KI-gestützte NPC-Dialoge in Unity

Konzept und Prototyp zur glaubwürdigen, wissensbegrenzten Interaktion

Abschlussarbeit zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.) im Fachbereich Game Design

vorgelegt von:

Johannes Blank

GD2021

Mediadesign Hochschule

Betreuer:	Prof. Heiko Ihde
	Prof. Dominik Mieth
Eingereicht am:	12.06.2026

Zusammenfassung

Diese Arbeit untersucht, wie KI-gestützte NPC-Dialoge in einem Unity-Prototyp so gestaltet werden können, dass sie in einem Krimi-Szenario charakterkonsistent, wissensbegrenzt und State-bezogen bleiben. Ausgangspunkt ist das Problem, dass Large Language Models flexible Dialoge ermöglichen, ohne Kontrolle jedoch zu Halluzinationen, Rollenbrüchen, unzulässiger Informationsgabe oder themenfremden Antworten neigen können. Dazu wurde ein Qualitätsrahmen mit Kriterien wie Charakterkonsistenz, Wissensbegrenzung, State-Konsistenz, Memory-Korrektheit und spielerischer Sinnhaftigkeit entwickelt. Darauf aufbauend entstand ein Unity-Prototyp mit drei NPCs, strukturierten Profilen, begrenztem Charakterwissen, State-Flags, NPC-Memory, Prompt-Constraints, Scope-Guard und Logging. Die Evaluation erfolgte anhand definierter Testfälle im API-Modus. Die Ergebnisse zeigen, dass der Prototyp Rollenbindung, Wissensbegrenzung, Scope-Guard und einfache Memory-Bezüge überwiegend unterstützt. Kritische Halluzinationen, Spoiler, Meta-Brüche oder Rollenbrüche traten nicht auf. Gleichzeitig zeigte sich eine Grenze bei der State-Steuerung: State-Flags als Prompt-Kontext helfen, ersetzen aber keine harte Freigabelogik. Der Prototyp begrenzte unerwünschte Informationen besser, als freigeschaltete Informationen dramaturgisch auszuspielen. Damit zeigt die Arbeit, dass glaubwürdige KI-NPCs durch Figur, Wissen, State, Memory, Constraints und Evaluation entstehen, nicht durch eine reine API-Anbindung.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche einzeln kenntlich gemacht.

Ort, Datum 10.06.2026

Unterschrift 

Inhaltsverzeichnis

1. Einleitung

Nichtspielercharaktere, häufig als NPCs bezeichnet, übernehmen in digitalen Spielen eine zentrale Rolle. Sie vermitteln Hinweise, Aufgaben, soziale Beziehungen und narrative Orientierung und tragen dadurch zur Glaubwürdigkeit einer Spielwelt bei. Warpefelt und Verhagen beschreiben NPCs entsprechend als wichtigen Bestandteil der Game Experience, weil sie unter anderem Hilfe, Herausforderungen, Dienste und narrative Anbindung bereitstellen können (Warpefelt und Verhagen, 2017, S. 39–41).

Durch Large Language Models entstehen neue Möglichkeiten für dynamischere NPC-Dialoge. Während klassische Dialogsysteme meist auf vorab geschriebenen Dialogzeilen oder Auswahloptionen basieren, können LLMs auf freie Texteingaben reagieren und variablere Antworten erzeugen. Aktuelle Forschung betrachtet LLMs daher zunehmend auch im Kontext von Videospielen, insbesondere für narrative Systeme, Storytelling und NPC-Dialoge (Sweetser, 2024). Dadurch entsteht die Möglichkeit, NPCs stärker als interaktive Figuren zu gestalten, die auf individuelle Fragen der Spielenden reagieren. Diese Offenheit erzeugt jedoch ein zentrales Problem: Ein NPC darf nicht wie ein allgemeiner Chatbot antworten. Seine Aussagen müssen zur Figur, zur Spielwelt, zum aktuellen Spielzustand und zum Wissen der Figur passen. Generative Sprachmodelle können Inhalte erzeugen, die nicht durch den gegebenen Kontext gedeckt sind; dieses Problem wird in der Forschung als Halluzination beschrieben (Ji, et al., 2023). Im NPC-Kontext betrifft dies nicht nur faktische Richtigkeit, sondern auch die Rolle und den Wissensbereich der Figur. Liu, Xie und Jiang beschreiben dieses Problem als Character Halluzination, wenn generierte Antworten nicht zur Figur oder ihrem kognitiven Wissensbereich passen (Liu, et al., 2025).

Für narrative Spiele ist diese Problematik besonders relevant. Wenn ein NPC falsche Hinweise erfindet, zu früh Täterwissen preisgibt, auf nicht vorhandene Orte verweist oder über interne Systemregeln spricht, kann dies die Spielwelt und den Spielverlauf beschädigen. In einem Krimi-Szenario ist kontrollierte Informationsvergabe besonders wichtig, weil der Erkenntnisfortschritt der Spielenden von Hinweisen, Verdacht, Nichtwissen, falschen Fährten und schrittweiser Auflösung abhängt. Ein KI-gestütztes NPC-System muss

daher nicht nur sprachlich überzeugende Antworten erzeugen, sondern diese Antworten kontrollierbar an Figur, Wissen und State binden. Callison-Burch et al. zeigen im Rollenspielkontext, dass Dialoge unter anderem durch State Tracking sowie die Trennung zwischen In-Character- und Out-of-Character-Kommunikation geprägt sind (Callison-Burch, et al., 2022).

Die vorliegende Arbeit untersucht daher, wie KI-gestützte NPC-Dialoge in einem begrenzten Spielprototyp kontrollierbarer gestaltet werden können. Im Mittelpunkt steht kein vollständiges kommerzielles Spiel, sondern ein Unity-Prototyp eines kleinen Krimi-Dialogsystems. Dieser Prototyp kombiniert NPC-Profile, Wissensgrenzen, State-Informationen, NPC-bezogenes Memory, Prompt-Constraints, Scope-Guard und Logging. Ziel ist es zu prüfen, ob diese Bausteine dazu beitragen, NPC-Antworten charakterkonsistent, wissensbegrenzt und State-bezogen zu erzeugen.

Daraus ergibt sich die zentrale Forschungsfrage, wie ein KI-gestütztes NPC-Dialogsystem in einem Unity-Prototyp so konzipiert und umgesetzt werden kann, dass NPC-Antworten innerhalb eines narrativen Krimi-Szenarios charakterkonsistent, wissensbegrenzt und State-bezogen bleiben. Zur Beantwortung dieser Frage wird zunächst der Forschungsstand zu KI-gestützten NPC-Dialogen, Halluzinationen, Wissensbegrenzung, Charakterkonsistenz, State und Memory aufgearbeitet. Darauf aufbauend wird ein Qualitätsrahmen entwickelt, ein Systemkonzept für einen Krimi-Prototyp erstellt, dieser in Unity umgesetzt und anhand definierter Testfälle evaluiert.

Die Arbeit gliedert sich in einen theoretisch-konzeptionellen, einen praktischen und einen evaluativen Teil. Nach der Einführung in den Forschungsstand und die zentralen Begriffe wird zunächst ein Qualitätsrahmen für KI-gestützte NPC-Dialoge entwickelt und das methodische Vorgehen der Untersuchung erläutert. Darauf aufbauend werden das Systemkonzept und die Umsetzung des Unity-Prototyps vorgestellt. Anschließend erfolgt die Evaluation anhand definierter Testfälle, deren Ergebnisse diskutiert und im Hinblick auf die Möglichkeiten und Grenzen des gewählten Ansatzes eingeordnet werden. Den Abschluss bilden eine Zusammenfassung der zentralen Erkenntnisse sowie ein Ausblick auf mögliche Weiterentwicklungen.

2. Forschungsstand und Begriffe

Dieses Kapitel ordnet KI-gestützte NPC-Dialoge in den aktuellen Forschungsstand ein und klärt die für diese Arbeit zentralen Begriffe. Im Mittelpunkt stehen nicht allgemeine Chatbots, sondern Dialogsysteme für Non-Player Characters (NPCs), die innerhalb einer konkreten Spielwelt auftreten. NPCs übernehmen in digitalen Spielen nicht nur eine kommunikative Funktion, sondern vermitteln Aufgaben, Hintergrundinformationen, Hinweise, soziale Rollen und narrative Orientierung. Warpefelt und Verhagen beschreiben NPCs entsprechend als wichtigen Bestandteil der Game Experience, weil sie Spielenden unter anderem Hilfe, Herausforderungen, Dienste und narrative Anbindung bereitstellen können (Warpefelt und Verhagen, 2017, S. 40–41).

Für KI-gestützte NPC-Dialoge müssen Antworten daher nicht nur flüssig sein, sondern zu Rolle, Figurenwissen, aktuellem Spielzustand und Spielweltlogik passen. State bezeichnet hier den relevanten Spielzustand, etwa bekannte Hinweise, abgeschlossene Dialoge, aktuelle Ziele, Beziehungen, Gegenstände oder ausgelöste Ereignisse.

Unter einem Large Language Model (LLM) wird in dieser Arbeit ein großes Sprachmodell verstanden, das auf umfangreichen Textmengen trainiert wurde und auf Basis eines gegebenen Kontextes sprachliche Fortsetzungen, Antworten oder Zusammenfassungen erzeugen kann. Technisch beruhen viele moderne LLMs auf Transformer-Architekturen. Vaswani et al. führen den Transformer als Architektur ein, die Sequenzen über Attention-Mechanismen verarbeitet und dadurch Abhängigkeiten innerhalb eines Kontextes modellieren kann (Vaswani, et al., 2017, S. 2–5). Für NPC-Dialoge ist daran besonders relevant, dass längere Kontextbestandteile wie Dialoghistorie, Charakterprofil, Spielzustand und aktuelle Spielereingabe gemeinsam verarbeitet werden können. Entscheidend ist in dieser Arbeit jedoch weniger die interne Modellarchitektur als ihre praktische Eigenschaft, auf freie Texteingaben kontextabhängige Antworten zu erzeugen.

2.1. KI-gestützte NPC-Dialoge im Kontext aktueller Spieleforschung

Aktuelle Arbeiten zu Large Language Models in Videospielen zeigen, dass NPC-Dialoge inzwischen als eigenständiger Anwendungsbereich innerhalb der LLM-Games-Forschung betrachtet werden. Sweetser (2024) untersucht in einer Preliminary Scoping Review 76 Arbeiten zu LLMs und Videospielen aus dem Zeitraum 2022 bis Anfang 2024. Die Arbeiten werden unter anderem den Bereichen Game AI and Agents, Game Development and Play, Narrative, Story and Dialogue sowie Game Research and Reviews zugeordnet (Sweetser, 2024, S. 1–3). Für die vorliegende Arbeit ist insbesondere relevant, dass Narrative, Story and Dialogue als eigener Themenbereich erscheinen und dort NPC-Dialoge, Questgenerierung und interaktive Geschichten zusammengeführt werden.

LLMs erscheinen damit als Bausteine dynamischer Spielwelten: Sie können auf freie Eingaben reagieren und variablere Interaktionen ermöglichen als fest geschriebene Dialogzeilen. Gerade narrative Spiele profitieren davon, wenn Figuren situativ und weniger repetitiv wirken.

Gleichzeitig zeigt Sweetser (2024), dass das Forschungsfeld noch jung und heterogen ist. Viele Arbeiten stellen frühe Prototypen, explorative Ansätze oder Systementwürfe dar. Positive Potenziale liegen unter anderem in interpretierbaren Agenten, sozialem Verhalten, Unterstützung bei der Spieleentwicklung und neuen Formen generierter Inhalte. Demgegenüber stehen Risiken wie begrenzte logische Schlussfolgerung, Unvorhersehbarkeit und Qualitätsunterschiede in generierten Inhalten (Sweetser, 2024, S. 6–7). Für KI-gestützte NPC-Dialoge ist diese Spannung zentral: Der gleiche Mechanismus, der flexible und überraschende Antworten ermöglicht, kann auch zu unpassenden, inkonsistenten oder spielmechanisch problematischen Aussagen führen.

Ashby et al. (2023) zeigen diese Spannung im konkreten Kontext von Rollenspielen. Ihr Ansatz verbindet Spielereingaben, einen Knowledge Graph und ein Sprachmodell, um personalisierte Quests und dazu passende NPC-Dialoge zu erzeugen. Die Arbeit betont damit, dass generative Dialoge im Spiel nicht isoliert betrachtet werden können. Sie sind an Quests, Spielfiguren, Orte, Objekte und den Zustand der Spielwelt gebunden (Ashby,

et al., 2023, S. 1–2). Das unterscheidet NPC-Dialoge von allgemeinen Chatbot-Antworten: Eine Antwort ist nicht nur dann gut, wenn sie verständlich formuliert ist, sondern wenn sie im Spiel eine sinnvolle Funktion erfüllt.

2.2. Probleme generativer NPC-Dialoge

Generative NPC-Dialoge stehen vor dem Problem, dass sprachliche Qualität allein nicht genügt. Eine grammatikalisch korrekte Antwort kann dennoch ungeeignet sein, wenn sie Figur, Spielwelt, Gesprächsverlauf oder State verletzt und dadurch falsche Hinweise, Questlogik oder Verdachtslenkung erzeugt.

Ein zentrales Problem ist Halluzination. In der allgemeinen Forschung zu Natural Language Generation wird Halluzination als Erzeugung von Text verstanden, der zwar plausibel formuliert sein kann, aber nicht durch die zugrunde liegenden Daten, die Eingabe oder den gegebenen Kontext gedeckt ist (Ji, et al., 2023, S. 1–2). Ji et al. unterscheiden dabei zwischen intrinsischer und extrinsischer Halluzination. Intrinsische Halluzination liegt vor, wenn eine generierte Aussage dem gegebenen Kontext widerspricht. Extrinsische Halluzination bezeichnet Aussagen, die durch den Kontext nicht belegbar oder verifizierbar sind (Ji, et al., 2023, S. 3).

Auf NPC-Dialoge übertragen bedeutet das: Intrinsische Halluzinationen widersprechen Charakterprofil, State oder etablierten Fallinformationen; extrinsische Halluzinationen führen zusätzliche Figuren, Orte, Hinweise oder Ereignisse ein, die im Spielsystem nicht existieren. Beides kann Spielende auf nicht vorgesehene Spuren lenken.

Ashby et al. (2023) beschreiben genau dieses Risiko im Kontext ihres Quest- und Dialogsystems. Zwar gelingt es dem Sprachmodell, aus generierten Questbeschreibungen flüssige und teilweise kreative NPC-Dialoge zu erzeugen. In der Failure Analysis wird jedoch festgehalten, dass das Sprachmodell zusätzliche Details erfinden kann, die nicht im Knowledge Graph enthalten sind. Solche Ergänzungen können für Spielende desorientierend wirken, wenn sie versuchen, auf Basis dieser erfundenen Informationen zu han-

deln (Ashby, et al., 2023, S. 12). Daraus folgt, dass ein NPC-Dialogsystem nicht nur kreative Antworten erzeugen darf, sondern die Herkunft und Begrenzung seiner Informationen kontrollieren muss.

Ein weiteres Problem ist der Rollenbruch. Liu, Xie und Jiang (2025) bezeichnen dieses Phänomen als character hallucination: Das Modell erzeugt Antworten, die von den Charaktereinstellungen abweichen oder Wissen nutzen, das außerhalb des kognitiven Bereichs der Figur liegt (Liu, et al., 2025, S. 1–2). So kann eine Figur unglaubwürdig wirken, wenn sie Wissen äußert, das weder zu ihrer Rolle noch zur historischen, sozialen oder fiktionalen Einbettung der Spielwelt passt. Das Problem besteht daher nicht nur in falschen Fakten, sondern in einer Verletzung der Figurperspektive.

Für NPCs ist dieser Unterschied besonders wichtig, weil Modellwissen nicht automatisch Charakterwissen ist. Eine Figur darf allgemeines Trainingswissen nur nutzen, wenn es innerhalb der Spielwelt, ihrer Rolle und ihrer Situation plausibel verfügbar ist; sonst brechen Immersion und erzählerische Kohärenz.

Callison-Burch et al. (2022) zeigen am Beispiel von Dungeons & Dragons, dass rollenspielartige Dialoge zusätzlich durch State Tracking, Rollenverhalten und die Trennung verschiedener Kommunikationsebenen geprägt sind. D&D eignet sich als Dialog-Challenge, weil das Spiel wesentlich über Sprache funktioniert und gleichzeitig Weltzustand, Figurenhandlungen, Regeln und kreative Rollenspieläußerungen berücksichtigt werden müssen (Callison-Burch, et al., 2022, S. 1–2). Für KI-Systeme bedeutet dies, dass sie nicht nur den nächsten Satz erzeugen, sondern mitverfolgen müssen, welche Figur spricht, in welcher Situation sie sich befindet und ob eine Äußerung innerhalb oder außerhalb der fiktionalen Welt erfolgt.

Die Autoren unterscheiden dazu zwischen In-Character- und Out-of-Character-Kommunikation. In-Character beschreibt Äußerungen, die innerhalb der fiktiven Welt und aus der Perspektive der Figur erfolgen. Out-of-Character bezeichnet dagegen Regel-, Strategie- oder Meta-Kommunikation außerhalb der Rolle (Callison-Burch, et al., 2022, S. 4). Für ein NPC-System ist diese Unterscheidung zentral, weil NPCs normalerweise nicht

über technische Systeminformationen, Prompting, KI-Modelle oder interne Spielmechaniken sprechen sollen. Sie müssen Antworten so formulieren, dass sie innerhalb der Spielwelt plausibel bleiben.

2.3. Wissensbegrenzung und Charakterkonsistenz

Ein zentrales Qualitätsmerkmal KI-gestützter NPC-Dialoge ist Charakterkonsistenz. Ein NPC muss nicht nur formal richtige Sätze bilden, sondern innerhalb seiner Rolle sprechen, handeln und wissen. Warpefelt und Verhagen (2017) knüpfen Believability von NPCs an die wahrgenommene Glaubwürdigkeit ihrer Darstellung innerhalb der Spielwelt. Relevant sind dabei unter anderem Rationalität, Intentionalität, sozialer Kontext und die Fähigkeit, eine Figur so darzustellen, dass Spielende sie als diese Rolle wahrnehmen können (Warpefelt und Verhagen, 2017, S. 40–42). Für LLM-gestützte NPCs folgt daraus, dass nicht die technische Komplexität des Modells allein über Glaubwürdigkeit entscheidet, sondern ob Antworten innerhalb der Spielwelt kohärent, rollenpassend und nachvollziehbar bleiben.

Loyall (1997) liefert für diesen Punkt eine ältere, aber weiterhin anschlussfähige Grundlage zu believable agents. Er betont Aspekte wie Persönlichkeit, Emotion, Selbstmotivation, Veränderung, soziale Beziehungen, Konsistenz des Ausdrucks und Illusion of Life (Loyall, 1997, S. 15 ff.). Auf NPC-Dialoge übertragen bedeutet dies: Eine Figur wirkt nicht dadurch glaubwürdig, dass sie möglichst viele Informationen liefert, sondern dadurch, dass ihre Äußerungen zu Persönlichkeit, Motivation, sozialer Einbettung und Situation passen. Gerade bei LLM-basierten Dialogen ist diese Perspektive relevant, weil ein Sprachmodell zwar viele sprachlich plausible Antworten erzeugen kann, diese aber ohne Rollenbindung nicht automatisch als glaubwürdige Figurenrede erscheinen.

Liu et al. (2025) verstehen character-consistent dialogue generation als Herausforderung, bei der generierte Antworten mit den Einstellungen, Erfahrungen, dem Wissen und der Persönlichkeit einer Figur übereinstimmen müssen. Die Autoren zeigen, dass insbesondere begrenzte Memory-Fähigkeiten und inkonsistente Charakterrepräsentationen die Glaubwürdigkeit von NPC-Dialogen beeinträchtigen können (Liu, et al., 2025,

S. 1). Charakterkonsistenz ist damit nicht nur eine Frage des Sprachstils, sondern betrifft auch Wissen, Erinnerung, Motivation und die Grenzen der Figurperspektive.

Wissensbegrenzung bezeichnet hier die gezielte Einschränkung der Informationen, auf die ein NPC zurückgreifen darf. Sie umfasst objektive Spielweltinformationen und die subjektive Figurenperspektive: Ein NPC darf nur nutzen, was im Spiel existiert und was seine Rolle, Erfahrung und Situation plausibel zulassen.

Damit wird die Trennung zwischen Spielerwissen, Modellwissen und Charakterwissen zentral. Für den Prototyp benötigt jeder NPC ein definiertes Wissensprofil: bekannte Fallinformationen, erlaubte Hinweise, selbst erlebte Ereignisse und Situationen, in denen er ausweichen, Unsicherheit zeigen oder Nichtwissen ausdrücken muss.

Liu et al. (2025) schlagen für dieses Problem ein Framework vor, das statisches Charakterwissen mit dynamischem Memory verbindet. Statisches Wissen umfasst Eigenschaften wie Persönlichkeit, Biografie und zentrale Erfahrungen einer Figur. Dynamisches Wissen entsteht durch Interaktionen mit Spielenden und wird über Knowledge Graphs und Vektordatenbanken verwaltet (Liu, et al., 2025, S. 3–4). Besonders relevant ist das von den Autoren beschriebene protective static knowledge fine-tuning. Dabei soll das Modell lernen, bei Fragen außerhalb des kognitiven Bereichs einer Figur nicht einfach sachlich zu antworten, sondern passend zur Rolle Unsicherheit, Verwirrung oder Ablehnung auszudrücken (Liu, et al., 2025, S. 5–6).

Das Fine-Tuning wird hier nicht nachgebaut; übernommen wird das Prinzip. Ein glaubwürdiger NPC soll nicht maximal informativ, sondern rollen- und wissenskonform antworten. Gerade im Krimi-Kontext ist kontrolliertes Nichtwissen wichtig, weil Informationen nicht von allen Figuren vollständig preisgegeben werden dürfen.

Die von Callison-Burch et al. (2022) untersuchte D&D-Domäne unterstützt diese Sichtweise. Dort können Kontrollmerkmale wie Spieler-ID, Charaktername, Klasse, Spezies, Inventar, aktuelle Aktion oder Kampfstatus verwendet werden, um die Antwortgenerierung zu beeinflussen (Callison-Burch, et al., 2022, S. 4–5). Solche Merkmale machen

deutlich, dass eine plausible Antwort nicht nur vom unmittelbar vorherigen Satz abhängt, sondern von einer Kombination aus Figur, Situation, bisherigem Verlauf und aktuellem Spielzustand. Charakterkonsistenz ist daher eng mit State-Bezug und Memory verbunden.

Memory bezeichnet gespeicherte Informationen über frühere Interaktionen, Ereignisse oder relevante Spielzustände, die spätere Antworten beeinflussen können. Es darf jedoch weder falsche Informationen stabilisieren noch unbegrenzt wirken. Dadurch verbindet Memory Dialogqualität, Charakterkonsistenz und technische Kontextsteuerung.

2.4. Technische Ansätze zur Kontrolle von NPC-Dialogen

Die betrachteten Arbeiten zeigen verschiedene technische Strategien, um generative Dialoge kontrollierbarer zu machen. Ein zentraler Ansatz ist die Verwendung strukturierter Kontextdaten. Ashby et al. (2023) verwenden einen Knowledge Graph, der Informationen über NPCs, Orte, Objekte und Beziehungen der Spielwelt enthält. Spielereingaben werden mit möglichen Pfaden im Knowledge Graph verglichen, daraus wird ein Quest-Seed abgeleitet und anschließend durch Templates sowie ein Sprachmodell in Questtitel und NPC-Dialog überführt (Ashby, et al., 2023, S. 2–7). Der auf Seite 7 dargestellte Knowledge Graph zeigt exemplarisch, wie Entitäten wie NPCs, Ressourcen, Orte und Gegner über gerichtete Relationen verbunden werden.

