

Interaktive Mustergenerierung als Funktionserweiterung des natürlichsprachlichen Dialogsystems JustLingo

Bachelorarbeit
von

Fabian Wiest

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Ralf H. Reussner
Betreuende Mitarbeiter:	Dipl.-Inform. Alexander Wachtel Dipl.-Inform. Sebastian Weigelt

Bearbeitungszeit: 20. August 2014 – 19. Dezember 2014

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 19. Dezember 2014

.....
(**Fabian Wiest**)

Abstract

Das Softwaresystem JustLingo ist ein Dialogsystem, das für die Arbeit mit Excel[®] entwickelt wurde. JustLingo ermöglicht es natürlichsprachliche Eingaben anhand von Mustern in die Formelsprache der Tabellenkalkulationssoftware zu übersetzen. Motiviert durch die schwere Erweiterbarkeit wird in dieser Arbeit eine Schnittstelle zur automatisierten Generierung neuer Muster entwickelt. Über die Mustergenerierung können Nutzer, ohne den Quellcode zu verändern, JustLingo neue Funktionalitäten und Wörter beibringen. Dabei erklärt der Benutzer in natürlichsprachlichem Englisch den Aufbau seines Musters an das Dialogsystem. Bei der Umsetzung wurde ein Dialogsystem gemischter Initiative gewählt, damit der Erstellungsprozess durch das System mittels gezielter Fragen und Hinweisen unterstützt werden kann. Die abschließende Evaluation hat gezeigt, dass eine automatisierte Erweiterung mit Hilfe eines Dialogsystems umsetzbar ist und funktioniert. In der zugehörigen Studie wurden von 10 verschiedenen Teilnehmern jeweils drei mathematische Probleme, unterschiedlicher Schwierigkeit, zum Wortschatz des Dialogsystems hinzugefügt. Dabei wurden von insgesamt 641 Eingaben 515 korrekt vom System erkannt.

Inhaltsverzeichnis

1. Einleitung	1
1.1. Einleitung	1
1.2. Motivation	1
1.3. Zielsetzung	2
1.3.1. Erweiterbarkeit	2
1.3.2. Systemunterstütztes Arbeiten	2
1.3.3. Abgrenzung	2
1.4. Aufbau der Arbeit	3
2. Grundlagen	5
2.1. Dialogsysteme	5
2.1.1. Dialoginitiative	5
2.1.2. Zielgerichtetes Argumentieren	6
2.2. Ausgangspunkt: Stand JustLingo	6
2.3. Verbesserungsmöglichkeiten	8
3. Konzept	9
3.1. Analyse der Fragestellung	9
3.1.1. Muster in JustLingo	10
3.1.2. Automatisches Lernen	11
3.2. Konzeptionelle Umsetzung	12
3.2.1. Systemaufbau	12
3.2.2. Abspeichern eines Musters	14
3.2.3. Benutzen der Muster	14
4. Implementierung	15
4.1. Systemarchitektur	15
4.1.1. Technische Voraussetzungen	15
4.1.2. Benutzeroberfläche	15
4.1.3. Model-View-Presenter Architektur	16
4.2. Mustererweiterung-Einheit	18
4.2.1. Vorverarbeitung	18
4.2.2. Kontextinterpretation	21
4.2.3. Musterinterpretation	23
4.3. Dialogmanagement	27
4.3.1. Answererzeugung	27
4.3.2. Generelles Dialogmanagement	29
4.3.3. Spezielles Dialogmanagement	30
4.3.4. Kontextspeicherung	34
4.4. Datenbanken	35
4.4.1. Aufbau der Datenbanken	35
4.4.2. Speichern eines Musters	37

5. Evaluation	41
5.1. Aufbau und Durchführung	41
5.2. Auswertung der erhobenen Daten	43
5.3. Erkenntnisse	50
6. Verwandte Arbeiten	51
6.1. Natural Language Programming System - NLC	51
6.2. Pegasus	52
6.3. Metafor	52
6.4. VoiceTone®	53
6.5. Mensch-Roboter Interaktion und Kommunikation	53
6.6. SmartSynth	54
6.7. Mercury Flight Reservation System	55
7. Zusammenfassung	57
8. Ausblick	59
Literaturverzeichnis	61
Abbildungsverzeichnis	63
Tabellenverzeichnis	65
9. Anhang	67
A. Fragebogen der Evaluation zur Bachelorarbeit	68
B. Vokabelliste zur Evaluation	73

1. Einleitung

1.1. Einleitung

Die von Microsoft[®] entwickelte Tabellenkalkulationssoftware Excel[®] ist ein mächtiges Werkzeug zum Berechnen von statistischen Funktionen und Auswerten von Daten. Dabei wird in Excel[®] eine eigens eingeführte Formelsprache zur Berechnung dieser Formeln verwendet. Vor allem für Neueinsteiger stellt diese Formelsprache eine Barriere dar. Aber auch erfahrene Excel[®]-Nutzer kommen beim Eingeben von komplexeren Funktionen an ihre Grenzen. Aus der Idee heraus die Tabellenkalkulationssoftware mit natürlicher Sprache zu steuern ist das Werkzeug JustLingo entstanden. JustLingo nimmt natürlichsprachliche Äußerungen von Nutzern entgegen und erzeugt daraus Excel[®]-Formeln. Um dabei die Benutzung möglichst angenehm zu machen, wurde zur Umsetzung ein interaktives Dialogsystem gewählt. Dadurch kann das System direkt auf die Nutzereingaben reagieren und bei auftretenden Fehlern gezielte Hinweise zur Behebung ausgeben. Durch den entstehenden Dialog kann der Nutzer die Arbeitsweise von JustLingo besser nachvollziehen und lernt bei jeder Benutzung neue Details.

1.2. Motivation

Bei der Betrachtung des JustLingo Prototyps ist aufgefallen, dass zwar bereits viele mathematische Funktionen verstanden werden, aber die Ergänzung des Wortschatzes nur durch den Eingriff in die Programmierung möglich ist. Motiviert durch diese Schwierigkeit soll in dieser Arbeit dem Nutzer ein Werkzeug angeboten werden, dass es möglich macht über das Dialogsystem JustLingo neue Funktionen beizubringen. Der Lernprozess soll dabei nicht über das Eingeben einer langen Anweisung, die alle nötigen Informationen enthalten muss, realisiert werden. Diese Herangehensweise wurde in der Forschung bereits mehrmals durch Systeme wie NLC, Pegasus oder Metafor benutzt(Siehe Kapitel 6). Stattdessen wird ein anderer Ansatz gewählt, der sich an der Weise, wie Menschen Informationen lernen, orientiert. Bei der Ausbildung in Schulen oder Universitäten erfolgt das Lernen nicht indem jeder Schüler oder Student ein Fachbuch zum Lesen bekommt und so automatisch das gesamte Wissen aufnimmt. Vielmehr baut das Bildungssystem auf Lehrer und Dozenten auf, die den Auszubildenden ihr Wissen mittels schrittweiser Erklärungen kommunizieren. Dabei erfolgt das Lernen über den Dialog, den beide Teilnehmer miteinander führen. Gleichzeitig hat der Empfänger des Wissens die Möglichkeit bei Missverständnissen Fragen zu stellen, um diese aufzulösen. Nach diesem Prinzip soll das automatische Lernen

in dieser Arbeit umgesetzt werden. Der Benutzer erklärt Schritt für Schritt dem System eine neue Funktion. Dabei versucht das System den Nutzer über gezielte Fragen anzuleiten und noch offene Punkte anzusprechen. Die Zusammenarbeit zwischen Nutzer und System spielt beim gemeinsamen Lösen eines Problems dabei eine große Rolle (vergleiche [Ort14]).

1.3. Zielsetzung

Der Schwerpunkt dieser Arbeit liegt in der Beantwortung der Frage, ob es möglich ist, für bestehende Dialogsysteme eine dialoggesteuerte und automatisierte Erweiterung zur Generierung neuer Muster zu erstellen. Dabei soll die Arbeitsweise des bestehenden Programmes nicht verändert werden und alle bisher implementierten Funktionalitäten erhalten bleiben. Gleichzeitig soll durch diese Ergänzung die modulare Struktur beibehalten werden, damit auch in Zukunft eine Erweiterung von JustLingo möglichst einfach zu realisieren ist.

