Integrations- und Prototyping-Strategie für ein Modulares, KI-Agentenbasiertes System

TL;DR (Too Long; Didn't Read)

Die optimale Architektur zur Realisierung der Vision des Knowledge Integration Protocol (KIP) ist ein Hybridsystem, das auf **LangGraph** aufbaut. Dessen explizite, zustandsbehaftete Orchestrierungsfähigkeiten sind für kontrollierbare und nachvollziehbare Agenteninteraktionen unerlässlich. Das KIP-Prinzip eines persistenten, graphbasierten Gedächtnisses wird durch einen **Neo4j Knowledge Graph** implementiert, auf den über das **GraphRAG**-Muster zugegriffen wird. Dies transformiert das Gedächtnis von einem passiven Datenspeicher zu einer aktiven Komponente für schlussfolgerndes Denken. Die System-Governance und die Ausrichtung an den KIP-Prinzipien werden durch ein dynamisches **Constitutional AI** Framework sichergestellt. Innerhalb dieses Rahmens ist ein "System-Gärtner"-Agent für die Überwachung und das Vorschlagen von Aktualisierungen der Verfassung selbst verantwortlich, wodurch das System an neue Ziele aus dem Backlog.md angepasst werden kann. Die empfohlene Prototyping-Roadmap konzentriert sich auf eine User-Story aus dem Backlog.md, die einen vollständigen Entwicklungszyklus abbildet (z. B. "Implementierung der Benutzerauthentifizierung"). Diese Wahl demonstriert eindrucksvoll die End-to-End-Fähigkeiten des Systems – von der Anforderungsanalyse durch einen "Product Manager"-Agenten bis hin zur Codegenerierung, Überprüfung und zum Testen durch ein Team von spezialisierten Entwickler-Agenten, die alle den Knowledge Graph nutzen, um den Kontext zu wahren und aus dem Prozess zu lernen.

Teil I: Der Interdisziplinäre Entwurf: Erkenntnisse aus der Grundlagenforschung

Dieser Abschnitt legt das fundamentale "Warum" für die technischen Entscheidungen in Teil II dar. Er übersetzt die philosophische Vision des Projekts in etablierte wissenschaftliche und methodische Konzepte und schafft so einen robusten intellektuellen Rahmen.

Das System als Geist: Kognitionswissenschaft als Architektonische Metapher

Das im KIP geforderte "persistente, graphbasierte Gedächtnis" sollte nicht als bloßer Datenspeicher, sondern als eine komputationelle kognitive Architektur konzipiert werden, die das menschliche Gedächtnis widerspiegelt. Die Kognitionswissenschaft lehrt uns, dass das menschliche Gedächtnis kein monolithischer Datenspeicher ist, sondern ein mehrstufiges System: Es besteht aus einem flüchtigen **sensorischen Speicher**, einem kapazitätsbeschränkten **Kurzzeitgedächtnis** (Arbeitsgedächtnis) und einem riesigen, assoziativen **Langzeitgedächtnis**.[1] Informationen werden aktiv zwischen diesen Speichern verarbeitet und transferiert.

Das von Baddeley und Hitch entwickelte **Arbeitsgedächtnismodell** beschreibt das Kurzzeitgedächtnis als dynamisches System, in dem die bewusste Informationsverarbeitung und Problemlösung stattfindet – die aktive "Werkbank" der Kognition.[1] Um die begrenzte Kapazität zu bewältigen, nutzt das Gehirn eine Technik namens

Chunking, bei der verwandte Informationen zu einzelnen Einheiten zusammengefasst werden.[2] Das Langzeitgedächtnis wiederum wird durch wiederholten Abruf gestärkt, was zu physischen Veränderungen der synaptischen Verbindungen führt und ein dichtes Informationsnetzwerk bildet. Neues Wissen wird leichter integriert, wenn es an bestehendes "Vorwissen" anknüpfen kann.[2]

Diese Modelle führen direkt zu der Notwendigkeit einer dualen Gedächtnisarchitektur für das Agentensystem. Es benötigt zwei unterschiedliche Speichertypen: ein schnelles, ephemeres "Arbeitsgedächtnis" für aktive Aufgaben und ein persistentes, assoziatives "Langzeitgedächtnis" zur Speicherung von Wissen. Die technische Umsetzung dieser Trennung ist naheliegend: Das zentrale Zustandsobjekt innerhalb eines Frameworks wie LangGraph dient als Arbeitsgedächtnis, das den unmittelbaren Kontext, Zwischenergebnisse und die aktive Kommunikation einer Aufgabe bereithält.

Das im KIP beschriebene "persistente, graphbasierte Gedächtnis" entspricht perfekt dem Konzept eines assoziativen Langzeitgedächtnisses und sollte als dedizierter Knowledge Graph (z. B. mit Neo4j) realisiert werden.

Darüber hinaus bietet das Konzept des **kollektiven Gedächtnisses** eine wirkungsvolle Metapher für die Rolle des "System-Gärtners". Das kollektive Gedächtnis ist nicht nur eine Ansammlung individueller Erinnerungen, sondern ein konstruiertes, intentionales und symbolisches System, das eine gemeinsame Identität und Erzählung für eine Gruppe (z. B. eine Nation, ein Unternehmen) schafft.[3] Es wird von den führenden Kräften der Gruppe gepflegt und dient der Erreichung institutioneller Ziele. Dies impliziert, dass der "System-Gärtner" nicht nur ein passiver Hausmeister ist, sondern der aktive Kurator des kollektiven Wissens und der Identität des Systems. Seine Hauptfunktion ist die Verwaltung des Langzeitgedächtnisses (des Knowledge Graphen). Dies umfasst das Bereinigen veralteter Informationen, das Verstärken wichtiger Verbindungen (analog zur synaptischen Stärkung), das Identifizieren von Wissenslücken und die Sicherstellung, dass das "kollektive Gedächtnis" mit den übergeordneten Zielen des Systems, die in seiner "Verfassung" festgelegt sind, übereinstimmt.

