

EINSATZ VON LLM TOOLS IM SOFTWARE ENGINEERING PROZESS

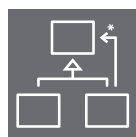
BACHELORARBEIT
ZUR ERLANGUNG DES AKADEMISCHEN GRADES
BACHELOR OF ENGINEERING (B.ENG.)

Jonas Gundlach

Betreuerin:
Prof. Dr. rer. nat. Andrea Baumann

Tag der Abgabe: XX.XX.XXXX

eingereicht bei
Universität der Bundeswehr München
Fakultät für Elektrotechnik und Technische Informatik



Universität der Bundeswehr München

Institut für
Software Engineering

Neubiberg, Juli 2024

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, insbesondere keine anderen als die angegebenen Informationen.

Der Speicherung meiner Bachelorarbeit zum Zweck der Plagiatsprüfung stimme ich zu. Ich versichere, dass die elektronische Version mit der gedruckten Version inhaltlich übereinstimmt.

Neubiberg, den XX.XX.XXXX

Jonas Gundlach

Zusammenfassung

Im Rahmen dieser Arbeit wird untersucht, inwieweit die drei LLM-Tools ChatGPT, Google Gemini und Mistral Le Chat im Bereich des Software Engineerings eingesetzt werden können. Dabei wird betrachtet, wie gut die drei Tools in der Lage sind, die in einem Software-Engineering-Prozess verwendeten Dokumente selbstständig zu erstellen. Die Dokumente werden am Beispiel eines „Mensch ärgere dich nicht“-Spiels erstellt.

Nach einer kurzen Beschreibung der drei LLM-Tools und einem Überblick über die Inhalte der verschiedenen Dokumente wird untersucht, inwieweit die Tools im Rahmen eines Software-Engineering-Prozesses unterstützen können. Da es zahlreiche Unterstützungsmöglichkeiten gibt, wird sich in dieser Arbeit lediglich auf die Dokumentenerstellung fokussiert.

In der Analyse werden die Ergebnisse der Tools, die auf dem beiliegenden USB-Stick zu finden sind, besprochen und auf ihre Fehler untersucht. Dabei stellen insbesondere das Format der UML-Diagramme und die Konsistenz zwischen den einzelnen Dokumenten eine Herausforderung dar. Anschließend werden auch die Funktionen für die Schnittstelle mit ChatGPT erstellt, da dieses Tool die Dokumente am erfolgreichsten erstellt hat.

Abschließend gibt es eine Zusammenfassung der Ergebnisse und eine Beschreibung, wie gut mit den einzelnen Tools gearbeitet werden konnte. Danach folgt ein Abschnitt zur Kritik an dieser Arbeit und der Arbeitsweise. Besonders aufgrund der vielen unterschiedlichen Aspekte, die bei der Erstellung der Dokumente berücksichtigt werden müssen, und der unterschiedlichen Ausgaben der drei Tools war es schwierig, den Überblick zu behalten. Zum Schluss folgt ein kurzer Ausblick auf offene Punkte, die in diesem Bereich noch untersucht werden können. Da die drei Tools noch relativ neu sind und die Entwicklung in den letzten Jahren erst richtig Fahrt aufgenommen hat, ist zu erwarten, dass sich in diesem Bereich noch viel tun wird.

Inhaltsverzeichnis

Erklärung	III
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung der Arbeit	2
1.3 Struktur der Arbeit	2
2 Grundlagen	5
2.1 LLM Tools	5
2.1.1 ChatGPT	6
2.1.2 Google Gemini	7
2.1.3 Le Chat	10
2.2 Software Engineering Prozess	12
2.2.1 Besprechungsprotokoll	13
2.2.2 Projekthandbuch	13
2.2.3 Risikoliste	14
2.2.4 Anforderungsspezifikation	15
2.2.5 Architekturdokument	17
2.2.6 Testspezifikation	18
3 Anwendung von LLM Tools im Software Engineering	19
3.1 Besprechungsprotokoll	19
3.2 Projekthandbuch	20
3.3 Risikoliste	20
3.4 Anforderungsspezifikation	21
3.5 Architekturdokument	22
3.6 Testspezifikation	22
4 Praxisergebnisse und Vergleich	25
4.1 Besprechungsprotokoll	26
4.2 Projekthandbuch	28
4.3 Risikoliste	29
4.4 Anforderungsspezifikation	32
4.5 Architekturdokument	49

5 Zusammenfassung und Ausblick	59
5.1 Zusammenfassung der Ergebnisse	59
5.2 Kritik an der Arbeit	61
5.3 Ausblick und zukünftige Entwicklungen	61
6 Anhang	I
Stichwortverzeichnis	XIII
Literaturverzeichnis	XV

1 Einleitung

Die rasante Entwicklung im Bereich der Künstlichen Intelligenz hat in den letzten Jahren bedeutende Fortschritte ermöglicht, insbesondere durch den Einsatz von Large Language Models (LLM). Diese mächtigen Tools, die auf umfangreichen Textkorpora trainiert wurden, haben das Potenzial, vielfältige Aufgaben im Software Engineering Prozess zu optimieren. In dieser Bachelorarbeit wird untersucht, wie LLM-Tools im Software Engineering eingesetzt werden können, um die Effizienz und Qualität der Entwicklungsprozesse zu verbessern.

In diesem ersten Kapitel wird zunächst auf die Motivation hinter dieser Arbeit eingegangen. Anschließend wird die Zielsetzung und dann die Struktur der Arbeit beschrieben.

1.1 Motivation

KIs ersetzen irgendwann die Menschen! Dies ist eine Aussage, die man mittlerweile recht häufig hört. Unter anderem durch Filme und Serien ist das Ansehen von KIs nicht immer positiv, und einige Menschen sehen eine große Gefahr darin. Besonders durch die rasante Entwicklung in den letzten zwei Jahren haben sogenannte Large Language Models (LLMs) viel Aufmerksamkeit auf sich gezogen und die Möglichkeiten, zum Beispiel in der Informationsbeschaffung, stark vergrößert. Gerade in Schulen und Universitäten haben solche Tools mittlerweile eine große Relevanz erlangt, was nicht von jedem positiv gesehen wird. Viele Pädagogen behaupten, dass LLM-Tools schlecht für die Entwicklung von Schülern und Studenten sind, da man nun Hausaufgaben oder ähnliche Arbeiten einfach an LLM-Tools abgeben und in wenigen Sekunden automatisch erstellen lassen kann. Doch wie gut sind eigentlich die Ausgaben von LLM Tools?

In einer vorherigen Projektarbeit zum Thema "Erzeugung von Algorithmen mit LLM-Tools für Java" wurde bereits untersucht wie gut drei unterschiedliche Tools einfache Algorithmen zum Sortieren von Listen und einen etwas komplexeren Algorithmus für Spiele wie Tic-Tac-Toe oder Vier Gewinnt erstellen können. Dies hat, zumindest bei den einfacheren Algorithmen, bereits sehr gut funktioniert, solange man bei funktionierendem Code nicht immer weiter nachfragt und etwas verbessern lassen will. Angestoßen durch ein begleitendes Modul zum Thema 'Secure Software Engineering' entstand das Interesse an der Frage, inwieweit LLMs auch bei der Erstellung von Software-Dokumentation unterstützen können. Insbesondere stellt sich dabei die Frage, ob LLMs in der Lage sind, eine vollständige und aussagekräftige Dokumentation autonom zu generieren oder ob sie eher als Hilfsmittel zur Unterstützung menschlicher Autoren dienen können.

1.2 Zielsetzung der Arbeit

Das Ziel der Arbeit ist es, herauszufinden, wie gut LLM-Tools geeignet sind, Dokumente aus dem Software-Engineering-Prozess zu erstellen oder bei der Erstellung zu unterstützen. Der Fokus liegt dabei vor allem auf der Erstellung der Kernabschnitte der einzelnen Dokumente, wie zum Beispiel dem fachliche Datenmodell und den einzelnen UML-Diagrammen. Des Weiteren sollen Studenten einen Eindruck erhalten, wie sie die Eingaben formulieren müssen und worauf es dabei ankommt. Zusätzlich soll eine Tendenz abgegeben werden, welche Tools für die Erstellung besonders gut geeignet sind und welche Tools weniger geeignet erscheinen.

1.3 Struktur der Arbeit

Grundsätzlich befasst sich die Arbeit mit drei ausgewählten LLM-Tools: ChatGPT und Google Gemini als vermutlich die bekanntesten Tools sowie Le Chat von Mistral AI als kleine europäische Alternative. Mit diesen Tools wird versucht, die Dokumente für ein "Mensch ärgere dich nicht"-Spiel selbstständig zu erstellen. Dabei wird ausschließlich mit den Ausgaben der Tools gearbeitet, ohne sie zu verändern oder zu verbessern. Stattdessen wird mit Hilfe von Eingaben versucht, die Tools zu korrigieren. Die zu erstellenden Dokumente und die Anforderungen an das Projekt orientieren sich an den Vorgaben des Moduls "Secure Software Engineering" an der Universität der Bundeswehr München, in der Fakultät "ETTI6" am Institut für "Software Engineering". Die verantwortliche Professorin ist Prof. Dr. rer. nat. Andrea Baumann, die mit ihren Kontaktdaten im Literaturverzeichnis aufgeführt ist.

Die Arbeit ist wie folgt strukturiert: Zunächst wird eine Einführung in die Historie der drei betrachteten Tools gegeben. Anschließend erfolgt eine detaillierte Beschreibung der in einem Software-Engineering-Prozess typischerweise erstellten Dokumente, einschließlich ihrer Inhalte und ihres Zwecks. Darauf aufbauend wird untersucht, wie die eingesetzten LLMs bei der Erstellung dieser Dokumente unterstützen können. Dieses Kapitel formuliert zugleich die zentrale Fragestellung der Analyse.

Im folgenden Kapitel werden die Ergebnisse der praktischen Anwendung der LLMs präsentiert und vergleichend betrachtet. Dabei wird insbesondere auf die Qualität der generierten Inhalte sowie auf mögliche Fehler und Limitationen eingegangen.

Die Arbeit schließt mit einer Zusammenfassung der wichtigsten Erkenntnisse ab. Zudem werden kritisch die Schwächen der durchgeführten Analyse und der Arbeitsweise reflektiert sowie, in einem Ausblick, mögliche Ansatzpunkte für zukünftige Forschungsarbeiten aufgezeigt.

Die erstellten Ausgaben, welche im Kapitel 4 analysiert werden, befinden sich auf dem beigelegten USB-Stick in Form von PDF-Dokumenten [Abschnitt 6 im Anhang]. In der Überschrift der Dokumente steht jeweils, um was es sich dabei handelt und dann in runden Klammern das Tool, welches für die Erstellung verwendet wurde, und eine Zahl, die wievielte Eingabe dies zu diesem Thema war. In den einzelnen Abschnitten wird als erstes der Prompt niedergeschrieben, der für die Erstellung der Ausgabe genutzt wurde und anschließend folgt die Ausgabe. In der Analyse wird immer auf das Kapitel innerhalb eines Dokuments und die Seite verwiesen. Der Verweis bezieht sich dabei immer auf den äquivalenten

Abschnitt im beiliegenden Dokument. Als Beispiel: Wenn im Abschnitt "Anforderungsspezifikation" auf einen Inhalt verwiesen wird, ist dieser in dem Dokument "04_Anforderungsspezifikation.pdf" zu finden. Zu Beginn jedes Unterkapitels wird aber auch nochmal erwähnt, in welchem Dokument sich die Ausgaben befinden.

Im Rahmen der Arbeit wurde eine Tabelle erstellt, die als Übersicht dient. In den einzelnen Kapiteln wird diese Tabelle schrittweise aufgedeckt, um einen schnellen Überblick über die Ergebnisse des jeweiligen Kapitels zu liefern.

2 Grundlagen

Im ersten Kapitel wird erläutert, was LLM-Tools sind, wie sie trainiert werden und wie sie funktionieren. Anschließend werden die einzelnen Tools, die in dieser Arbeit betrachtet werden, vorgestellt und deren Entwicklung beschrieben.

Im zweiten Teil des Kapitels wird auf die einzelnen Dokumente, die während des Software-Engineering-Prozesses erstellt werden, näher eingegangen und es wird festgelegt, welche Dokumente und Abschnitte in der Analyse betrachtet werden. Da sich die Arbeit bei der Erstellung der Dokumente an das Modul "Secure Software Engineering" orientiert, werden einige Kapitel und Abschnitte nicht betrachtet. Daher wird bei der Beschreibung der entsprechenden Abschnitte ohne weitere Begründung erwähnt, dass diese im Folgenden nicht weiter betrachtet werden. Der Vollständigkeit halber werden diese aber trotzdem hier im Grundlagenkapitel aufgelistet.

2.1 LLM Tools

Unter LLM-Tools versteht man Sprachmodelle, die auf einem Large Language Model (LLM) basieren. Ein LLM ist ein Deep-Learning-Algorithmus, der mit sehr großen Datensätzen trainiert wird. Diese Modelle finden häufig Anwendung im Bereich des Natural Language Processing (NLP), wo sie verwendet werden, um Abfragen in natürlicher Sprache zu beantworten oder Ergebnisse zu liefern. LLMs können neue Inhalte verstehen, zusammenfassen, generieren und vorhersagen. Durch das Training sammeln LLMs Milliarden von Parametern, bei denen es sich um Variablen handelt, die im Modell angepasst werden, um neue Inhalte abzuleiten.

LLMs basieren auf einem Transformer-Modell, das Eingaben in Token umwandelt und dann gleichzeitig mathematische Gleichungen ausführt, um Beziehungen zwischen den Token zu ermitteln. Dadurch kann der Computer Muster erkennen, die auch ein Mensch wahrnehmen würde, wenn ihm die gleiche Frage gestellt wird. Zudem verwenden Transformer-Modelle Selbstaufmerksamkeitsmechanismen, die es dem Modell ermöglichen, schneller zu lernen als herkömmliche Modelle. Somit kann das Transformer-Modell verschiedene Teile der Sequenz oder den gesamten Kontext eines Satzes berücksichtigen, um Vorhersagen zu generieren.

Grundsätzlich bestehen LLMs aus vier neuronalen Netzwerkschichten: der wiederkehrenden Ebene, der Einbettungsebene, der Feedforward-Ebene und der Aufmerksamkeitsebene. Diese Schichten arbeiten zusammen, um den Eingabetext zu verarbeiten und Ausgabeinhalte zu generieren.

Die wiederkehrende Ebene dient dazu, die Wörter des Eingabetextes der Reihe nach zu interpretieren

und die Beziehungen zwischen den Wörtern in einem Satz zu erfassen. Die Einbettungsebene erfasst die semantische und syntaktische Bedeutung der Eingabe, sodass das Modell den Kontext verstehen kann. Die Feedforward-Schicht besteht aus mehreren vollständig verbundenen Schichten, die die Eingabeeinbettungen transformieren. Dadurch ermöglichen diese Schichten dem Modell, Abstraktionen auf höherer Ebene zu verstehen und somit die Absicht des Benutzers mit der Texteingabe zu erfassen. Die Aufmerksamkeitsebene ermöglicht es dem Sprachmodell, sich auf einzelne Teile des Eingabetextes zu konzentrieren, die für die Aufgabe relevant sind, und dadurch die genauesten Ausgaben zu generieren.

Damit ein großes Sprachmodell Texteingaben empfangen und eine Ausgabevorhersage generieren kann, muss es zunächst allgemein geschult und anschließend feinabgestimmt werden, um spezifische Aufgaben ausführen zu können. Für die Schulung werden riesige Datenmengen im Petabyte-Bereich benötigt. Das Training verläuft mehrstufig und beginnt in der Regel mit einem unbeaufsichtigten Lernansatz, bei dem das Modell mit unstrukturierten und unbeschrifteten Daten trainiert wird, da diese in größeren Mengen verfügbar sind. In dieser Phase leitet das Modell Beziehungen zwischen verschiedenen Wörtern und Konzepten ab.

Anschließend erfolgt die Schulung und Feinabstimmung durch eine Form des selbstüberwachten Lernens. Dabei wird eine Datenkennzeichnung durchgeführt, durch die das Modell verschiedene Konzepte besser und genauer identifizieren kann. Im nächsten Schritt durchläuft das LLM den Transformationsprozess im Rahmen des Deep Learning. Die Transformer-Modellarchitektur ermöglicht es dem LLM, mittels eines Selbstaufmerksamkeitsmechanismus Beziehungen und Verbindungen zwischen Wörtern und Konzepten zu erkennen, indem bestimmten Elementen (Token) Bewertungen zugewiesen werden, um die Beziehungen festzulegen.

Entscheidend für die Leistungsfähigkeit und die Qualität eines Sprachmodells sind die Datensätze, die zum Trainieren verwendet werden [1]. Daher wird im weiteren mit drei unterschiedlichen LLM Tools gearbeitet, auf die in den folgenden Abschnitten einzeln eingegangen wird.

2.1.1 ChatGPT

ChatGPT wurde von OpenAI am 30. November 2022 veröffentlicht und entwickelte sich aufgrund seiner fortschrittlichen generativen KI-Fähigkeiten rasch zu einer internationalen Sensation. Wenn man ChatGPT fragt, ob es einen Infotext zu sich selbst erstellen kann, erhält man die Antwort in Prompt 2.1.

Die Beta-Version der Plattform war zunächst kostenlos verfügbar, und OpenAI bot den Nutzern weiterhin eine kostenlose Basisversion an. Im Februar 2023 führte OpenAI ein kostenpflichtiges Abonnement ein, das unbegrenzten Zugriff auf ChatGPT sowie eine erweiterte Palette von Funktionen und Diensten ermöglichte.

Trotz der begeisterten Resonanz löste ChatGPT auch einige Kontroversen aus. Es lieferte teilweise fehlerhafte Antworten, insbesondere bei Aufgaben wie Schlussfolgerungen ziehen, Nuancen analysieren, Meinungen bewerten oder Vorhersagen treffen. Pädagogen weltweit äußerten Bedenken hinsichtlich des Einsatzes von ChatGPT im Bildungsbereich, auch an Universitäten wurde der Einsatz kritisch diskutiert.

Prompt: Erstell mir einen Infotext über dich für meine Bachelorarbeit

ChatGPT ist ein fortschrittlicher KI-gestützter Sprachassistent, der auf der GPT-4-Architektur von OpenAI basiert. Entwickelt im Rahmen der Forschung und Entwicklung im Bereich Künstliche Intelligenz, ist ChatGPT in der Lage, natürliche Sprache zu verstehen und darauf zu antworten. Dies macht ihn zu einem vielseitigen Werkzeug für eine Vielzahl von Anwendungen, von der Beantwortung alltäglicher Fragen bis hin zur Unterstützung bei komplexeren Aufgaben wie dem Schreiben von Texten, der Generierung von Ideen und der Bereitstellung von Informationen.

Die GPT-4-Architektur, auf der ChatGPT basiert, verwendet tiefes maschinelles Lernen, um menschliche Sprache zu verarbeiten und zu erzeugen. Durch die Analyse großer Mengen an Textdaten kann das Modell Muster und Zusammenhänge erkennen, die es ihm ermöglichen, kontextuell relevante und kohärente Antworten zu geben. Diese Fähigkeiten machen ChatGPT zu einem nützlichen Instrument in Bereichen wie Bildung, Forschung, Kundenservice und mehr.

Ein weiterer bemerkenswerter Aspekt von ChatGPT ist seine Fähigkeit, sich an eine Vielzahl von Themen anzupassen und Informationen auf klare und präzise Weise zu vermitteln. Dies wird durch kontinuierliches Training und Verbesserung des Modells erreicht, wobei sowohl technische Fortschritte als auch Nutzerfeedback berücksichtigt werden.

Zusammengefasst ist ChatGPT ein leistungsstarker und vielseitiger KI-Assistent, der durch seine fortschrittliche Technologie und seine breite Anwendbarkeit einen wertvollen Beitrag in vielen Bereichen leistet.

Prompt 2.1: Infotext ChatGPT, Quelle: ChatGPT

Trotz dieser Kontroversen schien ChatGPT bereit zu sein, die Erstellung von Inhalten im Internet zu revolutionieren. Webentwickler und Content-Ersteller nutzten ChatGPT zunehmend als Recherche- und Schreibhilfe für Website-Texte.

Im März 2023 brachte OpenAI eine neue Version heraus, bekannt als GPT-4. Ein bedeutender Unterschied zu früheren Versionen bestand darin, dass GPT-4 Eingaben sowohl in Text- als auch in Bildformaten akzeptierte. Dies ermöglichte dem Chatbot, Daten aus Diagrammen, Grafiken und Screenshots zu interpretieren. Zudem machte GPT-4 deutlich weniger Denkfehler und sachliche Fehler, auch "Halluzinationen" genannt, als das zuvor verwendete GPT-3.5-Modell. Am Ende des Jahres hatte sich ChatGPT weiterentwickelt und konnte beispielsweise Bilder analysieren, Sprachansagen verarbeiten und gesprochene Antworten geben.

Im Mai 2024 kündigte OpenAI ein aktualisiertes Modell namens ChatGPT-4o an, das schnellere Ergebnisse sowie verbesserte Text- und Audiofunktionen versprach. Dieses Modell umfasst auch ein neues Konversations-KI-System namens "Sky" [2].

2.1.2 Google Gemini

Im Dezember 2023 stellte Google eine generative Künstliche Intelligenz mit dem Namen Google Gemini vor. Sie wurde von Google DeepMind entwickelt und ist laut Aussage von Google das größte, leistungsstärkste und vielseitigste LLM, das bisher von Google entwickelt und realisiert wurde [3]. In der ursprünglichen Ankündigung wurde das KI-Sprachmodell unter dem Namen Gemini vorgestellt. Im

Februar 2024 entschied Google, auch den bisher als Bard bekannten Chatbot in Gemini umzubenennen. Damit ist Gemini mittlerweile sowohl der Name des Google Chatbot als auch des zugrundeliegenden Sprachmodells.

Bei Nachfrage an Gemini, ob es einen Infotext über sich erstellen kann, erhält man die Antwort in Prompt 2.2 und Prompt 2.3.

Gemini 1.0 wurde in drei verschiedenen Größen optimiert: Gemini Ultra als das größte und leistungsfähigste Modell für hochkomplexe Aufgaben, Gemini Pro als das beste Modell zur Anwendung in einem breiten Aufgabenspektrum und Gemini Nano als das effizienteste Modell für Aufgaben, die direkt auf dem Gerät verarbeitet werden.

Die Gemini-Modelle sind von Grund auf multimodal konzipiert, was bedeutet, dass sie mit verschiedenen Arten von Informationen umgehen können. Gemini ist in der Lage, Texte, Programmcode, Bilder, Videos, Audioinformationen und deren Kombinationen zu analysieren, zu verstehen, zu kombinieren und zu generieren. Da Gemini auch mit Skizzen und handgeschriebenem Text umgehen kann, eignet es sich für den Einsatz in Fachbereichen wie zum Beispiel Mathematik oder Physik. Im Programmierbereich werden gängige Programmiersprachen wie Python, Java, C++ oder Go unterstützt. Gemini kann Programmcode analysieren, kommentieren und auch generieren. Dadurch steht Gemini in Konkurrenz zu anderen LLMs wie den GPT-Sprachmodellen von Open AI. Gemini soll jedoch die Konkurrenzprodukte in vielen Benchmarks deutlich übertreffen.

Gemini baut auf den ebenfalls von Google entwickelten Sprachmodellen LaMDA und PaLM 2 auf. Wie üblich für LLMs (Large Language Models) basiert auch Gemini auf einer Transformer-Architektur. Google hat die Architektur des Modells zur Skalierung der Trainingsfähigkeiten optimiert und mit einem Multi-Query-Aufmerksamkeitsmechanismus ausgestattet. Die unterstützte Kontextlänge beträgt 32.000 Tokens und in der Version 1.5 bis zu einer Million Tokens. Gemini wurde mit Daten verschiedener Modalitäten trainiert, und als Trainingsplattform verwendete Google Rechner-Cluster und Tensor Processing Units (TPUs) der neuesten Generation v4 und v5 als KI-Beschleuniger.

Google hat begonnen, die KI in zahlreiche eigene Produkte zu integrieren, um die Fähigkeiten der Websuche, des Chrome-Browsers, des Gmail-Services, verschiedener Entwicklungstools und anderer Produkte und Services zu verbessern und zu erweitern. Mitte Februar 2024 stellte Google Gemini 1.5 und 1.5 Pro vor, welche bis zu eine Million Tokens verarbeiten können. Diese neuen Versionen sollen die Benchmark-Ergebnisse der Vorgängerversion deutlich übertreffen [4].

Prompt: Erstell mir einen Infotext über dich für meine Bachelorarbeit
Was ist Gemini?

Gemini ist ein Large Language Model (LLM) von Google AI, das mit einem riesigen Datensatz aus Text und Code trainiert wurde. Es kann Text generieren, Sprachen übersetzen, verschiedene Arten von kreativen Inhalten schreiben und Ihre Fragen auf informative Weise beantworten. Gemini befindet sich noch in der Entwicklung, hat aber bereits gelernt, viele Arten von Aufgaben auszuführen, einschließlich:

- Befolgen Sie Anweisungen und erledigen Sie Aufgaben sorgfältig.
- Beantworten Sie Ihre Fragen umfassend und informativ, auch wenn sie offen, herausfordernd oder seltsam sind.
- Generieren Sie verschiedene kreative Textformate, wie Gedichte, Codes, Skripte, Musikstücke, E-Mails, Briefe usw. Ich werde mein Bestes tun, um alle Ihre Anforderungen zu erfüllen.

Technische Details

Gemini basiert auf der Transformer-Architektur und wird mit einem massiven Datensatz aus Text und Code trainiert. Es verwendet eine Vielzahl von Techniken, um seine Leistung zu verbessern, einschließlich:

- **Aufmerksamkeit:** Ermöglicht es Gemini, sich auf die relevantesten Teile einer Eingabe zu konzentrieren.
- **Beam Search:** Ermöglicht Gemini, die besten möglichen Ausgaben zu generieren.
- **Multitasking:** Ermöglicht Gemini, mehrere Aufgaben gleichzeitig zu lernen und auszuführen.

Anwendungen

Gemini kann für eine Vielzahl von Anwendungen eingesetzt werden, einschließlich:

- **Generierung von kreativen Inhalten:** Gemini kann verwendet werden, um Gedichte, Codes, Skripte, Musikstücke, E-Mails, Briefe usw. zu generieren.
- **Sprachübersetzung:** Gemini kann verwendet werden, um Text von einer Sprache in eine andere zu übersetzen.
- **Beantwortung von Fragen:** Gemini kann verwendet werden, um Ihre Fragen umfassend und informativ zu beantworten, auch wenn sie offen, herausfordernd oder seltsam sind.
- **Zusammenfassen von Texten:** Gemini kann verwendet werden, um lange Textpassagen zu zusammenfassen.
- **Schreiben verschiedener Arten von kreativen Textformaten:** Gemini kann verwendet werden, um verschiedene Arten von kreativen Textformaten zu schreiben, wie Gedichte, Codes, Skripte, Musikstücke, E-Mails, Briefe usw.