1.3.1. Erweiterbarkeit

Da bisherige Erweiterungen normalerweise nur durch Eingriff in die Programmierung möglich sind, soll durch diese Arbeit dem Nutzer eine Schnittstelle geboten werden, die diese Aufgabe für ihn übernimmt. Dabei wird es möglich gemacht neue, dem System bisher unbekannte, Wörter und deren Bedeutungen beizubringen. Der Nutzer kann diese Funktionen anschließend ebenso, wie die von der Grundversion mitgelieferten, benutzen. Durch diese Erweiterung wird es möglich, komplexere Funktionen einmal zum Wortschatz des Systems hinzuzufügen und diese später bei Bedarf aus dem Wissen von JustLingo abzurufen. Während des Erstellungsprozess, kann der Nutzer nicht nur von der Grundversion mitgelieferte Funktionen zusammensetzen, sondern sich in den Eingaben auch auf von ihm selbst erstellte Muster berufen. Die Mustergenerierung soll also ein Werkzeug darstellen, dass die Individualisierung, von JustLingo, nach den Wünschen des Nutzers, ermöglicht.

1.3.2. Systemunterstütztes Arbeiten

Der Erstellungsprozess des neuen Musters soll dabei im Dialog mit dem System erfolgen. Sobald der Nutzer einen Wunsch zur Erweiterung geäußert hat, soll das System alle seine Ausgaben auf die Mustererstellung ausrichten. Dabei wird es dem Nutzer in Form von Fragen mitteilen, welche Bestandteile des neuen Musters noch offen sind. Sollte der Nutzer nicht wissen, was zu tun ist, kann er sich an diesen Fragen orientieren. Neben den Fragen wird auch eine Reihe an Hinweisen zur Benutzung der Mustererweiterung ausgegeben werden. Diese Hinweise sollen dabei zuerst sehr allgemein gehalten werden und bei weiteren auftretenden Missverständnissen immer präziser werden. Dabei sollen als letzter Ausweg dem Nutzer Beispielsätze zur Benutzung der Erweiterung mitgeteilt werden. Allgemein soll in der Mustererweiterung die Initiative des Systems erhöht werden. In der bisherigen Umsetzung liegt die Initiative im Dialog fast immer beim Nutzer. JustLingo gibt dem Nutzer bisher lediglich Angaben darüber, zu welchem Ergebnis die aktuelle Eingabe geführt hat und stellt bei auftretenden Missverständnissen und Problemen zur Auflösung Fragen. In der Mustergenerierung soll das System dahingegen die Initiative im Gespräch übernehmen und den Nutzer so gezielt bei der Erstellung unterstützen.

1.3.3. Abgrenzung

Die Mustererweiterung soll den Ablauf des bisherigen Systems nicht beeinflussen. Ebenso wenig sollen durch diese die Reaktionen des bestehenden Dialogsystems verändert werden. Alle Erweiterungen, die der Nutzer mit der Mustergenerierung erstellen kann, haben lediglich Einfluss auf den Wissensstand von JustLingo selbst. Durch diese Erweiterung sollen keine neuen Funktionalitäten von Excel[®] realisiert werden. Alle Interaktionen mit

dem Dialogsystem erfolgen vorerst in englischer Sprache erfolgen. Dennoch wird darauf geachtet, dass die erstellten Muster so angelegt sind, dass sie in Zukunft auch mit anderen Sprachen funktionieren können.

1.4. Aufbau der Arbeit

In Kapitel 2 werden zuerst theoretische Grundlagen von Dialogsystemen beschrieben. Anschließend wird kurz der Ausgangspunkt dieser Arbeit gezeigt. In Kapitel 3 wird zuerst auf konzeptioneller Ebene erklärt, wie die Erweiterung geplant war und auf welche Besonderheiten geachtet wurde. Danach wird in Kapitel 4 ausführlich die Umsetzung der im vorherigen Kapitel erläuterten Konzepte erklärt. In Kapitel 5 wird eine Studie beschrieben, die zur Evaluierung der umgesetzten Erweiterung durchgeführt wurde. Darauf folgt im 6. Kapitel die Beschreibung der zu dieser Arbeit verwandten Arbeiten. Abschließend folgt eine kurze Zusammenfassung und das Fazit dieser Arbeit.

2. Grundlagen

2.1. Dialogsysteme

Ein Dialogsystem baut sich nach Douglas Walton aus vier fundamentalen Bestandteilen auf: Die Teilnehmer, die Vorgehensweise der Teilnehmer, die Auswirkung der einzelnen Schritte, die die Teilnehmer unternehmen und das Ziel, das durch den Dialog verfolgt wird (vergleiche [Wal00]). Als Teilnehmer werden dabei der Antragssteller und der Auskunftgeber unterschieden. Unter dem Antragssteller ist der Nutzer des Systems zu verstehen, der durch seine Anfragen Informationen vom Auskunftgeber, dem Dialogsystem, erwartet. Unter der Vorgehensweise versteht Walton das Verhalten der Gesprächsteilnehmer. Dabei wurden in der Forschung bereits verschiedenste Ansätze gewählt. Bei Dialogsystemen wie NLC [BB79] und Metafor [LL05], handelt es sich um stille Zuhörer. Wohingegen zum Beispiel bei VoiceTone[®] [GTHT⁺06] und in dieser Arbeit sich die beiden Gesprächsteilnehmer abwechselnd und nacheinander äußern. Der dritte Bestandteil bezieht sich auf die Auswirkungen der Äußerungen auf den früheren Gesprächsverlauf. Interpretiert beispielsweise der Auskunftgeber eine Anfrage falsch, wird der Antragsteller mit seinen folgenden Eingaben bemüht sein eine Korrektur vorzunehmen. Sollten wiederum die Auskünfte des Antragstellers für eine vollständige Erfüllung einer Aufgabe sich als unzureichend erweisen, wird der Auskunftgeber versuchen die fehlenden Informationen von seinem Gesprächsteilnehmer zu erhalten. Am Ende des Dialoges steht immer ein Ziel, dass beide Parteien verfolgen. Im Allgemeinen kann angenommen werden, dass dabei beide Teilnehmer im Dialog auf dieses gemeinsame Ziel hinarbeiten. Denn der Erfolg eines Dialoges entsteht nur durch die Zusammenarbeit und den Austausch von Informationen der beiden Akteure, die sie gemeinsam zur Erfüllung der Aufgabe führt (vergleiche [Ort14]).

2.1.1. Dialoginitiative

Bei der Gestaltung eines Dialoges zwischen Mensch und Maschine gibt es verschiedene Ansätze in Bezug auf die Initiative, die beide Gesprächsteilnehmer zeigen müssen. Dabei wird aus Sicht des Programmes zwischen drei verschiedenen Strategien unterschieden (vergleiche [NFK⁺00]): Die rein systemseitige Gesprächsinitiative, die gemischte Initiative und die vom Nutzer ausgehende Initiative. Ein Dialogsystem, bei dem der Gesprächsverlauf hauptsächlich durch das System gesteuert wird, stellt in vielen Fällen Einschränkungen an den Nutzer. Es wird bei jeder Eingabe durch das System exakt vorgeschrieben, welche Informationen in der nächsten Schritt vom Nutzer erwartet werden. Meist wird dabei pro Frage

und Antwort nur eine Information ausgetauscht, wie es beispielsweise im Flugbuchungssystem Mercury [SP00] und dem in der Arbeit von AT&T evaluierten Telefonsystem der Fall ist (siehe [NFK⁺00]). Dahingegen bietet der Ansatz der gemischten Initiative den Vorteil, dass der Nutzer auch Informationen und Details zum Gespräch beitragen kann, nach denen das Dialogsystem im aktuellen Zustand nicht gefragt hat. Dieses Art der Initiative wurde von den Testpersonen in der Studie von AT&T als besonders gut empfunden. Wohingegen die rein vom Nutzer ausgehende Initiative als unbefriedigend empfunden wurde. Bei einem solchen Vorgehen stellt das System keinerlei Einschränkungen an den Nutzer im Bezug auf die Eingabe. Die Richtung, die der Dialog in diesem Gespräch annimmt, wird dabei rein durch den Nutzer vorgeben. Das System beschränkt sich in seinen Äußerungen auf Resultate und gezielte Fragen, die bei der Bewältigung von Aufgaben helfen sollen.

