

Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Benutzung von
Large-Language-Models für das
Formulieren der Anforderungen aus
Visionsvideos

Bachelorarbeit

im Studiengang Informatik

von

Alan Dryaev

Prüfer: Prof. Dr. rer. nat. Kurt Schneider
Zweitprüfer: Dr. rer. nat. Jil Ann-Christin Klünder
Betreuer: M. Sc. Jianwei Shi

Hannover, 18.12.2023

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 18.12.2023

Alan Dryaev

Zusammenfassung

Heutzutage spielt Software eine zentrale Rolle in vielen Aspekten unseres Lebens. Um die Qualität von Software zu gewährleisten ist die Erhebung von Anforderungen an diese Software essentiell. Damit die Anforderungen richtig erhoben werden, ist es wichtig mit den Stakeholdern zusammenzuarbeiten.

Eine Möglichkeit diese Zusammenarbeit zu erleichtern ist mit der Sprache Gherkin. Gherkin ist eine Sprache die es ermöglicht Anforderungen so zu formulieren, dass auch Stakeholder ohne viel technisches Wissen diese verstehen können. Diese spezifischen Anforderungen sind wie ein Testfall aufgebaut, weshalb sie hier auch als Testspezifikation bezeichnet werden. Eine Spezifikation sollte überarbeitet werden, wenn der Stakeholder mit dieser nicht einverstanden ist.

Die Erstellung und Verbesserung von Testspezifikationen ist ein aufwändiger Prozess. Mit den aktuellen Entwicklungen der "Large Language Models" in den letzten Jahren, wurde auch das Interesse einiger Forscher im Requirements Engineering geweckt. Das Ziel dieser Arbeit ist es einen Weg zu finden, die Arbeit der Anforderungsingenieure zu erleichtern. Dafür soll das "Large Language Model" ChatGPT zum Einsatz kommen. In dieser Arbeit wird ein Konzept zur Erstellung von Testspezifikationen mit ChatGPT entwickelt. Außerdem wird eine Studie durchgeführt, um die Leistung von ChatGPT bei der Verbesserung von Testspezifikationen zu untersuchen. Die Verbesserung erfolgt auf der Grundlage von Stakeholder Kommentaren.

In der folgenden Studie gibt es in den Ergebnissen eine Tendenz, dass ChatGPT mit einer manuellen Verbesserung mithalten kann. ChatGPT hat im Punkt Effizienz bessere Ergebnisse erzielen können. Andererseits wurde beobachtet, dass unter anderem mehr Syntax- und Formulierungsfehler mit ChatGPT gemacht wurden. Ein "Large Language Models" ist ein nützliches Werkzeug, das Anforderungsingenieure unterstützen kann, aber sie nicht ersetzen kann.

Abstract

These days Software plays a central role in many aspects of our life. To ensure the quality of software the elicitation of requirements is essential. It is important to work with the stakeholders so that the requirements are elicited correctly.

One way to ease the work with the stakeholders is to use Gherkin. With Gherkin you can formulate requirements, that even stakeholders with little technical knowledge can understand. These specific requirements are built like a test case, that's why it's also referred to as test specification. A specification should be revised if the stakeholder does not agree with it.

Creating and improving test specifications is a consuming task. With recent development in the last few years large language models have caught attention by researchers in requirements engineering. The goal of this thesis is to find a way to help the requirements engineers in their work. The large language model ChatGPT is used for this purpose. In this thesis a concept for creating test specification is developed. Additionally a study is conducted to investigate the performance of ChatGPT in improving test specifications. The improvements are based on stakeholder comments.

The result of the study shows a tendency that ChatGPT can keep up with manual processing of test specifications. ChatGPT had the better results in terms of efficiency. On the other hand ChatGPT made more syntax and formulation errors. A large language model is a useful tool that can help requirements engineers but it can't replace them.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Problemstellung	2
1.2	Lösungsansatz	2
1.3	Struktur der Arbeit	3
2	Grundlagen	5
2.1	Large Language Model (LLM)	5
2.1.1	ChatGPT	6
2.2	Gherkin	8
2.2.1	Aufbau von Gherkin Szenarien	8
2.2.2	Qualität von Gherkin Szenarien	10
2.3	User Stories	11
2.3.1	Aufbau von User Stories	11
2.3.2	Anforderungsanalyse	11
2.3.3	Qualität von User Stories	12
3	Verwandte Arbeiten	13
4	Konzepte	15
4.1	User Stories: Erstellung	15
4.2	Erstellung der Testspezifikation mit ChatGPT	16
4.2.1	Aufbau der Prompts	18
5	Planung der Studie	21
5.1	Goal Question Metric (GQM)	21
5.1.1	Ziele (G)	21
5.1.2	Fragen (Q) und Metriken (M)	22
5.2	Kontext	25
5.3	Hypothese	26
5.4	Variablen	27
5.5	Probandenwahl	28
5.6	Designwahl	28
5.7	Instrumentierung	29
5.8	Ablauf der Studie	29

5.8.1	Ablauf	30
5.8.2	Methode: Verbesserung von Testspezifikationen mit ChatGPT	31
5.8.3	Fehlerkorrektur	33
5.9	Threats to Validity	34
6	Ergebnisse der Studie	37
6.1	Ergebnisse und Interpretation	37
6.2	Diskussion	45
6.2.1	Forschungsfragen	45
6.2.2	Grenzen der Arbeit	46
7	Schlussfolgerung und Ausblick	47
A	Grundlagen	49
A.1	Kriterien für gute Gherkin Szenarien	49
A.2	Beispiel für die Umformulierung von mehreren Szenarien in ein Szenario	50
B	Konzepte	51
B.1	Erstellung der Testspezifikation mit ChatGPT	51
B.1.1	Beispiel: Erstellung der Testspezifikation mit ChatGPT (Dialog)	51
C	Studie	57
C.1	Demografische Daten	57
C.2	Beschreibung der Objekte der Studie	58
C.3	Konzept der Verbesserung mit ChatGPT	59
C.3.1	Beispiel: Methode Verbesserung der Testspezifikation mit ChatGPT (Dialog)	60
C.4	Umfragen zur Studie	62
C.5	Beispiel: Fehlerkorrektur der Fehlertypen	63
D	Ergebnisse und Auswertung	65

Kapitel 1

Einleitung

Requirements Engineering ist ein zentraler Aspekt der Softwareentwicklung. Die Hauptaufgabe eines Anforderungsingenieurs besteht darin, Anforderungen an eine Software zu erheben und zu verwalten. Dabei müssen die Wünsche der Stakeholder berücksichtigt werden. Die Erhebung von Anforderungen ist eine Aufgabe, die entscheidend für den Erfolg eines Projektes ist [1, 2]. Um die Qualität von Software zu gewährleisten, müssen die Anforderungen präzise formuliert werden.

Im Bereich des Requirements Engineering werden Anforderungen in enger Zusammenarbeit mit Stakeholdern formuliert. Dadurch werden die Stakeholder in den Entwicklungsprozess eingebunden. Um die Kommunikation mit den Stakeholdern zu erleichtern, sollten die Anforderungen so verständlich wie möglich sein [3].

Eine Möglichkeit solche Anforderungen zu formulieren, bietet die Sprache Gherkin. Sie ist eine natürliche Sprache, die es ermöglicht, Anforderungen präzise und anwendungsnah zu formulieren [4]. Ein Vorteil ist, dass Gherkin auch für Stakeholder ohne technisches Fachwissen verständlich ist. Ein weiterer praktischer Nutzen dieser Sprache ist, dass die Anforderungen weiterverarbeitet werden können. Dazu gehört die Automatisierung von Testfällen mit Cucumber [5].

Die Gherkin-Syntax wird in dieser Arbeit zur Formulierung der Anforderungen eines Visionsvideos genutzt. Visionsvideos sind kurze Videos, die eine Vision eines Projektes zeigen. Visionsvideos sollen eigentlich Anforderungen der Stakeholder visualisieren. In dieser Arbeit sollen Testspezifikationen mit Hilfe der Inhalte eines Visionsvideos erstellt werden.

Anforderungen für Software zu formulieren kann durchaus eine Herausforderung für Anforderungsingenieure darstellen. Für Anforderungsingenieure ist es zeitintensiv und aufwändig Testspezifikationen manuell zu erstellen. Wenn ein Stakeholder Einwände gegen die in der Testspezifikation implizit dargestellten Anforderungen hat, müssen diese angepasst werden.

Zur Erstellung und Verbesserung von Spezifikationen, werden für diese

Arbeit "Large Language Models" genutzt. Ein "Large Language Model" ist eine künstliche Intelligenz, die auf der Transformer-Architektur [6] basiert und Anforderungen in natürlicher Sprache bearbeitet. Einige Forscher haben bereits die Leistung von "Large Language Models" in verschiedenen Bereichen untersucht. Ein Beispiel für ein Sprachmodell ist ChatGPT, zu dem bereits mehrere Arbeiten für den Bereich Requirements Engineering veröffentlicht wurden [7, 8, 9]. In diesen Arbeiten wurde die Leistung von ChatGPT im Bereich des Requirements Engineering untersucht. In der folgenden Arbeit soll die Effizienz und Effektivität von ChatGPT untersucht werden. Die untersuchte Aufgabe ist das Erstellen und Verbessern von Testspezifikationen.

1.1 Problemstellung

Die Forschung in diesem Bereich zeigt, dass es ein Interesse darin gibt, die Anforderungsingenieure in ihrer Arbeit zu unterstützen. Es ist eine Herausforderung im Requirements Engineering die Anforderungen gut zu formulieren. In der Arbeit von Piam [10] gibt es zudem Hinweise darauf, dass es Stakeholdern schwer fällt gewisse Anforderungen zu erkennen. Die Erhebung von Anforderungen kann mit einem großen Zeitaufwand verbunden sein [1]. Außerdem ist es eine Herausforderung den Wunsch des Stakeholders nach seinen Vorstellungen zu erfüllen.

Mit dieser Arbeit, soll untersucht werden, ob ein "Large Language Model" die Anforderungsingenieure und Stakeholder bei der Formulierung von Anforderungen unterstützen kann. Es gibt Hinweise darauf, dass der Einsatz von ChatGPT eine Leistungserhöhung bringen kann, jedoch gibt es auch Beobachtungen darüber, dass ChatGPT Fehler macht [7]. In dieser Arbeit wird überprüft wie effizient und effektiv ChatGPT Testspezifikationen erstellen und verbessern kann. Außerdem werden Faktoren geprüft, die einen Einfluss auf die Akzeptanz der Stakeholder haben.

1.2 Lösungsansatz

Im Rahmen dieser Arbeit wird überprüft, ob ChatGPT Anforderungsingenieure bei der Erstellung und Verbesserung von Testspezifikationen unterstützen kann. Um diese These zu überprüfen wird eine Studie durchgeführt. Vorher wird ein Konzept zur Erstellung von Testspezifikationen mit ChatGPT vorgestellt. Als Grundlage für die Testspezifikation werden User Stories und ein Visionsvideo verwendet. In der Studie soll hingegen die Verbesserung von Testspezifikationen mit ChatGPT untersucht werden. Dafür werden das Visionsvideo und die mit ChatGPT erstellte Testspezifikation werden als Eingabe für ChatGPT verwendet. Da ChatGPT keine visuellen Eingaben erlaubt, wird zu diesem

Zweck das Video textuell beschrieben. Diese Studie vergleicht die zwei Methoden *mit ChatGPT* und *ohne ChatGPT*. Die Methode mit ChatGPT wird in Kapitel 5.8.2 vorgestellt. Bei der Methode ohne ChatGPT verbessert ein Anforderungsingenieur die Testspezifikationen manuell. Dadurch soll herausgefunden werden, wie ChatGPT mit limitiertem Einfluss eines Anforderungsingenieurs im Vergleich zu einer manuellen Verbesserung abschneidet. Das Ziel ist die Leistung von ChatGPT mit möglichst wenig Einfluss von außen zu untersuchen. Nun folgen die Forschungsfragen, die im Laufe dieser Arbeit beantwortet werden.

[RQ1]: Wie effizient ist ein "Large Language Model" bei der Verbesserung von Testspezifikationen?

[RQ2]: Wie effektiv ist ein "Large Language Model" bei der Verbesserung von Testspezifikationen?

[RQ3]: Welche Faktoren beeinflussen die Akzeptanz einer Testspezifikation durch Stakeholder?

1.3 Struktur der Arbeit

Diese Arbeit ist wie folgt strukturiert. In Kapitel 2 gibt es eine Einführung in die Themen "Large Language Model", Gherkin und User Stories. Dann folgt Kapitel 3, in dem die verwandten Arbeiten aufgelistet werden und begründet wird, inwiefern sich diese Arbeit unterscheidet. Anschließend wird in Kapitel 4 das Konzept zur Erstellung der Testspezifikationen mit ChatGPT vorgestellt. In Kapitel 5 wird die Studie geplant und der genaue Ablauf der Studie erklärt. In diesem Kapitel wird auch die Methode für die Studie mit ChatGPT vorgestellt. Im Anschluss folgt Kapitel 6 mit der Präsentation der Ergebnisse und der Interpretation der Studie. Dort werden ebenfalls die Ergebnisse nochmal diskutiert und die Grenzen der Studie aufgezeigt. Abschließend wird die Schlussfolgerung in Kapitel 7 getroffen und ein Ausblick für zukünftige Forschung gezeigt.

Kapitel 2

Grundlagen

In dem folgenden Kapitel werden Grundlagen und Technologien erklärt, die zum Verständnis der in dieser Arbeit vorgestellten Konzepte dienen. Im ersten Unterkapitel wird das Thema "Large Language Model" behandelt. "Large Language Models" sind Sprachmodelle, die künstliche Intelligenz nutzen, um menschliche Sprache zu verstehen und Ausgaben zu generieren. Anschließend werden die Grundlagen von Gherkin, einer natürlichen Sprache, die dazu dient, Anforderungen an Software zu formulieren erklärt. Abschließend werden User Stories, also kurze Beschreibungen, die als Anforderungen an Software in der agilen Entwicklung dienen, erklärt.

2.1 Large Language Model (LLM)

Ein "Large Language Model" (LLM) ist eine generative künstliche Intelligenz, deren Aufgabe es ist, den nächsten Token für einen bestimmten Kontext zu berechnen [6]. LLMs werden mit großen Mengen an Daten trainiert und konnten sich bereits in einigen Bereichen des Software Engineerings bewähren, wie zum Beispiel der Erstellung von Code [11]. Die Nutzung von LLMs wird für den Bereich der Verarbeitung natürlicher Sprachen (NLP) immer relevanter [7, 12]. LLMs können sogar seit der Einführung der Transformer-Architektur [6] in einigen Bereichen des NLP mithalten und mit ihren Ergebnissen auch mit traditionellen Methoden konkurrieren [13].

Um die Ergebnisse der LLMs zu verbessern nutzt man "Fine-Tuning". Es gibt mehrere Möglichkeiten für das "Fine-Tuning" eines LLMs für präzisere Antworten. Ab GPT-3 gibt es beispielsweise die Funktion des "Few-shot Learning", das heißt, dass das Sprachmodell anhand von Beispielen lernen kann. Mit Hilfe des Few-Shot Learning kann man LLMs so beeinflussen, dass sie Aufgaben in dem geforderten Bereich besser lösen können [14].

Eine weitere Möglichkeit für das Fine-Tuning ist das Manipulieren von Parametern eines LLMs. Solche Sprachmodelle haben Millionen bis Milliarden von Parameter [15]. Normalerweise kann man einige dieser

Parameter, sogenannte Hyperparameter, ändern. Die Manipulation solcher Parameter kann die Qualität der Ergebnisse des LLMs stark beeinflussen. Allerdings gibt es einige Einschränkungen zur Anpassung dieser Parameter. Einer dieser Hyperparameter ist die Temperatur. Die Temperatur lässt sich zum Beispiel in Bing durch die Wahl eines Unterhaltungsstils ändern. Es gibt die Wahl zwischen “kreativ”, “ausgewogen” und “genau”. In ChatGPT gibt es keine direkte Möglichkeit, die Hyperparameter zu ändern. Die Hyperparameter werden durch die Formulierung des Prompts beeinflusst. Die Wirkung der Hyperparameter auf die Ergebnisse des LLMs kann von Modell zu Modell variieren [15].

Für diese Arbeit wird ein Zero Shot Ansatz genutzt. Das bedeutet, dass das genutzte LLM vor Durchführung einer Aufgabe nicht mit einem Datensatz trainiert wird [16]. Des Weiteren können keine Parameter verändert werden und wenn dann nur indirekt, da für diese Arbeit ChatGPT genutzt wird. Trotzdem kann das genutzte LLM, in bestimmten Aufgabenbereichen, noch bessere Ergebnisse als aktuelle Modelle erzielen, selbst wenn man kaum Fine-Tuning betreibt [7, 16]. Die Ergebnisse eines LLMs kann man mit der Optimierung des Prompts durch Prompt Engineering verbessern [17, 18].

In anderen wissenschaftlichen Arbeiten werden die Vorteile, Limitierungen und Fehler von solchen Modellen in Bezug auf das Schreiben von Code geprüft. Die meisten Studien stimmen darüber ein, dass LLMs exzellent beim Schreiben von Code abschneiden, jedoch werden einige syntaktische Fehler gemacht [15].

Eines der aktuell mächtigsten LLMs ist GPT-4 [11]. Die aktuellste Version (Stand Dezember 2023) von ChatGPT basiert auf GPT-4. Um für diese Arbeit die Leistung von LLMs im Bereich des Requirement Engineering zu untersuchen kommt ChatGPT zum Einsatz. Allerdings wird die kostenlos verfügbare Version mit GPT-3.5 verwendet. In dem folgenden Abschnitt wird die Funktionsweise dieses LLMs und wieso es für diese Arbeit zum Einsatz kommt erläutert.

2.1.1 ChatGPT

ChatGPT [19] ist ein Large Language Model das auf Konversationen spezialisiert ist. Sowohl die Versionen von GPT als auch ChatGPT wurden von OpenAI entwickelt. Die aktuellste Version von ChatGPT basiert auf GPT-4. GPT-3 hat über 175 Milliarden Parameter und wurde mit Textdaten aus verschiedenen Quellen trainiert. ChatGPT wurde im Gegensatz zu GPT-3 jedoch vorallem mit konversationellen Daten trainiert [20]. Die erweiterte Version von GPT-3 ist GPT-3.5. Die Antworten der neuen Version sollen den Angaben von OpenAI nach effizienter, schneller und besser generiert werden [21].

ChatGPT kann durch Eingaben in Form von Prompts, dazugehörige Ausgaben in natürlicher Sprache und vollständigen Sätzen generieren. Dabei

wird durchgehend ein konversationeller Stil von ChatGPT genutzt, dadurch kann es zum Beispiel auf verschiedene Punkte der Eingabe eingehen. Dabei kann diese in verschiedenen Sprachen antworten und formuliert dabei vollständige Sätze [19].

Eine Konversation mit ChatGPT startet man, indem man eine Frage stellt. Darauf generiert ChatGPT eine Antwort welche die Frage beantworten soll. ChatGPT besitzt eine "Regenerate"-Funktion, mit der man die letzte Ausgabe von ChatGPT neu generieren kann. Diese Funktion wird in dieser Arbeit oft nur als Regenerate beschrieben. Durch einen Klick auf Regenerate entsteht eine neue Chatsektion. Daraufhin kann man zwischen diesen Sektionen wechseln und die Konversation in den anderen Sektionen fortführen. Die vorangegangenen Konversationen werden gespeichert und man kann somit immer auf diese zugreifen. Es ist klar, dass ChatGPT innerhalb einer Konversation lernt und sich sowohl die Eingaben, als auch die Ausgaben, merkt. Wobei unklar ist inwiefern die künstliche Intelligenz durch vorangegangene Konversationen lernt.

Es kann durchaus sinnvoll sein, die Ausgabe von ChatGPT zu verbessern, dafür gibt es Hinweise in Bezug auf GPT-4. Außerdem muss beachtet werden, dass syntaktische und semantische Fehler gemacht werden. Antworten von GPT-4 können auch generisch sein und nicht viel mit der direkten Lösung der Aufgabe zu tun haben [12]. Diese Punkte können im Laufe dieser Arbeit auch in ChatGPT-3.5 beobachtet werden.

