04", "MTO05"],
      "throughput": 570,
      "batchSizeMin": 0,
      "batchSizeMax": 200,
      "qualification": "Abfüllen-Q1",
      "departments": ["10100", "10300", "20100", "20200"],
      "hygieneRelevant": true,
      "lastHygieneImpactClass": 99
    }
    …  (446 Betriebsmittel)
  ],

  "demands": [
    {
      "demandId": "D100005_001",
```

```
      "articleId": "100005",
      "quantity": 98.9,
      "packaging": "71125",
      "dueDate": "2026-05-19T23:59:59Z",
      "dispatcherGroup": "20"
    },
    {
      "demandId": "D100005_002",
      "articleId": "100005",
      "quantity": 50,
      "packaging": "71125",
      "dueDate": "2026-05-22T23:59:59Z",
      "dispatcherGroup": "20"
    }
    …  (1.395 Bedarfe)
  ],

  "packagingEquipmentCompatibility": [
    { "packaging": "71125", "predecessors": ["ACO04"] }
    …  (22 Zuordnungen)
  ],

  "workerQualifications": [
    {
      "worker": { "workerId": "6003", "workerName": "Mitarbeiter 6003" },
      "qualifications": [
        { "qualification": "Abfüllen-Q1", "category": "Q" },
        { "qualification": "Mischen",     "category": "A" }
        …
      ]
    }
    …  (103 Mitarbeitende)
  ],

  "workerAvailability": [
    {
      "worker": { "workerId": "6003" },
      "availabilities": [
        { "startDatetime": "2026-05-18T04:00:00Z", "possibleOvertime": "0"
}
        …
      ]
```

```
    }
    …  (101 Mitarbeitende)
  ],

  "equipmentUnavailabilities": [
    {
      "equipmentId": "4c350df3-325d-427c-a8bf-5309d3b71910",
      "equipmentKey": "AAR01",
      "unavailabilities": [
        { "start": "2026-05-20T22:00:00Z",
          "end":   "2026-05-21T21:59:00Z",
          "note":  "Instandhaltungswoche" }
        …
      ]
    }
    …  (346 Sperrzeiten)
  ],

  "customerOrderPositions": [],

  "metadata": {
    "createdAt": "2026-05-18T14:02:15Z",
    "sourceFiles": ["bedarf.json", "artikel.json", "arbeitsplan.json",
                    "betriebsmittel.json", "qualifikationen.json", …]
  }
}
```

Anhang B: Vollständiger Auszug aus dem Regelwerk

Der folgende Auszug stammt aus dem monolithischen Regelwerk. Wiedergegeben sind zwei vollständige Regeln, die den Aufbau des Dokuments zeigen.

```
## 1. Duplizierte IDs (validate_unique_ids-duplicate)
**Problem:** Mehrere Einträge haben dieselbe ID.
**Strategie:** Nummeriere Duplikate mit `_1`, `_2`, `_3`, etc.
**Beispiel:**
```json
// Vorher
"articles": [
    {"articleId": "SPE_AR_fil"},
    {"articleId": "SPE_AR_fil"}
```
```

```
]
// Nachher
"articles": [
    {"articleId": "SPE_AR_fil"},
    {"articleId": "SPE_AR_fil_2"}
]
// oder
"articles": [
    {"articleId": "SPE_AR_fil_2"},
    {"articleId": "SPE_AR_fil_1"}
]
Mit Referenzen: Aktualisiere alle betroffenen Referenzen.
// Vorher
"articles": [{"articleId": "ART_001"}, {"articleId": "ART_001"}],
"demands": [
    {"demandId": "D001", "articleId": "ART_001"},
    {"demandId": "D002", "articleId": "ART_001"}
]
// Nachher
"articles": [{"articleId": "ART_001"}, {"articleId": "ART_001_2"}],
"demands": [
    {"demandId": "D001", "articleId": "ART_001"},
    {"demandId": "D002", "articleId": "ART_001_2"}
]
```

Part II

```
---
name: density-values
applies_to: [DENSITY_VALUES]
description: Numerische Werte aus array_context ableiten (Gruppenfilter,
Median)
---
# Card: DENSITY_VALUES  (`[validate_density_values]`)
Deriving a numeric field value from array_context — worked examples.
The general principle ("filter to the same group first") lives in
`_core.md`; this card carries
the detailed examples.
Source: `llm-validation-fix-rules.md` lines 208-282 (inventory rule R22).
## KRITISCH: Domain-Intelligence bei array_context

**Evidenzreihenfolge:**
```

```
1. Dichte desselben `articleId` aus einem bestätigten gültigen Snapshot
2. Gleicher Artikelstamm beziehungsweise eindeutig gleiche Produktvariante
3. Median mehrerer Artikel mit gleichem `departmentId` und `workPlanId`
Nur positive, numerische Vergleichswerte verwenden. `relDensityMin` muss
nach dem Fix `> 0`
sein. Wenn `relDensityMax` vorhanden ist, zusätzlich `relDensityMin <= re-
lDensityMax` prüfen,
ohne `relDensityMax` unnötig zu verändern. Bei weniger als zwei vergleich-
baren Artikeln und
ohne historischen Treffer keine Dichte erfinden.
**Wenn du array_context nutzt, FILTERE die items_before/items_after intel-
ligent!**
**Für articles Array:**
1. **Prüfe ZUERST, ob die Artikel zur gleichen Gruppe gehören:**
   - Vergleiche `departmentId` und `workPlanId`
   - Vergleiche Artikel-Präfix (z.B. "SPE_ZU" vs "SPE_PU")
2. **NUTZE NUR Artikel aus der gleichen Gruppe für Statistik:**
   ```json
   // FEHLER: articles[7] hat relDensityMin = 0
   // TARGET: {"articleId": "SPE_ZU_kl", "departmentId": "20300", "work-
PlanId": "7202"}
   // items_before enthält:
   [
        {"articleId": "SPE_AR_fil", "departmentId": "10100", "relDensi-
tyMin": 0.02},  //IGNORIERE - anderes dept
      {"articleId": "SPE_PU_kl", "departmentId": "30100", "relDensityMin":
0.05},   //IGNORIERE - anderes dept+prefix
       {"articleId": "SPE_ZU_gr", "departmentId": "20300", "workPlanId":
"7202", "relDensityMin": 0.01},  //NUTZE - gleiche Gruppe!
   ]
   // Richtige Analyse:
   // - Filtere NUR Artikel mit departmentId="20300" UND workPlanId="7202"
   // - Oder zumindest: Artikel mit Präfix "SPE_ZU_*"
   // - Berechne Median NUR aus gefilterten Werten
   // - Reasoning: "Filtered array_context to same product type (SPE_ZU_*,
dept 20300). Median relDensityMin is 0.01"
   ```
3. **Begründung explizit erwähnen:**
   - "Used only articles from same department (20300) and workplan (7202)"
   - "Filtered to same product prefix (SPE_ZU_*) for domain accuracy"
   - "Excluded unrelated article types (SPE_PU_*, SPE_AR_*)"
```

Anhang C: Gegenüberstellung von Regelwerk und Regelkarte

Die folgende Gegenüberstellung zeigt die beiden Formen der Regelbereitstellung.

Merkmal	*Bedingung A - vollständiges Regelwerk*	*Bedingungen B und C - modulare Karten*
Ablage	eine Datei: runtimes/llm-validation-fix-rules.md	13 Dateien in app/skills/
Bestand	1 Dokument	1 Kernbestand + 12 Regelkarten
Umfang des Bestands	34.899 Zeichen · 937 logische Zeilen	Kernbestand 12.214 Zeichen · 349 Zeilen; Karten zusammen 37.989 Zeichen
Gesamtbestand	**34.899 Zeichen**	**50.203 Zeichen** *(Kern + alle Karten)*
Auswahl je Fall	keine - immer alles	Kernbestand **immer**, dazu die für den Validator-Tag zuständigen Karten
Je Prompt vorgelegt	**immer 34.899 Zeichen**	**13.284 bis 27.967 Zeichen**, Median **20.335** *(gemessen, n = 81)*
Karten je Lauf	-	2 bis 4, Median 2, Mittel 2,37 *(gemessen, n = 81)*
Abhängigkeit vom Fehlerbild	keine - Text über alle Fälle identisch	Fallabhängig
Kartengröße (ganze Datei)	-	737 bis 11.724 Zeichen, **Median 1.473,5**
Kartengröße (Rumpf ohne Frontmatter - was in den Prompt geht)	-	535 bis 11.540 Zeichen, **Median 1.266,5**
Prüfsummenwert des Bestands	a3c14bd1b66cc1e3…	4d3808849946e2b8… *(14 Dateien gesamt)*

Anhang D: Definition des GraphState

Der GraphState ist ein typisiertes Objekt mit 22 deklarierten Feldern, das die Verarbeitungskette durchläuft. Die folgende Definition entspricht dem eingefrorenen Messstand.