Prompt 2.2: Infotext Gemini 1, Quelle: Gemini

Ethische Überlegungen

Es ist wichtig, Gemini verantwortungsvoll und ethisch zu nutzen. Zu den ethischen Überlegungen, die bei der Verwendung von Gemini zu beachten sind, gehören:

- **Vermeidung von Vorurteilen:** Gemini wird mit einem riesigen Datensatz aus Text und Code trainiert. Dieser Datensatz kann Vorurteile enthalten, die sich in den Ausgaben von Gemini widerspiegeln können. Es ist wichtig, sich dieser Vorurteile bewusst zu sein und Maßnahmen zu ergreifen, um sie zu mindern.
- **Vermeidung von Schadenszufügung:** Gemini kann verwendet werden, um schädlichen Text zu generieren, z. B. Hassreden oder Gewaltandrohungen. Es ist wichtig, Gemini nicht zu verwenden, um anderen Schaden zuzufügen.
- **Respekt vor der Privatsphäre:** Gemini kann verwendet werden, um personenbezogene Daten zu generieren. Es ist wichtig, Gemini nicht zu verwenden, um die Privatsphäre anderer zu verletzen.

Fazit

Gemini ist ein leistungsstarkes Tool, das für eine Vielzahl von Anwendungen eingesetzt werden kann. Es ist jedoch wichtig, Gemini verantwortungsvoll und ethisch zu nutzen.

Prompt 2.3: Infotext Gemini 2, Quelle: Gemini

2.1.3 Le Chat

In Prompt 2.4 sieht man die Antwort die Le Chat gibt, wenn man ihn nach einem Infotext über sich fragt. Le Chat ist ein, von dem französischen KI-Startup Mistral AI veröffentlichter Chatbot. Mistral AI startete mit seinen frei verwendbaren Sprachmodellen, also auf der Grundlage von Open Source, erfolgreich durch. Nun hat Mistral AI sein bislang größtes Modell 'Mistral Large' veröffentlicht. Diesmal allerdings nicht auf der Basis von Open Source, sondern ausschließlich über die eigene Webseite und der KI-Infrastruktur Microsoft Azure. Es lassen sich allerdings API-Keys für Programmierschnittstellen erstellen, um z.B. Mistral Large über seinen eigenen Server laufen zu lassen und so für andere User auf der eigenen Homepage verfügbar zu machen. Mit Mistral Large wurde auch Le Chat veröffentlicht, welcher aktuell kostenfrei verwendet werden kann. Le Chat bietet derzeit noch sehr wenige Funktionen an. Es stehen lediglich Texteingabe und -ausgabe zur Verfügung. Die Datengrundlage reicht nur bis 2021, weshalb es auch hier, für die Jahre 2022 bis heute, zu der Problematik der Halluzination kommen kann. Grundsätzlich kann man zwischen drei Sprachmodellen auswählen: Large, Next und Small. Large bietet überlegene Denkfähigkeit, Next ist ein Prototyp-Modell für erhöhte Kürze und Small arbeitet schnell und kosteneffektiv [5].

Prompt: Erstell mir einen Infotext über Le Chat für meine Bachelorarbeit Einleitung

Le Chat ist ein fortschrittliches Sprachmodell, das auf künstlicher Intelligenz (KI) und maschinellern Lernen basiert. Entwickelt von einem Team aus Experten für KI und Sprachverarbeitung, hat Le Chat die Fähigkeit, menschliche Sprache zu verstehen, zu interpretieren und darauf zu reagieren, was es zu einem wertvollen Werkzeug in verschiedenen Branchen macht.

Funktionsweise

Le Chat nutzt eine Kombination aus Techniken des Natural Language Processing (NLP) und des Deep Learning. NLP ermöglicht es dem Modell, die Struktur und den Kontext der menschlichen Sprache zu verstehen, während Deep Learning-Algorithmen verwendet werden, um Muster in den Daten zu erkennen und die Genauigkeit der Vorhersagen zu verbessern.

Anwendungsbereiche

Le Chat kann in einer Vielzahl von Anwendungen eingesetzt werden. Einige der wichtigsten Anwendungsbereiche sind:

- **Kundenservice:** Le Chat kann als virtueller Assistent eingesetzt werden, um Kundenanfragen zu beantworten, Produktempfehlungen auszusprechen und den Kunden bei der Lösung von Problemen zu unterstützen.
- **Bildung:** Le Chat kann als Tutor-System eingesetzt werden, um Schülern und Studenten bei der Lösung von Aufgaben und beim Verständnis von komplexen Konzepten zu helfen.
- **Gesundheitswesen:** Le Chat kann als virtueller Gesundheitsassistent eingesetzt werden, um Patienten bei der Verwaltung ihrer Medikamente zu unterstützen, Gesundheitstipps zu geben und in Notfällen Erste Hilfe-Anweisungen bereitzustellen.

Vorteile

Le Chat bietet eine Reihe von Vorteilen gegenüber herkömmlichen Sprachmodellen. Einige der wichtigsten Vorteile sind:

- **Verbesserte Genauigkeit:** Le Chat nutzt fortschrittliche Deep Learning-Algorithmen, um die Genauigkeit der Vorhersagen zu verbessern und eine bessere Benutzererfahrung zu bieten.
- **Skalierbarkeit:** Le Chat kann leicht skaliert werden, um eine große Anzahl von Anfragen gleichzeitig zu bearbeiten, was es zu einer idealen Lösung für Unternehmen mit hohem Kundenaufkommen macht.
- **Kosteneffizienz:** Le Chat kann als kosteneffiziente Alternative zu menschlichen Kundenservice-Mitarbeitern eingesetzt werden, was zu erheblichen Einsparungen für Unternehmen führen kann.

Fazit

Le Chat ist ein innovatives Sprachmodell, das die Fähigkeit hat, die Art und Weise, wie wir mit Maschinen interagieren, zu revolutionieren. Mit seiner fortschrittlichen Technologie und seinen vielfältigen Anwendungsmöglichkeiten ist Le Chat ein wichtiger Meilenstein in der Entwicklung der KI und ein Werkzeug, das in Zukunft eine noch größere Rolle spielen wird.

Prompt 2.4: Infotext Le Chat, Quelle: Le Chat

2.2 Software Engineering Prozess

Software-Engineering ist eine technische Disziplin, die sich umfassend mit der Erstellung von Software beschäftigt, beginnend bei der Konzeption, über den Betrieb, bis hin zur Wartung. Die wesentlichen Aktivitäten im Software-Engineering umfassen die Softwarespezifikation, -entwicklung, -validierung und -weiterentwicklung [6]. Während des Prozesses werden mehrere Dokumente erstellt und im Verlauf angepasst. Diese umfassen Besprechungsprotokolle, eine Risikoliste, ein Projekthandbuch, eine Anforderungsspezifikation, ein Architekturdokument und eine Testspezifikation. In der Abbildung 2.1 sieht man, welche Inhalte die einzelnen Dokumente umfassen. Diese werden im Folgenden etwas genauer beschrieben, um einen Überblick über die Dokumente zu vermitteln. Als Quelle hierfür wurden die Foliensätze von dem Modul "Secure Software Engineering" verwendet [7].

Dokument	Inhalte
01 Besprechungsprotokoll	Protokolle
02 Projekthandbuch	Projekthandbuch
03 Risikoliste	Risiko- & Maßnahmentabelle
04 Anforderungsspezifikation	Einleitung Systemübersicht Fachliches Datenmodell Anwendungsfalldiagramm AF Benutzungsschnittstelle Dialog Nichtfunktionale Anforderungen
05 Architekturdokument	Technische Infrastruktur Komponentenmodell (Detaillierte Schnittstellenbeschreibung) AF
06 Testspezifikationen	Testkonzept AF Testdaten Testprotokoll

Abbildung 2.1: Dokumente im Software Engineering

2.2.1 Besprechungsprotokoll

Alle Team- und Kundengespräche werden systematisch in Besprechungsprotokollen dokumentiert. Diese Protokolle enthalten Aufträge, Beschlüsse, Feststellungen und Informationen, die in Form von Texten, Bildern, Diagrammen usw. festgehalten werden. Aufträge müssen mit einem Fälligkeitsdatum und einem Verantwortlichen versehen sein, während Beschlüsse klar und unmissverständlich formuliert werden sollten. Im Rahmen der Anforderungsanalyse werden in Beschlüssen die Benutzeranforderungen erfasst. Feststellungen hingegen sind Beschlüsse, die keine Abstimmung benötigen, können aber ebenfalls Anforderungen an das System enthalten. Informationen bieten den Projektmitgliedern wichtige Hinweise. Dabei ist zu beachten, dass ein Besprechungsprotokoll keine Erlebniserzählung sein soll.

Der Aufbau eines Besprechungsprotokolls gestaltet sich wie folgt: Am Anfang stehen Datum, Thema, Verfasser, Teilnehmer und Verteiler. Anschließend wird eine Tabelle mit den Spalten Nummer, Art, Beschreibung, Termin und Verantwortlicher erstellt. Die Nummer stellt eine eindeutige ID im Protokoll dar. Die Art spezifiziert, ob es sich um einen Auftrag (=A), einen Beschluss (=B), eine Feststellung (=F) oder eine Information (=I) handelt. Die Beschreibung fasst den Auftrag, den Beschluss, die Feststellung oder die Information kurz und präzise zusammen. Der Termin gibt an, bis wann der Auftrag erledigt sein muss. Der Verantwortliche benennt die Person, die den Auftrag ausführen muss, den Beschluss gefasst hat oder die Feststellung bzw. die Information geliefert hat.

2.2.2 Projekthandbuch

Das Projekthandbuch dient dazu, die Projektmitglieder umfassend über das Projekt zu informieren, einschließlich der Vorgeschichte, Vertragsbasis, Grundlagen, Ziele, Ergebnisse und mehr. Es bietet zudem eine Übersicht über die Projektorganisation, wie z.B. die Teamaufstellung, das Berichtswesen und die Dokumentenablage, sowie über wesentliche Projektabläufe wie Änderungsverfahren, Risikomanagement und Aufwandsverfolgung. Ziel des Projekthandbuchs ist es, neuen Teammitgliedern den Einstieg zu erleichtern und auf wichtige Dokumente hinzuweisen, die für die Projektabwicklung notwendig sind. Damit fungiert das Projekthandbuch als zentrale Anlaufstelle für das Projekt.

Das Dokument beginnt mit einer Einleitung, die die Rahmenbedingungen klärt. Dieser Abschnitt enthält den Zweck des Dokuments, Hinweise zur Redaktion, in denen festgelegt wird, wer für das Dokument verantwortlich ist, sowie Informationen zum Verteiler, also wer bei Änderungen informiert werden muss.

Im Anschluss folgt die Projektbeschreibung, die die Rahmenbedingungen des Projekts darlegt. Dieser Teil enthält einen Abschnitt zur Vorgeschichte, in dem beschrieben wird, wie es zu dem Projekt kam. Ein weiterer Abschnitt bietet eine inhaltliche Kurzdarstellung des Projekts. Danach werden in einem Abschnitt zur Vertragsbasis der zeitliche Projektrahmen, die Anzahl der Teammitglieder und der Umfang der Bearbeiter-Stunden festgelegt. Es folgt eine Beschreibung des Projektergebnisses, in der festgelegt wird, welche Produkte geliefert werden müssen. Ein weiterer Abschnitt zu sonstigen Besonderheiten beschreibt informelle Ziele.

Ein nachfolgendes Kapitel verweist auf die Risikoliste.

Daraufhin folgt ein Kapitel zur Projektorganisation, das die Teamstruktur und die Kommunikationswege beschreibt. Dieses Kapitel beginnt mit dem Teamaufbau, in dem die Aufgabenverteilung und die E-Mail-Adressen der einzelnen Teammitglieder vermerkt sind. Anschließend wird die Zusammenarbeit mit den Kunden definiert. Danach folgt ein Abschnitt über regelmäßige Pflichtbesprechungen für die Teammitglieder.

Grundsätzlich würden hier noch drei Abschnitte folgen, die jedoch nicht weiter betrachtet werden: Zum einen wird beschrieben, ob das Projekt besondere Softwarelizenzen benötigt. Danach gibt es einen Abschnitt zur Sicherheit, in dem beschrieben wird, wie mit Sicherheitsaspekten wie Backup, Zugangskontrollen und Verschlüsselungen umgegangen wird. Als Letztes gibt es einen Abschnitt für sonstige Spielregeln.

Nun würde ein Kapitel zum Thema "Planungen" folgen. Darauf wird jedoch im Modul "Secure Software Engineering" aufgrund der kurzen Laufzeit des Projekts verzichtet. Daher wird dies auch im Folgenden der Arbeit nicht weiter betrachtet.

Das Kapitel würde sich der Planung widmen und verschiedene Planungsaspekte behandeln. Der erste Abschnitt befasst sich mit dem Projektplan. Hier werden die festgelegten Meilensteine beschrieben, einschließlich der Startzeiten der einzelnen Aktivitäten, der parallelen Abläufe, der Abschlusszeiten der verschiedenen Tätigkeiten, der Entstehungszeitpunkte der Produkte und der möglichen Entscheidungspunkte. Der zweite Abschnitt behandelt die Restaufwandsschätzung. Hier wird wöchentlich eine Schätzung abgegeben, damit der Aufwand nicht aus dem Ruder läuft. Dafür gibt es einen Verweis auf ein externes Dokument.

Auch die folgenden beiden Kapitel zum Thema Qualitätssicherung und Datenschutz entfallen hier.

Anschließend folgt ein Kapitel zur Dokumentation und Ablage, in dem beschrieben wird, wie und wo Produkte bzw. Dokumente abgelegt werden. Es beginnt mit einem Abschnitt zum Dokumenteneingang und -ausgang, in dem festgelegt wird, wie mit extern eingehenden Informationen umgegangen wird und wie ausgehende Informationen protokolliert werden. Im zweiten Abschnitt wird die Ablage und Archivierung geregelt, wobei erläutert wird, wie die Daten im Projektarchiv zu organisieren sind und in welchen Verzeichnissen die jeweiligen Daten abgelegt werden.

Die letzten beiden Kapitel werden ebenfalls nicht weiter betrachtet: Als vorletztes Kapitel wird in der Referenzliste auf alle sonstigen existierenden Dokumente verwiesen, die mit der Projektabwicklung in Zusammenhang stehen.

Zuletzt werden in einem Abkürzungsverzeichnis und Glossar spezielle Abkürzungen und wichtige Begriffe erklärt.

2.2.3 Risikoliste

Die Risikoliste ist ein zentrales Instrument, das dem gesamten Team einen umfassenden Überblick über bekannte Risiken und die entsprechenden Maßnahmen bietet. Der Projektleiter muss diese Risiken stets im Auge behalten, um eine fundierte Projektplanung sicherzustellen. Zusätzlich kann die Risikoliste dem

Anforderungsanalysten bei der Identifikation von Anforderungen, dem Systemarchitekten bei der Wahl geeigneter technischer Lösungen und dem Testmanager bei der Entwicklung von Testfällen nützlich sein.

Der Inhalt der Risikoliste umfasst die Autoren des Dokuments, die Historie des Dokuments mit einer Zusammenfassung der Änderungen, die Liste der Risiken sowie eine Liste der möglichen Maßnahmen für jedes identifizierte Risiko. Die Historie enthält Spalten für Datum, Autor*in und Änderungen, um später vorgenommene Modifikationen leichter nachvollziehbar zu machen.

Die Risikoliste selbst setzt sich aus mehreren Attributen zusammen. Die Identifikationsnummer ist eine eindeutige Nummer zur Identifizierung des Risikos, während die Risikobezeichnung eine kurze Beschreibung des Risikos bietet. Eine ausführliche Darstellung des Risikos findet sich in der Spalte für die Risikobeschreibung. Das Datum beschreibt, wann das Risiko identifiziert wurde, und der Autor gibt an, wer das Risiko gemeldet hat. In der Spalte für Auswirkungen werden die möglichen Folgen des Risikos beschrieben. Die Risikowahrscheinlichkeit ist eine Schätzung der Eintrittswahrscheinlichkeit des Risikos, und der Risikoschaden oder das Risikoausmaß beschreibt das potenzielle Schadensausmaß, einschließlich eventueller Vertragsstrafen. Das Risikomaß berechnet sich aus der Multiplikation von Risikowahrscheinlichkeit und Risikoschaden. Die Risikoklasse ist eine Priorisierung der Risiken, typischerweise in Kategorien wie tolerierbar, unerwünscht, kritisch oder katastrophal. Schließlich wird für jedes Risiko ein Status festgelegt, der zwischen aktiv, eingetragen und geschlossen unterscheiden kann.

Die Maßnahmen zur Risikobewältigung umfassen ebenfalls mehrere Attribute. Zunächst wird der Typ der Maßnahme festgelegt, wobei zwischen Risiko verhindern, Risiko lindern oder minimieren, Risiko übertragen oder teilen und Risiko akzeptieren unterschieden werden kann. Eine detaillierte Beschreibung der Maßnahme folgt, und falls das Risiko den Status "geschlossen" erhält, wird hier die Begründung eingetragen. Wenn eine Maßnahme nicht sofort eingeleitet werden soll, wird das Ereignis, das die Maßnahme auslöst, in der Trigger-Spalte definiert. Die verantwortliche Person, die für die Umsetzung der Maßnahme zuständig ist, wird ebenfalls eingetragen. Der geplante Termin gibt an, bis wann die Maßnahme abgeschlossen sein soll, und der Ist-Termin beschreibt den voraussichtlichen Abschluss basierend auf aktueller Einschätzung. Der geplante Aufwand gibt die geschätzten Kosten der Maßnahme an, während der Ist-Aufwand den aktuellen Aufwand widerspiegelt. Auch für die Maßnahme gibt es einen Status, bei dem man zwischen geplant, aktiv oder beendet wählen kann. Das letzte Attribut umfasst die Restrisikowahrscheinlichkeit, den Restrisikoschaden, das Restrisikomaß und die Restrisikoklasse. Dies sind die geschätzte Wahrscheinlichkeit, der geschätzte Schaden, das geschätzte Maß und die geschätzte Klasse des Restrisikos nach Durchführung der Maßnahmen.

2.2.4 Anforderungsspezifikation

Die Anforderungsspezifikation definiert formell, welche Leistungen das zu entwickelnde System erbringen muss. Sie beschreibt detailliert, was das System leisten soll, ohne jedoch festzulegen, wie diese Leistungen erbracht werden. Das "Wie" wird im Architekturdokument behandelt. In der Praxis sind Anforderungen und Systementwurf eng miteinander verknüpft. Die Anforderungsspezifikation ist oft Teil

eines Vertrags und muss daher so vollständig und eindeutig wie möglich formuliert sein.

Diese Spezifikation dient mehreren Zwecken: Sie hilft Kunden und Endnutzern zu beurteilen, ob das System die gewünschten Leistungen erbringen wird und wie es später zu bedienen ist. Projektmanager nutzen die Spezifikation zur Erstellung des Angebots und zur Planung des detaillierten Entwicklungsprozesses. Administratoren können damit entscheiden, wie das System verwaltet werden muss. Für Systemarchitekten dient sie als Grundlage für ihr Architekturdokument, und Softwareentwickler nutzen sie, um zu verstehen, welches System entwickelt werden soll. Zusammen mit dem Architekturdokument bildet sie die Basis für die Implementierung. Testmanager verwenden die Spezifikation, um Validierungstests und Testfallspezifikationen zu entwickeln.

Der Inhalt einer Anforderungsspezifikation ist folgendermaßen aufgebaut:

Zunächst werden der Name des Softwareprodukts und des Herstellers genannt. Es folgt ein Vorwort, das die erwartete Leserschaft definiert, eine Versionshistorie des Dokuments mit Begründungen für neue Versionen und eine Zusammenfassung der Änderungen enthält sowie die Autoren des Dokuments nennt.

Die Einleitung beschreibt den Zweck und Umfang des Dokuments. Dies umfasst eine Beschreibung der Notwendigkeit des Systems, eine Kurzbeschreibung der Funktionalität und der Nachbarsysteme sowie eine Definition der Bedeutung des Systems für die gesamtwirtschaftlichen oder strategischen Ziele des Unternehmens. Es folgen Verweise auf weitere Ressourcen sowie Definitionen von Begriffen und Abkürzungen. Eine Übersicht des Dokuments schließt die Einleitung ab, welche allerdings in dieser Arbeit nicht weiter betrachtet wird.

Anschließend werden die Benutzeranforderungen definiert, welche aber ebenfalls nicht betrachtet werden. Diese enthalten eine Beschreibung der für den Benutzer bereitgestellten Dienste und der nichtfunktionalen Anforderungen in verständlicher Sprache, Diagrammen oder anderen für den Kunden nachvollziehbaren Notationen.

Es folgt ein Überblick über die Systemarchitektur, die die erwartete Systemarchitektur und die Einordnung des Systems in die Systemlandschaft des Kunden beschreibt.

Danach folgt das fachliche Datenmodell, das aus Entitätentypen besteht. Eine Entität ist ein individuelles und identifizierbares Exemplar von Dingen, Personen oder Begriffen der realen oder Vorstellungswelt, beschrieben durch seine Eigenschaften. Entitätentypen besitzen Attribute und Beziehungen, jedoch keine Fähigkeiten. Das fachliche Datenmodell wird mit Hilfe eines UML Klassendiagramms mit ergänzenden Beschreibungen bzw. Einschränkungen spezifiziert. Dieses enthält alle Entitätentypen mit deren Eigenschaften, Beziehungen und Einschränkungen.

Anschließend folgen Anwendungsfälle, die die funktionalen Anforderungen an das System beschreiben. Ein Anwendungsfalldiagramm gibt eine Übersicht über alle Anwendungsfälle. Ein Anwendungsfall besteht aus mehreren Szenarien, die konkrete Abläufe beschreiben, wie das System genutzt werden kann. Anwendungsfälle können textuell beschrieben oder durch Aktivitäts- oder Sequenzdiagramme ergänzt werden.

In der Anforderungsspezifikation gibt es dazu eine Übersicht aller Anwendungsfälle im UML Use-Case-Diagramm, welches alle Akteure, Anwendungsfälle, Beziehungen zwischen Akteur und Anwendungsfällen und eventuell include- und extend-Beziehungen zwischen den Anwendungsfällen darstellt. Ebenfalls gibt es eine Spezifikation jedes einzelnen Anwendungsfalls mit Hilfe von UML Aktivitätsdiagrammen und ergänzenden Beschreibungen. Diese enthalten eine genaue Ablaufbeschreibung, den Akteur, die Aufrufhäufigkeit, die Vorbedingung und den Auslöser sowie eine Dokumentation des Datenflusses.

Neben den Anwendungsfällen müssen auch die Benutzungsschnittstelle und die Dialoge beschrieben werden, über die die Anwendungsfälle ausgeführt werden. Eine Dialogbeschreibung enthält alle Details, einschließlich der Feldlängen und der zugeordneten Aktionen für jeden Button. Die Benutzungsschnittstelle stellt dar, wie die Dialoge miteinander verbunden sind.

Im Dokument werden alle Dialoge mit Bildern und strukturierter und detaillierter Beschreibung der Bestandteile spezifiziert. Die Benutzungsschnittstelle kann eine beliebige Darstellungsform sein.

Zum Schluss werden die nichtfunktionalen Anforderungen in natürlicher Sprache inklusiver Testbeschreibung aufgelistet.

2.2.5 Architekturdokument

Das Architekturdokument, auch bekannt als Softwarearchitektur-Dokument, beschreibt die Umsetzung des zu entwickelnden Systems. Die Anforderungsspezifikation [2.2.4] legt fest, "was" das System leisten soll, jedoch nicht "wie" dies erreicht wird. Diese Umsetzung wird im Architekturdokument detailliert beschrieben. In der Praxis sind die Anforderungsspezifikation und das Architekturdokument eng miteinander verzahnt.

Die Zielgruppe des Architekturdokuments umfasst hauptsächlich Softwareentwickler und Testmanager. Softwareentwickler nutzen das Dokument, um zu verstehen, wie das in der Anforderungsspezifikation beschriebene System entwickelt werden soll. Zusammen mit der Anforderungsspezifikation bildet es die Grundlage für die Implementierung. Testmanager verwenden das Architekturdokument, um Komponententests für das System zu entwickeln.

Das Architekturdokument besteht aus drei Hauptabschnitten: der Technischen Architektur, der Anwendungsarchitektur und das IT-Konzept.

Die Technische Architektur umfasst zwei Hauptkomponenten, die Technische Infrastruktur (TI-Architektur) und die Technische Struktur. Erstere beschreibt die Hardware- und Produktanforderungen, Zweitere die technischen Lösungen und Strukturen. Diese wird in der Arbeit aber nicht weiter betrachtet. Beide Komponenten werden durch erklärende Texte und Grafiken dargestellt.

Die Anwendungsarchitektur besteht ebenfalls aus zwei Hauptteilen: dem Komponentenmodell und dem Datenmodell. Das Komponentenmodell beschreibt die fachliche Struktur der Lösung auf einer groben Ebene mittels eines UML-Komponenten-Diagramms. Das Datenmodell hingegen beschreibt die Struktur der persistenten Daten und wird entweder durch ein UML-Komponenten-Diagramm oder Entity-Relationship-Diagramm dargestellt. Aber auch dieses wird hier nicht weiter betrachtet.

Das IT-Konzept enthält die detaillierte Schnittstellenbeschreibung. Diese beschreibt die Semantik und

Syntax der Komponentenschnittstellen ausführlich und wird durch Code oder Code-Dokumentation dokumentiert.

Darüber hinaus beinhaltet das IT-Konzept dynamische Beschreibungen exemplarischer Anwendungsfälle im Komponentenmodell, die mittels Aktivitätsdiagrammen oder Sequenzdiagrammen veranschaulicht werden.

2.2.6 Testspezifikation

Die Testfallspezifikation oder auch Software Test Dokumentation dient dazu, die Testdurchführung zu planen, die Testfälle zu spezifizieren und die Testergebnisse zu protokollieren. Mit Hilfe dieser ist es möglich, die Anwendung systematisch und strukturiert zu testen. Die Protokollierung der Testergebnisse ermöglicht es, eine Aussage über die Qualität zu geben. Nach einer Fehlerbehebung können die selben Tests erneut durchgeführt werden. Dadurch wird nichts vergessen.

Die Testspezifikation besteht aus einem Testkonzept, einer Testfall-Spezifikation und einem Testprotokoll. Das Testkonzept legt die Vorgehensweise, die verwendeten Mittel und den Ablauf der Testaktivitäten fest. In diesem Rahmen muss festgelegt werden, ob es irgendwelche Anforderungen an die Umsetzung gibt oder ob irgendwelche Dummy-Komponenten entwickelt werden müssen.

Die Testfall-Spezifikation listet die Testfälle auf. Ein Testfall besteht aus einer Kurzbeschreibung, der Testvoraussetzung, den Eingabewerten inkl. dem Testablauf, und den erwarteten Ausgabewerten.

Im Testprotokoll wird die Testdurchführung protokolliert. Dies beinhaltet, wann welcher Test durchgeführt wurde und ob der Test erfolgreich war oder nicht.

3 Anwendung von LLM Tools im Software Engineering

Dieses Kapitel befasst sich mit der Frage, wie LLM Tools im Software Engineering Prozess unterstützen können. Grundsätzlich gibt es da natürlich zahlreiche Möglichkeiten, die unmöglich alle im Rahmen einer Arbeit untersucht werden können. Daher widmet sich die Arbeit eher der selbstständigen Erstellung der Dokumente als der Unterstützung bei der Pflege oder ähnlichen Unterstützungsmöglichkeiten. Dazu wird im folgenden zu jedem Dokument beschrieben, was im anschließenden Kapitel "Praxisergebnisse und Vergleich" betrachtet wird. Die Abbildung 3.1 zeigt eine Tabelle mit möglichen KI Unterstützungen im Bezug auf die Inhalte der einzelnen Dokumente und soll einen kurzen Überblick dazu abbilden.