Mit ChatGPT ist es, ähnlich wie bei anderen LLMs, schwer das selbe Ergebnis zwei mal zu erzielen [9]. Dadurch können die Ergebnisse von ChatGPT nicht reproduziert werden. Dieser Umstand muss bei der Durchführung und Auswertung einer Studie berücksichtigt werden.

Wie bereits erwähnt wird für diese Arbeit ChatGPT mit der Version GPT-3.5 verwendet. Der Grund für diese Entscheidung ist, dass es die beste Version von ChatGPT ist, die frei zugänglich ist. Außerdem hat eine eigene Auswertung gezeigt, dass ChatGPT am besten für das Konzept und die Studie geeignet ist. Die genutzte Version von GPT-3.5 wurde mit Daten aus der Zeit bis September 2021 trainiert.

In dieser Arbeit wird die Effektivität und Effizienz, mit der das LLM ChatGPT, Gherkin Szenarien verbessert, untersucht. Dabei wird unter anderem untersucht, welche Fehler von ChatGPT gemacht werden. Zur Untersuchung der Leistung von LLMs im Bereich von der Erstellung und Verbesserung von Testspezifikationen, wird in dieser Arbeit die Sprache Gherkin betrachtet. In Gherkin werden die Testspezifikationen in natürlicher Sprache geschrieben. ChatGPT ist ein Large Language Model, somit ist es darauf ausgerichtet Aufforderungen in natürlicher Sprache zu verarbeiten. Deshalb ist es sinnvoll die Leistung von ChatGPT für diese Aufgabe zu untersuchen. Um die Sprache Gherkin in ihrer Grundlage zu verstehen, werden die Grundprinzipien und Einschränkungen, die für diese Arbeit gelten, im folgenden Kapitel erklärt.

2.2 Gherkin

Gherkin ist eine Sprache, die von Softwareentwicklern genutzt wird, um ein gemeinsames Verständnis für die Anforderungen eines Projekts zwischen Stakeholdern und Entwicklern zu schaffen. Unter anderem kommt diese bei der verhaltensgetriebenen Entwicklung zum Einsatz und wird von dem Tool Cucumber interpretiert. Mit Gherkin werden Anforderungen in natürlicher Sprache formuliert, sodass auch Personen ohne technisches Fachwissen diese Anforderungen verstehen können. Im weiteren Kontext werden diese Anforderungen auch als Akzeptanztest für die Software genutzt. Insgesamt soll so das Verhalten der Software mit Testfällen abgedeckt werden.

2.2.1 Aufbau von Gherkin Szenarien

Beispiel für eine Gherkin-Feature-Datei (Auszug)

Feature: Artikel zum Warenkorb hinzufügen

Als Nutzer möchte ich einen Artikel zu meinem Warenkorb hinzufügen, damit ich den Artikel später kaufen kann.

Scenario: Artikel erfolgreich zum Warenkorb hinzufügen

Angenommen der Nutzer hat einen Artikel ausgewählt

Wenn der Nutzer auf 'Zum Warenkorb hinzufügen' klickt

Dann sollte der Artikel zum Warenkorb hinzugefügt werden

Und angezeigt werden, dass der Artikel erfolgreich zum Warenkorb hinzugefügt wurde.

In Gherkin gibt es einige Schlüsselwörter, die zur Formulierung von Anforderungen an Software genutzt werden. Insgesamt ist die Sprache Gherkin durch folgende Schlüsselwörter definiert:

Feature, Background, Scenario, Scenario Outline, Given, When, Then, And, But, |, "", # [5]

Folgende Schlüsselwörter werden in dieser Arbeit verwendet und sind zum Verständnis der Szenarios notwendig:

Feature, Scenario, Angenommen, Wenn, Dann, Und

Die Entscheidung der Kürzung der Anzahl der Schlüsselwörter wurde getroffen, um ein Szenario möglichst einfach zu gestalten. Außerdem wurden die Schlüsselwörter *Given, When, Then* und *And* ins Deutsche übersetzt, um den Lesefluss eines Szenarios nicht zu stören. *Scenario* und *Feature* behalten den englischen Wortlaut. Dadurch kann man bei der Verwendung von *Feature* und *Scenario* davon ausgehen, dass es sich um die Schlüsselwörter in Gherkin handelt. Im Folgenden werden die Schlüsselwörter und der Kontext, in dem sie zueinander stehen, erklärt.

Das Schlüsselwort *Feature* leitet eine Feature-Datei ein. Eine Feature-Datei deckt genau eine Funktion des Systems ab. In dieser Arbeit wird

der Begriff einer Testspezifikation für eine Sammlung von Feature-Dateien verwendet. Die Testspezifikation soll alle Funktionen der Software abdecken. Nach dem *Feature* wird der Titel und eine Narrative beschrieben. Der Titel dient als Überschrift und soll kurz eine Funktion des Systems beschreiben. Nach Chelinsky [5] gibt es keine strikte Regel, die besagt, was in einer Narrative stehen muss. In dieser Arbeit wird konsequent das Connextra Format, wodurch auch User Stories formuliert werden, zur Formulierung der Narrative benutzt. Somit ist im Rahmen dieser Arbeit eine Narrative durch eine User Story definiert.

Connextra Format

Als <Rolle>
möchte ich <Funktion>
sodass <Ziel> [5]

Eine Funktion der Software enthält eine Sammlung von Szenarien, die jeweils einen möglichen Ablauf dieser Funktion beschreiben. In Gherkin bedeutet das, dass nach dem Schlüsselwort *Feature* eine Aufzählung von Szenarien folgt, die mit dem Schlüsselwort *Scenario* beginnen. Dadurch kann man eine komplette Abdeckung aller Szenarien einer Funktion erzielen. Ein Szenario beschreibt einen Interaktionsablauf eines Features, der ein bestimmtes Ergebnis anstrebt. Ein Szenario besteht aus mehreren Schritten und eine Zeile des Szenarios wird auch als Schritt bezeichnet [5]. Die Schritte eines Interaktionsablaufs werden durch folgende Schlüsselwörter beschrieben: *Angenommen*, *Wenn*, *Dann*, *Und*.

Bedeutung der Schlüsselwörter für Schritte in Gherkin [4]:

Angenommen leitet eine *Annahme* des Systemzustands zu Beginn des Szenarios ein. Allerdings kann es auch eine Voraussetzung für das Szenario sein. Es beschreibt eine *Aktion/Reaktion*, die in der Vergangenheit passiert ist.

Wenn leitet ein/e *Aktion/Ereignis* des Szenarios ein. Die Aktion kann durch einen Nutzer oder durch ein anderes System ausgelöst werden.

Dann leitet eine erwartete *Reaktion* des Systems auf eine *Aktion/Ereignis* unter einer bestimmten *Annahme* ein.

Mit *Und* kann man das vorherige Schlüsselwort wiederholen. So kann eine weitere *Annahme/Aktion/Reaktion* beschrieben werden.

Außerdem wurde im Rahmen dieser Arbeit eine für Gherkin nicht unbedingt übliche Erweiterung gemacht. Jedes Szenario kann aus mehreren Wenn-Dann-Paaren bestehen. Normalerweise sollte man das Szenario so formulieren, dass ein Szenario nur eine Sache testet [22]. Diese Erweiterung hat

mehrere Gründe. ChatGPT gibt oft Szenarien mit mehreren Wenn-Dann-Paaren aus und diese einzuschränken hat oft keine Wirkung. Außerdem ist es für die Studie sinnvoll, diese Paare zu erlauben, um den Lesefluss eines Features nicht zu stören. Aus zwei Szenarien mit je einem Wenn-Dann-Paar wird ein Szenario mit zwei Wenn-Dann-Paaren. Zur Verdeutlichung gibt es Beispiel in Anhang A.2.

Es gibt mehrere Möglichkeiten für das Formulieren von Schritten in einem Szenario. Diese unterscheiden sich im Detailgrad. Es gibt zwei Stile zur Beschreibung eines Szenarios, man unterscheidet zwischen imperativ und deklarativ. Beim imperativen Stil werden die Schritte detaillierter beschrieben, es wird genau angegeben, was passiert und wie der Nutzer interagieren muss. Beim deklarativen Stil werden die Schritte allgemeiner formuliert und die genaue Interaktion soll dann im weiteren Verlauf der Entwicklung spezifiziert werden. Eine Mischung aus beidem soll nach Chelimsky [5] das Beste sein. Dennoch wird für diese Arbeit konsequent der imperative Stil verwendet. Dadurch ist gewährleistet, dass konsistent ein Stil genutzt wird und genau klar, was in jedem Schritt gemacht werden muss, um das Ziel eines Szenarios zu erreichen. Außerdem können durch einen imperativen Stil Inhalte des Visionsvideos besser umgesetzt werden.

2.2.2 Qualität von Gherkin Szenarien

Es gibt keine festen Kriterien für die Qualität eines Szenarios. Mindestens muss für ein klares Gherkin-Szenario die Syntax und Semantik richtig angewendet werden. Das bedeutet, dass die in Kapitel 2.2.1 beschriebenen Regeln eingehalten werden müssen. Zum einen heißt das, dass das vorgegebene Format als auch die richtige Verwendung der Schlüsselwörter eingehalten werden muss. Ein Beispiel für die falsche Nutzung eines Schlüsselwortes wäre, wenn man in einem Dann-Schritt eine Aktion eines Nutzers beschreibt (siehe Kapitel 5.8.3). Das ist insofern semantisch falsch, als dass in einem Dann-Schritt ausschließlich die Reaktion eines Systems oder das beobachtete Ergebnis dargestellt werden darf. Bezogen auf Testspezifikationen in der verhaltensgetriebenen Entwicklung gibt es einige Richtlinien, die von Binamungu et al. [22] zusammengefasst wurden. Sowohl die Qualität der Szenarien im Einzelnen, als auch die Szenarien in Bezug zueinander werden bewertet. Nicht alle von Binamungu et al. beschriebenen Kriterien wurden übernommen. In Anhang A.1 sind die Kriterien für gute Gherkin Szenarien mit Kommentaren. Außerdem sollte ein Szenario nur auf das System beschränkt sein, mit dem der Nutzer interagiert.

Im Rahmen dieser Arbeit wird der Fokus auf funktionale Anforderungen gelegt. Deshalb werden Anforderungen in Gherkin formuliert, die Funktionen eines Systems abdecken. Zur Erstellung von Gherkin Szenarien werden User Stories in ChatGPT eingegeben. User Stories sind Anforderungen in natürlicher Sprache. Deswegen werden diese hier verwendet um Anforderungen in

Gherkin zu formulieren. Um ein möglichst gutes Ergebnis mit ChatGPT zu erzielen, werden User Stories geschrieben, die bestimmten Qualitätskriterien entsprechen und möglichst alle Anforderungen an eine Anwendung abdecken. Dafür wird im nächsten Kapitel beschrieben was User Stories sind und wie man gute User Stories schreibt.

2.3 User Stories

User Stories werden im Requirements Engineering und der agilen Entwicklung genutzt. Diese bieten eine Möglichkeit grobe Anforderungen an eine Anwendung zu formulieren. Dabei sind User Stories kurze und prägnante Sätze, die Funktionen einer Anwendung, beschreiben. Die User Stories werden unter Betrachtung eines Visionsvideos erstellt. In diesem Fall sind die User Stories nicht die klassischen, welche man aus der agilen Entwicklung kennt, sondern speziell Anforderungen an eine Anwendung die in einem Visionsvideo zu erkennen sind. Normalerweise wird ein Visionsvideo genutzt um Anforderungen an eine Anwendung zu visualisieren und so die Vision der Stakeholder darzustellen. Dieser Aspekt sollte bei der Erstellung der User Stories berücksichtigt werden.

2.3.1 Aufbau von User Stories

Es gibt keine feste Richtlinie für die Formulierung von User Stories, jedoch hat sich eine Struktur etabliert, die hier als Connextra Format bezeichnet wird. Im Connextra Format wird eine Rolle beschrieben, die eine bestimmte Funktion und damit ein übergeordnetes Ziel erreichen möchte. Dabei ist die User Story wie bereits beschrieben in natürlicher Sprache geschrieben. Eine User Story wird um Akzeptanzkriterien erweitert, die beschreiben, wann eine User Story als umgesetzt gilt. Diese Akzeptanzkriterien sind detailliertere Anforderungen, die Abläufe des Systems beschreiben. Dabei kann es sich zum Beispiel um eine Aktion eines Nutzers handeln oder eine Reaktion des Systems. Der Nutzer nimmt eine Aktion im System vor und bekommt eine Reaktion, die das Ziel der User Story erfüllt.

Stories lassen sich in ihrer Granularität unterscheiden. Sehr grobe User Stories werden auch als Epics bezeichnet. Dadurch lässt sich eine Baumstruktur bilden, welche an der Spitze ein Epic enthält. Darunter werden dann User Stories angeordnet.

2.3.2 Anforderungsanalyse

Im folgenden Abschnitt wird die Anforderungsanalyse von Rupp [3] betrachtet, um eine Grundlage zur Erstellung von User Storys zu schaffen. Zuerst sollte man die Ursprungsanforderungen *separieren*, also eine Anforderung in

mehrere unterteilen. Dann kann man daraus die notwendigen Anforderungen *extrahieren*. Anschließend *abstrahiert* man Anforderungen, um größere Anforderungen zu finden (zum Beispiel Epics). Im nächsten Schritt *ergänzt* man fehlende Anforderungen auf den existierenden Ebenen. Diese Anforderungen *verfeinert* man dann und bekommt so aus den Anforderungen zum Beispiel neue User Stories. Letztlich *verbessert* man die Anforderungen bis sie den Qualitätsstandards entsprechen. Dieser Ablauf wird als Grundlage für den Prozess in Kapitel 4.1 genutzt.

2.3.3 Qualität von User Stories

Es gibt einige Kriterien um die Qualität einer User Story zu messen. Die wichtigsten Merkmale die in jeder User Story vorkommen müssen sind die Rolle, die Funktion und der Zweck. Außerdem wird für die Qualität der User Stories das INVEST-Prinzip, welches von Rupp [3] beschrieben wurde und außerdem das QUS Framework, welches von Lucassen et. al. [23] entwickelt wurde, betrachtet.

Kapitel 3

Verwandte Arbeiten

Im folgenden Kapitel werden verwandte Arbeiten und ihre Beobachtungen aufgezählt. Anschließend wird gezeigt, wie sich diese Arbeit von den anderen Arbeiten unterscheidet.

Mollah et al. [24] haben eine Methode entwickelt, mit der Testfälle für Webanwendungen in der agilen Entwicklung erstellt und implementiert werden. Dabei wird eine dreistufige Methode zur Ableitung von Testfällen aus User Stories vorgestellt. Die Autoren haben die Anwendbarkeit anhand von drei Fallstudien gezeigt.

In der Arbeit von Zhai et al. [25] wird das Werkzeug C2S vorgestellt. Die Erstellung von Spezifikationen soll mit diesem automatisiert werden. Dabei wandelt C2S Kommentare natürlicher Sprache in eine formale Programmspezifikation um. Dafür lernt es von anderen Spezifikationen und kann mit Hilfe von Kommentaren dadurch unvollständige Spezifikationen vervollständigen. Laut den Autoren konnte das Tool zeigen, dass Kommentare in bestehenden Projekten verwendet werden können, um Aspekte der Softwaretechnik zu verbessern. Dabei muss beachtet werden, dass die Autoren von hochwertigen Kommentaren ausgehen.

Xie et al. [8] führten eine Studie durch, um die Fähigkeiten eines LLM, zur Generierung von Software-Spezifikationen aus Kommentaren oder Dokumentation, zu bewerten. Dabei werden 15 verschiedene "Large Language Models" mit traditionellen Ansätzen verglichen. Xie et al. schlussfolgern, dass LLMs mit Few-Shot Learning traditionelle Methoden übertreffen können und dass Prompt Engineering das potential noch weiter ausschöpfen kann. Die beste Leistung konnte das Open-Source-Modell StarCoder zeigen. Das Problem ist jedoch, dass durch ineffektive Prompts und der Mangel an Wissen der LLMs viele Fehler entstehen. Die Autoren empfehlen hybride Ansätze zu verwenden, also traditionelle Methoden mit der Verwendung von LLMs zu kombinieren.

Chemnitz et al. [26] verfolgt einen Ansatz, bei dem Frontend Code für das Angular Framework automatisch generiert wird. Dabei werden

Testspezifikationen als Eingabe für ein "Large Language Model" genutzt. Das Ziel der Autoren ist, die Entwicklungszeit für Webanwendungen zu reduzieren und die Softwarequalität zu erhöhen.

In der Arbeit von Fantechi et al. [9] wird die Leistung von ChatGPT, bei der Erkennung von Inkonsistenzen in Anforderungen untersucht. Die Autoren können beobachten, dass ChatGPT Inkonsistenzfehler in Anforderungen erkennen kann, die selbst traditionelle Tools nicht oder nur schwer erkennen können. Sie befragen Studenten, die als Experten für die Studie agieren. Dabei wird beobachtet, dass die Experten in den geprüften Kategorien besser als ChatGPT abschneiden. Allerdings soll der Aufwand wesentlich höher sein Anforderungen manuell auf Inkonsistenzen zu prüfen. Trotz einiger Einschränkungen, wie dem Mangel an Erklärbarkeit und Transparenz, sehen die Autoren in ChatGPT eine Möglichkeit der Ergänzung manueller Analysen.

Ruan et al. [7] stellen in ihrer Arbeit ein Framework, das auf ChatGPT basiert, vor. Das Framework dient zur automatischen Generierung von Anforderungsmodellen. Der Fokus liegt bei der Erstellung effektiver Prompts um noch bessere Ergebnisse zu erzielen. In der Arbeit werden manuell erstellte Modelle mit ihrem Framework automatisch generierten Modellen verglichen. In ihrer Studie beobachten sie, dass ChatGPT Anforderungsmodelle schnell erstellen kann und Teile der Modelle werden korrekt extrahiert. Trotzdem benötigt man Erfahrung in der Modellierung von Anforderungen um die Ergebnisse von ChatGPT zu prüfen. Außerdem soll der Prompt einen wichtigen Bestandteil für gute Ergebnisse sein. Die Autoren sehen das Potenzial bei ChatGPT die manuelle Anforderungsmodellierung zu unterstützen und somit noch effizienter zu arbeiten.

In der Masterarbeit von Piam [10] hat dieser untersucht, wie man Anforderungen aus Visionsvideos erheben kann. Dabei werden Gherkin Testspezifikationen aus Visionsvideos hergeleitet. Den Stakeholdern ist es in seiner Studie schwer gefallen alle Anforderungen in den Visionsvideos zu erkennen. Der Autor kommt zum Ergebnis, dass die Verwendung von Gherkin Spezifikationen die Anzahl der durch Stakeholder formulierten Anforderungen erhöht. Dies wird von Piam so interpretiert, dass unklare Anforderungen in Visionsvideos durch die Darstellung in Gherkin, von Stakeholdern besser erkannt werden. Unter anderem wurden die Ergebnisse von Piam für diese Studie verwendet.

In dieser Arbeit wird die Idee von Piam erweitert. Es wird ein Konzept entworfen, mit dem man Anforderungen aus einem Visionsvideo erheben kann. Der neue Ansatz in dieser Arbeit ist, ein Large Language Model zur Erstellung von Testspezifikationen aus einem Visionsvideo und User Stories zu verwenden. Außerdem wird in einer Studie die Leistung von Large Language Models bei der Verbesserung von Testspezifikationen verglichen. Dabei liegt die Effektivität und Effizienz, mit der das Large Language Model diese verbessern kann, im Fokus. Ähnlich zu den anderen Arbeiten werden die Ausgaben des Large Language Models auf Fehler geprüft.

Kapitel 4

Konzepte

In diesem Kapitel wird das Konzept der Erstellung von Testspezifikationen mit ChatGPT vorgestellt. Nach diesem Konzept wurden die Testspezifikationen für die Studie erstellt. Zuerst wird jedoch die Erstellung von User Stories für diese Arbeit erläutert.