```
class GraphState(TypedDict, total=False):
    """
    Gemeinsamer Zustand, der durch alle neun Knoten geführt wird.
    Jeder Knoten ergänzt die für ihn vorgesehenen Felder und fügt
    einen Eintrag zur Ablaufspur `trace` hinzu. `total=False` erlaubt,
    dass der Zustand während eines Laufs schrittweise aufgebaut wird.
    """
    # --- Identität und Lauf-Metadaten ---
    snapshot_id: str
    iteration: int
    # Tatsächlich von der Runtime angelegter `iteration-N`-Ordner.
    # Dieser Wert ist vom fachlichen Durchgang `iteration` getrennt.
    artifact_iteration_number: Optional[int]
    max_iterations: int
    architecture_mode: Literal["graph"]
    started_at: str
    finished_at: Optional[str]
    # --- Fehlerzustand ---
    errors_before: int
    errors_after: Optional[int]
    # Validierungsstand beim Eintritt in die Pipeline.
    initial_validation: Optional[dict]
    # Rohe Validierungsmeldungen nach Anwendung und Re-Validierung.
    final_validation: Optional[list]
    # --- Knotenausgänge ---
    # Knoten 2: Fehlerklassifikation.
    # Enthält Tag, Priorität, Begründung und ursprüngliche Fehlermeldung.
    classified_error: Optional[dict]
    # Knoten 3: Kontextsuche.
    # `lines_used` dokumentiert die Herkunft der verwendeten Daten.
    extracted_context: Optional[dict]
    # Knoten 4: Regelzuordnung.
    # Dokumentiert Modus, geladene Regelkarten und Hash des Regeltexts.
    matched_rules: Optional[dict]
    # Knoten 5: Korrekturerzeugung.
    # Beobachtungspunkt für die fachliche Korrektheit.
    correction_proposal: Optional[dict]
```