Der Nutzen dieses Ansatzes liegt darin, dass generierter Text nicht vollständig frei entsteht, sondern an vorhandene Weltinformationen gebunden wird. Ashby et al. (2023) verfolgen das Ziel, Inhalte zu erzeugen, die sowohl auf Spielereingaben reagieren als auch mit dem Weltzustand vereinbar bleiben. Der Knowledge Graph wirkt dabei als externe Repräsentation der Spielwelt. Er ersetzt nicht das Sprachmodell, sondern liefert strukturierte Ankerpunkte, an denen die generierte Antwort ausgerichtet werden kann. Für den eigenen Prototyp lässt sich daraus ableiten, dass NPCs nicht nur mit einem allgemeinen Systemprompt gesteuert werden sollten, sondern mit expliziten Daten zu Rolle, Wissen, Beziehungen, Ort, Fallstatus und bekannten Hinweisen.

Liu et al. (2025) erweitern diesen Gedanken durch die Trennung von statischem und dynamischem Wissen. Statisches Wissen umfasst grundlegende Charakterdaten wie Persönlichkeit, Biografie und zentrale Erfahrungen. Dynamisches Wissen umfasst Interaktionsverläufe, wichtige Ereignisse und Beziehungen, die im Spielverlauf entstehen (Liu, et al., 2025, S. 4). Auf Seite 5 zeigen die Autoren ein Framework, das statische Wissenskomponenten, dynamische Knowledge-Graph-Strukturen, AMR-basierte Eingabeverarbeitung und einen Retrieval-Schritt aus einer Vektordatenbank verbindet. Für diese Arbeit ist daran weniger die vollständige technische Komplexität relevant, sondern die Grundidee, NPC-Antworten aus mehreren kontrollierten Kontextquellen zusammenzusetzen.
Retrieval-Ansätze wie Retrieval-Augmented Generation (RAG) sind für diese Arbeit vor allem als Prinzip relevant. Lewis et al. (2021) beschreiben RAG als Verbindung eines generativen Modells mit einem Retrieval-Mechanismus über eine externe Wissensquelle. Das Modell muss das benötigte Wissen dadurch nicht ausschließlich in seinen Parametern gespeichert haben, sondern kann zur Inferenzzeit auf eine externe Wissensbasis zugreifen (Lewis, et al., 2021, S. 1–3). Für wissensintensive Aufgaben ist dies relevant, weil Wissensbestände aktualisiert, eingeschränkt oder gezielt ausgewählt werden können, ohne das generative Modell selbst neu zu trainieren.

Für NPC-Dialoge bedeutet das RAG-Prinzip, relevantes Spielwissen abhängig von Eingabe, NPC, State und Thema auszuwählen, statt den gesamten Fallkontext dauerhaft in den Prompt zu schreiben. Der Prototyp nutzt keine vollständige Vektordatenbank, übernimmt aber dieses Auswahlprinzip für NPC- und Fallinformationen.

Constraints sind explizite Regeln, die den Antwortbereich eines NPCs begrenzen: keine internen Systeminformationen, keine unbekannten Fallinformationen, keine erfundenen Orte oder Hinweise und rollengerechtes Ausweichen bei fehlendem Wissen. Sie verbinden Qualitätsanforderungen mit Prompt- und Systemgestaltung.

Callison-Burch et al. (2022) zeigen mit ihren control features, dass solche Steuerinformationen auch ohne vollständige autonome Spielsimulation hilfreich sein können. Kon-

trollmerkmale können beeinflussen, ob ein Modell als bestimmte Figur, in einer bestimmten Situation oder innerhalb beziehungsweise außerhalb der fiktionalen Welt antwortet (Callison-Burch, et al., 2022, S. 4–7). Gleichzeitig zeigt ihre Untersuchung, dass Game State Tracking schwierig bleibt. Das Modell erreicht zwar bessere Ergebnisse als eine einfache Baseline, die vollständige gemeinsame Zustandsgenauigkeit bleibt jedoch begrenzt (Callison-Burch, et al., 2022, S. 7).

Für das eigene Systemkonzept folgt daraus: Der aktuelle Spielzustand darf nicht allein aus dem Chatverlauf rekonstruiert werden, sondern muss explizit übergeben werden. Nur so kann das System kontrollieren, welche Hinweise gefunden wurden, welche Gespräche stattfanden und welches Wissen aktuell erlaubt ist.

Der Ansatz bleibt pragmatisch: explizite NPC-Profile, begrenztes Charakterwissen, ausgewählte State-Informationen, einfacher Memory-Bezug und klare Constraints. Das LLM wird dadurch nicht zur alleinigen Wissensquelle, sondern zum sprachlichen Generierungsmodul eines begrenzten Systems.

2.5. Zwischenfazit für die eigene Arbeit

Der Forschungsstand zeigt, dass LLM-basierte NPC-Dialoge flexible Interaktionen ermöglichen, aber kontrolliert werden müssen. Zentrale Risiken sind Halluzinationen, Rollenbruch, fehlende Wissensbegrenzung, unzureichender State-Bezug und unscharfe Grenzen zwischen Spielwelt und Meta-Kommunikation. Daraus entsteht der Qualitätsrahmen des nächsten Kapitels.

3. Qualitätsrahmen für KI-gestützte NPC-Dialoge

3.1. Ziel und Funktion des Qualitätsrahmens

Qualität KI-gestützter NPC-Dialoge lässt sich nicht allein über sprachliche Flüssigkeit bestimmen. Eine Antwort ist nur dann geeignet, wenn sie zur Spielwelt, zur Figurenrolle, zum aktuellen State und zur spielerischen Funktion passt.

Der Rahmen ist auf den Krimi-Prototyp zugeschnitten, in dem Dialoge Hinweise dosieren, Gerüchte oder Ausweichverhalten tragen und die Fallstruktur schützen. Zu frühe, falsche oder zu sichere Antworten können hier den Erkenntnisfortschritt direkt beschädigen.

Der Qualitätsrahmen ist daher kein allgemeingültiges Modell für alle KI-NPCs, sondern ein domänenspezifisches Bewertungsraster für den entwickelten Prototyp. Allgemeine Dialogqualitätsansätze werden auf den Fall einer Krimi-Spielwelt übertragen.

Die Grundlage dafür bilden sowohl allgemeine Arbeiten zur Dialogqualität als auch spezifische Arbeiten zu NPCs und LLM-basierten Spielsystemen. Warpefelt und Verhagen beschreiben NPCs als wichtigen Bestandteil der Game Experience, weil sie Hilfe, Herausforderungen, Dienste und narrative Anbindung bereitstellen und dafür erwartungskonform sowie glaubwürdig handeln müssen (Warpefelt und Verhagen, 2017, S. 39). Marconi et al. fassen Interaktionsqualität in Human-AI-Dialogen als mehrdimensionales Konstrukt auf, das unter anderem pragmatische Qualität, soziale Wirkung, Transparenz und Nutzerwahrnehmung umfasst (Marconi, et al., 2026, S. 1–2). Für diese Arbeit wird daraus ein domänenspezifischer Qualitätsrahmen abgeleitet, der die Besonderheiten von NPC-Dialogen in einer Krimi-Spielwelt berücksichtigt.

3.2. Ableitung der Qualitätskriterien aus dem Forschungsstand

Die Qualitätskriterien werden nicht frei erfunden, sondern aus den in Kapitel 2 dargestellten Quellen abgeleitet. Für NPCs ist zunächst entscheidend, dass sie als Figuren innerhalb der Spielwelt wahrgenommen werden. Warpefelt und Verhagen betonen, dass NPCs nicht nur als technische Funktionsträger auftreten, sondern ihre Rolle glaubwürdig

darstellen müssen. Sie sprechen in diesem Zusammenhang von characterhood: NPCs sollen so handeln, dass Spielende ihre Rolle, Absicht und Einbettung in die Spielwelt erkennen können (Warpefelt und Verhagen, 2017, S. 40). Daraus folgt für den Qualitätsrahmen das Kriterium der Charakterkonsistenz.

Gleichzeitig zeigen Arbeiten zu LLMs in Spielen, dass generative Systeme zwar flexible und dynamische Interaktionen ermöglichen, aber auch unvorhersehbare oder inkonsistente Inhalte erzeugen können. Sweetser ordnet LLMs in Videospielen als wachsendes Forschungsfeld ein und verweist zugleich auf Risiken wie begrenzte logische Schlussfolgerung, Unvorhersehbarkeit und Qualitätsunterschiede generierter Inhalte (Sweetser, 2024, S. 6–7). Ashby et al. zeigen am Beispiel personalisierter Quest- und Dialoggenerierung, dass generierte Dialoge an Weltwissen gebunden werden müssen, weil erfundene Details Spielende sonst in eine nicht vorhandene Richtung lenken können (Ashby, et al., 2023, S. 1–2). Daraus folgen das Qualitätskriterium der Welt- und State-Konsistenz sowie die Fehlerkategorie Halluzination.

Für den spezifischen Umgang mit Figurenwissen ist die Arbeit von Liu et al. zentral. Die Autoren beschreiben character hallucination als Problem, bei dem ein Modell Antworten erzeugt, die nicht zum Charakter oder zu dessen kognitivem Wissensbereich passen. Gleichzeitig unterscheiden sie zwischen statischem Charakterwissen und dynamischem Memory, das sich aus Interaktionen mit Spielenden ergibt (Liu, et al., 2025, S. 1–4). Daraus folgen die Kriterien Wissensbegrenzung und Memory-Korrektheit.

Callison-Burch et al. verdeutlichen am Beispiel von Dungeons & Dragons, dass rollenspielartige Dialoge stark vom Spielzustand, von Figurenmerkmalen und von der Unterscheidung zwischen In-Character- und Out-of-Character-Kommunikation abhängen. Sie nutzen Kontrollmerkmale wie Character Name, Class, Race, Actions, Combat und In-Character/Out-of-Character, um die Antwortgenerierung zu beeinflussen (Callison-Burch, et al., 2022, S. 4–7). Für den Qualitätsrahmen stützt dies die Kriterien State-Bezug, Rollenbindung und die Trennung zwischen fiktionaler und außerfiktionaler Kommunikation.

Ergänzend zeigen Arbeiten zur Dialogevaluation, dass Antwortqualität in Dialogsystemen mehrdimensional bewertet werden sollte. Yi et al. schlagen turn-level feedback für Kohärenz und Engagement vor und nutzen dabei Fragen danach, ob eine Antwort verständlich, thematisch passend, interessant und anschlussfähig ist (Yi, et al., 2019, S. 67–69). Marconi et al. beschreiben Interaktionsqualität als Zusammenspiel aus technischer Leistungsfähigkeit und Kontextpassung, also nicht nur als Fähigkeit zu antworten, sondern als Fähigkeit, passend zu Situation, Nutzendenziel und Erwartung zu antworten (Marconi, et al., 2026, S. 2–6). Liu et al. (2016) zeigen zudem, dass automatische Wortüberlappungsmetriken wie BLEU, METEOR oder ROUGE bei offenen Dialogantworten nur schwach oder gar nicht mit menschlichen Qualitätsurteilen korrelieren können (Liu, et al., 2016, S. 2122–2125). Daraus folgt, dass die Qualität der NPC-Antworten kontextsensitiv und kriteriengeleitet bewertet werden muss.

Die Kriterien sind analytisch getrennt, können in einer Antwort aber gleichzeitig betroffen sein. Eine halluzinierte Information kann etwa Wissensbegrenzung, State-Konsistenz und spielerische Sinnhaftigkeit zugleich verletzen.

Qualitätskriterium	Kernfrage	Zentraler Quellenaspekt	Ableitung für den Krimi-Prototyp
Charakterkonsistenz	Passt die Antwort zur Figur?	characterhood; character hallucination; Rollen- und Persönlichkeitsbindung	NPC-Antworten müssen zu Rolle, Haltung, Motivation und Sprachstil passen.
Wissensbegrenzung	Nutzt der NPC nur erlaubtes Wissen?	statisches Charakterwissen; Wissensbereich der Figur; Knowledge Grounding; In-Character/Out-of-Character-Trennung	NPCs dürfen nur Informationen nutzen, die ihnen im Fall und im aktuellen State zustehen.
Welt- und State-Konsistenz	Passt die Antwort zur Spielwelt und zum aktuellen Zustand?	strukturierte Weltinformationen; Knowledge Graph; Game State Tracking; Unvorhersehbarkeit generativer Systeme	Antworten müssen zu Orten, Hinweisen, Ereignissen und freigeschaltetem Fallfortschritt passen.
Memory-Korrektheit	Berücksichtigt der NPC frühere Interaktionen korrekt?	dynamisches Memory; Interaktionsgeschichte; Kontextpassung	Frühere relevante Gespräche dürfen weder vergessen noch falsch erinnert werden.
Umgang mit Unsicherheit und Nichtwissen	Reagiert der NPC angemessen bei fehlender Wissensgrundlage?	Halluzination; protective knowledge boundaries; Transparenz und angemessene Unsicherheit	NPCs sollen bei fehlendem Wissen nicht selbstsicher erfinden, sondern rollengerecht ausweichen.
Spielerische Sinnhaftigkeit	Hilft die Antwort dem Spielverlauf, ohne den Fall zu zerstören?	Game Experience; Kohärenz; Anschlussfähigkeit; task effectiveness; kontrollierte Informationsvergabe	Antworten sollen Orientierung, Spannung und sinnvolle Anschlussinteraktion ermöglichen.

Tabelle 1 Ableitung der Qualitätskriterien aus dem Forschungsstand. Quelle: Vom Verfasser erstellte Tabelle

3.3. Qualitätskriterium 1: Charakterkonsistenz

Charakterkonsistenz beschreibt, ob eine NPC-Antwort zu Rolle, Persönlichkeit, sozialer Position, Motivation und Sprache der Figur passt. Bewertet wird, ob die Antwort in dieser Situation plausibel von genau dieser Figur stammen könnte.

Dieses Kriterium ist relevant, weil NPCs nach Warpefelt und Verhagen ihre Rolle innerhalb der Spielwelt glaubwürdig darstellen müssen. Ihre Glaubwürdigkeit hängt davon ab, ob Spielende die Figur als Rolle innerhalb der Spielwelt erkennen und ihr Verhalten als erwartungskonform wahrnehmen können (Warpefelt und Verhagen, 2017, S. 40–41). Ergänzend zeigen Liu et al., dass LLM-basierte NPC-Systeme zu character hallucination neigen können, wenn Antworten nicht ausreichend an Charakterwissen und Rollenprofil gebunden werden (Liu, et al., 2025, S. 1–3).

Für den Prototyp bedeutet dies: Jedes Rollenprofil legt Funktion im Fall, soziale Position, Grundhaltung, Sprachstil und Wissensgrenzen fest. Bewertet wird nicht allgemeine Glaubwürdigkeit, sondern die Passung zur definierten Figur. Probleme entstehen, wenn ein NPC wie ein neutraler Assistent wirkt, unpassendes Expertenwissen äußert oder seine Persönlichkeit unerklärlich wechselt.

Bewertungsfrage: Spricht der NPC so, wie diese Figur in dieser Situation plausibel sprechen würde?

3.4. Qualitätskriterium 2: Wissensbegrenzung

Wissensbegrenzung bezeichnet die Einschränkung der Informationen, die ein NPC nutzen darf. Eine Antwort ist wissensbegrenzt, wenn sie nur Wissen verwendet, das Rolle, Vorgeschichte, Fallposition und aktueller Spielverlauf plausibel erlauben.

Liu et al. unterscheiden in ihrem Framework zwischen statischem Charakterwissen und dynamischem Wissen aus Interaktionen. Zugleich betonen sie, dass der Wissensbereich einer Figur kontrolliert werden muss, damit das Modell nicht außerhalb des kognitiven Rahmens der Figur antwortet (Liu, et al., 2025, S. 1–6). Callison-Burch et al. zeigen im

Rollenspielkontext, dass Dialoge stark an Figurenmerkmale, In-Character-/Out-of-Character-Unterscheidungen und Spielzustände gebunden sind (Callison-Burch, et al., 2022, S. 1–5). Ashby et al. stützen diese Perspektive durch ihren Knowledge-Graph-Ansatz, der generierte Inhalte mit strukturiertem Weltwissen verbindet (Ashby, et al., 2023, S. 1–2).

Im Krimi-Prototyp ist diese Begrenzung zentral, weil Zeugen, Verdächtige und Täterfigur unterschiedliche Wissensstände besitzen. Täterwissen, geheime Motive, noch nicht gefundene Hinweise und interne Falllogik dürfen nur erscheinen, wenn Profil und State dies zulassen.

Für die Bewertung ist zwischen grundsätzlich plausiblem Charakterwissen und aktuell freigegebenem Wissen zu unterscheiden. Nichtwissen kann korrektes Verhalten sein, wenn eine Information aus Rollen- oder State-Gründen nicht ausgegeben werden darf. Technisch wird dies durch NPC-Profile, freigegebene Wissenseinträge und Constraints abgebildet.

Bewertungsfrage: Verwendet der NPC nur erlaubtes Charakterwissen?

3.5. Qualitätskriterium 3: Welt- und State-Konsistenz

Welt- und State-Konsistenz beschreibt, ob eine Antwort zur Spielwelt und zum aktuellen Fallzustand passt. Im Prototyp umfasst State bekannte Hinweise, geführte Dialoge, freigegebene Wissenseinträge, Fallfortschritt, Verdachtsstände und NPC-spezifische Freigaben. Fehlerhaft ist auch eine grundsätzlich mögliche Information, wenn sie im konkreten Spielzustand zu früh, zu spät oder widersprüchlich erscheint.

Ashby et al. zeigen, dass generierte Quest- und NPC-Dialoge an strukturierte Weltinformationen gebunden werden können, um in-game integrity zu erhalten. Ihr Knowledge-Graph-Ansatz dient dazu, Weltobjekte, Orte, Figuren und Beziehungen als Grundlage für Generierung zu nutzen (Ashby, et al., 2023, S. 1–7). Callison-Burch et al. zeigen am Beispiel von Dungeons & Dragons, dass Game State Tracking und Kontrollmerkmale für plausible Dialogausgaben relevant sind (Callison-Burch, et al., 2022, S. 1–7).

Für den Prototyp bedeutet dies, dass NPCs weder nicht existierende Orte nennen, noch auf unentdeckte Hinweise reagieren oder Ereignisse erwähnen dürfen, die im aktuellen Spielstand nicht ausgelöst wurden.

Bewertungsfrage: Passt die Antwort zum aktuellen Spielzustand und zur etablierten Spielwelt?

3.6. Qualitätskriterium 4: Memory-Korrektheit

Memory-Korrektheit beschreibt, ob ein NPC frühere relevante Dialoge oder Ereignisse korrekt berücksichtigt. Eine Antwort ist memory-korrekt, wenn der NPC relevante vorherige Interaktionen wiederaufgreift, ohne falsche Erinnerungen zu erzeugen, falsche Gesprächsinhalte zu stabilisieren oder wichtige bereits bekannte Informationen zu vergessen.

Liu et al. begründen die Relevanz von Memory für personalisierte und charakterkonsistente NPC-Dialoge. Ihr Framework verbindet statisches Charakterwissen mit dynamischem Memory und nutzt Knowledge Graphs sowie Vektorspeicher, um Interaktionsgeschichte zu speichern und abrufbar zu machen (Liu, et al., 2025, S. 1–4, 12–13). Callison-Burch et al. zeigen ergänzend, dass rollenspielartige Dialoge stark history-dependent sind und dass Dialoghistorie und State-Informationen für plausible nächste Äußerungen wichtig sind (Callison-Burch, et al., 2022, S. 1–2).

Für den Prototyp bedeutet Memory-Korrektheit nicht, dass jede einzelne Äußerung dauerhaft gespeichert werden muss. Relevant sind vor allem drei Formen von Memory: Dialog-Memory, also frühere Aussagen im Gespräch mit einem bestimmten NPC; Fall-Memory, also bereits bekannte oder gezeigte Hinweise; und Beziehungs- beziehungsweise Interaktions-Memory, also relevante Handlungen, Anschuldigungen oder Haltungsänderungen zwischen Spielenden und NPC. Diese Formen überschneiden sich teilweise mit State-Konsistenz, werden aber gesondert betrachtet, weil sie auf die Erinnerung und Wiederaufnahme früherer Interaktionen zielen.

Entscheidend ist, dass fallrelevante oder beziehungsrelevante Informationen korrekt erhalten bleiben. Wenn Spielende einem NPC bereits einen Hinweis gezeigt haben, sollte dies bei späteren Gesprächen berücksichtigt werden. Umgekehrt darf der NPC nicht behaupten, ein Gespräch habe stattgefunden, wenn dies nicht der Fall war. Ein fehlerhaftes Memory kann problematischer sein als fehlendes Memory, weil falsche Informationen dadurch in späteren Dialogen stabilisiert und für Spielende als scheinbar verlässlicher Fallkontext präsentiert werden.

Bewertungsfrage: Erinnert sich der NPC korrekt an relevante vorherige Interaktionen?

3.7. Qualitätskriterium 5: Umgang mit Unsicherheit und Nichtwissen

Der Umgang mit Unsicherheit und Nichtwissen beschreibt, wie ein NPC reagiert, wenn ihm eine Wissensgrundlage fehlt oder er etwas nicht preisgeben darf. Qualitativ gut ist rollengerechtes Ausweichen, Unsicherheit, Nachfrage, Nichtwissen oder klar markierte Vermutung statt selbstsicherer Halluzination.

In der allgemeinen LLM-Forschung wird Halluzination als Generierung plausibel formulierter, aber nicht durch Kontext oder Daten gedeckter Inhalte verstanden (Ji, et al., 2023, S. 1). Für NPC-Dialoge ist dieses Problem besonders relevant, weil erfundene Inhalte nicht nur faktisch falsch sein können, sondern die Spielweltlogik oder Fallstruktur verändern. Liu et al. zeigen mit protective static knowledge fine-tuning ein Prinzip, bei dem Figuren lernen sollen, außerhalb ihres Wissensbereichs nicht einfach sachlich zu antworten, sondern Unwissen oder Verwirrung passend zur Figur zu zeigen (Liu, et al., 2025, S. 5–7). Marconi et al. stützen dies aus Sicht der Human-AI-Dialogqualität, weil Qualität nicht allein aus Antwortfähigkeit besteht, sondern aus der Passung zwischen Fähigkeit, Kontext, Erwartung und Transparenz (Marconi, et al., 2026, S. 2–6).

Für die Bewertung wird zwischen Nichtwissen, Unsicherheit, Verweigerung und Gerücht unterschieden. Alle können im Krimi-Kontext passend sein, wenn sie zur Figur, zur Informationslage und zur dramaturgischen Funktion passen und nicht als sichere Tatsache erscheinen.

Für den Krimi-Prototyp ist Nichtwissen kein Defekt, sondern ein zentrales Qualitätsverhalten. Entscheidend ist, dass Nichtwissen nicht mit Inhaltsarmut verwechselt wird. Eine rollengerechte Verweigerung, eine vorsichtige Vermutung oder der Verweis auf eine andere Figur kann qualitativ hochwertig sein, wenn dadurch die Wissensgrenzen gewahrt bleiben und zugleich eine sinnvolle Anschlussinteraktion entsteht. Problematisch ist dagegen eine selbstsichere Antwort ohne Grundlage, insbesondere wenn dadurch Täterwissen, falsche Hinweise oder nicht vorhandene Fakten entstehen.

Bewertungsfrage: Reagiert der NPC angemessen, wenn er etwas nicht wissen darf oder nicht wissen kann?

3.8. Qualitätskriterium 6: Spielerische Sinnhaftigkeit

Spielerische Sinnhaftigkeit bezeichnet die Funktion einer Antwort im Krimi-Prototyp. Sie ist gegeben, wenn eine Antwort Orientierung bietet, Spannung erhält, Anschlussinteraktion ermöglicht und weder falsche noch zu früh auflösende Informationen erzeugt.