Dokument	Inhalte	Mögliche KI Unterstützung (Erstellung folgender Inhalte)
01 Besprechungsprotokoll	Protokolle	Schriftliches Gespräch in Protokoll umformen
02 Projekthandbuch	Projekthandbuch	Text erstellen
03 Risikoliste	Risiko- & Maßnahmentabelle	Risiko- & Maßnahmentabelle Risiken/Maßnahme erstellen lassen, Attribute selber festlegen
04 Anforderungsspezifikation	Einleitung	Text erstellen
	Systemübersicht	Skizze oder Beschreibung
	Fachliches Datenmodell	UML-Klassendiagramm oder Beschreibung & Beschreibung mit Einschränkungen
	Anwendungsfalldiagramm	UML-Diagramm oder Beschreibung
	AF	UML-Aktivitätsdiagramm oder Beschreibung
	Benutzungsschnittstelle	Skizze oder Beschreibung
	Dialog	Skizze oder Beschreibung & Erklärung
	Nichtfunktionale Anforderungen	Nichtfunktionale Anforderungen & Test
05 Architekturdokument	Technische Infrastruktur	Skizze oder Beschreibung
	Komponentenmodell	UML-Komponentenmodell oder Beschreibung & Erklärung
	(Detaillierte Schnittstellenbeschreibung)	Als Java Code (nur als extra)
	AF	UML-Sequenzdiagramm oder Beschreibung
06 Testspezifikationen	Testkonzept	Tabelle & Ablauf
	AF	Tests zu den Anwendungsfällen (Testablauf & -ergebnis)
	Testdaten	Tabelle
	Testprotokoll	(Tabelle erstellen)/ Rahmen erstellen

Abbildung 3.1: Mögliche LLM Tool Unterstützung

3.1 Besprechungsprotokoll

Im Unterabschnitt 2.2.1 wurde bereits erläutert, welche Inhalte in einem Besprechungsprotokoll enthalten sind. Nun stellt sich die Frage, wie LLM-Tools bei der Erstellung eines solchen Dokuments unterstützen können.

Eine Möglichkeit wäre, das Besprechungsprotokoll aufzeichnen zu lassen und anschließend automatisch

daraus das Protokoll zu erstellen. Dies wird jedoch in dieser Arbeit nicht betrachtet, da lediglich bei der kostenfreien Version ChatGPT-4o Dokumente als Eingabe genutzt werden können. Daher wäre dies nur in einem begrenzten Umfang und ohne Vergleiche testbar.

Eine alternative Möglichkeit wäre, das Meeting zu verschriftlichen und die schriftlichen Inhalte als Texteingabe zu verwenden. Mithilfe dieses schriftlichen Gesprächs könnten die Tools dann automatisch die Tabelle des Besprechungsprotokolls erstellen. Die zu Beginn des Besprechungsprotokolls aufgelisteten Daten – Datum, Thema, Verfasser, Teilnehmer und Verteiler – könnten dabei vernachlässigt werden, da diese Informationen vermutlich separat in die Eingabe eingefügt werden müssten, um erstellt zu werden. Dadurch gäbe es keine Zeitersparnis oder sonstigen Vorteil, sich diese Informationen automatisch erstellen zu lassen.

3.2 Projekthandbuch

Das Projekthandbuch ist ein zentrales Dokument im Software Engineering Prozess, das alle wesentlichen Informationen und Richtlinien eines Projekts zusammenfasst. Die spezifischen Inhalte dieses Dokuments wurden bereits im Unterabschnitt 2.2.2 dargelegt. Doch wie können LLM-Tools bei der Erstellung des Projekthandbuches unterstützend wirken?

Im Rahmen dieser Arbeit wird sich auf die Abschnitte “Zweck des Dokuments”, “Redaktion” und “Verteiler” im Kapitel “Einleitung” des Projekthandbuches sowie auf die Abschnitte “Vorgeschichte” und “Inhaltliche Kurzdarstellung” im Kapitel “Projektdefinition” beschränkt. Der Grund dafür ist, dass die restlichen Abschnitte des Projekthandbuchs aus den Unterlagen für das Modul “Secure Software Engineering” stammen und hauptsächlich als Bild aus der Präsentation in das Projekthandbuch eingefügt werden. Die aufgezählten Abschnitte könnten automatisch erstellt werden. Falls spezifische Informationen dafür benötigt werden, müssten diese in die Eingabe integriert werden.

3.3 Risikoliste

Nachdem im Unterabschnitt 2.2.3 die Bestandteile der Risikoliste beschrieben wurden, stellt sich auch hier die Frage, wie LLM Tools bei der Erstellung und Pflege der Risikoliste mitwirken und unterstützen können.

Eine Möglichkeit besteht darin, die gesamte Risikoliste einschließlich der zugehörigen Maßnahmentabelle von LLM Tools erstellen zu lassen. Dabei sollten alle Attribute für die Risiken klar festgelegt werden. Zudem sollten die Werte der einzelnen Risikoattribute in einem konsistenten Verhältnis zueinander stehen.

3.4 Anforderungsspezifikation

Der Unterabschnitt 2.2.4 beschreibt die einzelnen Inhalte der Anforderungsspezifikation. Doch wie können LLM-Tools bei der Erstellung dieser unterstützen?

Zunächst könnte die Einleitung des Dokuments vollständig von den Tools erstellt werden. Falls hierfür spezifische Informationen benötigt werden, sollten diese in die Eingabe integriert werden.

Anschließend folgt die Systemübersicht. Da außer ChatGPT-4o keines der Tools eigenständig Dokumente erstellen kann, wird die Zeichnung vermutlich selbst erstellt werden müssen. Es sollte jedoch möglich sein, die Beschreibung der Komponenten und deren Beziehungen von den Tools erstellen zu lassen und diese dann in eine Zeichnung umzusetzen. Ebenso kann der Text, der das System beschreibt und in die Systemlandschaft einordnet, automatisch generiert werden. Der Fokus liegt hier jedoch auf der Zeichnung sowie auf der Beschreibung der Komponenten und deren Beziehungen.

Ähnlich verhält es sich beim fachlichen Datenmodell. Hier könnte das UML-Klassendiagramm als PlantUML-Code oder in einem ähnlichen Format ausgegeben werden. Da dies jedoch eher an den Programmierfähigkeiten scheitert als an der Fähigkeit, ein fachliches Datenmodell zu erstellen, fließt es nicht negativ in die Bewertung ein, wenn dieses nicht erstellt werden kann. Die Beschreibung des Diagramms sollte allerdings so präzise sein, dass die Erstellung des Diagramms manuell erfolgen kann. Die dazugehörigen Beschreibungen und Einschränkungen können ebenfalls von den Tools generiert werden.

Anschließend folgt das Anwendungsfalldiagramm. Dieses könnte ebenfalls mittels PlantUML oder ähnlichen Tools grafisch erstellt werden, was jedoch aus den oben genannten Gründen keine Voraussetzung ist. Alternativ können die LLM-Tools die Rollen, deren Verbindungen und die zugehörigen Anwendungsfälle auflisten. Falls zusätzliche Bemerkungen zu den Anwendungsfällen erforderlich sind, können diese ebenfalls von den Tools erstellt werden.

Bei den detaillierten Beschreibungen der einzelnen Anwendungsfälle mit Aktivitätsdiagramm sollte der Fokus auf dem Aktivitätsdiagramm und den dazugehörigen Beschreibungen der Aktivitäten liegen. Das Aktivitätsdiagramm könnte ebenfalls mit PlantUML oder ähnlichen Tools erstellt werden. Dies ist aber auch hier keine Anforderung an die Tools, aufgrund den oben angegebenen Gründen. Falls dies jedoch nicht erstellt werden kann, ist eine genaue und detaillierte Beschreibung des Aktivitätsdiagramms erforderlich. Die Beschreibung der Aktivitäten sollte, unabhängig von der Darstellung des Diagramms, vollständig von den Tools generiert werden können.

Darauf folgt die Benutzungsschnittstelle, welche entweder vollständig durch PlantUML-Code oder ähnlichen Tools oder textuell beschrieben werden sollte. Die zusätzlichen Bemerkungen und die Zuordnung der Dialoge zu den Anwendungsfällen spielen hier erst einmal eine untergeordnete Rolle.

Die einzelnen Dialoge können textuell beschrieben oder im Chat skizziert werden, wobei es den Tools überlassen bleibt, welche Methode sie wählen. Die einzelnen Eingabefelder und Buttons sollten dabei erläutert werden. Die Beschreibung, wann der Dialog aufgerufen wird und was beim Schließen des

Dialogs passiert, spielt zunächst eine untergeordnete Rolle.

Zuletzt werden die nichtfunktionalen Anforderungen mitsamt den dazugehörigen Tests aufgelistet. Beides sollte von den Tools eigenständig erstellt werden können.

3.5 Architekturdokument

Das Architekturdokument beschreibt die grundlegende Struktur und die Designentscheidungen eines Softwareprojekts. Die genauen Inhalte wurden bereits im Unterabschnitt 2.2.5 erläutert. In welchem Maße bestehen hier Unterstützungsmöglichkeiten durch LLM-Tools?

Die technische Infrastruktur besteht aus einer Grafik und einem erläuternden Text. Die Grafik kann optional als "Skizze" im Chat erstellt werden, was jedoch keine zwingende Anforderung an die verwendeten Tools darstellt. Wenn eine Skizze erstellt wird, sollte auch ein erläuternder Text hinzugefügt werden. Andernfalls liegt der Fokus auf der textuellen Ausarbeitung. Dabei ist es zunächst unerheblich, ob der Text in Stichpunkten oder als Fließtext verfasst wird.

Anschließend folgt das Komponentenmodell, das optional wieder in PlantUML oder ähnlichen Tools erstellt werden kann. Alternativ kann auch eine detaillierte Beschreibung angefertigt werden. Dazu gehört ein begleitender Text, der die Komponenten und Schnittstellen beschreibt und deren Funktion erläutert. Ob dieser Text in Stichpunkten oder als Fließtext verfasst wird, spielt auch hier keine Rolle. Das Hauptaugenmerk liegt auf dem Komponentenmodell.

Nach der Schnittstellenbeschreibung folgt die dynamische Beschreibung exemplarischer Anwendungsfälle. Diese umfasst ein Sequenzdiagramm sowie die für den Anwendungsfall erforderlichen Vorbedingungen. Das Sequenzdiagramm kann ebenfalls als PlantUML-Code oder mit ähnlichen Tools erstellt werden, was jedoch keine zwingende Anforderung darstellt. Sollte dies nicht möglich sein, genügt auch eine detaillierte Beschreibung des Diagramms. Die Vorbedingungen sollten ohne größere Schwierigkeiten erstellt werden können, wobei sie hier zunächst von untergeordneter Bedeutung sind. Das Hauptaugenmerk liegt erneut auf dem Diagramm.

3.6 Testspezifikation

Im Unterabschnitt 2.2.6 wurde bereits erklärt, welche Inhalte die Testspezifikation umfasst. Wie wäre hier eine Unterstützung durch LLM-Tools möglich?

Zunächst wird das Testkonzept betrachtet. Dieses kann automatisch generiert werden. Falls das generierte Konzept nicht den eigenen Vorstellungen entspricht, kann man zunächst Stichpunkte selbst schreiben und sich dann den Text daraus erstellen lassen. Auch die dazugehörige Tabelle sollte sich von den LLM-Tools erstellen lassen.

Anschließend folgen die Tests für die einzelnen Anwendungsfälle. Die Testfälle kann man vollständig

von den Tools erstellen lassen. Die Testdaten, die bei diesen Tests benötigt werden, werden in einer Tabelle zusammengefasst. Auch diese Tabelle könnten die Tools automatisch generieren.

Die unausgefüllte Tabelle des Testprotokolls, z.B. Spalten- und Zeilenbeschriftung, könnte ebenfalls von den LLM-Tools erstellt werden. Die Ergebnisse könnten die Tools auch in die Tabelle mit eintragen, diese müssten allerdings in der Eingabe mit gegeben werden. Allerdings stellt sich hier in beiden Fällen die Frage, ob dies zu einer Zeitersparnis oder einer Arbeitserleichterung führt.

4 Praxisergebnisse und Vergleich

In diesem Kapitel wird beschrieben, welche Eingabe zur Erstellung der Dokumente mit den LLM Tools genutzt wurde. Anschließend werden die Ausgaben analysiert und, falls weitere Eingaben erforderlich sind, diese näher erläutert. Die Abbildung 4.1 zeigt die Endgültige Tabelle mit einer Übersicht der Bewertungen der Tools bei den unterschiedlichen Unterstützungsmöglichkeiten. Im folgenden wird beschrieben, wie diese Bewertungen entstanden sind.

Dokument	Inhalte	Mögliche KI Unterstützung (Erstellung folgender Inhalte)	ChatGPT	Gemini	Le Chat
01 Besprechungsprotokoll	Protokolle	Schriftliches Gespräch in Protokoll umformen			
02 Projekthandbuch	Projekthandbuch	Text erstellen			
03 Risikoliste	Risiko- & Maßnahmentabelle	Risiko- & Maßnahmentabelle Risiken/Maßnahme erstellen lassen, Attribute selber festlegen			
04 Anforderungsspezifikation	Einleitung Systemübersicht Fachliches Datenmodell Anwendungsfalldiagramm AF Benutzungsschnittstelle Dialog Nichtfunktionale Anforderungen	Text erstellen Skizze oder Beschreibung UML-Klassendiagramm oder Beschreibung & Beschreibung mit Einschränkungen UML-Diagramm oder Beschreibung UML-Aktivitätsdiagramm oder Beschreibung Skizze oder Beschreibung Skizze oder Beschreibung & Erklärung Nichtfunktionale Anforderungen & Test			
05 Architekturdokument	Technische Infrastruktur Komponentenmodell (Detaillierte Schnittstellenbeschreibung) AF	Skizze oder Beschreibung UML-Komponentenmodell oder Beschreibung & Erklärung Als Java Code (nur als extra) UML-Sequenzdiagramm oder Beschreibung			
06 Testspezifikationen	Testkonzept AF Testdaten Testprotokoll	Tabelle & Ablauf Tests zu den Anwendungsfällen (Testablauf & -ergebnis) Tabelle (Tabelle erstellen)/ Rahmen erstellen			
			Bewertungsstufen Sehr gut Gut Befriedigend Ausreichend Mangelhaft		

Abbildung 4.1: Ergebnisse

Bewertung

Um die Ausgaben der Tools besser vergleichen und ein nachvollziehbares Fazit ziehen zu können, wurde ein Bewertungssystem mit fünf Stufen entwickelt. Diese Bewertungsstufen ermöglichen eine strukturierte und transparente Methode zur Beurteilung der Qualität der Ausgaben von LLM-Tools. Die fünf Bewertungsstufen sind wie folgt definiert:

“Sehr gut” wird vergeben, wenn der Inhalt der Ausgabe fachlich korrekt ist und das Format den Vorgaben entspricht, sodass die Ausgabe direkt in das entsprechende Dokument eingefügt werden kann.

“Gut” wird vergeben, wenn der Inhalt leichte Fehler enthält, die jedoch mit wenigen Anpassungen korrigiert werden können, und das Format den Vorgaben entspricht. Eine Ausgabe wird ebenfalls als

“Gut” eingestuft, wenn der Inhalt korrekt ist, das Format jedoch kleinere Fehler aufweist. In diesem Fall sollte der richtige Inhalt leicht aus der Ausgabe kopiert und in das entsprechende Dokument im richtigen Format eingefügt werden können.

“Befriedigend” wird vergeben, wenn sowohl der Inhalt als auch das Format kleinere Fehler aufweisen, die sich jedoch mit wenigen Anpassungen korrigieren lassen, oder wenn der korrekte Inhalt aus der Ausgabe kopiert werden kann.

“Ausreichend” wird vergeben, wenn in der Ausgabe inhaltlich Teile fehlen, diese aber durch präzisere Eingaben teilweise verbessert werden können, auch wenn die Ausgabe immer noch nicht vollständig ist.

“Mangelhaft” wird vergeben, wenn der Inhalt der Ausgabe falsch oder unverständlich ist und sich auch nicht korrigieren lässt.

Diese Bewertungslinien sind allerdings nur grobe Orientierungen. In die Bewertung fließen auch noch spezifische Aspekte von dem Dokument mit ein, welches erstellt wird. Außerdem fließt auch der subjektive Eindruck mit in die Bewertung mit ein, wie “angenehm” sich die Dokumente erstellen lassen. Daher kann hier teilweise ein Tool, welches vielleicht auch nicht so gut ein Dokument erstellt, besser bewertet werden als ein Tool, was vielleicht ähnliche Fehler hat, aber womit deutlich Schwieriger zu arbeiten war.

Dadurch sind die Bewertungen eher Subjektiv angelehnt, sollen aber trotzdem eine grobe Richtung geben, wie gut das jeweilige Dokument erstellt wurde.

4.1 Besprechungsprotokoll

Für die Erstellung des Besprechungsprotokolls wurde ein verschriftlichtes Gespräch verwendet (siehe Kundengespräch im Anhang). Es wurde der folgende Prompt genutzt:

Erstell mir aus folgendem Gespräch ein erstes Besprechungsprotokoll für ein Projekt. Das Protokoll soll nur die wichtigen Punkte in Form einer Tabelle mit den Spalten “Nummer”: was eine eindeutige Nummer zur Identifikation ist, “Art”: eine Auswahl, ob es eine Information, ein Auftrag, eine Feststellung oder ein Beschluss ist, “Beschreibung”: Was den Punkt kurz und präzise zusammenfasst, “Termin”: ein genaues Datum, bis wann der Auftrag erledigt sein muss und “Verantwortlich”: Welches Teammitglied verantwortlich ist, enthalten. Aufträge müssen mit einem genauen Fälligkeitsdatum und einem Verantwortlichen versehen sein. Beschlüsse müssen klar und unmissverständlich formuliert werden. Feststellungen sind Beschlüsse, die keine Abstimmung benötigen und Informationen bieten den Projektmitgliedern wichtige Hinweise. [Hier folgte das Gespräch]

Prompt 4.1: Prompt Besprechungsprotokoll

Die Eingabe wurde bei ChatGPT, Gemini und Le Chat getätigt. Die Ergebnisse befinden sich in der Datei 01_Besprechungsprotokolle.pdf auf dem beiliegenden USB-Stick.

Es fiel auf, dass die Ausgaben der verschiedenen Tools trotz des gleichen Prompts teilweise erheblich voneinander abwichen. Dies betraf sowohl die Anzahl der erstellten Einträge im Besprechungsprotokoll als auch die Art und Weise, wie die Informationen zusammengefasst wurden. In einigen Fällen wurden mehrere Punkte zu einem Eintrag zusammengefasst, während in anderen Fällen mehrere Einträge daraus entstanden. Dies trat insbesondere bei der Anforderung auf, dass die Anwendung mit JavaFX als Server-Client-Architektur mit RMI entworfen werden sollte. Hier haben die Tools manchmal einen Eintrag dafür erstellt und manchmal drei einzelne Einträge. Dieses Verhalten zeigte sich auch, wenn derselbe Prompt in neuen Chats mit demselben Tool verwendet wurde. Besonders bei Gemini trat dieses Problem sehr häufig auf [siehe Kapitel Gemini 01/Gemini 05/...].

Ein wichtiger Aspekt bei der Erstellung des Prompts war die Notwendigkeit einer genauen Beschreibung der Spalten und der verschiedenen Arten (Information, Auftrag, Feststellung, Beschluss). Ohne diese genaue Beschreibung nutzten die Tools ein eigenes Format, das eher einer Stichpunktliste ähnelte [siehe Kapitel GPT 05/Gemini 06/Le Chat 03]. Doch auch die genaue Definition der Arten führte nicht immer zu konsistenten Ergebnissen, da die Zuweisung der Kategorien häufig unterschiedlich vorgenommen wurde.

Auf die Spalte "Termin" musste ebenfalls ein besonderes Augenmerk gelegt werden. Wenn im Prompt nicht explizit angegeben war, dass ein genaues Datum erforderlich ist, arbeiteten die Tools oft mit relativen Angaben wie "+2 Wochen" [siehe Kapitel GPT 04/Le Chat 02]. Selbst mit der Klarstellung, dass der Termin ein genaues Datum sein sollte, traten bei Le Chat Probleme auf: Es wurden immer Termine gewählt, die in der Vergangenheit lagen [siehe Kapitel Le Chat 01]. Bei Nachfrage wurde angegeben, dass die Termine beispielhaft gewählt und auf das aktuelle Datum, den 29.03.2023, bezogen seien.

Google Gemini hatte, wie bereits erwähnt, erhebliche Schwierigkeiten, nur die wichtigsten Punkte aus dem Gespräch herauszufiltern. Häufig wurden Besprechungsprotokolle mit etwa 30 Punkten erstellt, wobei jede einzelne Information separat und auch unwichtige Informationen aufgeführt wurden. Gemini fügte zudem eine neue Art "Frage" selbstständig hinzu, wodurch im Besprechungsprotokoll teilweise mehrere Punkte für eine Information erstellt wurden. Bei erneuten Eingaben variierte die Länge des Besprechungsprotokolls stark. Trotz des Hinweises, nur die wichtigen Punkte in das Protokoll aufzunehmen, neigte Gemini weiterhin dazu, sehr detaillierte und kleinteilige Protokolle zu erstellen [siehe Kapitel Gemini 05/Gemini 03].

Le Chat hingegen hielt sich häufig zu knapp. Dadurch wurden immer wieder wichtige Anforderungen nicht in das Besprechungsprotokoll aufgenommen, und es musste darauf geachtet werden, dass alle relevanten Informationen enthalten waren. Besonders die Anforderungen, dass mehrere Spiele gleichzeitig gespielt werden können und dass ansprechende Animationen verwendet werden sollen, wurden nicht mit aufgeführt [siehe Kapitel Le Chat 01/Le Chat 02].

Zusammenfassend lässt sich sagen, dass die Erstellung des Besprechungsprotokolls durch die LLM-Tools grundsätzlich gut funktioniert. Besonders ChatGPT zeigt wenig Schwankungen in seinen generierten Ausgaben und enthält fast immer die gleichen Punkte im Besprechungsprotokoll, formuliert diese nur unterschiedlich. Daher wird ChatGPT mit "Sehr gut" bewertet. Die anderen beiden Tools haben damit

etwas mehr Schwierigkeiten, doch wenn man das Besprechungsprotokoll immer wieder neu generieren lässt, kommt irgendwann ein anständig erstelltes Protokoll zustande. Da die Zusammenfassung des Gesprächs auf die wesentlichen Punkte die Hauptaufgabe eines Besprechungsprotokolls ist, erhalten Le Chat und Gemini lediglich die Bewertung "Ausreichend". Dass Le Chat und Gemini nicht alle Anforderungen in das Protokoll aufnehmen, kann im Nachhinein das Projekt verzögern und die Kosten in die Höhe treiben. Geminis erstellte Protokolle mit etwa 30 Punkten verfehlen hingegen komplett den Sinn eines Besprechungsprotokolls.

4.2 Projekthandbuch

Das Projekthandbuch wurde in zwei Phasen erstellt. Zunächst wurde die Einleitung verfasst, die den Zweck des Dokuments, die Redaktion und den Verteiler erläutert. Hierfür wurde der folgende Prompt im selben Chat eingegeben, in dem auch das Besprechungsprotokoll erstellt wurde:

Erstell mir für dieses Projekt die Einleitung für das Projekthandbuch. Die Einleitung besteht aus einem Abschnitt für den Zweck des Dokuments, einen Abschnitt zur Redaktion, in welchem geklärt wird, wer für das Dokument verantwortlich ist, und einen Abschnitt zu dem Verteiler, also wer bei Änderungen zu informieren ist. Verantwortlich für das Dokument ist der Projektleiter und über Änderungen wird das gesamte Team informiert. Dazu wird eine entsprechende Nachricht in den Discord Channel geschrieben.

Prompt 4.2: Prompt Einleitung Projekthandbuch

Die Einleitung erforderte spezifische Informationen, die im Prompt angegeben werden mussten. In diesem Projekt sind dies die Verantwortlichkeit des Projektleiters und der Verteiler über den Discord-Channel. Die Ergebnisse der Tools sind im Dokument "02_Projekthandbuch.pdf" zu finden. Alle Abschnitte wurden von den Tools gut erstellt, ohne Auffälligkeiten zu zeigen.

Der zweite Teil betraf das Kapitel "Projektdefinition" mit den Abschnitten "Vorgeschichte" und "Inhaltliche Kurzdarstellung". Dafür wurde nach der Ausgabe der Einleitung im Chat folgender Prompt eingegeben:

Erstelle mir nun das Kapitel "Projektdefinition" des Projekthandbuches. Der erste Abschnitt soll die Vorgeschichte des Projekts beschreiben und anschließend soll ein Abschnitt eine inhaltliche Kurzdarstellung geben.

Prompt 4.3: Prompt Projektdefinition Projekthandbuch

Diese zweite Eingabe benötigte keine weiteren Informationen, da diese im Gespräch bereits vorgegeben waren und die Tools darauf zugreifen konnten. Auch diese Abschnitte sind im Dokument "02_Projekt-

handbuch.pdf” zu finden. Die Tools konnten die erforderlichen Informationen aus dem Gespräch gut extrahieren und die Abschnitte wurden gut erstellt.

Bei der Erstellung der Einleitung zeigte sich, dass eine genaue Beschreibung der benötigten Abschnitte entscheidend war. Ohne diese klare Vorgabe tendierten die Tools dazu, eigene Strukturen und Inhalte zu erstellen, die nicht den Anforderungen entsprachen [Kap. Le Chat 03].

Abschließend lässt sich feststellen, dass die Abschnitte von allen drei Tools gut erstellt wurden und daher alle drei mit der Note “Sehr gut” bewertet werden.

4.3 Risikoliste

Auch die Risikoliste wurde in zwei Schritten erstellt. Zunächst wurde nur die Risikotabelle erstellt und im zweiten Schritt die Tabelle mit den Maßnahmen. Die Ergebnisse der Tools sind in den entsprechenden Excel Tabellen “03_Gemini_Risikoliste.xlsx”, “03_GPT_Risikoliste.xlsx” sowie “03_LeChat_Risikoliste.xlsx” zu finden.

Im folgenden ist der Prompt für die Risikotabelle wiedergegeben:

Erstell mir für diese Projekt eine Risikoliste. Diese soll aus mehreren Spalten bestehen: “ID” für eine eindeutige Identifikationsnummer, “Beschreibung” für eine ausführliche Beschreibung des Risikos und der Auswirkungen, “Datum” für den Zeitpunkt, wann das Risiko identifiziert wurde, “Autor” für die Person die das Risiko gemeldet hat, “Wahrscheinlichkeit (in%)” für einen Schätzwert der Eintrittswahrscheinlichkeit des Risikos, “Schaden (in €)” für eine Schätzung wie groß der Schaden ist, “Maß (in €)” was das Produkt aus Wahrscheinlichkeit und Schaden ist, “Risikoklasse” für eine Priorisierung der potentiellen Risiken wo zwischen Tolerierbar, Unerwünscht, Kritisch und Katastrophal unterschieden wird und “Status” wo zwischen aktiv, eingetreten und geschlossen unterschieden wird. Das Risiko ist Tolerierbar wenn das Risikomaß geringer als 0,1% des Projektvolumen ist, Unerwünscht wenn es größer als 0,1% ist, Kritisch wenn es größer als 1% ist und Katastrophal wenn es größer als 10% ist. In der Risikoliste sollen Team-, Technische-, Methodische-, Kunden-, Fachliche-, Produkt-, Management- und Planungsrisiken betrachtet werden. Diese sollen mit einer Leerzeile getrennt werden, in welcher die Oberbegriffe stehen. Das Projektvolumen beträgt 500000€.