```
    # Vollständige Antwort von Knoten 5.
    # Knoten 6 prüft diese gegen das erwartete Ausgabeschema.
    correction_response: Optional[dict]
    # Knoten 6: Technische Prüfung.
    # Enthält Schemaergebnis, Wiederholungen und erkannte Fehler.
    technical_check: Optional[dict]
    # Knoten 7: Anwendung und Re-Validierung.
    # Beobachtungspunkt für neu entstandene Fehler.
    applied: Optional[dict]
    # Knoten 8: Ergebnisbewertung.
    # Die fachliche Entscheidung wird bewusst als eigener Zustand abgelegt.
    decision: Optional[dict]
    # Knoten 9: Variantenneutrales Endergebnis.
    # Wird deterministisch und ohne weiteren Modellaufruf erzeugt.
    final_answer: Optional[dict]
    manual_intervention_required: bool
    # --- Nachvollziehbarkeit ---
    # Ablaufspur mit Knotenname, Zeitstempel, Prüfwerten
    # und Ausführungsdauer jedes Verarbeitungsschritts.
    trace: list[dict]
# Erlaubte Werte für `decision["action"]`.
# `stop_uncertain` bildet Fälle ab, in denen keine belastbare
# automatisierte Korrektur bestimmt werden kann.
DECISION_ACTIONS = (
    "continue",
    "stop_valid",
    "stop_max_iter",
    "stop_uncertain",
)
```

Anhang E: Auswertungsprotokoll

Die folgenden Auswertungsregeln wurden nach Abschluss der Datenerhebung und vor Einsicht in die nach Bedingungen aufgeschlüsselten Korrektheitsergebnisse schriftlich festgelegt. Sie sind seither unverändert.

Bezug: Rohdatensatz der Hauptmessung vom 22.08.2026, Prüfsumme db867a26…

Codestand: Produktcode 15f2a44, Messstand 2ab9546

Status: seit der Festlegung unverändert

Die folgenden Auswertungsregeln wurden nach Abschluss der Datenerhebung und vor Einsicht in die nach Bedingungen aufgeschlüsselten Korrektheitsergebnisse schriftlich festgelegt.

E.1 Grundeinheit und Analyseeinheit

Die Grundeinheit der Bewertung ist die einzelne erwartete Änderung, nicht der Lauf und nicht der Fall. Der Testfallkatalog umfasst siebzehn Fälle mit insgesamt 29 erwarteten Änderungen. Bei fünf Wiederholungen je Fall und Bedingung ergeben sich 145 Änderungsinstanzen je Bedingung und 435 insgesamt.

Die Analyseeinheit für Aussagen über Fehlerarten und Systemverhalten bleibt davon getrennt und beträgt siebzehn. Wiederholungen sind keine zusätzlichen Fälle, sie bilden die Stabilität innerhalb eines Falls ab. Aus demselben Grund werden Ergebnisse in absoluten Zahlen berichtet und nicht in Prozentangaben.

E.2 Bestimmung des erwarteten Werts

Der erwartete Wert einer Änderung ist der Zustand vor der Injektion, nicht der injizierte Wert.

Die Referenz führt je Änderung den Zielpfad sowie den Zustand vor und nach der Injektion. Der Vorzustand liegt in zwei Schreibweisen vor, einmal mit Feldnamen und einmal als bloßer Wert bei Elementen innerhalb einer Liste. Beide werden vor dem Vergleich in dieselbe Form überführt. Alle 29 Werte ließen sich dabei fehlerfrei und typrichtig auflösen.

Diese Überführung ist eine Formatvereinheitlichung und keine Vergleichstoleranz.

E.3 Die dreistufige Bewertung

Die Stufen laufen in fester Reihenfolge und werden nie vertauscht.

Stufe 1 - Pfadzuordnung

Eine erzeugte Korrektur wird einer erwarteten Änderung genau dann zugeordnet, wenn beide Pfadangaben auf dasselbe Element im Datenbestand auflösen. Als Auflösungsgrundlage dient die eingefrorene Katalogdatei des jeweiligen Falls, nicht der aktuelle Zwischenstand eines Laufs. Diese Festlegung ist notwendig, damit die Grundlage für alle drei Bedingungen identisch ist und unabhängig davon, wie weit ein Lauf gekommen ist.

Löst ein Pfad mehrdeutig auf, gilt die Zuordnung als nicht bestimmbar. Löst ein Pfad auf kein Element auf, gilt dasselbe. Eine ungefähre Zuordnung gibt es nicht. Ohne eine erfolgreiche Stufe 1 finden Stufe 2 und Stufe 3 nicht statt.

Stufe 2 - Übereinstimmung mit dem erwarteten Wert
Erst nach erfolgreicher Zuordnung wird der vorgeschlagene Wert mit dem erwarteten verglichen, und zwar ohne jede Berücksichtigung dessen, worauf sich der Vorschlag stützt.

Bedingung	Ausgang
Wert stimmt mit dem erwarteten überein	korrekt
Wert weicht ab	falsch
kein bewertbarer Wert vorhanden oder Pfad nicht deterministisch auflösbar	nicht bestimmbar

Stufe 3 - Belegbarkeit
Unabhängig von Stufe 2 wird geprüft, ob der vorgeschlagene Wert durch die Daten gedeckt ist, die dem Modell vorlagen. Diese Stufe beantwortet die Frage nach der Deckung durch den Kontext und nicht die nach der Richtigkeit. Mögliche Ausgänge sind ja im Sinne einer Halluzination, nein und nicht bestimmbar.

E.4 Wahrheitstabelle beider Größen
Die beiden Größen überschreiben einander in keiner Richtung.

Fall	Übereinstimmung	Halluzination
erwarteter Wert getroffen, Belege vorhanden	korrekt	nein
erwarteter Wert verfehlt, Belege stützen den Wert	falsch	nein
erwarteter Wert verfehlt, Belege stützen nicht	falsch	ja
erwarteter Wert verfehlt, Belegartefakt fehlt	falsch	nicht bestimmbar
erwarteter Wert getroffen, Belegartefakt fehlt	korrekt	nicht bestimmbar
kein Wert vorgeschlagen, manuelle Prüfung verlangt	nicht bestimmbar	nicht bestimmbar
Pfad nicht deterministisch auflösbar	nicht bestimmbar	nicht bestimmbar

Die Trennung muss protokolliert werden, weil die Belegbarkeitsprüfung für einen erwartungsgemäßen und für einen abweichenden, aber gestützten Wert dieselbe Antwort liefert. Als Korrektheitsurteil verwendet, hätte sie abweichende Werte als korrekt gezählt.

Die zweite Tabellenzeile ist deshalb der Kern des Protokolls, denn ein Vorschlag kann vom erwarteten Wert abweichen und dennoch durch die vorgelegten Daten gedeckt sein.

E.5 Gründe für nicht bestimmbar

Jede Einstufung als nicht bestimmbar wird mit einem der folgenden fünf Gründe ausgewiesen. Ein unbegründetes nicht bestimmbar gibt es nicht.

Kein Wert vorgeschlagen. Manuelle Prüfung verlangt. Belegartefakt fehlt. Pfad mehrdeutig. Pfad ohne Treffer.

E.6 Behandlung fehlender Belegartefakte

Fehlt das Artefakt mit den Suchergebnissen eines Laufs, bleibt das Korrektheitsurteil bewertbar, weil es nicht von der Evidenz abhängt. Die Belegbarkeit gilt als nicht bestimmbar, mit ausgewiesenem Grund.

Begründung: Eine leere und eine fehlende Belegmenge sind nicht voneinander unterscheidbar. Eine leere Menge als nicht gestützt zu werten, würde ein fehlendes Artefakt in eine Halluzination verwandeln.