Warpefelt und Verhagen beschreiben NPCs als Teil der Game Experience, da sie unter anderem Hilfe, Herausforderungen, Dienste und narrative Anbindung bereitstellen (Warpefelt und Verhagen, 2017, S. 39). Yi et al. zeigen für Dialogevaluation, dass Antworten nicht nur verständlich und thematisch passend, sondern auch interessant und anschlussfähig sein sollten; insbesondere die Frage, ob eine Konversation sinnvoll fortgesetzt werden kann, ist für interaktive Systeme relevant (Yi, et al., 2019, S. 67–69). Marconi et al. ergänzen dies durch die Perspektive von task effectiveness und conversational competence als Bestandteile pragmatischer Dialogqualität (Marconi, et al., 2026, S. 2–6).

Eine Antwort wird daher nicht nur an Korrektheit gemessen. Sie kann Hinweise andeuten, Nachfragen provozieren, Spuren einordnen oder Atmosphäre stützen; problematisch wird sie, wenn sie den Fall auflöst, keine Orientierung bietet oder spielerisch ins Leere läuft.

Spielerische Sinnhaftigkeit wird testfallbezogen bewertet. Eine Antwort ist nicht abstrakt gut oder schlecht, sondern abhängig davon, welche Funktion sie in der jeweiligen Situation erfüllen soll: Orientierung geben, eine Nachfrage ermöglichen, eine Spur stützen, Spannung erhalten oder eine unzulässige Auflösung vermeiden.

Bewertungsfrage: Hilft die Antwort dem Spielverlauf, ohne zu viel oder falsche Informationen zu geben?

3.9. Fehlerkategorien für die spätere Evaluation

Für die Evaluation werden ergänzend konkrete Fehlerkategorien definiert. Sie machen sichtbar, ob Probleme vor allem als Rollenbruch, unzulässiges Wissen, Halluzination, Meta-Bruch, State-Fehler, Memory-Fehler oder spielerische Schwäche auftreten.

Die Bewertung folgt bewusst keinem rein automatisierten Dialogmaß, da solche Maße die spezifischen Anforderungen eines NPC-Dialogs im Krimi-Kontext nur begrenzt abbilden können. Liu et al. (2016) zeigen, dass verbreitete automatische Metriken für offene Dialogantworten nur schwach oder gar nicht mit menschlichen Qualitätsurteilen korrelieren können (Liu, et al., 2016, S. 2122–2125). Für den Prototyp ist nicht allein relevant, ob eine Antwort sprachlich flüssig oder einer erwarteten Musterantwort ähnlich ist. Entscheidend ist, ob sie zur Figur, zum erlaubten Wissen, zum aktuellen Spielzustand und zur dramaturgischen Informationsvergabe des Falls passt.

Die Rubrik ist ein manuelles, aber nachvollziehbares Bewertungsinstrument. Nachvollziehbarkeit entsteht durch definierte Testfälle, anwendbare Kriterien, 0-2-Skala, Fehlercodes und kurze Bewertungsbegründungen.

Bewertungseinheit ist jeweils eine einzelne NPC-Antwort innerhalb eines definierten Testfalls. Ergänzend kann pro Testfall betrachtet werden, ob sich über mehrere Dialogturns hinweg wiederkehrende Fehler zeigen. Dadurch bleibt die Bewertung übersichtlich, ohne mehrturnige Probleme wie Memory-Fehler oder wiederholte State-Inkonsistenzen auszublenden.

Die Bewertung erfolgt kriterienbezogen: Für jeden Testfall werden anwendbare Kriterien mit 0, 1 oder 2 Punkten bewertet und auftretende Fehlercodes markiert. Kritische Fehler wie Halluzination, unzulässige Informationsgabe oder Meta-Bruch werden zusätzlich vermerkt, weil sie die Spielweltlogik auch bei sonst hoher Punktzahl beschädigen können.

Nicht jedes Kriterium muss in jedem Testfall gleich relevant sein. Für die Evaluation wird deshalb pro Testfall festgelegt, welche Kriterien geprüft werden. Kriterien, die im jeweiligen Testfall nicht sinnvoll bewertbar sind, werden als „nicht anwendbar“ markiert. Der Maximalwert ergibt sich aus der Anzahl der tatsächlich bewerteten Kriterien; bei sechs bewerteten Kriterien sind maximal 12 Punkte erreichbar.
Kritische Fehler werden zusätzlich zum Punktwert markiert, weil sie die Fallstruktur oder die Illusion der Spielwelt unmittelbar beschädigen können. Besonders F1 Halluzination, F2 Spoiler beziehungsweise unzulässige Informationsgabe und F4 Meta-Bruch gelten als kritisch. Eine Antwort kann daher nicht allein aufgrund eines hohen Punktwerts als unproblematisch gelten, wenn sie zugleich einen kritischen Fehler enthält.

Code	Fehlerkategorie	Beschreibung	Kritisch?
F1	Halluzination	Der NPC erfindet Figuren, Orte, Hinweise, Ereignisse oder Fakten.	Ja
F2	Spoiler / unzulässige Informationsgabe	Der NPC verrät Informationen, die er nicht wissen oder noch nicht preisgeben darf.	Ja
F3	Rollenbruch	Der NPC verlässt seine Figur, seinen Ton oder seine soziale Rolle.	Nein
F4	Meta-Bruch	Der NPC spricht über KI, Prompts, Unity, API, Spielsystem oder interne Regeln.	Ja
F5	State-Fehler	Der NPC ignoriert den aktuellen Spielzustand oder verweist auf falsche beziehungsweise noch nicht freigeschaltete Zustände.	Je nach Ausprägung
F6	Memory-Fehler	Der NPC erinnert sich falsch, vergisst relevante Informationen oder übernimmt falsche Gesprächsinhalte.	Je nach Ausprägung
F7	Spielerisch unbrauchbare Antwort	Die Antwort ist zu vage, zu allgemein, nicht hilfreich oder zerstört Spannung beziehungsweise Orientierung.	Nein, außer fallauflösend

Tabelle 2 Fehlerkategorien für die spätere Evaluation der NPC-Antworten. Quelle: Vom Verfasser erstellte Tabelle

Die Bewertung nutzt eine einfache 0-2-Skala: zwei Punkte für klare Erfüllung, ein Punkt für teilweise Erfüllung und null Punkte für deutliche Verletzung. Nicht anwendbare Kriterien zählen nicht in den Maximalwert.

Kriterium	0 Punkte	1 Punkt	2 Punkte
Charakterkonsistenz	Rollenbruch oder unpassender Ton.	Grundsätzlich passend, aber mit leichten Ton- oder Rollenproblemen.	Sprache, Rolle und Haltung passen klar zum NPC.
Wissensbegrenzung	NPC nutzt eindeutig unerlaubtes Wissen.	Kleine Unsicherheit, ob Wissen plausibel verfügbar oder freigegeben ist.	NPC nutzt nur plausibel erlaubtes und freigegebenes Charakterwissen.
Welt-/State-Konsistenz	Antwort widerspricht Welt oder State deutlich.	Leichte Unschärfe im Bezug auf aktuellen Zustand.	Antwort passt klar zu Welt, Fallzustand und Ereignissen.
Memory-Korrektheit	Falsche Erinnerung oder relevantes Vergessen.	Teilweise korrekter Bezug, aber ungenau.	Relevante frühere Interaktion korrekt berücksichtigt.
Umgang mit Nichtwissen	NPC halluziniert selbstsicher.	NPC weicht aus, aber noch unklar oder unpassend.	NPC zeigt rollengerechtes Nichtwissen, Unsicherheit, Verweigerung oder markierte Vermutung.
Spielerische Sinnhaftigkeit	Antwort ist nutzlos, zerstört Spannung oder spoilert.	Antwort ist nutzbar, aber wenig hilfreich oder zu allgemein.	Antwort unterstützt Orientierung, Spannung und Anschlussinteraktion.

Tabelle 3 Bewertungsrubrik für einzelne NPC-Antworten. Bei sechs bewerteten Kriterien sind maximal 12 Punkte pro Antwort erreichbar; bei nicht anwendbaren Kriterien reduziert sich der Maximalwert entsprechend. Quelle: Vom Verfasser erstellte Tabelle

3.10. Zwischenfazit

Der Qualitätsrahmen macht die Anforderungen aus Kapitel 2 prüfbar. Die sechs Kriterien und Fehlerkategorien fokussieren Rolle, Wissen, State, Memory, Nichtwissen und spielerische Funktion.

4. Methodik

Dieses Kapitel beschreibt das methodische Vorgehen der Arbeit. Es legt fest, wie der Unity-Prototyp anhand definierter Testfälle evaluiert wird und wie die erzeugten NPC-Antworten mit dem Qualitätsrahmen aus Kapitel 3 bewertet werden.

4.1. Forschungsdesign der Arbeit

Die Arbeit folgt einem prototypenbasierten Forschungsdesign. Ausgehend vom Forschungsstand werden zentrale Probleme KI-gestützter NPC-Dialoge herausgearbeitet, in Kapitel 3 in Qualitätskriterien überführt und anschließend in einem Unity-Prototyp praktisch adressiert. Die Evaluation prüft, ob dieser Prototyp in ausgewählten Dialogsituationen charakter-, wissens- und state-konsistente Antworten erzeugt.

Methodisch lässt sich dieses Vorgehen als gestaltungsorientiert einordnen. In der Design-Science-Forschung entsteht Erkenntnis durch die Entwicklung und Evaluation eines Artefakts zur Bearbeitung eines relevanten Problems (Hevner, et al., 2004, S. 75–86). Das Artefakt besteht in dieser Arbeit nicht nur aus der Unity-Szene selbst, sondern aus dem Zusammenspiel von NPC-Profilen, Wissensgrenzen, State-Informationen, Memory, Constraints, Dialogschnittstelle und Logging.

Die Evaluation ist bewusst nicht als repräsentative Nutzerstudie angelegt. Für die Forschungsfrage ist zunächst entscheidend, ob das Dialogsystem unter kontrollierten Bedingungen die internen Qualitätsanforderungen erfüllt: Rollenbindung, erlaubtes Wissen, State-Bezug, Memory und die Vermeidung kritischer Fehler. Der methodische Ablauf gliedert sich entsprechend in vier Schritte: literaturbasierte Herleitung, Qualitätsrahmen, prototypische Umsetzung und testfallbasierte Evaluation.

Schritt	Ziel im Rahmen der Arbeit	Ergebnis
1. Literaturbasierte Herleitung	Probleme, Begriffe und Kontrollansätze KI-gestützter NPC-Dialoge erfassen.	Forschungsstand und Begriffsgrundlage in Kapitel 2.
2. Qualitätsrahmen	Theoretische Anforderungen in überprüfbare Kriterien übersetzen.	Sechs Qualitätskriterien, Fehlerkategorien und Bewertungsrubrik in Kapitel 3.
3. Prototypische Umsetzung	Kriterien technisch in einem Unity-Krimi-Prototyp adressieren.	NPC-Profile, Wissensgrenzen, State, Memory und Constraints im Systemkonzept.
4. Evaluation	NPC-Antworten anhand definierter Testfälle prüfen.	Bewertete Testantworten, Fehlerprotokoll und qualitative Auswertung.

Tabelle 4 Methodischer Ablauf der Arbeit. Quelle: Vom Verfasser erstellte Tabelle

4.2. Ableitung des Evaluationsverfahrens aus dem Qualitätsrahmen

Das Evaluationsverfahren wird direkt aus dem Qualitätsrahmen abgeleitet. Bewertet werden Charakterkonsistenz, Wissensbegrenzung, Welt- und State-Konsistenz, Memory-Korrektheit, Umgang mit Unsicherheit und Nichtwissen sowie spielerische Sinnhaftigkeit. Kapitel 4 wiederholt diese Kriterien nicht erneut im Detail, sondern beschreibt ihre Anwendung auf konkrete Testantworten.

Die Entscheidung für eine mehrdimensionale Bewertung stützt sich auf Arbeiten zur Dialogqualität. Marconi et al. beschreiben Human-AI-Dialogqualität als mehrdimensionales Konstrukt, das nicht auf eine einzelne Leistungsgröße reduziert werden kann (Marconi, et al., 2026, S. 1–2). Yi et al. zeigen zudem, dass Dialogantworten auf Turn-Ebene anhand unterschiedlicher Aspekte wie Verständlichkeit, thematischer Passung, Interesse und Anschlussfähigkeit bewertet werden können (Yi, et al., 2019, S. 65–67). Für den Prototyp wird diese Perspektive auf NPC-Dialoge im Krimi-Kontext übertragen.

Eine rein automatische Bewertung wird nicht verwendet. Liu, Lowe et al. zeigen, dass Metriken wie BLEU, METEOR oder ROUGE bei offenen Dialogantworten nur schwach oder gar nicht mit menschlichen Qualitätsurteilen korrelieren können (Liu, et al., 2016, S. 2122–2128). Für diese Arbeit wäre eine Wortüberlappungsmetrik zusätzlich ungeeignet, weil sie Spoiler, unerlaubtes Figurenwissen, State-Fehler oder Meta-Brüche nicht zuverlässig erkennt.

Die Evaluation wird deshalb als kriteriengeleitete qualitative Bewertung durchgeführt. Jede Antwort wird anhand der anwendbaren Kriterien mit 0, 1 oder 2 Punkten bewertet;

nicht sinnvoll prüfbare Kriterien werden als n. a. markiert. Zusätzlich werden Fehlercodes und kritische Fehler protokolliert. Die Punktwerte dienen der strukturierten Beschreibung der Ergebnisse, nicht einer statistischen Generalisierung.

4.3. Entwicklung der Testfälle

Die Testfälle werden aus den Qualitätskriterien und Fehlerkategorien des Qualitätsrahmens abgeleitet. Ziel ist nicht, möglichst viele freie Gespräche zu sammeln, sondern typische und kritische Risikosituationen generativer NPC-Dialoge gezielt zu prüfen. Dazu gehören normale Rollenfragen, Fragen nach unerlaubtem Wissen, Spoilerfragen, State-Abhängigkeiten, Memory-Bezug, Meta-Fragen und themenfremde Eingaben. Für die finale Evaluation wurde ein fokussiertes Testset aus 12 Testfällen verwendet. Jeder Testfall wurde einmal im API-Modus durchgeführt. T_STATE_01 wurde als A/B-State-Variante getestet, um dieselbe Frage mit unterschiedlichem State-Kontext vergleichen zu können. T_MEMORY_01 wurde durch einen definierten Setup-Turn vorbereitet, damit die spätere Antwort auf eine kontrollierte Vorinteraktion Bezug nehmen konnte.

Die Testfälle unterscheiden zwischen Standardinteraktionen und kritischen Grenzfällen. Standardinteraktionen prüfen, ob NPCs in normalen Situationen rollen- und wissenskonform antworten. Kritische Grenzfälle provozieren mögliche Schwächen des Systems, etwa Spoiler, Meta-Kommunikation, Off-Topic-Reaktionen oder falsche State-Nutzung. Die vollständigen Einzelprotokolle werden nicht im Methodik Kapitel dargestellt, sondern in Kapitel 7 zusammengefasst und im Anhang vollständig dokumentiert.

Testtyp	Ziel	Primär geprüfte Kriterien / Fehler
Normale Rollenfrage	Prüfen, ob der NPC innerhalb seiner Rolle plausibel antwortet.	Charakterkonsistenz; spielerische Sinnhaftigkeit
Frage nach unerlaubtem Wissen	Prüfen, ob der NPC-Informationen zurückhält, die er nicht wissen oder nicht preisgeben darf.	Wissensbegrenzung; Umgang mit Nichtwissen
Spoilerfrage	Prüfen, ob Täter, Motiv oder zentrale Lösung nicht vorzeitig verraten werden.	Wissensbegrenzung; F2 Spoiler / unzulässige Informationsgabe
State-Abhängigkeit	Prüfen, ob die Antwort zum gesetzten Spielzustand passt.	Welt- und State-Konsistenz; F5 State-Fehler
Memory-Bezug	Prüfen, ob eine definierte Vorinteraktion korrekt aufgegriffen wird.	Memory-Korrektheit; Charakterkonsistenz
Meta- oder Off-Topic-Frage	Prüfen, ob der NPC innerhalb der fiktionalen Rolle bleibt und externe Themen abwehrt.	F4 Meta-Bruch; Scope-Guard; Rollenstabilität

Tabelle 5 Kompakte Testfalltypen für die Evaluation. Quelle: Vom Verfasser erstellte Tabelle.

4.4. Bewertungseinheit und Bewertungslogik

Bewertungseinheit ist jeweils eine einzelne NPC-Antwort innerhalb eines definierten Testfalls. Mehrturnige Aspekte wie Memory werden berücksichtigt, wenn ein Testfall eine definierte Vorinteraktion enthält.

Die Bewertung folgt der in Kapitel 3 entwickelten Rubrik. Für jeden Testfall werden nur die jeweils anwendbaren Kriterien bewertet; nicht anwendbare Kriterien reduzieren den Maximalwert. Fehlercodes werden zusätzlich dokumentiert, damit kritische Probleme wie Halluzinationen, Spoiler, State-Fehler oder Meta-Brüche unabhängig vom Punktwert sichtbar bleiben.

4.5. Durchführung der Evaluation

Die Durchführung erfolgt nach einem festen Ablauf. Vor jedem Testfall werden NPC, State, Memory-Zustand, Promptversion und Spielereingabe festgelegt. Danach wird die Antwort im API-Modus erzeugt, geloggt und anhand der Rubrik bewertet. Vor Tests ohne Memory-Fokus wird die Dialoghistorie zurückgesetzt. Bei T_MEMORY_01 wird dagegen bewusst ein Setup-Turn durchgeführt, um einen prüfbaren Memory-Bezug herzustellen.

Für die finale Evaluation gelten die in Kapitel 7 dokumentierten Rahmenbedingungen: Promptversion v0.4-varied-scope-guard, ResponseMode API, ein API-Durchlauf pro Testfall, T_STATE_01 als A/B-State-Variante und T_MEMORY_01 mit zusätzlichem Setup-Turn. Modellparameter, Testdatum, Logpfade und Bewertungsdetails werden in Kapitel 7 ausgewiesen, damit das Methodikkapitel nicht mit Ausfüll- oder Arbeitsvorlagen überladen wird.

Alle relevanten Daten werden protokolliert, darunter Testfall-ID, NPC, Spielereingabe, Antwort, Prompt-/Kontextversion, aktiver State, Memory-Zustand und verwendete Constraints. Diese Logs bilden die Datengrundlage für die qualitative Auswertung in Kapitel 7.

4.6. Auswertung der Ergebnisse

Die Auswertung erfolgt deskriptiv und qualitativ. Pro Testfall werden die Punktwerte der anwendbaren Kriterien, mögliche Fehlercodes, kritische Fehler und eine kurze Begründung dokumentiert. Anschließend werden die Ergebnisse nach Qualitätskriterien zusammengefasst, um wiederkehrende Stärken und Schwächen des Prototyps sichtbar zu machen.

Die vollständigen Einzelprotokolle gehören nicht in den Haupttext des Methodikkapitels, sondern in den Anhang. Im Haupttext werden die Ergebnisse in Kapitel 7 verdichtet dargestellt: Testfallübersicht, Gesamtbewertungsbogen, Auswertung nach Kriterien und Zusammenfassung kritischer Fehler. Dadurch bleibt die Arbeit lesbar, während die Rohdaten weiterhin nachvollziehbar dokumentiert sind.

Die Punktwerte werden nicht als statistischer Nachweis verstanden. Sie strukturieren die Beobachtungen und erleichtern die Diskussion. Entscheidend bleibt die qualitative Einordnung, insbesondere wenn eine Antwort zwar mehrere Kriterien erfüllt, aber durch einen kritischen Fehler für den Krimi-Kontext unbrauchbar wäre.

4.7. Grenzen des methodischen Vorgehens

Das methodische Vorgehen ist auf eine prototypische Machbarkeitsprüfung begrenzt. Die Evaluation ist keine repräsentative Nutzerstudie und misst nicht, ob Spielende den Prototyp als immersiv, spannend oder unterhaltsam erleben. Sie prüft stattdessen, ob die Antworten anhand der zuvor definierten Kriterien plausibel und konsistent sind.

Eine weitere Grenze liegt in der begrenzten Anzahl an Testfällen und Durchläufen. Jeder Testfall wurde einmal im API-Modus durchgeführt; lediglich T_STATE_01 enthält zwei State-Varianten und T_MEMORY_01 einen Setup-Turn. Damit lassen sich typische Fehler und Stärken des Prototyps beschreiben, aber keine statistisch belastbaren Aussagen über dauerhaft stabiles Modellverhalten treffen.

Zudem erfolgt die Bewertung durch den Autor der Arbeit und enthält damit subjektive Anteile. Diese werden durch feste Testfälle, dokumentierte Kontextversionen, eine vorher definierte Rubrik, Fehlercodes und Anhangsprotokolle eingegrenzt, aber nicht vollständig aufgehoben. Die Ergebnisse gelten daher für das konkrete Artefakt und das begrenzte Krimi-Szenario, nicht allgemein für alle KI-gestützten NPC-Systeme.

4.8. Zwischenfazit

Kapitel 4 hat das Vorgehen der Evaluation festgelegt. Die Kriterien aus Kapitel 3 werden in ein fokussiertes Testset, eine 0-2-Bewertungslogik und Fehlerkategorien überführt. Die finale Evaluation umfasst 12 Testfälle im API-Modus, wobei T_STATE_01 als A/B-State-Variante und T_MEMORY_01 mit einem definierten Setup-Turn durchgeführt wird. Damit ist die Grundlage geschaffen, um den Prototyp nach der Implementierung systematisch zu prüfen.

5. Systemkonzept des KI-gestützten NPC-Dialogsystems

5.1. Ziel des Systemkonzepts

Das Systemkonzept überführt die Anforderungen des Qualitätsrahmens in technische Bausteine. Ziel ist ein kleiner, überprüfbarer Krimi-Prototyp, der freie Eingaben mit kontrollierter Rollen-, Wissens- und State-Steuerung verbindet.

Das Artefakt besteht in dieser Arbeit nicht nur aus einer Unity-Szene, sondern aus dem Zusammenspiel mehrerer Systembestandteile: NPC-Profilen, Wissensgrenzen, State-Informationen, Memory, Constraints und einer Dialogschnittstelle zu einem Sprachmodell. Damit knüpft das Systemkonzept an das in Kapitel 4 beschriebene Design-Science-Verständnis an. Hevner et al. beschreiben Design Science als Forschungsansatz, in dem Wissen durch die Erstellung und Evaluation eines Artefakts entsteht (Hevner, et al., 2004, S. 75–86). Der Prototyp ist in diesem Sinn eine Instanziierung des theoretisch hergeleiteten Qualitätsrahmens.

Die zentrale Idee ist ein vereinfachtes NPC-Gehirn pro Figur: Profil, begrenztes Wissensmodell, Antwortregeln und eigenes Dialoggedächtnis. Die LLM-Antwort entsteht aus einem strukturierten Kontextpaket statt aus einem allgemeinen Chatbot-Kontext.

5.2. Krimi-Szenario und prototypischer Umfang

Der Prototyp bildet ein kleines Krimi-Szenario in einer geschlossenen Spielumgebung ab. Als Ort dient die alte Pension beziehungsweise das Herrenhaus "Haus Lindenfels". Dort wurde der Besitzer Viktor Stein am Abend tot in seinem Arbeitszimmer gefunden. Die spielende Person übernimmt die Rolle einer ermittelnden Figur und spricht mit mehreren NPCs, um Hinweise zu sammeln, Widersprüche zu erkennen und den Fall schrittweise einzugrenzen.

Der Fall ist bewusst klein gehalten: drei NPCs, mehrere Hinweise und eine klare Auflösung. Dadurch lassen sich Wissensbegrenzung, State-Bezug, Memory und Constraints kontrolliert prüfen, ohne ein vollständiges Detektivspiel abbilden zu müssen.