Prompt 4.4: Prompt Risikotabelle

Dieser ist sehr umfangreich formuliert. Damit die Tabelle mit den richtigen Spalten erstellt wird, wurde im Prompt jede einzelne Spalte aufgezählt und beschrieben. Ebenfalls ist es wichtig, die möglichen Werte für die Risikoklasse zu definieren, da hier sonst immer “Hoch”, “Mittel” und “Niedrig” von den Tools verwendet wird. Es musste auch festgelegt werden, wann die einzelnen Risikoklassen auftreten, da die Zuweisung ansonsten recht schwammig ausfällt und schwerwiegende Risiken eine geringere Risikoklasse erhalten

als eher unwichtige Risiken. Außerdem sollte beschrieben werden, welche Arten (Team-, Technische-, Methodische-, Kunden-, Fachliche-, Produkt-, Management- und Planungsrisiken) von Risiken betrachtet werden sollen und dass diese Arten in einer Leerzeile stehen, welche die dazugehörigen Risiken von den anderen Arten trennt. Ansonsten kann es passieren, dass die Risiken durcheinander geschrieben werden und damit nicht den Arten zuzuordnen sind. Damit der Risikoschaden zwischen den Tools vergleichbar ist, musste das Projektvolumen definiert werden. Ansonsten wird auch dieses von Tools festgelegt und führt zu Ungenauigkeiten im Vergleich der einzelnen Ausgaben.

Grundsätzlich war die Erstellung der Risiken kein Problem. Schwierig war es eher, die gewünschte Formatierung zu erhalten, sowie eine richtige Zuweisung der restlichen Attribute. Besonders Gemini und Le Chat haben dabei Schwierigkeiten. Bei allen dreien Tools war auffällig, dass die Person, die in der Spalte "Autor" eingetragen wird, die Person ist, in dessen Tätigkeitsbereich das Risiko fällt. Zum Beispiel das Risiko, dass jemand krank wird, wodurch Verzögerungen entstehen, wurde von der Projektleiterin erstellt. Dabei werden auch Kunden Risiken vom Kunden erstellt, was falsch ist.

ChatGPT hat die Tabelle sehr gut erstellt. Lediglich die Zuweisung der Risikoklasse war falsch. Nach einem Hinweis diesbezüglich wurden diese jedoch, mit einer Ausnahme, richtig korrigiert [Tabelle 2 & 3 im Abschnitt 02_Risiken].

Ein Problem bei Gemini ist, dass das Risikomaß nicht richtig berechnet wird und auch die Risikoklassen nicht korrekt zugewiesen werden. Das Komma der Werte ist um eine Stelle zu weit links. Es muss alles einmal mit 10 multipliziert werden, damit die Werte stimmen. Ein noch größeres Problem ist, dass die Tabelle nicht in einem Zug erstellt werden kann. Gemini erstellt die Tabelle nur zu einem Teil und hört dann auf. [Tabelle 1 in 02_Risiken]. Auch wenn man Gemini fragt, ob die vollständige Tabelle generiert werden kann, wird die Erstellung vorzeitig abgebrochen. Man muss explizit nur nach den noch offenen Risikoarten fragen. Diese werden dann erstellt, jedoch soll man den Schaden und das Maß selbst eintragen und berechnen [Tabelle 2 in 02_Risiken]. Nach anschließender Frage, ob Gemini den Risikoschaden und das Schadensmaß festlegen kann, reagiert Gemini mit der Ausgabe, dass detaillierte Informationen über das Projekt und die potenziellen Risiken benötigt werden, um genaue Werte festlegen zu können.

Le Chat hatte ein ähnliches Problem wie bereits im Besprechungsprotokoll [Abschnitt 4.1], dass das Datum immer auf den 01.04.2023 gesetzt wird. Außerdem hatte Le Chat Probleme damit, die Tabelle in das gewünschte Format zu bringen, in dem die Risikoarten in einer Zeile stehen und die dazugehörigen Risiken darunter aufgelistet werden. Auch hier wurden die Risikoklassen falsch bestimmt und auch nach einem Hinweis wurden diese nicht korrigiert. Jedoch wurde dabei der Risikoschaden der einzelnen Risiken geändert. Auch hier ist die Zuweisung der Risikoklassen nicht richtig [02_Risiken].

Die Maßnahmentabelle wurde mit folgendem Prompt erstellt:

Erstelle mir nun eine dazu passende Maßnahmentabelle. Auch diese besteht aus mehreren Spalten: "Typ" beschreibt ob die Maßnahme das Risiko verhindert, lindert oder überträgt. In "Beschreibung" wird die Maßnahme beschrieben. Falls ein Risiko als nicht mehr relevant eingestuft wird, wird in der Spalte "Beschreibung" eine Begründung eingetragen und die Maßnahme auf "beendet" gesetzt. "Trigger" ist das Ereignis, das den Start der Maßnahme veranlasst, falls diese nicht sofort eingeleitet werden soll. "Verantwortlicher" ist die zuständige Person für die Durchführung der Maßnahme. "Status" unterscheidet zwischen geplant, aktiv und beendet. Anschließend gibt es jeweils eine Spalte für "Restwahrscheinlichkeit (in %)", "Restschaden (in €)", "Restmaß (in €)" und "Restklasse" was die geschätzte Wahrscheinlichkeit, geschätzter Schaden, Maß und Klasse des Restrisikos, nach Durchführung der Maßnahme entsprechen. Für jedes Risiko sollen zwei Maßnahmen erstellt werden. Dazu wird über die zwei Maßnahmen eine Zeile mit der ID von dem Risikon beschrieben, auf die sich die Maßnahmen beziehen.

Prompt 4.5: Prompt Maßnahmentabelle

Der Prompt für die Maßnahmen ist ähnlich wie der für die Risikotabelle. Es wird jede Spalte der Tabelle einmal beschrieben, damit das Format der Tabelle mit den Vorgaben übereinstimmt. Anschließend wird angegeben, dass die ID des dazugehörigen Risikos mit in die Tabelle übernommen werden soll, damit man die Maßnahmen den Risiken zuordnen kann. Die Probleme sind hier ähnlich zu denen bei der Risikotabelle, jedoch erstellen alle drei Tools grundsätzlich anständige Maßnahmen für die Risiken. Bei allen dreien fühlt sich allerdings die Zuweisung der Attribute zu den Maßnahmen repetitiv an. Häufig wechseln sich die auswählbaren Parameter ab und auch die Wahrscheinlichkeiten und der Restschaden sind meistens immer wieder die gleichen Zahlen.

ChatGPT erstellt auch die Maßnahmentabelle zunächst sehr gut. Lediglich die Zuweisung der Restklasse stimmt nicht überein [Tabelle 1 in 03_Maßnahmen]. Bei der zweiten Ausgabe ist der Restschaden falsch dargestellt. Dieser müsste mit dem Faktor 1000 multipliziert werden damit das Restmaß dazu passt [Tabelle 2 in 03_Maßnahmen].

Bei Gemini wird für die Begründung, warum ein Risiko als nicht mehr relevant eingestuft wird, eine eigene Spalte erstellt. Auch wenn man die Beschreibung dafür im Prompt ändert, wird die Spalte erstellt. Außerdem ist die Formatierung bei Gemini manchmal nicht so gut, da bei jeder 2. Zeile alle Daten ab der Spalte "Begründung" um eine Stelle nach links verschoben werden. Lässt man sich die Antwort neu generieren oder auch wenn man Gemini sagt, dass die Formatierung korrigiert werden soll, bleibt das Problem bestehen. Ein weiteres Problem ist, dass die Spalte "Trigger" nicht gefüllt wird und auch das Problem, dass die Tabelle nicht vollständig generiert wird sondern mitten drin aufhört, tritt wieder auf. Teilweise ist das Restmaß falsch berechnet und auch die Zuweisung der Restklasse ist nicht immer richtig [03_Maßnahmen].

Besonders aufgefallen bei Le Chat ist, dass die Attribute für die Maßnahmen sehr repetitiv festgelegt wurden. Die Restwahrscheinlichkeit beträgt für jede Maßnahme 5% und auch der Restschaden ist häufig

für die Maßnahmen für eine Risikoart gleich. Auch hier ist die Restklasse nicht entsprechend zugewiesen worden sondern ist, bis auf bei der ersten Maßnahme, auf den Wert “Unerwünscht” festgelegt. Auch die Festlegung des Typs der Maßnahmen wirkt sehr wiederholend, da hier immer zwischen “verhindern” und “lindern” gewechselt wird [03_Maßnahmen].

Zusammenfassend lässt sich sagen, dass die Erstellung von Risiken und dazugehörigen Maßnahmen gut funktioniert, allerdings die Festlegung der zusätzlichen Attribute nicht den Vorgaben entspricht und häufig zu Fehlern führt. Ebenso die richtige Formatierung der Tabelle ist häufig etwas schwierig, lässt sich aber in den meisten Fällen korrigieren. Da ChatGPT deutlich weniger Probleme bei der Erstellung hatte und auch die Zuweisung der Attribute, abgesehen von der Risikoklasse, und die Formatierung gut funktioniert hat, wird es mit “Gut” bewertet.

Le Chat hatte etwas mehr Probleme bei der Bewältigung der Aufgabe. Bei der Risikotabelle war die Formatierung insofern kein Problem, da man die einzelnen Spalten einfach in eine eigene Tabelle übernehmen konnte und die Risikoarten selber einfügt. Erst bei der Maßnahmentabelle hat Le Chat Probleme mit der Zuweisung der Attribute bekommen. Daher wird Le Chat mit “Befriedigend” bewertet. Gemini hat am meisten Probleme gemacht. Alleine schon die Tatsache, dass Gemini mitten in der Generierung diese als fertig betrachtet und dadurch die Tabellen unvollständig erstellt werden, sorgt dafür, dass Gemini für solche Erstellungen eher ungeeignet ist. Die anderen Probleme, wie die fehlerhafte Formatierung und dass die Attribute falsch berechnet und festgelegt werden, kommen noch hinzu. Dies sorgt dafür, dass Gemini hier die Bewertung “Mangelhaft” erhält.

Wenn man sich allerdings nur die Erstellung von Risiken mit dazugehörigen Maßnahmen anschaut, erhalten alle drei Tools die Bewertung “Sehr gut”. Die erstellten Risiken und Maßnahmen waren zwar alle eher welche, die man vermutlich als erstes benennt und auch projektunabhängig sind, wurden aber dennoch sehr gut erstellt. Auch bei Gemini werden einzelne Risiken mit dazugehörigen Maßnahmen in einem Rutsch erstellt, weshalb man dieses Problem bei der Erstellung der Risiken und Maßnahmen nicht ankreiden kann.

4.4 Anforderungsspezifikation

Für die Anforderungsspezifikation wurde für jeden Inhalt ein eigener Prompt erstellt. Daher wird im folgenden die Erstellung der Einleitung, der Systemarchitektur, des Fachliche Datenmodells, des Anwendungsfalldiagramms, der detaillierten Beschreibung von Anwendungsfällen mit Aktivitätsdiagrammen, der Benutzungsschnittstelle mit der Dialognavigation, der detaillierten Ausarbeitung von Dialogen und der nichtfunktionalen Anforderungen einzeln beschrieben und anschließend bewertet. Die Ergebnisse der Tools für die einzelnen Eingaben sind in der Datei 04_Anforderungsspezifikation.pdf auf dem beiliegenden USB-Stick zu finden.

Einleitung

Für die Einleitung wurde der folgende Prompt verwendet:

Erstell mir für die Anforderungsspezifikation eine Einleitung. Diese soll kurz, mit 2-3 Sätzen, in das Dokument einführen. Anschließend soll ein Abschnitt zum "Zweck und Umfang des Dokuments" kommen, welcher eine Beschreibung der Notwendigkeit des Systems für den Kunden sowie eine Kurzbeschreibung der Funktionalität und der Nachbarsysteme beinhaltet. Danach wichtige "Begriffe und Abkürzungen" erklärt werden. Zum Schluss kommt ein Abschnitt "Verweise auf sonstige Ressourcen", wo auf die Spielregeln verwiesen wird.

Prompt 4.6: Prompt Einleitung Anforderungsspezifikation

Bei der Erstellung des Prompts musste darauf geachtet werden, den Abschnitt "Zweck und Umfang des Dokuments" genauer zu beschreiben, da hier ansonsten die generierte Ausgabe sehr ausschweifend und unspezifisch erstellt wird. Grundsätzlich hat die Erstellung gut geklappt.

Bei ChatGPT ist lediglich auffällig, dass beim Abschnitt "Zweck und Umfang des Dokuments" nicht der Zweck der Anwendung beschrieben wurde, sondern der Zweck des Dokuments. Die Kurzbeschreibung der Funktionalität wurde dafür gut erstellt. Bei der Kurzbeschreibung der Nachbarsysteme wurden die einzelnen Teile der Anwendung erläutert, also das Spielsystem, das Kommunikationsprotokoll für RMI und die Benutzeroberfläche, und keine anderweitigen Systeme, da die Anwendung alleinstehend ist [1 Einleitung (GPT 01), S. 6].

Die Erstellung der Texte mit Gemini und Le Chat hat sehr gut funktioniert [2 Einleitung (Gemini 01), S. 7 & 3 Einleitung (Le Chat 01), S. 8].

Bewertet wird ChatGPT mit "Gut", da hier der Zweck des Dokuments beschrieben wurde, obwohl im Prompt explizit nach einer Beschreibung der Notwendigkeit des Systems für den Kunden gefragt wurde. Le Chat und Gemini werden mit "Sehr gut" benotet, da es hier keine Auffälligkeiten gibt.

Systemarchitektur

Die Systemarchitektur wurde mit folgenden Prompt erstellt:

Erstell mir für die Anforderungsspezifikation die Systemarchitektur. Diese beinhaltet einen groben Überblick über die erwartete Systemarchitektur und Einordnung des Systems in die Systemlandschaft des Kunden. Falls Schnittstellen zu Nachbarsystemen bestehen, dann müssen diese abgebildet werden.

Prompt 4.7: Prompt Systemarchitektur

Auch hier muss beschrieben werden, was genau erstellt werden soll. Wichtig ist, dass man in denselben Chat schreibt, in dem man die Dokumente vorher schon erstellt hat. Dadurch wissen die Tools, was zu erstellen ist.

Bei ChatGPT wurde sogar ein Diagramm in den Chat "gezeichnet". Die Beschreibung und das Diagramm sind jedoch für die Systemarchitektur schon viel zu genau. Es hätte gereicht, den Client, den Server und die Datenbank darzustellen und diese richtig miteinander zu verbinden. Bei der ersten Erstellung wurden die Verbindungen zwischen Client und Server und Server und Datenbank unidirektional dargestellt [4 Systemübersicht (GPT 01), S. 9]. Nach einer weiteren Eingabe, ob das so gewollt ist, wurde dies jedoch korrigiert und die Verbindung wurde bidirektional eingezeichnet [5 Systemübersicht (GPT 02), S. 12].

Gemini hatte das gleiche Problem wie ChatGPT, dass hier die Komponenten bereits zu genau beschrieben werden. Die Komponenten ähneln teilweise denen, die bei dem Komponentendiagramm erstellt werden müssen, als auch den Klassen aus dem fachlichen Datenmodell. Des Weiteren hat Gemini zunächst keine Datenbank erstellt [6 Systemübersicht (Gemini 01), S. 15]. Auf Nachfrage, ob eine Datenbank hinzugefügt werden sollte, wurde zunächst nur erklärt, woran man dies ausmachen kann. Erst mit einer expliziten Eingabe, die Datenbank hinzuzufügen, wird dies gemacht. Bei der Beschreibung der Schnittstellen wurde beschrieben, dass zwischen Client und Server RMI verwendet wird. Allerdings wird bei der Datenbank nur von der Datenbank-Schnittstelle geschrieben und es wurde nicht richtig definiert, mit was und wie die Datenbank verbunden ist [7 Systemübersicht (Gemini 02), S. 17].

Nach der Eingabe kam bei Le Chat eine kurze textuelle Beschreibung der Client- und Server-Anwendung sowie der Datenbank. Außerdem wurde beschrieben, dass Schnittstellen zu Nachbarsystemen ebenfalls mit aufgenommen werden sollen. Diese Beschreibungen haben soweit gut gepasst. Auf Nachfrage, wie denn dann die Abbildung aussehen soll, wurde beschrieben, dass der Client, der Server und die Datenbank jeweils eigenständige Blöcke sein sollen. Dies ist soweit richtig. Es wurde jedoch nicht genau beschrieben, wie die Beziehungen zwischen den Komponenten aussehen. Daher wurde nochmal eine Eingabe getätigt, wie denn die Beziehung zwischen den Komponenten aussieht. Hier wurde richtig dargestellt, dass der Client und der Server kommunizieren und der Server mit der Datenbank, jedoch nicht der Client mit der Datenbank. Die restliche Beschreibung, wie diese Komponenten kommunizieren, ist wieder etwas zu ausführlich. Die Beschreibung reicht jedoch, um die Skizze selber zu erstellen, was kein negatives Kriterium ist, da bei den kostenfreien Versionen nicht davon ausgegangen wird, dass diese das können [8 Systemübersicht (Le Chat 01), S. 19].

Zusammenfassend lässt sich hier sagen, dass alle drei Tools die Systemarchitektur etwas zu genau erzeugen. Jedoch lassen sich bei ChatGPT und Le Chat die richtigen Inhalte aus der Ausgabe kopieren und die zu detaillierten Beschreibungen können einfach gelöscht werden, weshalb die Tools mit der Note "Gut" bewertet werden. Geminis Ausgabe kann allerdings aufgrund der fehlenden Beschreibung, wie die Datenbank mit den anderen Komponenten verbunden ist, nicht einfach kopiert und gekürzt werden. Daher wird Gemini hier mit "Befriedigend" bewertet.

Fachliches Datenmodell

Das erste größere und anspruchsvollere Dokument ist das Fachliche Datenmodell. Dieses wurde mit folgendem Prompt erstellt:

Erstell mir nun für die Anforderungsspezifikation das Fachliche Datenmodell. Dieses soll mit Hilfe eines UML Klassendiagrammes mit ergänzenden Beschreibungen bzw. Einschränkungen spezifiziert werden. Das Modell soll alle Entitätstypen mit deren Eigenschaften, Beziehungen und Einschränkungen besitzen.

Prompt 4.8: Prompt Fachliches Datenmodell

Dabei muss explizit erwähnt werden, dass ein UML-Klassendiagramm gewünscht ist, damit die Ausgabe das richtige Format besitzt. Außerdem sollte erwähnt werden, dass ergänzende Beschreibungen mit erstellt werden sollen, damit diese Vorgabe auch direkt erfüllt ist und die Tools ihr Diagramm näher erläutern.

Bei ChatGPT war direkt auffällig, dass für jede Klasse eine ID erstellt wurde. Dies führt dazu, dass die Anwendung sehr datenbanklastig ist und grundsätzlich ist dies bei manchen Klassen auch einfach nicht nötig. Zum Beispiel beim User könnte man je nach Login-Daten den Namen oder die E-Mail-Adresse eindeutig machen und damit den User identifizieren. Des Weiteren haben die Beziehungen nicht immer gepasst, zum Beispiel die 1:1 Beziehung zwischen "Spielfigur" und "Feld". Es muss natürlich nicht auf jedem Feld eine Spielfigur stehen, sondern auf einem Feld kann (0..1) eine Spielfigur stehen. Dieses Problem zieht sich auch durch die gesamten Versuche, das fachliche Datenmodell zu korrigieren. Häufig passiert es auch, dass wenn ein anderer Fehler korrigiert wird, eine richtige Beziehung zu etwas Falschem geändert wird. Bei der erstellten Skizze waren ebenfalls die Beziehungen falsch eingezeichnet. Außerdem zeigten beide Pfeile zwischen zwei Klassen in die gleiche Richtung [9 Fachliches Datenmodell (GPT 01), S. 21]. Auf Nachfrage, ob die Beziehung nicht in beide Richtungen zeigen sollten, wurde gesagt, dass das stimmt und die Beziehungen bidirektional sein sollten, jedoch wurde an der Zeichnung nichts geändert [10 Fachliches Datenmodell (GPT 02), S. 25]. Des Weiteren ist aufgefallen, dass bei der Klasse "Feld" ein Attribut gefehlt hat, das beschreibt, ob das Feld belegt ist oder nicht. Außerdem wurde in einem String in "Spiel" das Spielbrett gespeichert. Auf Nachfrage, wie dies implementiert werden soll, wurde geantwortet, dass dies nicht die beste Lösung ist, sondern dass das Spielbrett dynamisch in der Anwendung generiert und angezeigt werden sollte [18 Fachliches Datenmodell (GPT 10), S. 65]. Weitere Probleme hatte ChatGPT mit der Speicherung der Farbe und der Positionen der Spielfiguren. Manchmal wurden diese Attribute in "Spieler", manchmal in "Spielfigur" und teilweise auch in beiden gespeichert [16 Fachliches Datenmodell (GPT 08), S. 54 & 17 Fachliches Datenmodell (GPT 09), S. 60]. Damit es später einfacher zu realisieren ist, an mehreren Spielen teilnehmen zu können, wurde die Eingabe, dass es mehr Sinn macht, den "Spieler" in "User" und "Spieler" aufzuteilen, getätigt. ChatGPT hat dies dann genau wie beschrieben umgesetzt [13 Fachliches Datenmodell (GPT 05), S. 37]. Des Weiteren wurde der Würfel vergessen, welcher jedoch nach Hinweis diesbezüglich in "Spiel" hinzugefügt wurde [19

Fachliches Datenmodell (GPT 11), S. 79 & 20 Fachliches Datenmodell (GPT 12), S. 76]. Nach der letzten Ausgabe des fachlichen Datenmodells wurde, nach Aufforderung, ein PlantUML-Diagramm erstellt [20 Fachliches Datenmodell (GPT 12), S.76]. In diesem wurde an den Assoziationen Leseunterstützung angefügt. Diese Leseunterstützung ist jedoch nur von oben nach unten hilfreich. Zum Beispiel “Feld steht auf Spielfigur” ist keine hilfreiche Beschriftung. Des Weiteren sind alle Festlegungen der Vielfachheit auf der falschen Seite. Als Beispiel muss an der Assoziation zwischen “Spieler” und “Spielfigur” die “1” auf der Seite von “Spieler” sein und die “1..4” auf der Seite von “Spielfigur”. Des Weiteren würden Pfeile an den Assoziationen helfen, die Flussrichtung zu verdeutlichen.

Die zu dem Dokument gehörigen ergänzenden Beschreibungen und Einschränkungen wurden erst auf Nachfrage erstellt, doch ab da an passten diese ganz gut. Dass beim Format direkt die Länge der Strings begrenzt wurde, zeigt, dass ChatGPT hier auch sicherheitstechnische Aspekte mit einbezieht [ab 14 Fachliches Datenmodell (GPT 06), S. 42].

Geminis Ausgabe war ähnlich zu der von ChatGPT. Auch hier wurden für alle Klassen eine ID erstellt und es wurden bereits die Referenzen in den Klassen auf andere Klassen eingefügt. Dies gehört jedoch nicht in das fachliche Datenmodell. Ebenfalls auffällig ist, dass lediglich die IDs eindeutig sind. Dadurch müsste man sich mit seiner ID anmelden, da es passieren kann, dass es mehrere “Spieler” mit dem gleichen Namen und Passwort gibt. Grundsätzlich ist die Implementierung etwas eigenartig, da jeder “Spieler” einen “Spielstand” besitzt. Hier werden allerdings auch allgemeine Informationen wie ein Spielbrett, ein Spielstand und das Datum und die Uhrzeit gespeichert. Es würde mehr Sinn machen, diese Eigenschaften in “Spiel” nur einmal für alle “Spielstände” zu speichern. Ebenfalls eher schlecht umgesetzt sind die Beschreibungen der Assoziationen zwischen den Klassen. Hier werden entweder “... zu 1” oder “... zu *” Beziehungen beschrieben und teilweise sind die Assoziationen nur unidirektional. Des Weiteren haben manche Klassen, die für das Spiel benötigt werden, Assoziationen mit “Spieler”. Da aber die Spielinformationen eines Spielers über “Spielstand” laufen, benötigt man diese nicht, sondern es reichen Assoziationen mit “Spielstand”. Dann lässt sich automatisch der dazugehörige “Spieler” damit verbinden. Andersherum verhält es sich mit der Assoziation “Nimmt an vielen Spielen teil” von der Klasse “Spieler”. Da “Spieler” mehrere “Spielstände” haben kann und diese zu einem “Spiel” gehören, ist diese Assoziation unnötig. Die Eigenschaften haben auch keine, wie bei einem UML-Klassendiagramm üblich, Festlegung des Typs der Eigenschaften [21 Fachliches Datenmodell (Gemini 01), S. 83]. Als zweite Eingabe wurde angewiesen, die Klasse “Spieler” zu einer Klasse “User” und einer Klasse “Spieler” aufzuteilen, um den Login-Prozess zu vereinfachen. Da allerdings bereits bei der ersten Eingabe die Klasse “Spieler” lediglich die allgemeinen Benutzerinformationen umfasst und die Spielinformationen in den entsprechenden Klassen gespeichert werden, war die Eingabe unnötig, hätte aber trotzdem funktionieren sollen. Gemini hatte jedoch mit der Umsetzung Probleme. Zum einen wurde die Klasse “Spielfigur” entfernt und die Klasse “Spieler” hat die Eigenschaft “Position” erhalten [22 Fachliches Datenmodell (Gemini 02), S. 87]. Auf Nachfrage, warum die “Spielfigur”-Klasse entfernt wurde, wurde diese einzeln neu ausgegeben und nicht in das fachliche Datenmodell wieder eingebunden [23 Fachliches Datenmodell (Gemini 03), S. 90]. Mit der Eingabe, nochmal das gesamte UML-Klassendiagramm auszugeben, wurde die Spielfigur einfach wieder eingesetzt, allerdings hat dann “Spieler” auf einmal eine “Spielfigur” und einem “Spielstand” sind ebenfalls nochmal mehrere

“Spielfiguren” zugeordnet. Ebenfalls gehörte nur noch ein “Spielstand” zu einem “Spiel” und zu einem “Spieler”. Hier wurde also nur noch ein Einzelspieler-Modus implementiert [24 Fachliches Datenmodell (Gemini 04), S. 91]. Auf Nachfrage, ob nur ein Einzelspieler-Spiel erstellt wird, wurde zugestimmt und mehrere Lösungsvorschläge erstellt und auf eine weitere Nachfrage, einen davon in das Diagramm einzubinden, wurde dies umgesetzt. Allerdings sind immer noch zahlreiche Fehler in dem Diagramm vorhanden [25 Fachliches Datenmodell (Gemini 05), S. 94 & 26 Fachliches Datenmodell (Gemini 06), S. 96]. Bereits die erste Erstellung des fachlichen Datenmodells und auch die späteren Ausgaben betrachtet keine die Anforderung, dass ein Chat im Spiel vorhanden sein soll und dass es ein Punktesystem geben soll. Dies ist sehr kritisch, da die nachträgliche Einfügung von Anforderungen kosten- und zeitintensiv ist.