Das System als Organismus: Prinzipien der Selbstorganisation

Das Agentensystem sollte keine starre, von oben nach unten gerichtete Befehlsstruktur sein, sondern ein selbstorganisierendes System, das vom "System-Gärtner" geleitet und nicht mikrogemanagt wird. Die Systemtheorie beschreibt selbstorganisierende Systeme durch Merkmale wie **Selbstreferenz** und **operationale Geschlossenheit**: Sie handeln auf der Grundlage ihres internen Zustands und ihrer Logik, nicht nur aufgrund direkter externer Befehle, und ihr Verhalten wirkt zirkulär auf ihr eigenes zukünftiges Verhalten zurück.[5]

Ordnung entsteht in solchen Systemen durch **Emergenz** aus den Interaktionen der Systemkomponenten, anstatt von außen aufgezwungen zu werden.[7] Externe Ereignisse liefern den "Anlass" (Trigger), aber die Struktur bildet sich von innen heraus. Diese Systeme neigen dazu, sich in

Attraktoren einzupendeln – stabile Zustände oder Verhaltensmuster, zu denen sie nach Störungen zurückkehren, was eine Form von "Selbstheilungskraft" darstellt.[7] Ein

weiteres Schlüsselprinzip ist die

Anschlussfähigkeit, bei der Operationen so strukturiert sind, dass nachfolgende Operationen nahtlos anknüpfen können, was die Kontinuität des Systems sichert.[7]

Diese Prinzipien haben direkte Auswirkungen auf das Design des Orchestrators. Seine Hauptaufgabe sollte es sein, die Emergenz von Lösungen zu fördern, anstatt sie zu diktieren. Ein starrer, schrittweiser Plan würde das System spröde machen. Stattdessen definiert der Orchestrator den "Problemraum" und die "Spielregeln" (Kommunikationsprotokolle, Ziele). Der tatsächliche Lösungsweg – die Abfolge der Agentenaktionen – sollte eine emergente Eigenschaft der Zusammenarbeit des Agententeams sein, die durch Feedbackschleifen gesteuert wird.[8]

Die Rolle des "System-Gärtners" lässt sich als die eines Verwalters der System-Attraktoren verstehen. Anstatt in jede Aufgabe einzugreifen, beobachtet der Gärtner das Gesamtverhalten des Systems und "stupst" es sanft in Richtung wünschenswerter Attraktoren (z. B. effiziente Problemlösung, qualitativ hochwertiger Code) und weg von unerwünschten (z. B. sich wiederholende Schleifen, Agentenkonflikte). Seine Eingriffe wären systemisch, nicht aufgabenspezifisch. Beispielsweise könnte er die Belohnungsfunktion in der Systemverfassung anpassen, den Prompt eines leistungsschwachen Agenten modifizieren oder eine Wartungsroutine für den Knowledge Graphen auslösen, um fehlende Informationen zu ergänzen, die wiederholt zu Fehlern führen.

Das System als Hochleistungsorganisation: Lehren aus der Organisationspsychologie

Die Interaktionsprotokolle für Agententeams sollten auf bewährten Modellen der Zusammenarbeit von menschlichen Spezialisten basieren. Die Organisationspsychologie zeigt, dass die Grundlage effektiver Teams **Vertrauen** ist. Teammitglieder müssen sich sicher fühlen, um Hilfe zu bitten und ihre Meinung zu äußern.[9] Effektive Teams praktizieren konstruktive

Konfliktbereitschaft, anstatt Meinungsverschiedenheiten zu vermeiden.[9] Ein gemeinsames Identitätsgefühl ("Wir-Gefühl") führt zu besserer Kommunikation, mehr Wissensaustausch und höherer Leistung.[10]

Diese Erkenntnisse müssen direkt in das Design der Agenten einfließen. Die

System-Prompts für Spezialagenten müssen nicht nur deren Fähigkeiten, sondern auch ihre kollaborative Haltung kodifizieren. "Vertrauen" lässt sich in einem Agentensystem als die Bereitschaft übersetzen, Feedback anzunehmen und um Unterstützung zu bitten. Daher müssen die Prompts Anweisungen enthalten wie: "Wenn du unsicher bist, frage den 'Architekt'-Agenten um Klärung" oder "Gib bei der Überprüfung von Code des 'Entwickler'-Agenten konstruktive Kritik, die auf die Verbesserung der Lösung abzielt, nicht nur auf das Aufzeigen von Fehlern." Dies programmiert die Prinzipien von Vertrauen und Konfliktbereitschaft direkt in das Verhalten der Agenten.[9]

Der Orchestrator übernimmt die Rolle eines Teamleiters, der eine positive Teamkultur fördert. So wie ein menschlicher Leiter ein "Wir-Gefühl" schafft [10], sollte der "System Prompt" des Orchestrators (Teil der Verfassung) Meta-Anweisungen für das gesamte Team enthalten: "Ihr seid ein Team von Experten-KI-Agenten, die zusammenarbeiten, um hochwertige Software zu entwickeln. Euer oberstes Ziel ist der kollektive Erfolg. Unterstützt eure Teamkollegen, teilt Informationen frei und arbeitet gemeinsam an der Lösung von Problemen." Dies etabliert den kollaborativen Kontext, bevor die eigentliche Aufgabe beginnt.

Das System als zweites Gehirn: Die Zettelkasten-Methode als konkretes Gedächtnismodell

Die Zettelkasten-Methode von Niklas Luhmann bietet eine praxiserprobte Blaupause für die Implementierung des "persistenten, graphbasierten Gedächtnisses" des KIP. Ihr Kernprinzip ist die **Atomizität**: eine Idee pro Notiz (Zettel), was Wissen modular und wiederverwendbar macht.[11] Die Stärke des Systems liegt in den

Verknüpfungen zwischen den Notizen, nicht in einer starren Ordnerhierarchie; es ist ein Netzwerk, kein Baum.[12] Das Wissen wird

von unten nach oben aufgebaut, wobei Muster und übergeordnete Strukturen (Struktur-Notizen) aus dem Netzwerk der atomaren Notizen emerieren.[11]

Dieser Ansatz führt zu einer klaren Vorgabe für das Schema des Knowledge Graphen: Die Knoten im Neo4j-Graphen sollten "Zettel" sein. Jeder Knoten sollte ein einzelnes, atomares Wissenselement repräsentieren (z. B. ein bestimmtes Programmiermuster, eine Benutzeranforderung, eine Entscheidung aus einer früheren Aufgabe). Die

Kanten im Graphen sind die expliziten Verknüpfungen zwischen diesen Zetteln. Dies schafft eine direkte Abbildung: Graph-Knoten = Zettel, Graph-Kante = Verknüpfung.