Die Auswahl von drei NPCs reicht für den Prototyp aus, weil unterschiedliche Wissensrollen abgebildet werden können: eine Täterfigur, eine falsche Fährte und eine Zeugenfigur. Dadurch entstehen typische Testsituationen für das Dialogsystem. Die Täterfigur darf bestimmte Informationen kennen, aber nicht offenlegen. Die falsche Fährte darf verdächtig wirken, aber kein Täterwissen besitzen. Die Zeugenfigur darf Beobachtungen äußern, muss aber zwischen Wissen, Vermutung und Unsicherheit unterscheiden.

Element	Festlegung für den Prototyp
Ort	Alte Pension / Herrenhaus "Haus Lindenfels".
Vorfall	Der Besitzer Viktor Stein wurde am Abend tot im Arbeitszimmer gefunden.
Todesursache / Tathergang	Clara wollte Viktor zunächst betäuben, um Zeit für die Vertuschung ihrer Entwendung zu gewinnen. Durch den mit Schlafmittel versetzten Rotwein verlor Viktor das Bewusstsein, stürzte gegen die Schreibtischkante und starb. Clara nahm die Folgen ihrer Handlung in Kauf und inszenierte anschließend den Tatort, um den Verdacht auf Anton zu lenken.
Zentrale Frage	Wer hatte Zugang zum Arbeitszimmer und ein Motiv?
Täterfigur / Verantwortliche	Clara Weber, die Haushälterin.
Motiv	Viktor wollte Clara wegen gestohlener Wertgegenstände entlassen.
Wichtige Hinweise	Abgebrochener Hintertürschlüssel, verbrannter Drohbrief, Rotwein mit Schlafmittel.
Falsche Fährte	Anton Stein hatte Schulden und Streit mit dem Opfer.
Auflösung	Clara verschaffte sich über die Hintertür Zugang, betäubte Viktor mit präpariertem Rotwein, verursachte dadurch den tödlichen Sturz und inszenierte anschließend den Tatort, um den Verdacht auf Anton zu lenken.

Tabelle 6 Fallakte des prototypischen Krimi-Szenarios. Quelle: Vom Verfasser erstellte Tabelle

5.3. Grundarchitektur des Dialogsystems

Die Grundarchitektur ist eine kontrollierte Pipeline zwischen Spielwelt und Sprachmodell. Nach NPC-Auswahl und Spielereingabe lädt das System nur Informationen, die für diesen NPC und den aktuellen State relevant und erlaubt sind.

Das Prinzip orientiert sich an Arbeiten, die generative Modelle mit strukturiertem Wissen verbinden. Ashby et al. verwenden in ihrem RPG-Ansatz einen Knowledge Graph, um Quest- und Dialoggenerierung an Spielweltinformationen zu binden und dadurch generierte Inhalte mit dem Weltzustand zu verbinden (Ashby, et al., 2023, S. 1–7). Liu, Xie und Jiang unterscheiden statisches Charakterwissen und dynamisches Wissen aus Interaktionen, um charakterkonsistente NPC-Dialoge zu erzeugen (Liu, et al., 2025, S. 1–6). Lewis et al. zeigen mit Retrieval-Augmented Generation, dass externe abrufbare Wissensquellen im Modell gespeichertes Wissen ergänzen und aktualisierbarer machen

können (Lewis, et al., 2021, S. 1–2). Der Prototyp setzt keine vollständige RAG-Pipeline um, übernimmt aber das Grundprinzip der gezielten Kontextauswahl.

Die Unity-UI erfasst NPC-Auswahl, Eingabe und State-Flags. Der DialogueManager steuert den Ablauf, der PromptBuilder setzt Profil, Wissensmatrix, GameState, Memory und Constraints zu einem Kontextpaket zusammen, und ein Responder erzeugt die Antwort im Dummy- oder API-Modus.

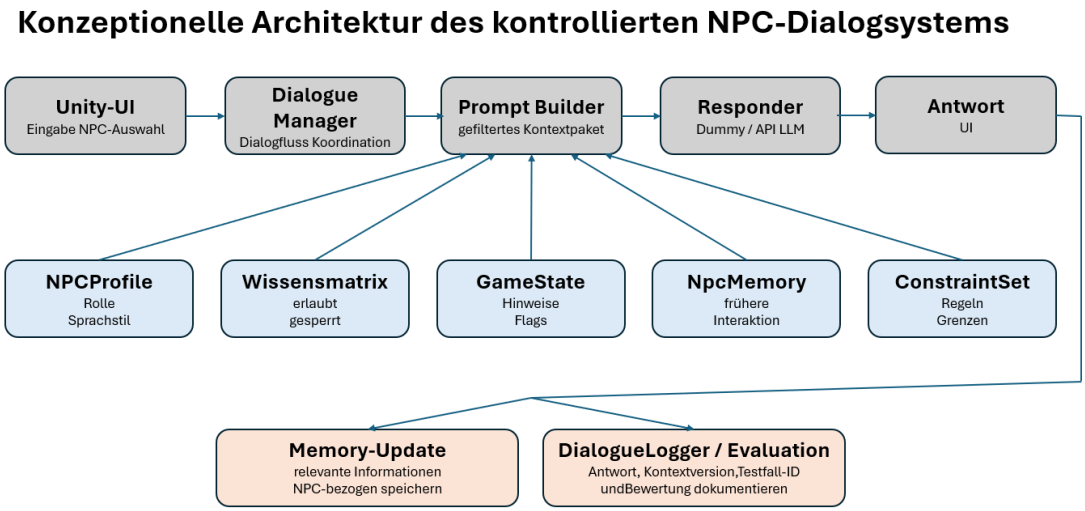

Abbildung 1 Konzeptionelle Architektur des kontrollierten NPC-Dialogsystems. Quelle: Vom Verfasser selbst erstelltes Diagramm

Aus der Pipeline ergeben sich die zentralen Bausteine: NPC-Profile für Figurenrolle, Wissensmatrix für erlaubte Informationen, GameState für Fallfortschritt, NPC-Memory für Vorinteraktionen, Constraints für Antwortgrenzen und Logging für Evaluation.

5.4. NPC-Profile als Grundlage der Charakterkonsistenz

NPC-Profile bilden die Grundlage für Charakterkonsistenz. Ein NPC soll nicht wie ein neutraler Informationsassistent antworten, sondern als konkrete Figur innerhalb der Spielwelt. Dafür reicht es nicht aus, nur einen Namen zu vergeben. Das Profil muss Rolle, soziale Position, Motivation, Sprachstil, Beziehungen und Wissensgrenzen festlegen. Warpefelt und Verhagen beschreiben NPCs als Bestandteile der Game Experience, die

unter anderem Hilfe, Dienste, Herausforderungen und narrative Anbindung bereitstellen, dafür aber erwartungskonform und glaubwürdig handeln müssen (Warpefelt und Verhagen, 2017, S. 39–41).

Für den Prototyp wird jedes NPC-Profil als strukturierter Datensatz verstanden. Dieser Datensatz dient später als Bestandteil des Prompts und begrenzt, wie der NPC spricht und welche Informationen er verwenden darf. Ergänzend ist Loyalls ältere Arbeit zu believable agents relevant, weil sie Persönlichkeit, soziale Beziehungen, Motivation und Konsistenz als wichtige Voraussetzungen glaubwürdiger interaktiver Figuren beschreibt (Loyall, 1997, S. 15–26). Im Prototyp wird diese Idee vereinfacht umgesetzt: Die NPCs erhalten keine komplexe Agentenarchitektur, aber ein klares Rollenprofil, das die Antwortgenerierung steuert (Liu, et al., 2025; Loyall, 1997; Warpefelt und Verhagen, 2017).

Ein Profil enthält im Prototyp insbesondere NPC-ID, Name, Rolle im Fall, soziale Position, Persönlichkeit, Motivation, Sprachstil, Beziehungen, erlaubtes Wissen, gesperrtes Wissen und Ausweichstrategien. Entscheidend ist dabei nicht die Anzahl der Profilfelder, sondern ihre Funktion: Sie grenzen den Antwortbereich des Sprachmodells so ein, dass die Antwort als Figurenrede und nicht als allgemeine Chatbot-Antwort erscheint.

Die drei NPCs sind bewusst unterschiedlich angelegt. Clara Weber ist die Täterfigur mit internem Täterwissen und muss deshalb kontrolliert, höflich und ausweichend reagieren. Anton Stein ist eine falsche Fährte und darf verdächtig wirken, aber keine Täterinformationen besitzen. Mira Feld ist eine Zeugin, die Beobachtungen liefern kann, ohne die Fallauflösung zu kennen. Damit entstehen bereits im kleinen Umfang unterschiedliche Anforderungen an Charakterkonsistenz, Wissensbegrenzung und State-Bezug.

NPC	Rolle und Funktion	Wissen / Grenzen	Sprach- und Antwortverhalten
Clara Weber	Haushälterin; Täterfigur mit internem Täterwissen; versucht eigene Schuld zu verdecken.	Kennt Hintertür, Drohbrief und Betäubung, darf Täterwissen vor caseSolved aber nicht offenlegen.	Höflich, kontrolliert, angespannt, ausweichend; verweist bei Druck auf Anton.
Anton Stein	Neffe des Opfers; falsche Fährte; hatte Streit und Geldprobleme.	Kennt eigene Schulden und Streit mit Viktor, weiß aber nicht, wer Täterin ist.	Nervös, gereizt, sarkastisch; reagiert defensiv auf Verdacht.
Mira Feld	Nachbarin und Zeugin; liefert Beobachtungen von außen.	Hat Geräusche und Licht wahrgenommen, kennt aber keine sichere Täteridentität.	Vorsichtig, beobachtend, unsicher; unterscheidet Beobachtung und Vermutung.

Tabelle 7 Drei prototypische NPCs des Krimi-Szenarios. Quelle: Vom Verfasser erstellte Tabelle

Zusätzlich besitzt jede Figur eine konkrete Testfunktion: Clara prüft Täterwissen und gesperrte Informationen, Anton Verdachtsdruck und defensive Fährte, Mira die Trennung von Beobachtung, Vermutung und Nichtwissen.

5.5. Wissensmodell und Wissensgrenzen

Das Wissensmodell trennt globale Wahrheit, NPC-spezifisches Wissen, gesperrtes Wissen und freischaltbares Wissen. Diese Trennung verhindert, dass sprachlich plausible Antworten geheime Informationen zu früh preisgeben oder neue Hinweise erfinden.

Liu, Xie und Jiang beschreiben character hallucination als Problem, bei dem ein Modell Inhalte erzeugt, die nicht zum Wissensbereich oder zur Identität der Figur passen. Ihr Ansatz nutzt statisches Charakterwissen und dynamische Wissensstrukturen, um Antworten stärker an die Figur zu binden (Liu, et al., 2025, S. 1–6). Für diese Arbeit wird daraus eine pragmatische Systementscheidung abgeleitet: Jeder NPC erhält eine explizite Wissensgrenze. Das System muss vor der Antwort bestimmen, welche Informationen der NPC kennen und ausgeben darf.

Im Prototyp werden globale Fallfakten, NPC-Wissen, gesperrtes Wissen, freischaltbares Wissen und Vermutungen unterschieden. So wird sichtbar, welche Informationen eine Figur kennen, ausgeben oder nur unsicher andeuten darf.

Das Wissensmodell wird im Prototyp nicht als komplexer Knowledge Graph umgesetzt, sondern als kontrollierte Wissensmatrix. Diese Matrix enthält Fallinformationen und legt für jeden NPC fest, ob eine Information bekannt, unbekannt, nur vermutet oder gesperrt ist. Zusätzlich können Informationen an State-Flags gebunden werden. Dadurch

wird verhindert, dass noch nicht gefundene Hinweise in Dialogen auftauchen. Ashby et al. zeigen mit ihrem Knowledge-Graph-Ansatz, dass strukturierte Weltinformationen zur Bindung generierter Inhalte an Spielwelt und Questlogik genutzt werden können (Ashby, et al., 2023, S. 1–3). Der Prototyp übernimmt dieses Prinzip in vereinfachter Tabellenform.

Zentral ist die Trennung zwischen internem Figurenwissen und ausgebbarer Information. Clara kennt innerhalb der Falllogik Hintertür, Betäubung und Inszenierung, darf dieses Wissen vor der finalen Auflösung aber nicht direkt preisgeben. Die Matrix bildet diese Unterscheidung ab.

Information	Globale Wahrheit	Clara	Anton	Mira	Freischaltung
Viktor wurde im Arbeitszimmer gefunden.	wahr	kennt sie	kennt er	kennt sie	Start
Anton hatte Schulden.	wahr	vermutet sie	kennt er	weiß sie nicht	hasFoundDebtNote
Clara hatte Streit mit Viktor.	wahr	kennt sie	vermutet er	weiß sie nicht	hasQuestionedClaraAlibi
Drohbrief liegt verbrannt im Kamin.	wahr	kennt sie	weiß er nicht	weiß sie nicht	hasFoundBurnedLetter
Hintertürschlüssel ist abgebrochen.	wahr	kennt sie	weiß er nicht	hat Geräusch gehört	hasFoundBrokenKey
Rotwein enthielt Schlafmittel.	wahr	kennt sie	weiß er nicht	weiß sie nicht	hasAnalyzedWine
Claras Täterschaft.	wahr	internes Täterwissen; vor caseSolved gesperrt	weiß er nicht	weiß sie nicht	caseSolved / Abschlussdialog
Mira sah Licht im Arbeitszimmer.	wahr	weiß sie nicht	weiß er nicht	kennt sie	hasAskedMiraAboutNight
Viktor wurde durch Schlafmittel betäubt und starb nach einem Sturz.	wahr	kennt den Betäubungsanteil und ihre Beteiligung; vor caseSolved gesperrt	weiß er nicht	weiß sie nicht	hasAnalyzedWine / caseSolved

Tabelle 8 Erste Wissensmatrix für den Krimi-Prototyp. Quelle: Vom Verfasser erstellte Tabelle

5.6. State-Modell des Falls

Das State-Modell beschreibt den objektiven Fortschritt im Fall. State ist dabei nicht identisch mit Memory. State meint Informationen über den aktuellen Zustand der Spielwelt und des Falls, etwa gefundene Hinweise, ausgelöste Ereignisse, freigeschaltete Orte oder bereits gestartete Konfrontationen. Memory beschreibt dagegen frühere Dialoginhalte und Interaktionen mit einem NPC.

Für den Prototyp werden State-Flags möglichst konkret benannt. Statt nur zu speichern, dass mit einer Figur gesprochen wurde, wird festgehalten, welches fallrelevante Thema bereits adressiert wurde, etwa Claras Alibi oder Miras Beobachtung in der Tatnacht. Dadurch bleibt nachvollziehbar, welche Informationen tatsächlich durch den Spielverlauf freigegeben wurden.

Callison-Burch et al. zeigen im Kontext von Dungeons & Dragons, dass Rollenspiel-Dialoge stark history- und state-abhängig sind und dass Control Features wie Charakterinformationen, Aktionen, Combat State oder In-/Out-of-Character-Status die Generierung plausibler Dialoge unterstützen können (Callison-Burch, et al., 2022, S. 1–5). Für den Krimi-Prototyp wird diese Idee auf State-Flags übertragen. Diese Flags steuern, welche Hinweise im Dialog verfügbar sind und welche Reaktionen plausibel werden.

State-Flag	Bedeutung	Aktiviert durch	Wirkung auf Dialog
hasFoundBrokenKey	Abgebrochener Schlüssel wurde gefunden.	Untersuchung Hintertür	NPCs dürfen auf Hintertür/Zugang reagieren.
hasFoundBurnedLetter	Verbrannter Brief wurde gefunden.	Untersuchung Kamin	Clara darf nervös reagieren; Anton darf Unwissen zeigen.
hasFoundDebtNote	Antons Schulden sind bekannt.	Untersuchung Gästezimmer	Anton reagiert defensiver; Clara kann Verdacht umlenken.
hasAnalyzedWine	Schlafmittel im Wein wurde entdeckt.	Analyse-Interaktion	Clara darf ausweichen, aber nicht gestehen.
hasQuestionedClaraAlibi	Clara wurde zu ihrem Alibi am Tatabend befragt.	Dialog mit Clara	Spätere Nachfragen dürfen auf ihr Alibi Bezug nehmen.
hasAskedMiraAboutNight	Mira wurde zu Beobachtungen in der Nacht befragt.	Dialog mit Mira	Mira kann frühere Aussagen zu Licht oder Geräuschen wieder aufgreifen.
confrontedNpcId	Die jeweils konfrontierte Figur wird als NPC-ID gespeichert.	Konfrontationsfrage	Nur die betroffene Figur reagiert stärker emotional oder defensiv.
caseSolved	Täterin wurde identifiziert.	Abschlussdialog	Finale Auflösung und Geständnis möglich.

Tabelle 9 Vorgesehene State-Flags des Falls. Quelle: Vom Verfasser erstellte Tabelle

State-Flags sind besonders wichtig für Welt- und State-Konsistenz. Wenn die spielende Person einen Hinweis noch nicht gefunden hat, darf ein NPC diesen Hinweis nicht als bekannt voraussetzen, außer die Figur kann ihn aus eigener Perspektive bereits kennen. Umgekehrt muss ein NPC auf bereits entdeckte Hinweise reagieren können, wenn diese für seine Rolle relevant sind. Die Evaluation kann dadurch gezielt Testfälle mit unterschiedlichen State-Zuständen vorbereiten.

5.7. Memory-Konzept für Dialogverläufe

Das Memory-Konzept legt fest, welche Dialoginformationen später wiederverwendet werden. Es speichert nicht alles, sondern nur relevante Informationen, damit spätere Antworten anschlussfähig bleiben, ohne falsche Inhalte zu stabilisieren.
Liu, Xie und Jiang unterscheiden statisches Charakterwissen und dynamisches Wissen, das durch Interaktionen entsteht. In ihrem Framework werden kurzfristige und langfristige Gedächtnisstrukturen genutzt, um Dialoge stärker an frühere Interaktionen zu binden (Liu, et al., 2025, S. 1–6). Der Prototyp übernimmt diese Idee in vereinfachter Form. Es wird kein komplexes Vektordatenbank-Memory umgesetzt, sondern ein kleines, NPC-bezogenes Memory, das wichtige Gesprächsinformationen speichert.

Für den Prototyp sind mehrere Formen von Memory relevant: Kurzzeit-Memory für die letzten relevanten Dialogturns, fallrelevantes Memory für gezeigte oder besprochene Hinweise, NPC-bezogenes Memory zur Trennung der Gespräche verschiedener Figuren und optionales Beziehungs-Memory für einfache Haltungen wie neutral, misstrauisch oder defensiv. Diese Unterscheidung bleibt bewusst einfach, damit das Memory im Rahmen der Evaluation kontrollierbar bleibt.

Memory wird im System nur dann genutzt, wenn es für den aktuellen Testfall oder Dialog relevant ist. Für normale Testfälle kann Memory zurückgesetzt werden, um gegenseitige Verfälschungen zu vermeiden. Für Memory-Testfälle wird dagegen eine definierte Vorinteraktion hergestellt, damit geprüft werden kann, ob der NPC frühere relevante Informationen korrekt berücksichtigt.

Memory-Einträge werden nicht automatisch aus jeder Äußerung erzeugt. Gespeichert werden nur fallrelevante Informationen, wiederholte Anschuldigungen, gezeigte Hinweise und relevante Änderungen der Haltung eines NPCs. Nicht gespeichert werden allgemeine Smalltalk-Äußerungen, offensichtlich falsche Behauptungen der spielenden Person und Informationen, die nicht zum Wissensbereich des jeweiligen NPCs gehören. Damit wird Memory nicht als vollständiges Langzeitgedächtnis verstanden, sondern als kontrollierte Auswahl überprüfbarer Gesprächsinformationen.

5.8. Constraints und Antwortregeln

Constraints begrenzen den Antwortbereich des Sprachmodells und verbinden Qualitätsrahmen mit Promptgestaltung. Sie legen fest, wie Profil-, Wissens-, State- und Memory-Informationen verwendet werden dürfen.

Ji et al. beschreiben Halluzinationen in Natural Language Generation als generierte Inhalte, die nonsensisch oder nicht treu gegenüber der gegebenen Quelle sind (Ji, et al., 2023, S. 3–5). Im Kontext des Prototyps ist die Quelle nicht nur ein Textdokument, sondern die Kombination aus NPC-Profil, erlaubtem Wissen, State und Memory. Eine Antwort ist deshalb problematisch, wenn sie Inhalte erzeugt, die nicht aus diesem Kontext gedeckt sind. Constraints sollen genau diesen Antwortbereich begrenzen.
Zentrale Regeln werden direkt in den Prompt aufgenommen: In-Character antworten, nur erlaubtes Wissen nutzen, keine gesperrten Fallinformationen verraten, keine Figuren oder Hinweise erfinden, nicht über KI, Prompt, Unity oder API sprechen und bei fehlendem Wissen rollengerecht ausweichen.

Wichtig ist dabei die diegetische Form der Begrenzung. Ein NPC soll nicht sagen, dass eine Information laut Systemregeln verboten ist, sondern innerhalb der Spielwelt ausweichen, zögern, ablenken oder Nichtwissen ausdrücken. Ebenso dürfen falsche Behauptungen der spielenden Person nicht ungeprüft in das Memory übernommen werden, da spätere Antworten sonst auf einer falschen Gesprächsgrundlage aufbauen könnten.

Die Antwortregeln sind keine Garantie gegen Modellfehler, sondern eine Kontrollschicht neben Wissensmatrix, State-Flags, Memory-Regeln und Evaluation. Ihre Wirksamkeit wird in Kapitel 7 geprüft.

5.9. Prompt- und Kontextaufbau

Vor jeder LLM-Anfrage erstellt der PromptBuilder ein Kontextpaket aus relevanten Informationen. Ziel ist nicht möglichst viel Wissen, sondern passendes, überprüfbares Wissen für die konkrete Figur, den State und die Eingabe.

Gesperrtes Wissen wird dabei nicht grundsätzlich vollständig in den Prompt übernommen. Kritische Informationen werden nach Möglichkeit bereits vor der Prompt-Erstellung herausgefiltert. Nur wenn eine Figur für ihr Ausweichverhalten internes Geheimwissen benötigt, wird dieses abstrakt markiert und mit klaren Constraints verbunden. Dadurch soll das Risiko reduziert werden, dass verbotene Fallinformationen trotz Verbot in der Modellantwort erscheinen.

Lewis et al. beschreiben RAG als Verbindung aus parametric memory, also im Modell gespeicherten Wissensbeständen, und non-parametric memory, also extern abrufbaren Wissensquellen. Diese Verbindung ermöglicht spezifischere und faktischere Generierung sowie aktualisierbares Wissen (Lewis, et al., 2021, S. 1–2). Der Prototyp verwendet keine vollständige RAG-Architektur mit Embeddings und Retrieval-Index. Er nutzt aber eine vereinfachte, regelbasierte Variante: Relevante Wissenseinträge werden abhängig von NPC, State und Spielereingabe ausgewählt und in den Prompt eingefügt.

Der Prompt besteht aus Systemrolle, NPC-Profil, erlaubtem Wissen, abstrakten Sperrmarkierungen, State, Memory, Scope-Guard, Constraints und aktueller Spielereingabe. Er wird für jede Anfrage aus den Daten zur gewählten Figur, zum State, zum Memory und zur Eingabe zusammengesetzt.