E.7 Kennzahlen auf Laufebene

Je Lauf werden sechs Zahlen erfasst. Alle sechs beziehen sich ausschließlich auf die Übereinstimmung mit dem erwarteten Wert, nicht auf die Belegbarkeit.

Kennzahl	Bedeutung
erwartet	Zahl der erwarteten Änderungen des Falls
korrekt erfüllt	richtiger Pfad und richtiger Wert
falsch adressiert	richtiger Pfad, abweichender Wert
nicht bestimmbar	adressiert, aber ohne bewertbaren Wert
nicht adressiert	keine erzeugte Korrektur setzte an dieser Änderung an
zusätzliche, nicht zuordenbare	Korrekturen ohne zugehörige erwartete Änderung

Bewertet wird die vollständige zeitliche Folge aller Vorschläge eines Laufs, aufsteigend nach Iteration, je Vorschlag zuerst die Hauptkorrektur und danach die mitgeführten weiteren Änderungen. Die einzeilige Zusammenfassung, die der Ausführungsablauf je Lauf schreibt, führt nur eine Korrektur und ist als Bewertungsgrundlage nicht ausreichend.

Der zusammenfassende Indikator. Ein Lauf erfüllt den Indikator genau dann, wenn alle erwarteten Änderungen auf dem richtigen Pfad mit dem richtigen Wert erfüllt wurden und keine zusätzliche fachliche Änderung auftrat.

Drei Festlegungen gehören dazu. Er misst ausschließlich die Übereinstimmung mit dem erwarteten Wert und nicht die Abwesenheit von Halluzinationen. Ein abweichender, aber belegter Vorschlag erfüllt ihn nie und kann in der Halluzinationsdimension gleichwohl unauffällig sein. Und er ist eine nach der Erhebung

festgelegte Aggregationsregel und keine der vor dem Einfrieren definierten Einzelwertkategorien, was bei der Ergebnisdarstellung kenntlich gemacht wird.
Ein teilweise richtiger Mehrfehlerlauf bleibt über seine Einzeländerungen sichtbar und wird nicht zu einem einzigen Verdikt zusammengefasst.

E.8 Funktionaler Endzustand als Sensitivitätsanalyse

Ergänzend zur pfadgenauen Bewertung wird geprüft, welcher Wert am erwarteten Zielpfad im Endzustand des Datenbestands vorliegt. Diese Betrachtung wurde nach Kenntnis der Primärergebnisse ergänzt und ist als Sensitivitätsanalyse angelegt.
Eine Änderung gilt als funktional erfüllt, wenn der erwartete Wert am erwarteten Zielpfad vorliegt, unabhängig davon, auf welcher Ebene die erzeugte Korrektur angesetzt hat, und sofern dabei keine zusätzliche fachliche Änderung entstanden ist.
Drei Festlegungen gehören dazu. Die Betrachtung ersetzt die Primärkennzahl nicht und wird immer gemeinsam mit ihr berichtet. Ein fehlerfreier Endzustand belegt keine Übereinstimmung mit der Referenz, weil ein Wert so gesetzt werden kann, dass die Prüfregel nicht mehr greift, ohne dass der zugrunde liegende Sachverhalt richtig abgebildet wäre. Und die Primärkennzahl wird nicht rückwirkend durch die günstigere der beiden Betrachtungen ersetzt.

E.9 Folgefehler

Folgefehler werden als eigene Größe geführt und mit der Korrektheit nicht verrechnet.
Grundlage ist der Vergleich des Fehlerstands vor der Korrektur mit dem Stand nach erfolgreicher Neuberechnung der Validierung. Erhoben werden je Lauf die behobenen, die verbliebenen und die neu entstandenen Fehler sowie die Arten der neu entstandenen.
Die bloße Fehlerzahl reicht dafür nicht aus, weil eine unveränderte Zahl sowohl bedeuten kann, dass nichts geschehen ist, als auch, dass ein Fehler behoben und ein anderer erzeugt wurde. Die Identität eines Fehlers wird aus Prüfkennung und Meldungsinhalt gebildet, weil die Validierungs-Engine keine stabile Fehlerkennung vergibt. Das ist eine Näherung und wird als solche berichtet.
Die Berechnung erfolgt für alle drei Bedingungen über dieselbe gemeinsam genutzte Funktion. Der in Bedingung C zusätzlich abgelegte Ablaufzustand dient ausschließlich der Gegenprobe und nicht als eigene Quelle, weil sonst eine Bedingung nach einem anderen Verfahren bewertet würde als die anderen beiden. Ein neu entstandener Fehler ändert die Bewertung des ursprünglichen Vorschlags nicht rückwirkend.

E.10 Erhebbarkeit der vier Halluzinationskategorien

Für jede der vier vor der Messung definierten Kategorien ist festgelegt, auf welcher Grundlage sie erhoben wird und für welche Bedingungen sie einen Vergleich trägt.

Kategorie 1, die fachliche Halluzination, wird über den Abgleich des vorgeschlagenen Werts mit den im Suchkontext vorliegenden Belegen erhoben und trägt einen Vergleich über alle drei Bedingungen.
Kategorie 2, die strukturelle Halluzination, wird über das abgelegte Ergebnis der Schemaprüfung erhoben. Dieses Artefakt entsteht ausschließlich im Graph-Pfad, weshalb die Kategorie außerhalb der Bedingung C nicht erhebbar ist und keinen Vergleich trägt.
Kategorie 3, die Regelhalluzination, setzt einen maschinellen Abgleich der in einer Begründung beanspruchten Regelreferenzen mit dem tatsächlich geladenen Regeltext voraus. Ein solches Verfahren wurde für die Hauptmessung nicht spezifiziert, weshalb die Kategorie definiert bleibt und keine Messwerte trägt. Das ist von einem durchgängig unbestimmten Messergebnis zu unterscheiden.
Kategorie 4, die Folgefehlererzeugung, wird nach den Regeln aus E.9 erhoben und trägt einen Vergleich über alle drei Bedingungen.

E.11 Bezugsgrößen

Jede relative Angabe wird mit ihrer Bezugsgröße genannt. Es werden drei unterschieden.

Bezugsgröße	Definition
Gesamtbasis	alle 145 erwarteten Änderungsinstanzen je Bedingung
bewertbare Basis für die Korrektheit	korrekt erfüllte plus falsch adressierte Instanzen
bestimmbare Basis für die Belegbarkeit	Instanzen, für die die Belegbarkeit entschieden werden konnte

Die letzten beiden sind nicht identisch und dürfen nicht gegeneinander gesetzt werden, denn ein Lauf ohne Belegartefakt bleibt in der Korrektheitsbasis und fällt aus der Belegbarkeitsbasis heraus.
Ein Quotient über alle vier Halluzinationskategorien wird nicht gebildet, weil er ein Verhältnis mit unbestimmter Bezugsgröße wäre.

E.12 Wiederholungen und Klassenbildung

Je Fall und Bedingung liegen fünf Wiederholungen vor, insgesamt 51 Zellen.
Die fünf Wiederholungen werden nicht zu einem Verdikt je Fall verdichtet. Eine Mehrheitsregel würde eine Schwelle einführen, die nachträglich gesetzt wäre, und die beobachtete Instabilität gerade wegmitteln.