Ein entscheidender Punkt ist, dass die Wissensintegration ein aktiver, von Agenten gesteuerter Prozess sein muss. Das System darf nicht einfach Rohdaten in den Graphen abladen. Die Zettelkasten-Methode erfordert, dass Informationen verarbeitet und in eigenen Worten neu formuliert werden, um das Verständnis sicherzustellen; das bloße Sammeln von Zitaten ist ein Trugschluss ("Collector's Fallacy").[12] Dies impliziert, dass nach Abschluss einer Aufgabe ein dedizierter "Reflektor"-Agent für die "Reflexion" verantwortlich sein muss. Dieser Agent analysiert das Aufgabenprotokoll (das "Arbeitsgedächtnis"), extrahiert die wichtigsten Erkenntnisse und Entscheidungen, formuliert sie als atomare "Zettel" und fragt dann den bestehenden Knowledge Graphen ab, um relevante Knoten für die Verknüpfung zu finden. Dieser aktive Prozess der Zusammenfassung, Synthese und Verknüpfung ist der Weg, wie das System wirklich

lernt und neues Wissen integriert, was verhindert, dass der Graph zu einer unübersichtlichen Datenhalde wird.

Teil II: Detaillierte Strategische Analyse und Empfehlungen

Dieser Hauptteil des Berichts nutzt die im ersten Teil entwickelten grundlegenden Konzepte, um die spezifischen Forschungsfragen mit konkreten technischen Empfehlungen zu beantworten.

Die Architektonische Synthese: Philosophie in Code weben

Die optimale Architektur ist ein Hybridsystem, das verschiedene Muster kombiniert: Es ist ein **zustandsbasiertes, graph-basiertes System** für die Orchestrierung, bei dem jeder Knoten als **Microservice** (ein Spezialagent) betrachtet werden kann. Die Kommunikation innerhalb des Graphen wird durch das Zustandsobjekt verwaltet, während das System über **ereignisgesteuerte** Prinzipien mit der Außenwelt interagiert.

Framework-Vergleich und Empfehlung

Eine vergleichende Analyse der führenden Multi-Agenten-Frameworks zeigt deutliche Unterschiede in ihrer Eignung für dieses Projekt. Die Bewertung erfolgt anhand von Kriterien, die sich direkt aus den Anforderungen des KIP ableiten, wie Kontrollierbarkeit, Beobachtbarkeit und eine explizite Prozessdarstellung.

Merkmal	LangGraph	AutoGen	CrewAI	KIP-Ausrichtung	Begründung
Orchestrierungsmodell	Graph-basierter Zustandsautomat [14]	Konversationell / Ereignisgesteuert [16]	Rollenbasiert, sequenziell/hierarchisch [14]	Hoch	Ermöglicht die explizite Modellierung und Steuerung komplexer, nicht-linearer Prozesse, was für die KIP-Vision unerlässlich ist.
Zustandsmanagement	Zentrales, veränderbares Zustandsobjekt [15]	Agenten-spezifische Speicher / Nachrichten [19]	Strukturiert, rollenbasiert [20]	Hoch	Ein zentraler Zustand ist die direkte technische Umsetzung des Konzepts eines "Arbeitsgedächtnisses" und entscheidend für die Beobachtbarkeit.

Kontrolle & Beobachtbarkeit	Hoch (expliziter Graph, Zustandsverfolgung) [14]	Mittel (Protokolle) [14]	Niedrig (abstrahiert) [14]	Hoch	Die Fähigkeit, den genauen Zustand und Pfad des Systems jederzeit zu inspizieren, ist für die Rolle des "System-Gärtners" fundamental.
Flexibilität & Anpassung	Hoch (Low-Level-Kontrolle) [21]	Hoch (benutzerdefinierte Agenten/Tools) [14]	Mittel (strukturierte Rollen) [14]	Hoch	Bietet die größte Flexibilität, um die spezifischen KIP-Prozesse (z.B. Reflexion, konstitutionelle Prüfungen) als Knoten zu implementieren.
Schwäche / Kritik	Steilere Lernkurve, vordefinierte Pfade [23]	Intransparente "Magie", weniger Kontrolle [22]	Weniger flexibel für komplexe, nicht-lineare Aufgaben [14]	Gering	Die "vordefinierten Pfade" sind für dieses Projekt ein Vorteil, da sie eine bewusste Gestaltung der Systemprozesse ermöglichen.

Empfehlung: LangGraph ist für dieses Projekt eindeutig am besten geeignet.

Während AutoGen bei dynamischen Konversationen [16] und CrewAI bei klar definierten, hierarchischen Aufgaben [25] glänzen, bietet die explizite, zustandsbehaftete Graph-Architektur von LangGraph die notwendige Kontrolle, Beobachtbarkeit und das deterministisch-flexible Workflow-Management, das zur Implementierung der komplexen, mehrstufigen Prozesse des KIP erforderlich ist.

Technischer Entwurf für das persistente Gedächtnis des KIP

- **Technologiewahl: Neo4j Knowledge Graph:** Ein Vergleich von Graphdatenbanken zeigt, dass Neo4j über ein ausgereiftes Ökosystem, eine starke Community und vor allem exzellente Integrationen mit KI-Frameworks wie LangChain/LangGraph zur Implementierung von GraphRAG verfügt.[26] Während TigerGraph bei massiv-skaligen Analysen Leistungsvorteile bieten kann [29], ist die Ausgewogenheit von Funktionen, Benutzerfreundlichkeit und starkem GenAI-Tooling bei Neo4j für die Anforderungen dieses Projekts überlegen.
- **Implementierungsmuster: GraphRAG:** Das System wird das GraphRAG-Muster verwenden, um mit seinem Langzeitgedächtnis zu interagieren.[30]
  - **Ingestion:** Der "Reflektor"-Agent wird LLMs nutzen, um Entitäten und Beziehungen aus Aufgabenprotokollen und unstrukturierten Daten (wie Dokumentationen) zu extrahieren und so Graph-Knoten (Zettel) und Kanten (Verknüpfungen) zu erstellen.[31] Dies automatisiert den Prozess der Wissenskuration.[33]
  - **Retrieval:** Zu Beginn einer neuen Aufgabe fragt ein Agent den Knowledge Graphen ab. Dies ist keine einfache Schlüsselwortsuche, sondern ein **Multi-Hop-Reasoning**-Prozess.[30] Um beispielsweise zu beantworten: "Wie soll ich die Authentifizierung implementieren?", kann der Agent den Knoten Benutzerauthentifizierung abrufen, den Verknüpfungen zu OAuth2-Muster-Knoten folgen, die wiederum mit spezifischen Code-Snippet-Knoten und Frühere Implementierungsentscheidung-Knoten verknüpft sind. Dies liefert einen reichhaltigen, vernetzten Kontext, den eine reine Vektorsuche nicht bieten kann.[30]

Technische Definition des "System-Gärtners"

Der "System-Gärtner" wird als separater, langlebiger LangGraph-Prozess implementiert, der auf einer Meta-Ebene operiert. Seine Funktionen umfassen:

1. **Systemzustandsüberwachung:** Periodische Analyse von Protokollen und Leistungsmetriken (z. B. Aufgabenerfolgsraten, Token-Verbrauch pro Agent).
2. **Knowledge Graph Kuration:** Auslösen automatisierter Wartungsaufgaben für den Graphen, wie das Identifizieren verwaister Knoten oder das Vorschlagen neuer Verknüpfungen, unter Verwendung von Metriken zur Graphenqualität.[33]
3. **Verfassungsaufsicht:** Überwachung der Agentenkonversationen auf mögliche Verstöße gegen die "Verfassung" und Initiierung eines Überprüfungsprozesses bei Abweichungen.
4. **Parameter-Tuning:** Vorschlagen von Anpassungen an Agenten-Prompts oder Werkzeugkonfigurationen auf der Grundlage von Leistungsanalysen, die von einem Menschen überprüft werden.