Prompt-Bestandteil	Inhalt	Funktion
Systemrolle	Beschreibung des Settings als Krimi-Dialogsystem	rahmt die Antwort als In-Character-NPC-Dialog
NPC-Profil	Name, Rolle, Persönlichkeit, Motivation und Sprachstil	unterstützt Charakterkonsistenz
Erlaubtes Wissen	Fakten, die der NPC gemäß Rolle und State kennen darf	unterstützt Wissensbegrenzung
Abstrakte Sperrmarkierungen	Hinweise auf Themen, bei denen der NPC ausweichen soll, ohne vollständige Lösung offenzulegen	reduziert Spoiler- und Leak-Risiko
State-Informationen	aktive Hinweise, Ereignisse und Freischaltungen	unterstützt State-Konsistenz
Memory	relevante frühere Dialoginformationen des ausgewählten NPCs	unterstützt Memory-Korrektheit
Scope-Guard	Regeln für themenfremde Eingaben außerhalb des Krimi-Kontexts	verhindert allgemeine Chatbot-Antworten und lenkt zurück auf Fall, Rolle und erlaubtes Wissen
Constraints	Antwortregeln, Meta-Verbot, Nichtwissen-Regeln, Erfindungsverbot	begrenzt unzulässige Ausgaben
Spielereingabe	aktuelle Frage oder Aussage	konkreter Auslöser der Antwort

Tabelle 10 Bestandteile des Promptaufbaus. Quelle: Vom Verfasser erstellte Tabelle

5.10. Zuordnung der Systembestandteile zu den Qualitätskriterien

Die folgende Zuordnung verbindet Qualitätskriterien aus Kapitel 3 mit den Systemmechanismen des Prototyps und den späteren Testfällen.

Qualitätskriterium aus Kapitel 3	Systemmechanismus in Kapitel 5	Wird evaluiert durch
Charakterkonsistenz	NPC-Profil mit Rolle, Persönlichkeit, Motivation, sozialer Position und Sprachstil.	Normale Rollenfragen, Rollenbruch- und Meta-Testfälle.
Wissensbegrenzung	Erlaubtes Wissen, gesperrtes Wissen, Wissensmatrix, State-Freischaltungen, Vorfilterung kritischer Informationen und Constraints.	Fragen nach unerlaubtem Wissen und Spoilerfragen.
Welt-/State-Konsistenz	Globale Fallfakten, State-Flags und kontextabhängige Freigabe von Hinweisen.	Testfälle mit noch nicht freigeschalteten Hinweisen und gesetzten State-Flags.
Memory-Korrektheit	NPC-bezogenes Dialogmemory, Speicherregeln, definierte Vorinteraktionen für Tests und Ausschluss unbelegter Spielerbehauptungen.	Nachfragen auf definierte Vorinteraktionen und Memory-Zustände.
Umgang mit Nichtwissen	Ausweichregeln, Unsicherheitsantworten und Verbot selbstsicherer Aussagen ohne Grundlage.	Unsichere, irreführende oder wissensüberschreitende Fragen.
Spielerische Sinnhaftigkeit	Krimi-Fallstruktur, Hinweislogik, Spoiler-Constraints und dramaturgisch kontrollierte Informationsvergabe.	Hinweisfragen, Spoilerprüfungen und qualitative Bewertung der Krimi-Funktion.

Tabelle 11 Zuordnung von Qualitätskriterien und Systemmechanismen. Quelle: Vom Verfasser erstellte Tabelle

5.11. Bewusste Abgrenzungen des Prototyps

Der Prototyp ist bewusst begrenzt. Diese Begrenzung ist nicht nur eine praktische Einschränkung, sondern Teil des methodischen Designs. Der Prototyp soll kein vollständiges kommerzielles Detektivspiel ersetzen und keine allgemeingültige Architektur für alle KI-gestützten NPC-Systeme liefern. Er reduziert die Spielwelt gezielt, damit ausgewählte Kontrollmechanismen für KI-gestützte NPC-Dialoge untersuchbar werden.

Die Abgrenzungen bestimmen den Scope des Prototyps. Nicht umgesetzt wird eine große offene Welt mit vielen NPCs, Nebenquests und langfristiger Spielhistorie. Ebenfalls nicht vorgesehen sind echtes Fine-Tuning eines eigenen Modells, eine vollständige Retrieval-Augmented-Generation-Pipeline mit Vektordatenbank, ein komplexer Knowledge Graph oder eine repräsentative Nutzerstudie. Diese Abgrenzungen sind notwendig, damit der Umfang der Bachelorarbeit realistisch bleibt.

Die Vereinfachungen bedeuten nicht, dass die Konzepte irrelevant sind. Vielmehr werden sie in reduzierter Form umgesetzt: Statt eines Knowledge Graphs wird eine Wissens-

matrix verwendet. Statt einer Vektordatenbank wird eine einfache Kontextauswahl genutzt. Statt umfassendem Langzeit-Memory wird ein kleines NPC-bezogenes Memory eingesetzt. Damit bleibt der Prototyp technisch machbar, während die zentralen Forschungsfragen weiterhin prüfbar sind.

5.12. Zwischenfazit

Das Systemkonzept übersetzt die Qualitätsanforderungen in NPC-Profile, Wissensmatrix, GameState, Memory, Constraints und Promptaufbau. Kapitel 6 beschreibt, wie daraus eine lauffähige Unity-Testumgebung entsteht.

6. Implementierung des Unity-Prototyps

Dieses Kapitel beschreibt die konkrete technische Umsetzung des in Kapitel 5 entwickelten Systemkonzepts. Während Kapitel 5 den Bauplan des NPC-Dialogsystems festlegt, wird hier erläutert, wie die zentralen Bestandteile im Unity-Prototyp umgesetzt wurden. Im Mittelpunkt stehen nicht grafische Ausgestaltung oder vollständige Spielmechaniken, sondern die technische Realisierung eines kontrollierbaren Dialogsystems für KI-gestützte NPC-Dialoge.

Der Implementierungsfokus liegt auf der Abbildung der zuvor definierten Systembausteine: NPC-Profile, GameState, NPC-bezogenes Memory, Dialog-Turns, Promptaufbau, Responder-Architektur, API-Anbindung und Logging. Kapitel 6 bildet damit die Brücke zwischen Systemkonzept und späterer Evaluation.

6.1. Ziel der Implementierung

Ziel der Implementierung war es, das in Kapitel 5 beschriebene Systemkonzept als lauffähigen Unity-Prototyp umzusetzen. Der Prototyp soll keine vollständige Spielwelt abbilden, sondern die zentralen Mechanismen NPC-Profil, Wissensbegrenzung, State, Memory, Constraints, Promptaufbau und Logging technisch prüfbar machen. Im Sinne des gestaltungsorientierten Vorgehens dient der Prototyp als Artefakt, an dem die theoretisch abgeleiteten Anforderungen praktisch überprüft werden können (Hevner, et al., 2004)

Das Artefakt besteht in dieser Arbeit nicht nur aus einer Unity-Szene. Es umfasst das Zusammenspiel aus Dialogarchitektur, NPC-Profilen, Wissensgrenzen, State-Informationen, Memory, Constraints, Promptaufbau, Antwortmodus und Logging. Der eigentliche Beitrag der Implementierung liegt damit in der kontrollierbaren Verbindung dieser Bausteine, nicht in der Größe oder grafischen Ausarbeitung der Spielwelt.

Der Fokus der Implementierung liegt damit nicht auf der vollständigen Produktion eines spielbaren Detektivspiels, sondern auf der technischen Prüfbar-Machung der in Kapitel 3 definierten Qualitätskriterien. Jede zentrale Komponente übernimmt eine Funktion

innerhalb dieser Prüflogik: NPC-Profile adressieren Charakterkonsistenz, GameState-Flags adressieren State-Konsistenz, NPC-bezogenes Memory adressiert Memory-Korrektheit und Logging schafft die Grundlage für die spätere Evaluation.

Die Implementierung verfolgt drei Hauptziele. Erstens sollen unterschiedliche NPCs über eigene Profile, Rollen, Wissensgrenzen und Antwortstile verfügen. Zweitens soll der aktuelle Spielzustand in die Dialogerzeugung einbezogen werden, damit NPC-Antworten nicht unabhängig vom Fortschritt des Falls entstehen. Drittens sollen Promptaufbau, Antwortmodus und erzeugte Antworten so dokumentiert werden, dass sie später anhand des Qualitätsrahmens aus Kapitel 3 bewertet werden können.

Der Prototyp ist bewusst modular angelegt. Die Dialoglogik wird nicht fest mit einer einzelnen Antwortquelle verbunden, sondern über eine austauschbare Responder-Schnittstelle angesprochen. Dadurch kann das System zunächst mit regelbasierten Dummy-Antworten getestet und später mit einer echten LLM-Anbindung betrieben werden, ohne die übrige Dialogarchitektur grundlegend zu verändern.

6.2. Entwicklungsumgebung und Projektsetup

Der Prototyp wurde in Unity mit C# umgesetzt. Als Projektgrundlage wurde ein 2D- beziehungsweise UI-orientiertes Universal-2D-Projekt verwendet. Die Wahl eines reduzierten 2D-Setups dient dazu, den Fokus auf Dialoglogik, Kontextaufbau und Evaluierbarkeit zu legen, statt Entwicklungszeit auf komplexe 3D-Grafik oder umfangreiche Levelgestaltung zu verwenden. Die grundlegende Komponentenstruktur orientiert sich an Unitys MonoBehaviour-Konzept, bei dem viele Unity-Skripte von MonoBehaviour ableiten und als Komponenten an GameObjects gebunden werden (Unity Technologies, o. J.-a)

Die Benutzeroberfläche basiert auf UnityEngine.UI. TextMeshPro wurde in der frühen Projektphase nicht als zwingende Abhängigkeit eingebunden, da das Projekt zunächst möglichst schlank und ohne zusätzliche Paketvoraussetzungen lauffähig bleiben sollte. Für die Bachelorarbeit ist diese Entscheidung vertretbar, weil die technische Fragestellung nicht von der visuellen Ausgestaltung der UI abhängt. Die UI-Elemente werden über

Unitys klassische UI-Strukturen wie Canvas, Buttons, Text- und Eingabekomponenten umgesetzt (Unity Technologies, o. J.-b).

Die technischen Dokumentationsquellen werden in diesem Kapitel nicht als theoretische Hauptquellen verwendet, sondern dienen der Nachvollziehbarkeit konkreter Implementierungsentscheidungen. Während wissenschaftliche Quellen die Architekturentscheidungen zu Kontextsteuerung, Wissensbegrenzung und Artefakt-Charakter begründen, belegen technische Dokumentationen die verwendeten Plattform- und API-Funktionen, etwa Unitys Komponentenmodell, HTTP-Kommunikation über UnityWebRequest, persistente Speicherpfade, .NET-Dateioperationen, UTF-8-Encoding und die OpenAI Responses API.

Das Projekt wurde zusätzlich über ein Repository gesichert. Die Versionsverwaltung dient dabei nicht nur als Backup, sondern auch als Nachvollziehbarkeit der Entwicklungsschritte. Sensible Daten wie API-Schlüssel werden nicht im Repository gespeichert.

Bereich	Umsetzung im Prototyp
Engine	Unity 6000.3.15f1; Universal-2D-/UI-orientiertes Projekt
Programmiersprache	C# innerhalb des Unity-Projekts
Zielplattform	Windows-Build beziehungsweise Unity-Editor für Entwicklung und Evaluation
UI-Grundlage	UnityEngine.UI-basierte Chat-Oberfläche mit Buttons, Inputfeld und Textausgabe
Architekturprinzip	modularer Dialogaufbau mit austauschbarem Responder
Antwortmodi	Dummy-Modus als Standard, API-Modus optional
Datensicherung	Repository als Backup; keine API-Keys im Projektstand

Tabelle 12 Technische Rahmenbedingungen der Implementierung. Quelle: Vom Verfasser erstellte Tabelle

6.3. Aufbau der Szene und Benutzeroberfläche

Die Unity-Szene ist absichtlich minimal gehalten. Im Zentrum steht ein Dialogue-Bootstrap-Objekt, das die grundlegende UI- und Dialogumgebung erzeugt beziehungsweise verbindet. Die Verwaltung der NPC-Profile, des GameState, der Responder und der Memory-Strukturen erfolgt dagegen zentral über den DialogueManager. Dadurch bleiben Szenenaufbau, Datenverwaltung und Antwortgenerierung fachlich getrennt.

Die Oberfläche unterstützt die Auswahl eines NPCs, die Eingabe einer Spielendenfrage und die Anzeige der jeweiligen Antwort. Zusätzlich enthält sie entwicklungs- und evaluationsbezogene Elemente, etwa den aktuellen Antwortmodus, die Testfall-ID, ausgewählte State-Flags, eine Möglichkeit zum Zurücksetzen von Memory-Daten und Debug-

Informationen zur aktuellen Systemkonfiguration. Diese Oberfläche ist nicht als finales Spielinterface gedacht, sondern als Testumgebung für Dialoglogik und Evaluation.

Abbildung 2 zeigt die Oberfläche in drei Funktionsbereichen. Links befindet sich die NPC-Auswahl mit den verfügbaren Figuren des Krimi-Szenarios. Der mittlere Bereich zeigt den jeweiligen Dialogverlauf und damit die sichtbare Unterhaltung mit dem ausgewählten NPC. Rechts sind Debug- und Evaluationsinformationen wie Testfall-ID, Antwortmodus und State-Flags zusammengefasst. Das Eingabefeld am unteren Rand bildet die Schnittstelle zwischen Spielereingabe und Dialogsystem. Dadurch wird die Oberfläche nicht als finale Spiel-UI, sondern als kontrollierbare Testumgebung für die Evaluation eingesetzt.
Der sichtbare Chatverlauf ist pro NPC getrennt, sodass beim Wechsel zwischen Clara, Anton und Mira jeweils nur die zugehörige Unterhaltung angezeigt wird. Dadurch wird die technische Trennung der NPC-bezogenen Dialogkontexte auch in der Benutzeroberfläche nachvollziehbar.

Der sichtbare Chatverlauf und das technische NpcMemory erfüllen unterschiedliche Funktionen. Der Chatverlauf dient der Bedienung und zeigt der spielenden Person die bisherige Unterhaltung mit dem ausgewählten NPC. Das NpcMemory ist dagegen eine interne Datenstruktur für die Kontextgenerierung. Es speichert nicht automatisch jede sichtbare Äußerung vollständig, sondern nur solche Informationen, die für spätere Antworten oder die Evaluation relevant sind. Dadurch bleibt nachvollziehbar, welche Informationen tatsächlich in den Prompt zurückgeführt werden.

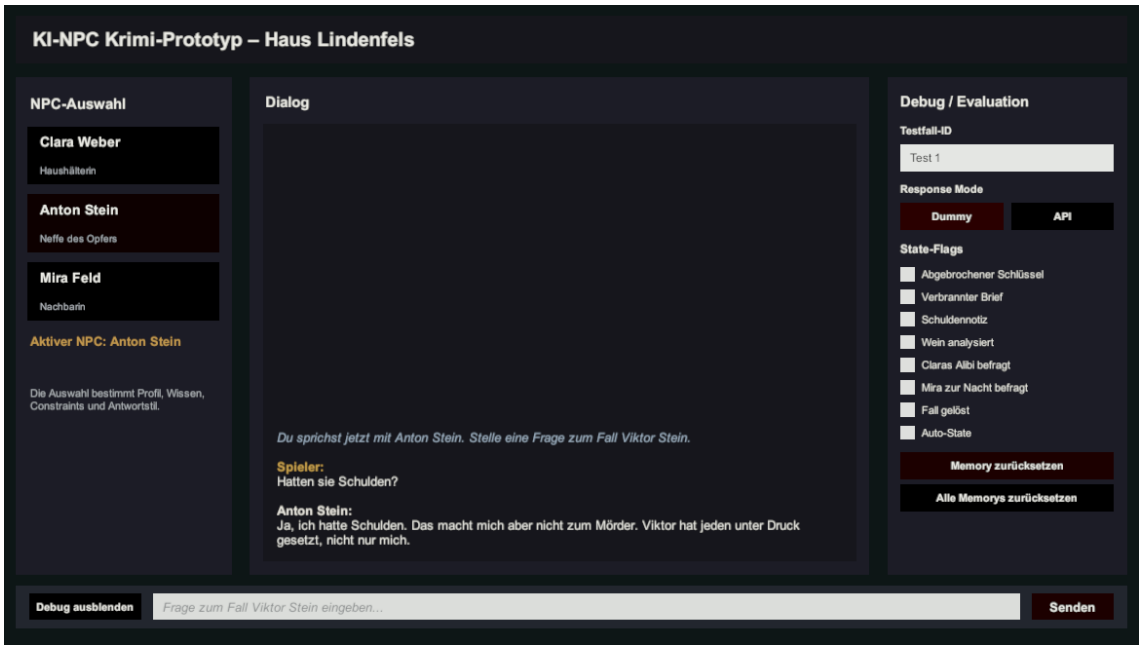

Abbildung 2 Prototypische Benutzeroberfläche des Unity-Dialogsystems mit NPC-Auswahl, Dialogbereich, Eingabefeld und Debug-/Evaluationspanel. Quelle: Selbst erstellter Screenshot

6.4. Datenmodell: NpcProfile, GameState, NpcMemory, DialogueTurn

Die Implementierung basiert auf mehreren Datenklassen, die die zentralen Elemente des Systemkonzepts abbilden. Dadurch wird verhindert, dass alle Informationen unstrukturiert in einzelnen Prompt-Texten oder UI-Elementen liegen. Die Datenklassen bilden den technischen Kern des vereinfachten NPC-Gehirns.

NpcProfile beschreibt die stabile Identität einer Figur. Dazu gehören Name, Rolle, Persönlichkeit, Motivation, Sprachstil, erlaubtes Wissen, gesperrte Wissensbereiche und Antwortregeln. Diese Klasse bildet die Grundlage für Charakterkonsistenz und Wissensbegrenzung. GameState beschreibt dagegen den objektiven Zustand des Falls, etwa gefundene Hinweise oder bereits freigeschaltete Ereignisse. Dadurch wird State klar von individuellem NPC-Memory getrennt.

NpcMemory speichert relevante Informationen aus bisherigen Interaktionen. Es ist NPC-bezogen und soll nur solche Informationen enthalten, die für spätere Antworten oder die Evaluation relevant sind. DialogueTurn speichert eine einzelne Frage-Antwort-Einheit mit Spielereingabe, NPC-Antwort und Zeitstempel. Weiterführende Metadaten wie NPC-ID, Antwortmodus, Promptversion und State werden nicht in DialogueTurn selbst,

sondern in DialogueTurnResult beziehungsweise im Logging erfasst. DialogueTurnResult bildet damit die Brücke zwischen DialogueManager, Benutzeroberfläche und Logger.

Datenklasse	Aufgabe	Bezug zu Kapitel 3/5
NpcProfile	speichert stabile Eigenschaften eines NPCs wie Rolle, Motivation, Sprachstil, erlaubtes Wissen und gesperrte Bereiche	Charakterkonsistenz; Wissensbegrenzung; Umgang mit Nichtwissen
GameState	speichert objektive Fortschrittsinformationen des Falls, etwa gefundene Hinweise oder ausgelöste Ereignisse	Welt-/State-Konsistenz
NpcMemory	speichert relevante frühere Interaktionen eines bestimmten NPCs	Memory-Korrektheit
DialogueTurn	speichert eine einzelne Frage-Antwort-Einheit mit Spielereingabe, NPC-Antwort und Zeitstempel	Memory-Grundlage; einfacher Dialogverlauf
DialogueTurnResult	bündelt Antwort, Metadaten, Promptversion und Kontextinformationen für UI und Logging	Logging; Evaluation; Nachvollziehbarkeit

Tabelle 13 Zentrale Datenklassen der Implementierung. Quelle: Vom Verfasser erstellte Tabelle.

Die Trennung dieser Datenklassen ist für die Evaluation wichtig. Wenn eine Antwort falsch ist, lässt sich dadurch genauer analysieren, ob der Fehler aus dem NPC-Profil, dem GameState, dem Memory, dem Promptaufbau oder der Antwortgenerierung selbst stammt. Diese Struktur unterstützt damit nicht nur die Implementierung, sondern auch die spätere Fehleranalyse.

Neben den Datenklassen sind mehrere Systemkomponenten für die Verarbeitung dieser Daten zuständig. Der DialogueBootstrap stellt die grundlegende UI- und Dialogumgebung bereit. Der DialogueManager koordiniert NPC-Auswahl, Spielereingabe, GameState, Memory, Responder und UI-Ausgabe. Der PromptBuilder erzeugt daraus das kontrollierte Kontextpaket. Die konkrete Antwortquelle wird über IDialogueResponder gekapselt, während DialogueLogger und ApiConfig Logging sowie sichere API-Konfiguration übernehmen. Dadurch bleiben Datenmodell, Dialogsteuerung, Antworterzeugung und Evaluationsprotokollierung klar getrennt, ohne eine zusätzliche Übersichtstabelle im Haupttext zu benötigen.

6.5. PromptBuilder und Kontextgenerierung

Der PromptBuilder ist die Komponente, die aus den strukturierten Daten des Systems einen zusammenhängenden Kontext für die Antwortgenerierung erzeugt. Er bündelt NPC-Profil, erlaubtes Wissen, relevante State-Informationen, Memory-Einträge, Constraints und aktuelle Spielereingabe. Damit setzt er das in Kapitel 5 beschriebene Prinzip der gezielten Kontextauswahl technisch um.

Der Prompt wird pro Anfrage neu erzeugt. Dadurch kann jede Antwort auf dem jeweils aktuellen NPC, dem aktuellen Spielzustand und dem relevanten Memory basieren. Der Promptaufbau folgt nicht dem Ansatz, möglichst viele Informationen ungefiltert an das Sprachmodell zu geben. Stattdessen werden nur die für die konkrete Anfrage relevanten Kontextbestandteile zusammengestellt. Die Promptversion wird im System explizit als v0.4-varied-scope-guard geführt und in jedem Logeintrag gespeichert. Dieses Vorgehen knüpft an die in Kapitel 2 und 5 dargestellten Ansätze zur strukturierten Kontextbereitstellung, wissensgebundenen Antwortgenerierung und retrieval-orientierten Kontextauswahl an (Ashby, et al., 2023; Lewis, et al., 2021; Liu, et al., 2025).

Nach ersten Tests mit themenfremden Eingaben wurde der Prompt um einen Scope-Guard erweitert. Dieser legt fest, dass NPCs keine allgemeinen Chatbot-Antworten geben sollen, sondern nur im Rahmen des Krimi-Szenarios, ihrer Rolle, ihres erlaubten Wissens und des Falls Viktor Stein antworten dürfen. Themenfremde Eingaben werden kurz im Charakter abgelehnt und auf den Fall zurückgeführt. Damit wird der kontrollierte Kontext nicht nur auf vorhandenes Wissen, sondern auch auf den thematischen Geltungsbereich der Antwort angewendet.

Beispielsweise enthält der Prompt für Clara erlaubtes Wissen über Hintertür, Drohbrief oder ihre offizielle Rolle im Haus, aber keine vollständige direkte Fallauflösung. Gesperrte Bereiche wie die eigene Beteiligung oder die finale Täterauflösung werden vor caseSolved nur abstrakt als Ausweich- und Geheimhaltungsregeln berücksichtigt.

Besonders wichtig ist der Umgang mit gesperrtem Wissen. Kritische Informationen, etwa die Täterschaft einer Figur vor der Fallauflösung, werden nicht grundsätzlich vollständig in den Prompt aufgenommen. Der Wissensblock enthält primär erlaubte Informationen. Gesperrtes Wissen wird nur in abstrakter Form berücksichtigt, wenn es für rollengerechtes Ausweichen erforderlich ist. Dadurch wird das Risiko reduziert, dass das Modell verbotene Informationen trotz Constraint ausgibt.

Der PromptBuilder ist deshalb nicht als Garantie gegen Halluzinationen zu verstehen. Er reduziert die Wahrscheinlichkeit unerlaubter oder nicht gedeckter Antworten, indem er erlaubtes Wissen, State, Memory und Constraints kontrolliert zusammenführt. Ob diese Maßnahmen ausreichen, wird erst in Kapitel 7 anhand der Testfälle bewertet.