4.1 User Stories: Erstellung

In diesem Abschnitt wird der Prozess zur Erstellung von User Stories für dieses Projekt beschrieben. Aus der vorangegangenen Masterarbeit von Fokam Piam [10] wurden in der Benutzerstudie unter anderem Kommentare zu Visionsvideos geschrieben. In diesen Kommentaren sind Anforderungen aus einem Visionsvideo an CVParse beschrieben. Mit folgender Systematik (Abbildung 4.1) wurden für für diese Arbeit User Stories aus diesen Anforderungen formuliert:

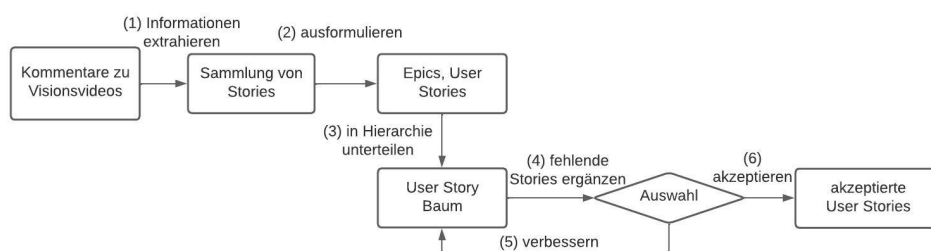


Abbildung 4.1: Prozess zur Erstellung von User Stories

Bevor man mit der Erstellung der User Stories beginnen kann, sammelt man alle Anforderungen aus den Kommentaren.

- (1) Extrahiere die wichtigsten Informationen aus den Anforderungen der Kommentare und unterteile diese in User Stories und Epics.

- (2) Formuliere vollständige User Stories und Epics mit den Informationen aus der Sammlung.
- (3) Unterteile die formulierten Epics und User Stories in eine Hierarchie.
- (4) Ergänze fehlende Stories. Prüfe dann, ob die User Stories den Auswahlkriterien entsprechen. Wenn die User Stories nicht akzeptiert werden, führe Schritt (5) durch. Wenn die User Stories akzeptiert werden, schließe den Prozess mit Schritt (6) ab.
- (5) Da die User Stories nicht akzeptiert wurden, benötigen diese noch eine Verbesserung. Somit verbessert man die User Stories und wiederholt Schritt (4).
- (6) Nun liegt als Ergebnis die fertigen Epics und User Stories des Visionsvideos vor.

4.2 Erstellung der Testspezifikation mit ChatGPT

Im folgenden Abschnitt wird das Konzept zur Erstellung einer Testspezifikation für bestimmte Anwendungen mit Nutzung von ChatGPT vorgestellt. Mit Hilfe dieses Konzepts erstellt man eine Grundlage für die Zusammenarbeit mit den Stakeholdern, indem man eine Testspezifikation erstellt, die in der Benutzerstudie von den Stakeholdern kommentiert werden soll. Nach der Ausführung der in diesem Konzept beschriebenen Schritte, soll der Anforderungsingenieur mit Hilfe von ChatGPT erste Feature-Dateien erstellen und dann zu einer Testspezifikation zusammenfassen. Der Anforderungsingenieur soll Eingabeaufforderungen in ChatGPT eingeben, sodass dieser als Ausgabe von ChatGPT die Feature-Dateien bekommt. Wenn die in der Feature-Datei enthaltenen Szenarien den Kriterien für qualitative Szenarien entsprechen, dann wird die Feature-Datei akzeptiert und zur Spezifikation hinzugefügt. Falls die Szenarien einer Feature-Datei nicht den Kriterien für qualitative Szenarien entsprechen, dann wird die Datei vorher verbessert oder eine andere Feature-Datei genommen. Insgesamt stellen nach diesem Prozess akzeptierte Feature-Dateien, die alle Funktionen der Anwendung abdecken zusammen eine Testspezifikation dar.

Der genaue Ablauf zur Erstellung der Testspezifikation mit ChatGPT ist in Abbildung 4.2 dargestellt. Vor dem Beginn, erstellt der Anforderungsingenieur einen Teil der Eingabedaten. Dafür schreibt der Anforderungsingenieur die User Stories der Anwendung nach Kapitel 4.1 und eine ausführliche Beschreibung der Anwendung. Die Beschreibung der Anwendung soll als Fließtext, welcher in natürlicher Sprache geschrieben ist, formuliert werden. Dieser soll möglichst alle Funktionen und die Oberfläche der Anwendung beschreiben. In dem Beispiel von Anhang B.1.1 wurde diese Beschreibung unter Betrachtung eines Visionsvideos geschrieben. Anschließend fügt man

die Beschreibung der Anwendung in den Prompt Kontext und die User Stories in den Prompt Aufgabe ein. Die Prompts sind im Detail in Kapitel 4.2.1 beschrieben. Als Nächstes wird zuerst der Prompt Kontext in ChatGPT eingegeben. ChatGPT akzeptiert den Prompt und gibt eine Ausgabe an, die für diese Aufgabe keine relevanten Informationen enthält. Wenn man daraufhin den Prompt Aufgabe einfügt, sollte man von ChatGPT eine Testspezifikation erhalten, welche auf den User Stories und Beschreibungen basiert. Ein Beispiel für den Ablauf bis zu diesem Punkt ist in Anhang B.1.1 dargestellt.

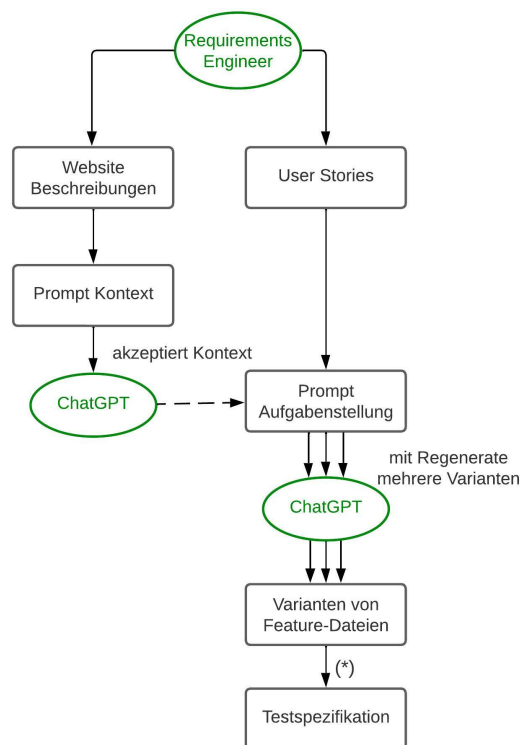


Abbildung 4.2: Erstellung der Testspezifikation mit ChatGPT. Der Schritt (*) beschreibt die Auswahl und Verbesserung der Feature-Dateien und ist in 4.3 beschrieben

Im letzten Schritt der Erstellung von Testspezifikationen mit ChatGPT (siehe Abbildung 4.3) werden die mit der Regenerate-Funktion von ChatGPT erzeugten Feature-Dateien verglichen. Man wählt zuerst jeweils das Feature, welches in einer Feature-Datei am besten umgesetzt wurde. Wenn zum Beispiel das Feature 3 in Feature-Datei 4 besser umgesetzt wurde, als in Feature-Datei 1,2 und 3, dann wählt man das Feature 3 aus Feature-Datei 4. Das Feature kann man im Anschluss dann noch mit ChatGPT oder direkt überarbeiten und dann zur Testspezifikation

hinzufügen. Durch das Hinzufügen eines Features zur Testspezifikation gilt das Feature als vom Anforderungsingenieur akzeptiert. Eine so vollständig erstellte Testspezifikation gilt somit auch als akzeptiert.

Beispiel für (*) für eine Testspezifikation mit 4 Features

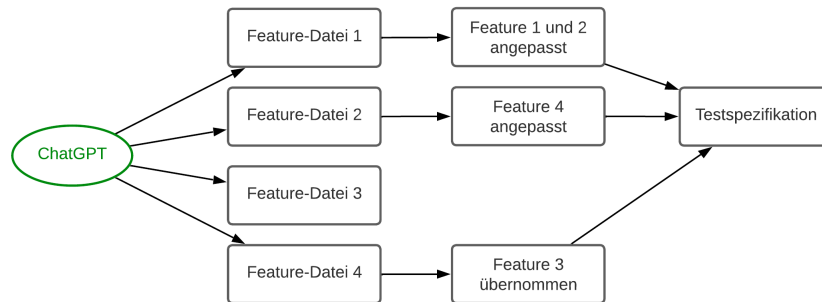


Abbildung 4.3: Beispiel für den letzten Schritt (*) aus Abbildung 4.1, Selektion und Anpassung der Features

4.2.1 Aufbau der Prompts

Im weiteren Verlauf wird der Ausdruck Prompt für eine Eingabeaufforderung genutzt, die in natürlicher Sprache geschrieben wurde, aber auch Teile besitzen kann, welche reine Informationen in Form von Daten darstellen. Dieser Prompt soll als Eingabe in ChatGPT genutzt werden.

Die Qualität der Ausgabe hängt unter anderem von der Qualität der Prompts ab. Außerdem schwankt die Qualität bei mehreren separaten Anfragen trotz identischer Prompts. Diese zwei Aspekte werden eingebunden, indem man den Fokus auf den Prompt legt und mit der Regenerate Funktion von ChatGPT mehrere Varianten von Ausgaben generiert. Um die notwendigen Eingaben und die Eingaben für eine Verbesserung der Antwort von ChatGPT herauszufinden, wurden mit Hilfe verschiedener Quellen [7, 16, 5, 22, 27] und der eigenen Auswertung von Ergebnissen mit ChatGPT zwei Prompts entwickelt. Dabei wurde zuerst eine erste Version erstellt, die nur eine kurze Beschreibung der Anwendung und die User Stories enthielt. Dann wurde inkrementell mit Hilfe der Ausgaben von ChatGPT dieser Entwurf immer weiter verfeinert, bis das Konzept dass in diesem Kapitel beschrieben wird, entstanden ist. Dabei wurden wichtige Eingaben und Kriterien für qualitative Feature-Dateien gesammelt.

Für den Prompt müssen wie bereits beschrieben folgende Eingaben vom Anforderungsingenieur erstellt werden: User Stories und eine Beschreibung

der Anwendung. Der Prompt ist in zwei Teile aufgeteilt, um den Kontext von der Aufgabenstellung zu trennen und einen einzelnen Prompt nicht zu lang zu gestalten. Im ersten Prompt wird ein inhaltlicher Kontext gegeben. Dieser Kontext beinhaltet die allgemeine Funktion der Anwendung und eine detaillierte Beschreibung aller Funktionen, die abgedeckt werden sollen. Im zweiten Prompt wird die Aufgabe gestellt. Hier wird ChatGPT eine Rolle, die Aufgabe, User Stories, das vorgegebene Format und ein ausführliches Beispiel gegeben. Es handelt sich um eine Zero Shot Anfrage, da es vorher kein Beispiel in Form von Eingabe und geforderter Ausgabe gibt. Das bedeutet, dass ChatGPT nicht mit einem Datensatz vorher trainiert wurde. Die verschiedenen Aspekte des Prompts benötigen eine genaue Beschreibung, um ihre Funktion und Nutzen für den Prompt zu verdeutlichen.

Beide Prompts werden als Template (siehe unten) und mit eingefügten Daten (Anhang B.1.1) dargestellt. Die daraus resultierende Testspezifikation soll dann im Laufe der Studie mit Kommentaren von Stakeholdern verbessert und erweitert werden. Für die Verbesserung der Testspezifikation mit ChatGPT wird dafür ein Konzept entwickelt.

Erstellung von Testspezifikationen

Prompt Kontext

Auf der Webseite *[Name der Anwendung]* kann man *[Funktion der Anwendung]*. Diese Webseite ist in Features mit folgenden Beschreibungen unterteilt:

"""

[Beschreibung der Anwendung]

"""

Prompt Aufgabe

Gegeben seien die Beschreibungen von *[Name der Anwendung]*. Nun wird eine Testspezifikation in Gherkin zu allen Features der Anwendung benötigt.

Du bist ein *[Rolle]* und sollst mit Hilfe von User Stories und den Beschreibungen Feature-Dateien in Gherkin schreiben.

"""

[User Stories]

"""

Die Feature-Dateien müssen in Gherkin-Code ausgegeben werden. Eine Feature-Datei hat genau ein Feature und besteht aus mehreren Szenarien. Ein Szenario soll dabei den Ablauf für das Feature beschreiben. Es müssen folgende Gherkin keywords verwendet werden: Feature, Background, Scenario, Scenario Outline, Wenn, Dann, Und, Aber;

Folgendes Format muss für einen Gherkin Test eingehalten werden:

```

"""
Feature <y>: <Feature>
<User Story>
Scenario <z>: <Titel>
Angenommen <Vor Interaktion gegeben>
Wenn <Aktionen des Nutzer>
Dann <Reaktionen des Systems>
"""

```

Beispiel 1:

```

"""
[Beispiel]
"""

```

Das "Beispiel 1" ist kein inhaltlicher Kontext für *[Anwendung]*.

Beschreibung der Anwendung: Ist eine ausführliche Beschreibung der Funktionen der Anwendung. Die Beschreibung soll ein Fließtext in natürlicher Sprache sein, der zum Beispiel einen Ablauf mit der Anwendung beschreibt und der von einem Stakeholder geschrieben werden kann. In diesem Fall wurde eine Beschreibung mit Hilfe eines Visionsvideos erstellt. Dabei wurden die Interaktion und die Oberfläche in einem Fließtext beschrieben.

User Stories: Eine Sammlung von User Stories, zu welchen die Feature-Dateien geschrieben werden sollen.

Beispiel: Das Beispiel dient der Genauigkeit der Antwort. Die Feature-Dateien sollen in diesem Fall im imperativen Format geschrieben werden, also jeder Schritt soll genau beschrieben werden. Dafür wurde ein Beispiel für das Hinzufügen eines Artikels in den Warenkorb gegeben.

Rolle: Die Rolle soll ChatGPT in seiner Antwort beeinflussen. Für dieses Projekt wurde die Rolle "Requirements Engineer" zugewiesen, sodass ChatGPT möglichst mit der Qualität eines Anforderungsingenieurs antworten soll.

Kapitel 5

Planung der Studie

Im folgenden Kapitel wird die Studie geplant. Zur Planung der Studie werden die Schritte, welche von Wohlin et al. [28] beschrieben wurden, systematisch abgearbeitet. Dabei wird auch die Goal Question Metric (GQM) Methode verwendet, welche von Basili et al. [29] erfunden wurde. Im Laufe dieser Studie werden die zwei Methoden *mit ChatGPT* und *ohne ChatGPT* untersucht. Anschließend wird der genaue Ablauf der Studie beschrieben und die Methode *mit ChatGPT* vorgestellt.

5.1 Goal Question Metric (GQM)

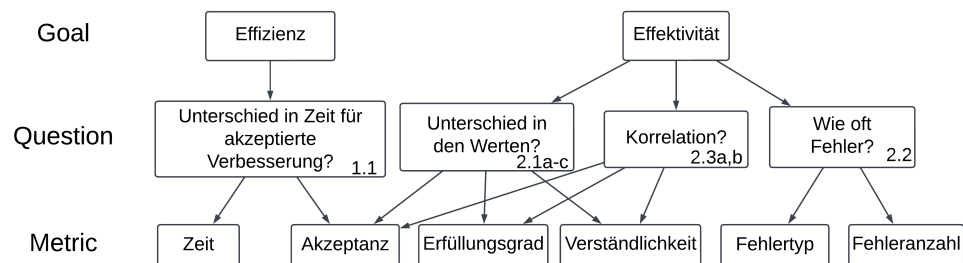


Abbildung 5.1: Goal Question Metric für die Ziele der Studie (Fragen sind abgekürzt)

5.1.1 Ziele (G)

Im ersten Unterkapitel werden die Ziele der Studie definiert. Um die Studie planen zu können, müssen die Ziele der Studie genau definiert sein. Das übergeordnete Ziel der Studie ist es, die Forschungsfragen dieser Arbeit zu beantworten.

Nun werden die Ziele definiert, die sich auf die Forschungsfragen der Arbeit (Kapitel 1.2) fokussieren sollen. Dabei wird beachtet, dass die in

GQM definierten Fragen, diese Forschungsfragen beantworten sollen. Die Ziele werden mit Hilfe eines Templates [29] aufgestellt.

Ziel 1

Analysiere die Nutzung von ChatGPT mit dem Ziel, die Verbesserung von Testspezifikationen nach einer Iteration basierend auf Kommentaren von einem Stakeholder zu untersuchen

in Bezug auf Effizienz

Vom Blickwinkel eines Anforderungsingeeurs mit Erfahrung in ChatGPT und Gherkin

Im Kontext eines Experiments

Ziel 2

Analysiere die Nutzung von ChatGPT mit dem Ziel, die Verbesserung von Testspezifikationen nach einer Iteration basierend auf Kommentaren von einem Stakeholder zu untersuchen

in Bezug auf Effektivität

Vom Blickwinkel eines Stakeholders und eines Anforderungsingenieurs mit Erfahrung in ChatGPT und Gherkin

Im Kontext eines Experiments

Für das Experiment wird nur eine Gruppe an Subjekten betrachtet. Einem Probanden wird jeweils eine Verbesserung *mit ChatGPT*, sowie eine Verbesserung *ohne ChatGPT* gegeben. Diese müssen dann betrachtet und bewertet werden. Bei der Studie handelt es sich um eine "Blocked subject-object study" [28]. Das bedeutet, dass mehrere Subjekte im Kontext mehrerer Objekte untersucht werden. In diesem Fall bekommen mehrere Probanden jeweils zwei Objekte, also jeweils eine Verbesserung die *mit ChatGPT* und *ohne ChatGPT* gefertigt wurde. Dieser Entwurf wurde gewählt um eine Vergleichbarkeit der Ergebnisse der Objekte *mit ChatGPT* und *ohne ChatGPT* zu schaffen. Für diese Studie werden die Probanden zufällig gewählt (Kapitel 5.5). Außerdem gibt es eine Variation in der Reihenfolge der Präsentation der Objekte.

5.1.2 Fragen (Q) und Metriken (M)

Frage 1.1

Wird *mit ChatGPT* weniger Zeit als *ohne ChatGPT* für eine akzeptierte Testspezifikation benötigt?

Metrik 1.1:

$$a\text{-Zeit} = \frac{g\text{-Zeit}}{\#a\text{-Verbesserungen}}$$

Die Zeit, die für eine akzeptierte Spezifikation benötigt wird. Sie setzt

sich zusammen aus:

- *g-Zeit*: Die Zeit, die für die Verbesserung der Testspezifikation benötigt wurde.
- *#a-Verbesserungen*: Die Anzahl der akzeptierten Verbesserungen.

Akzeptanz - Wahrheitswert für die Akzeptanz. Für die *Akzeptanz* wird für beide Methoden die folgende Aussage getätigt: "Ich akzeptiere Verbesserung x". Diese Aussage kann mit "Ja" akzeptiert oder "Nein" abgelehnt werden.

Um die in der Formel enthaltenen Variablen zu erhalten, wird zum Einen die Zeit zur Erstellung der Verbesserung benötigt. Dafür wird die Zeit gemessen, die von Beginn der Bearbeitung der Testspezifikation, bis zur letzten Änderung an der Testspezifikation vergeht.

Außerdem muss die Anzahl akzeptierter Verbesserungen gezählt werden. Um die *#a-Verbesserungen* zu bestimmen, muss die *Akzeptanz* gemessen werden. Die Summe aller akzeptierter Verbesserungen über einen Probanden ergibt somit *#a-Verbesserungen*.

Insgesamt wird mit dieser Metrik die Anzahl der Verbesserungen, die in einer bestimmten Zeit akzeptiert wurden, berechnet. Die Ergebnisse der zwei Methoden *mit ChatGPT* und *ohne ChatGPT* sollen mit dem Mann-Whitney-U-Test auf einen signifikanten Unterschied geprüft werden. Für die Beurteilung der Effizienz wird Meinung des Stakeholders berücksichtigt, da die akzeptierten Verbesserungen gezählt werden.

Frage 2.1a

Wie unterscheidet sich der Erfüllungsgrad der Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*?

Metrik 2.1a:

Erfüllungsgrad - Grad zu dem der Proband findet, dass sein Kommentar berücksichtigt wurde. Dieser Wert wird mit einer Likert-Skala, die die Zustimmung von 1-6 misst, bestimmt. Die Aussage für den Erfüllungsgrad lautet:

Die Verbesserung hat meinen Kommentar gut berücksichtigt.