Die Kognitive Orchestrierung: Vom Individuum zum intelligenten Kollektiv

Der Haupt-Orchestrator wird als StateGraph in LangGraph realisiert. Bei Eingang einer übergeordneten Aufgabe (z. B. einer User-Story aus Backlog.md) ist der erste Knoten im Graphen ein "Planner"-Agent. Dieser zerlegt die Aufgabe in Teilaufgaben und identifiziert die benötigten Spezialagenten-Rollen aus der claude-code-templates-Bibliothek. Diese Planung wird in das Zustandsobjekt geschrieben. Der Orchestrator nutzt dann bedingte Kanten, um die Aufgaben dynamisch an die entsprechenden Spezialisten zu leiten. Dies ermöglicht einen flexiblen, puppenspielerartigen Arbeitsablauf, bei dem der Orchestrator die Fäden zieht, ohne jeden Schritt starr vorzugeben.[37]

Die Kommunikation und das Zustandsmanagement werden durch ein zentrales Zustandsobjekt realisiert, das als Python TypedDict definiert wird, um Typsicherheit und einen klaren Datenvertrag zu gewährleisten.[38] Dieses Objekt wird Felder wie

task_description, sub_tasks, artifacts (für Code, Dokumente etc.) und conversation_history enthalten.[40] Agenten kommunizieren primär durch die Modifikation dieses zentralen Zustandsobjekts. Ein Knoten (Agent) empfängt den aktuellen Zustand, führt seine Arbeit aus und gibt ein Wörterbuch mit den Änderungen zurück, die auf den Zustand angewendet werden sollen.[15] Um eine zuverlässige Systemlogik zu ermöglichen, müssen die Agenten strukturierte Ausgaben

anstelle von reinem Fließtext zurückgeben.[23]

Die "Verfassung" des Agenten-Systems: Engineering von prinzipientreuer Autonomie

Die "System-Verfassung" wird als ein mehrschichtiges Framework implementiert:

- **Schicht 1: Ethische Kerndirektiven (Universelle Prinzipien):** Nicht verhandelbare Grundsätze, die auf der KIP-Philosophie und allgemeinen KI-Sicherheitsprinzipien basieren (z. B. abgeleitet aus der UN-Menschenrechtscharta), um sicherzustellen, dass das System "hilfreich, ehrlich und harmlos" ist.[41]
- **Schicht 2: Projektspezifische Ziele (Strategische Prinzipien):** Übersetzen die übergeordneten Ziele des Backlog.md-Projekts in Leitprinzipien (z. B. "Priorisiere modularen, testbaren und wartbaren Code").
- **Schicht 3: Operative Regeln (Taktische Prinzipien):** Spezifische Anweisungen für häufige Situationen (z. B. "Eskaliere bei schwerwiegenden Architekturfehlern im Code-Review an den 'Architekt'-Agenten").

Die Ausrichtung an dieser Verfassung erfolgt mittels **Constitutional AI (CAI)**.[43] Der Prozess umfasst eine Phase des überwachten Lernens, in der das Modell lernt, seine eigenen Ausgaben anhand der Verfassung zu kritisieren und zu überarbeiten, gefolgt von einer Phase des Reinforcement Learning from AI Feedback (RLAIF), in der ein Belohnungsmodell auf der Grundlage von KI-generierten Präferenzen trainiert wird, die sich an der Verfassung orientieren.[42] Dies ist skalierbarer und konsistenter als die alleinige Abhängigkeit von menschlichem Feedback.[41]

Eine statische Verfassung ist jedoch eine Schwachstelle. Projekte und Werte entwickeln sich weiter, was eine **dynamische Werteanpassung** erfordert.[44] Die Lösung ist der "System-Gärtner" als Verfassungshüter. Er überwacht das System auf "Wertedrift" und kann Änderungen an den Schichten 2 und 3 der Verfassung vorschlagen. Diese Vorschläge werden einem Menschen zur Genehmigung vorgelegt. Nach der Genehmigung kann der Gärtner einen neuen CAI-Feinabstimmungslauf auslösen, um den Orchestrator an die aktualisierte Verfassung anzupassen. Dies schafft ein System, das seine eigene Governance-Struktur unter menschlicher Aufsicht modifizieren kann – ein entscheidender Schritt über die standardmäßige CAI hinaus.[47]

Die Prototyping-Strategie für Backlog.md

Die Entwicklung des ersten Prototyps folgt einer strukturierten, iterativen Roadmap, die auf bewährten Praktiken für Proof-of-Concepts (PoC) basiert.[48]

Roadmap und Anforderungen für den Prototyp

Phase	Hauptziele	Kern-Liefergegenstände	Benötigte Agenten (claude-code-templates)	Erfolgsmetriken (KPIs)
Phase 1: Fundament & Kerndienste	Aufbau der Kernarchitektur. Implementierung des Basis-Orchestrators (LangGraph) und des Knowledge Graphen (Neo4j).	Lauffähiger LangGraph-Dienst, laufende Neo4j-Instanz, grundlegendes Zustandsmanagement (TypedDict).	Keine	Erfolgreiche Bereitstellung, grundlegende Graph-Konnektivität.
Phase 2: Einzelagenten-Aufgabenausführung	Integration eines einzelnen Spezialagenten. Testen der Werkzeugnutzung und grundlegender Zustandsaktualisierungen.	Ein einfacher Graph, der eine Aufgabe ("schreibe eine 'hello world'-Funktion") an einen Python_Developer-Agenten übergibt und den Code im Zustand speichert.	Python_Developer	Erfolgreiche Codegenerierung, Artefakt im Zustand gespeichert.