In dieser Arbeit wurde zur Realisierung der Ansatz der gemischten Initiative gewählt. Das System schlägt dem Nutzer vor, was er als nächstes angegeben soll und versucht über gezielte Fragen an noch fehlenden Informationen zu gelangen. Dabei kann der Nutzer selbst entscheiden, ob er den Empfehlungen des Dialogsystems folgt.

2.1.2. Zielgerichtetes Argumentieren

Eine wichtige Vorgehensweise zum Erreichen des Zieles eines Dialoges ist das sogenannte „goal-directed reasoning“ (deutsch: zielgerichtetes Argumentieren, siehe [Wal00]). Damit ist gemeint, dass ein Gesprächsteilnehmer alle seine Überlegungen, Fragen und Aussagen auf das Erfüllen eines vorher festgelegten Zieles ausrichtet. Dies kann nach Walton durch die vier im folgenden ausgeführten Systemreaktionen gewährleistet werden.

Der wichtigste Bestandteil ist das Stellen von Fragen. Dabei werden zwei verschiedene Fragetypen unterschieden: Die „Auswahlfragen“ und die so genannten „Suchfragen“. Bei den Auswahlfragen handelt es sich meist um Fragen, bei denen entweder ein Ja oder ein Nein als Antwort erwartet wird. Über solche Fragen lassen sich die Richtung, die ein Gesprächsverlauf annimmt sehr einfach limitieren, da immer nur eine bestimmte Menge an Abzweigungen zur Auswahl steht. Die sogenannten Suchfragen schränken den Dialogverlauf dahingegen nicht ein, da zu ihnen keine eindeutig festgelegten Antworten existieren. Diese Fragen werden vor allem benutzt, um die grobe Richtung eines Dialoges anzugeben und dem Nutzer mitzuteilen, was zur Erreichung des Zieles noch an Informationen nötig ist. Als zweiten Punkt nennt Walton das Aufstellen von Behauptungen. Damit ist gemeint, dass das System bestimmte Annahmen und Aussagen in Bezug auf die Nutzereingaben trifft. Der Nutzer kann das System dann anschließend verbessern, falls die getroffene Annahme falsch war. Das Dialogsystem muss daraufhin seine Annahmen zurückziehen und ändern können, was als dritter Bestandteil eines Dialoges identifiziert wurde. Als letzter Punkt nennt Walton, dass ein Dialogsystem auch in der Lage sein muss seine Argumente und Annahmen voranzutreiben und gegenüber dem Nutzer zu verteidigen. Dies kann beispielsweise durch die bereits beschriebenen Arten von Fragen geschehen.

2.2. Ausgangspunkt: Stand JustLingo

Die erste Version von JustLingo wurde bereits 2013 von einer Gruppe von Studenten im Rahmen der Lehrveranstaltung *Praxis der Softwareentwicklung* entworfen und entwickelt [KIT13]. Bei diesem Projekt handelte es sich um die Implementierung einer Erweiterung, ein so genanntes Add-In, für Microsoft[®] Excel[®]. Dabei wurde das System als separates Fenster in die Tabellenkalkulationssoftware eingebaut. JustLingo ermöglichte es Benutzern mathematische Formeln in natürlichsprachlichem Englisch an JustLingo zu erklären. Diese wurden automatisch in die von Excel[®] benutzte Formelsprache übersetzt und konnten zur Arbeit direkt in die geöffnete Tabelle eingefügt werden. Anschließend wurde JustLingo

durch die Bachelorarbeit von Philipp Voigt erweitert. In dieser Arbeit wurden die bereits bestehende Interpretation der Nutzereingaben komplett überarbeitet und um neue Funktionalitäten erweitert. Zu diesen zählt das Erkennen von komplexeren Zusammenhängen in den Nutzereingaben beim Formulieren von mathematischen Formeln. Über die sogenannten Referenzen kann der Nutzer nun mehrere Eingaben miteinander verbinden. Durch die Neuerung wurde die Eingabe komplexere Funktionen deutlich vereinfacht, da diese nun nicht mehr in einem Satz erfolgen müssen, sondern stückweise eingegeben werden können. Die zweite Neuerung, die in dieser Arbeit umgesetzt wurde, war ein Dialogsystem. Durch diese Erweiterung wurde JustLingo von einem passiven Zuhörer zu einem Dialogsystem mit teilweise gemischter Initiative verbessert. Bei der Implementierung wurde der in Abbildung 2.1 dargestellte modulare Systemaufbau gewählt. Dabei erhält der Nutzer über die Benutzeroberfläche die Möglichkeit, Eingaben an JustLingo zu übergeben. Diese werden innerhalb der Sprachanalyse-Einheit verarbeitet. Dabei werden zuerst alle Wörter in den Nutzereingaben annotiert, zu denen eine Bedeutung in den Systemdatenbanken gefunden werden konnte. Anschließend wird versucht die Nutzereingabe mathematisch zu interpretieren. Dazu identifiziert JustLingo zuerst alle in der Eingabe vorhandenen Operationen und Operanden. Danach versucht JustLingo die gefundenen Bestandteile zusammzusetzen. Schlussendlich werden die mathematischen Funktionen in die zugehörigen Excel[®]-Formeln übersetzt. Die annotierte Eingabe wird nun an die Dialogmanagement-Einheit übergeben. Diese versucht anhand der von der Sprachanalyse-Einheit erhaltenen annotierten Eingabe und dem im Dialog mit dem Nutzer vorliegenden Kontext, eine passende Systemantwort zur aktuellen Eingabe zu erzeugen. Die Antwort wird dann über die Benutzeroberfläche an den Nutzer ausgegeben. Bei den Systemantworten kann es sich um Aussagen über ausgeführte Aktionen und während der Interpretation entstandene Fragen handeln. Sollte dem Nutzer eine Frage ausgegeben werden erwartet die Dialogmanagement-Einheit in der nächsten Eingabe eine passende Antwort zu dieser Frage.

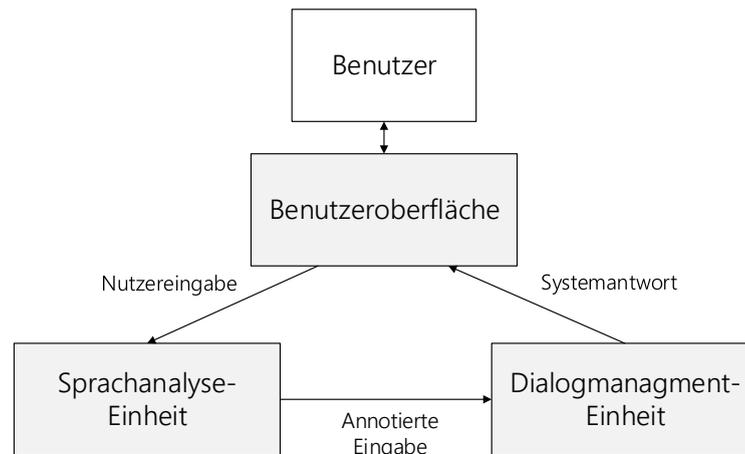


Abbildung 2.1.: Systemaufbau des JustLingo-Prototyps (*Quelle:* [Voi14])

2.3. Verbesserungsmöglichkeiten

Die Arbeit von Voigt stellte eine deutliche Verbesserung gegenüber der ersten Version des Projektes dar. Allerdings wird einem erfahrenen Excel[®]-Nutzer bei der Arbeit mit JustLingo schnell auffallen, dass der Wortschatz an erkannten Funktionen eingeschränkt ist. Eine Erweiterung um neue Funktionalitäten ist aber nur über das direkte Eingreifen in die Programmierung möglich. Dazu bräuchten außenstehende Entwickler aber Zugriff auf den Quellcode. Damit andere dennoch JustLingo für ihre persönlichen Zwecke erweitern können, wurde nach einer Möglichkeit gesucht, dies ohne Änderung der Programmierung möglich zu machen. Dabei kam die Idee auf, die Erweiterbarkeit über das bereits vorhandene Dialogsystem zu organisieren. Diese Arbeit widmet sich daher der Frage, ob es möglich ist das bereits bestehende Dialogsystem JustLingo um eine selbst lernende Komponente zu erweitern. Zusätzlich fiel bei näherem betrachten auf, dass die Answererzeugung des Prototyps statisch umgesetzt war. Dies machte eine Weiterverwendung so kompliziert, dass dieser Teil des Systems komplett neu geplant und umgesetzt wurde.