Für diese Formel wird noch die Gesamtzahl der Kommentare eines Probanden für *#Verbesserungen* gezählt.

Frage 2.1b

Wie unterscheidet sich die Verständlichkeit der verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*?

Metrik 2.1b:

Verständlichkeit - Grad zu dem der Proband die Verbesserung versteht. Dieser Wert wird auch mit einer Likert-Skala, die die Zustimmung von 1-6 misst, bestimmt. Die Aussage für die Verständlichkeit lautet:
Die Verbesserung ist gut formuliert und leicht zu verstehen.

Frage 2.1c

Wie unterscheidet sich die Akzeptanz der verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*?

Metrik 2.1c:

$$\text{Akzeptanzquote} = \frac{\#a\text{-Verbesserungen}}{\#\text{Verbesserungen}}$$

Akzeptanzquote - Verhältnis der akzeptierten Verbesserungen zu gesamter Anzahl an Verbesserungen pro Proband.

Die für diese Studie verwendete Likert Skala ist in Anhang C.3 definiert. Die erreichten Werte werden dann jeweils für *Akzeptanz*, *Verständlichkeit* und *Zufriedenheit* der Methoden *mit ChatGPT* und *ohne ChatGPT* in einer Tabelle gesammelt. Dann werden die Ergebnisse aller Teilnehmer, zwischen den zwei "Gruppen" *mit ChatGPT* und *ohne ChatGPT*, mit dem Mann-Whitney-U-Test evaluiert.

Frage 2.2

Wie oft werden bestimmte Fehler *mit ChatGPT* und *ohne ChatGPT* gemacht?

Metrik 2.2

Fehlertyp - Art des Fehlers

Fehleranzahl - Anzahl von Fehlern in einer Spezifikation.

Die Anzahl der Fehler werden jeweils für jede Zeile berechnet. Falls der selbe Fehler mehrmals in einem Szenario vorkommt, wird dieser nur ein mal gezählt. Zur Vereinfachung der Zählung der Fehler, wird die *Fehleranzahl* der folgenden *Fehlertypen* gezählt:

Syntaxfehler - Verletzung des Formats aus Kapitel 2.2.1

Formulierungsfehler - Fehler in der Formulierung innerhalb der Schritte (siehe Kapitel 2.2.1 unter Bedeutung der Schlüsselwörter)

Und-Fehler - Gherkin Schlüsselwort *Und* wird nicht verwendet, obwohl es nötig wäre.

Frage 2.3a

Wie hängt Erfüllungsgrad mit Akzeptanz zusammen?

Metrik 2.3a

Erfüllungsgrad - siehe Metrik 2.1

Akzeptanz - Wahrheitswert für die Akzeptanz.

Frage 2.3b

Wie hängt Verständlichkeit mit Akzeptanz zusammen?

Metrik 2.3b

Verständlichkeit - siehe Metrik 2.1

Akzeptanz - siehe Metrik 1.1

Um die Signifikanz der Korrelation zu überprüfen, wird eine biserale Rangkorrelation durchgeführt. Dabei sind Verständlichkeit und Erfüllungsgrad die ordinalen Skalen und Akzeptanz das dichotome Merkmal durchgeführt. Zuerst werden beide Datensätze von *mit ChatGPT* und *ohne ChatGPT* zusammengeführt. Dann wird die Korrelation von Verständlichkeit mit Akzeptanz und Erfüllungsgrad mit Akzeptanz durchgeführt.

Als Nächstes wird der Kontext definiert in dem die Studie stattfindet.

5.2 Kontext

Für den Kontext der Studie sollte sich damit auseinandergesetzt werden, unter welchen Umständen die Studie durchgeführt wird.

Die Studie wird vollständig Online durchgeführt. Des Weiteren handelt es sich bei den Teilnehmenden um Studenten und Auszubildende, welche durch öffentliche Aushänge geworben werden. Die Anwendung, deren Testspezifikation für diese Studie betrachtet wird, heißt CVParser. CVParser ist ein Projekt, das im Rahmen des Softwareprojekts 2021 an der Leibniz Universität Hannover durch ein Visionsvideo vorgestellt wurde. Somit ist CVParser nur ein Prototyp, dessen Funktionen eingeschränkt sind. Aus diesem Prototyp werden dann 6 Szenarien in einer Testspezifikation gesammelt und den Probanden präsentiert. Bei der Testspezifikation handelt es sich somit um ein "toy problem"[28], also ein Problem, was nicht von realem Ausmaß ist. Trotzdem ist die Testspezifikation dadurch auch für Studierende leichter zu verstehen. Insgesamt ist es eine allgemeine Studie, denn es wird allgemein untersucht, ob die Nutzung von ChatGPT eine Verbesserung in bestimmten Bereichen erreichen kann. Die Studie wird auf der Online-Präsentationsplattform BigBlueButton durchgeführt und das Visionsvideo wird dort gezeigt. Zuerst werden Kommentare von den Stakeholdern in einer Online-Umfrage eingetragen. Dann werden die Verbesserungen *mit ChatGPT* und *ohne ChatGPT*

erstellt. Die Verbesserungen der Testspezifikation werden in einer Google Tabelle gegenübergestellt. Dabei soll der Proband, unter Betrachtung der Verbesserungen, Bewertungen in einer Online-Umfrage verteilen. Der Zeitraum für die Durchführung des Experiments mit einem Probanden wird auf einen Tag beschränkt. Dabei findet zuerst ein erster Termin statt, an dem Kommentare geschrieben werden und innerhalb von 25 Stunden ein zweiter Termin, an dem die Ergebnisse bewertet werden sollen.

5.3 Hypothese

In dem Kapitel 5.1 zu GQM wurden bereits die Ziele definiert. Das erste Ziel besteht darin, die Effizienz mit der ChatGPT Testspezifikationen verbessern kann, zu untersuchen. Für die Effizienz wird die Zeit für eine akzeptierte Spezifikation (Metrik 1.1) gemessen. Die Hypothese für Ziel 1 wird folgendermaßen definiert:

Nullhypothese [H1.0]:

Das Erstellen von Verbesserungen der Testspezifikationen nach einer Iteration *mit ChatGPT* und *ohne ChatGPT* unterscheidet sich nicht in der Effizienz.

Hypothese [H1.1]:

Es gibt einen Unterschied in der Effizienz bei der Verbesserung der Testspezifikationen nach einer Iteration *mit ChatGPT* und *ohne ChatGPT*.

Um die Nullhypothese zu verwerfen muss es somit einen signifikanten Unterschied in *a-zeit ohne ChatGPT* und *mit ChatGPT* geben.

Das zweite Ziel besteht darin die Effektivität mit der ChatGPT die Testspezifikation verbessert zu untersuchen. Für die Effektivität werden die Werte Erfüllungsgrad, Verständlichkeit, Akzeptanz (Metrik 2.1a-c) in den Gruppen *ohne ChatGPT* und *mit ChatGPT* gemessen. Auf dieser Grundlage werden mehrere Hypothesen formuliert:

Nullhypothese [H2.1a.0]: Die verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT* unterscheiden sich nicht im *Erfüllungsgrad*.

Hypothese [H2.1a.1] Es gibt einen Unterschied in dem *Erfüllungsgrad* der verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*

Nullhypothese [H2.1b.0]: Die verbesserten Testspezifikationen

mit *ChatGPT* und *ohne ChatGPT* unterscheiden sich nicht in der *Verständlichkeit*.

Hypothese [H2.1b.1] Es gibt einen Unterschied in der *Verständlichkeit* der verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*

Nullhypothese [H2.1c.0]: Die verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT* unterscheiden sich nicht in der *Akzeptanz*.

Hypothese [H2.1c.1] Es gibt einen Unterschied in der *Akzeptanz* der verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*

Um eine der Nullhypothesen zu verwerfen muss somit in einer der Kategorien *Erfüllungsgrad*, *Verständlichkeit* oder *Akzeptanz* ein signifikanter Unterschied gezeigt werden. Das erfolgt wie in Metrik 2.1 beschrieben, mit dem Mann-Whitney-U-Test.

Zusätzlich wird noch eine Hypothese zu der Untersuchung der Korrelation von *Verständlichkeit* mit *Akzeptanz* und *Erfüllungsgrad* mit *Akzeptanz* formuliert (Metrik 2.3a-b):

Nullhypothese [H2.3a.0] Es gibt keine Korrelation von *Erfüllungsgrad* mit *Akzeptanz*

Hypothese [H2.3a.1]: Es gibt eine Korrelation von *Erfüllungsgrad* mit *Akzeptanz*

Nullhypothese [H2.3b.0] Es gibt keine Korrelation von *Verständlichkeit* mit *Akzeptanz*

Hypothese [H2.3b.1]: Es gibt eine Korrelation von *Verständlichkeit* mit *Akzeptanz*

Um diese Nullhypothesen zu prüfen, wird eine biserale Rangkorrelation durchgeführt. Wenn es eine Korrelation gibt, kann die Nullhypothese dementsprechend verworfen werden.

5.4 Variablen

Vor dem Entwurf der Studie werden nun die unabhängigen und abhängigen Variablen gewählt. Die unabhängige Variable ist die Verwendung von *ChatGPT*. Diese unterscheidet sich in den Methoden *mit ChatGPT* und *ohne ChatGPT*. Die abhängigen Variablen sind die Variablen der Metriken, die in Kapitel 5.1 definiert wurden.

5.5 Probandenwahl

Die Auswahl der Probanden ist ein wichtiger Prozess, um das Ergebnis der Studie verallgemeinern zu können. Daher sollten die gewählten Probanden eine Stichprobe der Gesamtheit darstellen. Diese Stichprobe soll dann repräsentativ für Stakeholder der Softwareentwicklung sein. Nun wird beschrieben wie die Probanden für die Studie gewählt wurden. In diesem Fall wurde nach Studenten und Auszubildenden gesucht, also nach "Convenience Sampling". Das bedeutet, dass Probanden gewählt wurden, die am Nächsten zur Verfügung standen. Die Probanden wurden durch öffentliche Aushänge, persönliche Gespräche und Präsentationen geworben.

Anfangs waren zehn Probanden für die Studie geplant. Allerdings wurde eine gleiche Verteilung der Probanden auf die zwei Anforderungsingenieure in der Studie angestrebt. Deshalb wurden noch zwei weitere Probanden geworben, sodass man nach Abschluss der Studie auf insgesamt zwölf Probanden kommt. Das erste Kriterium war das Interesse aus der Sicht eines Stakeholders an der Anwendung CVParser. Das zweite Kriterium war das Bestehen eines Tests zur Sprache Gherkin. Wenn beide Kriterien erfüllt waren, konnten die Probanden an der Studie teilnehmen. Eine demografische Verteilung der Probanden befindet sich in Anhang C.1.

5.6 Designwahl

Die Probanden sind in zwei Gruppen unterteilt. Beide Methoden *mit ChatGPT* und *ohne ChatGPT* werden jeweils an jeder Gruppe durchgeführt. Dabei unterscheidet sich der Ingenieur, der die Verbesserung für die Gruppe mit der jeweiligen Methode durchführt. Die Verteilung der Gruppen erfolgte anhand folgender Tabelle:

	Probandennummer	
	Gruppe 1: 1,2,3,4,5,12	Gruppe 2: 6,7,8,9,10,11
Ingenieur 1	mit ChatGPT	ohne ChatGPT
Ingenieur 2	ohne ChatGPT	mit ChatGPT

Tabelle 5.1: Übersicht über Verteilung der Probanden auf die Ingenieure und der Verwendung von ChatGPT

Ingenieur 1 ist der Autor dieser Bachelorarbeit, Alan Dryaev und Ingenieur 2 ist Jianwei Shi der wissenschaftliche Mitarbeiter, der den Autoren in seiner Arbeit betreut hat. Zusätzlich wird bei Gruppe 1 die Verbesserung *mit ChatGPT* und bei Gruppe 2 *ohne ChatGPT*, als erstes präsentiert. Die Probanden sollen einen Kommentar zu einer Testspezifikation schreiben und anschließend die Ergebnisse der Verbesserungen *mit ChatGPT* und *ohne ChatGPT* bewerten. Um ein ausgewogenes Design zu erreichen, werden beiden Anforderungsingenieuren *mit ChatGPT* und *ohne ChatGPT* die selben

Kommentare gegeben. In den beiden Gruppen wird der selbe Kommentar genutzt, um die Testspezifikation zu verbessern. Dadurch wird verhindert, dass es durch die individuelle Erfahrung einen Einfluss auf die Kommentare gibt. Für die Randomisierung wird bei der Bewertung der Verbesserungen nicht angezeigt, welche der beiden Verbesserungen *mit ChatGPT* oder *ohne ChatGPT* erstellt wurde. Zusätzlich wird die Reihenfolge der beiden Verbesserungen variiert. In der Studie wird der Fokus auf die *Verbesserung der Testspezifikation* gelegt. Dafür wird dieser Prozess unter der Verwendung der beiden Methoden *mit ChatGPT* und *ohne ChatGPT* untersucht. Die ursprüngliche Testspezifikation an sich ist für die Evaluation nicht relevant.

5.7 Instrumentierung

Das Ziel der Instrumentierung ist, die Mittel, die für die Durchführung der Studie relevant sind, zu identifizieren. Diese Mittel sollten so gewählt werden, dass das Ergebnis dadurch nicht beeinträchtigt wird [28]. Deshalb erfolgt nun eine Identifizierung der Objekte, Richtlinien und Messinstrumente dieser Studie.

Objekte: Testspezifikation zu CVParse, Visionsvideo zu CVParse, ChatGPT, Kommentare der Stakeholder, Tabelle der Verbesserungen, LimeSurvey und BigBlueButton (Eine detaillierte Beschreibung der Objekte ist in Anhang C.2)

Richtlinien

Es gibt eine Richtlinie für die Probanden, damit diese genau wissen, was ihre Aufgabe während Studie ist. Dafür gibt es im Aushang an die Studenten eine Beschreibung zur Studie. Außerdem gibt es auf dem ersten Fragebogen einen Fragenkatalog (siehe Anhang C.2), der die Probanden beim Schreiben von Kommentaren unterstützen soll.

Messinstrumente

Um die Daten einer Studie zu erheben, benötigt man Messinstrumente. Die für die Studie relevanten Daten werden mit Hilfe von Online-Fragebögen gesammelt. Der erste Fragebogen dient der Sammlung von Kommentaren. Mit dem zweiten Fragebogen werden demographische Daten und die Bewertungen der Verbesserungen gesammelt.

5.8 Ablauf der Studie

In diesem Kapitel wird der Ablauf der Studie detailliert beschrieben. Die Studie zielt darauf ab, die Effektivität und Effizienz mit der ChatGPT Testspezifikationen verbessert zu untersuchen. Dafür werden die zwei Me-

thoden *mit ChatGPT* und *ohne ChatGPT* unterschieden. Die Interaktion *mit ChatGPT* erfolgt dem Konzept in Kapitel 5.8.2 nach. Zum Schluss wird noch beschrieben, wie Fehler in den verbesserten Testspezifikationen behoben wurden.

5.8.1 Ablauf

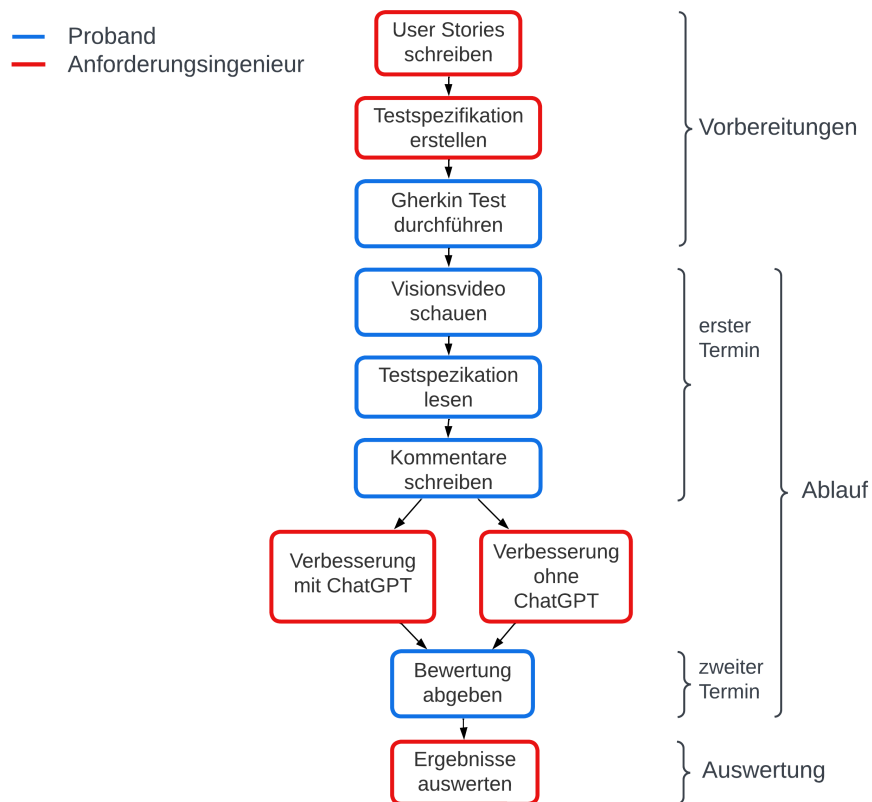


Abbildung 5.2: Ablauf der Studie mit Vorbereitung

Die Studie besteht aus mehreren Phasen. Im Folgenden werden diese Phasen detailliert beschrieben.

Vorbereitung: Vor Beginn der Studie formuliert der Anforderungsingenieur User Stories zu CVParser mit dem Konzept in Kapitel 4.1. Die User Stories entstehen unter Betrachtung des Visionsvideos zu CVParser und den Kommentaren zum Visionsvideo aus der Arbeit von Piam [10]. Auf Basis dieser User Stories wird dann mit dem Konzept in Kapitel 4.2 eine Testspezifikation zu CVParser erstellt. Als nächstes erfolgt die Probandenwahl wie in Kapitel 5.5 beschrieben. Nun wird der Ablauf der Studie mit einem Probanden beschrieben:

Dem Probanden wird ein Test für die Sprache Gherkin geschickt. Diesen

Test muss der Proband vor dem ersten Termin absolvieren. Wenn der Proband diesen Test bestanden hat, wird dieser zu einem ersten Termin in BigBlueButton eingeladen.

Erster Termin: Sobald der Proband in BigBlueButton erscheint, beginnt die Studie. Zuerst wird in BigBlueButton das Visionsvideo zu CV-Parser präsentiert. Anschließend bekommt der Proband einen Link zu einer Umfrage in LimeSurvey (Anhang C.2). Bei dieser Umfrage, soll der Proband Kommentare aus der Sicht eines Stakeholders zu sechs Szenarien schreiben. Zu diesen sollte der Proband mindestens vier Kommentare schreiben. Es werden Hinweise für das Schreiben eines Kommentares vorgegeben (siehe Anhang C.2). Der Proband kann zu jeder Zeit nochmal Teile des Visionsvideos sehen. Nachdem der Proband die Kommentare fertig geschrieben hat, ist der erste Teil der Studie abgeschlossen.

Verbesserung der Testspezifikation: Nach Abschluss des ersten Termins, erstellen zwei Anforderungsingenieure die Verbesserungen. Dabei arbeitet ein Ingenieur *mit ChatGPT* nach dem Konzept in Kapitel 5.8.2 und der andere Ingenieur *ohne ChatGPT*. Die Verbesserungen werden anschließend nochmal korrigiert. Diese Korrektur wird, wie in Kapitel 5.8.3 beschrieben, für beide Methoden vorgenommen. Dabei werden inhaltlich keine Änderungen vorgenommen. Für jedes Szenario werden die Verbesserungen *mit ChatGPT* und ohne ChatGPT in einer Tabelle gegebenübergestellt (Anhang C.4). Die Tabelle wird in Google Tabellen erstellt, sodass der Proband diese später auch betrachten kann. Die Unterschiede zwischen den Verbesserungen und die Veränderungen gegenüber der ursprünglichen Testspezifikation werden fettgedruckt.