Auch Le Chat hatte zu Beginn nur eine Klasse “Spieler”. Für eine Anwendung, wo ein Spieler an mehreren Spielen teilnehmen können soll, macht es jedoch Sinn, diese Klasse auf zwei Klassen, “User” und “Player”, aufzuteilen. Des Weiteren wurden bei Le Chat, wie es eigentlich in einem UML-Klassendiagramm gehört, die Typen der Attribute nicht festgelegt. Außerdem stellt sich die Frage, warum eine eindeutige ID benötigt wird, wenn die E-Mail-Adresse eines Spielers eindeutig sein muss. Daneben wurde auch die Anforderung des Kunden, dass ein Punktesystem eingebunden werden soll, ignoriert, da es kein Attribut zum Speichern der Punktzahl gibt. Die Bezeichnung der Beziehungen zwischen den Klassen ist in der textuellen Beschreibung für die Beziehungen eher an einem ER-Modell orientiert und nicht an einem UML-Klassendiagramm. Die genaue Bezeichnung für die Vielfachheit muss aus der Beschreibung selbst erstellt werden. Diese ist jedoch auch teilweise fehlerhaft. Zum Beispiel macht es keinen Sinn, dass ein Player an einem oder mehreren Spielen teilnehmen kann, da die Klasse “Player” die Eigenschaften eines Spielers im Spiel beschreibt. Diese können schlecht in mehreren Spielen gleich sein. Auch einfachere Sachen wie die Zuweisung der Spielfiguren waren ein Problem. Zum Beispiel wurde bei der Beschreibung der Beziehung zwischen Player und Spielfigur definiert, dass ein Player eine oder mehrere Spielfiguren besitzen kann. Dies entspricht aber nicht den normalen Regeln von Mensch ärgere dich nicht, wo jeder Spieler immer genau vier Spielfiguren besitzt. Daneben wurde auch hier der “Player”-Klasse ein Attribut für die Position auf dem Spielbrett festgelegt. Nach der Eingabe [29 Fachliches Datenmodell (Le Chat 03), S. 103] wurden genau diese Beziehungen richtig geändert, jedoch waren immer noch Fehler enthalten, wie zum Beispiel, dass ein Spiel eine oder mehrere Spielfiguren enthalten kann. Anschließend wurde gefragt, ob der Würfel auch in das fachliche Datenmodell mit aufgenommen werden sollte, woraufhin sowohl eine eigene Klasse “Dice” als auch ein Attribut “dice: Dice” in der “Game”-Klasse erstellt wurde. Anschließend wurde gefragt, ob das fachliche Datenmodell in PlantUML erstellt werden kann [30 Fachliches Datenmodell (Le Chat 04), S. 105]. Hier fällt besonders auf, dass die Verbindungen zwischen den Klassen alles Kompositionen sind, was aber soweit auch Sinn ergibt, da wenn z.B. die “Game”-Klasse gelöscht wird, auch die dazugehörigen Klassen gelöscht werden können. Diese treten nur innerhalb der “Game”-Klasse auf. Außerdem fällt auf, dass hier bereits die Primär- und Fremdschlüssel dargestellt werden. Dies ist für das fachliche Datenmodell allerdings zu spezifisch. Ein weiterer Fehler ist die Beziehung zwischen “Game” und “Figure”. Hier wird festgelegt, dass ein Spiel aus 16 Figuren besteht. Allerdings gibt es auch ein Attribut “max_players” und es wurde bereits festgelegt, dass ein Spiel aus 1..4 Spielern besteht. Daher kann es auch sein, falls nicht immer mit KI-Gegnern auf 4 Spieler aufgefüllt wird, dass weniger als 16 Figuren auf dem Spielfeld sind. Hier lässt

sich allerdings eh kritisch hinterfragen, ob die Komposition zwischen “Game” und “Figure” überhaupt notwendig ist, da durch die Verbindung zwischen “Figure” und “Player” und der zwischen “Game” und “Player” die Figuren sowieso genau einem Spiel zugeordnet sind. Le Chat erstellt das Format und die Beschreibung der Attribute auch erst auf Nachfrage [30 Fachliches Datenmodell (Le Chat 04), S. 107]. Jedoch hält sich die Ausgabe in Grenzen. Bei dem Format wird lediglich der Typ der Attribute beschrieben und es gibt keine Eingrenzung der Größe und der Länge der Werte.

Alle drei Tools schaffen es, ein grobes Grundgerüst für das fachliche Datenmodell zu erstellen, jedoch war bei allen dreien auffällig, dass für jede Klasse eine eindeutige ID erstellt wurde und auch schon Referenzen in die Klassen eingefügt wurden. Das Hauptproblem bei ChatGPT und Gemini stellt die richtige Darstellung der Assoziationen zwischen den Klassen dar. Besonders die Beschreibung der Vielfachheit stellt die beiden Tools vor Schwierigkeiten. Dies funktioniert bei Le Chat etwas besser, aber immer noch nicht fehlerfrei. Dafür hatte Le Chat mit mehreren kleineren generellen Problemen zu kämpfen. Es ist außerdem auffällig, dass ab einem gewissen Punkt die Erstellung des Prompts und das Analysieren der neuen Antwort länger dauert als die Fehler selbstständig anzupassen. Daher ist es nicht allzu schlimm, wenn ein Attribut irgendwo doppelt vorkommt oder eine Assoziation falsch beschriftet ist. Deshalb und aufgrund der Komplexität dieses Diagrammes erhalten sowohl ChatGPT als auch Le Chat die Bewertung “Gut”. Das Grundgerüst kann hier ohne Probleme übernommen werden und anschließend an die eigenen Bedürfnisse und Anforderungen angepasst werden. Gemini erhält aufgrund der Nichtbeachtung der Anforderungen die Note “Ausreichend”.

Anwendungsfalldiagramm

Erstell mir nun für die Anforderungsspezifikation ein Anwendungsfalldiagramm. Dieses soll alle Anwendungsfälle beinhalten und den Rollen zuweisen, die für die Applikation benötigt werden.

Prompt 4.9: Prompt Anwendungsfalldiagramm

Mit diesem Prompt wurde das Anwendungsfalldiagramm erstellt. Der Prompt ist sehr einfach gehalten und beschreibt lediglich kurz den Inhalt des Diagramms.

ChatGPTs erste Ausgabe enthielt zwei Rollen, den “Spieler” und den “User”. Der “User” hatte den Anwendungsfall “Spiel erstellen” [31 Anwendungsfälle (GPT 01), 109]. Dadurch kam die Frage auf, ob man als Spielersteller besondere Rechte haben sollte. ChatGPT hat daraufhin die neue Rolle “Gastgeber” hinzugefügt. Fraglich ist allerdings die Einzeichnung in das PlantUML-Diagramm. Demzufolge erbt der “Gastgeber” von “Spieler” und dieser von “User”. Das ist allerdings nur teilweise richtig. Der “Gastgeber” muss von “Spieler” erben, da auch der “Gastgeber” die zum Spielen benötigten Anwendungsfälle benötigt. Allerdings sollte “Spieler” nicht von “User” erben, da die Anwendungsfälle, auf die der “User” zugreifen kann, während eines Spiels nicht benötigt werden. Des Weiteren hatte “Spieler” die Anwendungsfälle “Zug machen” und “Spielfigur bewegen”. Dadurch war zunächst unklar, welche Tätigkeiten “Zug machen” beinhaltet [32 Anwendungsfälle (GPT 02), S. 112]. Auf Nachfrage antwortete ChatGPT allerdings, dass

zu "Zug machen" alle Tätigkeiten gehören, die man während eines Zuges durchführen muss (Würfeln, Auswählen und Bewegen der Spielfigur, Schlagen gegnerischer Figuren und Beenden des Zuges). Daher ist der Anwendungsfall "Spielfigur bewegen" unnötig. Auf einen Hinweis diesbezüglich entfernte ChatGPT den Anwendungsfall aus dem Diagramm [33 Anwendungsfälle (GPT 03), S. 115]. Zuletzt konnte der Anwendungsfall "Chat nutzen" mit der Rolle "User" durchgeführt werden. Dies entspricht allerdings nicht den Anforderungen, da nur während eines Spiels ein Chat vorhanden sein soll. Auch hier wurde der Anwendungsfall nach einem Hinweis von ChatGPT wieder entfernt [34 Anwendungsfälle (GPT 04), S. 118]. Grundsätzlich würde das Anwendungsfalldiagramm so passen, allerdings muss man sich die Frage stellen, ob man noch eine Rolle für administrative Zwecke benötigt, welche direkt Daten, wie zum Beispiel User, aus der Datenbank löschen und hinzufügen kann. Positiv herausheben sollte man, dass ChatGPT bei jeder Ausgabe automatisch Code für ein Diagramm bei PlantUML erzeugt hat.

Das Anwendungsfalldiagramm von Gemini erstellt bei der ersten Ausgabe lediglich Anwendungsfälle, die zum Spielen benötigt werden. Die Rolle "Spielverwaltung" beschreibt dabei die Aufgaben, die auf dem Server passieren. Die Rollen "Spieler" und "Spielverwaltung" sind laut Gemini miteinander verwoben. Die Rolle "Spieler" beschreibt dabei, was der Spieler auf der Client-Seite macht. Die Rolle "Spielverwaltung" beschreibt, was anschließend auf der Server-Seite passiert. Dies ist allerdings nicht das, was ein Anwendungsfalldiagramm darstellen soll [35 Anwendungsfälle (Gemini 01), S. 121]. Daher wurde mit derselben Eingabe eine neue Antwort generiert. Hier fällt zum einen auf, dass die Rolle "Gast", obwohl in der Beschreibung der Rolle definiert wird, dass diese zum Beispiel keine Spiele spielen kann, den Anwendungsfall "Spiel suchen" und "Spiel beitreten" durchführen kann. Zum anderen kann die Rolle "Spieler" sowohl generelle Benutzerinteraktionen durchführen als auch Spielaktionen [36 Anwendungsfälle (Gemini 02), S. 124]. Es macht Sinn, dies voneinander zu trennen, was Gemini auch richtig macht. Ein weiterer negativer Punkt ist der Anwendungsfall "Benutzer verwalten" vom "Administrator". Dieser Anwendungsfall ist zu grob definiert und sollte in einzelne Anwendungsfälle, also "Benutzer erstellen", "Benutzer bearbeiten" und "Benutzer löschen", unterteilt werden. Auch Gemini erstellt für die Aktionen im Spiel einen großen Anwendungsfall "Spiel spielen", wo gewürfelt wird, die Spielfigur bewegt wird und Spielaktionen ausgeführt werden. Negativ fällt außerdem auf, dass auch hier die Anforderungen, dass es einen Chat und ein Punktesystem geben soll, nicht eingebunden wurden [37 Anwendungsfälle (Gemini 03), S. 126].

Eine erste Ausgabe mit direkt vier Rollen erstellt dagegen Le Chat. Hier wird eine eigene Rolle "Gast" erstellt, welche ein Benutzer erhält, der noch nicht eingeloggt ist, und es wird direkt eine Rolle "Admin" angegeben. Ein Unterschied zu ChatGPT und Gemini ist hier, dass, wenn ein Spieler am Zug ist, jede Aktion ein eigener Anwendungsfall ist. Dabei ist zu beachten, dass "Figur setzen" und "Figur schlagen" nicht getrennt werden dürfen, da diese Aktionen zusammengehören. Wenn eine Figur gesetzt wird, muss automatisch überprüft werden, ob sich auf dem Feld bereits eine Figur befindet; wenn ja, muss diese automatisch geschlagen werden. Außerdem sind die Anwendungsfälle "Benutzer verwalten" und "Profil bearbeiten" zu ungenau definiert. Hier muss man für jede Aktion einen eigenen Anwendungsfall erstellen [38 Anwendungsfälle (Le Chat 01), S. 129]. Diese beiden Punkte wurden Le Chat als Eingabe gegeben und anschließend anständig korrigiert. Auf Nachfrage wurde hier auch Code für ein PlantUML-Diagramm erzeugt. Dieses ist jedoch sehr unübersichtlich, da für jeden Anwendungsfall ein "Männchen" erstellt

wird und diese einfach untereinander aufgelistet werden. Die Versuche, ihn davon abzubringen und das Diagramm ähnlich wie ChatGPT zu erstellen, waren alle vergebens [39 Anwendungsfälle (Le Chat 02), S. 130].

Zusammenfassend lässt sich sagen, dass alle drei Tools das Anwendungsfalldiagramm schon ganz gut beschreiben. Ein paar Ungenauigkeiten gibt es dennoch, und aus diesem Grund werden sowohl ChatGPT als auch Le Chat mit "Gut" bewertet. Dass Le Chat das PlantUML-Diagramm nicht übersichtlich erstellen kann, fließt hier nicht negativ in die Bewertung mit ein, da das Erstellen des Diagramms keine Anforderung war. Es reicht, wenn die Diagramme korrekt beschrieben werden, sodass man diese selbst erstellen kann. Dies ist bei der Beschreibung von Le Chat gegeben. Gemini hingegen wird auch hier nur mit "Ausreichend" benotet, da die Anforderungen nicht alle umgesetzt werden. Dies ist ein sehr wichtiger Punkt bei der Erstellung der Anforderungsspezifikation. Nachträglich eingefügte Anforderungen sind kosten- und zeitintensiv und sollten unbedingt vermieden werden.

Anwendungsfälle

Nun geht es um die Erstellung der detaillierten Beschreibung für die einzelnen Anwendungsfälle. Dazu wurden zwei Anwendungsfälle ausgesucht: einmal "Anmelden" als leichteren Anwendungsfall und "Zug machen" (ChatGPT), "Figur setzen oder schlagen" (Le Chat) bzw. "Spiel spielen" (Gemini) als komplexeren Anwendungsfall. Dazu wurde mit zwei Prompts gearbeitet. Für den Anwendungsfall "Anmelden" wurde folgende Eingabe benutzt:

Erstell mir nun für die Anforderungsspezifikation, basierend auf dem Anwendungsfalldiagramm, eine detaillierte Beschreibung für den Anwendungsfall "Anmelden". Dabei soll zunächst der Anwendungsfall in einem Satz beschrieben werden, was dieser macht. Anschließend soll der Akteur, der Auslöser, die Vorbedingung und die Häufigkeit beschrieben werden. Danach soll ein Aktivitätsdiagramm erstellt werden. Dabei werden die Aktivitäten dem Client oder dem Server zugewiesen, wo diese bearbeitet werden. In dem Diagramm sollen keine Dialogbeschreibung, wie Fenster mit Fehlermeldung anzeigen, beschrieben werden. Es sollen lediglich Dienste beschrieben werden, die der Server anbietet. Zuletzt werden die Aktivitäten aus dem Diagramm kurz beschrieben.

Prompt 4.10: Prompt AF Anmelden

Der Prompt beschreibt die Struktur des Dokuments mit den Inhalten. Außerdem wird der Aufbau des Aktivitätsdiagramms beschrieben und erklärt, dass keine Dialogbeschreibungen in das Diagramm gehören. Einige Punkte werden hier direkt ganz gut erstellt. Auffällig ist, dass bei der "Häufigkeit" keine grobe Schätzung angegeben wird, sondern Aussagen wie "Jedes Mal, wenn der Benutzer die Anwendung nutzen möchte." Dies ist keine sinnvolle Beschreibung der Häufigkeit.

Die erste Ausgabe wurde mit ChatGPT-4 erstellt [42 AF Anmelden (GPT 03), S. 138]. Dabei ist vor allem aufgefallen, dass die Beschreibung nicht zu dem PlantUML-Diagramm passt. Die Aktivitäten

sind anders benannt, und auch die Anzahl der Aktivitäten stimmt nicht überein. Außerdem wird sowohl in der Beschreibung als auch in dem Diagramm bereits viel zu genau beschrieben, was passiert. Zum Beispiel ist die Aktivität "Session erstellen" für die Beschreibung eines Anwendungsfalls viel zu genau. Dies beschreibt bereits das "Wie" der Umsetzung und gehört somit in das Architekturdokument. Aufgrund dessen, dass ChatGPT-4 hier zu viel erstellt hat, wurde die Nachricht mit ChatGPT-3.5 nochmal generiert [40 AF Anmelden (GPT 01), S. 133]. Man sieht direkt, dass deutlich weniger Aktivitäten und keine Beschreibung der Implementierung erstellt wurden. Die Aktivitäten passen zum Anwendungsfall und beschreiben diesen gut. Wenn man jedoch die Diagramme von ChatGPT-4 und ChatGPT-3.5 vergleicht, fallen einige Fehler auf, die beide machen. Zum einen wird die Aktivität, die überprüft wird, in den Entscheidungsknoten hineingeschrieben. Diese muss jedoch als eigene Aktivität oben darüber stehen. Des Weiteren sollten die Beschriftungen an den Pfeilen des Entscheidungsknotens in eckige Klammern gesetzt werden. Auch dass die Pfade, die aus dem Entscheidungsknoten entstehen, wieder zusammenführen, ist falsch. Bei dem Diagramm der 3.5er Version fehlen außerdem der Start- und Endknoten, und auch bei der 4er Version sollten zwei Endknoten vorhanden sein: einmal für "Anmeldung fehlgeschlagen" und einmal für "Anmeldung erfolgreich". Wenn man die 3.5er Version auf diese Hinweise allgemein hinweist [42 AF Anmelden (GPT 93), S. 139], führen zwar die beiden Pfade nach dem Entscheidungsknoten nicht mehr zusammen, jedoch endet der eine ohne Endknoten, und im Entscheidungsknoten steht immer noch die Aktivität. Wenn man genau sagt, was er mit welcher Aktivität machen soll, wird das Aktivitätsdiagramm komplett fehlerhaft mit unterbrochenen Verbindungen und einer alleinstehenden Aktivität erstellt [42 AF Anmelden (GPT 93), S. 139]. Gibt man den groben Hinweis der 4er Version [44 AF Anmelden (GPT 05), S. 143], setzt diese die Vorgaben richtig um. Lediglich die Dialoginteraktion "Benutzer klickt auf 'Anmelden'" ist noch vorhanden, welche jedoch nach einem Hinweis auch entfernt wird [45 AF Anmelden (GPT 06), S. 145]. Zuletzt fällt noch auf, dass im Entscheidungsknoten nur "ja" steht. Hier ist die Frage, ob man in PlantUML einen leeren Entscheidungsknoten ohne Text erstellen kann. Ähnlich verhält es sich an den Endknoten. Diese können vermutlich auch nicht beschriftet werden, weshalb direkt davor noch Aktivitäten mit der Beschreibung erstellt werden. Des Weiteren verwendet ChatGPT den Benutzernamen bzw. die E-Mail-Adresse und ein Passwort als Anmeldedaten. Im fachlichen Datenmodell [20 Fachliches Datenmodell (GPT 12), S. 76] ist jedoch keiner dieser Attribute als einzigartig definiert, weshalb sich aus diesen Daten nicht eindeutig auf einen User schließen lässt. Es könnten mehrere User mit diesen Daten existieren.

Der von Gemini erstellte Anwendungsfall weist ebenfalls einige Probleme auf. Zum einen definiert Gemini folgende Vorbedingung: Die Anmeldeinformationen des Users (Benutzername und Passwort) müssen korrekt sein. Dies ist jedoch nicht richtig, da man auch ohne Konto oder mit falschen Daten versuchen kann, sich anzumelden. Dazu ist auch die Anmeldung mit Benutzername und Passwort nicht richtig, da keines dieser Attribute im fachlichen Datenmodell als einzigartig definiert worden ist [26 Fachliches Datenmodell (Gemini 06), S. 96]. Dadurch kann man mit diesen beiden Daten nicht eindeutig einen Benutzer identifizieren. Die einzelnen Aktivitäten passen soweit, lediglich die Aktivitäten "Sende Anmeldebestätigung" und "Speichere Anmeldesitzung" sind etwas zu genau für die Anwendungsfallbeschreibung. Zu der ersten Ausgabe wird auch ein Mermaid-Code generiert, welcher allerdings zu einem Syntaxfehler führt [46 AF Anmelden (Gemini 01), S. 147]. Daher wurde gesagt, dass

doch ein PlantUML-Code für das Aktivitätsdiagramm erstellt werden soll. Dies wurde auch gemacht, allerdings zeigte das Aktivitätsdiagramm die Form eines Sequenzdiagramms. Dazu wurde geschrieben, dass der PlantUML-Code “folgendes Bild” beschreibt, wo dann ein Link zu einem Aktivitätsdiagramm aus dem Internet verlinkt wurde. Dieses passte jedoch nicht zu dem PlantUML-Code. Es ist zwar ein UML-Aktivitätsdiagramm für einen Anmeldeprozess, allerdings ist dieses nicht in dem Format, wie es sein soll, und entspricht außerdem nicht der vorherigen Beschreibung von Gemini [47 AF Anmelden (Gemini 02), S. 150].

Le Chat erstellt wieder nur auf Nachfrage ein PlantUML-Diagramm, welches vom Format her auch nicht ganz passend ist, da hier Client und Server übereinander stehen und nicht nebeneinander. Außerdem gibt es hier dieselben Probleme wie bei ChatGPT: Im Entscheidungsknoten steht die Aktivität, und die Pfade danach werden wieder zusammengeführt. Die textuelle Beschreibung der ersten Ausgabe [01 Le Chat Anmelden] ist ganz gut, und man könnte daraus, mit ein bisschen Mitdenken, ein richtiges Aktivitätsdiagramm erstellen [48 AF Anmelden (Le Chat 01), S. 153]. Bei der zweiten Ausgabe, die mit derselben Eingabe erstellt worden ist, fällt auf, dass trotz des Hinweises, dass keine Dialogeigenschaften beschrieben werden sollen, die erste Aktivität “Anmeldeformular anzeigen” genau das tut. Des Weiteren hat auch Le Chat für die Anmeldedaten kein Attribut gewählt, das laut dem fachlichen Datenmodell einzigartig ist [30 Fachliches Datenmodell (Le Chat 04), S. 105]. Daher tritt hier dasselbe Problem wie bei Gemini und ChatGPT auf, dass nicht eindeutig ein Benutzer bestimmt werden kann [49 AF Anmelden (Le Chat 02), S. 155].

Der Anwendungsfall “Zug machen” oder “Spielfigur setzen oder schlagen” wurde direkt nach dem Anwendungsfall “Anmelden” erstellt. Dadurch wurde beim Prompt die Beschreibung, was erstellt wurde, weggelassen. Stattdessen wurde beschrieben, dass, wenn in einer Aktivität ein Wert gespeichert werden soll, auf den Wert im fachlichen Datenmodell referenziert werden muss. Außerdem wurde der Name des Anwendungsfalls bei den Tools entsprechend angepasst. Für ChatGPT sieht der Prompt zum Beispiel wie folgt aus:

Erstell mir nun eine detaillierte Beschreibung für den Anwendungsfall “Zug machen”. Wenn bei einer Aktivität ein Wert gespeichert wird, sollte in der Beschreibung der Aktivität beschrieben werden, um welchen Wert es sich aus dem Fachlichen Datenmodell handelt. Als Beispiel sollte der gewürfelte Wert in Spiel.wuerfel gespeichert werden.

Prompt 4.11: Prompt AF Zug machen

Alle drei Tools haben dabei nicht geprüft, ob mindestens eine Figur das Startfeld bereits verlassen hat oder nicht. Wenn nicht, müsste mit maximal drei Würfeln probiert werden, eine sechs zu würfeln.

Die Ausgabe von ChatGPT [50 AF Zug Machen (GPT 01), S. 158] scheint zunächst sehr überwältigend, doch beim näheren Betrachten ist diese eigentlich schon ziemlich gut. Das Format des PlantUML-Diagramms zeigt dieselben Fehler auf wie bereits beim Anwendungsfall “Anmelden”, lassen sich aber später alle korrigieren [52 AF Zug Machen (GPT 03), S. 166]. Eine Aktivität, die etwas erschreckend

klings, ist Aktivität 6 "Berechne mögliche Züge". Grundsätzlich ist dies nicht gerade eine effiziente Herangehensweise, was bei einem Mensch-ärgere-dich-nicht-Spiel allerdings nicht ganz so schlimm ist, da sich hier die möglichen Züge auf maximal vier beschränken. Es fällt auch auf, dass das Würfeln auf der Server-Seite stattfindet. Dies ist an sich kein gutes Spielerlebnis, wenn man als Spieler nicht würfelt und auch das Ergebnis nicht sieht. Das Würfeln sollte am besten ein eigener Anwendungsfall sein, da das Würfelergebnis wieder auf die Client-Seite gebracht werden muss und dies innerhalb eines Anwendungsfalls schwierig zu implementieren ist. Auch die Verweise auf das fachliche Datenmodell halten sich mit zwei Verweisen in Grenzen, und die Klasse "Spielzug" wird überhaupt nicht betrachtet. Als Hilfe wurde daher nochmal eine Eingabe mit dem fachlichen Datenmodell [20 Fachliches Datenmodell (GPT 12), S. 76] und der Anweisung, sich dieses zu merken, erstellt. Anschließend wurde der Prompt 4.11 erneut eingegeben. Diese Ausgabe [51 AF Zug Machen (GPT 02), S. 162] schafft es etwas besser, die Verweise auf das fachliche Datenmodell zu beschreiben. Lediglich "Spiel.status" und "Feld.belegt" werden nicht erwähnt. Das Würfeln wird auf die "Client"-Seite geschoben. Jedoch ist dieser Ablauf, wie bereits beschrieben, schwierig zu implementieren, und man sollte das Würfeln als eigenen Anwendungsfall beschreiben. Es entfällt auch die Aktivität, die überprüft, ob ein Spieler an der Reihe ist. Ebenfalls negativ ist, dass die "Client"-Seite im Aktivitätsdiagramm zu "Spieler" umbenannt wurde. Dies wird jedoch nach einem Hinweis wieder zurück geändert [53 AF Zug Machen (GPT 04), S. 169]. Ein Fehler ist auch, dass die Position der Spielfigur zweimal gespeichert wird: einmal in Aktivität 7 und einmal in Aktivität 9. Was nach einem Hinweis aber auch korrigiert wird [54 AF Zug Machen (GPT 05), S. 173].