Stattdessen werden drei schwellenfreie Klassen gebildet, nämlich einheitlich erfüllt bei allen fünf Wiederholungen mit erfülltem Indikator, einheitlich ausgeblieben bei keiner der fünf und uneinheitlich für alles dazwischen.
Je Zelle werden zusätzlich berichtet: die Zahl der Wiederholungen mit erfülltem Indikator, die Zahl verschiedener fachlicher Ergebnisse und die aufgetretenen Abschlusszustände.

Eine Zelle gilt als entscheidungsstabil, wenn alle fünf Wiederholungen in den fünf variablen der in E.7 genannten Kennzahlen übereinstimmen, also in allen außer der je Fall konstanten Zahl der erwarteten Änderungen.
Ein Lauf, der eine manuelle Prüfung verlangt, gilt nicht als Fehlschlag, sondern wird als eigener Zustand geführt. Die zugehörigen erwarteten Änderungen gelten als nicht adressiert und ihr Wertsurteil als nicht bestimmbar. Bereits zuvor im selben Lauf korrekt durchgeführte Änderungen bleiben davon unberührt.

E.13 Technisch unvollständige Läufe

Drei Sonderfälle sind vorab geregelt. In allen dreien gilt, dass die Läufe im Datensatz bleiben, keine Werte ergänzt werden und der Grund ausgewiesen wird.

Sonderfall	Umfang	Behandlung
kein Korrekturvorschlag erzeugt	9 Läufe, alle mit dem Zustand „manuelle Prüfung verlangt“, betroffen sind die kombinierten Fälle K06 und K07, verteilt auf 7 Läufe der Bedingung A sowie je einen in B und C	alle erwarteten Änderungen gelten als nicht adressiert, mit Grund
Ablaufzustand fehlt	4 Läufe der Bedingung C, sämtlich im Fall K07	der zuletzt abgelegte Vorschlag wird nach den Regeln E.3 und E.4 bewertet, die Läufe tragen ein eigenes Kennzeichen für den Verlust der Ablaufspur; ein vollständiger Abschluss der Verarbeitung wird für sie nicht behauptet
Belegartefakt fehlt	ein Lauf des Falls K06 in Bedingung A, fünfte Wiederholung, ab der zweiten Iteration	Korrektheit bleibt bewertbar, Belegbarkeit gilt als nicht bestimmbar nach E.6

E.14 Was ausdrücklich nicht bewertet wird

Die Referenz führt je Änderung nur Zielpfad, Vorzustand und Nachzustand. Sie enthält keine Angabe darüber, mit welcher Operation eine Korrektur durchzuführen wäre.
Für mehrere Fälle sind zwei Operationen gleichermaßen geeignet, etwa das Setzen eines Listenwerts und das Hinzufügen eines Listenelements. Die gewählte Operation wird deshalb nur beschreibend berichtet und nicht als Korrektheitskriterium geführt. Eine erwartete Operation wird nicht nachträglich aus Modellausgaben rekonstruiert.

E.15 Was ausdrücklich nicht eingeführt wurde

Fünf naheliegende Erleichterungen wurden geprüft und verworfen. Jede von ihnen hätte die gemessenen Werte verbessert, ohne die gemessene Eigenschaft zu verändern.

Eine numerische Toleranz beim Wertvergleich. Eine Ähnlichkeitsheuristik bei der Pfadzuordnung. Eine Gewichtung einzelner Änderungen oder Fälle. Eine Mehrheitsregel über die fünf Wiederholungen. Eine neue Halluzinationsdefinition nach Sichtung der Ergebnisse.

E.16 Geltungsbereich des Protokolls

Die Regeln wurden nach Abschluss der Datenerhebung festgelegt. Das ist eine Abweichung von einem vorregistrierten Vorgehen und wird als solche ausgewiesen. Zwei Umstände begrenzen das Risiko. Sie wurden festgelegt, bevor die nach Bedingungen aufgeschlüsselten Korrektheitsergebnisse eingesehen wurden, und sie wurden schriftlich fixiert, bevor sie angewendet wurden.

Eine Anpassung der Regeln an ein bereits bekanntes Ergebnis ist damit ausgeschlossen. Eine unbewusste Beeinflussung durch die vorangegangene Sichtung der Rohdaten lässt sich nicht vollständig ausschließen.

Die vor dem Einfrieren definierten Prüffunktionen für Wertvergleich, Belegbarkeit, strukturelle Prüfung, Folgefehler und Pfadidentität wurden zur Anwendung dieser Regeln unverändert übernommen.

Anhang F: Testfallkatalog und Referenzwerte

Der Katalog umfasst siebzehn Fälle mit zusammen 29 erwarteten Änderungen, davon zehn isolierte Fälle mit elf und sieben kombinierte Fälle mit achtzehn Änderungen. Der erwartete Wert entspricht dem Zustand vor der Fehlerinjektion.

Fall	Art	Fehlerart (Validator)	Änderungen
I01	Standard	unique_ids	1
I02	Standard	demand_article_ids	1

I03	Standard	density_values	1
I04	Standard	work_plan_ids	1
I05	Standard	article_department_presence	1
I06	Standard	article_equipment_department_consistency	1
I07	Standard	work_item_configs_completeness	1
I08	Standard	start_end_operation_existence	2
I09	Standard	work_item_equipment_availability	1
I10	Standard	packaging_references	1
K04	Komplex	unique_ids · demand_article_ids	2
K05	Komplex	density_values · work_plan_ids	2
K06	Komplex	packaging_references · unique_ids	2
K07	Komplex	equipment_predecessor_references · equipment_worker_qualification_compatibility	2
K08	Komplex	unique_ids · demand_article_ids · density_values	3
K09	Komplex	work_plan_ids · packaging_references · packaging_equipment_compatibility_references	3
K10	Komplex	unique_ids · equipment_predecessor_references · start_end_operation_existence	4
		Summe	**29**

Anhang G: Erhebungsbogen der Expertenerhebung

Der folgende Erhebungsbogen wurde den Teilnehmenden vollständig in dieser Form vorgelegt. Die realen Kennungen der vorbereiteten Datenbestände sind durch Platzhalter ersetzt.

- **Expertenbewertung – Vergleich der Systemvarianten**
- **Ziel der Bewertung**

Mit dieser Expertenbewertung soll untersucht werden, ob und in welcher Form Fachpersonen Unterschiede zwischen den getesteten Systemvarianten wahrnehmen.

Im Mittelpunkt stehen insbesondere:

- fachliche Qualität der Korrekturen
- Nachvollziehbarkeit der Entscheidungen
- Möglichkeit zur Fehlerrekonstruktion
- Qualität der bereitgestellten Informationen
- praktische Einsetzbarkeit

Für die Bewertung werden drei Konfigurationen betrachtet:

- **Variante 1 – Monolithische Referenz**

Workbook + prozedurale Verarbeitung

- **Variante 2 – Modulare Regelbereitstellung**

Regelkarten + prozedurale Verarbeitung

- **Variante 3 – Graphbasierte Variante**

Regelkarten + Graph-Verarbeitung

Damit ergeben sich drei relevante Vergleiche:

- **Variante 1 ↔ Variante 3:** Gesamtvergleich der monolithischen Referenz mit der graphbasierten Variante
- **Variante 1 ↔ Variante 2:** Einfluss der modularen Regelbereitstellung

- **Variante 2 ↔ Variante 3:** zusätzlicher Einfluss der Graph-Orchestrierung
- **Bewertungsskala**

Für die Vergleichsfragen gilt:

Wert	Bedeutung
1	deutlicher Vorteil der erstgenannten Variante