Zweiter Termin: Sobald der Proband zum zweiten Termin in BigBlueButton erscheint, beginnt der zweite Teil der Studie. Der Proband bekommt einen Link zu einer Umfrage in LimeSurvey (Anhang C.3). In dieser Umfrage muss der Proband die Verbesserungen bewerten. Damit der Proband die Verbesserungen bewerten kann, wird diesem noch ein Link zu der Tabelle mit den Verbesserungen geschickt. Außerdem werden demographische Daten zu dem Probanden gesammelt.

Auswertung: Mit dem Ergebnis und der Auswertung der Studie sollen dann die Forschungsfragen und die in Kapitel 5.3 definierten Hypothesen beantwortet werden.

5.8.2 Methode: Verbesserung von Testspezifikationen mit ChatGPT

Im Folgenden wird das Konzept für die Methode *mit ChatGPT* der Studie vorgestellt. Um den Einfluss des Anforderungsingenieurs auf die Ergebnisse zu minimieren, wurde dieses Konzept erstellt. Da die Qualität der Ausgaben von ChatGPT schwankt, wird der Anforderungsingenieur die Regenerate-Funktion (siehe Kapitel 2.1.1) von ChatGPT nutzen, um mehrere Varianten

einer Ausgabe zu erzeugen. Außerdem darf der Anforderungsingenieur nur die in diesem Kapitel beschriebenen Prompts nutzen. Zur Erstellung der Prompts und der Interaktion mit ChatGPT wurde sich an Kapitel 4.2 orientiert.

Es dürfen maximal fünf Verbesserungen durch Regenerate pro Kommentar generiert werden, bis sich der Anforderungsingenieur für eine Verbesserung entscheiden muss und diese dann akzeptiert. Diese Beschränkung wird vorgenommen, um eine Grenze für die maximale Anzahl an Interaktionen mit ChatGPT zu setzen. Da nach Erfahrung schon nach spätestens fünf Regenerates nützliche Ergebnisse vorliegen, ist diese Grenze für die Studie gewählt worden. Falls aus der Sicht des Ingenieurs nach fünf Regenerates immer noch keine der fünf Varianten die Kommentare des Probanden erfüllt, darf der Anforderungsingenieur bis zu zwei weitere Durchläufe für diesen Kommentar durchführen können (siehe Anhang C.1).

Das Hauptkriterium für die Akzeptanz einer Verbesserung aus der Sicht des Ingenieurs ist, dass die Verbesserung den Kommentar des Probanden entspricht. Die Erfüllung des Wunsches des Stakeholders hat also eine höhere Priorität, als die Fehler, die ChatGPT gemacht hat. Der Grund dafür ist, dass der Ingenieur weiß, dass Fehler nach der Beschreibung in Kapitel 5.8.3 behoben werden. Es dürfen Schritte aus den Szenarien der Verbesserung gelöscht werden. Allerdings dürfen keine Inhalte hinzugefügt werden. Zur Interaktion mit ChatGPT dürfen nur die folgenden Prompts genutzt werden. Ein Beispiel für die Verwendung dieser Prompts mit ChatGPT ist in Anhang C.3.1.

Verbesserung von Testspezifikationen

Prompt Kontext

Auf der Webseite CVParser kann man sein LinkedIn-Profil in einen Lebenslauf umwandeln. Diese Webseite wird nun beschrieben.

Beschreibung:

""""

[textuelle Beschreibungen]

""""

Für diesen Kontext gibt es nun folgende Testspezifikation in Gherkin:

""""

[Testspezifikation]

""""

Prompt Aufgabe

Du bist ein *[Rolle]* und sollst die Testspezifikation in Gherkin verbessern. *Alternativ, statt "Zu Szenario 'x' wurde folgender Kommentar geschrieben", falls man sich nicht spezifisch auf ein Szenario bezieht: "Folgender Kommentar wurde von einem Stakeholder geschrieben"*

textuelle Beschreibungen: Ist die selbe Beschreibung wie bei der Erstellung der Testspezifikation. Also Beschreibungen der Funktionen, die im

Visionsvideo gezeigt wurden.

Testspezifikation: Hier wird die Testspezifikation verwendet, die mit Hilfe von ChatGPT nach Konzept in Kapitel 4.2 erstellt wurde.

Anwendung des Konzepts für Studie *mit ChatGPT*: Zuerst wird der Kontext Prompt in ChatGPT eingegeben. Dann wird drei mal “Regenerate” genutzt. Das bedeutet, dass drei Chatsektionen entstehen. Für die erste Sektion werden insgesamt fünf Varianten mit Regenerate erstellt. In Bezug auf Abbildung 5.3 gilt somit $n = 3$ und $m = 5$ für die Studie. Der Anforderungsingenieur muss sich dann für die erste Sektion entscheiden, welche der fünf Varianten der Verbesserungen der Spezifikation auswählen soll. Falls aus der Sicht des Anforderungsingenieurs keiner der fünf Varianten den Kommentar erfüllt, dürfen weitere fünf Varianten in einer neuen Chatsektion erstellt werden. Außerdem gibt es einen Ausnahmefall, wenn mehr als zwei Anforderungen in einem Kommentar formuliert sind. In diesem Fall darf der Ingenieur für jede Anforderung ein Szenario aus verschiedenen Varianten der Verbesserungen wählen. Abbildung C.1 zeigt ein Beispiel für zwei Chatsektionen mit drei Varianten von Verbesserungen. Dieser Prozess wird für jeden Kommentar wiederholt.

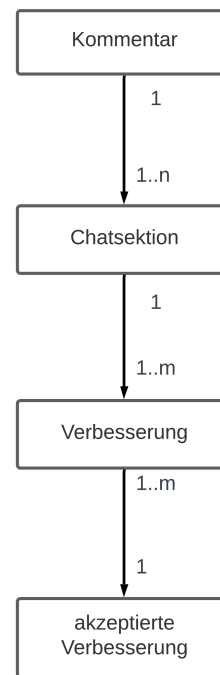


Abbildung 5.3: Struktur für den Prozess der Verbesserung mit ChatGPT

5.8.3 Fehlerkorrektur

Fehler in den Verbesserungen können dazu führen, dass die Verständlichkeit der Verbesserungen verfälscht wird (siehe Kapitel 5.9). Im folgenden Abschnitt wird beschrieben, wie die Fehler in den verbesserten Testspezifikationen behoben wurden. Allgemein werden drei Fehlertypen für die Studie unterschieden: *Syntaxfehler*, *Formulierungsfehler* und *Und-Fehler*. Diese Fehler werden bei beiden Methoden behoben. Zusätzlich werden Rechtschreib- und Grammatikfehler behoben. Es ist anzumerken, dass keine inhaltlichen Veränderungen gemacht werden. Die Ergebnisse von ChatGPT zu interpretieren und inwieweit diese korrigiert werden müssen, liegt im Ermessen des Anforderungsingenieurs. Die Fehlertypen sind in Kapitel 5.1.2 definiert. Als nächstes wird erklärt wie diese behoben wurden. Beispiele für die Korrektur jedes Fehlertypen sind in Anhang C.5 verdeutlicht.

Syntaxfehler: Die Syntax wird korrigiert. Der Satzbau muss eventuell nach der Korrektur angepasst werden. **Formulierungsfehler:** Falsch verwendete Schlüsselwörter werden ausgetauscht. Dann wird der Schritt gegebenenfalls umformuliert. **Und-Fehler:** *Und* wird ergänzt. Dafür wird ein neuer Schritt hinzugefügt.

5.9 Threats to Validity

Im Folgenden werden die Threats to Validity dieser Studie und die Maßnahmen, um diese zu lösen, erläutert. Dabei geht es darum, die Gefahren zu diskutieren, die das Ergebnis der Studie verfälschen könnten. Die Gefahren, die hier beschrieben werden, wurden bereits in der Studie, die in Kapitel 5.8.1 beschrieben wird, berücksichtigt.

Conclusion Validity

Die Schlussfolgerungsvalidität soll sicherstellen, dass die richtigen Schlussfolgerungen für die verwendeten Methoden bei bestimmten Ergebnissen getroffen werden. Dafür ist es vor allem wichtig, dass Ergebnisse auf eine statistische Signifikanz geprüft werden. [28].

Probleme der Internetverbindung mit dem Probanden könnten die Ergebnisse verfälschen. Um das zu verhindern, wird es vorher einen Probendurchlauf mit den Probanden geben.

Es könnte zu wenig Teilnehmer geben, um eine statistische Signifikanz zu zeigen. Deshalb wird vorher mit einem Beispieldatensatz berechnet, ob statistische Signifikanz überhaupt möglich ist.

Um den Threat "Fishing and the error rate" [28] zu verhindern, ist es wichtig, die Metriken und Formeln, mit denen die Ergebnisse ausgewertet werden, schon vorher klar zu definieren. Dadurch wird verhindert, dass Daten ausgewählt werden um das gewünschte Ergebnis zu erzielen.

Die Fragen und Aufgaben an die Probanden sollten klar formuliert sein und die Messskalen müssen klar definiert werden. Dadurch wird sichergestellt, dass die benutzten Skalen auch für die Erfüllung der Fragen in GQM geeignet sind.

External Validity

Die externe Validität hängt vom Design der Studie ab. Die Frage die man sich hier stellt ist, ob eine Generalisierung der Ergebnisse möglich ist. Die größten Risiken liegen bei den gewählten Probanden, der Umgebung und zeitliche Beschränkungen während der Studie [28].

Was soll passieren, wenn den Probanden keine Kommentare einfallen? Die Probanden sollen mindestens fünf Kommentare zu fünf Szenarien schreiben. Damit es den Probanden leichter fällt Kommentare zu den Szenarien zu schreiben, wird es eine Hilfestellung geben. Jeder Proband bekommt einen

Katalog an Fragen, an denen er sich orientieren kann (siehe Anhang C.2). Außerdem wird die Anzahl der Szenarien insgesamt auf sechs erhöht. Hier ist es wichtig die Hilfestellung möglichst allgemein zu formulieren, um den Probanden in seiner Entscheidung nicht zu stark zu beeinflussen.

Der Aufwand für die Probanden könnte zu hoch sein. Um den Aufwand für die Probanden zu messen wurde eine Pilotstudie durchgeführt. Dabei kam heraus, dass 5 Szenarien für die Studie geeignet sind.

Maß der Interaktion des Anforderungsingenieurs *mit ChatGPT* könnte Einfluss auf das Ergebnis haben. Um den Einfluss möglichst gering zu halten, wurde ein Konzept in Kapitel 5.8.2 entwickelt, dass der Anforderungsingenieur befolgen soll.

Technische Probleme könnten den Ablauf des Experiments beeinflussen. ChatGPT könnte sich zum Zeitpunkt des Experiments in einer Wartung befinden oder ausgelastet sein. Langsame oder keine Antworten von ChatGPT wären die Folge. Dadurch, dass die Antworten länger brauchen, könnte das Ergebnis beeinflusst werden und weniger Zeit für den Rest des Experiments sein. Die Vorkehrungen dafür sind, die Recherche der Zeiten von Überlastungen von ChatGPT und der Zeiten von Wartungen oder Releases neuer Feature von ChatGPT. Trotzdem kann es nicht komplett verhindert werden, dass es technische Probleme während den Sitzungen *mit ChatGPT* gibt.

Internal Validity

Bei der internen Validität sollen Faktoren untersucht werden, die eine Gefahr, für die Untersuchung des kausalen Zusammenhang zwischen des verwendeten Konzepts der Studie und dem Ergebnis, darstellen [28].

Um zu verhindern, dass Fehler den Lesefluss und das Verständnis einer Verbesserung einschränken und somit das Ergebnis verfälschen können, werden diese behoben (siehe Kapitel 5.8.2). Zu diesen Fehlern zählen: Rechtschreibfehler, Grammatikfehler, syntaktische Fehler und Formulierungsfehler.

Obwohl ChatGPT nicht direkt das Testobjekt ist, könnte ChatGPT nach jeder Iteration des Experiments dazulernen. Das könnte das Ergebnis beeinflussen. Außerdem könnte die Arbeit *mit ChatGPT* die vor dem Experiment stattgefunden hat, das Ergebnis von ChatGPT beeinflussen. Diese Punkte können nicht komplett verhindert werden. Dennoch wird nach jeder Verbesserung ein neuer Chat *mit ChatGPT* benutzt. Das Experiment wird gesondert auf einem anderen ChatGPT-Account durchgeführt. Dieser Account wurde vorher nicht im Rahmen dieser Arbeit verwendet.

Der Leiter dieser Studie hat sich intensiv mit der Erstellung von Testspezifikationen beschäftigt. Es besteht eine höhere Erfahrung mit der Verbesserung von Testspezifikationen *mit ChatGPT* und *ohne ChatGPT*. Deshalb wird das Ziel in GQM so formuliert, dass die Studie aus der Sicht eines Anforderungsingenieurs mit Erfahrung in ChatGPT und Gherkin

ist. Außerdem wird noch eine zweite Person bei der Verbesserung von Testspezifikation mit beiden Methoden zum Einsatz kommen.

Der Fakt, dass Studenten zum Einsatz kommen, statt erfahrene Stakeholder, könnte das Experiment beeinflussen, da die Fähigkeiten der Studenten stark variieren können. Der Grund hierfür ist eine zeitliche Einschränkung und der Mangel an Experten. Um die Probanden vorzubereiten wird eine Gherkin Schulung durchgeführt und ein Visionsvideo zu CVParser gezeigt. Außerdem können auch Studenten beurteilen, ob die Änderungen an den Spezifikationen ihren Vorstellungen entsprechen.

Die Anforderungsingenieure werden auf die zwei Methoden *mit ChatGPT* und *ohne ChatGPT* gleichmäßig verteilt. Dadurch wird ein Lerneffekt und den Einfluss der individuellen Fähigkeiten der Anforderungsingenieure minimiert. Die Verteilung erfolgt anhand der Tabelle 5.1.

Construct Validity

Bei der Konstruktvalidität untersucht man, inwiefern die Studie das untersuchte Problem repräsentiert [28].

Das Projekt CVParser welches für das Experiment verwendet wird, ist klein und könnte vom Ausmaß nicht direkt mit realen Projekten vergleichbar sein. Da die Studie zeitlich begrenzt ist und damit ein Verständnis für die Stakeholder geschaffen wird, werden die Stakeholder vorher ein Visionsvideo zu CVParser sehen. Dieses Visiosvideo zeigt eine Interaktion mit der Website und nur einen Teil der Website. Für die Studie wird dieses Visionsvideo verwendet, um Missverständnisse der Stakeholder zu vermeiden.

Kapitel 6

Ergebnisse der Studie

Im folgenden Kapitel werden die Ergebnisse der Studie präsentiert und ausgewertet. Jedes Ergebnis wird dann noch interpretiert. Dabei werden die Hypothesen 5.3 überprüft. Schließlich wird es eine Diskussion zu den Ergebnissen geführt, in der unter anderem mögliche Einschränkungen der Studie erörtert werden.

6.1 Ergebnisse und Interpretation

Die Daten die in dieser Studie gesammelt wurden, werden nun präsentiert und im Kontext der Ergebnisse interpretiert. Zuerst werden die Ergebnisse zu Hypothese [H1] präsentiert. Dafür wird nochmal die Nullhypothese wiederholt.

Ergebnis [E1]

[H1.0]: Das Erstellen von Verbesserungen der Testspezifikationen nach einer Iteration *mit ChatGPT* und *ohne ChatGPT* unterscheidet sich nicht in der Effizienz.

Das erste Ergebnis dient zur Messung der Effizienz von ChatGPT. Die Effizienz wurde ermittelt, indem die Zeit zur Verbesserung einer Testspezifikation gemessen und die Akzeptanz dieser Verbesserung überprüft wurde. Die Formel lautet: $a\text{-Zeit} = \frac{g\text{-Zeit}}{\#a\text{-Verbesserungen}}$. Das bedeutet die gesamte Zeit, die zur Verbesserung der Testspezifikation aufgewendet wurde, wird durch die Anzahl der einzelnen Verbesserungen geteilt, die vom Probanden akzeptiert wurden. Dafür wird die Zeit für jeden Probanden und die Akzeptanzquote in einer Tabelle (Anhang D.2) zusammengefasst.

Auf den ersten Blick lassen sich in den Zeiten kaum Unterschiede erkennen. Auffällig ist jedoch, dass bei Proband 10 und 12 eine große Differenz zwischen *mit ChatGPT* und *ohne ChatGPT* in der Zeit vorliegt. In der Akzeptanzquote sieht man schon eine Tendenz, dass eher Verbesserungen

mit ChatGPT akzeptiert wurden, als ohne. Um die Effizienz zu prüfen wird die Formel von *a-Zeit* für die Werte berechnet. Eine Darstellung als Boxplot kann mehr über die Verteilung von *a-Zeit mit ChatGPT* und *ohne ChatGPT* zeigen.

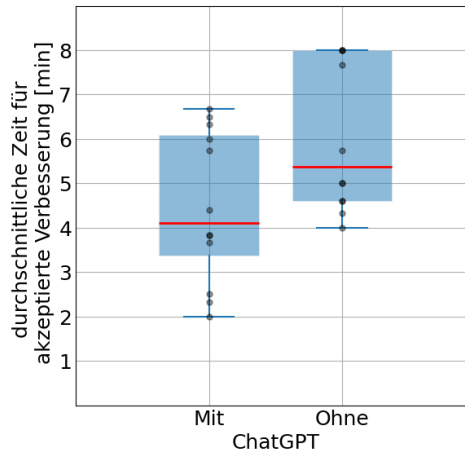


Abbildung 6.1: Zeit für eine akzeptierte Verbesserung (*a-Zeit*)

Nun wird Abbildung 6.1 betrachtet. Auf der *y*-Achse ist *a-Zeit* abgebildet und auf der *x*-Achse die Methoden *mit ChatGPT* und *ohne ChatGPT*. Die berechneten Werte für *a-Zeit* sind im Anhang D.3. Die Verteilungen der beiden Boxplots sind ähnlich, jedoch nicht gleich. Es sind keine Ausreißer zu erkennen. Der Median von *a-Zeit mit ChatGPT* ist geringer als *ohne ChatGPT*.

Interpretation

Zur Untersuchung eines signifikanten Unterschiedes zwischen den beiden Methoden hinsichtlich *a-Zeit* gibt, wird der Mann-Whitney-U-Test (U-Test) nach Bortz [30] durchgeführt. Um die Hypothese [H1.1] zu akzeptieren, muss die Nullhypothese [H1.0] widerlegt werden. Dafür führt man den U-Test mit einem zweiseitigen Test und einem Signifikanzniveau von $\alpha = 5\%$ durch. Wenn dieser Test ergibt, dass $p < 0.05$ ist, kann die Nullhypothese verworfen werden. Da Bindungen zwischen den Rängen vorliegen, wird noch die Korrektur für verbundene Ränge vorgenommen [30]. Der *z* Wert wurde nach der Formel von Bortz [30] berechnet und anschließend mit dem Tool der "Social Science Statistics" [31] in einen *p*-Wert umgewandelt. Zur Referenz des *p*-Wertes, wurde außerdem die Tabelle für *z*-Werte im Buch von Bortz [30] nachgeschlagen. Dadurch bekommt man die folgenden Werte: $z = -2,053$ und $p = 0,04$. Da der Wert *p* unter dem Signifikanzniveau von 5% liegt, kann die Nullhypothese [H1.0] somit verworfen werden und die Hypothese [H1.1] akzeptiert werden. Damit unterscheidet sich die Zeit der Verbesserung einer akzeptierten Testspezifikation zwischen *mit ChatGPT*

und *ohne ChatGPT*. Wie bereits erläutert, liegt der Median *mit ChatGPT* unter dem von *ohne ChatGPT*. Das deutet darauf hin, dass eine akzeptierte Verbesserung *mit ChatGPT* schneller erreicht werden konnte als *ohne ChatGPT*.

Als nächstes wird die Effektivität untersucht. Diese ist in drei Abschnitte unterteilt. Zuerst kommt ein Abschnitt zu Erfüllungsgrad, Verständlichkeit und Akzeptanz. Dann wird die Korrelation der Werte untersucht. Letztlich gibt es eine Fehleranalyse. Für die Untersuchung des Erfüllungsgrades wird die Nullhypothesen zu [H2.1a] betrachtet.

Ergebnis [E2.1a]

[H2.1a.0]: Die verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT* unterscheiden sich nicht im *Erfüllungsgrad*.