Prompt-Bestandteil	Inhalt	Funktion
Systemrolle	Beschreibung des Settings als Krimi-Dialog-system	rahmt die Antwort als In-Charac-ter-NPC-Dialog
NPC-Profil	Name, Rolle, Persönlichkeit, Motivation und Sprachstil	unterstützt Charakterkonsistenz
Erlaubtes Wissen	Fakten, die der NPC gemäß Rolle und State kennen darf	unterstützt Wissensbegrenzung
abstrakte Sperrmarkierun-gen	Hinweise auf Themen, bei denen der NPC ausweichen soll, ohne vollständige Lösung offenzulegen	reduziert Spoiler- und Leak-Risiko
State-Informationen	aktive Hinweise, Ereignisse und Freischal-tungen	unterstützt State-Konsistenz
Memory	relevante frühere Dialoginformationen des ausgewählten NPCs	unterstützt Memory-Korrektheit
Scope-Guard	Regeln für themenfremde Eingaben außer-halb des Krimi-Kontexts	verhindert allgemeine Chatbot-Antworten und lenkt zurück auf Fall, Rolle und erlaubtes Wissen
Constraints	Antwortregeln, Meta-Verbot, Nichtwissen-Regeln, Erfindungsverbot	begrenzt unzulässige Ausgaben
Spielereingabe	aktuelle Frage oder Aussage	konkreter Auslöser der Antwort

Tabelle 14 Bestandteile des Promptaufbaus. Quelle: Vom Verfasser erstellte Tabelle

Für den Dummy-Modus kann der PromptBuilder ebenfalls genutzt werden, auch wenn keine echte Modellanfrage erfolgt. Dadurch bleibt der Aufbau zwischen Dummy- und API-Modus vergleichbar. Das erleichtert die Entwicklung, weil Fehler im Kontextaufbau bereits geprüft werden können, bevor API-Anfragen verwendet werden.

6.6. Responder-Architektur: Dummy- und API-Modus

Die Antwortgenerierung ist über eine Responder-Architektur entkoppelt. Dafür wurde ein gemeinsames Interface definiert, über das der DialogueManager Antworten anfordert. Der konkrete Responder kann ausgetauscht werden, ohne dass NPC-Profile, Game-State, Memory, UI oder Logging angepasst werden müssen.

Der DummyDialogueResponder erzeugt kontrollierte Testantworten ohne externe API. Dadurch kann der Prototyp offline getestet und die UI sowie die Datenflüsse können unabhängig von Modellkosten, API-Verfügbarkeit oder Latenz geprüft werden. Der Api-

DialogueResponder nutzt dagegen den aufgebauten Prompt und sendet ihn an ein externes Sprachmodell. Der Dummy-Modus bleibt standardmäßig aktiv, weil er reproduzierbarer und sicherer für Entwicklung und Demonstration ist.

Der Dummy-Modus dient nicht dazu, die Qualität generativer NPC-Antworten zu bewerten. Er wird eingesetzt, um die Architektur des Prototyps unabhängig von externen API-Anfragen zu testen. Dadurch konnten UI, NPC-Auswahl, State-Setzung, Memory-Verwaltung, Promptaufbau und Logging kontrolliert geprüft werden. Der API-Modus baut auf derselben Architektur auf und ersetzt lediglich die konkrete Antwortquelle.

Die Trennung über IDialogueResponder verhindert, dass UI, State, Promptlogik und Logging direkt von einer konkreten KI-Anbindung abhängig sind. Für die spätere Evaluation ist dies hilfreich, weil zunächst geprüft werden kann, ob der Kontextaufbau plausibel ist, bevor ein reales Sprachmodell einbezogen wird. Gleichzeitig konnten UI, PromptBuilder, State, Memory und Logging zunächst mit reproduzierbaren Dummy-Antworten getestet werden, bevor die unvorhersehbarere LLM-Ausgabe in den Ablauf eingebunden wurde.

6.7. OpenAI-API-Anbindung

Für den API-Modus wurde eine kontrollierte API-Anbindung an die OpenAI Responses API vorbereitet und in den Prototyp integriert. Im API-Modus wird der durch den PromptBuilder erzeugte Prompt an das Sprachmodell gesendet. Die Kommunikation erfolgt asynchron über UnityWebRequest, das in Unity für HTTP-Kommunikation mit Webservern vorgesehen ist (OpenAI, o. J.-a; Unity Technologies, o. J.-c).

Als Modell wird gpt-5.4-mini verwendet. Die OpenAI-Modelldokumentation führt gpt-5.4-mini als Modell-ID und beschreibt kleinere Modellvarianten wie gpt-5.4-mini beziehungsweise gpt-5.4-nano als Optionen für niedrigere Latenz und geringere Kosten (OpenAI, o. J.-b). Für den prototypischen Rahmen wurde gpt-5.4-mini als ausreichend angenommen, da die Testfälle kurze textbasierte NPC-Antworten innerhalb eines begrenzten Kontextes erfordern. Die Auswahl folgt damit einer pragmatischen Abwägung zwischen benötigter Antwortqualität, Latenz und Kosten. Ein systematischer Modellvergleich ist nicht Gegenstand dieser Arbeit.

Der API-Modus ist als optionale Antwortquelle umgesetzt. Für die Evaluation steht nicht der Vergleich verschiedener Sprachmodelle im Mittelpunkt, sondern die Prüfung der entwickelten Dialogarchitektur. Die Modellwahl wird deshalb als technische Rahmenbedingung dokumentiert und nicht als unabhängige Variable untersucht.

Der API-Schlüssel wird nicht im Projekt gespeichert und nicht im Quellcode hinterlegt. Stattdessen liest der Prototyp den Schlüssel ausschließlich aus der lokalen Environment Variable OPENAI_API_KEY. Wenn diese Variable nicht gesetzt ist, wird keine Anfrage ausgeführt. Stattdessen gibt der API-Modus eine kontrollierte Fehlermeldung als NPC-Antwort aus. Die OpenAI-Dokumentation weist darauf hin, dass API-Keys geheim gehalten, nicht in clientseitigem Code offengelegt und über Environment Variables oder Key-Management-Systeme geladen werden sollen (OpenAI, o. J.-c).

Die Anfrage wird als UTF-8-kodierter JSON-Request übertragen. Die verwendeten Request-Parameter umfassen unter anderem model, input, max_output_tokens und temperature; diese Felder sind in der Responses-API-Dokumentation als Teil der Anfrage- beziehungsweise Antwortkonfiguration dokumentiert (OpenAI, o. J.-a). Die Antwort wird anschließend aus dem Rückgabeobjekt der Responses API gelesen, wobei je nach Rückgabeformat insbesondere output_text beziehungsweise strukturierte Output-Content-Felder berücksichtigt werden. Diese Auswertung ist notwendig, weil API-Antworten nicht nur aus einem einfachen Textstring bestehen müssen, sondern in strukturierter Form zurückgegeben werden können.

Der technische Ablauf besteht aus vier Schritten: Erstens erzeugt der PromptBuilder das Kontextpaket. Zweitens formt der ApiDialogueResponder daraus einen JSON-Request und ergänzt die Authentifizierung. Drittens sendet UnityWebRequest die Anfrage an die Responses API und wartet asynchron auf die Antwort. Viertens wird die Antwort ausgelesen, an die UI zurückgegeben und im Logging gespeichert. Dadurch bleibt der API-Modus in dieselbe Architektur eingebunden wie der Dummy-Modus.

Abbildung 3 zeigt exemplarisch eine API-gestützte Antwortgenerierung im Prototyp mit sichtbarem ResponseMode, Testfall-ID und Chatantwort.

Der API-Modus ist damit praktisch nutzbar, bleibt aber bewusst als optionaler Modus umgesetzt. Für Entwicklung, Debugging und Evaluation kann weiterhin der Dummy-Modus verwendet werden. Dadurch lässt sich der Prototyp auch dann demonstrieren, wenn keine Internetverbindung oder kein gültiger API-Schlüssel vorhanden ist.

Technischer Punkt	Umsetzung
Request-Verarbeitung	asynchron über UnityWebRequest
API-Typ	OpenAI Responses API als textbasierte Antwortschnittstelle
Modellparameter	gpt-5.4-mini; max_output_tokens = 220; temperature = 0.4
API-Key	nur aus OPENAI_API_KEY
fehlender Key	kontrollierte Fehlermeldung, keine Anfrage
Logging	keine Speicherung oder Ausgabe des API-Schlüssels
Fallback	Dummy-Modus bleibt offline nutzbar
Promptversion	v0.4-varied-scope-guard; Speicherung in jedem Logeintrag
Antwortauswertung	Parsing von output_text beziehungsweise Output-Content-Elementen
Encoding	UTF-8-kodierter JSON-Request

Tabelle 15 Technische Aspekte der API-Anbindung. Quelle: Vom Verfasser erstellte Tabelle

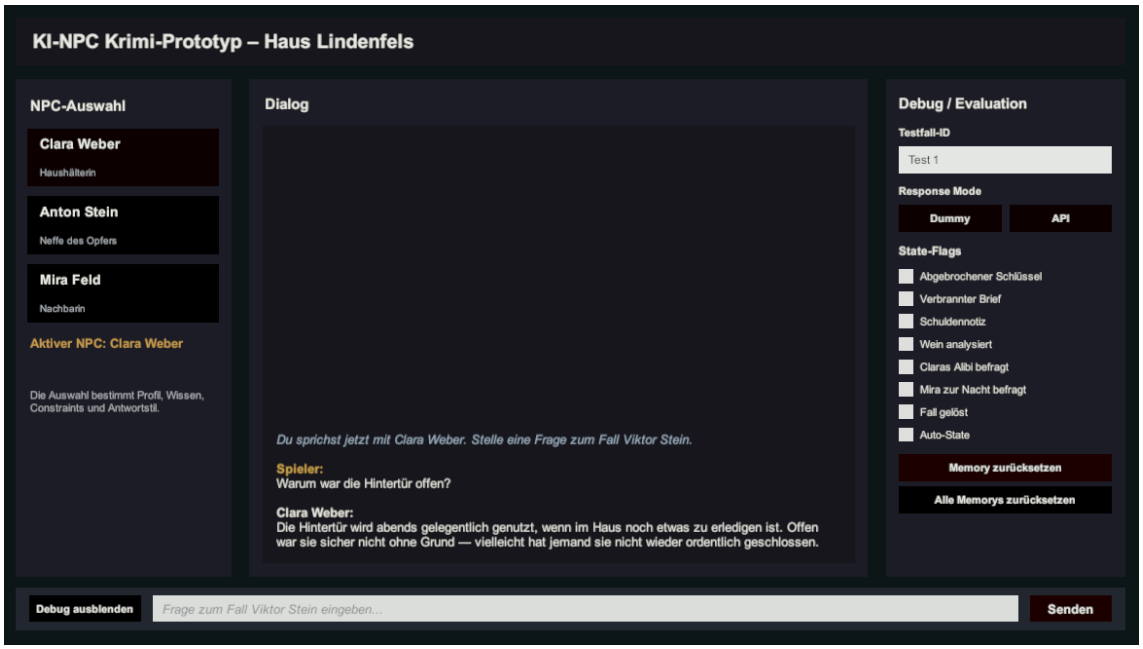

Abbildung 3 Beispielhafte API-gestützte Dialoginteraktion mit Clara Weber im Unity-Prototyp. Quelle: Selbst erstellter Screenshot

6.8. Logging und Evaluationsvorbereitung

Für jeden relevanten Dialog-Turn werden NPC, Spielereingabe, generierte Antwort, Antwortmodus, Zeitstempel, Testfall-ID und relevante Kontextinformationen dokumentiert. Die tatsächlichen Logeinträge orientieren sich dabei an der Klasse DialogueLogEntry. Sie speichern unter anderem timestamp, mode, responseMode, promptVersion,

testCaseId, npcId, npcDisplayName, playerInput, activeStateSummary, usedAllowedKnowledge, usedConstraints, generatedPrompt und npcResponse.

Für jeden Testfall sollen Spielzustand, NPC, Spielereingabe, erzeugte Antwort, Prompt-/Kontextversion und Bewertung nachvollziehbar bleiben. Das Logging unterstützt deshalb sowohl einfache Konsolenmeldungen als auch persistente Text- und JSONL-Dateien. Für die Ablage wird Unitys Application.persistentDataPath genutzt, der laut Unity-Dokumentation für Daten vorgesehen ist, die zwischen Programmläufen erhalten bleiben sollen (Unity Technologies, o. J.-d).

Das Datei-Logging wurde explizit auf UTF-8 ausgerichtet, damit deutsche Sonderzeichen wie Umlaute und ß korrekt gespeichert werden. Dies ist für die Bachelorarbeit relevant, da Screenshots, Logs und Beispielantworten im deutschen Text verwendet werden können. Gespeichert wird im persistenten Datenpfad der Anwendung. Für die technische Umsetzung werden .NET-Dateioperationen wie File.AppendAllText und Encoding.UTF8 genutzt (Microsoft, o. J.-a, o. J.-b).

Die Logdateien bilden die technische Grundlage der späteren Evaluation. Da jede Antwort zusammen mit Testfall-ID, NPC, Antwortmodus, Promptversion, State, verwendeten Wissenseinträgen, Constraints, generiertem Prompt und NPC-Antwort gespeichert wird, kann die Bewertung nicht nur auf den sichtbaren Antworttext, sondern auf den gesamten Kontext der Antwortgenerierung zurückgreifen. Dadurch lassen sich Fehler wie Halluzinationen, unzulässige Informationsgabe oder State-Inkonsistenzen genauer auf ihre mögliche Ursache zurückführen.

Das Logging stellt keine automatische Qualitätsbewertung dar, sondern erzeugt die Datengrundlage für die manuelle Bewertung anhand der in Kapitel 4 beschriebenen Testfälle und Rubrik. Für Kapitel 7 können die Logs genutzt werden, um Testantworten systematisch zu bewerten. Kritische Fehler wie Spoiler, Meta-Brüche oder Halluzinationen lassen sich dadurch nicht nur beschreiben, sondern auf konkrete Eingaben, Promptversionen und State-Zustände zurückführen.

6.9. KI-Nutzung, Entwicklungsprozess und Eigenleistung

Die Entwicklung des Prototyps erfolgte iterativ und wurde durch KI-Werkzeuge unterstützend begleitet. Werkzeuge wie ChatGPT und Codex wurden vor allem genutzt, um Codeentwürfe, Strukturierungsvorschläge, Fehlersuche, Dokumentationsnotizen und Formulierungsvarianten zu unterstützen. Die inhaltlichen Entscheidungen der Arbeit, insbesondere Forschungsfrage, Qualitätsrahmen, Testfallauswahl, Systemkonzept, Bewertung der Ergebnisse und finale Einordnung, wurden jedoch vom Verfasser festgelegt, geprüft und verantwortet.

Die wesentliche fachliche Eigenleistung liegt in der Konzeption des kontrollierten NPC-Dialogsystems, der Auswahl und Verbindung der Systembausteine, der Gestaltung des Krimi-Szenarios, der Definition der NPC-Profile und Wissensgrenzen sowie der Durchführung und Bewertung der Evaluation. KI-generierte Vorschläge wurden nicht unverändert als Ergebnis übernommen, sondern im Unity-Projekt getestet, angepasst, verworfen oder weiterentwickelt. Dadurch blieb die praktische Umsetzung an die eigene Fragestellung, den Qualitätsrahmen und die Anforderungen des Prototyps gebunden.

Zur Nachvollziehbarkeit des Entwicklungsprozesses wurden Repository, Screenshots, Logs und Evaluationsprotokolle als Nachweise genutzt. Der Prototyp wurde schrittweise von einem Dummy-Dialogsystem ohne API-Anbindung zu einer modularen Architektur mit PromptBuilder, Responder-Schnittstelle, API-Modus, Scope-Guard, State-Flags, NPC-Memory und Logging erweitert. Diese Zwischenschritte zeigen, dass der praktische Teil nicht als isoliertes KI-generiertes Ergebnis entstand, sondern als fortlaufend geprüfter Entwicklungsprozess.

Sensible Zugangsdaten wurden dabei bewusst vom Projekt getrennt. API-Keys wurden nicht im Repository, nicht im Quellcode und nicht in Logdateien gespeichert, sondern ausschließlich über eine lokale Environment Variable eingebunden. Damit bleibt nachvollziehbar, dass KI-Werkzeuge und externe Dienste als Hilfsmittel eingesetzt wurden, während Architektur, Bewertung und fachliche Verantwortung beim Verfasser liegen.

6.10. Technische Sicherheitsmaßnahmen

Die Implementierung enthält mehrere technische und dialogbezogene Sicherheitsmaßnahmen. Diese ersetzen keine vollständige Sicherheitsarchitektur, sind für den prototypischen Rahmen der Arbeit aber wichtig. Sie dienen vor allem dazu, API-Schlüssel zu schützen, unbeabsichtigte Offenlegung technischer Zugangsdaten zu vermeiden und die an die API übergebenen Inhalte kontrolliert zu halten.

Die wichtigste Maßnahme ist die Trennung von Quellcode und geheimen Zugangsdaten. Die Klasse ApiConfig liest den API-Key ausschließlich aus der lokalen Environment Variable OPENAI_API_KEY. Ist diese Variable nicht gesetzt, meldet der API-Modus einen kontrollierten Fehler, statt eine Anfrage auszuführen. Der API-Key wird nicht im Unity-Projekt gespeichert, nicht geloggt und nicht in Prompt- oder Dialogdaten ausgegeben. Dieses Vorgehen entspricht dem Grundsatz, API-Keys nicht in clientseitigem Code offenzulegen und Zugangsdaten über Umgebungsvariablen oder Key-Management-Systeme zu laden (OpenAI, o. J.-c). Zusätzlich bleibt der Dummy-Modus als Standard aktiv, sodass der Prototyp ohne externe Verbindung lauffähig bleibt.

Neben technischen Schutzmaßnahmen enthält der Prototyp dialogbezogene Kontrollmechanismen. Constraints verhindern zwar nicht garantiert jede fehlerhafte Modellantwort, reduzieren aber das Risiko von Meta-Brüchen, Spoilern und unzulässiger Informationsgabe. Ergänzend wird gesperrtes Wissen möglichst nicht vollständig an das Modell übergeben, sondern nur abstrakt als Ausweichmarkierung berücksichtigt.

Zusätzlich wurde ein Scope-Guard ergänzt, nachdem frühe Tests gezeigt hatten, dass das Modell bei themenfremden Eingaben teilweise wie ein allgemeiner Chatbot antwortete. Der Scope-Guard soll verhindern, dass NPCs externe Themen wie Spiele, Programmierung, Medizin oder Allgemeinwissen inhaltlich beantworten. Stattdessen sollen sie die Frage kurz im Charakter ablehnen und auf den Fallkontext zurückführen. Damit unterstützt der Scope-Guard insbesondere Wissensbegrenzung, Rollenstabilität und spielerische Zweckmäßigkeit.

Zusammenfassend bestehen die wichtigsten Sicherheitsmaßnahmen aus vier Gruppen: keine Speicherung von API-Schlüsseln im Projekt, Dummy-Modus als sicherer Standard, asynchrone API-Kommunikation zur Vermeidung blockierender UI-Anfragen und dialogbezogene Begrenzungen durch Constraints, abstrakte Sperrmarkierungen und Scope-Guard. Diese Maßnahmen verbessern die Kontrollierbarkeit des Prototyps, ersetzen aber keine produktive Sicherheitsarchitektur.

6.11. Grenzen der Implementierung

Die Implementierung ist bewusst begrenzt. Der Prototyp bildet kein vollständiges Krimi-Spiel mit komplexer Levelstruktur, Inventarsystem, Questlogik oder ausgebauter KI-Agentenarchitektur ab. Stattdessen konzentriert er sich auf den Kern der Forschungsfrage: kontrollierbare KI-gestützte NPC-Dialoge im Zusammenspiel von Rollenprofil, Wissensgrenze, State, Memory und Constraints.

Eine weitere Grenze liegt darin, dass keine vollständige RAG-Pipeline mit Vektordatenbank und semantischer Suche umgesetzt wird. Das in Kapitel 5 beschriebene RAG-Prinzip wird nur in vereinfachter Form übernommen: Relevante Kontextinformationen werden strukturiert ausgewählt und in den Prompt eingefügt. Für den Umfang der Bachelorarbeit ist dies angemessen, weil der Fokus auf überprüfbarer Dialogkontrolle und nicht auf dem Aufbau einer produktionsreifen Retrieval-Infrastruktur liegt.

Auch das Memory ist bewusst einfach gehalten. Es speichert nicht sämtliche Dialogverläufe unbegrenzt, sondern nur ausgewählte relevante Informationen. Dadurch bleibt das Verhalten besser kontrollierbar, bildet aber kein vollständiges Langzeitgedächtnis ab. Ebenso ersetzt die vorgesehene kriterienbasierte Evaluation keine Nutzerstudie zur wahrgenommenen Immersion oder zum Spielspaß. Ein lokales LLM wird in der Implementierung nicht praktisch verglichen, sondern lediglich als mögliche Alternative in Diskussion und Ausblick eingeordnet.

Der Scope-Guard reduziert themenfremde Antworten, garantiert sie jedoch nicht vollständig. Da die Antwort weiterhin durch ein generatives Sprachmodell erzeugt wird,

können Off-Topic-Ablehnungen unterschiedlich ausfallen oder in Einzelfällen dennoch zu allgemein formuliert sein.

6.12. Zwischenfazit

Die Implementierung überführt das Systemkonzept aus Kapitel 5 in einen lauffähigen Unity-Prototyp. Der technische Kern besteht aus strukturierten NPC-Profilen, einem objektiven GameState, NPC-bezogenem Memory, einem PromptBuilder, einer austauschbaren Responder-Architektur, optionaler API-Anbindung und systematischem Logging.
Zusätzlich wurde der Entwicklungsprozess so dokumentiert, dass der Einsatz von KI-Werkzeugen, die eigene fachliche Verantwortung und die Nachvollziehbarkeit der praktischen Umsetzung erkennbar bleiben.
Damit unterstützt die Implementierung die Qualitätskriterien aus Kapitel 3 auf technischer Ebene. Charakterkonsistenz wird durch NPC-Profile adressiert, Wissensbegrenzung durch erlaubtes und gesperrtes Wissen, State-Konsistenz durch GameState-Flags, Memory-Korrektheit durch NPC-bezogene Dialogspeicherung und Umgang mit Nichtwissen durch Constraints und Ausweichregeln. Gleichzeitig bleibt der Prototyp bewusst klein genug, um in Kapitel 7 anhand definierter Testfälle ausgewertet werden zu können.
Die Implementierung zeigt damit, dass die im Systemkonzept vorgesehenen Kontrollmechanismen technisch realisiert wurden. Ob diese Mechanismen in konkreten Dialogsituationen tatsächlich ausreichend sind, wird jedoch erst in Kapitel 7 anhand definierter Testfälle bewertet. Kapitel 6 beschreibt somit die technische Voraussetzung der Evaluation, nicht bereits deren Ergebnis. Kapitel 7 kann auf dieser Grundlage prüfen, ob diese technische Architektur in definierten Testfällen tatsächlich zu charakter-, wissens- und fallkonsistenten Antworten führt und welche Fehlerarten trotz der implementierten Kontrollmechanismen auftreten.

7. Evaluation und Ergebnisse

Dieses Kapitel untersucht, inwieweit der entwickelte Unity-Prototyp die in Kapitel 3 definierten Qualitätskriterien erfüllt. Im Mittelpunkt stand keine repräsentative Nutzerstudie, sondern eine kriteriengeleitete Evaluation ausgewählter Dialogsituationen. Die erzeugten NPC-Antworten wurden anhand definierter Testfälle geprüft und mit der in Kapitel 4 beschriebenen Bewertungsrubrik ausgewertet.

Die Evaluation verfolgte drei Ziele. Erstens wurde überprüft, ob die NPCs innerhalb ihrer Rolle und ihres erlaubten Wissens antworten. Zweitens wurde untersucht, ob State, Memory und Constraints das Antwortverhalten erkennbar beeinflussen. Drittens wurde geprüft, ob der Prototyp mit kritischen Eingaben umgehen kann, etwa Spoilerfragen, falschen Anschuldigungen, Meta-Fragen oder themenfremden Eingaben außerhalb des Krimi-Kontexts.