Phase 3: Multi-Agenten-Kollaboration & Gedächtnisintegration	Implementierung der vollständigen End-to-End-User-Story. Demonstration von Agenten-Teamarbeit und Interaktion mit dem Knowledge Graphen.	Voll funktionsfähige Demonstration der ausgewählten User-Story. Der Knowledge Graph enthält Knoten, die die Anforderungen, den Code und die Entscheidungen des Prozesses repräsentieren.	Product_Manager, Python_Developer, Code_Reviewer, Reflector	Erfolgreicher Abschluss der User-Story, messbare Codequalität, sinnvolle Daten im Knowledge Graphen, erfolgreicher Abruf dieser Daten bei einer Folgeanfrage.

Optimale User-Story für den ersten Prototyp

Empfehlung: Ein Epic oder eine umfassende User-Story, die einen vollständigen Entwicklungszyklus erfordert. Ein starker Kandidat aus einem typischen Backlog.md wäre: **"Als Benutzer möchte ich mich für ein Konto registrieren und einloggen können, damit ich auf personalisierte Inhalte zugreifen kann."**

Begründung: Diese Wahl ist einer einfachen, isolierten Aufgabe überlegen, da sie die Demonstration der innovativsten Merkmale des Systems erzwingt [50]:

- **Multi-Agenten-Kollaboration:** Sie erfordert einen Product_Manager-Agenten zur Definition der Anforderungen, einen Developer-Agenten zum Schreiben der Logik und einen Code_Reviewer-Agenten zur Validierung der Arbeit.
- **Zustandsmanagement:** Der Prozess erzeugt mehrere Artefakte (Anforderungsdokument, API-Spezifikation, Codedateien), die im Zustandsobjekt verwaltet werden müssen.
- **Langzeitgedächtnis:** Das System kann die Implementierungsdetails im Knowledge Graphen speichern. Wenn später eine neue Story wie "Passwort zurücksetzen implementieren" eingeführt wird, kann das System den Graphen abfragen, um die bestehende Authentifizierungsarchitektur zu verstehen und so seine Lernfähigkeit zu demonstrieren.

Minimaler Satz von Spezialagenten

Basierend auf der gewählten User-Story wäre der anfängliche Satz von zu integrierenden Agenten aus der claude-code-templates-Bibliothek:

1. **Product_Manager_Agent:** Zur Analyse der User-Story und zur Erstellung detaillierter Anforderungen.
2. **Python_Developer_Agent:** Zum Schreiben der Kernanwendungslogik.
3. **Code_Reviewer_Agent:** Zur Analyse des generierten Codes auf Qualität, Korrektheit und Einhaltung von Standards.
4. **Reflector_Agent:** Zur Durchführung der Nachbereitung der Aufgabe und zur Aktualisierung des Knowledge Graphen.

Quellen

[1]

Referenzen

1. Forschung der Gedächtnis: Überblick & Modelle | StudySmarter, Zugriff am August 5, 2025, https://www.studysmarter.de/schule/psychologie/forschung-der-gedaechtnis/
2. Gedächtnismodelle: Übersicht & Vergleich - StudySmarter, Zugriff am August 5, 2025, https://www.studysmarter.de/schule/psychologie/grundlagendisziplinen-der-psychologie/gedaechtnismodelle/
3. Kollektives Gedächtnis - Wikipedia, Zugriff am August 5, 2025, https://de.wikipedia.org/wiki/Kollektives_Ged%C3%A4chtnis
4. Kollektives Gedächtnis | Geschichte und Erinnerung | bpb.de, Zugriff am August 5, 2025, https://www.bpb.de/themen/erinnerung/geschichte-und-erinnerung/39802/kollektives-gedaechtnis/
5. Selbstorganisationsansatz - Leonhard, Zugriff am August 5, 2025, http://www.dobusch.net/pub/uni/200106sa.pdf
6. Selbstorganisation - Wikipedia, Zugriff am August 5, 2025, https://de.wikipedia.org/wiki/Selbstorganisation

7. Geschichte systemtheoretischer Konzepte – Verhalten: Grundlagen und Modelle, Zugriff am August 5, 2025, https://zuugs.hfh.ch/verhalten/chapter/geschichte-systemtheoretischer-konzepte/
8. Systemtheorie - Was ist das? - PURE Consultant, Zugriff am August 5, 2025, https://www.pureconsultant.de/de/management/systemtheorie/
9. 4 Modelle zur Verbesserung der Teameffektivität [2025] - Asana, Zugriff am August 5, 2025, https://asana.com/de/resources/team-effectiveness
10. Evidenzbasierte Wirtschaftspsychologie (26) - Kontrolle ist gut, aber Vertrauen ist besser. So fördern Sie Vertrauen in Ihrem T - Department Psychologie - LMU München, Zugriff am August 5, 2025, https://www.psy.lmu.de/evidenzbasiertesmanagement/dokumente/ebm_dossiers/ebm_26_vertrauen.pdf
11. Zettelkasten Method: How To Take Smart Notes For Knowledge Management. | by Disputant, Zugriff am August 5, 2025, https://disputant.medium.com/zettelkasten-method-how-to-take-smart-notes-for-knowledge-management-a66f636ede6c
12. Getting Started • Zettelkasten Method, Zugriff am August 5, 2025, https://zettelkasten.de/overview/
13. Zettelkasten deutsch - Smarte Notizen schreiben | Wissensmanagement 2.0 - YouTube, Zugriff am August 5, 2025, https://m.youtube.com/watch?v=avb6b0nL3NQ&pp=ygUcI25pY2h0b2huZW1laW5lbnpldHRlbGthc3Rlbg%3D%3D
14. LangGraph vs AutoGen vs CrewAI: Best Multi-Agent Tool?, Zugriff am August 5, 2025, https://www.amplework.com/blog/langgraph-vs-autogen-vs-crewai-multi-agent-framework/
15. LangGraph - LangChain Blog, Zugriff am August 5, 2025, https://blog.langchain.com/langgraph/
16. AutoGen vs. LangGraph vs. CrewAI:Who Wins? | by Khushbu Shah | ProjectPro - Medium, Zugriff am August 5, 2025, https://medium.com/projectpro/autogen-vs-langgraph-vs-crewai-who-wins-02e6cc7c5cb8
17. AutoGen: Microsofts Framework für kollaborative KI-Agenten - DataScientest, Zugriff am August 5, 2025, https://datascientest.com/de/was-ist-autogen
18. CrewAI new features. Agentic AI framework | by Xin Cheng - Medium, Zugriff am August 5, 2025, https://billtcheng2013.medium.com/crewai-new-features-ff79f55cdc79
19. LangGraph vs AutoGen: Comparing AI Agent Frameworks - PromptLayer, Zugriff am August 5, 2025, https://blog.promptlayer.com/langgraph-vs-autogen/
20. AI Agent Memory: A Comparative Analysis of LangGraph, CrewAI, and AutoGen, Zugriff am August 5, 2025, https://dev.to/foxgem/ai-agent-memory-a-comparative-analysis-of-langgraph-crewai-and-autogen-31dp
21. Let's Compare CrewAI, AutoGen, Vertex AI, and LangGraph Multi-Agent