3. Konzept

Im Konzept Kapitel wird das allgemeine in dieser Arbeit gewählt Vorgehen beschrieben. Dazu wird zunächst die Fragestellung, die diese Arbeit beantworten soll, analysiert. Anschließend werden die Strukturen, die JustLingo zur Verarbeitung von natürlichsprachlichen Eingaben verwendet, beschrieben. Im letzten Abschnitt dieses Kapitels wird ein Einblick in die umgesetzte Implementierung der Mustererweiterung gezeigt.

3.1. Analyse der Fragestellung

In dieser Arbeit wurde der Schwerpunkt auf die automatisierte Erweiterbarkeit des Dialogsystems selbst gelegt. Dabei lag der Fokus auf dem selbständigen Lernen neuer Wörter, die dann zur weiteren Arbeit verwendet werden können. Damit ist zum einen gemeint, dass das Dialogsystem neue Bezeichnungen für bereits Bekanntes lernen kann. Zum anderen lag aber der Schwerpunkt darin, dem Nutzer ein Werkzeug an die Hand zu geben, mithilfe dessen es ihm möglich ist, mittels Erklärungen, dem Dialogsystem neue mathematische Funktionen beizubringen. Dem Nutzer soll die Möglichkeit gegeben werden Erweiterungen an JustLingo vorzunehmen, ohne den Quelltext bearbeiten zu müssen. Dies soll über das bereits vorhandene Dialogsystem geschehen. Dabei werden die vom Nutzer erhaltenen Informationen interpretiert und automatisch anhand der Beschreibungen der Wortschatz erweitert.

Zum Beispiel hat der Nutzer in seiner Tabellenkalkulationssoftware Daten über die Seitenlängen von verschiedenen gleichseitigen Dreiecken. Er möchte nun den Flächeninhalt eines dieser Dreiecke abhängig von der Kantenlänge berechnen lassen. Weder Excel[®] noch JustLingo kennen dazu die passende Formel. Der Nutzer entscheidet sich deshalb JustLingo die neue Formel zu erklären. Der entstehende Dialog zwischen Nutzer und System ist in Tabelle 3.1 zu sehen. Dabei beginnt der Nutzer indem er dem System mitteilt, dass er nun ein neues Muster erklärt. Gleichzeitig teil er den Namen der neuen Funktion mit. Anschließend erklärt er nach und nach die vollständige Formel und lässt diese schlussendlich speichern. Der Nutzer kann nun zu jeder seiner Kantenlängen, durch benutzen der neu erstellten Formel, den Flächeninhalt ausrechnen lassen und in seine Tabelle einfügen lassen. Im Folgenden soll nun genau betrachtet werden was ein solches Muster auszeichnet und welche Informationen zur Erstellung benötigt werden.

System
<i>How can I help you?</i>
Nutzer
<i>Create a new formula called trianglearea.</i>
System
<i>I created a new pattern called TRIANGLEAREA. May I ask you for the formula?</i>
Nutzer
<i>Sure! The formula is square x divided by 4.</i>
System
<i>Allright, I added $(SQUARE(x) / 4)$ to your pattern.</i>
Nutzer
<i>Now multiply the previous result by the root of 3.</i>
System
<i>Your formula was extended to $((SQUARE(x) / 4) * ROOT(3))$.</i>
Nutzer
<i>Save my new formula.</i>
System
<i>Your pattern was successfully added. You can now use TRIANGLEAREA(x).</i>

Tabelle 3.1.: Beispieldialog einer Mustererstellung

3.1.1. Muster in JustLingo

Die Spracherkennung in JustLingo läuft in zwei Schritten ab: Zuerst findet wie in den Grundlagen beschrieben eine sogenannte Schlüsselwortannotation statt, bei der den einzelnen Wörtern der Eingabe Bedeutungen zugeordnet werden. Um die Verbindung zwischen den einzelnen Wörtern herzustellen findet anschließend ein Abgleich mit den dem System bekannten Mustern statt. Aus den Mustern ergibt sich für das System die Bedeutung des Wortes in der Nutzereingabe. Gleichzeitig kann über die Muster zugeordnet werden, welche Systemreaktion der Nutzer durch seine Eingabe bezwecken will. Um also eine vollständige Erweiterbarkeit gewährleisten zu können, muss die in dieser Arbeit umgesetzte Erweiterung all diese Informationen dem Wortschatz von JustLingo hinzufügen.

Beispielhafter Aufbau eines Musters

In Abbildung 3.1 sind alle Bestandteile des zur Operation *Addieren* gehörenden Musters beschrieben. Wie zu sehen ist, gibt es für die Addition im Englischen mehrere Synonyme. Jedes dieser Synonyme wird dabei von JustLingo zur Vereinheitlichung auf das Muster „Sum“ abgebildet. Über das vorliegende Wort kann nun ausgesagt werden, dass es sich um einen Stellvertreter der Operation „Sum“ handelt. Bei besagtem Wort muss es sich also um ein Substantiv handeln, das nun mehrere Parameter zur korrekten Funktion erwartet. Um herauszufinden wie diese über natürliche Sprache angegeben werden können, findet ein Abgleich zwischen Nutzereingabe und den dem System bekannten Musteranwendungen statt. Zuletzt enthält das Muster einen Verweis für welche Formel in der Tabellenkalkulationssoftware es steht.

Bei der Umsetzung der Mustererweiterung mussten also alle diese Bestandteile bei der Erstellung miteinbezogen werden. Um die Benutzung aber möglichst transparent für den Nutzer zu machen, müssen dem System lediglich Informationen zur Formel, Anzahl der Eingabeparameter und einen Namen zur Identifikation angegeben werden. Das definieren eines Musters, sowie die Erstellung möglicher Musteranwendungen wird dabei automatisch durchgeführt.