2	eher Vorteil der erstgenannten Variante
3	kein relevanter Unterschied
4	eher Vorteil der zweitgenannten Variante
5	deutlicher Vorteil der zweitgenannten Variante
–	Kann ich nicht beurteilen

Ein nicht wahrgenommener Unterschied ist ebenfalls ein relevantes Ergebnis.

---

2. Vorbereitete Snapshots – Satz A

Nr.	Name	Snapshot-ID
1	BA-EVAL A1	5b28fdd2-3c93-4b77-b059-036bd78f4ae6
2	BA-EVAL A2	17d31b99-631d-4047-b05d-bed08ebe80fb
3	BA-EVAL A3	9e16a7d7-71b8-4fa6-8195-eb4938c36e7c
4	BA-EVAL A4	f5a0ce39-a8d8-4e4f-b5d9-969eab5f2436
5	BA-EVAL A5	ac15c188-9c2a-41b3-bd94-2dbd4197d996
6	BA-EVAL A6	a58b22a8-8ba1-4789-8778-da24833cab43

---

3. Kurze Erklärung der Anwendung

- **Chat**

Im Chat kannst du das System auffordern, einen der vorbereiteten Snapshots zu analysieren und vorhandene Fehler zu korrigieren.

Beispielsweise:

Analysiere den vorbereiteten Snapshot und korrigiere die gefundenen Fehler.

- **Einstellungen**

Oben rechts im Chat befindet sich ein Zahnrad. Dort können zwei für die Untersuchung zentrale Eigenschaften der Verarbeitung eingestellt werden.

- **Regelbereitstellung**
- **Workbook:** Es wird ein zusammenhängendes Regelwerk verwendet.
- **Regelkarten:** Abhängig vom erkannten Fehler werden relevante Regelbereiche ausgewählt.
- **Orchestrierung**
- **Prozedural:** Der bestehende Verarbeitungsablauf wird verwendet.
- **Graph:** Der Korrekturprozess wird über explizite Verarbeitungsschritte und Zustände orchestriert.

Die beiden Einstellungen können unabhängig voneinander verändert werden.

Du musst die technischen Unterschiede nicht im Detail kennen. Entscheidend ist, die Varianten auszuprobieren und darauf zu achten, ob Unterschiede wahrnehmbar sind.

- **Review Board**

Im Review Board findest du die erzeugten Korrekturvorschläge. Dort kannst du unter anderem sehen:

- welche Änderung vorgeschlagen wird
- den vorherigen Wert
- den vorgeschlagenen neuen Wert
- die Begründung des Systems

Anschließend kannst du:

- einen Vorschlag genehmigen und anwenden
- einen Vorschlag ablehnen
- einen vorgeschlagenen Wert ändern
- **Dashboard**

Im Dashboard werden zusätzliche Statusinformationen und Kennzahlen dargestellt.

Öffne das Dashboard zumindest einmal und verschaffe dir einen kurzen Überblick.

- **Bewertungsfragen**
- **Frage 1 – Fachliche Qualität**
- **Variante 1 ↔ Variante 3**

Bei welcher Variante hattest du insgesamt den besseren Eindruck von der fachlichen Qualität der erzeugten Korrekturen?

Bewertung: **1 – 2 – 3 – 4 – 5 – Kann ich nicht beurteilen**

Optional: Woran hast du deine Einschätzung festgemacht?

- **Frage 2 – Nachvollziehbarkeit**
- **Variante 1 ↔ Variante 3**

Bei welcher Variante konntest du besser nachvollziehen, wie und warum eine Korrektur zustande gekommen ist?

Bewertung: **1 – 2 – 3 – 4 – 5 – Kann ich nicht beurteilen**

Optional: Welche Informationen haben dabei besonders geholfen oder gefehlt?

- **Frage 3 – Fehlerrekonstruktion**
- **Variante 1 ↔ Variante 3**

Bei welcher Variante konntest du bei einer unklaren oder problematischen Korrektur besser rekonstruieren, an welcher Stelle im Verarbeitungsablauf die Ursache liegen könnte?

Bewertung: **1 – 2 – 3 – 4 – 5 – Kann ich nicht beurteilen**

Optional: Was hat dir bei der Rekonstruktion geholfen?

- **Frage 4 – Informationsbereitstellung**
- **Variante 1 ↔ Variante 3**

Bei welcher Variante wurden die Informationen, die du für die Beurteilung einer Korrektur benötigst, hilfreicher bereitgestellt?

Bewertung: **1 – 2 – 3 – 4 – 5 – Kann ich nicht beurteilen**

Optional: Welche Informationen waren besonders wichtig oder haben gefehlt?

- **Frage 5 – Einfluss der Regelbereitstellung**
- **Variante 1 ↔ Variante 2**

Workbook ↔ Regelkarten

Welchen Unterschied hast du zwischen Workbook und Regelkarten wahrgenommen, falls überhaupt?

Wert	Bedeutung
1	deutlicher Vorteil Workbook
2	eher Vorteil Workbook
3	kein relevanter Unterschied
4	eher Vorteil Regelkarten
5	deutlicher Vorteil Regelkarten
–	Kann ich nicht beurteilen

Worin bestand dieser Unterschied für dich?

- **Frage 6 – Einfluss der Graph-Orchestrierung**
- **Variante 2 ↔ Variante 3**

Prozedural ↔ Graph

Welchen Unterschied hast du zwischen prozeduraler und graphbasierter Verarbeitung wahrgenommen, falls überhaupt?

Wert Bedeutung

1 deutlicher Vorteil Prozedural

2 eher Vorteil Prozedural

3 kein relevanter Unterschied

Wert Bedeutung

4 eher Vorteil Graph

5 deutlicher Vorteil Graph

– Kann ich nicht beurteilen

Worin bestand dieser Unterschied für dich?

- **Frage 7 – Praktischer Einsatz**

Welche der drei getesteten Varianten würdest du für einen praktischen Einsatz bevorzugen?

- Variante 1 – Workbook + Prozedural
- Variante 2 – Regelkarten + Prozedural
- Variante 3 – Regelkarten + Graph
- Keine klare Präferenz
- Kann ich nicht beurteilen

Unter welchen Voraussetzungen würdest du das System tatsächlich einsetzen?

- **Frage 8 – Gesamtbeurteilung und Verbesserung**

Was ist aus deiner Sicht der wichtigste Unterschied zwischen den getesteten Varianten und was sollte am System als Erstes verbessert werden?

Anhang H: Transkript und schriftliche Zusammenfassung

Von einer der beiden Erhebungen liegt ein Transkript vor, das mit Einwilligung der teilnehmenden Person wiedergegeben wird. Das Transkript wurde automatisiert durch das eingesetzte Konferenzwerkzeug Microsoft Teams erzeugt und anschließend von mir gegen die Aufzeichnung geprüft. Die zweite Erhebung wurde nicht aufgezeichnet und im Anschluss schriftlich zusammengefasst. E1 und E2 bezeichnen die teilnehmenden Personen, I die interviewende Person. Namen, Funktionsbezeichnungen und weitere identifizierende Angaben wurden entfernt oder ersetzt.

Entfernt wurden Passagen ohne Bezug zum Untersuchungsgegenstand sowie Abschnitte mit firmeninternen, kundenbezogenen oder anderweitig vertraulichen Projektinformationen, deren Wiedergabe nicht gestattet ist. Jede Entfernung ist durch eine Auslassungsmarkierung gekennzeichnet und enthält den betroffenen Zeitraum sowie den Grund der Entfernung. Die Zeitangaben beziehen sich auf den Zeitstempel der Aufzeichnung.

Sprachliche Eigenheiten der gesprochenen Sprache, Wiederholungen und Verständigungslaute wurden unverändert übernommen. Erkennbare Fehler der automatischen Transkription sind in eckigen Klammern korrigiert.