Frameworks | Infinite Lambda Blog, Zugriff am August 5, 2025, https://infinitelambda.com/compare-crewai-autogen-vertexai-langgraph/
22. The Evolution of AI Agents frameworks: From Autogen to LangGraph - BigHub, Zugriff am August 5, 2025, https://www.bighub.ai/blog/the-evolution-of-ai-agents-frameworks-from-autogen-to-langgraph
23. Die Orchestrierung von AI-Agenten verstehen - Botpress, Zugriff am August 5, 2025, https://botpress.com/de/blog/ai-agent-orchestration
24. LangGraph is Not a True Agentic Framework | by Saeed Hajebi | Medium, Zugriff am August 5, 2025, https://medium.com/@saeedhajebi/langgraph-is-not-a-true-agentic-framework-3f010c780857
25. Mastering Agents: LangGraph Vs Autogen Vs Crew AI - Galileo AI, Zugriff am August 5, 2025, https://galileo.ai/blog/mastering-agents-langgraph-vs-autogen-vs-crew
26. Generative AI - Ground LLMs with Knowledge Graphs - Neo4j, Zugriff am August 5, 2025, https://neo4j.com/generativeai/
27. Choosing the best graph database for your organization: A practical guide - Linkurious, Zugriff am August 5, 2025, https://linkurious.com/blog/choosing-the-best-graph-database/
28. 7 Best AWS Neptune Alternatives In 2025 - PuppyGraph, Zugriff am August 5, 2025, https://www.puppygraph.com/blog/aws-neptune-alternatives
29. Buyer's Guide For Graph Databases - TigerGraph, Zugriff am August 5, 2025, https://www.tigergraph.com.cn/wp-content/uploads/2021/06/TigerGraph-Buyers-Guide-Comparison.pdf
30. How to Improve Multi-Hop Reasoning With Knowledge Graphs and ..., Zugriff am August 5, 2025, https://neo4j.com/blog/genai/knowledge-graph-llm-multi-hop-reasoning/
31. Introduction to the Neo4j LLM Knowledge Graph Builder - Graph Database & Analytics, Zugriff am August 5, 2025, https://neo4j.com/blog/developer/llm-knowledge-graph-builder/
32. Neo4j LLM Knowledge Graph Builder - Extract Nodes and Relationships from Unstructured Text, Zugriff am August 5, 2025, https://neo4j.com/labs/genai-ecosystem/llm-graph-builder/
33. KARMA: Leveraging Multi-Agent LLMs for Automated Knowledge Graph Enrichment - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2502.06472v1
34. AWS Marketplace: CitiusTech Knowledge Graph Solution - Amazon.com, Zugriff am August 5, 2025, https://aws.amazon.com/marketplace/pp/prodview-zuzpu76p6zoxk
35. Integrate LLM workflows with Knowledge Graph using Neo4j and APOC - Medium, Zugriff am August 5, 2025, https://medium.com/data-science/integrate-llm-workflows-with-knowledge-graph-using-neo4j-and-apoc-27ef7e9900a2
36. LLM Evaluation Metrics: The Ultimate LLM Evaluation Guide - Confident AI, Zugriff am August 5, 2025,

https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation
37. [2505.19591] Multi-Agent Collaboration via Evolving Orchestration - arXiv, Zugriff am August 5, 2025, https://arxiv.org/abs/2505.19591
38. state graph node - GitHub Pages, Zugriff am August 5, 2025, https://langchain-ai.github.io/langgraph/concepts/low_level/
39. LangGraph Tutorial: Understanding State Management - Unit 1.1 Exercise 1, Zugriff am August 5, 2025, https://aiproduct.engineer/tutorials/langgraph-tutorial-understanding-state-management-unit-11-exercise-1
40. 5. Customize state - GitHub Pages, Zugriff am August 5, 2025, https://langchain-ai.github.io/langgraph/tutorials/get-started/5-customize-state/
41. On 'Constitutional' AI - The Digital Constitutionalist, Zugriff am August 5, 2025, https://digi-con.org/on-constitutional-ai/
42. Claude's Constitution - Anthropic, Zugriff am August 5, 2025, https://www.anthropic.com/news/claudes-constitution
43. Constitutional AI: Harmlessness from AI Feedback \ Anthropic, Zugriff am August 5, 2025, https://www.anthropic.com/research/constitutional-ai-harmlessness-from-ai-feedback
44. Superalignment with Dynamic Human Values - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2503.13621v1
45. [2406.11039] Dynamic Normativity: Necessary and Sufficient Conditions for Value Alignment, Zugriff am August 5, 2025, https://arxiv.org/abs/2406.11039
46. Adaptive Alignment: Dynamic Preference Adjustments via Multi-Objective Reinforcement Learning for Pluralistic AI - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2410.23630v1
47. SEAL and the Hidden Risks of Self-Editing AI Models - Annielytics.com, Zugriff am August 5, 2025, https://www.annielytics.com/blog/ai/seal-and-the-hidden-risks-of-self-editing-ai-models/
48. AI Proof of Concept (PoC): What It Is & How to Build One? - Quinnox, Zugriff am August 5, 2025, https://www.quinnox.com/blog/ai-proof-of-concept-guide/
49. Guide to AI Proof-of-concept (PoC) for risk mitigation with examples - Innowise, Zugriff am August 5, 2025, https://innowise.com/blog/ai-proof-of-concept/
50. 5 Steps to an AI Proof of Concept - Intel, Zugriff am August 5, 2025, https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/five-steps-ai-proof-of-concept-whitepaper.pdf
51. How can AI Proof of Concept (PoC) Drive Success in Your AI Journey? - BiztechCS, Zugriff am August 5, 2025, https://www.biztechcs.com/blog/how-can-ai-proof-of-concept-poc-drive-success-in-your-ai-journey/
52. Langzeitgedächtnis - Kognitive Fähigkeit - CogniFit, Zugriff am August 5, 2025, https://www.cognifit.com/de/wissenschaft/kognitive-fahigkeiten/langzeitgedachtnis