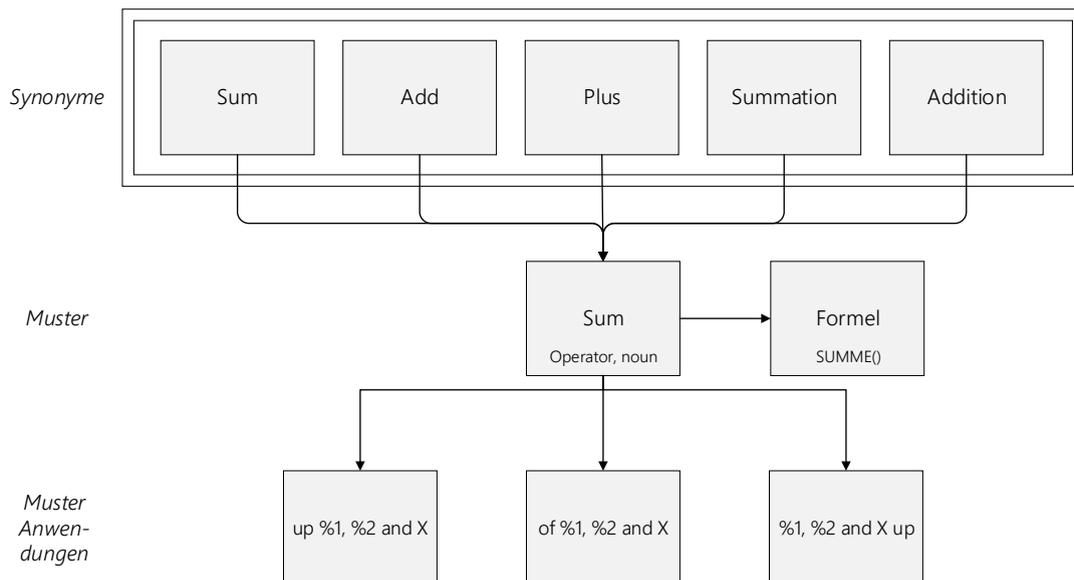


Abbildung 3.1.: Aufbau des Addieren-Musters

3.1.2. Automatisches Lernen

Es gibt in der Forschung verschiedenste Ansätze, um das Interagieren mit Computer über natürliche Sprache zu realisieren. Zum Beispiel über das Schreiben von Geschichten wie beim System Metafor [LL05]. Oder über multimodale Eingabemöglichkeiten [HG04]. In dieser Arbeit soll das kommunizieren neuer Muster ausschließlich über an das Dialogsystem gerichtete Texteingaben erfolgen. Dabei soll der Nutzer beim Erstellen das Gespräch mit dem System suchen und in erklärender Weise, möglichst wie ein Lehrer seinem Schüler neues Wissen vermittelt, auf das System zugehen. Das System selber reagiert dabei auf die Nutzereingaben und gibt dem Nutzer Rückmeldungen, aus denen der Nutzer Rückschlüsse darauf ziehen kann, was das System bisher verstanden hat. Das System selber soll im Gespräch auch die Initiative suchen und den Nutzer bei der Vollendung der Erstellung helfen, indem es gezielte Fragen stellt. Das Dialogsystem bemerkt automatisch, wenn ein Muster komplett und zum Abspeichern bereit ist. Daraufhin wird es den Nutzer fragen, ob dieser sein Muster nun abspeichern möchte. Nach einer Bestätigung wird automatisch das Erstellen der nötigen Datenbankeneinträge durchgeführt und die Funktion kann anschließend sofort, ohne Neustart des Systems, vom Nutzer verwendet werden. Dadurch hat sich das System im Dialog mit dem Nutzer neues Wissen komplett automatisch angeeignet.

3.2. Konzeptionelle Umsetzung

3.2.1. Systemaufbau

Der Systemaufbau dieser Arbeit ist als eine Erweiterung der bisherigen Version von JustLingo umgesetzt. Dabei wurde der in den Grundlagen beschriebene modulare Aufbau durch eine parallel ablaufende Modulreihenfolge erweitert. Eine Übersicht des Ablaufes ist in Abbildung 3.2 zu sehen. Die Idee die Interaktion mit JustLingo über ein Dialogsystem zu organisieren, wie es Voigt in seiner Arbeit [Voi14] eingeführt hat, wurde beibehalten und um spezielle Dialogabläufe zur Mustergenerierung erweitert.

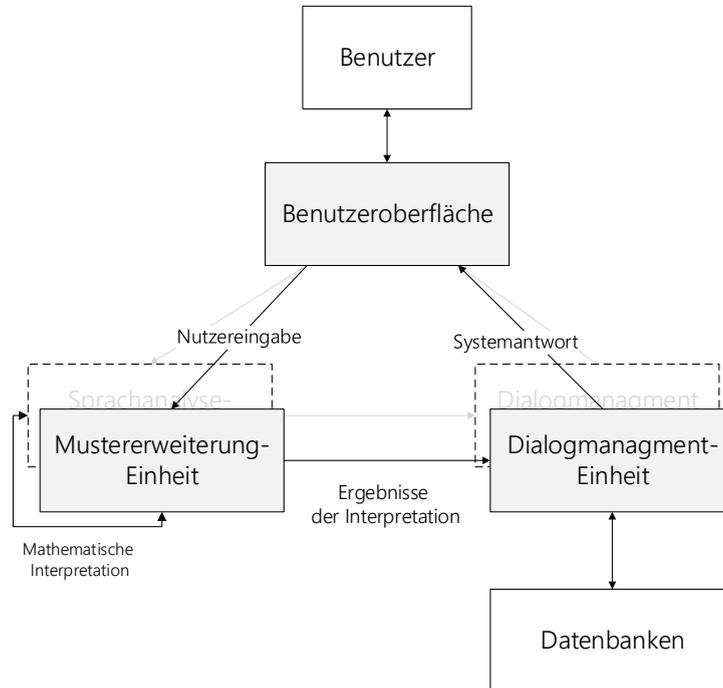


Abbildung 3.2.: Übersicht des Systemaufbaus

Von der Benutzeroberfläche aus wird die Nutzereingabe an die Verarbeitungspipeline von JustLingo weitergeleitet. Dort wird automatisch entschieden, um welche Art der Anfrage es sich handelt und wie mit der Verarbeitung fortgefahren werden soll. Wurde in der Nutzereingabe eine Anfrage zur Mustererweiterung festgestellt, wird anschließend die Mustererweiterungseinheit aufgerufen. Innerhalb dieses Moduls findet die gesamte Analyse der Nutzereingaben in Bezug auf die Mustererweiterung statt. Abhängig von den Eingaben werden dabei mögliche Antworten für den Nutzer erzeugt und Informationen aus den Eingaben extrahiert. Schlussendlich werden die entstandenen Antworten an die Dialogmanagementeinheit übergeben, die daraufhin entscheidet, welche Systemantwort an den Nutzer ausgegeben werden soll. Diese vorerst grobe Einführung wird nun auf konzeptioneller Ebene genauer beleuchtet. Die genaue Umsetzung dieser Module findet sich im Kapitel der Implementierung (siehe Kapitel 4.2).

Mustererweiterung-Einheit

Das bisherige Dialogsystem von JustLingo ist auf das Erkennen von mathematischen Funktionen zugeschnitten. Für die in der Mustergenerierung benötigten Aufgaben ist es nicht ausreichend. Deshalb wurde ein komplett neues Modul, die Mustererweiterung-Einheit zum bisherigen System hinzugefügt. Dieses Modul unterteilt sich wiederum in drei neue Bestandteile: Dazu gehört ein Teilmodul, das auf die Erfassung des vom Nutzer gewählten Namens ausgerichtet ist. Dieses Modul erkennt Erstbenennungen, Umbenennungen und unzulässige Benennungen. Zu nicht erlaubten Benennungen des neuen Musters zählen alle dem Dialogsystem bekannten Wörter, die bereits für andere Zwecke verwendet werden. Das zweite Teilmodul ist auf das Erkennen von mathematischen Formeln zugeschnitten. Dazu gehört das Verstehen und Zusammensetzen der Nutzereingaben, sowie das Erzeugen passender Ausgaben für den Nutzer. Um dem Nutzer die Möglichkeit anzubieten Funktionen anzugeben, die Parameter als Eingaben erwarten wurde zusätzlich ein weiteres Modul umgesetzt, das den Umgang mit Variablen übernimmt. Variablen sind ein völlig neuer Bestandteil des Dialogsystems, da die bisherige Version von JustLingo lediglich mit zur Eingabezeit fest definierten Parametern, wie Zahlen, Zellenangaben oder Spaltennamen umgehen kann. Diese können während der Mustergenerierung dennoch verwendet werden, da viele mathematische Formeln neben Variablen auch Konstanten enthalten können. Da auch schon in der Formelsprache der Tabellenkalkulationssoftware solche Funktionen existieren, wurde das Erstellen von neuen Mustern, die keine Eingabeparameter benötigen, auch berücksichtigt. Schlussendlich werden die aus den Nutzereingaben erkannten Informationen in eine Mustervorlage zwischengespeichert. Diese Vorlage wird bei einer Aufforderung zum Speichern automatisch in die Datenbank geschrieben. Für jedes neue Muster wird dabei eine neue Vorlage angelegt und es ist immer nur möglich ein Muster gleichzeitig zu erstellen.

Dialogmanagement-Einheit

In der Dialogmanagement-Einheit finden alle Entscheidungen, die im Zusammenhang mit dem Nutzer geführten Dialog getroffen werden, statt. Um das Dialogmanagement möglichst übersichtlich zu gestalten wurde es in mehrere logische Bestandteile aufgeteilt. Dabei ist jeder dieser Teile auf einen bestimmten Gesprächskontext spezialisiert. Als Einstiegspunkt der Dialogmanagement-Einheit wurde zunächst ein Modul zur generellen Interpretation der Nutzereingaben entworfen. In diesem befindet sich die Dialogauswahl immer dann, wenn im aktuellen Gesprächsverlauf kein Kontext existiert, oder das Thema gewechselt wird. Nachdem der Nutzer nun die Mustergenerierung gestartet hat wird das Dialogsystem sukzessive über Suchfragen versuchen das Gespräch zum erfolgreichen Erstellen eines Musters zu führen: Dazu wurde zu jedem Aspekt der Mustergenerierung ein spezielles Dialogmanagement entworfen. Einen Teil zum Finden des Namens, einen um die Formel zu erhalten und einen dritten, der den Nutzer beim Speichern und Bearbeiten unterstützt. Wenn das Dialogsystem sich im Namens- oder im Formelmodus befindet werden dem Nutzer dabei neben Fragen auch Hinweise zur Benutzung des Systems mitgeteilt. So wird beispielsweise im Formelmodus zuerst nach der Formel gefragt, dann wird ein Hinweis zur Benutzung von Variablen ausgegeben und am Ende werden dem Nutzer ganze Eingaben gezeigt, die ihm als Beispiel dienen sollen. Die Dialogmanagement-Einheit wurde so gestaltet, dass auf jede Nutzereingabe immer eine Systemausgabe folgt. Je nach Eingabe können in einem Durchlauf dem Nutzer aber auch mehrere Informationen mitgeteilt werden. Dies passiert zum Beispiel, wenn der Nutzer in seiner Anfrage zum Erstellen eines neuen Musters gleichzeitig den Namen mit angibt. Dann wird sowohl eine Bestätigung über den Übergang zur Mustergenerierung, als auch eine Ausgabe bezüglich des gefundenen Namens ausgegeben.