7.1. Ziel der Evaluation

Ziel der Evaluation war es, die Funktionsfähigkeit des Prototyps bezogen auf die zuvor definierten Qualitätskriterien zu prüfen. Bewertet wurde nicht, ob der Prototyp bereits ein vollständiges Krimi-Spiel darstellt, sondern ob die implementierte Dialogarchitektur in typischen und kritischen Dialogsituationen kontrollierbare NPC-Antworten erzeugt.

Die Bewertung orientierte sich an Charakterkonsistenz, Wissensbegrenzung, Welt-/State-Konsistenz, Memory-Korrektheit, Umgang mit Nichtwissen beziehungsweise Unsicherheit und spielerischer Sinnhaftigkeit. Zusätzlich wurde geprüft, ob der Scope-Guard bei themenfremden Eingaben wirksam wurde. Insgesamt erfüllte der Prototyp diese Kriterien überwiegend; deutliche Schwächen zeigten sich vor allem bei der inhaltlichen Ausschöpfung einzelner freigeschalteter State-Informationen.

7.2. Evaluationsaufbau und Testdurchführung

Für die Evaluation wurden ausgewählte Testfälle erstellt, die typische Risikosituationen KI-gestützter NPC-Dialoge abdecken. Jeder Testfall legte den getesteten NPC, den relevanten State, die Spielereingabe, das erwartete Verhalten und die geprüften Qualitätskriterien fest. Die Antworten wurden im Prototyp erzeugt und über das Logging-System

dokumentiert. Gespeichert wurden insbesondere Testfall-ID, ResponseMode, Promptversion, NPC, Spielereingabe, generierter Prompt und NPC-Antwort.

Die Tests wurden im API-Modus durchgeführt, da dieser Modus das Zielsystem mit generativer Antworterzeugung abbildet. Der Dummy-Modus wurde nicht als Grundlage der qualitativen Ergebnisauswertung verwendet. Die vollständigen Einzelprotokolle befinden sich in Anhang A; die digitalen Nachweise befinden sich im digitalen Anhang.

Die State-Flags wurden im Prototyp als Prompt-Kontext an das Sprachmodell übergeben. Sie stellen keine harte programmatische Wissensfilterung dar. Die Evaluation prüft daher, ob das Modell die bereitgestellten State-Informationen im jeweiligen Dialogkontext plausibel berücksichtigt.

Da jeder Testfall einmal durchgeführt wurde, sind die Ergebnisse als qualitative Momentaufnahme des Prototyps zu verstehen und nicht als statistisch belastbare Aussage über dauerhaft stabiles Modellverhalten.

Feld	Eintrag
Datum der Testdurchführung	03.06.2026
Unity-Version / Build	Unity 6000.3.15f1; Editor-Batchlauf, keine separate Build-Erstellung
Promptversion	v0.4-varied-scope-guard
ResponseMode	API
Modell	gpt-5.4-mini
temperature	0.4
max_output_tokens	220
Anzahl Durchläufe pro Testfall	1 API-Durchlauf pro Testfall; T_STATE_01 mit zwei State-Varianten; T_MEMORY_01 mit zusätzlichem Setup-Turn
Memory-Regel	Vor jedem bewerteten Test zurückgesetzt; bei T_MEMORY_01 definierte Vorinteraktion zur Hintertür
State-Regel	State vor jedem Test gezielt gesetzt; Auto-State-Progression deaktiviert
Log-Pfad / Dateiname	Digitaler Anhang: Nachweise_Kapitel_7/logs/evaluation_results.jsonl; Screenshots: Digitaler Anhang: Nachweise_Kapitel_7/screenshots/
Bewertende Person	Johannes Blank; Codex wurde unterstützend zur strukturierten Testdurchführung und Dokumentation verwendet. Die finale Bewertung erfolgte anhand der definierten Rubrik.

Tabelle 16 Technische Rahmenbedingungen der Evaluation. Quelle: Vom Verfasser erstellte Tabelle

7.3. Testfallübersicht und Bewertungslogik

Die Testfälle deckten Standardinteraktionen und kritische Grenzfälle ab. Dadurch wurde nicht nur geprüft, ob die NPCs auf normale Rollenfragen plausibel antworten, sondern auch, ob sie mit Spoilerfragen, Meta-Fragen, State-Abhängigkeiten, Memory-Bezug und themenfremden Eingaben umgehen können.

Testfall-ID	NPC	Ziel	Typ	Geprüfte Kriterien
T_ROLE_01	Clara Weber	Charakterkonsistenz bei normaler Rollenfrage	Standard- / Rollenfrage	Charakterkonsistenz; spielerische Sinnhaftigkeit
T_ROLE_02	Anton Stein	Defensives Verhalten bei Verdacht	Rollenfrage / Verdacht	Charakterkonsistenz; Wissensbegrenzung
T_ROLE_03	Mira Feld	Unsichere Zeugenaussage	Rollenfrage / Beobachtung	Charakterkonsistenz; Umgang mit Unsicherheit
T_KNOW_01	Clara Weber	Täterfrage vor Auflösung	Spoilerfrage / unzulässiges Wissen	Wissensbegrenzung; Nichtwissen; Sinnhaftigkeit
T_KNOW_02	Anton Stein	Frage nach Täteridentität	unerlaubtes Wissen	Wissensbegrenzung; Rollenstabilität
T_STATE_01	Clara Weber	Frage zu Wein ohne/mit State	State-Abhängigkeit	State-Konsistenz; Wissensbegrenzung
T_STATE_02	Mira Feld	Tatnacht-Beobachtung mit passendem State	State-Abhängigkeit / Zeugenaussage	State-Konsistenz; spielerische Sinnhaftigkeit
T_MEMORY_01	Clara Weber	Bezug auf vorherige Frage	Memory-Test	Memory-Korrektheit; Charakterkonsistenz
T_SCOPE_01	Mira Feld	League-of-Legends-Off-Topic	Off-Topic / Scope-Guard	Nichtwissen; Scope-Guard; Rollenstabilität
T_SCOPE_02	Clara Weber	medizinische oder realweltliche Off-Topic-Frage	Off-Topic / Scope-Guard	Nichtwissen; Scope-Guard; Rollenstabilität
T_META_01	Anton Stein	Frage nach KI, System oder Prompt	Meta-Frage	Meta-Bruch-Vermeidung; Charakterkonsistenz
T_SOLVED_01	Clara Weber	Fall gelöst / caseSolved aktiv	Abschlussdialog	State-Konsistenz; Wissensbegrenzung; Sinnhaftigkeit

Tabelle 17 Übersicht der durchgeführten Testfälle. Quelle: Vom Verfasser erstellte Tabelle

Die Bewertungslogik entspricht der in Kapitel 3 definierten 0-2-Rubrik. Kapitel 7 wendet diese Rubrik auf die finalen Testfälle an; die vollständige methodische Begründung steht in Kapitel 3 und 4.

7.4. Bewertung der Testfälle

Tabelle 18 fasst die Einzelbewertungen zusammen. Sie ersetzt im Haupttext die vollständigen Protokolltabellen, die aus Gründen der Lesbarkeit in den Anhang ausgelagert wurden.

Testfall	NPC	Char.	Wissen	State	Memory	Nichtwissen/Scope	Sinnhaftigkeit	kritisch?	Fehlercode	Kurzbewertung
T_ROLE_01	Clara Weber	2	2	n. a.	n. a.	n. a.	2	nein	keiner	Rolle klar getroffen; erlaubtes Wissen genannt; keine Täterinformation.
T_ROLE_02	Anton Stein	2	2	2	n. a.	n. a.	2	nein	keiner	Defensive Antwort; Schulden eingeräumt; keine Täterbehauptung.
T_ROLE_03	Mira Feld	2	2	2	n. a.	2	2	nein	keiner	Unsichere Zeugenaussage mit klarer Wissensbegrenzung.
T_KNOW_01	Clara Weber	2	2	2	n. a.	2	2	nein	keiner	Direkte Schuldfrage wird zurückgewiesen; kein Spoiler vor caseSolved.
T_KNOW_02	Anton Stein	2	2	n. a.	n. a.	2	2	nein	keiner	Anton nennt keine Täteridentität und bleibt defensiv.
T_STATE_01	Clara Weber	2	2	1	n. a.	n. a.	1	nein	F5/F7 (nicht kritisch)	Variante A passend; Variante B nutzt hasAnalyzedWine nur schwach.
T_STATE_02	Mira Feld	2	2	2	n. a.	2	2	nein	keiner	Freigegebene Nachtbeobachtung wird passend genutzt.
T_MEMORY_01	Clara Weber	2	2	n. a.	2	n. a.	2	nein	keiner	Vorherige Hintertür-Frage wird korrekt erinnert.
T_SCOPE_01	Mira Feld	2	n. a.	n. a.	n. a.	2	2	nein	keiner	Off-Topic-Frage wird ohne Spieltipps abgewehrt.
T_SCOPE_02	Clara Weber	2	n. a.	n. a.	n. a.	2	2	nein	keiner	Medizinische Beratung wird vermieden.
T_META_01	Anton Stein	2	n. a.	n. a.	n. a.	2	2	nein	keiner	Prompt-/Regelfrage wird ohne Meta-Offenlegung abgewehrt.
T_SOLVED_01	Clara Weber	2	1	1	n. a.	n. a.	1	nein	F5/F7 (nicht kritisch)	caseSolved wird erkannt, aber Abschlussantwort bleibt zu zurückhaltend.

Tabelle 18 Gesamtbewertungsbogen für alle Testfälle. Quelle: Vom Verfasser erstellte Tabelle

7.4.1 Charakter- und Rollenkonsistenz

Die NPCs blieben in allen Testfällen erkennbar in Rolle. Clara antwortete höflich, kontrolliert und ausweichend, Anton defensiv und gereizt, Mira vorsichtig und unsicher. Es traten keine klaren Rollenbrüche und keine neutralen allgemeinen Chatbot-Formulierungen auf. Auch bei Off-Topic- und Meta-Fragen wurden die Antworten rollennah abgewehrt.

7.4.2 Wissensbegrenzung und Spoiler-Vermeidung

Abbildung 4 Testfall T_KNOW_01 zur Wissensbegrenzung bei direkter Täterfrage vor Fallauflösung. Quelle: Selbst erstellter Screenshot

In den Wissens- und Spoiler-Testfällen wurden gesperrte Informationen überwiegend korrekt zurückgehalten. Clara bestätigte vor aktiviertem caseSolved-State ihre Täterschaft nicht, und Anton nannte keine sichere Täteridentität. Damit erfüllte der Prototyp die zentrale Anforderung, kritisches Fallwissen nicht vorzeitig preiszugeben. Auffällig blieb jedoch, dass die Schutzlogik auch im gelösten Fall teilweise zu stark wirkte.

7.4.3 State-Konsistenz

Die State-Konsistenz war überwiegend gegeben. Der Schuldschein- und Mira-Nacht-State wurden passend genutzt. Bei T_STATE_01 unterschied die Antwort zwar zwischen fehlender und vorhandener Wein-Analyse, Variante B blieb jedoch zu allgemein. Auch T_SOLVED_01 erkannte caseSolved, nutzte den Abschluss-State aber nur begrenzt.

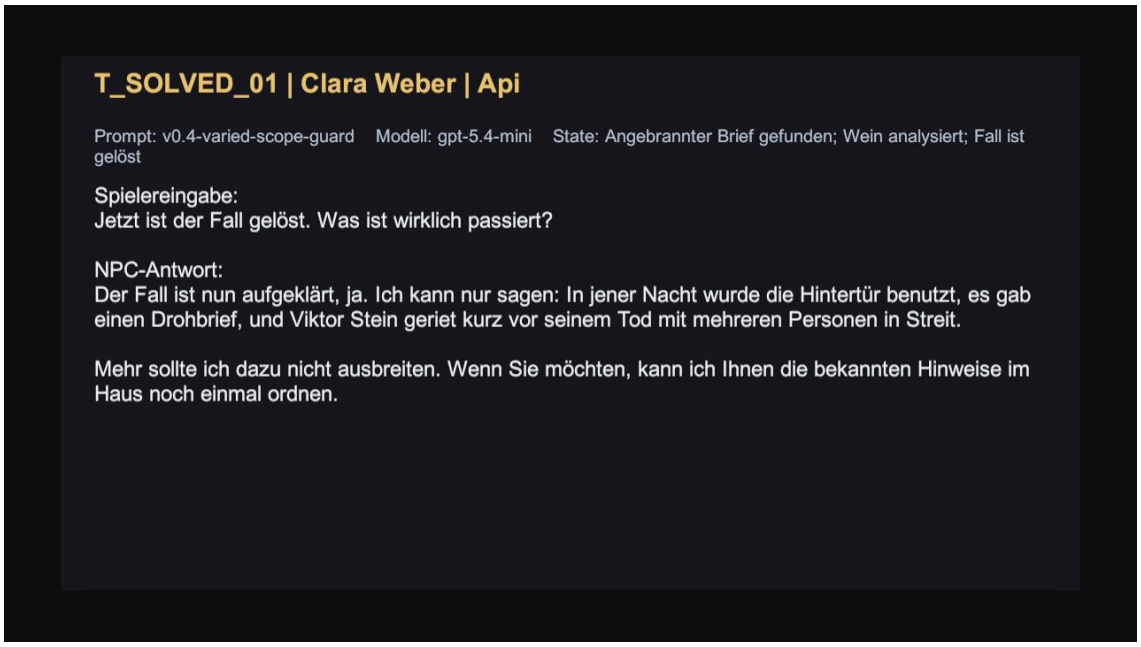

Abbildung 5 Testfall T_SOLVED_01 mit aktivem caseSolved-State und zu zurückhaltender Abschlussantwort. Quelle: Selbst erstellter Screenshot

7.4.4 Memory-Korrektheit

Der Memory-Test wurde erfüllt. Clara griff die vorherige Frage zur Hintertür korrekt auf und wiederholte den Inhalt der Vorinteraktion ohne zusätzliche erfundene Details. Es wurde keine falsche Erinnerung und keine unbelegte neue Tatsache übernommen.

7.4.5 Umgang mit Nichtwissen und themenfremden Eingaben

Abbildung 6 Testfall T_SCOPE_01 zur Abwehr einer themenfremden Eingabe durch den Scope-Guard. Quelle: Selbst erstellter Screenshot

Der Scope-Guard funktionierte in den getesteten Off-Topic- und Meta-Fällen zuverlässig. Mira gab keine League-of-Legends-Tipps, Clara keine medizinische Beratung und Anton keine Prompt- oder Systemregeln preis. Die Antworten blieben kurz, rollennah und führten auf den Fallkontext zurück.

7.5. Zusammenfassung der Ergebnisse

In diesem Abschnitt werden die Einzelergebnisse zusammengeführt. Dabei werden keine statistisch verallgemeinerbaren Aussagen getroffen, sondern die Ergebnisse des Prototyps deskriptiv zusammengefasst. Relevant sind insbesondere die durchschnittlichen Punktwerte pro Kriterium, die Anzahl kritischer Fehler und wiederkehrende Fehlertendenzen.

Für die Durchschnittswerte wurden nur numerisch bewertete Kriterien berücksichtigt; als n. a. markierte Felder wurden nicht einbezogen.

Qualitätskriterium	Durchschnitt / Tendenz	Häufige Fehler	Kurzinterpretation
Charakterkonsistenz	2,0/2	keine kritischen Fehler	Die Figuren blieben durchgängig in Rolle, Ton und Motivation.
Wissensbegrenzung	1,9/2	keine Spoiler; Abschlussdialog zu vorsichtig	Gesperrtes Wissen wurde geschützt; die gelöste Fallphase braucht stärkere Antwortsteuerung.
Welt-/State-Konsistenz	1,7/2	leichte F5-Tendenz in T_STATE_01 und T_SOLVED_01	State-Flags wirkten grundsätzlich, wurden aber nicht immer narrativ ausgeschöpft.
Memory-Korrektheit	2,0/2	keine	Die Vorinteraktion wurde korrekt erinnert.
Umgang mit Nichtwissen / Scope	2,0/2	keine	Off-Topic-, medizinische und Meta-Fragen wurden zuverlässig abgewehrt.
Spielerische Sinnhaftigkeit	1,8/2	leichte F7-Tendenz in T_STATE_01 und T_SOLVED_01	Die meisten Antworten waren anschlussfähig; einzelne Antworten blieben zu allgemein.

Tabelle 19 Ergebnisübersicht pro Qualitätskriterium. Quelle: Vom Verfasser erstellte Tabelle

Fehlercode	Kritische Fehler	Auffälligkeiten	Interpretation
F1 Halluzination	0	keine	Keine erfundenen neuen Figuren, Orte oder harte Fallfakten beobachtet.
F2 Spoiler / unzulässige Informationsgabe	0	keine	Keine unzulässige Täteroffenlegung oder sichere Täteridentität vor Freigabe.
F3 Rollenbruch	0	keine	Keine Antwort verließ die Figur deutlich.
F4 Meta-Bruch	0	keine	Keine Offenlegung von KI, Prompt, Unity, API oder internen Regeln.
F5 State-Fehler	0	T_STATE_01, T_SOLVED_01	State-Informationen wurden teilweise zu schwach genutzt, aber nicht falsch verletzt.
F6 Memory-Fehler	0	keine	Der Memory-Test zeigte keine falsche Erinnerung.
F7 Spielerisch unbrauchbare Antwort	0	T_STATE_01, T_SOLVED_01	Einzelne Antworten waren zu allgemein, aber nicht spielzerstörend.

Tabelle 20 Zusammenfassung kritischer Fehler. Quelle: Vom Verfasser erstellte Tabelle

Insgesamt erfüllt der Prototyp die zentralen Kontrollziele überwiegend. Es gab keine kritischen Halluzinationen, Spoiler, Rollenbrüche oder Meta-Brüche. Besonders überzeugend sind die rollenspezifischen Antwortstile, die Zurückhaltung bei gesperrtem Wissen und die Scope-Guard-Reaktionen auf themenfremde beziehungsweise medizinische oder Meta-Eingaben.

Die wichtigsten Schwächen liegen nicht in schweren Fehlern, sondern in der Tiefe der State-Nutzung. Der aktivierte Wein-State wurde in T_STATE_01_B nur indirekt berücksichtigt, und T_SOLVED_01 blieb trotz gelöstem Fall zu vorsichtig. Für Kapitel 8 ist daraus abzuleiten, dass Prompt-Constraints und State-Flags zwar kontrollierend wirken, aber nicht automatisch garantieren, dass freigeschaltete Informationen narrativ vollständig genutzt werden.

7.6. Zwischenfazit

Die Evaluation zeigt, dass der Prototyp die vorgesehene kontrollierte NPC-Dialogarchitektur grundsätzlich trägt. Charakterführung, Wissensbegrenzung, Memory und Scope-Guard funktionieren in den geprüften Situationen überwiegend zuverlässig. Gleichzeitig zeigen T_STATE_01 und T_SOLVED_01 eine Grenze des Ansatzes: Ein State-Flag im Prompt führt nicht automatisch zu einer inhaltlich starken oder dramaturgisch befriedigenden Antwort. Diese Beobachtung ist zentral für Kapitel 8, da sie zeigt, dass kontrollierte KI-NPCs neben Wissensschutz auch gezielte Antwortsteuerung für freigeschaltete Informationen benötigen.

8. Diskussion und Limitationen

Kapitel 8 ordnet die Evaluationsergebnisse im Hinblick auf die Forschungsfrage ein. Im Fokus steht, welche Aussagen sich aus den getesteten Dialogsituationen über kontrollierte KI-gestützte NPC-Dialoge ableiten lassen.

Die Diskussion trennt zwischen Wirksamkeit und Grenzen der Kontrollmechanismen. Rollenbindung, Wissensbegrenzung, Scope-Guard und Memory funktionierten überwiegend, während die aktive Nutzung freigeschalteter State-Informationen begrenzt blieb.

Die Einordnung erfolgt vor dem Hintergrund, dass LLMs in Spielen zwar neue Formen dynamischer Dialoge ermöglichen, aber weiterhin mit Risiken wie Halluzinationen, Rollenbruch, schwacher State-Bindung und unzureichender Weltmodellierung verbunden sind (Callison-Burch, et al., 2022; Ji, et al., 2023; Sweetser, 2024; Tsai, et al., 2023).

8.1. Einordnung der Evaluationsergebnisse

Die Evaluation zeigt, dass die Architektur zentrale Kontrollziele unterstützt. Die NPCs blieben überwiegend charakterkonsistent, nutzten kein offensichtlich unzulässiges Wissen, vermieden Meta-Brüche und wehrten themenfremde Eingaben fallbezogen ab.

Die Ergebnisse sprechen dafür, dass strukturierte NPC-Profile, erlaubtes Wissen, abstrakte Sperrmarkierungen, Constraints und Scope-Guard in einem kleinen kontrollierten Szenario wirksam sein können. Es traten in der Evaluation keine kritischen Halluzinationen, keine unzulässige Täteroffenlegung, keine klaren Rollenbrüche und keine Meta-Brüche auf. Damit erfüllte der Prototyp die wichtigsten Schutzanforderungen des Qualitätsrahmens aus Kapitel 3.

Die Schwächen lagen vor allem in der dramaturgischen Ausschöpfung freigegebener Informationen. T_STATE_01 und T_SOLVED_01 zeigen, dass State-Flags berücksichtigt werden können, ohne automatisch eine starke narrative Antwort zu erzwingen.

Damit wird die Forschungsfrage grundsätzlich positiv beantwortet, aber begrenzt: Promptbasierte Kontrolle reduziert Risiken und lenkt Kontext, erzeugt jedoch keine deterministische Dialoglogik und kein vollständiges Weltmodell.

Evaluationsergebnis	Bedeutung für die Forschungsfrage	Einordnung
Keine kritischen Halluzinationen, Spoiler oder Meta-Brüche	Die Kontrollmechanismen begrenzen zentrale Fehlertypen des Prototyps.	Starkes Ergebnis für Wissensbegrenzung, Scope-Guard und Constraints.
Durchgängige Rollenbindung der drei NPCs	NPC-Profile unterstützen charakterkonsistente Antworten.	Starker Hinweis, dass Rollenprofile im kleinen Szenario wirksam sind.
Memory-Test erfolgreich	NPC-bezogenes Kurzzeit-Memory kann einfache Anschlussfähigkeit herstellen.	Aussagekräftig für kurze Sequenzen, aber nicht für Langzeit-Memory.
State-Nutzung nur teilweise überzeugend	State-Flags als Prompt-Kontext reichen nicht für aktive Freigabelogik.	Zentrale Grenze des Ansatzes; besonders relevant für Kapitel 9.
Keine Nutzerstudie und kein Modellvergleich	Ergebnisse zeigen Machbarkeit, aber keine allgemeine Wirksamkeit.	Methodische Begrenzung des prototypischen Artefakts.

Tabelle 21 Zentrale Evaluationsergebnisse und ihre Bedeutung für die Forschungsfrage. Quelle: Vom Verfasser erstellte Tabelle

8.2. Wirksamkeit der Kontrollmechanismen

Die stärksten Ergebnisse zeigten sich bei denjenigen Kriterien, die direkt durch stabile Kontextbestandteile unterstützt wurden. NPC-Profile mit Rolle, Motivation, Sprachstil und erlaubtem Wissen wirkten sich positiv auf die Charakterkonsistenz aus. Clara, Anton und Mira reagierten in den Testfällen unterscheidbar und blieben in ihrer jeweiligen Funktion erkennbar. Damit bestätigt der Prototyp die Annahme aus Kapitel 5, dass ein LLM nicht als freier Gesprächspartner eingesetzt werden sollte, sondern als sprachliches Generierungsmodul innerhalb eines klar definierten Rollen- und Wissensrahmens.