53. Systemtheorie Eine Einführung für Führungskräfte, Wirtschafts- und Sozialwissenschafter 3. Auflage Prof. Dr. Markus Schwaninger - AMS Forschungsnetzwerk, Zugriff am August 5, 2025, https://forschungsnetzwerk.ams.at/dam/jcr:c0f1df0b-c518-4892-966c-ca5723d83491/systemtheorie-%20einfuehrung.pdf
54. Arbeits- und Organisationspsychologie - ReadingSample - NET, Zugriff am August 5, 2025, https://beckassets.blob.core.windows.net/product/readingsample/542760/9783540747048_excerpt_001.pdf
55. Projekt- und Teamarbeit in der digitalisierten Arbeitswelt - Kooperationsstelle - Universität Göttingen, Zugriff am August 5, 2025, https://kooperationsstelle.uni-goettingen.de/fileadmin/projekt_und_teamarbeit_in_der_digitalisierten_arbeitswelt/dokumentation/kooperationsstelle/2021_Book_Projekt-UndTeamarbeitInDerDigi.pdf
56. Teams synergetisch entwickeln | Beiträge zur Entwicklung von Teams in der Arbeitswelt unter besonderer Berücksichtigung der Synergetik als Lehre der Selbstorganisation - ResearchGate, Zugriff am August 5, 2025, https://www.researchgate.net/publication/319653842_Teams_synergetisch_entwickeln_Beitrage_zur_Entwicklung_von_Teams_in_der_Arbeitswelt_unter_besonderer_Berucksichtigung_der_Synergetik_als_Lehre_der_Selbstorganisation
57. Zettelkastenmethodologie - Zettlr Docs, Zugriff am August 5, 2025, https://docs.zettlr.com/de/academic/zkn-method/
58. Zettelkasten Methode für Anfänger: Nie wieder Wissen verlernen - Zenkit, Zugriff am August 5, 2025, https://zenkit.com/de/blog/zettelkasten-methode/
59. AutoGen — Orchestrator-Worker Agents Design Pattern | by Steve Zebib | oracle-saas-paas, Zugriff am August 5, 2025, https://medium.com/oracle-saas-paas/autogen-orchestrator-worker-agents-design-pattern-eef8698459b2
60. Was ist AutoGen? Unser vollständiger Leitfaden zur Multi-Agenten ..., Zugriff am August 5, 2025, https://skimai.com/de/was-ist-autogen-unser-vollstandiger-leitfaden-zur-autogen-multi-agenten-plattform/
61. Was ist LLM-Orchestrierung? - IBM, Zugriff am August 5, 2025, https://www.ibm.com/de-de/think/topics/llm-orchestration
62. Systemtheorie einfach erklärt: Einführung - YouTube, Zugriff am August 5, 2025, https://www.youtube.com/watch?v=OX2BIGWH6RM
63. Einführung in Systemtheorie und Konstruktivismus von Fritz B. Simon - Carl-Auer Verlag, Zugriff am August 5, 2025, https://www.carl-auer.de/einfuhrung-in-systemtheorie-und-konstruktivismus
64. Systemtheorie nach Luhmann einfach erklärt - Fritz Führungskreise, Zugriff am August 5, 2025, https://www.fritz.tips/systemtheorie-nach-luhmann-einfach-erklaert
65. Einführung in Systemtheorie und Konstruktivismus - socialnet Rezensionen, Zugriff am August 5, 2025, https://www.socialnet.de/rezensionen/28174.php
66. Einführung in Systemtheorie und Konstruktivismus - socialnet Rezensionen,

Zugriff am August 5, 2025, https://www.socialnet.de/rezensionen/3507.php
67. Einführung in die systemische Organisationstheorie - EconBiz, Zugriff am August 5, 2025, https://www.econbiz.de/Record/einf%C3%BChrung-in-die-systemische-organisationstheorie-simon-fritz/10003463326
68. Wie man mit der Zettelkasten-Methode Informationen & Ideen verwaltet - ClickUp, Zugriff am August 5, 2025, https://clickup.com/de/blog/237515/zettelkastenmethode
69. Das Zettelkasten-Prinzip: Dein digitaler Wissensschatz - Notioneers.io, Zugriff am August 5, 2025, https://notioneers.io/das-zettelkasten-prinzip-dein-digitaler-wissensschatz/
70. The Zettelkasten Method: A Beginner's Guide | Goodnotes Blog, Zugriff am August 5, 2025, https://www.goodnotes.com/blog/zettelkasten-method
71. Niklas Luhmanns Zettelkasten: ein „preadaptive ... - Uni Bielefeld, Zugriff am August 5, 2025, https://www.uni-bielefeld.de/einrichtungen/bits/ueber-uns/knowledge-transfer/2018/2018-04-19_Vortrag_Das_Digitalisierungsprojekt_Luhmann-Zettelkasten.pdf
72. Constitutional AI - Daniela Amodei (Anthropic - YouTube, Zugriff am August 5, 2025, https://www.youtube.com/watch?v=Tjsox6vfsos
73. Claude AI's Constitutional Framework: A Technical Guide to Constitutional AI | by Generative AI | Medium, Zugriff am August 5, 2025, https://medium.com/@genai.works/claude-ais-constitutional-framework-a-technical-guide-to-constitutional-ai-704942e24a21
74. Collective Constitutional AI: Aligning a Language Model with Public Input - Anthropic, Zugriff am August 5, 2025, https://www.anthropic.com/research/collective-constitutional-ai-aligning-a-language-model-with-public-input
75. Neo4j's LLM Graph Builder seems useless : r/Rag - Reddit, Zugriff am August 5, 2025, https://www.reddit.com/r/Rag/comments/1i1980p/neo4js_llm_graph_builder_seems_useless/
76. Next Level Agent State Management with LangGraph | Part 9 ..., Zugriff am August 5, 2025, https://www.youtube.com/watch?v=tlrHl7iKpD0
77. LangGraph Tutorial: Building LLM Agents with LangChain's Agent Framework - Zep, Zugriff am August 5, 2025, https://www.getzep.com/ai-agents/langgraph-tutorial/
78. LangGraph: Build Stateful AI Agents in Python, Zugriff am August 5, 2025, https://realpython.com/langgraph-python/
79. What are hierarchical multi-agent systems? - Milvus, Zugriff am August 5, 2025, https://milvus.io/ai-quick-reference/what-are-hierarchical-multiagent-systems
80. AI Agents: Evolution, Architecture, and Real-World Applications - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2503.12687v1
81. How we built our multi-agent research system - Anthropic, Zugriff am August 5, 2025, https://www.anthropic.com/engineering/built-multi-agent-research-system
82. Application-Driven Value Alignment in Agentic AI Systems: Survey and