3.2.2. Abspeichern eines Musters

Die Speicherung eines neuen Musters kann erst erfolgen, sobald alle nötigen Informationen vom Nutzer an das System mitgeteilt wurden. Das System wird den Nutzer dann fragen, ob er das fertige Muster speichern möchte. Antwortet der Nutzer daraufhin bejahend beginnt das Schreiben in die Datenbanken. Jeder Bestandteil des Musters wird dabei in eine andere Datenbank geschrieben (siehe Kapitel 4.4). Zuerst wird der Name, der als Identifikator dient zur sogenannten Wörterdatenbank hinzugefügt. Danach wird automatisch ein Eintrag in der Befehlsdatenbank erstellt, der die Informationen zur Benutzung des Musters enthält. Dazu gehört die Anzahl der benötigten Parameter und die erzeugte Formel. Zuletzt wird ein Eintrag bezüglich der Benutzung in ganzen Sätzen in die Syntax-Datenbank vorgenommen. Konnten das Muster erfolgreich in den Datenbanken abgelegt werden, erhält der Nutzer eine Bestätigung. Anschließend wird die Mustererweiterung verlassen. Der Nutzer kann nun direkt sein neues Muster für Berechnungen benutzen, es wird kein Neustart des Systems benötigt.

3.2.3. Benutzen der Muster

Die mit der Mustergenerierung erstellten Muster können nun wie alle standardmäßig im Wortschatz von JustLingo vorhandenen Funktionen benutzt werden. Sie können als Parameter an andere Funktionen übergeben werden und sie können vom System automatisch in die aktuell geöffnete Tabelle geschrieben werden. Darüber hinaus können die Funktionen als Bestandteile weiterer Muster benutzt werden. Dadurch ist es möglich vor allem sehr komplexe Formeln zuerst in mehrere Teilformeln zu zerlegen und diese dann zusammensetzen (Getestet in der Evaluation 5).

4. Implementierung

Im Kapitel der Implementierung werden die technischen Einzelheiten der im Konzept vorgestellten Planung im Detail erklärt. Da JustLingo direkt in die Tabellenkalkulationssoftware eingebaut ist, wird zuerst auf die daraus resultierenden Voraussetzungen und dem allgemeinen Aufbau des Systems eingegangen. Danach werden alle verwendeten Datenstrukturen und einige für den Ablauf wichtige Algorithmen näher beschrieben.

4.1. Systemarchitektur

Bei der Implementierung von JustLingo wurde großen Wert auf Erweiterbarkeit, Wiederverwendbarkeit und einfache Systemwartung gelegt. Aus diesem Grund wurde das System nach dem Entwurfsmuster des *Model-View-Presenter* aufgebaut. Durch das gewählte Entwurfsmuster wird eine strikte Trennung zwischen der äußeren Ansicht (*View*), der Steuerlogik des Systems (*Presenter*) und der Programmlogik (*Model*) erreicht. Jede dieser drei Komponenten bietet zur Benutzung feste Schnittstellen und ist auch nur über diese benutzbar. Dadurch können die einzelnen Bestandteile jederzeit durch Andere ersetzt werden, die dieselben Benutzungsschnittstellen anbieten.

4.1.1. Technische Voraussetzungen

In der ursprünglichen Idee war JustLingo als ein Programm zur Steuerung der Tabellenkalkulationssoftware Excel[®] 2013 der Microsoft[®] Corporation geplant. Excel[®] bietet über das .NET-Rahmenwerk die Möglichkeit an Erweiterungen, sogenannte *Add-Ins*, die direkt in die Tabellenkalkulationssoftware eingebaut sind, zu erstellen. Diese Erweiterungen können bei Bedarf vom Nutzer über das eingebaute Menüband hinzugefügt, beziehungsweise entfernt werden.

Das Programmieren eines solchen Add-In ist der Benutzung des .NET-Rahmenwerkes verbunden, daher wurde zur Entwicklung die Programmiersprache C# verwendet [Kü12]. Die gesamte Entwicklung wurde mit Hilfe der Programmierumgebung Visual Studio[®] von Microsoft[®] durchgeführt. Zur Versionsverwaltung wurde die bereits in Visual Studio[®] eingebaute Erweiterung Team Foundation Server verwendet.

4.1.2. Benutzeroberfläche

In Abbildung 4.1 ist ein Screenshot einer Tabellenkalkulation mit JustLingo zu sehen. Um das parallele Arbeiten mit einem geöffneten Dokument zu ermöglichen, kann das Dialogfenster jederzeit aus- und eingeblendet werden. Dabei wird der gesamte Dialogverlauf

ab dem Start des Programms angezeigt, damit der Benutzer alle Schritte nachvollziehen kann. Unterhalb des Dialogfensters befinden sich weitere Schaltflächen, über die der Nutzer zusätzliche Funktionen anschalten kann. Zum Beispiel den so genannten „Maus-Modus“ (die Stecknadel-Schaltfläche), der es ermöglicht Zellen oder Bereiche im aktuell geöffneten Dokument auszuwählen. Die ausgewählten Bereiche werden daraufhin automatisch in das Eingabefenster in Form einer für JustLingo verständlichen Formulierung übertragen. Rechts davon befindet sich eine Schaltfläche um zusätzliche Informationen zum Programmablauf zu erhalten. Bei Aktivierung werden die von den Modulen während der Ausführung erzeugten Ausgaben im Dialogfenster angezeigt. Dies kann auch nachträglich geschehen, da immer der gesamte Dialogablauf gespeichert wird. In der Mitte ist die Eingabefläche für den Nutzer zu sehen. Rechts von der Eingabefläche befindet sich eine Schaltfläche zur Aktivierung der Spracheingabe. Zuletzt gibt es noch eine Schaltfläche mit der der Nutzer seine Eingaben bestätigen kann. Statt diesen Knopf zu drücken, kann der Benutzer seine Eingaben aber auch mit der Eingabetaste bestätigen.

Da JustLingo den Nutzer bei der Arbeit mit Excel unterstützen soll, kann bei der Benutzung direkt auf die geöffnete Tabelle zugegriffen und diese bearbeitet werden. Dazu gehört Werte auszulesen sowie das Schreiben in vom Nutzer definierte Zellen. In der Abbildung 4.1 wurde beispielsweise eine neue Funktion „SINALPHA“ definiert, die den Sinus des Winkels einer Ecke eines rechtwinkligen Dreieckes ausrechnet. Im Anschluss wurde für die erste Zeile der Tabelle die Berechnung durchgeführt und eingetragen. Die Tabellenkalkulationssoftware hat im Anschluss die selbe Berechnung für jede weitere Zeile der Tabelle ausgeführt.