Da das zuletzt erstellte fachliche Datenmodell voller Fehler war und nicht verwendet werden kann, wurde hier die Frage nach den Verweisen auf das fachliche Datenmodell weggelassen [55 AF Zug Machen (Gemini 01), S. 176]. Die erste Ausgabe konnte scheinbar mit dem Anwendungsfall "Spiel spielen" nichts anfangen und wusste nicht so richtig, was dieser laut dem Anwendungsfalldiagramm beinhaltet. Hier wurden Aktivitäten wie "Wähle Spieltyp und Spieloptionen:" erstellt, wo zum Beispiel gewählt wird, ob Einzelspieler oder Mehrspieler gespielt werden soll. Daher wurde die Eingabe nochmal gemacht, mit dem Zusatz, dass die Beschreibung des Anwendungsfalls aus dem Anwendungsfalldiagramm mit in den Prompt eingefügt wurde [56 AF Zug Machen (Gemini 02), S. 179]. In der Ausgabe werden Aktivitäten wie "Wähle Aktion:" und "Verarbeite Aktion:" beschrieben. Daher beschreibt dieses Aktivitätsdiagramm keinen genauen Ablauf, sondern beschreibt grob, was allgemein passiert. Dies ist allerdings für die Beschreibung eines Anwendungsfalls falsch. Was man hier allerdings positiv anmerken muss, ist, dass überprüft wird, ob das Spiel beendet ist oder nicht. Dennoch wurde auch hier nochmal eine neue Antwort generiert mit dem gleichen Prompt [57 AF Zug Machen (Gemini 03), S. 182]. Diese beschreibt auch brauchbar den Ablauf des Zuges im Aktivitätsdiagramm. Dennoch ist auch hier das Problem, dass der Spieler erst würfelt, das Ergebnis an den Server geschickt wird und der Spieler anschließend eine Spielfigur auswählt. Dies ist in der Implementierung schwer umzusetzen, weshalb das Würfeln als eigener Anwendungsfall betrachtet werden sollte. Ebenfalls negativ fällt auf, dass hier nicht mehr geprüft wird, ob das Spiel beendet ist, wie bei der vorherigen Antwort.

Bei Le Chat [58 AF Zug Machen (Le Chat 01), S. 185] war zu Beginn die Beschreibung etwas kurz und es gab, außer "Spiel.wuerfel", was explizit im Prompt genannt wurde, keine Referenz auf das fachliche Datenmodell. Erst mit der nächsten Eingabe [59 AF Zug Machen (Le Chat 02), S. 187] wurden die

Beschreibungen etwas ausführlicher und es gab Referenzen auf das fachliche Datenmodell. Diese Referenzen waren jedoch falsch und entsprachen nicht dem zuvor erstellten fachlichen Datenmodell. Hier wurde nämlich häufig auf die Position der Figuren und der Felder über “.x” und “.y” Werte zugegriffen. Grundsätzlich ist das natürlich eine gute Art und Weise, die Position leichter zu definieren. Jedoch wurde im fachlichen Datenmodell über einfache Int-Werte die Position des Feldes gespeichert. Außerdem ist die Beschreibung der Aktivitäten nicht entsprechend den Regeln von “Mensch ärgere dich nicht”. In der Aktivität “Figur setzen oder schlagen” wird beschrieben, dass eine geschlagene Figur aus der Liste der Figuren des Spielers entfernt wird. Dies ist natürlich nicht richtig, sondern die Figur sollte einfach auf ein Startfeld der entsprechenden Farbe bzw. des entsprechenden Spielers gesetzt werden. Außerdem wurde Würfeln als eine Aktivität hier aufgeführt. Im Anwendungsfalldiagramm wurde allerdings für Würfeln ein eigener Anwendungsfall erstellt, weshalb die Aktivität hier unnötig und falsch ist. Nach einem Hinweis diesbezüglich hat Le Chat ausgegeben, dass das stimmt und die Aktivität entfernt werden muss [60 AF Zug Machen (Le Chat 03), S. 189]. Anschließend wurde, auf Nachfrage, eine Beschreibung des Aktivitätsdiagramms erstellt [61 AF Zug Machen (Le Chat 04), S. 190]. Dabei wurde einmal die Aktivität “Figur setzen oder schlagen” in zwei Aktivitäten aufgeteilt: einmal “Figur setzen” und “Figur schlagen”, wobei mit einem Entscheidungsknoten entschieden wird, in welche Aktivität gegangen wird. Die Beschreibung der Flusslinien ist sehr sporadisch und sagt sogar eigentlich aus, dass erst die Aktivitäten miteinander verbunden sind und diese dann erst mit den Entscheidungsknoten und den Endpunkten verbunden werden. Dies sollte aber nicht so sein, da die Entscheidungsknoten zwischen den Aktivitäten stehen müssen. Dazu wird beschrieben, dass es nur zwei Endpunkte gibt: für den Fall, dass erfolgreich die Figur gesetzt oder geschlagen wird und das Verweigern des Zugs. Dies wäre natürlich möglich, um dem Spieler aber eine bessere Rückmeldung zu geben, macht es Sinn, für jeden Fehler, der auftauchen kann, einen Endzustand zu erstellen. Ebenfalls negativ auffällig ist, dass bei den ersten beiden Aktivitäten “Figur auswählen” und “Zielfeld auswählen”, jeweils die Information an den Server übermittelt wird. Besser und einfacher wäre es aber, die Infos zusammen in einem Zug an den Server zu senden. Damit würde es reichen, eine Funktion zu erstellen und an den Server zu senden. Der “Trick”, das zuletzt erstellte fachliche Datenmodell als Eingabe zu nutzen, hat nicht wie bei ChatGPT funktioniert. Wenn man Le Chat nach dem aktuellen fachlichen Datenmodell fragt, auf welches er sich referenziert, wird auch ein völlig anderes Datenmodell ausgegeben [62 AF Zug Machen (Le Chat 05), S. 192].

Es lässt sich zusammenfassen, dass alle drei Tools es ganz gut geschafft haben, den leichten Anwendungsfall “Anmelden” zu erstellen. Alle drei Tools hatten jedoch Probleme mit Dialoginteraktionen oder zu genauen Beschreibungen der Implementierung. Außerdem haben alle drei es nicht umgesetzt, richtige Daten für die Anmeldung zu nutzen, mit denen ein Nutzer eindeutig zugeordnet werden kann. Bei der Beschreibung des Anwendungsfalls “Zug machen” hatten allerdings alle drei Tools größere Probleme. Die Verweise auf das fachliche Datenmodell haben bei ChatGPT und Le Chat zu Beginn kaum funktioniert, was allerdings zumindest bei ChatGPT nach der Eingabe mit der Beschreibung des fachlichen Datenmodells größtenteils geklappt hat. Generell hatte ChatGPT die wenigsten Probleme von den dreien und hat das Aktivitätsdiagramm entsprechend den vorherigen Diagrammen gut erstellt. Daher wird ChatGPT hier mit “Gut” benotet, da dennoch ein paar kleine Fehler vorhanden waren. Le

Chat konnte nicht dazu gebracht werden, richtige Verweise auf das fachliche Datenmodell zu geben, und die Beschreibung passte auch nicht zum Anwendungsfalldiagramm. Allerdings ist die grundsätzliche Beschreibung der Aktivitäten als grober Ablauf gut erstellt worden, und man könnte mit dieser leicht das Aktivitätsdiagramm selbstständig erstellen. Daher wird Le Chat gerade noch so mit der Note “Befriedigend” benotet. Bei Gemini wurden die Verweise auf das fachliche Datenmodell weggelassen, da dieses falsch und nicht verwendbar war. Zusätzlich hatte Gemini Probleme bei der Erstellung der Aktivitäten, und die Ausgabe musste zweimal neu generiert werden, da Gemini nicht wusste, was zum Anwendungsfall “Spiel spielen” dazugehört. Da die dritte Ausgabe dann einigermaßen brauchbar war und man aus der Beschreibung selbstständig das Aktivitätsdiagramm erstellen kann, wird Gemini mit “Ausreichend” benotet.

Benutzungsschnittstelle

Erstell mir nun für die Anforderungsspezifikation die Benutzungsschnittstelle. Diese zeigt, von welchem Dialog aus man zu anderen navigieren kann. Es sollen alle Dialoge enthalten sein, mit all deren Verbindungen zueinander.

Prompt 4.12: Prompt Benutzungsschnittstelle

Mit diesem Prompt wird die Benutzungsschnittstelle beschrieben, die zeigt, wie man zwischen den Dialogen navigieren kann.

Bei ChatGPT wurde die Eingabe zweimal getätigt.

Die erste Eingabe [63 Benutzungsschnittstelle (GPT 01), S. 195] enthielt im Chat ein “gezeichnetes” Diagramm. Dort waren jedoch sowohl die Verbindungen zwischen den Dialogen als auch die Einzeichnung der Navigation etwas schwierig, da hier nur in eine Richtung navigiert wird. Außerdem passte dieses auch nicht richtig zu der textuellen Beschreibung, die mit generiert wurde. Bei der Beschreibung ist aufgefallen, dass ein “Passwort-zurücksetzen-Dialog” enthalten ist. Allerdings ist im Anwendungsfalldiagramm kein Anwendungsfall für das Passwort zurücksetzen erstellt worden, weshalb man entweder den Dialog aus dem Diagramm löschen oder den Anwendungsfall “Passwort zurücksetzen” nachträglich erstellen muss. Ähnlich verhält es sich mit den Anwendungsfällen “Punkte anzeigen” und dem Dialog “Anzeige der Spieler-Liste”. Außerdem ist fraglich, ob ein noch nicht angemeldeter Nutzer die Buttons “Spiel erstellen” und “Spiel beitreten” sehen soll. Hier wäre es angebrachter, einen eigenen Dialog für nicht-angemeldete Nutzer zu erstellen und zusätzlich ein Hauptmenü.

Nach der zweiten Eingabe [64 Benutzungsschnittstelle (GPT 02), S. 198] erhielt man ebenfalls eine textuelle Beschreibung, von welchem Dialog man wohin navigieren kann, einen PlantUML-Code für ein UML-Diagramm sowie eine Beschreibung der Dialoge. Das PlantUML-Diagramm hatte zunächst einen Syntaxfehler, welcher jedoch nach einem Hinweis korrigiert wurde. Jedoch ist das Diagramm recht unübersichtlich, da zum einen die Pfeile nur in eine Richtung zeigen und zum anderen die Struktur

etwas durcheinander ist. Die Beschreibung ist dafür, abgesehen von einem Punkt, sehr gut. Wenn man einem Spiel beigetreten ist oder ein Spiel gestartet hat, gelangt man zu dem Dialog "Spielbildschirm". Wenn man dort direkt das Spielfeld sieht, was bei einem Hauptbildschirm während des Spiels eigentlich der Fall ist, muss man den Anwendungsfall "Spiel anzeigen" aus dem Anwendungsfalldiagramm wieder entfernen.

Die von Gemini erstellte textuelle Beschreibung der Benutzungsschnittstelle sieht für sich ganz gut aus [65 Benutzungsschnittstelle (Gemini 01), S. 202]. Allerdings passt diese nicht mit dem Anwendungsfalldiagramm zusammen. In den Dialogen wurden Buttons bzw. Dialognavigationen beschrieben, für die es keine Anwendungsfälle gibt, zum Beispiel "Pausieren des Spiels". Des Weiteren startet man scheinbar direkt im "Hauptmenü" und es gibt keinen Startdialog für nicht angemeldete User, in welchem die Anwendungsfälle "Anmelden" und "Registrieren" umgesetzt werden. Nach der Eingabe, dass noch ein Startdialog erstellt werden soll, wurde dies aber nachträglich erstellt [66 Benutzungsschnittstelle (Gemini 02), S. 205]. Hier gibt es dann allerdings die Möglichkeit, als Gast einem Spiel beizutreten. Dies passt weder zu dem fachlichen Datenmodell noch zu dem Anwendungsfalldiagramm.

Le Chat hat bei der Erstellung der Benutzungsschnittstelle probiert, diese in die Ausgabe mit zu "zeichnen" [67 Benutzungsschnittstelle (Le Chat 01), S. 207]. Die Abbildung war jedoch völlig durcheinander und unübersichtlich. Die Verbindung zwischen den Dialogen ist unverständlich und ergeben keinen Sinn. Auch mit Eingaben, dass die Verbindung doch überarbeitet werden soll, wurde dies nicht besser. Dafür kann man sagen, dass die erstellten Dialoge soweit zu den Anwendungsfällen passen und die meisten enthalten sind. Jedoch sind auch einige Dialoge erstellt worden, für die man eigentlich keinen eigenen Dialog erstellen muss. Zum Beispiel "Spiel verlassen". Da würde ein Button reichen, mit dem man zurück zum Hauptmenü gelangt.

Auch die textuelle Beschreibung, welche nicht zu der Zeichnung passt, ist nicht vollständig. Bei der Beschreibung des Verlaufs werden Anwendungsfälle, wie zum Beispiel "Registrieren", vergessen. Außerdem wird nur die Möglichkeit, ein Spiel zu erstellen oder beizutreten, beschrieben. In dem Anwendungsfalldiagramm gibt es aber noch den Anwendungsfall "Spiel suchen". Bei der Beschreibung fehlt auch, wie der Admin zu seinen verfügbaren Dialogen navigieren kann.

Keines der drei Tools schafft es, die Benutzungsschnittstelle passend zu dem Anwendungsfalldiagramm zu erstellen. Es werden Dialoge für Anwendungsfälle erstellt, die vorher nicht beschrieben wurden, und für manche Anwendungsfälle, für die es einen Dialog geben sollte, gibt es überhaupt keinen Dialog. Bei Gemini und Le Chat ist dieses Problem deutlich stärker aufgetreten als bei ChatGPT, weshalb Gemini und Le Chat nur mit "Befriedigend" bewertet werden. Das Grundgerüst wird einigermaßen brauchbar erstellt und man kann die überflüssigen Dialoge kürzen bzw. die Beschreibung um die restlichen Elemente erweitern. ChatGPT wird hier mit "Gut" bewertet, da das Problem hier nicht so häufig auftritt.

Dialog

Für die Erstellung der Dialoge wurden ebenfalls zwei recht einfache Prompts verwendet, die lediglich beschreiben, wie das Dokument aufgebaut sein soll. Einmal für den Anwendungsfall "Anmelden" und

einmal für den Dialog “Spielbildschirm”.

Erstell mir nun für die Anforderungsspezifikation einen passenden Dialog für den Anwendungsfall “Anmelden”. Dabei wird zunächst beschrieben, wann der Dialog geöffnet wird und wozu er dient. Anschließend folgt eine Skizze des Dialoges. Zuletzt werden die einzelnen Elemente des Dialogs beschrieben und definiert, was dort eingetragen wird bzw. was beim drücken passiert.

Prompt 4.13: Prompt Dialog Anmelden

ChatGPT hat in der Ausgabe wieder eine “gezeichnete” Skizze erstellt [68 Dialog Anmelden (GPT 01), S. 209]. Bei der ersten Ausgabe des “Anmelden”-Dialogs wurde noch ein Button für “Passwort vergessen?” erstellt. Grundsätzlich macht dies natürlich Sinn, hier einzufügen, allerdings wurde im Anwendungsfalldiagramm kein Anwendungsfall dafür erstellt. Positiv auffällig ist, dass die erste Ausgabe in dem Chat erstellt wurde, wo GPT-3.5 in dem Anwendungsfall “Anmelden” festgelegt hat, dass sich der User mit einem Benutzernamen und einem Passwort anmeldet [40 AF Anmelden (GPT 01), S. 133] und die zweite Ausgabe [69 Dialog Anmelden (GPT 02), S. 211] in dem Chat erstellt wurde, wo GPT-4o sich vorher beim Anwendungsfall für eine E-Mail und ein Passwort entschieden hat [41 AF Anmelden (GPT 02), S. 135]. Dies passt mit den Dialogen überein. Bemängeln kann man dabei, dass bei der Beschreibung der Elemente nicht auf das Fachliche Datenmodell verwiesen wurde. Nach einer groben Eingabe, dass doch auf die Werte aus dem Fachlichen Datenmodell referenziert werden soll, wurde dies jedoch richtig gemacht [70 Dialog Anmelden (GPT 03), S. 213].

Bei der Erstellung des Dialogs “Spielbildschirm” [73 Dialog Spielbildschirm (GPT 01), S. 219] wurde auch wieder eine textuelle Beschreibung und eine Skizze erstellt. Die Skizze umfasst allerdings nur einen Vermerk in eckigen Klammern, wo das Spielbrett graphisch dargestellt werden soll. Außerdem wurde auch hier nochmal die Eingabe gemacht, dass auf das Fachliche Datenmodell referenziert werden soll [74 Dialog Spielbildschirm (GPT 02), S. 221]. Interessant ist, dass ein extra Button für das Würfeln erstellt wurde und einer, um einen Zug zu machen. Bei dieser Implementierung ist es naheliegender, dass für beide Buttons ein eigener Anwendungsfall gestartet wird. Bei Nachfrage, zu beschreiben, wenn ein Anwendungsfall gestartet wird, gibt ChatGPT aus, dass Würfeln ein eigener Anwendungsfall ist [75 Dialog Spielbildschirm (GPT 03), S. 224]. Die restlichen Verweise auf das Fachliche Datenmodell und die Beschreibung, welcher Anwendungsfall gestartet wird, passen soweit.

Gemini erstellt in der Beschreibung der Buttons Verweise auf andere Dialoge und Anwendungsfälle [71 Dialog Anmelden (Gemini 01), S. 215]. Negativ fällt dafür auf, dass in der Beschreibung des Dialoges auch das Verhalten beschrieben wird. Hier fügt Gemini unter anderem eine Beschreibung ein, was passiert, wenn man auf “Anmelden” klickt. Diese Beschreibung entspricht der eines Sequenzdiagrammes von einem Anwendungsfall und ist hier, in der Beschreibung eines Dialoges, falsch. Die Benutzerdaten entsprechen dem Anwendungsfalldiagramm, allerdings ist auch hier das Problem vorhanden, dass im Fachlichen Datenmodell keiner der Anmeldedaten als einzigartig beschrieben wurde.

Ähnlich verhält es sich bei dem Dialog für den Spielbildschirm [76 Dialog Spielbildschirm (Gemini 01), S. 227]. Auch hier werden im Abschnitt “Verhalten” eher die Anwendungsfälle beschrieben. Genauso

wie bei ChatGPT gibt es auch hier einen Button für das Würfeln und die getrennte Möglichkeit, einen Zug zu machen. Dies indiziert, dass für diese beiden Aktionen getrennte Anwendungsfälle erstellt werden sollten. In dem Dialog werden außerdem die möglichen Züge für den Spieler angezeigt. Dies passt allerdings nicht zu den Beschreibungen der Anwendungsfälle, da dies nirgends aufgeführt wird. Grundsätzlich kann man hier allerdings sagen, dass die Beschreibungen der Anwendungsfälle hier deutlich besser sind als bei den Anwendungsfällen selbst. Hier wird auch wieder ein Chat eingebunden. Ob das allerdings zufällig ist oder weil sich Gemini wieder an die Anforderung erinnert hat, ist fraglich. Das Punktesystem wird weiterhin nicht betrachtet.

Beim "Anmelden"-Dialog hat Le Chat wieder nur eine textuelle Beschreibung erstellt [72 Dialog Anmelden (Le Chat 01), S.217]. Diese ist soweit passend. Für das Anmelden hat sich Le Chat auf die E-Mail-Adresse und ein Passwort festgelegt. Im Fachlichen Datenmodell wurde allerdings nur gesagt, dass die ID eindeutig ist, das heißt, dass entweder die E-Mail-Adresse ebenfalls eindeutig gemacht werden muss oder sich diese nicht fürs Anmelden eignet. Ansonsten könnten zwei Benutzer mit derselben E-Mail-Adresse und demselben Passwort nicht unterschieden werden. Außerdem fehlt ein Button, um zum "Registrieren"-Dialog zu kommen, welcher zumindest laut dem Anwendungsfalldiagramm vorhanden sein sollte. Passen würde es jedoch zur Benutzungsschnittstelle, da hier kein Dialog fürs Registrieren eingebunden wurde.

Bei der Beschreibung des Dialogs "Spielbrett" werden die Inhalte gut beschrieben [77 Dialog Spielbildschirm (Le Chat 01), S. 230]. Es fehlt lediglich ein Button oder Ähnliches, um den Spielstand anzuzeigen. Dies ist laut dem Anwendungsfalldiagramm ein eigener Anwendungsfall, wobei jedoch die Frage ist, was dieser beschreiben soll oder ob dieser durch das dauerhafte Anzeigen des Spielbrettes hinfällig geworden ist. Bei Nachfrage, ob Le Chat beschreiben kann, wann ein Anwendungsfall gestartet wird, gibt dieser eine zu dem Dialog passende Beschreibung aus [78 Dialog Spielbildschirm (Le Chat 02), S. 232]. Auch hier fehlen allerdings die Anwendungsfälle "Spielstand anzeigen" und "Spielergebnis anzeigen".

Alle drei Tools haben die Dialoge gut und passend erstellt. Es gab lediglich ein paar Inkonsistenzen zu den vorherigen Dokumenten. Die Anwendungsfälle wurden hier anders dargestellt als im Anwendungsfalldiagramm und bei der detaillierten Beschreibung der einzelnen Anwendungsfälle. Allerdings waren die Lösungen, die bei den Dialogen ausgegeben wurden, eigentlich besser als die vorherigen. Dadurch und weil die Dialoge größtenteils so übernommen werden können und nur die überflüssigen Teile selbstständig gestrichen werden müssen, werden alle drei Tools hier mit "Gut" bewertet. Dass die Konsistenz hier nicht vollständig gegeben ist, wird dadurch ausgeglichen, dass die größten Probleme in den vorherigen Dokumenten, wie zum Beispiel die Trennung der Anwendungsfälle "Würfeln" und "Zug machen" sowie die Erstellung der IDs jeder Klasse und das Fehlen eines einzigartigen Benutzernamens oder ähnlichem, hier richtig geändert werden. Dadurch könnte man an dieser Stelle sagen, dass man die vorherigen Dokumente nochmal anpasst.

Nichtfunktionale Anforderungen

Erstell mir nun für die Anforderungsspezifikation die nichtfunktionalen Anforderungen und jeweils einen dazu gehörigen Test.

Prompt 4.14: Prompt nichtfunktionale Anforderungen

Dieser sehr einfach gehaltene Prompt wurde für die Erstellung der nichtfunktionalen Anforderungen verwendet. Es musste lediglich erwähnt werden, dass zu jeder Anforderung ein Test erstellt werden soll. Bei ChatGPT [79 Nichtfunktionale Anforderungen (GPT 01), S. 224], Gemini [80 Nichtfunktionale Anforderungen (Gemini 01), S. 235] und Le Chat [81 Nichtfunktionale Anforderungen (Le Chat 01), S. 237] werden gute Anforderungen erstellt, und die Tests passen auch sehr gut zu diesen. Hier werden alle drei Tools mit "Sehr gut" bewertet.

4.5 Architekturdokument

Da auch das Architekturdokument ein umfangreiches Dokument ist, werden hier ebenfalls die Ausgaben der technischen Infrastruktur, des Komponentenmodells und der dynamischen Beschreibung exemplarischer Anwendungsfälle im Komponentenmodell mithilfe eines Sequenzdiagramms jeweils einzeln beschrieben und bewertet. Die Ergebnisse der Tools für die einzelnen Eingaben sind in der Datei 05_Architekturdokument.pdf auf dem beiliegenden USB-Stick zu finden.

Technische Infrastruktur

Da die technische Infrastruktur auf die Systemübersicht aufbaut und alle drei Tools Schwierigkeiten hatten, sich an diese zu erinnern, wurde die textuelle Beschreibung der jeweiligen Tools ihnen nochmal als Eingabe mitgegeben. Der folgende Prompt wurde verwendet, um die Technische Infrastruktur zu erstellen:

Erstell mir nun, basierend auf folgender Systemarchitektur, ein Diagramm und/oder eine Textuelle Beschreibung der Technischen Infrastruktur. Diese listet die Hardware- und Produktanforderungen auf: Hardware, Betriebssysteme, Produkte, Nachbarsysteme. Dabei sollen die einzelnen Betriebssysteme mit Versionsnummer und die Verbindungen zwischen den Komponenten mit der Verbindungsart beschrieben werden: [Textuelle Beschreibung der Systemübersicht des jeweiligen Tools]

Prompt 4.15: Prompt Technische Infrastruktur

ChatGPT erstellt hier wieder eine Skizze in die Ausgabe mit [1 Technische Infrastruktur (GPT 01),

S. 4]. Diese ist auch ganz gut, lediglich die Datenbank ist etwas verrutscht. Negativ aufgefallen ist die Beschreibung der Verbindung von der KI-Komponente. Hier wird beschrieben, dass diese in die Spiel-Logik auf der Server-Seite integriert ist und daher keine externe Verbindung erforderlich ist. In der Beschreibung der Systemübersicht wird die Spiel-Logik allerdings nur auf der Client-Seite erstellt. Den Anwendungsfällen entsprechend ist die Spiel-Logik auf der Server-Seite allerdings korrekt, da zum Beispiel der Zug erst auf der Server-Seite validiert wird [04_Anforderungsspezifikation, S. 173 ff.]. Fragt man ChatGPT danach, wird ausgegeben, dass die Spiel-Logik beim Server hinzugefügt werden soll. Ansonsten wurde die technische Infrastruktur von ChatGPT gut erstellt.

Bei Gemini fällt direkt auf, dass sowohl eine Datenbank für den Client als auch für den Server erstellt wird [2 Technische Infrastruktur (Gemini 01), S. 8]. Diese ist eigentlich unnötig, da es vermutlich einfacher und effizienter ist, die Spielstände und die Benutzerdaten, wenn diese benötigt werden, vom Server an den Client zu senden, als diese die ganze Zeit doppelt gespeichert zu haben. Des Weiteren fällt auf, dass die Verbindung zwischen Client und Server nicht über RMI, sondern über HTTPS läuft. Aus diesem Grund wurde als zweite Eingabe auch hier die Beschreibung der Systemübersicht mitgegeben [3 Technische Infrastruktur (Gemini 02), S. 10]. Dies hat allerdings nichts geändert. Auf eine explizite Eingabe, dass die Kommunikation zwischen Client und Server über RMI läuft, wurde nur erklärt, was verändert werden muss, wie der Ablauf dann ist und was die Vor- und Nachteile davon sind [4 Technische Infrastruktur (Gemini 03), S. 13]. Es wurde allerdings nicht in die technische Infrastruktur übernommen.

Auch Le Chat versucht, eine Skizze in die Ausgabe mit einzufügen [5 Technische Infrastruktur (Le Chat 01), S. 15]. Dies klappt hier allerdings nicht so richtig, da die Komponenten mit den Verbindungen nicht richtig verbunden werden. Während ChatGPT die Datenbank extern, als eigene Komponente definiert, ist bei Le Chat diese Teil des Servers. Außerdem werden bei den Nachbarsystemen ein externes Authentifizierungssystem sowie ein Zahlungssystem erstellt, was allerdings nicht Teil der Anforderungen war. Daher sind diese hier falsch. Jedoch passt die Zuweisung der Versionen für die Betriebssysteme sowie der Produkte ganz gut.

Die Erstellung der technischen Infrastruktur funktioniert bei ChatGPT und Le Chat schon ganz gut, und es gibt nur ein paar Kleinigkeiten, die angepasst werden müssen. Daher werden diese mit "Gut" bewertet. Gemini hingegen hat ein paar mehr Probleme und beschreibt unter anderem nicht, dass die Kommunikation zwischen Client und Server über RMI läuft. Da dies allerdings eine Anforderung an das System ist, wird Gemini nur mit "Befriedigend" bewertet.

Komponentenmodell

- 1) Erstell mir nun für das Architekturdokument das Komponentenmodell. Das Komponentenmodell beschreibt die fachliche Struktur der Lösung auf einer groben Ebene mittels eines UML-Komponenten-Diagramms.
- 2) Beim Komponentenmodell handelt es sich nicht um die Technische Infrastruktur. Hier sollen die Komponenten und deren Schnittstellen erstellt werden, die später für die Programmierung verwendet werden.