Perspectives - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2506.09656v1
83. What is a Multi-Agent System? | IBM, Zugriff am August 5, 2025, https://www.ibm.com/think/topics/multiagent-system
84. From RAG to Multi-Agent Systems: A Survey of Modern Approaches in LLM Development, Zugriff am August 5, 2025, https://www.preprints.org/manuscript/202502.0406/v1
85. AI Proof of Concept: Benefits, Stages, Challenges, and More | by Sandra Parker | Medium, Zugriff am August 5, 2025, https://sandra-parker.medium.com/ai-proof-of-concept-benefits-stages-challenges-and-more-407f218a56a7
86. How to Build an AI Proof of Concept – That Investors LOVE - InvoZone, Zugriff am August 5, 2025, https://invozone.com/blog/build-an-ai-proof-of-concept/
87. Comparision of AWS Neptune vs Neo4j | by Kavitha Reddy - Medium, Zugriff am August 5, 2025, https://medium.com/@kavithareddy.gade/comparision-of-aws-neptune-vs-neo4j-e9110a827057
88. How To Choose A Graph Database: We Compare 8 Favorites - Cambridge Intelligence, Zugriff am August 5, 2025, https://cambridge-intelligence.com/choosing-graph-database/
89. Understanding Knowledge Graph Stores: A Comprehensive Comparison - Medium, Zugriff am August 5, 2025, https://medium.com/@iamshowkath/understanding-knowledge-graph-stores-a-comprehensive-comparison-d7b9248c1ecd
90. Multi-agent systems - which approaches are giving you the best results?, Zugriff am August 5, 2025, https://community.latenode.com/t/multi-agent-systems-which-approaches-are-giving-you-the-best-results/31075
91. [2507.21969] Towards Cognitive Synergy in LLM-Based Multi-Agent Systems: Integrating Theory of Mind and Critical Evaluation - arXiv, Zugriff am August 5, 2025, https://arxiv.org/abs/2507.21969
92. arXiv:2402.01968v2 [cs.MA] 29 Jan 2025, Zugriff am August 5, 2025, https://arxiv.org/pdf/2402.01968
93. A Comprehensive Survey on Multi-Agent Cooperative Decision-Making: Scenarios, Approaches, Challenges and Perspectives - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2503.13415v1
94. Multi-Modal and Multi-Agent Systems Meet Rationality: A Survey - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2406.00252v2
95. Modeling Theory of Mind in Multi-Agent Games Using Adaptive Feedback Control - arXiv, Zugriff am August 5, 2025, https://arxiv.org/abs/1905.13225
96. Unified Mind Model: Reimagining Autonomous Agents in the LLM Era - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2503.03459v2
97. [2502.11882] Leveraging Dual Process Theory in Language Agent Framework for Real-time Simultaneous Human-AI Collaboration - arXiv, Zugriff am August 5, 2025, https://arxiv.org/abs/2502.11882

98. Cognitive Architectures for Language Agents - arXiv, Zugriff am August 5, 2025, http://arxiv.org/pdf/2309.02427
99. Cognitive Architectures for Language Agents - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2309.02427v3
100. Can A Cognitive Architecture Fundamentally Enhance LLMs? Or Vice Versa? Ron Sun - arXiv, Zugriff am August 5, 2025, https://arxiv.org/pdf/2401.10444
101. KARMA: Leveraging Multi-Agent LLMs for Automated Knowledge Graph Enrichment, Zugriff am August 5, 2025, https://powerdrill.ai/discover/summary-karma-leveraging-multi-agent-llms-for-automated-cm70yvllybobq07s3xrbumuie
102. Automating Knowledge Graph Creation and Curation, Zugriff am August 5, 2025, https://www.datajps.com/en/blog/automating-knowledge-graph-creation-and-curation
103. I Tried to Automate Knowledge Graph Schema and It Blew My Mind - YouTube, Zugriff am August 5, 2025, https://www.youtube.com/watch?v=Plu1ILTpaN8
104. A Complete List of All the LLM Evaluation Metrics You Need to Think About : r/LangChain, Zugriff am August 5, 2025, https://www.reddit.com/r/LangChain/comments/1j4tsth/a_complete_list_of_all_the_llm_evaluation_metrics/
105. a knowledge graph–enhanced LLM framework for pan-cancer question answering | GigaScience | Oxford Academic, Zugriff am August 5, 2025, https://academic.oup.com/gigascience/article/doi/10.1093/gigascience/giae082/7943459
106. Large Language Model–Driven Knowledge Graph Construction in Sepsis Care Using Multicenter Clinical Databases: Development and Usability Study, Zugriff am August 5, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11986385/
107. Enhancing LLM Tool Use with High-quality Instruction Data from Knowledge Graph - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2506.21071v1
108. [2503.22115] Beyond Single-Sentence Prompts: Upgrading Value Alignment Benchmarks with Dialogues and Stories - arXiv, Zugriff am August 5, 2025, https://arxiv.org/abs/2503.22115
109. How to build safer development workflows with Constitutional AI, Zugriff am August 5, 2025, https://pieces.app/blog/constitutional-ai
110. C3AI: Crafting and Evaluating Constitutions for Constitutional AI - arXiv, Zugriff am August 5, 2025, https://arxiv.org/html/2502.15861v1
111. Could an AI Essay Writer Write a Better Constitution Than Humans? - Yomu AI, Zugriff am August 5, 2025, https://www.yomu.ai/resources/could-an-ai-essay-writer-write-a-better-constitution-than-humans
112. Constitutional AI: The Essential Guide | Nightfall AI Security 101, Zugriff am August 5, 2025, https://www.nightfall.ai/ai-security-101/constitutional-ai
113. Comparing Open-Source AI Agent Frameworks - Langfuse Blog, Zugriff am August 5, 2025, https://langfuse.com/blog/2025-03-19-ai-agent-comparison
114. Comparing LLM Agent Frameworks Controllability and Convergence:

LangGraph vs AutoGen vs CREW AI | by ScaleX Innovation, Zugriff am August 5, 2025, https://scalexi.medium.com/comparing-llm-agent-frameworks-langgraph-vs-autogen-vs-crew-ai-part-i-92234321eb6b

115. Zugriff am Januar 1, 1970, https.zuugs.hfh.ch/verhalten/chapter/geschichte-systemtheoretischer-konzepte/

116. Zugriff am Januar 1, 1970, https.zettelkasten.de/overview/