Für diese Hypothese wird der Erfüllungsgrad gemessen, mit dem übergeordneten Ziel die Effektivität zu untersuchen. Dabei sollen die Daten den Methoden entsprechend in zwei Gruppen für den U-Test unterteilt werden. Zuerst werden die Ergebnisse präsentiert.

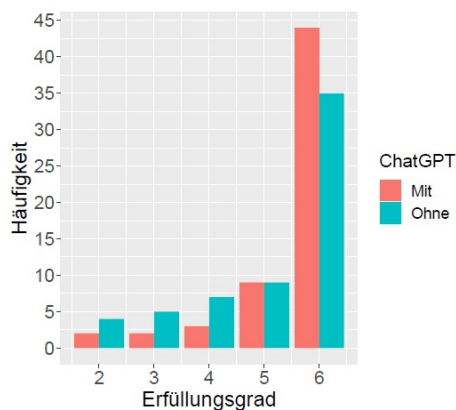


Abbildung 6.2: Häufigkeit der Erfüllung

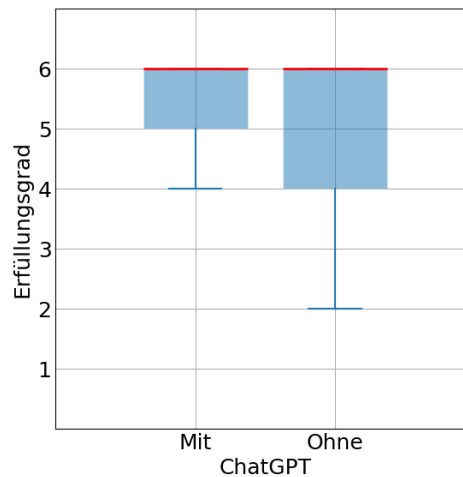


Abbildung 6.3: Boxplot: Erfüllung mit und *ohne ChatGPT*

In Abbildung 6.2 ist auf der x-Achse der Erfüllungsgrad und auf der y-Achse die Häufigkeit, der Erfüllungsgrade, die von den Probanden vergeben wurde. Insgesamt sind es 60 Datenpunkte pro Methode *mit ChatGPT* und *ohne ChatGPT*. Dabei ist auffällig, dass der höchste Wert 6 öfter für *mit ChatGPT* vergeben wurde. Das könnte schon darauf hindeuten, dass es einen höheren Erfüllungsgrad *mit ChatGPT* gab. Die Mediane sind wie in Abbildung 6.3 zu sehen gleich, sodass es bei den Boxplots keine sichtlichen

Unterschiede gibt.

Interpretation

Der signifikante Unterschied für den Erfüllungsgrad *mit ChatGPT* und *ohne ChatGPT*, wird auch hier mit dem U-Test mit Bindungen gezeigt. Da die Berechnungsschritte gleich sind, wird für genauere Details zur Auswertung dieses Tests auf die Interpretation von [E1] hingewiesen. Um die Nullhypothese [H2.1a] zu verwerfen, muss der Test eine Signifikanz von $p < 0,05$ aufweisen. Das Ergebnis des Tests ist $z = -1,924$ und $p = 0,0544$. Das Ergebnis ist mit $p > 0,05$ somit nicht signifikant. Die Nullhypothese kann somit nicht verworfen werden. Das bedeutet, der Erfüllungsgrad *mit ChatGPT* und *ohne ChatGPT* unterscheidet sich nicht signifikant.

Ergebnis [E2.1b]

[H2.1b.0]: Die verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT* unterscheiden sich nicht in der *Verständlichkeit*.

Für diese Hypothese wird die Verständlichkeit gemessen. Auch hier ist das Ziel die Effektivität von ChatGPT zu untersuchen. Analog zum Erfüllungsgrad wird auch bei der Verständlichkeit der U-Test durchgeführt.

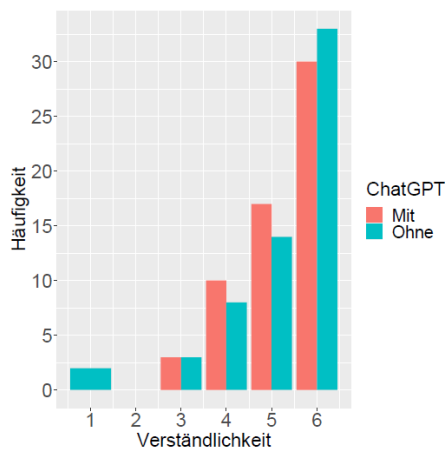


Abbildung 6.4: Häufigkeit der Verständlichkeit

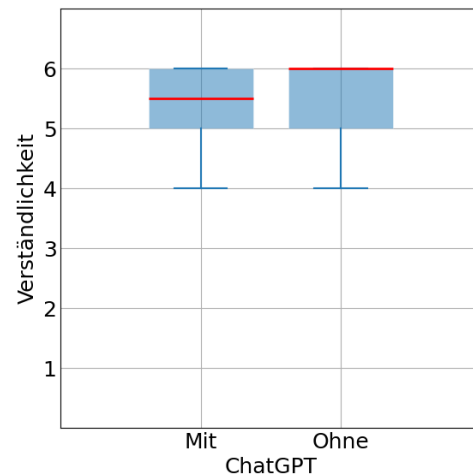


Abbildung 6.5: Boxplot: Verständlichkeit mit und *ohne ChatGPT*

In Abbildung 6.4 ist auf der x-Achse die Verständlichkeit und der y-Achse die Häufigkeit des Wertes der Verständlichkeit bei den Probanden. Auch hier gibt es 60 Datenpunkte für die Verständlichkeit. Es ist zu beobachten, dass der höchste Wert 6 *ohne ChatGPT* etwas häufiger vorkommt. Außerdem wird die 4 und 5 *mit ChatGPT* häufiger von Probanden vergeben. Insgesamt ist jedoch zusammen mit dem Boxplot in Abbildung 6.5 kein eindeutiger

Trend erkennbar. Der Median *ohne ChatGPT* ist um 0,5 größer.

Interpretation

Nun werden analog zum Erfüllungsgrad die Werte z und p für die Verständlichkeit berechnet. Dadurch bekommt man für $z = -0,316$ und für $p = 0,753$. Für einen zweiseitigen Test bei einer Signifikanz von 5%, kann somit die Hypothese nicht verworfen werden. Also: Die Verständlichkeit *mit ChatGPT* und *ohne ChatGPT* unterscheidet sich nicht signifikant.

Ergebnis [E2.1c]:

[H2.1c.0]: Die verbesserten Testspezifikationen *mit ChatGPT* und *ohne ChatGPT* unterscheiden sich nicht in der *Akzeptanz*.

Um die Hypothese bezüglich der Effektivität zu untersuchen, wird die Akzeptanz gemessen und ausgewertet. Die Daten werden wieder analog zum Erfüllungsgrad mit dem U-Test ausgewertet. Wichtig ist zu beachten, dass es hier weniger Datenpunkte gibt, da die Akzeptanzquote pro Proband betrachtet wird. In der Tabelle 6.1 ist die gesamte Anzahl

Skala	Akzeptanz	
	Mit ChatGPT	Ohne ChatGPT
akzeptiert	52	44
nicht akzeptiert	8	16

Tabelle 6.1: Akzeptanz der Probanden insgesamt

der akzeptierten Verbesserungen zu sehen. Dabei ist zu beobachten, dass mehr Verbesserungen *mit ChatGPT* akzeptiert wurden, als *ohne ChatGPT*. Außerdem wurden *ohne ChatGPT* doppelt so viele Verbesserungen nicht akzeptiert, als *mit ChatGPT*. Um die Auswertung mit dem U-Test vornehmen zu können, wird die Akzeptanzquote pro Proband berechnet. Diese Akzeptanzquoten sind der Tabelle D.2 zu entnehmen. Diese Werte werden nun in einem Boxplot dargestellt.

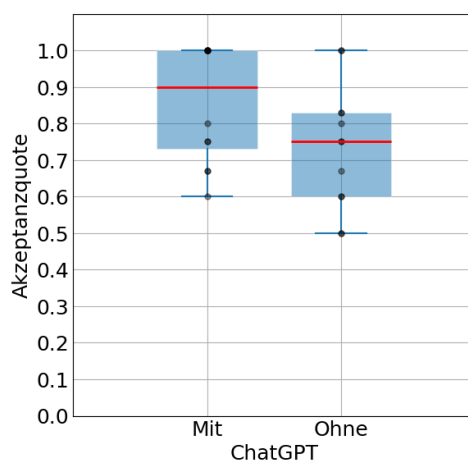


Abbildung 6.6: Boxplot: Verständlichkeit *mit und ohne ChatGPT*

In Abbildung 6.6 werden 12 Datenpunkte für beide Methoden *mit ChatGPT* und *ohne ChatGPT* aus Tabelle D.2 als Boxplot dargestellt. Der Median *mit ChatGPT* ist um 0,15 größer, als *ohne ChatGPT*. Die Verteilungen sind relativ ähnlich, jedoch ist der Boxplot *mit ChatGPT* höher gelegen. Dadurch könnte man argumentieren, dass die Akzeptanzquote *mit ChatGPT* durchschnittlich höher ist.

Interpretation

Um die Signifikanz zu überprüfen wird wieder der U-Test durchgeführt (siehe [E1] Interpretation). Dieser Test ergibt für $z = 1,507$ und $p = 0,13181$ für einen zweiseitigen Test. Da p über dem 5% Signifikanzniveau liegt, kann die Nullhypothese nicht verworfen werden. Das bedeutet, es gibt keinen signifikanten Unterschied der Akzeptanz *mit ChatGPT* und *ohne ChatGPT*.

Ergebnis [E2.2]:

In diesem Abschnitt werden die Fehler die *mit ChatGPT* und *ohne ChatGPT* gemacht wurden analysiert. Dafür werden die Fehler gezählt und der Aufwand geschätzt um diese Fehler zu korrigieren. Für dieses Ergebnis gibt es keine Hypothese die widerlegt werden soll. Die Fehlertypen werden wie in Kapitel 5.1.2 beschrieben gezählt. Diese werden in 5.8.3 nochmal genauer erklärt. Hier ist nochmal eine Aufzählung der Fehlertypen: *Formulierungsfehler*, *Und-Fehler* und *Syntaxfehler*. Der gesamte Code, der untersucht wurde, beläuft sich bei beiden Methoden auf etwa 500 Zeilen. Nun wird das Ergebnis mit Hilfe einer Tabelle zusammengefasst.

Wie in Tabelle 6.2 zu erkennen, wurden *mit ChatGPT* wesentlich mehr Fehler gemacht. Bei allen Fehlertypen ist *mit ChatGPT* weit vorne. *Und-Fehler* wurden *ohne ChatGPT* gar nicht gemacht.

Interpretation Bei der Methode *mit ChatGPT* wurden wesentlich mehr

	Syntaxfehler	Formulierungsfehler	Und-Fehler
mit ChatGPT	5	4	0
ohne ChatGPT	19	21	14

Tabelle 6.2: Anzahl der Fehler *mit ChatGPT* und *ohne ChatGPT*

Fehler gemacht. Der Grund dafür könnte sein, dass die Ergebnisse von ChatGPT komplett unverändert bei der Fehleranalyse übernommen wurden. Es ist wichtig zu beachten, dass die Fehler *mit ChatGPT* auf die Anzahl der Zeilen gesehen, für jeden Fehlertypen ca. 3-4% betragen. Insgesamt wurden *mit ChatGPT* 54 Fehler gemacht, was ca. einer 10% Fehlerquote entsprechen würde. *Ohne ChatGPT* wurden 9 Fehler gemacht, was ca. einer 2% Fehlerquote entsprechen würde. Die Fehler, die mit beiden Methoden gemacht wurden, waren in den meisten Fällen relativ einfach zu beheben. Im folgenden wird der Aufwand der Fehlerkorrektur geschätzt.

Der *Und-Fehler* war leicht zu beheben, meistens musste das Wort "und" durch das Schlüsselwort *Und* ersetzt werden.

Ein *Formulierungsfehler* zu beheben, hat einen leichten bis mittleren Aufwand gehabt. Meistens musste das Schlüsselwort getauscht werden. *Mit ChatGPT* wurde oft der Fehler gemacht, dass statt *Wenn*, in einem Schritt *Dann* verwendet wurde. Dadurch verletzte der Schritt die Formulierung eines Schrittes, da im *Dann* Schritt eine Aktion des Nutzers beschrieben wird.

Syntaxfehler waren leicht zu beheben. Da nur das Format eingehalten werden muss, sind diese Fehler relativ einfach zu sehen. Meistens musste hier ein Schlüsselwort ausgetauscht werden, da zum Beispiel statt einem {*Wenn,Und*} die Reihenfolge {*Wenn,Wenn*} verwendet wird. Außerdem wurde der Fehler gemacht, dass die Schlüsselwörter falsch oder gar nicht verwendet werden.

Ergebnis [E2.3a-b]:

Als nächstes wird eine Korrelation der Metriken überprüft. Dafür wird untersucht, ob es eine Korrelation von Erfüllungsgrad mit Akzeptanz und Verständlichkeit mit Akzeptanz gibt. Folgend werden die Ergebnisse mit den zwei Nullhypothesen präsentiert.

[H2.3a.0] Es gibt keine Korrelation von *Erfüllungsgrad* mit *Akzeptanz*

[H2.3b.0] Es gibt keine Korrelation von *Verständlichkeit* mit *Akzeptanz*

Die Ergebnisse *mit ChatGPT* und *ohne ChatGPT* werden für diesen Abschnitt zusammengefasst. Um eine Korrelation darzustellen, werden die Ergebnisse in die zwei Gruppen der Akzeptanz *Ja* und *Nein* unterteilt. Die Ergebnisse werden jeweils als Boxplot für Erfüllungsgrad und Verständlichkeit dargestellt.

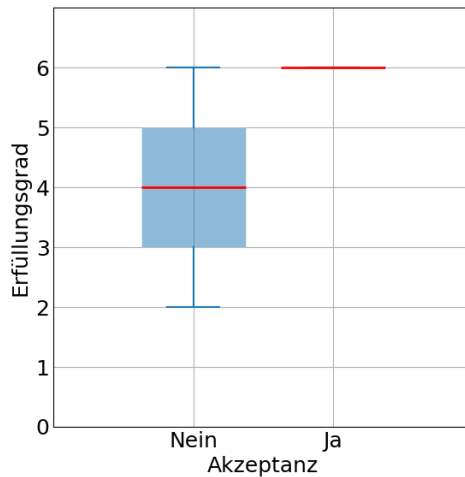


Abbildung 6.7: Boxplot: Erfüllung mit Akzeptanz

Die Boxplots in Abbildung 6.7 deutet auf einen Zusammenhang zwischen hohem Erfüllungsgrad und positiver Akzeptanz. Dabei ist zu erkennen, dass der Median für *Ja* um zwei Werte höher ist, als für *Nein*. Der Boxplot für akzeptierte Verbesserungen besteht nur aus einem Median, das bedeutet dass mindestens 75% der Werte bei 6 liegen.

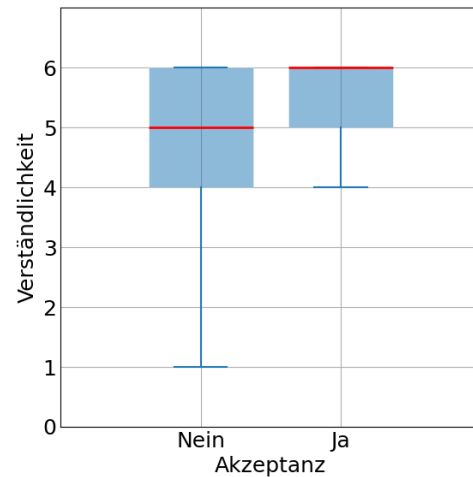


Abbildung 6.8: Boxplot: Verständlichkeit mit Akzeptanz

Abbildung 6.8 zeigt den Zusammenhang von Akzeptanz und Verständlichkeit. Hier ist nicht eindeutig zu erkennen, welcher Zusammenhang bestehen könnte. Der Median von *Ja* ist nur um 0,5 höher. Hier wird vorerst nur ein leichter Zusammenhang von Verständlichkeit mit Akzeptanz erkannt.

Interpretation

Um eine Korrelation zu untersuchen muss der Korrelationskoeffizient berechnet werden. In diesem Fall wird der Korrelationskoeffizient für das biserale ρ , wie von Bortz et al. [32] beschrieben, berechnet. Dafür wird dem Erfüllungsgrad ein Rang zugeordnet. Die Ränge die daraus resultieren sind die Rangreihe X. Das alternative Merkmal Y ist die Unterteilung der Akzeptanz in die Gruppen *akzeptiert* und *nicht akzeptiert*. Da Bindungen in den Rängen von X auftreten, muss die Formel für ρ angepasst werden [32]. Um eine Signifikanz der Korrelation zu zeigen, wird der U-Test verwendet. Beim U-Test muss auch hier die Korrektur wegen verbundener Ränge vorgenommen werden [30]. Weil es große Differenzen der Stichprobenumfänge der beiden Gruppen gibt ($N_1:N_2 > 1:3$, hier $N_1 = 24$ und $N_2 = 96$), wird zusätzlich eine Korrektur von z nach Verdooren (1963) vorgenommen (von Bortz [32] beschrieben).

Für die Korrelation von Erfüllungsgrad mit Akzeptanz ist für $\rho = 0,665$ herausgekommen. Da $\rho > 0$, bedeutet es, dass es einen positiven Zusammenhang zwischen Erfüllungsgrad und Akzeptanz gibt. Aufgrund der positiven

Korrelation, kann ein einseitiger U-Test durchgeführt werden. Der Test resultiert mit $z_{corr} = -5,432$ und $p < 0,00001$. Die Signifikanz dieser Korrelation, ist mit $p < 0,00001$, der Wert von p liegt unter $\alpha = 5\%$. Die Schlussfolgerung dieser Berechnung ist, dass man die Nullhypothese verwerfen kann. Somit gibt es eine positive Korrelation zwischen Erfüllungsgrad und Akzeptanz.

Das Ergebnis der Korrelation von Verständlichkeit mit Akzeptanz ist $\rho = 0,302$. Somit liegt eine positive Korrelation vor. Beim einseitigen U-Test bekommt man für $z_{corr} = -2,09651002$ und $p = 0,018$. Da $\rho > 0$ und $p < 5\%$, kann für ein Signifikanzniveau von 5% die Aussage getroffen werden, dass es eine positive Korrelation von Verständlichkeit mit Akzeptanz gibt. Die Nullhypothese kann somit verworfen werden.

6.2 Diskussion

Im folgenden werden die Ergebnisse der Studie im Kontext der Forschungsfragen dieser Arbeit diskutiert. Außerdem wird auf die Grenzen der Arbeit eingegangen und wie diese die Interpretation der Ergebnisse beeinflussen könnten. Die folgenden Aussagen werden im Kontext der Metriken, die für diese Studie entwickelt wurden, getroffen.

6.2.1 Forschungsfragen

[RQ1]: Wie effizient ist ein "Large Language Model" bei der Verbesserung von Testspezifikationen?

Ein "Large Language Model" konnte in unserer Studie mit den Ergebnissen eines Anforderungsingenieurs ohne Unterstützung von ChatGPT mithalten. Es konnte eine signifikant erhöhte Effizienz bei der Verwendung von ChatGPT beobachtet werden. ChatGPT kann die Effizienz bei der Verbesserung von Testspezifikationen, in Anbetracht der Limitierungen, erhöhen.

[RQ2]: Wie effektiv ist ein "Large Language Model" bei der Verbesserung von Testspezifikationen?

In Bezug auf die Effektivität, konnte bei den Ergebnissen mit ChatGPT kein signifikanter Unterschied erkannt werden. Es gab keinen signifikanten Einfluss auf Erfüllungsgrad, Verständlichkeit und Akzeptanz bei Verwendung von ChatGPT. Es werden Tendenzen erkannt, dass ChatGPT bessere Ergebnisse als ein Anforderungsingenieur ohne die Unterstützung von ChatGPT liefert. Dennoch konnte kein signifikanter Unterschied der Ergebnisse gezeigt werden. Somit kann keine konkrete Aussage für die Effektivität von "Large Language Models" getroffen werden. Dennoch konnte beobachtet werden, dass das LLM wesentlich mehr syntaktische Fehler und Formulierungsfehler gemacht hat. Ein LLM kann Testspezifikationen in wenigen Sekunden generieren, jedoch muss wieder Aufwand in die Korrektur der Spezifikationen investiert werden.