Prompt 4.16: Prompt Komponentenmodell

Mithilfe des ersten Prompts wurde versucht, das Komponentenmodell zu erstellen. Dies hat jedoch bei keinem der Tools funktioniert. Alle drei Tools haben sich zu sehr an die technische Infrastruktur gehalten und sind von diesem Format nicht weggekommen. Dadurch wurde bei den drei Tools mit dem zweiten Prompt versucht, diese davon etwas wegzubringen.

Bei ChatGPT hat dies dann auch schon ganz gut funktioniert, allerdings wird sich trotzdem noch stark an der technischen Infrastruktur orientiert [7 Komponentenmodell (GPT 02), S. 19]. Außerdem sind keine Schnittstellen zwischen den Komponenten beschrieben worden. Es fällt auch auf, dass die "Game-Logic", welche die Spielregeln implementiert und lokale Spielzüge validiert, nur auf der Client-Seite ist, was nicht den Anwendungsfällen entspricht, wo festgelegt wurde, dass die Validierung der Spielzüge auf der Server-Seite geschieht. Nach einer Eingabe, in der ChatGPT dies erklärt wird, wird die "Game-Logic" auf die Server-Seite geschoben [8 Komponentenmodell (GPT 03), S. 23]. Außerdem werden die für RMI nötigen Komponenten "Client Communication" und "Server Communication" hinzugefügt, welche vorher durch die "Communication"-Komponente zwischen Client und Server falsch dargestellt wurden. Da nun allerdings auf der Server-Seite "Game Logic" und "Game Manager" vorhanden waren, wurde ChatGPT gefragt, ob man diese nicht zusammenfassen könnte [9 Komponentenmodell (GPT 04), S. 27]. Dies wurde bejaht und die Komponenten wurden zu "Game Manager/Logic" zusammengefasst. Da der vom Kunden angeforderte Chat nirgends erwähnt wurde, wurde ChatGPT diesbezüglich nochmal gefragt, ob dieser in einer Komponente enthalten sei oder eine eigene Komponente für diesen benötigt wird [10 Komponentenmodell (GPT 05), S. 31]. Auch hier wurde dann die Komponente "Chat Manager" richtig erstellt und in das Diagramm eingefügt. Zusätzlich könnte man noch eine "Authentication"-Komponente in das Diagramm mit aufnehmen, welche sich um die Anmeldungen, Registrierungen und ähnliche Anwendungsfälle kümmert. Da der Chat ohne Probleme in das Diagramm mit aufgenommen wurde, wird das mit einer "Authentication"-Komponente ähnlich sein, weshalb dies nicht als extra Eingabe getätigt wurde. Zum Schluss wurde ChatGPT nochmal nach den Schnittstellen zwischen den Komponenten gefragt [11 Komponentenmodell (GPT 06), S. 35]. Hier versucht ChatGPT allerdings, die Funktionen, die dort benötigt werden, zu erstellen. Diese sind zum einen nicht vollständig und zum anderen an dieser Stelle falsch. Auch mit weiteren Eingaben lassen sich die Schnittstellen nicht richtig darstellen.

Bei Gemini war die erste Ausgabe [12 Komponentenmodell (Gemini 01), S. 39] sehr oberflächlich, und

nach der Beschreibung der Beziehungen gibt es auch mehrere Spielserver. Die zweite Eingabe hat nichts an dem Komponentenmodell verändert, weshalb entschieden wurde, als dritte Eingabe eine genaue Beschreibung mit den benötigten Komponenten zu benutzen [13 Komponentenmodell (Gemini 02), S. 42]. Daraufhin wurde dann das Komponentendiagramm gut erstellt. Es wurde allerdings nicht beschrieben, wer welche Schnittstelle bereitstellt und wer auf diese zugreift. Auf Nachfrage, wie diese aufgebaut sind und wer auf welche Schnittstelle zugreift, ist Gemini mit der Antwort wieder mehr in Richtung technische Infrastruktur gegangen [14 Komponentenmodell (Gemini 03), S. 44]. Hier wurde wieder von der Kommunikation zwischen den unterschiedlichen Systemen wie Client, Server, Datenbank und KI-Modul gesprochen.

Wie zu Beginn bereits erwähnt, hatte Le Chat dasselbe Problem wie ChatGPT, dass es sich zu nah an der technischen Infrastruktur orientiert hat [15 Komponentenmodell (Le Chat 01), S. 46]. Allerdings hat man Le Chat auch nach mehreren Eingaben nicht davon weg bringen können [16 Komponentenmodell (Le Chat 02), S. 48]. Die Beschreibung der Komponenten war eher dürftig, und es wurden keine Komponenten für die RMI-Verbindung erstellt. Erst nach einer expliziten Eingabe, dass eine "ClientCommunication"- und eine "ServerCommunication"-Komponente erforderlich ist, wurden diese erstellt [17 Komponentenmodell (Le Chat 03), S. 50]. Das daraufhin erstellte Diagramm war allerdings eher wieder in der Form der technischen Infrastruktur. Die Komponenten besaßen Interfaces, welche allerdings nicht richtig gepasst haben. Erst nach einer expliziten Eingabe, wie das Komponentendiagramm auszusehen hat, wurde dieses anständig erstellt [18 Komponentenmodell (Le Chat 04), S. 52]. Allerdings haben auch hier die Schnittstellen gefehlt. Auf Nachfrage diesbezüglich wurden diese sogar einigermaßen brauchbar erstellt [19 Komponentenmodell (Le Chat 05), S. 54]. Allerdings werden auch hier schon die Methoden der Schnittstellen aufgelistet. Aber auch hier sind diese an der falschen Stelle erstellt worden und sind ebenfalls nicht vollständig. Dazu wurde beschrieben, dass die "Data"-Komponente auf andere Schnittstellen zugreift. Dies ist allerdings falsch, da die Datenbank ja keine Funktionen von alleine aufruft, sondern lediglich einen Return liefern soll. Nach einem Hinweis korrigierte Le Chat dies allerdings auch [20 Komponentenmodell (Le Chat 61), S. 57]. Ein ähnliches Verhalten sieht man bei den anderen Komponenten auch, dass die Komponenten auf der Server-Seite in Richtung Client-Seite Funktionen schicken können. Hier ist die Frage, wie zum Beispiel eine Spielfeldaktualisierung umgesetzt werden soll. Wenn der Server die aktuellen Positionen der Spielfiguren von alleine an den Client schicken soll, müssen die Komponenten auf der Server-Seite Richtung Client-Seite kommunizieren können. Wenn man allerdings eine Funktion wie "Spielfeld aktualisieren()" auf der Client-Seite hinzufügt, könnte man die Daten über einen Return-Wert zurückschicken. In diesem Fall müssten die Komponenten auf der Server-Seite nicht in Richtung Client-Seite kommunizieren können. Der letzte auffällige Punkt ist, dass die Komponente "ClientLogic" die Schnittstelle "ClientLogicInterface" besitzt, auf welche von der Komponente "Client-Anwendung" zugegriffen wird. Nach dem Komponentendiagramm ist die "ClientLogic"-Komponente allerdings die letzte auf der Client-Seite. Hier ist vermutlich die Anweisung "Als erstes gibt es eine ClientLogic-Komponente, welche mit einer ClientCommunication-Komponente verbunden ist" ungeeignet gewesen, da hier interpretiert wird, dass die "ClientLogic" nur die Logik auf der Client-Seite umsetzt, nicht aber die visuelle Darstellung. Daher hätte man die Komponente in der Eingabe besser "View" oder ähnlich nennen sollen.

Zusammenfassend lässt sich feststellen, dass alle drei Tools zunächst Schwierigkeiten hatten, von der technischen Infrastruktur wegzukommen. ChatGPT hat dies dann aber mit der zweiten Eingabe von alleine ganz gut geschafft. Le Chat und Gemini musste genau gesagt werden, welche Komponenten das Komponentenmodell beinhalten soll. Die Verbindungen zwischen diesen hat Le Chat dann relativ gut erstellt, während Gemini auch dies nicht selbstständig geschafft hat, sondern wieder zurück zur technischen Infrastruktur gegangen ist. Aus diesem Grund wird ChatGPT hier am besten, mit der Note “Befriedigend”, benotet, anschließend folgt Le Chat mit der Note “Ausreichend” und dann Gemini mit der Bewertung “Mangelhaft”.

Sequenzdiagramm

Für das Sequenzdiagramm werden ebenfalls die beiden Anwendungsfälle “Anmelden” und “Zug machen” betrachtet. Der Prompt kann dabei einfach gehalten werden und sieht wie folgt aus:

Erstell mir für den Anwendungsfall “Anmelden” ein Sequenzdiagramm für das Architekturdokument.

Prompt 4.17: Prompt Sequenzdiagramm

Bei allen drei Tools und bei beiden Anwendungsfällen werden keine Schnittstellen in das Sequenzdiagramm mit aufgenommen. Da diese allerdings auch nicht beim Komponentenmodell erstellt wurden, wundert dies hier nicht allzu sehr.

Die erste Ausgabe von ChatGPT ist schon sehr gut [21 AF Nutzer Anmelden (GPT 01), S. 59]. Die Komponenten passen soweit zum Komponentenmodell und die Übergabeparameter auch zu dem Aktivitätsdiagramm vom Anwendungsfall “Anmelden”. Dennoch gibt es ein paar Kleinigkeiten, die auffallen. Zum einen wird die Benutzerinteraktion, “Anmelden klicken()” und “Anmeldeergebnis anzeigen(success)”, mit in das Sequenzdiagramm aufgenommen. Dies gehört hier natürlich nicht mit rein. Zum anderen wird im Diagramm nicht dargestellt, dass die “GameManagerLogic”-Komponente die Anmeldedaten, die sie aus der Datenbank erhält, mit den eingegebenen Daten vergleicht. Dies wurde hier nur als Beschreibung bei der Komponente angegeben. Hier kann man aber auch eine extra Funktion erstellen, die die Komponente bei sich selbst aufruft. Diese beiden Sachen werden nach einem Hinweis gut korrigiert. Die Erstellung der Validierung wird sowohl mit einer genauen Beschreibung [22 AF Nutzer Anmelden (GPT 02), S. 61] als auch mit einer ungenauen Beschreibung [23 AF Nutzer Anmelden (GPT 03), S. 63] richtig erstellt. Bei der ungenauen Beschreibung wird lediglich zwischen der erfolgreichen und nicht-erfolgreichen Validierung unterschieden und je nachdem unterschiedliche Funktionen zurückgesendet. Dies muss allerdings nicht gemacht werden, vor allem da ab “ServerCommunication” immer die gleiche Funktion/Response zurückgeht. Man könnte bereits bei der “GameManagerLogic”-Komponente z.B. ein “true” oder “false” und entweder die Daten eines Users oder “null” zurückliefern. Grundsätzlich gilt allerdings auch hier beim Sequenzdiagramm, dass man einen eindeutigen Parameter beim Anmelden benötigt, da ansonsten die Anmeldung nicht eindeutig einem User zugewiesen werden kann.

Geminis erste Ausgabe verwendet nicht die Komponenten aus dem fachlichen Datenmodell [24 AF Nutzer Anmelden (Gemini 01), S. 65]. Hier wird lediglich beschrieben, was der Spieler, die Client-Anwendung, der Spielservers und eine Authentifizierungskomponente machen. Erst nach einem Hinweis, die Komponenten aus dem Komponentenmodell zu verwenden, wird dies gemacht [25 AF Nutzer Anmelden (Gemini 02), S. 67]. Hier wird dann allerdings rein textuell beschrieben, was passiert, und nicht mit Methoden und Übergabeparametern. Doch auch dies wird nach einem Hinweis geändert [26 AF Nutzer Anmelden (Gemini 03), S. 70]. Gemini macht hier, und auch schon vorher, den gleichen Fehler wie ChatGPT und beschreibt, was der Benutzer macht. Dies wird auch mit einem Hinweis von Gemini nicht geändert [27 AF Nutzer Anmelden (Gemini 04), S. 73]. Des Weiteren fällt auf, dass bei der Auflistung der beteiligten Komponenten auch die Spiellogik aufgeführt wird. Allerdings kommt diese in der Beschreibung des Sequenzdiagramms nicht vor. Grundsätzlich passen die Beschreibungen nicht immer. Zum einen wird statt "ServerCommunication", welche bei den beteiligten Komponenten aufgelistet wird, immer von "Spielservers" geschrieben. Zum anderen wird beschrieben, dass "Authentication" in "Data" die Anmeldeinformationen prüft. Dies funktioniert allerdings nicht. Man sollte sich besser Daten, die mit dem Benutzernamen übereinstimmen, ausgeben lassen und diese dann in "Authentication" überprüfen. Dafür sollte allerdings der Benutzername eindeutig einem Nutzer zugewiesen werden können. Des Weiteren ist auch hier wieder der Fehler mit den nicht eindeutigen Anmeldedaten vorhanden.

Bei Le Chat wird ein PlantUML-Code mit erstellt, welcher allerdings zu einem Syntaxfehler führt [28 AF Nutzer Anmelden (Le Chat 01), S. 76]. Die dazu erstellte Beschreibung ist nur grob und nennt keine Funktionen. Es wird nur der Ablauf beschrieben, wer an wen sendet. Dadurch, dass im Komponentenmodell keine "View"-Komponente oder Ähnliches vorhanden ist, verwendet Le Chat hier von alleine eine "Client-Anwendung"-Komponente. Dies ist daher korrekt. Nach einem Hinweis, dass Funktionen mit in der Ausgabe erstellt werden sollen, macht Le Chat das ganz gut [29 AF Nutzer Anmelden (Le Chat 02), S. 78]. Ein Fehler der auffällt ist, dass als Anmeldedaten ein Benutzername und Passwort verwendet werden. In dem Dialog und auch dem Anwendungsfall wurde allerdings die E-Mail-Adresse und ein Passwort verwendet. Auch hier ist aber in beiden Fällen wieder das Problem, dass keiner der Attribute einzigartig ist und dadurch nicht eindeutig ein Benutzer identifiziert werden kann.

Der Prompt für den Anwendungsfall "Zug machen" ist äquivalent. Bei allen dreien fällt hier allerdings auf, dass keine Aktualisierung der anderen Clients stattfindet. Daher stellt sich die Frage, ob dies einfach vergessen wurde oder ob die Tools an dieser Stelle davon ausgehen, dass die Aktualisierung in einem separaten Anwendungsfall geschieht.

Grundsätzlich stellt man bei ChatGPT fest, dass das Sequenzdiagramm [30 AF Zug Machen (GPT 01), S. 79] nicht zum Aktivitätsdiagramm [04_Anforderungsspezifikation, S. 173 ff.] passt. Es wird nicht mehrmals zwischen Client und Server hin und her gewechselt. Außerdem wird das Würfeln hier nicht betrachtet. Das Sequenzdiagramm passt allerdings deutlich besser als das Aktivitätsdiagramm, weshalb diese Inkonsistenz nicht negativ ist. An dieser Stelle könnte man das Anwendungsfalldiagramm noch einmal überarbeiten lassen. Negativ fällt aber trotzdem auf, dass nicht betrachtet wird, was passiert, wenn das Feld bereits besetzt ist. Nach einem Hinweis wird dies allerdings hinzugefügt [31 AF Zug Machen (GPT 02), S. 81]. Dazu ist immer noch unvollständig, dass nicht überprüft wird, welche Farbe die

Figur auf dem Feld hat. Die Figur wird einfach automatisch auf das Startfeld zurückgesetzt. Zusätzlich stellt man sich die Fragen, ob und wo der Zug mit dem gewürfelten Wert verglichen und überprüft wird.

Bei Geminis Ausgabe [32 AF Zug Machen (Gemini 01), S. 83] wird unter den beteiligten Komponenten eine "Spielfigur"- und eine "Würfel"-Komponente aufgelistet. Dies ist nicht passend zum Komponentenmodell. Des Weiteren wird auch wieder der Spieler in die Beschreibung aufgenommen. Zusätzlich wird hier, wie beim Sequenzdiagramm für "Anmelden", nur beschrieben, was passiert und keine Methoden erstellt. Nach einem Hinweis werden diese allerdings wieder erstellt [33 AF Zug Machen (Gemini 02), S. 86]. Es wird außerdem zweimal vom Client zum Server und wieder zurückgegangen: einmal fürs Würfeln und einmal für den Zug. Dazu fehlen bei den Funktionen die Attribute, die zum Zug dazugehören. Darüber hinaus wird hier wieder von "Spielserver" statt "ServerCommunication" gesprochen. Zusätzlich gibt es keine Überprüfung, ob der Spieler würfeln darf. Erst nachdem die gewürfelte Zahl wieder beim Client angekommen ist, wird überprüft, ob der Spieler am Zug ist. Dadurch könnte man zum Beispiel ganz häufig würfeln und damit den Server überlasten. Aber auch ohne böse Absicht ist es recht ineffizient, erst am Ende zu überprüfen, ob der Spieler überhaupt am Zug ist. Dadurch wird unnötig Rechenzeit verschwendet, welche man auch sparen kann, indem man die Überprüfung als erstes auf dem Server macht. Ebenfalls auffällig ist, dass die Beschreibung bei Punkt 13 einfach aufhört. Vermutlich hat Gemini hier die Ausgabe einfach wieder abgebrochen.

Der von Le Chat erstellte PlantUML-Code ist auch hier wieder fehlerhaft und die Beschreibung wurde nur grob erstellt [34 AF Zug Machen (Le Chat 01), S. 89]. Daher wurde auch hier nach einer genauen Beschreibung mit Funktionen gefragt, welche dann erstellt wurden [35 AF Zug Machen (Le Chat 02), S. 91]. Bei den Funktionen fällt zum einen auf, dass es nur ein Attribut für das ZielFeld gibt. Dies entspricht nicht dem Aktivitätsdiagramm. Dort wurden zwei Attribute, "Figur.x" und "Figur.y", verwendet. Außerdem stellt sich die Frage, warum eine Spiel-ID übergeben wird. Da "Figur" im fachlichen Datenmodell eine einzigartige Id besitzt, benötigt man die SpielId eigentlich nicht. Außerdem stellt sich hier unter anderem die Frage, wie die anderen Clients das aktualisierte Spiel erhalten. Bei der Frage, ob der Punkt 10 an einen Client oder an alle gesendet wird [36 AF Zug Machen (Le Chat 03), S. 93], antwortet Le Chat, dass die Aktualisierung an alle Clients gesendet werden muss und erklärt, wie das umgesetzt wird. Eine weitere Sache, die auffällt, ist, dass das gesamte Spiel aus der Datenbank geholt, die neue Position aktualisiert und anschließend das gesamte Spiel wieder in die Datenbank geladen wird. Hier wäre es aber eigentlich effizienter, einfach den entsprechenden Wert der Spielfigur zu aktualisieren. Wenn man Le Chat danach fragt, wird die Frage mit "Ja" beantwortet [37 AF Zug Machen (Le Chat 04), S. 94]. Ein weiterer Punkt, der ebenfalls auffällt, ist, dass auch hier nicht geprüft wird, ob das Feld bereits belegt ist. Le Chat fügt dies zwar in die Beschreibung ein [38 AF Zug Machen (Le Chat 05), S. 95], allerdings ohne Methoden, sondern als reine Beschreibung. Des Weiteren wird auch nicht geprüft, ob der Spieler, der den Zug durchführen will, an der Reihe ist und ob der Zug dem gewürfelten Wert entspricht.

Zusammenfassend lässt sich sagen, dass die Tools das Sequenzdiagramm recht gut erstellen, vor allem wenn man bedenkt, dass die Erstellung des Komponentenmodells eher schlecht funktioniert hat. Das größte Problem stellt die Konsistenz zu den vorherigen Dokumenten dar, die teilweise schon viele Eingaben zurückliegen und an die sich die Tools nicht mehr erinnern können. ChatGPT schafft

es als einziges, das Sequenzdiagramm grafisch in PlantUML zu erstellen, und auch die Beschreibung ist gut. Da ChatGPT neben der Aktualisierung nur die Überprüfung des Feldes vergisst, wird ChatGPT hier mit "Gut" bewertet. Geminis Beschreibung der beteiligten Komponenten passt wieder nicht und auch, dass wieder nach Funktionen und der Entfernung des Users gefragt werden muss, obwohl dies in den Eingaben direkt davor schon gemacht wurde, fällt negativ auf. Ebenso ist der Ablauf des Anwendungsfalls schlecht gewählt. Hier könnte man allerdings noch argumentieren, dass man diesen einfach selbstständig auf zwei Anwendungsfälle aufteilt. Da aber auch die Parameter vom Bewegen der Spielfigur unvollständig sind, wird Gemini hier nur mit "Ausreichend" bewertet. Le Chat ist eine Mischung aus ChatGPT und Gemini, was die Qualität angeht. Dass auch hier wieder nach einer genauen Beschreibung gefragt werden muss und das fehlende Einfügen der Methoden zum Prüfen, ob das Feld bereits belegt ist, fällt negativ auf. Dafür sind die anderen Methoden gut erstellt worden, weshalb Le Chat gerade noch so mit "Befriedigend" bewertet wird. Man erhält aber bei allen drei Tools ein Grundgerüst und eine Idee, was erstellt werden soll und wie das Sequenzdiagramm in etwa aussehen soll.

Interfaces in Java

Da die Erstellung mit ChatGPT bis hierhin am besten geklappt hat, wurde probiert, die Interfaces in Java zu erstellen [siehe "Code der Interfaces" im Anhang]. Dabei fällt direkt auf, dass die erstellten Klassen nicht zu den Schnittstellen bzw. zu den Komponenten aus dem Komponentenmodell und damit auch nicht zu den Sequenzdiagrammen passen, da hier die Klassen "AuthenticationService.java" und "GameService.java" keiner Komponente zugewiesen werden können. Es könnte allerdings sein, dass ChatGPT mehrere Schnittstellen für eine Komponente erstellt hat. Doch wenn man ChatGPT fragt, welche Klassen zu welchen Komponenten gehören, wird für jede Klasse eine Komponente aufgelistet. Dazu fällt auf, dass auch für die Komponente "UserInterface" eine Klasse erstellt wurde, wobei sich die Frage stellt, wer darauf zugreifen soll. Grundsätzlich sollte diese Komponente nur von "ClientCommunication" Funktionen aufrufen und dann einen Return erhalten. Darüber hinaus passen die Namen in den Klassen nicht immer zu denen aus den Sequenzdiagrammen. Teilweise fehlen auch Funktionen bzw. sind in den Klassen falsch. Zum Beispiel ist die Methode "moveOccupyingFigureToStart(int figureID)" nicht in der Klasse "DatabaseAccess", sondern bei "GameService" enthalten. Außerdem ist es irritierend, dass zum Beispiel in "ClientCommunication.java" die Methoden "void" sind, aber dafür eine Callback-Funktion implementiert ist. Besonders bei der Methode für den Login, da hier nur true oder false zurückgegeben wird und es eigentlich auch nie vorkommen sollte, dass der Server von selbst eine LoginResponse zurückschickt. Dies könnte man nämlich noch bei der "MoveResponse" argumentieren. Man könnte hier die Callback-Funktion nutzen, um das Spielbrett zu aktualisieren. Doch da dies in den Dokumenten noch nicht betrachtet wurde, wird dies nicht der Hintergedanke von ChatGPT gewesen sein.

Grundsätzlich kann man daher sagen, dass die Erstellung der Interfaces nicht wirklich funktioniert. Man erhält hier ein grobes Grundgerüst, muss dieses jedoch selbstständig anpassen, da die Beziehungen zwischen den Komponenten auf der Serverseite durch die beiden extra Komponenten umgestellt wurden. Auch die Art und Weise, wie die Rückgaben umgesetzt wurden, führt dazu, dass man die Interfaces noch einmal anpassen muss. Als Bewertung würde ChatGPT hier "Ausreichend" erhalten, da einige

Sachen fehlen.

5 Zusammenfassung und Ausblick

Im Verlauf dieser Arbeit wurde deutlich, wie LLM-Tools bei der Erstellung von Dokumenten im Software Engineering Prozess unterstützen können. Im abschließenden Kapitel werden die Ergebnisse der Analyse zusammengefasst, um die Erkenntnisse und die Effektivität der eingesetzten Tools zu bewerten. Anschließend wird ein Ausblick gegeben, der eine grobe Einschätzung zur zukünftigen Entwicklung und dem Potenzial von LLM-Tools im Software Engineering liefert. Hierbei wird sowohl auf die erwarteten technologischen Fortschritte als auch auf mögliche neue Anwendungsfelder eingegangen.

5.1 Zusammenfassung der Ergebnisse

Es lässt sich grundsätzlich sagen, dass alle drei Tools es geschafft haben, bei den meisten Dokumenten eine solide und korrekte Basis zu erstellen. Dabei fällt vor allem auf, dass einfachere Sachen wie die Erstellung von Texten oder das Designen von Dialogen sehr gut klappt. Schwierige Diagramme, die vielleicht auch aufeinander aufbauen, klappen hingegen noch nicht so gut und beinhalten häufig noch Fehler. Auch die Konsistenz zwischen den einzelnen Dokumenten ist durchwachsen. Dies stellt sich besonders bei den Sequenzdiagrammen raus, da diese recht wenig zu dem dazugehörigen Aktivitätsdiagramm und dem Fachlichen Datenmodell passen. Daher sind die Tools aktuell noch eher ungeeignet für große zusammenhängende Projekte. Wobei man beachten muss, dass ChatGPT und Le Chat da deutlich weniger Probleme hatten als Gemini. Zusätzlich ist es manchmal schwierig, die Tools von einer gewissen Formulierung oder Implementierung weg zu bekommen, was man besonders bei dem Komponentenmodell und dem zuvor verwendeten Begriff "Komponente" im Rahmen der Technischen Infrastruktur gemerkt hat. Wenn sich die Tools mal mit einer Ausgabe in die komplett falsche Richtung bewegen, kann es helfen, sich die erste Ausgabe zu einem Dokument nochmal ausgeben zu lassen. Da diese immer etwas verschieden waren, konnte man so die Tools leicht korrigieren. Dies war besonders beim Besprechungsprotokoll hilfreich.

Es wurde probiert, die Anfangseingaben einfach zu halten und so zu formulieren, dass auch Nutzer ohne Vorwissen diese stellen würden. Dabei muss beachtet werden, dass wenn man spezielle Anforderungen an die Ausgabe hat, diese mit in die Eingabe eingefügt werden müssen. Besonders bei Tabellen, die ein gewissen Format haben sollen, muss dieses mit angegeben werden. Auch falls zusätzliche Informationen, die die Tools nicht wissen können, müssen diese mit in die Eingabe geschrieben werden. Fraglich ist es dann teilweise, ob unwissende Nutzer es schaffen, die Ausgaben auf ihre Korrektheit zu prüfen und nachfolgend richtige und passende Eingaben zu tätigen. Aber auch ohne das erhält man eine grobe Idee was gefordert ist und wie die Inhalte erstellt werden müssen.