I.1 Transkript der Erhebung mit E1

Datum: 27.08.2026

Dauer der Aufzeichnung: 28 Minuten

Form: Videokonferenz mit Bildschirmfreigabe, automatische Transkription
Einwilligung: liegt vor

00:03 E1: Echt schwer, gerade irgendwie eine wirkliche Aussage treffen, was jetzt wirklich besser ist oder nicht besser ist.

00:14 I: Ja, deshalb waren die Qualitätsergebnisse aus den 255 Messungen, die ich gemacht hab, wirklich mit, also wenn man das Ganze auf einer Skala von 0 bis 100 betrachtet, das waren 2 Prozent eigentlich Unterschiede, das ist kaum bei zweihundertfünfundfünfzig.

00:37 E1: Ich kann jetzt noch einmal sagen, ich habe jetzt da eben noch mal einen anderen Fehler eingebaut bei unserem Snapshot. Mal schauen, aber du musst halt da immer wieder eigentlich sagen, also wenn du es wirklich vergleichen willst, müsstest du ja den gleichen Fehler in mehreren Snapshots einbauen in Wahrheit und dann mit unterschiedlichen Settings diesen Fehler testen und dir dann anschauen, was hat er denn für ein Ergebnis gefunden.

01:08 I: Das Problem, was noch dazu kommt, worauf ich jetzt im Nachgang draufgekommen bin, es gibt ja diese Memory und diese Schleife, um das System sich zu verbessern. Das trägt dazu bei, dass die Antworten einfach besser werden mit der Zeit.

01:24 I: Egal ob Regelkarten oder nicht, ja.

01:24 E1: Genau, und das müsstest du auch ausschalten eigentlich, ja.

01:29 I: Den müsste ich ausschalten, gegebenenfalls.

[Auslassung: Bearbeitung eines Korrekturvorschlags am Bildschirm, 01:35 bis 02:47]

02:47 I: Aber das ist spannend. Wir haben immer gesagt, dass die Warnings eigentlich keine Rolle spielen. Aber können die Warnings dazu beitragen, dass ein Error besser verstanden wird, wenn man diese richtig interpretiert?

03:03 E1: Na, glaub ich fast nicht. Sind unabhängige Prüfungen eigentlich. Also vielleicht in der einen oder anderen Konstellation, aber eher nicht, würde ich sagen.

03:17 E1: So, was hat er da jetzt gemacht? Ich hab ihm das rausgelöscht. Dann müsste er eigentlich [den Wert] haben, und dann hat er jetzt gesagt, er will das hinzufügen. Mensch hat genau diesen Wert schon mal für dieses Objekt bereits bestätigt. Aha, okay, aber das ist natürlich easy, wenn man das natürlich schon einmal gemacht hat.

[Auslassung: technische Bedienung, 03:40 bis 09:06]

09:39 E1: Aber ja, sonst finde ich die Bedienung eigentlich ganz cool, grundsätzlich.

10:02 E1: Für exakt einen Fehler wurde ein Korrekturvorschlag erstellt. Ein weiterer Fehler bleibt offen. Okay, das hast du so gesagt, oder weiß er nicht wie.

10:31 I: Er macht immer eins nach dem anderen.

[Auslassung: Anmerkungen zur Sortierung und Benennung in der Oberfläche, 10:33 bis 11:47]

11:49 I: Das System hat echt standgehalten bei den 255 Snapshots. Ich hab gedacht, ich schieße die VM damit, aber es hat echt gut funktioniert.

12:01 E1: Also da muss man sich keine Sorgen machen, das muss schon funktionieren.

[Auslassung: Anmerkungen zum Löschen von Snapshots, 12:09 bis 12:57]

13:03 I: Zu den Fehlern, die aufgepoppt sind. Würdest du behaupten, dass die Fehler, die du jetzt betrachtet hast, zumindest bei den einigen Snapshots, wirklich nahe an der Realität sind, oder so weit davon entfernt, dass in der Realität völlig andere Fehler aufpoppen oder völlig andere Herausforderungen, die die Facharbeiter haben?

13:22 E1: Na, das sind schon, also die Fehler, die ich jetzt gesehen hab, das sind schon reale Fehler. Also das kann schon passieren. Ich hab jetzt eben den letzten, den ich bei dem Snapshot absichtlich eingebaut hab, das war eben, dass da Disponent 2 fehlt, also diese Abteilung, oder wie es heißt im Snapshot.

13:41 E1: Das ist jetzt auch in der Realität ein paar Mal aufgetreten. Aber das sind schon alles grundsätzlich reale Fehler. Du hast zum Beispiel irgendwie die Work Item Keys, die fehlen, oder so. Das kann schon passieren, genau so.

14:03 E1: Genau, Department-ID und Department-Name, das hab ich jetzt mal rausgelöscht gehabt, das war eben etwas, was realistisch öfter mal fehlt. Ein Work Item Key, der mal fehlt, ja genau.

[Auslassung: organisatorische Themen ohne Bezug zum Untersuchungsgegenstand, 14:10 bis 24:30]

24:48 I: Danke fürs Testen. Ich werde das so auswerten.

25:02 I: So werde ich das Ganze begrenzen, und ich glaube, das ist eigentlich sehr wohl ein valides Resultat.

[Auslassung: Verabschiedung und organisatorische Absprachen, 25:25 bis 27:56]

Ende der Aufzeichnung.

I.2 Schriftliche Zusammenfassung der Erhebung mit E2

Datum der Erhebung: 31.08.2026

Datum der Zusammenfassung: 31.08.2026

Form: Gespräch ohne Aufzeichnung, im Anschluss schriftlich zusammengefasst
Status: keine wörtliche Wiedergabe, Zusammenfassung durch die interviewende Person

Rahmen. E2 nutzte das System innerhalb eines Zeitraums von mehreren Tagen. Die aktive Nutzung umfasste nach eigener Einschätzung mehrere Stunden. Dabei wurden vorbereitete Datenbestände mit unterschiedlichen Systemkonfigurationen bearbeitet. Das Gespräch fand im Anschluss an die Nutzung statt.

Abgegebene Urteile. E2 gab auf Grundlage der Erprobung keine Präferenz zwischen den untersuchten Systemvarianten an. Die Unterschiede zwischen den

Varianten wurden innerhalb des verfügbaren Testzeitraums nicht als ausreichend deutlich wahrgenommen, um daraus eine fundierte vergleichende Bewertung abzuleiten. Entsprechend wurden zu den Vergleichsfragen überwiegend keine eindeutigen Urteile abgegeben.

Begründung für nicht abgegebene Urteile. Als wesentlicher Grund wurde die begrenzte Nutzungsdauer genannt. Nach Einschätzung von E2 reicht eine mehrstündige Erprobung innerhalb weniger Tage nicht aus, um Unterschiede zwischen den Varianten zuverlässig zu beurteilen. Insbesondere Eigenschaften wie Nachvollziehbarkeit, Fehlerrekonstruktion und der praktische Nutzen zusätzlicher Systeminformationen würden sich erst bei wiederholter Nutzung und bei unterschiedlichen Fehlerfällen deutlicher zeigen.

Weitere Anmerkungen. E2 bewertete die betrachteten Fehlerfälle grundsätzlich als für den Anwendungskontext relevant. Für einen aussagekräftigeren Vergleich wäre aus Sicht von E2 eine längere Erprobung unter wiederkehrenden Arbeitsabläufen sinnvoll.