[RQ3]: Welche Faktoren beeinflussen die Akzeptanz einer Testspezifikation durch Stakeholder?

Das Ergebnis der Studie hat gezeigt, dass es eine Korrelation von Erfüllungsgrad mit Akzeptanz und Verständlichkeit mit Akzeptanz gibt. Die Szenarios möglichst verständlich zu formulieren, ist somit sinnvoll, wenn man eine hohe Akzeptanz der Stakeholder möchte. Der Erfüllungsgrad steht den Ergebnissen nach, in einem engeren Zusammenhang mit der Akzeptanz als die Verständlichkeit. Das bedeutet, dass die Inhalte des Kommentars zu berücksichtigen ein wichtiger Bestandteil der Akzeptanz der Stakeholder ist.

6.2.2 Grenzen der Arbeit

Im folgenden werden die Beschränkungen der Studie aufgezählt. Bei der Arbeit mit ChatGPT gab es einige Schwierigkeiten die berücksichtigt werden müssen. Es ist wichtig zu erwähnen, dass sich die Oberfläche der Website zum sechsten Probanden geändert hat. Bei der Verbesserung der Testspezifikationen der ersten beiden Probanden gab es technische Probleme bei ChatGPT. Dadurch musste die Arbeit an den Testspezifikationen später fortgesetzt werden.

Es gibt Hinweise darauf, dass ChatGPT im Laufe der Studie besser darin geworden ist Testspezifikationen zu erstellen.

Es gab auch einen Lerneffekt der Ingenieure durch ChatGPT. Der Lerneffekt war sowohl positiv als auch negativ. Die Ingenieure hatten den Eindruck, dass es ihnen leichter fiel die Testspezifikationen zu schreiben, nachdem sie für die Studie mit ChatGPT gearbeitet haben. Andererseits machte ein Ingenieur Fehler, die ChatGPT gemacht hatte. Allgemein haben die Ingenieure sich durch die Ausgaben von ChatGPT inspirieren lassen. Die hohe Diskrepanz einiger Werte deutet darauf hin, dass die Ingenieure einen Lerneffekt im Verlauf der Studie hatten.

Kapitel 7

Schlussfolgerung und Ausblick

Zum Ende dieser Arbeit werden die Ergebnisse nochmal zusammengefasst. Im Kontext der These, das Anforderungsingenieure und Stakeholder durch den Einsatz von Large Language Models unterstützt werden können, wird eine Schlussfolgerung getroffen. Abschließend gibt es einen kurzen Ausblick für zukünftige Forschung.

Bei der Arbeit mit "Large Language Models" muss berücksichtigt werden, dass die Ausgaben Fehler enthalten können. Die Ergebnisse der Studie mit ChatGPT bestätigen diese Aussage. Die Testspezifikationen wurden auf Syntaxfehler und Formulierungsfehler geprüft. Dabei wurden beispielsweise mit ChatGPT neunzehn und ohne ChatGPT fünf Syntaxfehler gefunden.

Diese Studie zeigt nicht, welche Ergebnisse ein Ingenieur mit Unterstützung von ChatGPT erzielen kann. Sondern Ergebnisse von ChatGPT, die von einem Ingenieur ausgewählt wurden. Obwohl der Einfluss des Anforderungsingenieurs minimiert wurde, gibt es die Tendenz, dass ChatGPT mit der manuellen Verbesserung mithalten kann. Außerdem waren die Verbesserungen mit ChatGPT, unter Berücksichtigung der Einschränkungen, effizienter.

Die Ergebnisse von Large Language Models sollten mit Vorsicht genutzt und genau auf Fehler geprüft werden. Aus diesem Grund sollte ein Anforderungsingenieur, der eine Spezifikation mit einem Large Language Model erstellt, trotzdem manuell bearbeiten. Man benötigt eine gewisse Expertise im Anwendungsgebiet, um Fehler zu erkennen. Insgesamt wird die Vermutung getroffen, dass ein Large Language Model als Werkzeug nützlich sein kann, aber nicht den Prozess an sich ersetzen kann. Diese These könnte man in zukünftiger Forschung untersuchen.

In Zukunft könnte man untersuchen wie Large Language Models die Anforderungsingenieure bei der Verbesserung von Testspezifikationen unterstützen können. Außerdem könnte man die Leistung eines LLMs bei der Erstellung von Testspezifikationen vergleichen. Da GPT-3.5 eingesetzt wurde, wäre der nächste Schritt die Leistung von GPT-4 in diesem Anwendungsgebiet zu untersuchen.

Anhang A

Grundlagen

A.1 Kriterien für gute Gherkin Szenarien

1. "Declarative (high level) steps are preferred to imperative (low level) steps"
2. "Business terminology should be consistently used across the specification"
3. "Scenarios should focus on the benefit they offer to users, if implemented"
4. "Scenarios should use the terminology understood by all project stakeholders"
5. "Scenarios should clearly separate Given, When and Then steps"
6. "Use past tense for contexts (Given), present tense for events (When), and "should" for outcomes (Then)"
7. "A good quality scenario should be ..., testable, ..., unambiguous, complete..." [22]

Anmerkungen (nummeriert nach Kriterien):

- 1) Dieses Kriterium ist schon detailliert in Kapitel 2.2.1 beschrieben.
- 2) Allgemein sollte der Wortgebrauch zwischen den Szenarien konsistent sein. Das benutzte Vokabular sollte eindeutig sein. Außerdem: Je kleiner das Vokabular, umso besser.
- 3) Dieses Kriterium ist durch die Verwendung von User Stories in den Features gewährleistet, da diese den Nutzen für den Nutzer verdeutlichen.
- 4) Die Szenarien sollten so formuliert werden, dass sie für alle Stakeholder verständlich sind.
- 5) Siehe Kapitel 2.2.1
- 6) Die verschiedenen Zeitformen müssen eingehalten werden.
- 7) Ein Szenario ist nur testbar, wenn es mit den Mitteln, die zur Verfügung stehen, umsetzbar ist. Es muss eindeutig sein, es darf keine Zweideutigkeiten geben. Es muss komplett sein und somit einen abgeschlossenen Prozess beschreiben.

A.2 Beispiel für die Umformulierung von mehreren Szenarien in ein Szenario

zwei Szenarien:

Scenario 1: Artikel zum Warenkorb hinzufügen, wenn nicht auf Lager
Angenommen der Nutzer hat einen Artikel ausgewählt, der nicht auf Lager ist
Wenn der Nutzer auf 'Zum Warenkorb hinzufügen' klickt
Dann sollte ein Nachricht angezeigt werden, dass der Artikel nicht auf Lager ist

Scenario 2: Nachricht wegklicken
Angenommen der Nutzer hat eine Nachricht bekommen
Wenn der Nutzer auf 'OK' klickt
Dann sollte die Nachricht verschwinden

Daraus wird ein Szenario:

Scenario 1: Artikel zum Warenkorb hinzufügen, wenn nicht auf Lager
Angenommen der Nutzer hat einen Artikel ausgewählt, der nicht auf Lager ist
Wenn der Nutzer auf 'Zum Warenkorb hinzufügen' klickt
Dann sollte angezeigt werden, dass der Artikel nicht auf Lager ist
Wenn der Nutzer auf 'OK' klickt
Dann sollte die Nachricht verschwinden

Anhang B

Konzepte

B.1 Erstellung der Testspezifikation mit ChatGPT

B.1.1 Beispiel: Erstellung der Testspezifikation mit ChatGPT (Dialog)

User

Prompt Kontext

Auf der Webseite CVParser kann man sein LinkedIn-Profil in einen Lebenslauf umwandeln. Hier ist eine Beschreibung dieser Website:

””””

Zum Anfang ist man nicht angemeldet und befindet sich auf der Seite zum Anmelden. Hier kann man auf einen “create” link klicken, wodurch man zu einem Registrierungsformular weitergeleitet wird. Beim Registrierungsformular gibt man seine E-Mail und Passwort ein und wiederholt das Passwort. Anschließend klickt man auf “create” und erhält eine Email mit einem Bestätigungscode. Wenn man den Bestätigungscode bei “Input code here” einfügt und auf den “create” klickt, wird ein Account bei CVParser erstellt. Man befindet sich nun wieder auf der Anmelde Seite, hier kann man E-Mail und Passwort eingeben und auf “Login” drücken und wird dann eingeloggt. Man kann jetzt durch die verschiedenen Bereiche der Seite navigieren. Bei “Tool” kann man einen Lebenslauf extrahieren. Man kann hier sein LinkedIn Profil als Text einfügen. Nun klickt man auf “extract” , dann werden die Informationen des LinkedIn Profils als Lebenslauf dargestellt und in der Datenbank von CVParser gespeichert. Bei “Edit” kann man seinen Lebenslauf bearbeiten. Man kann seinen Lebenslauf anpassen, auf “Confirm” klicken und die Änderungen werden gespeichert. Bei “About” kann man Tutorials sehen, wenn man auf ein Tutorial klickt, dann kann

man es sich ansehen. Man kann durch den Smiley auf sein Profil. Hier kann man mit "Reset Password" sein Passwort zurücksetzen. Außerdem kann der Nutzer, während er eingeloggt ist, auf "Logout" klicken und wird ausgeloggt.

""""

ChatGPT

Vielen Dank für die detaillierte Beschreibung der Interaktionen auf der CVParser-Webseite. Hier ist eine zusammenfassende Beschreibung der Schritte:...

User

Prompt Aufgabe

Gegeben seien die Beschreibungen von CVParser. Nun wird eine Testspezifikation in Gherkin zu allen Features der Website benötigt. Du bist ein Requirements Engineer und sollst mit Hilfe von User Stories und den Beschreibungen Feature-Dateien in Gherkin schreiben.

User Stories:

""""

US1: Als unregistrierter Nutzer möchte ich mich registrieren, damit ich einen Account bei CVParser habe.

US2: Als unregistrierter Nutzer möchte ich den Bestätigungscode eingeben, damit meine E-Mail bestätigt ist.

US3: Als Nutzer möchte ich mich einloggen, damit ich die Funktionen von CVParser nutzen kann.

US4: Als Nutzer möchte ich die Bereiche der Website erkunden, damit ich die dort angebotenen Funktionen nutzen kann.

US5: Als Nutzer möchte ich meinen Lebenslauf aus meinem LinkedIn Profil erstellen, damit ich meinen Lebenslauf nicht manuell schreiben muss.

US6: Als Nutzer möchte ich meinen Lebenslauf bearbeiten, damit ich Details verändern kann.

US7: Als Nutzer möchte ich Tutorials anschauen, damit ich weiß wie man CVParser benutzt.

US8: Als Nutzer möchte ich mein Passwort ändern, damit ich für die Sicherheit meines Accounts sorgen kann.

US9: Als Nutzer möchte ich mich ausloggen, damit keiner unbefugt Zugriff auf meinen Account hat.

""""

Format:

Die Feature-Dateien müssen in Gherkin-Code ausgegeben werden. Eine Feature-Datei hat genau ein Feature und besteht aus mehreren Szenarien. Ein Szenario soll dabei den Ablauf für das Feature beschreiben. Es müssen folgende Gherkin keywords verwendet werden: Feature, Background, Scenario, Scenario Outline, Wenn, Dann, Und, Aber; Folgendes Format muss für einen Gherkin Test eingehalten werden:

''''''

Feature <y>: <Feature>

<User Story>

Scenario <z>: <Titel>

Angenommen <Vor Interaktion gegeben>

Wenn <Aktionen des Nutzer>

Dann <Reaktionen des Systems>

''''''

Beispiel 1:

''''''

Feature 1: Artikel zum Warenkorb hinzufügen

Als Nutzer möchte ich einen Artikel zu meinem Warenkorb hinzufügen, damit ich den Artikel später kaufen kann.

Scenario 1: Artikel erfolgreich zum Warenkorb hinzufügen

Angenommen der Nutzer hat einen Artikel ausgewählt

Wenn der Nutzer auf "Zum Warenkorb hinzufügen" klickt

Dann sollte der Artikel zum Warenkorb hinzugefügt werden

Scenario 2: Artikel zum Warenkorb hinzufügen, wenn nicht auf Lager

Angenommen der Nutzer hat einen Artikel ausgewählt, der nicht auf Lager ist

Wenn der Nutzer auf "Zum Warenkorb hinzufügen" klickt

Dann sollte der Nutzer eine Nachricht sehen, dass der Artikel nicht auf Lager ist

Wenn der Nutzer auf "OK" klickt

Dann sollte die Nachricht verschwinden

''''''

Das "Beispiel 1" ist kein inhaltlicher Kontext für die CVParser Webseite.

ChatGPT

Hier sind die Feature-Dateien in Gherkin-Code für die gegebenen User Stories zur CVParser-Website:

Abschnitt 1: Anmeldung

Scenario 1: Registrierung eines neuen Nutzers
Angenommen der Nutzer ist auf der Startseite
Wenn der Nutzer auf den "create" Link klickt
Dann öffnet sich das Registrierungsformular
Und der Nutzer gibt seine E-Mail und ein Passwort ein
Und das Passwort wird wiederholt
Und der Nutzer klickt auf den "create" Knopf
Dann sollte eine Bestätigungsmail an die E-Mail des Nutzers gesendet werden

Scenario 2: Eingabe des Bestätigungscode
Angenommen der Nutzer hat eine Bestätigungsmail erhalten
Wenn der Nutzer den Bestätigungscode aus der E-Mail in das Textfeld "Input code here" eingibt
Und der Nutzer klickt auf den "create" Knopf
Dann wird ein Account bei CVParse erstellt

Scenario 3: Einloggen
Angenommen der Nutzer ist auf der Startseite
Wenn der Nutzer seine E-Mail und sein Passwort eingibt
Und der Nutzer klickt auf den "Login" Knopf
Dann wird der Nutzer eingeloggt

Scenario 4: Navigation durch die Website
Angenommen der Nutzer ist eingeloggt
Wenn der Nutzer auf den Smiley klickt
Dann wird er zu seinem Profil weitergeleitet
Wenn der Nutzer auf "About" klickt
Dann wird er zu den Tutorials weitergeleitet
Wenn der Nutzer auf "Edit" klickt
Dann wird er zur Bearbeitung seines Lebenslaufs weitergeleitet
Wenn der Nutzer auf "Tool" klickt
Dann wird er zur Extraktion weitergeleitet

Abschnitt 2: Extraktion

Scenario 5: Erstellen eines Lebenslaufs aus LinkedIn
Angenommen der Nutzer ist auf der "Tools"-Seite
Wenn der Nutzer sein LinkedIn-Profil kopiert

Und der Nutzer fügt das Profil ein
Und der Nutzer klickt auf "extract"
Dann werden Informationen des LinkedIn Profils als Lebenslauf dargestellt
Und in der Datenbank von CVParser gespeichert

Scenario 6: Bearbeiten des Lebenslaufs
Angenommen der Nutzer ist auf der "Edit"-Seite
Wenn der Nutzer Änderungen in seinem Lebenslauf vornimmt
Und der Nutzer klickt auf "Confirm"
Dann werden die Änderungen gespeichert

Abschnitt 3: Extraktion

Scenario 7: Ansehen von Tutorials
Angenommen der Nutzer ist auf der "Tools"-Seite
Wenn der Nutzer auf das Tutorial klickt
Dann sieht er das Tutorial

Scenario 8: Passwortänderung
Angenommen der Nutzer ist auf seinem Profil eingeloggt
Wenn der Nutzer auf "Reset password" klickt
Dann wird sein Passwort zurückgesetzt

Scenario 9: Ausloggen
Angenommen der Nutzer ist eingeloggt
Wenn der Nutzer auf "Logout" klickt
Dann wird er ausgeloggt

Das sind die Gherkin-Dateien, die die verschiedenen Features und Szenarien für die CVParser-Website basierend auf den gegebenen User Stories beschreiben.

ChatGPT hat über jedem Szenario noch den Feature-Titel und die User Story dazu ausgegeben.

Anhang C

Studie

C.1 Demografische Daten

Nr.	Alter	aktueller Studiengang	Abschluss	Abschluss Fach
1	20-24	Informatik	Abitur	-
2	25-29	Wirtschaft	Abitur	-
3	25-29	Ang.Informatik	Bachelor	Ang. Informatik
4	25-29	Informatik	Bachelor	Informatik
5	20-24	-	Ausbildung	Elektroniker IuK
6	25-29	Informatik	Bachelor	Informatik
7	20-24	-	Ausbildung	Physiotherapeut
8	-	Informatik	Abitur	-
9	25-29	Englisch, Politik	Abitur	-
10	25-29	Elektrotechnik	Master	Energietechnik
11	20-24	Mathematik	Abitur	-
12	20-24	Informatik	Abitur	-

Tabelle C.1: Demografische Daten der Studie.

Das Zeichen ”-“ bedeutet, dass es keine Angabe vom Probanden gab. Wenn hier kein Studiengang angegeben wird, bedeutet dies, dass die Person aktuell nicht studiert.

C.2 Beschreibung der Objekte der Studie

- *Testspezifikation zu CVParser*: Die Testspezifikation zu der Kommentare geschrieben werden und die verbessert werden soll. Die Szenarien der Testspezifikation werden dem Probanden einzeln gezeigt.
- *Visionsvideo zu CVParser*: Video, dass den Probanden mit den Funktionen von CVParser vertraut machen soll.
- *ChatGPT*: Das Large Language Model, dass zu Verbesserung der Testspezifikation in der Methode *mit ChatGPT* zum Einsatz kommt.
- *Kommentare der Stakeholder*: Nachdem die Kommentare verfasst wurden, werden sie genutzt, um die Testspezifikation zu verbessern.
- *Tabelle der Verbesserungen*: Es wird für jeden Probanden eine Tabelle in Google Tabellen erstellt. Diese Tabelle enthält beide Verbesserungen die *mit ChatGPT* und *ohne ChatGPT* erstellt wurden, die bewertet werden sollen.
- *LimeSurvey*: Ist die Umfrage-Plattform auf der die Fragebögen vom Probanden ausgefüllt werden.
- *BigBlueButton*: Ist die Präsentationsplattform, auf der das Visionsvideo gezeigt wird. Außerdem wird die Studie über diese Plattform moderiert.