Im Bezug auf die Handhabung, war ChatGPT sehr angenehm und hat wenig Probleme gemacht. Es lässt sich leicht eine Antwort neu generieren und auch alte Eingaben konnten nach Belieben nochmal geändert werden. Auch die Qualität der Ausgaben war von ChatGPT am besten, was vermutlich auch mit an der Version 4o liegt. Negativ ist dabei allerdings, dass man nur eine gewisse Anzahl an Eingaben machen kann. Sobald das Eingabelimit erreicht ist, kann man mit der 3.5er Version weiter arbeiten. Nach einem gewissen Zeitintervall lässt sich mit 4o weiterarbeiten. Alternativ muss ein Abonnement abgeschlossen werden, um unbegrenzt mit 4o arbeiten zu können. Da die Ausgaben der 3.5er Version allerdings einen spürbaren negativen Qualitätsunterschied im Vergleich zu 4o haben, ist der Rückgriff auf 3.5 eher unangenehm. Wenn man Zeit hat ist es besser einfach zu warten und dann mit 4o weiter zu arbeiten. Im Verlauf meiner Arbeit hat es sich bewährt, erst mit den anderen Tools, vor allem Le Chat, die Eingaben zu machen, da die Eingaben häufig umgeschrieben werden mussten, bis man das gewünschte Ergebnis erhalten hat. Anschließend kann man die "fertige" Eingabe ChatGPT geben. Dadurch verbraucht man keine "Aufladungen" bei ChatGPT mit Eingaben, die einen zunächst nicht weiter bringen, sondern gibt ChatGPT nur die fertig formulierten Eingaben.

Die Arbeit mit Gemini war sehr problematisch, da das Erinnerungsvermögen vergleichsweise kurz war. Bereits nach ein oder zwei Tagen wusste ChatGPT nicht mehr, was vorher erstellt wurde, obwohl man im selben Chat weiter arbeitet. Teilweise wurde sogar direkt nach einer Eingabe das erstellte wieder vergessen. Aus diesem Grund musste die Erstellung mit Gemini in einem Zug gemacht werden, was aber in der Praxis im Alltag nicht umsetzbar ist, da man vermutlich über mehrere Wochen oder sogar Monaten an so einem Projekt arbeitet. Besonders als Student verläuft so ein Projekt meistens während eines Semesters oder Trimesters. Die Alternative wäre, Gemini bei jeder Eingabe den Rahmen des Projektes mit den benötigten Informationen mit zu geben. Dies ist aber grade zum Ende hin nicht umsetzbar, da die Dokumente von immer mehr anderen Dokumenten abhängig werden. Des Weiteren ist Gemini sehr stark von den Eingaben abhängig und hat bei Fragen häufig nur eine Antwort mit Erklärung geliefert und nicht in Kontext mit den vorherigen Eingaben gesetzt. Man musste extra die Anweisungen geben dies zu tun. Zusätzlich hat Gemini gerne mal die Ausgaben auf Englisch geschrieben. Außerdem kann man nur die letzte Eingabe nochmal verändern und es gab auch keine Funktion, um eine Antwort neu generieren zu lassen. Hier musste dann der Punkt am Ende eines Satzes entfernt werden, damit dies nochmal gemacht wurde. Dazu kommt noch, dass Ausgaben irgendwann einfach abbrechen. Diese Aspekte führten dazu, dass die Arbeit mit Gemini sehr mühsam und schwierig war.

Le Chat war eine kleine Mischung aus den ChatGPT und Gemini. Bei der Qualität der Ausgaben hat man häufig gemerkt, dass Le Chat noch nicht so weit entwickelt ist wie ChatGPT-4o, allerdings ist die Bedienung ähnlich angenehm wie mit ChatGPT und auf jeden Fall besser als mit Gemini. Grundsätzlich gab es hier keine besonderen Vorkommnisse bei der Arbeit mit Le Chat. Lediglich wurden bei den Ausgaben häufig ein Link mit eingefügt, der zu einem Bild führen soll, allerdings mit "/XyXyXy" geendet hat und nur ein Platzhalter war. Dies war zu Beginn sehr verwirrend, da der Link nicht sichtbar war, sondern nur gesehen werden konnte, wenn man die gesamte Eingabe kopiert. Des Weiteren ist manchmal die Warnung, "Der Inhalt kann gefährliche oder sensible Themen beinhalten", für die Eingabe, "Formuliere die Beschreibung der Aktivitäten etwas genauer. Beschreibe welcher Wert wo gespeichert wird mithilfe des Fachlichen Datenmodells." angezeigt worden.

5.2 Kritik an der Arbeit

Die Kritik an meiner Arbeit dient dazu, die Schwächen meiner Vorgehensweise zu reflektieren und Bereiche für zukünftige Verbesserungen zu identifizieren.

Dabei gibt es zwei große Bereiche: Die Erstellung der Dokumente bzw. die Arbeit mit den LLM Tools und die Analyse der Ergebnisse.

Bei der Arbeit der LLM Tools wäre es sinnvoller gewesen, im Vorhinein festzulegen, wie umfangreich und wie viele Eingaben für ein Dokument für ein Tool getätigt werden sollen. Der Gedanke, die selben Eingaben zu benutzen war schon gut, durch die unterschiedlichen Ausgaben ist es allerdings nicht möglich, dies für ein Inhalt fortzuführen. Daher hätte man entweder festlegen sollen, dass nur die erste Ausgabe analysiert wird oder sich auf eine gewisse Anzahl an Eingaben und die Genauigkeit festzulegen. Bei beiden Methoden werden allerdings die Möglichkeiten der Tools eingegrenzt und auch der Bezug zur Realität wäre geschmälert worden. Dafür könnten die Ergebnisse und die Tools allerdings deutlich besser verglichen werden. Hier muss man entscheiden, welcher Aspekt wichtiger ist. Eine weitere Möglichkeit wäre eine Art Mischung zu machen. Man könnte sich auf eine grobe Anzahl an Eingaben festlegen, was aber keine feste Grenze ist. Dadurch wären die Ergebnisse etwas vergleichbarer und man würde nicht den Aspekt der Verbesserung außer acht lassen. In jedem Fall sollte sich aber darüber im Vorhinein Gedanken gemacht und festgelegt werden.

Bei der Analyse hätte man sich ebenfalls zunächst im Vorhinein überlegen sollen, wie man vorgeht. Durch die unterschiedlichen Ergebnisse und die vielen Anforderungen, die vom Kunden festgelegt als auch die Spielregeln, gab es viele Dinge zu bedenken. Dass die Tools dann die Dokumente nicht konsistent erstellt haben, hat die Analyse zusätzlich erschwert. Dadurch gab es Punkte, die beim Erstellen der Prompts oder bei der Analyse der Ausgaben nicht bedacht wurden, dann beim Schreiben der Texte allerdings erkannt wurden und nicht nachträglich geändert werden konnten. Auch wenn in späteren Dokumenten auf einmal Aspekte aufkommen, welche bei den vorherigen Dokumenten nicht näher betrachtet wurden, lässt sich dies schwer korrigieren. Dafür hätte es geholfen, sich eine Liste mit allen Dingen, die implementiert werden müssen, zu schreiben und in welchen Dokumenten diese vorkommen müssen um dann strukturiert sagen zu können, welche Aspekte fehlen, welche inkonsistent zu vorherigen Angaben sind und welche richtig enthalten sind. Dadurch hätte man die Analyse strukturierter und vollständiger schreiben können und es müsste weniger nachträglich verbessert werden.

5.3 Ausblick und zukünftige Entwicklungen

Zunächst wird ein kleiner Ausblick auf mögliche Fortführungen und Weiterentwicklungen zu diesem Thema gegeben und anschließend eine Einschätzung, wo die Reise mit LLM Tools noch hingehen könnte.

Zum einen wurde es nicht geschafft, die Testspezifikation erstellen zu lassen. Dies sollte größtenteils kein Problem sein, da in dem Dokument nur textuell beschrieben wird, wie der Testablauf aussieht und einzelne Tests schriftlich erstellt werden. In den bisherigen Dokumenten waren solche Beschreibung

nie ein Problem, weshalb es nicht zu erwarten ist, dass es hier anders ist. Lediglich Gemini könnte Probleme bei der Erstellung der Testdaten bekommen, da es sich hierbei häufig um eine sehr große Tabelle handelt. Bei Gemini könnte es wieder passieren, dass die Generierung der Ausgabe einfach abgebrochen wird.

Anschließend könnte man die Implementierung des Spiels als Java-Code fortsetzen.

Auch die Abschnitte und Inhalte, die hier bisher nicht betrachtet wurden, da sie keine allzu große Relevanz haben, könnten noch erstellt werden.

Als zusätzliche Unterstützungsmöglichkeiten könnte man sich die Pflege der Dokumente angucken. Schaffen es die Tools zum Beispiel die Risikoliste passend zum aktuellen Abschnitt anzupassen? Oder diese bei einem eingetretenen oder gelösten Risiko anzupassen? Können die Tools zum Beispiel die Anwendungsfälle nachträglich überarbeiten, so dass diese zu den Sequenzdiagrammen passen? All dies wären weitere Unterstützungsmöglichkeiten, die die Tools bereitstellen könnten.

Die Unterstützungsmöglichkeiten lassen sich grundsätzlich auch in Unternehmen und Behörden einsetzen. Hier sollte man allerdings beachten, dass die in dieser Arbeit vorgestellten LLM Tools Produkte von Unternehmen sind. Hier wird immer der Unternehmensgewinn an erster Stelle stehen und daher wird sich vermutlich auch an den Daten, die als Eingaben benutzt werden, bedient. Daher sollte man aufpassen, was man als Eingabe gibt. Eine Lösung könnten zukünftig Lokale LLM Tools bieten, die vollständig auf einem eigenen System laufen. Innerhalb Unternehmen und Behörden, wo meistens auch mehr Rechenleistung zur Verfügung steht, können auch größere LLMs als Inselsystem ohne Außenanbindung laufen (z.B. das neue LLM der Bundeswehr). Besonders in Behörden, wo mit eingestuftem Inhalten gearbeitet wird, wäre das eine gute Möglichkeit, die Vorteile von LLM Tools mit einzubinden. Eine weitere Möglichkeit wären speziell trainierte LLM Tools. Es könnten irgendwann LLM Tools zum Kauf zur Verfügung stehen, die nur allgemeine Informationen zum Trainieren erhalten haben, damit die Grundfähigkeiten zur Verfügung stehen. Anschließend könnte man das LLM Tool auf einen speziellen Anwendungsbereich hin trainieren. In diesem Fall zum Beispiel für das Software Engineering. Dadurch könnten sich die Fähigkeiten des LLM Tools in diesem Bereich verstärken und weniger Rechenleistung benötigen, als ein Tool, welches alles ein bisschen kann.

Um nochmal zurück auf den Studenten und auf Schüler zu kommen: Es lässt sich mit ziemlicher Sicherheit sagen, dass in Zukunft die Bedenken, dass KI für diese nicht förderlich sondern eher schädlich sind, nicht abnehmen werden. Wenn allerdings die LLM Tools in der Arbeitswelt eine unabdingbare Rolle spielen, ist es von Vorteil wenn Schüler und Studenten sich mit diesen Tools schon im Vorhinein auseinandersetzen. Sie lernen mit den Tools umzugehen und wie Eingaben gestellt werden müssen, um die gewünschte Antworten zu erhalten. Des Weiteren wird auch geübt die Ausgaben auf ihre Korrektheit zu analysieren.

Auf jeden Fall sind die Large Language Models auf dem Weg, einen großen Einfluss auf das zukünftige Leben zu nehmen.

6 Anhang

Kundengespräch

****Kunde (Herr Müller):**** Guten Tag, ich bin Herr Müller. Wie geht es ihnen?

****Entwickler (Frau Schmidt):**** Hallo, ich bin Frau Schmidt. Mir geht es gut. Wie geht es ihnen?

****Kunde (Herr Müller):**** Mir geht es auch gut. Ich möchte ein digitales "Mensch ärgere dich nichtSpiel entwickeln lassen und brauche Ihre Expertise dafür.

****Entwickler (Frau Schmidt):**** Es freut mich, dass Sie sich für unsere Dienstleistungen interessieren. Können Sie mir bitte mehr über Ihr Projekt erzählen? Welche Funktionen und Eigenschaften sollte das Spiel haben?

****Kunde:**** Natürlich. Zunächst möchte ich, dass das Spiel als JavaFX Anwendung auf dem Computer läuft. Die Anwendung soll eine Server-Client-Applikation, die mit Hilfe von Java realisiert wird, sein. Server und Client sollen über RMI kommunizieren. Es sollte das klassische "Mensch ärgere dich nichtSpielbrett und die entsprechenden Regeln enthalten. Außerdem soll es auch möglich sein, mehrere Spiele gleichzeitig zu spielen.

****Entwickler:**** Verstehe. Sollen die Spieler gegen den Computer, gegen andere Spieler online oder beides spielen können?

****Kunde:**** Beides wäre ideal. Es sollte einen Einzelspielermodus gegen KI-Gegner sowie einen Mehrspielermodus geben, in dem man online gegen Freunde oder zufällige Gegner spielen kann.

****Entwickler:**** Gut, dann werden wir einen Multiplayermodus integrieren. Wie viele Spieler sollen maximal teilnehmen können?

****Kunde:**** Wie im klassischen Spiel sollten es bis zu vier Spieler sein.

****Entwickler:**** Perfekt. Haben Sie Vorstellungen bezüglich des Designs? Soll das Spiel ein klassisches, traditionelles Aussehen haben oder möchten Sie ein modernes Design?

****Kunde:**** Ein klassisches Design mit einer ansprechenden, modernen Benutzeroberfläche wäre gut. Es sollte an das traditionelle Spiel erinnern, aber gleichzeitig frisch und ansprechend wirken.

****Entwickler:**** Verstanden. Gibt es besondere Funktionen oder Features, die Ihnen wichtig sind? Zum Beispiel Chat-Funktionalitäten, Punktesysteme oder spezielle Animationen?

****Kunde:**** Ein Chat während des Spiels wäre nett. Ein Punktesystem könnte die Motivation erhöhen, also wäre das auch eine gute Idee. Animationen sollten einfach, aber ansprechend sein - nichts Übertriebenes.

****Entwickler:**** Das klingt alles machbar. Wie sieht es mit dem Zeitrahmen aus? Haben Sie eine bestimmte Deadline, bis wann das Spiel fertig sein soll?

****Kunde:**** Ich denke, ein Zeitraum von sechs Monaten sollte machbar sein. Glauben Sie, dass das realistisch ist? Darf ich außerdem Fragen wie ihr Team aufgestellt ist?

Prompt 6.1: Kundengespräch

****Entwickler:**** Ja, sechs Monate sind ein realistischer Zeitraum für dieses Projekt, vorausgesetzt, wir haben regelmäßige Feedback-Schleifen und Sie können uns zeitnah Ihre Anforderungen und Rückmeldungen geben. Wir sind ein Team aus 4 Personen. Ich bin die Projektleiterin, der Herr Schneider ist unser Anforderungsanalyst und fachlicher Chefdesigner, die Frau Fischer ist unsere Systemarchitektin und der Herr Becker ist der Testmanager.

****Kunde:**** Das klingt gut. Was wären die nächsten Schritte?

****Entwickler:**** Wir würden zunächst einen detaillierten Projektplan erstellen, der alle Funktionen und Meilensteine festlegt. Danach beginnen wir mit der Entwicklung des Prototyps. Während der gesamten Entwicklung würden wir regelmäßige Meetings abhalten, um den Fortschritt zu besprechen und eventuelle Anpassungen vorzunehmen.

****Kunde:**** Das klingt nach einem guten Plan. Wie sind sie eigentlich dazu gekommen, Entwickler zu werden?

****Entwickler:**** Ich habe mich schon als Kind für Computerspiele interessiert und auch wie diese im Hintergrund funktionieren. Daher habe ich recht früh angefangen das Programmieren zu lernen. Das hat mir immer viel Spaß gemacht und so habe ich nach der Schule eine Ausbildung in der Branche angefangen.

****Kunde:**** Ah ich verstehe. Sie sind jetzt mittlerweile Projektleiterin oder?

****Entwickler:**** Ja, genau.

****Kunde:**** Fehlt ihnen die Arbeit als Programmiererin? Als Projektleitern machen Sie ja wahrscheinlich nicht mehr ganz so viel in dem Bereich oder?

****Entwickler:**** Ne, das stimmt. Aber man ist ja trotzdem noch damit in Berührung. Wenn mal irgendwo Schwierigkeiten bei der Implementierung auftreten, kann ich auch helfen diese zu lösen. Ein wenig fehlen tut es mir schon, aber meine jetzige Tätigkeit als Projektleiterin macht mir auch sehr viel Spaß.

****Kunde:**** Alles klar. Ich freue mich darauf, mit Ihnen zusammenzuarbeiten.

****Entwickler:**** Vielen Dank, Herr Müller. Wir freuen uns auch auf die Zusammenarbeit. Wir werden Ihnen in Kürze einen detaillierten Projektplan und ein Angebot zukommen lassen.

****Kunde:**** Perfekt, ich warte auf Ihre Nachricht. Einen schönen Tag noch!

****Entwickler:**** Ihnen auch, Herr Müller. Bis bald!

Prompt 6.2: Kundengespräch

Tabelle der auf dem USB-Stick enthaltenen Dateien

Datei	Beschreibung
01_Besprechungsprotokolle.pdf	Enthält die erstellten Besprechungsprotokolle der drei Tools.
02_Projekthandbuch.pdf	Enthält die erstellten Abschnitte vom Projekthandbuch der drei Tools.
03_Gemini_Risikoliste.xlsx	Enthält die erstellte Risiko- und Maßnahmentabelle von Gemini.
03_GPT_Risikoliste.xlsx	Enthält die erstellte Risiko- und Maßnahmentabelle von ChatGPT.
03_LeChat_Risikoliste.xlsx	Enthält die erstellte Risiko- und Maßnahmentabelle von Le Chat.
04_Anforderungsspezifikation.pdf	Enthält die erstellten Dokumente der Anforderungsspezifikation der drei Tools.
05_Architekturdokument.pdf	Enthält die erstellten Dokumente des Architekturdokumentes der drei Tools.

Code der Interfaces

```
1 public interface UserInterface {
2     /**
3      * Requests to log in the user with the provided email and password.
4      *
5      * @param email    The email of the user.
6      * @param password The password of the user.
7      */
8     void login(String email, String password);
9
10    /**
11     * Displays the login result to the user.
12     *
13     * @param success true if login was successful, false otherwise.
14     */
15     void displayLoginResult(boolean success);
16 }
```

code/UserInterface.java

Code 6.1: UserInterface [ChatGPT]

```

1 public interface ClientCommunication {
2     /**
3      * Forwards the login request to the server.
4      *
5      * @param email    The email of the user.
6      * @param password The password of the user.
7      */
8     void forwardLoginRequest(String email, String password);
9
10    /**
11     * Receives the login response from the server.
12     *
13     * @param success true if login was successful, false otherwise.
14     */
15    void receiveLoginResponse(boolean success);
16
17    /**
18     * Sends a move request to the server.
19     *
20     * @param playerId    The ID of the player making the move.
21     * @param figureID    The ID of the figure being moved.
22     * @param startFieldID The starting field ID of the figure.
23     * @param endFieldID  The ending field ID of the figure.
24     */
25    void sendMoveRequest(int playerId, int figureID, int startFieldID, int ↵
        endFieldID);
26
27    /**
28     * Receives the move response from the server.
29     *
30     * @param response The response from the server.
31     */
32    void receiveMoveResponse(MoveResponse response);
33 }

```

code/ClientCommunication.java

Code 6.2: ClientCommunication [ChatGPT]

```
1 public interface ServerCommunication {
2     /**
3      * Forwards the login request to the game manager logic.
4      *
5      * @param email    The email of the user.
6      * @param password The password of the user.
7      * @return true if login was successful, false otherwise.
8      */
9     boolean forwardLoginRequest(String email, String password);
10
11     /**
12      * Sends the login response to the client.
13      *
14      * @param success true if login was successful, false otherwise.
15      */
16     void sendLoginResponse(boolean success);
17
18     /**
19      * Forwards a move request to the game logic.
20      *
21      * @param playerId    The ID of the player making the move.
22      * @param figureID    The ID of the figure being moved.
23      * @param startFieldID The starting field ID of the figure.
24      * @param endFieldID  The ending field ID of the figure.
25      * @return a response indicating the result of the move.
26      */
27     MoveResponse forwardMoveRequest(int playerId, int figureID, int startFieldID ↵
28         , int endFieldID);
29
30     /**
31      * Sends the move response to the client.
32      *
33      * @param response The response from the game logic.
34      */
35     void sendMoveResponse(MoveResponse response);
36 }
```

code/ServerCommunication.java

Code 6.3: ServerCommunication [ChatGPT]

```
1 public interface AuthenticationService {
2     /**
3      * Retrieves user data from the database based on the provided email.
4      *
5      * @param email The email of the user.
6      * @return User data including email and hashed password.
7      */
8     User retrieveUserData(String email);
9
10    /**
11     * Compares the provided password with the hashed password from the database↵
12     *
13     * @param password The provided password.
14     * @param hashedPassword The hashed password from the database.
15     * @return true if the passwords match, false otherwise.
16     */
17     boolean comparePasswords(String password, String hashedPassword);
18 }
```

code/AuthenticationService.java

Code 6.4: AuthenticationService [ChatGPT]

```
1 public interface GameManagerLogic {
2     /**
3      * Validates user credentials by checking the database.
4      *
5      * @param email    The email of the user.
6      * @param password The password of the user.
7      * @return true if the user is validated, false otherwise.
8      */
9     boolean validateUser(String email, String password);
10
11     /**
12      * Processes a move request.
13      *
14      * @param playerId  The ID of the player making the move.
15      * @param figureID  The ID of the figure being moved.
16      * @param startFieldID The starting field ID of the figure.
17      * @param endFieldID The ending field ID of the figure.
18      * @return a response indicating the result of the move.
19      */
20     MoveResponse processMove(int playerId, int figureID, int startFieldID, int ↵
21         endFieldID);
22 }
```

code/GameManagerLogic.java

Code 6.5: GameManagerLogic [ChatGPT]

```
1 public interface GameService {
2     /**
3      * Validates a move request.
4      *
5      * @param playerId    The ID of the player making the move.
6      * @param figureID    The ID of the figure being moved.
7      * @param startFieldID The starting field ID of the figure.
8      * @param endFieldID  The ending field ID of the figure.
9      * @return true if the move is valid, false otherwise.
10    */
11    boolean validateMove(int playerId, int figureID, int startFieldID, int ↵
        endFieldID);
12
13    /**
14     * Checks if a field is occupied.
15     *
16     * @param fieldID The ID of the field to check.
17     * @return true if the field is occupied, false otherwise.
18     */
19    boolean isFieldOccupied(int fieldID);
20
21    /**
22     * Moves the figure to the new position.
23     *
24     * @param figureID The ID of the figure being moved.
25     * @param newFieldID The new field ID where the figure will be moved.
26     */
27    void moveFigure(int figureID, int newFieldID);
28
29    /**
30     * Moves the occupying figure back to its start position.
31     *
32     * @param figureID The ID of the occupying figure.
33     */
34    void moveOccupyingFigureToStart(int figureID);
35 }
```

code/GameService.java

Code 6.6: GameService [ChatGPT]

```
1 public interface DatabaseAccess {
2     /**
3      * Retrieves user data from the database based on the provided email.
4      *
5      * @param email The email of the user.
6      * @return User data including email and hashed password.
7      */
8     User retrieveUserData(String email);
9
10    /**
11     * Retrieves field occupancy status and the occupying figure ID.
12     *
13     * @param fieldID The ID of the field to check.
14     * @return Occupancy status and occupying figure ID.
15     */
16    OccupancyStatus checkFieldOccupied(int fieldID);
17
18    /**
19     * Updates the position of a figure in the database.
20     *
21     * @param figureID The ID of the figure.
22     * @param newFieldID The new field ID where the figure will be moved.
23     */
24    void updateFigurePosition(int figureID, int newFieldID);
25 }
```

code/DatabaseAccess.java

Code 6.7: DatabaseAccess [ChatGPT]

```
1 public class MoveResponse {
2     private boolean valid;
3     private boolean captured;
4     private int occupyingFigureID;
5
6     // Constructors, getters, and setters omitted for brevity
7 }
```

code/MoveResponse.java

Code 6.8: MoveResponse [ChatGPT]

```
1 public class OccupancyStatus {
2     private boolean occupied;
3     private int occupyingFigureID;
4
5     // Constructors , getters , and setters omitted for brevity
6 }
```

code/OccupancyStatus.java

Code 6.9: OccupancyStatus [ChatGPT]

Stichwortverzeichnis

Anforderungsspezifikation, 15, 21, 32
Anwendung von LLM Tools im Software Engineering, 19
Architekturdokument, 17, 22, 49
Ausblick und zukünftige Entwicklungen, 61

Besprechungsprotokoll, 13, 19, 26

Einleitung, 1

Grundlagen, 5
Grundlagen ChatGPT, 6
Grundlagen Google Gemini, 7
Grundlagen Le Chat, 10

Interfaces in Java, 56

Kritik an der Arbeit, 61

LLM Tools, 5

Motivation, 1

Praxisergebnisse und Vergleich, 25
Projekthandbuch, 13, 20, 28

Risikoliste, 14, 20, 29

Struktur der Arbeit, 2

Testspezifikation, 18, 22

Zielsetzung der Arbeit, 2
Zusammenfassung der Ergebnisse, 59
Zusammenfassung und Ausblick, 59

Literaturverzeichnis

- [1] Milfad Sadar. *Large Language Models - Grundlagen KI-getriebener Kommunikation*. Zugriff am 27. Mai 2024. 2023. URL: <https://weissenberg-group.de/was-ist-ein-large-language-model/#:~:text=Zum%20IPA%20Whitepaper-,Wie%20funktioniert%20ein%20LLM%3F,zwischen%20den%20Token%20zu%20ermitteln> (siehe Seite 6).
- [2] Jim Greene. *ChatGPT(software)*. Zugriff am 28. Mai 2024. 2024. URL: <https://eds.p.ebscohost.com/eds/detail/detail?vid=3&sid=1983b266-2f53-4090-a0f2-1aef00f51d1f%40redis&bdata=Jmxhbmc9ZGUmc2l0ZT1IZHMTbGl2ZQ%3d%3d#AN=163030996&db=ers> (siehe Seite 7).
- [3] Sundar Pichai und Demis Hassabis. *Gemini - unser größtes und leistungsfähigstes KI-Modell*. Zugriff am 29. Mai 2024. 2023. URL: <https://blog.google/intl/de-de/unternehmen/technologie/gemini/> (siehe Seite 7).
- [4] Dipl.-Ing. (FH) Stefan Luber. *Was ist Google Gemini?* Zugriff am 29. Mai 2024. 2024. URL: <https://www.bigdata-insider.de/was-ist-google-gemini-a-f855ef105c02f4533c18e913d4643007/> (siehe Seite 8).
- [5] Matthias Kindt.
Mistral AI: Französisches KI-Startup stellt Chatbot "Le Chat" und Kooperation mit Microsoft Azure vor. Zugriff am 16. Mai 2024. 2024. URL: <https://www.unidigital.news/mistral-ai-franzoesisches-ki-startup-stellt-chatbot-le-chat-und-kooperation-mit-microsoft-azure-vor/> (siehe Seite 10).
- [6] Ian Sommerville. *Software Engineering*. 9. Auflage. Harlow, England: Addison-Wesley, 2010. ISBN: 978-0-13-703515-1 (siehe Seite 12).
- [7] Andrea Baumann. *Vorlesung 04a, 05a, 05b, 06a, 08a, 11a*. Secure Software Engineering, Universität der Bundeswehr München. Abgerufen am 30. Juni 2023. 2023 (siehe Seite 12).