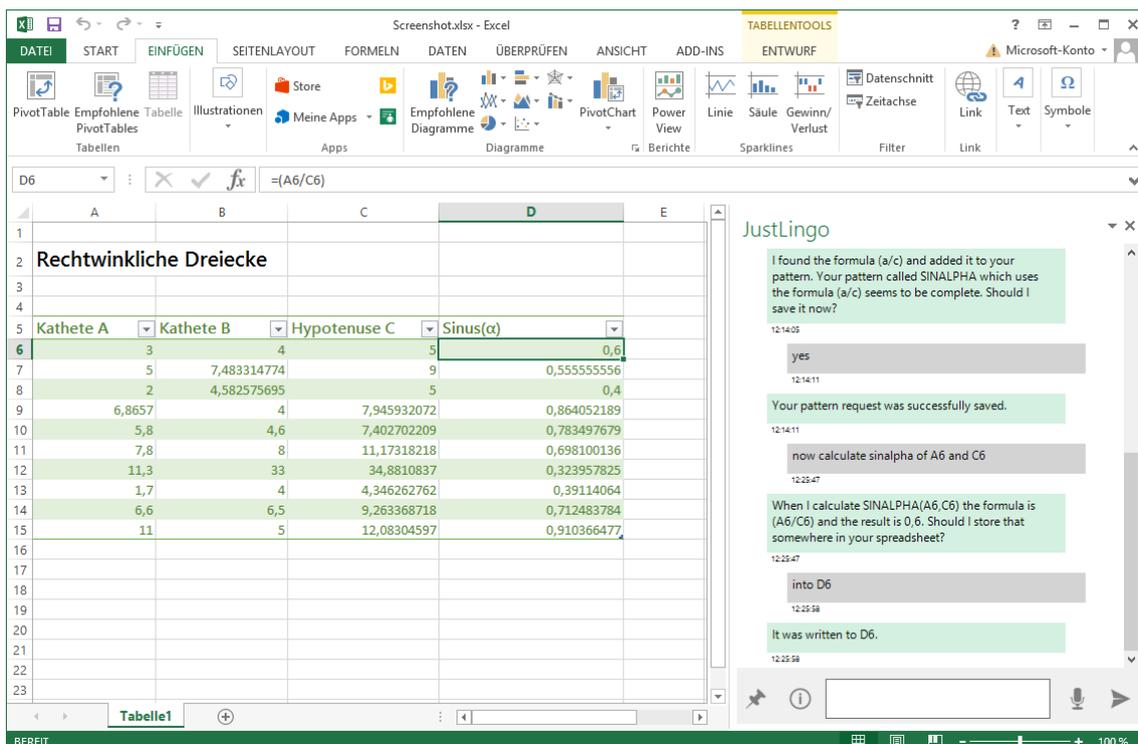


Abbildung 4.1.: Benutzeroberfläche des Dialogsystems

4.1.3. Model-View-Presenter Architektur

Bei einer Model-View-Presenter Architektur findet eine Aufteilung des Systems in drei Bausteine statt. Jeder dieser Bausteine verfügt über festgelegte Schnittstellen über die Daten ausgetauscht werden können. Da sich die einzelnen Bausteine nicht untereinander

kennen sind sie austauschbar, ohne das die gesamte Programmierung verändert werden muss. In JustLingo wurde dieses Architekturmuster wie in Abbildung 4.2 implementiert. Der detaillierte Aufbau der einzelnen Bestandteile wird nun im Folgenden beschrieben.

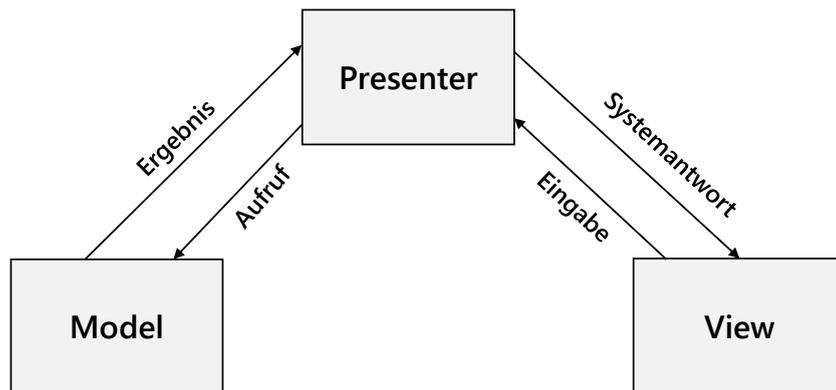


Abbildung 4.2.: Model-View-Presenter

View

Bei der ursprünglichen Idee des Model-View-Presenter Konzeptes, fand noch ein Datenaustausch zwischen View und Model statt, um die Aktualisierung des Views zu gewährleisten. Bei der Entwicklung von JustLingo wurde ein anderer Ansatz, der diese Verbindung nicht mehr erfordert gewählt: Der so genannte passive View [Fow06], dessen Logik vom Model in den Presenter verlegt wurde. Vorteile dieser Anordnung sind, dass View und Model komplett unabhängig voneinander entwickelt und verändert werden können. Das einzige Verbindungsglied besteht im Presenter, der gleichzeitig die beiden anderen Module koordiniert. Der View selbst übernimmt die Aufgabe dem Nutzer die vom System generierten Ausgaben anzuzeigen und die Eingaben vom Nutzer entgegen zu nehmen und direkt an den Presenter weiterzuleiten.

Presenter

Der Presenter übernimmt in JustLingo die Koordinierung und Steuerung des Programmablaufes. Er nimmt die Nutzereingaben vom View entgegen, übergibt sie an das Model, dass die Verarbeitung übernimmt. Von dort wird das Ergebnis zurück zum Presenter gesendet, der es zur Anzeige an den View übergibt. Der Presenter ist gleichzeitig der Einstiegspunkt des gesamten Programmes. Er erstellt beim Start von JustLingo alle nötigen Klassen. Der Presenter selbst ist dabei unabhängig von den beiden anderen Modulen, er kann diese lediglich ansteuern, hat aber keinen Einfluss auf deren interne Funktion.

Model

Im Model findet sich die gesamte Programmlogik und alle Datenbanken. Die Eingabe, die das Model vom Presenter erhält wird dann an die Verarbeitungspipeline übergeben und das Resultat wird zurück an den Presenter geschickt. An dieser Stelle wird auch offensichtlich, warum die Anordnung als Model-View-Presenter so praktisch ist. Denn alle Änderungen, die während dieser Arbeit an JustLingo vorgenommen wurden, sind innerhalb des Model zu finden. Weder am Presenter, noch am View wurden Veränderungen vorgenommen.

4.2. Mustererweiterung-Einheit

Die bereits im Konzept beschriebene Mustererweiterung-Einheit übernimmt die Aufgabe im Dialog mit dem Nutzer den Funktionsumfang von JustLingo zu erweitern. Dabei wurde der Erstellungsprozess in mehrere voneinander getrennten Modulen aufgeteilt (siehe Abbildung 4.3). Die Vorverarbeitung nimmt die zur Mustergenerierung nötigen Annotationen an der Nutzereingabe vor. Die von der Mustergenerierung vorverarbeitete Nutzereingabe wird nun an das Modul der sprachlichen Analyse übergeben. Dieses Modul gehört zum ursprünglichen Modulaufbau und übernimmt die mathematische Interpretation der Nutzereingabe. Anschließend wird das Ergebnis an die Kontextinterpretation weitergeleitet, die den Zusammenhang zwischen Nutzereingaben und dem Gesprächsverlauf herstellt. Zuletzt wird die Musterinterpretation durchgeführt, die aus den Nutzereingaben die zur Mustergenerierung nötigen Informationen extrahiert. Der genaue Aufbau der einzelnen Module und deren internen Abläufe werden nun im Folgenden detailliert beschrieben.

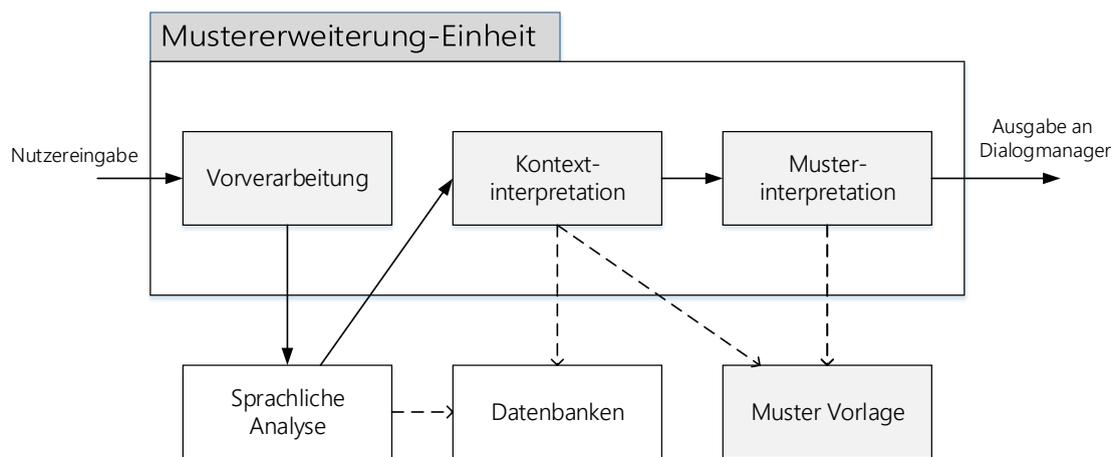


Abbildung 4.3.: Verarbeitungsablauf der Mustererweiterung-Einheit

4.2.1. Vorverarbeitung

Die Vorverarbeitung bildet den ersten Teilschritt der neuen Erweiterung von JustLingo. Innerhalb dieses Moduls werden zwei wichtige Aufgaben durchgeführt. Zuerst findet eine Entfernung von Referenzen innerhalb der Nutzereingabe statt und im Anschluss wird eine Markierung von Variablen vorgenommen.