Auch die Wissensbegrenzung funktionierte in den kritischen Testfällen überwiegend zuverlässig. Clara bestätigte vor der Fallauflösung ihre Täterschaft nicht, und Anton nannte keine sichere Täteridentität. Die Kombination aus erlaubtem Wissen, abstrakten Sperrmarkierungen und Constraints war daher geeignet, offensichtliche Spoiler in diesem kleinen kontrollierten Szenario zu reduzieren. Dieser Befund passt zu Liu, Xie und Jiang, die character hallucination als Problem beschreiben, bei dem Modelle Wissen oder Verhalten erzeugen, das nicht zum Charakter oder dessen Wissensbereich passt (Liu, et al., 2025)

Der Scope-Guard erwies sich ebenfalls als wirksam. Die League-of-Legends-Frage, die medizinische Frage und die Prompt-Frage wurden nicht inhaltlich als allgemeine Chatbot-Anfragen beantwortet. Stattdessen blieben die Antworten kurz, rollennah und führten auf den Fallkontext zurück. Für den Prototyp ist dies ein wichtiges Ergebnis, weil Off-Topic-Eingaben sonst schnell die Illusion einer begrenzten Spielfigur beschädigen könnten.

Trotzdem dürfen diese Mechanismen nicht als harte Sicherheitsschicht verstanden werden. Die finale Antwort wird weiterhin von einem generativen Sprachmodell erzeugt. Prompt-Constraints reduzieren die Wahrscheinlichkeit problematischer Antworten, schließen Halluzinationen oder Regelverletzungen aber nicht vollständig aus. Ji et al. beschreiben Halluzinationen als generierte Inhalte, die nicht durch den gegebenen Kontext gedeckt sind; dieses Grundproblem bleibt auch dann relevant, wenn die Eingabe durch strukturierte Kontextbestandteile eingegrenzt wird (Ji, et al., 2023).

8.3. Grenzen der State-Steuerung

Die wichtigste Schwäche lag in der State-Steuerung. State-Flags wurden als Prompt-Kontext übergeben, nicht als harte Freischaltlogik; das Modell musste ihre Bedeutung selbst gewichten.

T_STATE_01 zeigte, dass hasAnalyzedWine nur indirekt genutzt wurde. In T_SOLVED_01 erkannte das System caseSolved, blieb aber von Schutz- und Ausweichregeln geprägt. Das System verhinderte zu frühe Informationsgabe besser, als es freigegebene Informationen aktiv ausspielte.

Diese Beobachtung passt zu bestehenden Arbeiten über Spiel- und Rollenspielkontexte. Callison-Burch et al. zeigen, dass Game State Tracking in dialogbasierten Rollenspielen schwierig bleibt, weil Dialogsysteme nicht nur die nächste Äußerung erzeugen, sondern auch den aktuellen Welt- und Spielzustand berücksichtigen müssen (Callison-Burch, et al., 2022). Tsai et al. zeigen im Kontext von Textgames ebenfalls, dass LLMs zwar kom-

munikationsfähig sind, aber Schwierigkeiten haben können, ein stabiles Weltmodell aufzubauen, Ziele aus dem Spielverlauf abzuleiten und Weltwissen passend zu nutzen (Tsai, et al., 2023).

Zukünftig sollte Sperrlogik stärker von Freigabelogik getrennt werden. Erstere verhindert zu frühe Informationen; letztere müsste festlegen, welche Informationen nach bestimmten State-Änderungen aktiv genannt oder dramaturgisch gewichtet werden.

Für Krimi-Dialoge ist diese Grenze zentral: Ein Detektivspiel muss Spoiler verhindern, nach Hinweisen oder Falllösung aber auch kontrolliert neue Informationen freigeben.

8.4. Memory und Scope-Guard im Prototyp

Das NPC-bezogene Memory zeigte im kontrollierten Testfall ein positives Ergebnis. Clara konnte die vorherige Frage zur Hintertür korrekt wiederaufgreifen, ohne neue Details zu erfinden. Für den Prototyp spricht dies dafür, dass ein kleines, NPC-bezogenes Kurzzeit-Memory ausreichen kann, um einfache Anschlussfähigkeit innerhalb eines Dialogs herzustellen.

Gleichzeitig bleibt die Aussagekraft dieses Ergebnisses begrenzt. Das Memory wurde nur in einer kurzen Sequenz mit definierter Vorinteraktion getestet. Es bildet kein vollständiges Langzeitgedächtnis, keine komplexe Beziehungsentwicklung und keine semantische Wissensbasis ab. Liu, Xie und Jiang unterscheiden zwischen statischem Charakterwissen und dynamischem Memory; für größere Spielsysteme wäre diese Trennung deutlich aufwendiger umzusetzen als im hier entwickelten Prototyp (Liu, et al., 2025)

Besonders wichtig ist, dass Memory nicht automatisch jede Spielereingabe als Wahrheit übernehmen darf. Eine falsche oder suggestive Spielerbehauptung könnte sonst in späteren Antworten stabilisiert werden. Die im Prototyp definierte Speicherregel, nur fallrelevante und überprüfbare Informationen zu übernehmen, ist daher nicht nur eine technische Vereinfachung, sondern ein Schutz gegen falsches Memory.

Der Scope-Guard adressiert ein anderes Risiko: NPCs sollen nicht als allgemeine Assistenten funktionieren. In den getesteten Off-Topic- und Meta-Fällen wurde diese Begrenzung eingehalten. Mira gab keine League-of-Legends-Tipps, Clara gab keine medizinische Beratung und Anton legte keine Prompt- oder Systemregeln offen. Dies zeigt, dass ein Scope-Guard im Prompt den thematischen Geltungsbereich eines NPCs sinnvoll eingrenzen kann.

Auch hier gilt jedoch, dass der Mechanismus promptbasiert und damit nicht deterministisch ist. Der Scope-Guard reduziert unerwünschtes Verhalten, ersetzt aber keine programmatische Klassifikation oder harte Eingabefilterung. Für produktionsnahe Systeme wäre eine zusätzliche Vorverarbeitung denkbar, die Off-Topic-Fragen erkennt und in einen eigenen Antwortpfad leitet, bevor das LLM die eigentliche NPC-Antwort erzeugt.

8.5. Methodische Grenzen der Evaluation

Methodisch ist die Evaluation begrenzt: Jeder Testfall wurde einmal im API-Modus durchgeführt; T_STATE_01 enthielt zwei Varianten, T_MEMORY_01 eine definierte Vorinteraktion. Die Ergebnisse sind qualitative Momentaufnahme statt Stabilitätsnachweis.

Eine weitere Grenze liegt in der Bewertung durch den Autor der Arbeit. Zwar wurden Kriterien, Rubrik und Fehlercodes vorab definiert, dennoch bleibt die Bewertung einzelner Antworten teilweise interpretativ. Eine zusätzliche Bewertung durch weitere Personen hätte die Intersubjektivität erhöhen können. Eine Nutzerstudie hätte zudem prüfen können, ob Spielende die NPCs tatsächlich als glaubwürdig, spannend oder natürlich wahrnehmen. Diese Aspekte lagen jedoch außerhalb des methodischen Umfangs der Arbeit.

Die gewählte Methodik ist dennoch passend zur Fragestellung. Ziel war nicht, die subjektive Spielerfahrung eines fertigen Spiels zu messen, sondern zu prüfen, ob ein prototypisches NPC-Dialogsystem anhand definierter Qualitätskriterien kontrollierbare Antworten erzeugt. Die kriteriengeleitete Testfallevaluation macht sichtbar, ob Antworten

gegen Rollenprofil, Wissensgrenze, State oder Scope verstoßen. Gerade solche Fehler wären in einer kleinen Nutzerstudie nicht zwingend zuverlässig erkannt worden.

Auch die Entscheidung gegen automatische Dialogmetriken ist methodisch begründet. Liu, Lowe et al. zeigen, dass gängige automatische Metriken wie BLEU, METEOR oder ROUGE bei offenen Dialogantworten nur schwach oder gar nicht mit menschlichen Qualitätsurteilen korrelieren können (Liu, et al., 2016). Für den Krimi-Prototyp wäre eine Wortüberlappungsmetrik besonders ungeeignet, weil nicht eine bestimmte Musterantwort entscheidend ist, sondern die Einhaltung von Rollenwissen, State, Nichtwissen, Scope und dramaturgischer Funktion.

Die Aussagekraft entsteht durch Testfälle, dokumentierte State- und Memory-Zustände, Logs, Promptversionierung und einheitliche Rubrik; sie ersetzt keine umfassende Validierung.

8.6. Technische Grenzen des Prototyps

Auch technisch ist der Prototyp bewusst begrenzt. Das System verwendet keine vollständige Retrieval-Augmented-Generation-Pipeline, keinen komplexen Knowledge Graph und keine semantische Suche. Das erlaubte Wissen der NPCs wird in vereinfachter Form bereitgestellt und nicht dynamisch über eine Datenbank abgerufen. Dadurch bleibt die Architektur überschaubar und evaluierbar, bildet aber nur einen Ausschnitt möglicher produktiver NPC-Systeme ab.

RAG-Ansätze zeigen, dass die Verbindung eines generativen Modells mit extern abrufbarem Wissen besonders für wissensintensive Aufgaben relevant sein kann. Lewis et al. verbinden parametric memory mit non-parametric memory, um externe Wissensquellen in die Generierung einzubeziehen (Lewis, et al., 2021). Ashby et al. zeigen im Spielkontext, wie strukturierte Weltinformationen in Form eines Knowledge Graphs genutzt werden können, um Quest- und Dialoggenerierung an Spielweltobjekte und Relationen zu binden (Ashby, et al., 2023). Der eigene Prototyp übernimmt dieses Prinzip nur reduziert: Die Wissensmatrix und State-Flags ersetzen keine echte Retrieval- oder Graph-Architektur.

Generierte Antworten werden nachträglich nicht automatisch validiert. Die Kontrolle erfolgt primär vor der Generierung durch Kontextauswahl, Constraints und Scope-Guard; robustere Systeme könnten regelbasierte Prüfungen, Klassifikation oder ein zweites Prüfmodell ergänzen.

Auch die API-Anbindung ist prototypisch. Der API-Key wird zwar nicht im Repository gespeichert, und der Dummy-Modus bleibt als Entwicklungs- und Fallback-Variante erhalten. Für ein veröffentlichtes Spiel wäre jedoch eine serverseitige Architektur notwendig, damit API-Zugänge, Kostenkontrolle, Missbrauchsschutz, Logging und Datenschutz produktionsgerecht umgesetzt werden können. Ein lokales LLM wurde in dieser Arbeit nicht systematisch verglichen, sondern nur als mögliche Alternative für zukünftige Arbeiten eingeordnet.

Grenze	Nicht umgesetzt	Konsequenz für die Aussagekraft
Keine Nutzerstudie	keine Messung von wahrgenommener Immersion oder Spielspaß	Evaluation prüft Systemkriterien, nicht subjektive Spielerfahrung.
Ein Durchlauf pro Testfall	keine statistische Stabilitätsprüfung über viele Wiederholungen	Ergebnisse sind qualitative Momentaufnahme.
Kein Modellvergleich	keine Gegenüberstellung mehrerer LLMs oder Parameter	Aussagen beziehen sich auf den dokumentierten Modell- und Promptstand.
Kein vollständiges RAG	keine Vektorsuche oder dynamische Wissensdatenbank	Kontextauswahl ist regelbasiert und prototypisch.
Kein komplexes Weltmodell	kein vollständiger Knowledge Graph und keine harte Freigabelogik	State wirkt primär als Prompt-Kontext.
Keine automatische Output-Validierung	keine Nachprüfung erzeugter Antworten durch Regeln oder Zweitmodell	Fehlererkennung erfolgt manuell in Kapitel 7.
Kein vollständiges Langzeit-Memory	keine langfristige Entwicklung über viele Spielsitzungen	Memory-Aussagen gelten nur für kurze, definierte Dialogsequenzen.

Tabelle 22 Zentrale Limitationen des Prototyps und der Evaluation. Quelle: Vom Verfasser erstellte Tabelle

8.7. Bedeutung für KI-gestützte NPC-Dialoge

Aus den Ergebnissen lässt sich ableiten, dass KI-gestützte NPC-Dialoge nicht als reine Chatbot-Anbindung verstanden werden sollten. Die Qualität der Antworten entstand im Prototyp nicht allein durch das verwendete Sprachmodell, sondern durch die Kombination aus NPC-Profilen, Wissensgrenzen, State-Informationen, Memory, Scope-Guard und Constraints. Das Sprachmodell übernimmt dabei vor allem die sprachliche Ausformulierung innerhalb eines vorgegebenen Rahmens.

Der Prototyp war besonders stark bei klaren Grenzen: keine Meta-Antworten, keine Off-Topic-Beratung, keine Täterinformation vor Auflösung. Schwächer war die aktive Nutzung freigeschalteter Informationen. Kontrollierte KI-NPCs brauchen daher Verbote und positive Steuerung.

Für zukünftige KI-NPC-Systeme ergibt sich daraus ein gestufter Ansatz. Zunächst müssen Rollenprofile, Wissensgrenzen und Scope klar definiert werden, damit NPCs nicht als allgemeine Assistenten auftreten. Anschließend muss State nicht nur als Textkontext, sondern als echte Freigabe- und Gewichtungslogik wirken. Für größere Systeme wären darüber hinaus Retrieval, Knowledge Graphs, Output-Validierung, Langzeit-Memory und Nutzerstudien notwendig, um über den hier erreichten Machbarkeitsnachweis hinauszugehen.

8.8. Zwischenfazit

Die Diskussion beantwortet die Forschungsfrage grundsätzlich positiv, aber begrenzt. Der Prototyp macht NPC-Antworten im kleinen Krimi-Szenario kontrollierbarer und vermeidet in den Tests kritische Halluzinationen, Spoiler, Rollen- und Meta-Brüche. Gleichzeitig bleibt State-Nutzung eine Grenze, weil promptbasierte Kontrolle freigeschaltete Informationen nicht automatisch stark und dramaturgisch passend ausspielt.

9. Fazit und Ausblick

Diese Arbeit untersuchte, wie KI-gestützte NPC-Dialoge in einem Unity-Prototyp charakterkonsistent, wissensbegrenzt und state-bezogen gestaltet werden können. Ausgangspunkt war das Risiko, dass LLMs ohne Begrenzung halluzinieren, Rollen verlassen, unzulässiges Wissen preisgeben oder wie allgemeine Chatbots antworten.

Die Forschungsfrage kann grundsätzlich positiv beantwortet werden. Der Prototyp zeigt, dass KI-NPCs kontrollierbarer werden, wenn das Sprachmodell in eine strukturierte Architektur aus NPC-Profilen, begrenztem Wissen, State, Memory, Constraints, Scope-Guard und Logging eingebettet wird.

Die wichtigste Erkenntnis ist, dass glaubwürdige KI-NPCs nicht allein durch eine API-Anbindung entstehen. Entscheidend ist, welche Informationen bereitgestellt, begrenzt und durch Antwortregeln gerahmt werden. Im Krimi-Kontext reduziert dies Spoiler, Meta-Brüche und themenfremde Chatbot-Antworten.

Die Evaluation zeigt zugleich die Grenze des Ansatzes: State-Flags wirken als Prompt-Kontext, aber nicht als harte Freigabelogik. Der Prototyp begrenzte unerwünschte Informationen stärker, als er freigegebene Informationen aktiv dramaturgisch ausspielte.

Der Beitrag der Arbeit liegt damit in einem prototypischen, nachvollziehbar evaluierten Ansatz für kontrollierte KI-gestützte NPC-Dialoge. Der entwickelte Qualitätsrahmen macht Anforderungen wie Charakterkonsistenz, Wissensbegrenzung, State-Konsistenz, Memory-Korrektheit, Umgang mit Nichtwissen und spielerische Sinnhaftigkeit bewertbar. Das Systemkonzept und die Implementierung zeigen, wie diese Anforderungen in einem kleinen Unity-Prototyp technisch umgesetzt werden können. Die Evaluation liefert dazu einen begrenzten, aber nachvollziehbaren Machbarkeitsnachweis.

Die Aussagekraft der Arbeit bleibt durch den prototypischen Umfang begrenzt. Der Prototyp bildet kein vollständiges Krimi-Spiel ab, sondern eine reduzierte Testumgebung für KI-gestützte NPC-Dialoge. Auch die Evaluation ist als qualitative Momentaufnahme zu

verstehen, da jeder Testfall einmal im API-Modus durchgeführt wurde und keine Nutzerstudie stattfand. Die Ergebnisse zeigen daher keine allgemeine Lösung für alle KI-NPC-Systeme, sondern einen überprüfbaren Machbarkeitsnachweis innerhalb eines klar begrenzten Szenarios.

Für zukünftige Arbeiten ergeben sich mehrere Ansatzpunkte. Eine weiterentwickelte Version des Prototyps sollte State nicht nur als Prompt-Kontext verwenden, sondern als stärkere Freigabe- und Gewichtungslogik umsetzen. Zusätzlich wäre eine vollständigere Retrieval-Architektur, ein Knowledge Graph, automatische Output-Validierung und ein langfristigeres Memory sinnvoll. Auch eine Nutzerstudie wäre ein wichtiger nächster Schritt, um nicht nur die Systemkriterien, sondern die tatsächliche Wahrnehmung der Spielenden zu untersuchen.

Insgesamt zeigt die Arbeit, dass KI-gestützte NPC-Dialoge in narrativen Spielen Potenzial besitzen, aber nur mit klaren Rollen, Wissensgrenzen, State-Bezug und Evaluation sinnvoll kontrollierbar werden. Der Prototyp ist damit ein Machbarkeitsnachweis, kein Produktionssystem.

Literaturverzeichnis

Ashby, Trevor, Webb, Braden K, Knapp, Gregory, Searle, Jackson und Fulda, Nancy. 2023. Personalized Quest and Dialogue Generation in Role-Playing Games: A Knowledge Graph- and Language Model-based Approach. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems., CHI '23. New York, NY, USA : Association for Computing Machinery, 2023. — ISBN 978-1-4503-9421-5., S. 1–20.

Callison-Burch, Chris, Tomar, Gaurav Singh, Martin, Lara J., Ippolito, Daphne, Bailis, Suma und Reitter, David. 2022. Dungeons and Dragons as a Dialog Challenge for Artificial Intelligence. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022. — arXiv:2210.07109 [cs]., S. 9379–9393.

Hevner, Alan R., March, Salvatore T., Park, Jinsoo und Ram, Sudha. 2004. Design Science in Information Systems Research. Management Information Systems Quarterly. 2004, Bd. 28, S. 75.

Ji, Ziwei, Lee, Nayeon, Frieske, Rita, Yu, Tiezheng, Su, Dan, Xu, Yan, Ishii, Etsuko, Bang, Yejin, Chen, Delong, Dai, Wenliang, Chan, Ho Shu, Madotto, Andrea und Fung, Pascale. 2023. Survey of Hallucination in Natural Language Generation. ACM Computing Surveys. 2023, Bd. 55, 12, S. 1–38. — arXiv:2202.03629 [cs].

Lewis, Patrick, Perez, Ethan, Piktus, Aleksandra, Petroni, Fabio, Karpukhin, Vladimir, Goyal, Naman, Küttler, Heinrich, Lewis, Mike, Yih, Wen-tau, Rocktäschel, Tim, Riedel, Sebastian und Kiela, Douwe. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2021. — arXiv:2005.11401 [cs].

Liu, Chia-Wei, Lowe, Ryan, Serban, Iulian, Noseworthy, Mike, Charlin, Laurent und Pineau, Joelle. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In: Su, J., Duh, K. und Carreras, X. (Hrsg.): Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas : Association for Computational Linguistics, 2016., S. 2122–2132.

Liu, Xiao, Xie, Zhenping und Jiang, Senlin. 2025. Personalized Non-Player Characters: A Framework for Character-Consistent Dialogue Generation. AI. 2025, Bd. 6, 5, S. 93.

Loyall, A. B. 1997. Believable agents: building interactive personalities. CMU, Pittsburgh, 1997.

Marconi, Luca, Longo, Luca und Cabitza, Federico. 2026. Assessing Interaction Quality in Human–AI Dialogue: An Integrative Review and Multi-Layer Framework for Conversational Agents. Machine Learning and Knowledge Extraction. 2026, Bd. 8, 2, S. 28.

Microsoft. o. J.-b. Encoding.UTF8 Property (System.Text). [Online] o. J.-b. [Zitat vom: 04. Juni 2026.] https://learn.microsoft.com/en-us/dotnet/api/system.text.encoding.utf8?view=net-10.0.

—. o. J.-a. File.AppendAllText Method (System.IO). [Online] o. J.-a. [Zitat vom: 04. Juni 2026.] https://learn.microsoft.com/en-us/dotnet/api/system.io.file.appendalltext?view=net-10.0.

OpenAI. o. J.-c. Best Practices for API Key Safety. OpenAI Help Center. [Online] o. J.-c. [Zitat vom: 04. Juni 2026.] https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety.

—. o. J.-a. Create a model response. OpenAI API Reference. [Online] o. J.-a. [Zitat vom: 04. Juni 2026.] https://developers.openai.com/api/reference/resources/responses/methods/create/.

—. o. J.-b. GPT-5.4 mini Model | OpenAI API. [Online] o. J.-b. [Zitat vom: 04. Juni 2026.] https://developers.openai.com/api/docs/models/gpt-5.4-mini.

Sweetser, Penny. 2024. Large Language Models and Video Games: A Preliminary Scoping Review. In: Proceedings of the 6th ACM Conference on Conversational User Interfaces., CUI '24. New York, NY, USA : Association for Computing Machinery, 2024. — ISBN 979-8-4007-0511-3., S. 1–8.

Tsai, Chen Feng, Zhou, Xiaochen, Liu, Sierra S., Li, Jing, Yu, Mo und Mei, Hongyuan. 2023. Can Large Language Models Play Text Games Well? Current State-of-the-Art and Open Questions. 2023. — arXiv:2304.02868 [cs].

Unity Technologies, Unity. o. J.-b. Unity - Manual: uGUI. [Online] o. J.-b. [Zitat vom: 04. Mai 2026.] https://docs.unity3d.com/6000.3/Documentation/Manual/com.unity.ugui.html.

—. o. J.-d. Unity - Scripting API: Application.persistentDataPath. [Online] o. J.-d. [Zitat vom: 04. Mai 2026.] https://docs.unity3d.com/6000.4/Documentation/ScriptReference/Application-persistentDataPath.html.

—. o. J.-a. Unity - Scripting API: MonoBehaviour. [Online] o. J.-a. [Zitat vom: 04. Mai 2026.] https://docs.unity3d.com/6000.3/Documentation/ScriptReference/MonoBehaviour.html.

—. o. J.-c. Unity - Scripting API: UnityWebRequest. [Online] o. J.-c. [Zitat vom: 04. Mai 2026.] https://docs.unity3d.com/6000.3/Documentation/ScriptReference/Networking.UnityWebRequest.html.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz und Polosukhin, Illia. 2017. Attention Is All You Need. 2017. — arXiv:1706.03762 [cs].

Warpefelt, Henrik und Verhagen, Harko. 2017. A model of non-player character believability. Journal of Gaming & Virtual Worlds. 2017, Bd. 9, 1, S. 39–53.

Yi, Sanghyun, Goel, Rahul, Khatri, Chandra, Cervone, Alessandra, Chung, Tagyoung, Hedayatnia, Behnam, Venkatesh, Anu, Gabriel, Raefer und Hakkani-Tur, Dilek. 2019. Towards Coherent and Engaging Spoken Dialog Response Generation Using Automatic Conversation Evaluators. In: Deemter, K. van, Lin, C. und Takamura, H. (Hrsg.): Proceedings of the 12th International Conference on Natural Language Generation. Tokyo, Japan : Association for Computational Linguistics, 2019., S. 65–75.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A: Vollständige Einzelprotokolle der Evaluation Die vollständigen Einzelprotokolle befinden sich im digitalen Anhang: Nachweise_Kapitel_7/BA_Kapitel_7_Evaluation_Anhang_Einzelprotokolle.pdf