Hinweis zur Belastbarkeit. Diese Zusammenfassung wurde nach dem Gespräch ohne vorhandene Audio- oder Videoaufzeichnung erstellt. Sie gibt ausschließlich Inhalte wieder, die nachträglich sicher rekonstruiert werden konnten. Die Formulierungen sind sinngemäß und stellen keine wörtlichen Aussagen von E2 dar. Unsichere oder nicht eindeutig erinnerbare Inhalte wurden nicht aufgenommen.

Anhang I: Nachweis zum Einsatz generativer Künstlicher Intelligenz

In der eidesstattlichen Erklärung ist angegeben, dass im Zuge dieser Arbeit Werkzeuge der generativen Künstlichen Intelligenz eingesetzt wurden. Dieser Abschnitt dokumentiert Art und Umfang der Nutzung, das jeweils angewandte Vorgehen und die Abgrenzung zwischen KI-gestützter Unterstützung und eigener wissenschaftlicher beziehungsweise technischer Leistung.

Vorgehen

Für alle Einsatzformen galt dasselbe Vorgehen. Ich habe den Inhalt vorgegeben, das erwartete Ergebnis vorab festgelegt und jedes Ergebnis gegen ein zuvor bestimmtes Kriterium geprüft. Erfüllte ein Ergebnis das Kriterium nicht, wurde es überarbeitet oder verworfen und nicht übernommen. Die Verantwortung für den übernommenen Inhalt liegt in jedem Fall bei mir.

Erstellung von Abbildungen

Die Abbildungen 1, 2, 3, 5, 9 und 10 wurden unter Einsatz generativer Künstlicher Intelligenz grafisch aufbereitet. Eingesetzt wurde dafür Claude Code Opus 5.

Als Eingabe diente jeweils eine von mir erstellte inhaltliche Spezifikation. Sie enthielt die darzustellenden Komponenten, Knoten, Ein- und Ausgaben, Beziehungen, Übergänge und Beschriftungen, abgeleitet aus der implementierten Architektur beziehungsweise den in Kapitel 6 festgelegten Messvorschriften.

Bezeichnungen wurden aus dem Quellcode übernommen und nicht neu formuliert. Erwartet wurde eine Darstellung, die genau diese Elemente und keine weiteren zeigt.

Die Abnahme erfolgte gegen den implementierten Stand. Geprüft wurde, ob jeder dargestellte Knoten und jede Kante im Code existiert, ob die Beschriftungen den dort verwendeten Bezeichnern entsprechen und ob keine Elemente ergänzt wurden, die im System kein Gegenstück haben. Für Abbildung 10 galt zusätzlich, dass die dargestellten Werte den in den Tabellen 15 und 16 berichteten Zahlen entsprechen müssen. Abweichungen führten zur Überarbeitung; an mehreren Stellen waren dafür mehrere Durchgänge nötig. Das Werkzeug hat damit weder die Architektur noch die fachlichen Zusammenhänge bestimmt, sondern bereits festgelegte Inhalte visuell umgesetzt. Auswahl der dargestellten Elemente und Freigabe lagen bei mir.

Ohne Einsatz generativer Künstlicher Intelligenz entstanden die Abbildungen 4, 6, 7 und 8. Sie zeigen Auszüge aus Regelwerk, Laufartefakten und Benutzeroberfläche des untersuchten Systems.

KI-gestützte Softwareentwicklung (AI-Assisted Coding)

Generative KI wurde im Rahmen der Softwareentwicklung eingesetzt, insbesondere bei Implementierung, Fehlersuche, Refactoring und Überarbeitung einzelner Codeabschnitte. Eingesetzt wurde dafür Claude Opus 5 & Claude Fable. Der Einsatz erfolgte als AI-assisted Coding und ist für ein Vorhaben dieser Art kein Zusatz, sondern die gewählte Entwicklungsweise.

Vorgegeben habe ich Anforderungen, Systemarchitektur, Schnittstellen, Zustandsmodell, Vergleichsbedingungen und Messlogik. Eingabe war jeweils die zu erfüllende Anforderung samt bestehender Codebasis, erwartet wurde ein Vorschlag, der sich in die vorhandene Struktur einfügt, ohne deren Aufbau zu verändern.

Kein Vorschlag wurde ungeprüft übernommen. Als Abnahmekriterium galt die tatsächliche Ausführung. Ein Abschnitt wurde erst übernommen, wenn er die für das Projekt vorhandenen Tests bestand und im Zielsystem das erwartete Verhalten

zeigte. Änderungen am Zustandsmodell oder an der Messlogik habe ich zusätzlich manuell gegen die in Kapitel 5 festgelegten Kontrollvariablen geprüft, weil dort jede unbeabsichtigte Abweichung die Vergleichbarkeit der Bedingungen berührt hätte. Der für die Hauptmessung verwendete Programmstand wurde vor Beginn der Erhebung eingefroren und versioniert - die zugehörigen Commit-Kennungen und Prüfsummen sind in Kapitel 5 ausgewiesen.

Davon zu unterscheiden sind die Modellaufrufe innerhalb des untersuchten Systems selbst. Die Aufrufe auf Basis von Azure OpenAI GPT-4.1 sind Gegenstand der Untersuchung und Bestandteil der verglichenen Verarbeitungspfade. Sie stellen keine Unterstützung beim Verfassen dieser Arbeit dar.

Sprachliche und redaktionelle Unterstützung

Generative KI wurde bei der Überarbeitung selbst verfasster Textteile eingesetzt, konkret für Hinweise zu Grammatik, Satzbau, Verständlichkeit, sprachlicher Präzision und zur Reduktion von Wiederholungen. Eingesetzt wurde dafür Claude Opus 5.

Eingabe war jeweils ein von mir verfasster Abschnitt. Erwartet wurden Verbesserungsvorschläge auf sprachlicher Ebene, ausdrücklich keine inhaltlichen Ergänzungen. Übernommen habe ich einen Vorschlag nur dann, wenn die Aussage des Abschnitts unverändert blieb. Vorschläge, die eine Aussage verschärften, abschwächten oder Zahlen, Bezugsgrößen und Einschränkungen berührten, wurden verworfen. Forschungsdesign, Interpretation der Messergebnisse und Schlussfolgerungen stammen von mir und wurden nicht generiert.

Formale Prüfung und Literaturarbeit

Ergänzend wurden KI-gestützte Werkzeuge zur Suche nach formalen Auffälligkeiten eingesetzt, etwa nach inkonsistenten Zitierweisen, Abweichungen vom verwendeten Zitierstil, doppelten Textstellen und fehlerhaften Querverweisen. Diese Werkzeuge liefern Hinweise, keine Entscheidungen. Jeder Hinweis wurde gegen die zugrunde liegende Quelle beziehungsweise die bibliografischen Angaben geprüft und erst danach übernommen oder verworfen. Generative KI wurde nicht als wissenschaftliche Quelle verwendet und hat keinen Eintrag im Literaturverzeichnis erzeugt.

Grundsatz der Verwendung

Generative KI wurde in dieser Arbeit als Werkzeug verwendet, dessen Ergebnisse jeweils gegen ein vorab festgelegtes Kriterium geprüft wurden. Die Verantwortung für Forschungsfrage, methodische Entscheidungen, Implementierung, Datenerhebung, Auswertung, Interpretation, Auswahl der wissenschaftlichen Quellen und die finale Fassung der Arbeit liegt bei mir.