C.3 Konzept der Verbesserung mit ChatGPT

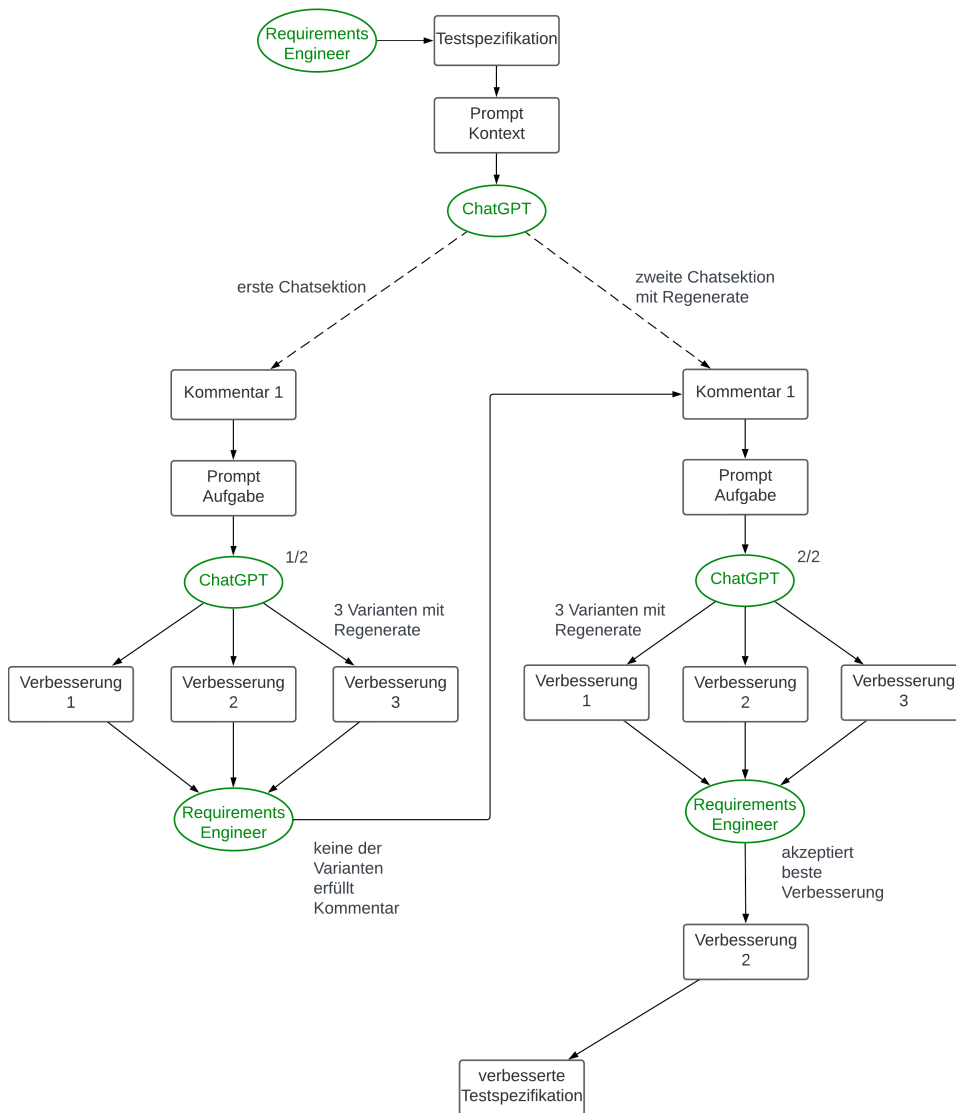


Abbildung C.1: Beispiel für den Prozess der Verbesserung von der Testspezifikation mit ChatGPT mit zwei Chatsektionen und drei Varianten von Verbesserungen ($n = 2$ und $m = 3$ bezogen auf Abbildung 5.3)

C.3.1 Beispiel: Methode Verbesserung der Testspezifikation mit ChatGPT (Dialog)

User

Auf der Webseite CVParser kann man sein LinkedIn-Profil in einen Lebenslauf umwandeln. Diese Webseite wird nun beschrieben.

Beschreibung:

””””

Zum Anfang ist man nicht angemeldet und befindet sich auf der Seite zum Anmelden. Hier kann man auf einen “create link” klicken, wodurch man zu einem Registrierungsformular weitergeleitet wird. Beim Registrierungsformular gibt man seine E-Mail und Passwort ein und wiederholt das Passwort. Anschließend klickt man auf den “create” Knopf und erhält eine Email mit einem Bestätigungscode. Wenn man den Bestätigungscode bei “Input code here” einfügt und auf den “create” Knopf klickt, wird ein Account bei CVParser erstellt. Man befindet sich nun wieder auf der Anmelde Seite, hier kann man E-Mail und Passwort eingeben und auf den “Login” Knopf drücken und wird dann eingeloggt. Man kann jetzt durch die verschiedenen Bereiche der Seite navigieren. Bei “Tool” kann man einen Lebenslauf extrahieren. Man kann hier sein LinkedIn Profil als Text einfügen. Nun klickt man auf “extract” , dann werden die Informationen des LinkedIn Profils als Lebenslauf dargestellt und in der Datenbank von CVParser gespeichert. Bei “Edit” kann man seinen Lebenslauf bearbeiten. Man kann seinen Lebenslauf anpassen, auf “Confirm” klicken und die Änderungen werden gespeichert. Bei “Tools” kann man Tutorials sehen, wenn man auf ein Tutorial klickt, dann kann man es sich ansehen. Man kann durch den Smiley auf sein Profil. Hier kann man mit “Reset password” sein Passwort zurücksetzen. Außerdem kann der Nutzer, während er eingeloggt ist, auf “Logout” klicken und wird ausgeloggt.

””””

für diesen Kontext gibt es nun einen Auszug aus der Testspezifikation in Gherkin:

””””

Feature 1: Registrierung bei CVParser

Als unregistrierter Nutzer möchte ich mich registrieren, damit ich einen Account bei CVParser habe.

Szenario 1: Registrierung eines neuen Benutzers

Angenommen der Nutzer befindet sich auf der Anmeldeseite

Wenn der Nutzer auf den ”Create” Link klickt

Dann sollte der Nutzer zum Registrierungsformular weitergeleitet werden

Wenn der Nutzer seine E-Mail und ein Passwort eingibt

Und das Passwort wiederholt
Und auf "Create" klickt
Dann sollte der Nutzer eine Bestätigungs-E-Mail erhalten
...
...
""""

Im nächsten Schritt werden Kommentare von Stakeholdern präsentiert, die Verbesserungen zu den Testspezifikationen beschreiben.

ChatGPT

Die Testspezifikation in Gherkin scheint die Funktionalitäten der CVParser-Webseite gut abzudecken. Sie beschreibt in einzelnen Features und Szenarien, wie sich ein Nutzer registriert, seinen Lebenslauf erstellt und bearbeitet, sein Passwort zurücksetzt und Tutorials ansieht. Jedes Feature hat klare Schritte und erwartete Ergebnisse definiert.

Wenn diese Tests implementiert und ausgeführt werden, können sie die Funktionen der Webseite auf ihre Richtigkeit überprüfen und sicherstellen, dass die einzelnen Komponenten wie Registrierung, Lebenslaufextraktion, Bearbeitung und andere wie erwartet funktionieren.

Ist etwas unklar oder gibt es spezifische Fragen zu den einzelnen Features?

User

Du bist ein Requirements Engineer und sollst die Testspezifikation in Gherkin verbessern. Zu Szenario 6 wurde folgender Kommentar geschrieben: "In welchem Form ist das Tutorial, das der Benutzer sehen sollte? Ein Video oder eine Instruktion in Form von Texte? Ich denke das Szenario hier kann detaillierter sein." Verbessere die Testspezifikation entsprechend.

ChatGPT

Natürlich, ich werde das Szenario 6 überarbeiten, um klarer darzustellen, welches Format das Tutorial hat und wie der Nutzer es sehen kann:

Feature 6: Tutorials anzeigen

Als Nutzer möchte ich Tutorials anschauen, um CVParser effektiv nutzen zu können.

Szenario 6: Anzeigen von Tutorials

Angenommen der Nutzer ist auf der Seite "About"
 Wenn der Nutzer auf die Option "Tutorials" oder "Anleitungen" klickt
 Dann sollte der Nutzer eine Auswahl an verfügbaren Tutorials sehen
 Und jedes Tutorial sollte mit einer klaren Kennzeichnung versehen sein,
 ob es sich um ein Video oder eine schriftliche Anleitung handelt
 Wenn der Nutzer auf ein bestimmtes Tutorial klickt
 Dann sollte das Tutorial in dem ausgewählten Format angezeigt werden
 Und der Nutzer sollte in der Lage sein, das gesamte Tutorial anzusehen
 oder zu lesen

Diese Überarbeitung soll sicherstellen, dass die Auswahl und das
 Ansehen von Tutorials klarer definiert sind, indem sowohl das Format
 der Tutorials als auch die Schritte zur Anzeige präziser beschrieben
 werden.

C.4 Umfragen zur Studie

Szenario 1: Registrierung eines neuen Benutzers
 Angenommen der Nutzer befindet sich auf der Anmeldeseite
Wenn der Nutzer auf den "Create" Link klickt
Dann sollte der Nutzer zum Registrierungsformular weitergeleitet werden
Wenn der Nutzer seine E-Mail und ein Passwort eingibt
Und das Passwort wiederholt
Und auf "Create" klickt
Dann sollte der Nutzer eine Bestätigungs-E-Mail erhalten

Kommentar:

• **Fragenkatalog**
Welche neuen Funktionen wünschen Sie sich / fehlen Ihnen?
 Möchten Sie eine **funktionale Änderung** haben? -> Welche?
 Ist ein **Ablauf zu umständlich**? -> Inwiefern soll dieser einfacher sein?
 Fehlt Ihnen ein **alternatives Szenario**? -> Welches? und was soll passieren?
Was soll passieren, **wenn etwas nicht funktioniert**?
 Fehlt vielleicht eine **genauere Beschreibung eines Vorgangs**? -> Was soll passieren?

Abbildung C.2: Umfrage zur Sammlung von Kommentaren

*Nun werden folgende Aussagen getätigt.
 Geben sie Ihre Zustimmung jeweils für Verbesserung 1 und Verbesserung 2 von 1-6 an.
 1: stimme überhaupt nicht zu.
 2: stimme nicht zu
 3: stimme eher nicht zu
 4: stimme eher zu
 5: stimme zu
 6: stimme voll zu.

	Verbesserung 1						Verbesserung 2					
	1	2	3	4	5	6	1	2	3	4	5	6
Die Verbesserung hat meinen Kommentar gut berücksichtigt.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Verbesserung ist gut formuliert und leicht zu verstehen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

*Wenn Sie die Verbesserung akzeptieren, bedeutet das, dass Sie bereit sind, die vorgeschlagene Änderung in Ihrem Programm umzusetzen. Ihre Entscheidung sollte auf Grundlage des Kommentares getroffen werden. Sie dürfen auch beide Verbesserungen akzeptieren.

	Ja	Nein
Ich akzeptiere Verbesserung 1	<input type="radio"/>	<input type="radio"/>
Ich akzeptiere Verbesserung 2	<input type="radio"/>	<input type="radio"/>

Abbildung C.3: Umfrage zur Sammlung von Bewertungen

Kommentar 1: Ich möchte mich nicht registrieren müssen, um den Dienst nutzen zu können.

Verbesserung 1	Verbesserung 2
neues Szenario: Kontoerstellung ohne Registrierung	Szenario 1: Kontoerstellung ohne Registrierung
Angenommen der Nutzer befindet sich auf der Anmeldeseite	Angenommen der Nutzer befindet sich auf der Anmeldeseite
Wenn der Nutzer auf "Weiter ohne Registrierung" klickt	Wenn der Nutzer "ich möchte mich nicht registrieren" ankreuzt
Dann sollte der Nutzer direkt zum Tool weitergeleitet werden, um seinen Lebenslauf zu erstellen	Und auf den "Create" Link klickt
	Dann zeigt die Webseite „wir brauchen ein mal Ihre E-Mail Adresse, um Ihnen einen Bestätigungslink zu schicken“
	Wenn der Nutzer seine E-Mail eingibt
	Und auf "Create" klickt
	Dann sollte der Nutzer eine Bestätigungs-E-Mail erhalten

Abbildung C.4: Beispiel für Tabelle der Verbesserungen *mit ChatGPT* und *ohne ChatGPT* zu Szenario 1

C.5 Beispiel: Fehlerkorrektur der Fehlertypen

Beispiel: Syntaxfehler
 Dann: Der Nutzer sollte die Option haben, die Zwei-Faktor-Authentifizierung zu aktivieren.
 Und wenn: Der Nutzer die Zwei-Faktor-Authentifizierung aktiviert.
 Dann: Ein Bestätigungscode sollte per SMS an die registrierte Telefonnummer

des Nutzers gesendet werden.

Korrektur:

Dann Der Nutzer sollte die Option haben, die Zwei-Faktor-Authentifizierung zu aktivieren.

Wenn der Nutzer die Zwei-Faktor-Authentifizierung aktiviert

Dann Ein Bestätigungscode sollte per SMS an die registrierte Telefonnummer des Nutzers gesendet werden.

Hier gibt es mehrere Korrekturen. Es haben sich ":" und "." eingeschlichen, diese werden entfernt. In diesem Fall ist "Und wenn" falsch, da die Abfolge Dann, Und, Dann nicht erlaubt ist. Hier sollte stattdessen nur *Wenn* verwendet werden.

Beispiel: Formulierungsfehler

Wenn der Nutzer eigene Kategorien hinzufügen oder bestehende Kategorien bearbeiten möchte

Dann sollte der Nutzer die Möglichkeit haben, neue Kategorien anzulegen oder bestehende zu bearbeiten

Und nach dem Anpassen auf "Confirm" klicken können

Dann sollten die Änderungen, einschließlich neuer Kategorien, erfolgreich gespeichert werden

Korrektur:

Wenn der Nutzer eine neue Kategorie anlegt oder bestehende bearbeitet

Und nach dem Anpassen auf "Confirm" klickt

Dann sollten die Änderungen, einschließlich neuer Kategorien, erfolgreich gespeichert werden

Im *Wenn* Schritt dieses Beispiels wird keine Aktion des Nutzers beschrieben, sondern seine Intention. Im *Dann* Schritt wird eine Möglichkeit für den Nutzer beschrieben. Das ist prinzipiell möglich, aber wenn eine Möglichkeit des Nutzers beschrieben wird, dann kann darauf keine Reaktion des Systems folgen. Somit muss der *Dann* Schritt in einen *Wenn* Schritt umformuliert werden. Der erste *Wenn* Schritt ist somit redundant und wird gelöscht. Zusätzlich muss auch die Formulierung "klicken können" in "klickt" umformuliert werden, somit ist es einheitlich mit dem verbesserten Schritt.

Beispiel: Und-Fehler:

Wenn der Nutzer sein Passwort eingibt und auf "bestätigen" klickt.

Korrektur:

Wenn der Nutzer sein Passwort eingibt

Und auf "bestätigen" klickt

Anhang D

Ergebnisse und Auswertung

Skala	Erfüllungsgrad		Verständlichkeit	
	Mit ChatGPT	Ohne ChatGPT	Mit ChatGPT	Ohne ChatGPT
6	44	35	30	33
5	9	9	17	14
4	3	7	10	8
3	2	5	3	3
2	2	4	0	0
1	0	0	0	2

Tabelle D.1: Häufigkeiten von 1-6 in den Kategorien Verständlichkeit und Erfüllungsgrad

Proband Nr.	Zeit [min]		Akzeptanzquote	
	Mit ChatGPT	Ohne ChatGPT	Mit ChatGPT	Ohne ChatGPT
1	22	23	5/5	3/5
2	23	23	4/6	4/6
3	20	24	3/4	3/4
4	12	15	6/6	3/6
5	19	15	3/5	3/5
6	22	26	6/6	6/6
7	23	23	6/6	5/6
8	23	23	6/6	5/6
9	24	32	4/5	4/5
10	10	24	4/4	3/4
11	13	12	2/3	3/3
12	7	33	3/4	2/4

Tabelle D.2: Zeit zur Verbesserung der Testspezifikation und Akzeptanzquote der Probanden *mit ChatGPT* und *ohne ChatGPT*

Proband Nr.	a-Zeit [min]	
	Mit ChatGPT	Ohne ChatGPT
1	4.40	7.67
2	5.75	5.75
3	6.67	8.00
4	2.00	5.00
5	6.33	5.00
6	3.67	4.33
7	3.83	4.60
8	3.83	4.60
9	6.00	8.00
10	2.50	8.00
11	6.50	4.00
12	2.33	16.50

Tabelle D.3: Berechnete Werte für a-Zeit

Literaturverzeichnis

- [1] Deepali Bajaj, Anita Goel, S. C. Gupta, and Hunar Batra. MUCE: a multilingual use case model extractor using GPT-3. *International Journal of Information Technology*, 14(3):1543–1554, May 2022.
- [2] Roaa Elghondakly, Sherin Moussa, and Nagwa Badr. Waterfall and agile requirements-based model for automated test cases generation. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 607–612. IEEE.
- [3] Chris Rupp and SOPHIST-Gesellschaft für Innovatives Software-Engineering. *Requirements-Engineering und -Management: das Handbuch für Anforderungen in jeder Situation*. Hanser, München, 7., aktualisierte und erweiterte auflage edition, 2021.
- [4] SmartbearSoftware. Cucumber - gherkin reference, 2019, last accessed: 30.10.2023
<https://cucumber.io/docs/gherkin/reference/>.
- [5] David Chelimsky, editor. *The RSpec book: behaviour-driven development with RSpec, Cucumber, and Friends*. Pragmatic, Lewisville, Tex, 2010. OCLC: ocn551430947.
- [6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [7] Kun Ruan, Xiaohong Chen, and Zhi Jin. Requirements modeling aided by ChatGPT: An experience in embedded systems. In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, pages 170–177. IEEE.
- [8] Danning Xie, Byungwoo Yoo, Nan Jiang, Mijung Kim, Lin Tan, Xiangyu Zhang, and Judy Lee. Impact of large language models on generating software specifications, 06 2023.
- [9] Alessandro Fantechi, Stefania Gnesi, Lucia Passaro, and Laura Semini. Inconsistency detection in natural language requirements using ChatGPT: a preliminary evaluation. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 335–340. IEEE.
- [10] R. A. Fokam Piam. Erhebung und Validierung von testbaren Anforderungen durch Visionvideos, Masterarbeit 2023, Gottfried Wilhelm Leibniz Universität Hannover.
- [11] OpenAI. Gpt-4 technical report, 2023.

- [12] Boqi Chen, Kua Chen, Shabnam Hassani, Yujing Yang, Daniel Amyot, Lysanne Lessard, Gunter Mussbacher, Mehrdad Sabetzadeh, and Dániel Varró. On the use of GPT-4 for creating goal models: An exploratory study. In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, pages 262–271. IEEE.
- [13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.
- [14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [15] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling*, 22(3):781–793, June 2023.
- [16] Elisabeth Métais, Farid Meziane, Vijayan Sugumaran, Warren Manning, and Stephan Reiff-Marganiec. Natural language processing and information systems: 28th international conference on applications of natural language to information systems, NLDB 2023, derby, UK, June 21–23, 2023, proceedings.
- [17] Hans-Georg Fill, Peter Fettke, and Julius Köpke. Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, pages 3:1–15 Pages, April 2023. Artwork Size: 3:1-15 Pages Publisher: Enterprise Modelling and Information Systems Architectures (EMISAJ).
- [18] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9), Jan. 2023.
- [19] OpenAI. "chatgpt", last accessed: 02.12.2023 <https://openai.com/blog/chatgpt>.
- [20] Helmut Degen and Stavroula Ntoa. Artificial intelligence in HCI: 4th international conference, AI-HCI 2023, held as part of the 25th HCI international conference, HCII 2023, copenhagen, denmark, july 23–28, 2023, proceedings, part II.
- [21] OpenAI. "models", last accessed: 03.12.2023 <https://platform.openai.com/docs/models>.
- [22] Leonard Peter Binamungu, Suzanne M. Embury, and Nikolaos Konstantinou. Characterising the Quality of Behaviour Driven Development Specifications.

- In Viktoria Stray, Rashina Hoda, Maria Paasivaara, and Philippe Kruchten, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 383, pages 87–102. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Business Information Processing.
- [23] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. Van Der Werf, and Sjaak Brinkkemper. Improving agile requirements: the Quality User Story framework and tool. *Requirements Engineering*, 21(3):383–403, September 2016.
- [24] Humaid Mollah and Petra Van Den Bos. From user stories to end-to-end web testing. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 140–148. IEEE.
- [25] Juan Zhai, Yu Shi, Minxue Pan, Guian Zhou, Yongxiang Liu, Chunrong Fang, Shiqing Ma, Lin Tan, and Xiangyu Zhang. C2S: translating natural language comments to formal program specifications. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 25–37, Virtual Event USA, November 2020. ACM.
- [26] Leon Chemnitz, David Reichenbach, Hani Aldebes, Mariam Naveed, Krishna Narasimhan, and Mira Mezini. Towards Code Generation from BDD Test Case Specifications: A Vision. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 139–144, Melbourne, Australia, May 2023. IEEE.
- [27] Microsoft. Introduction to prompt engineering 2023, last accessed: 01.12.2023 <https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering>.
- [28] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [29] V.R. Basili and H.D. Rombach. The tame project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, 1988.
- [30] Jürgen Bortz and Christof Schuster. *Statistik für Human- und Sozialwissenschaftler*. Springer-Lehrbuch. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [31] SmartbearSoftware. Quick p value from z score calculator - social science statistics, 2023, last accessed: 12.12.2023 <https://www.socscistatistics.com/pvalues/normaldistribution.aspx>.
- [32] Jürgen Bortz, Gustav A. Lienert, and Klaus Boehnke. *Verteilungsfreie Methoden in der Biostatistik: mit 247 Tabellen*. Springer-Lehrbuch. Springer, Berlin Heidelberg, 2., korrigierte und aktualisierte aufl edition, 2000.