Entfernung von Referenzen

Wie bereits in den Grundlagen beschrieben reagiert JustLingo auf Rückbezüge und Referenzen in den Nutzereingaben. Bei Benutzung der Mustergenerierung können die von JustLingo gefundenen Rückbezüge aber in manchen Situationen falsch aufgelöst werden. In dem Beispiel in Abbildung 4.4 bezieht sich der Nutzer offensichtlich bei seiner Beschreibung nicht auf eine vorher eingegebene Funktion, sondern auf sein neues Muster selbst. An dieser Stelle würde die Referenz dennoch aufgelöst werden was entweder dazu führt, dass eine frühere Nutzereingabe für *it* eingesetzt wird, oder bisher noch keine Eingaben existieren und JustLingo den Nutzer sinngemäß auf diesen Fehler hinweist. Da der Nutzer aber mit seiner Aussage eigentlich bezwecken wollte, dass die Formel „a + b“ zu seinem neu erstellten Muster hinzugefügt werden soll, muss die Referenz entfernt werden. Diese Referenzenentfernung wird nur solange vorgenommen, bis der Nutzer zum ersten mal seinem Muster eine Formel hinzugefügt hat. Ab diesem Moment werden die Referenzierungen

in den Eingaben nämlich dazu benutzt Schritt für Schritt die Formeln an die Mustergenerierung zu übergeben. Zusätzlich wird im späteren Programmablauf in den Prozess der Kontextabspeicherung eingegriffen (siehe 4.3.4), um ständig garantieren zu können, dass Referenzen immer korrekt aufgelöst werden.

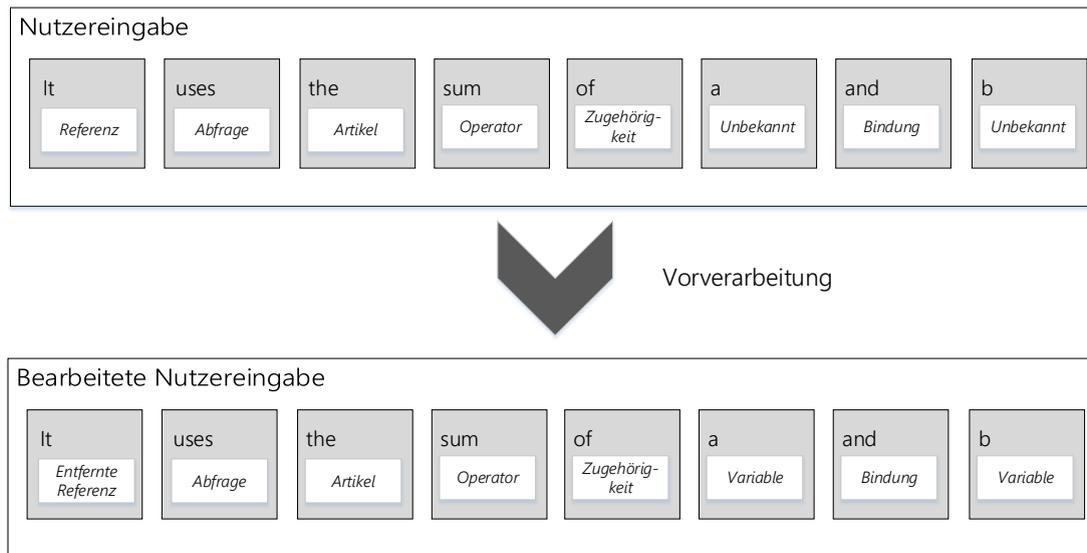


Abbildung 4.4.: Annotation der Eingabe: „It uses the sum of a and b.“

Variablen Annotation

Beim Arbeiten mit JustLingo ist es möglich direkt Bereiche, Spalten, Reihen und einzelne Zellen im aktuell geöffneten Dokument der Tabellenkalkulationssoftware anzusprechen. Um aber dem System JustLingo neue Funktionen beizubringen sind diese statischen Bereiche nicht ausreichend. Deshalb wurde die Verarbeitung um eine neue Art von Operand, die Variable, erweitert. Diese ermöglicht es, allgemeine Formeln an JustLingo zu übermitteln, sodass erst bei der späteren Benutzung entschieden werden muss, was statt der Variable eingesetzt werden soll. Das definieren statischer Operanden, zum Beispiel Zahlen und Zellen, die bei jedem Funktionsaufruf immer gleich bleiben, ist aber dennoch möglich.

Um die Variablen in der Eingabe zu finden, werden alle unmarkierten Wörter in Betracht gezogen. Diese werden dann zuerst auf explizite Variablen-Benennungen untersucht. Im Satz „*The formula is variable a minus b*“ wird zum Beispiel die Variable „a“ durch das Schlüsselwort „*variable*“ explizit ausgewiesen. Nachdem alle expliziten Benennungen vorgenommen wurden, wird nach weiteren möglichen Variablennamen gesucht. Da das System alle unbekannt Wörter als potentielle Variablen ansieht, wird im Folgenden ein Indikator berechnet, der angibt wie wahrscheinlich es ist, dass ein bestimmtes Wort eine Variable ist. Dazu werden den einzelnen Wörtern Gewichte zugewiesen. Je höher das Gewicht eines Wortes am Ende ist, desto wahrscheinlicher ist es, dass es sich dabei um eine Variable handelt. Bei der Berechnung werden zwei Umstände angenommen: Kürzere Wörter bekommen ein höheres Grundgewicht, als lange Wörter. Vermehrtes auftreten eines Wortes erhöht dessen Gewichtung. Um zu verdeutlichen wie die Markierung abläuft, wird dies anhand des Satzes: „*sum up var1, v2 and c and divide it by var1 and v2*“ erklärt. In diesem Satz sind die Wörter *up*, *var1*, *v2* und *c* unbekannt und werden deshalb bei der Gewichtung betrachtet. Vor der Ausführung wurde eine maximale Länge für den Namen fest implementiert, in diesem Beispiel sei dies $n = 5$. Die maximale Länge der gesuchten Wörter wird dabei beim Programmieren festgelegt und ist nur so wieder veränderbar.

Vorgehen:

1. Suche alle passenden Wörter, die als *unbekannt* Markiert sind.
 - im Beispiel: {var1; v2; c; up}
2. Zähle das Vorkommen der Wörter:
 - {(var1,2);(v2,2);(c,1); (up,1)}
3. Berechne die Gewichte der gewählten Wörter:
 - $Gewicht = n - Laenge(Wort) + (Anzahl(Wort) - 1) * n$

Daraus ergibt sich für den gewählten Satz das in Tabelle 4.1 festgehaltene Ergebnis. Durch die nun berechneten Gewichtungen können absteigend die Wörter als Variablen markiert werden. Dieses Vorgehen erzielt besonders gute Ergebnisse, wenn dem System vorher die Anzahl an benutzen Parametern bekannt ist. Da diese zusätzliche Information, bei der Erstellung eines Musters nicht immer von Anfang an definiert sein muss und aus Sicht des Nutzers nicht intuitiv ist, wurde die Variablenmarkierung auf den Spezialfall reduziert, bei dem nur Benennungen der Länge eins, also die Kleinbuchstaben von a bis z, erkannt werden. Dadurch kann das Ermitteln der Anzahl der Parametern automatisch bei jeder Änderung in der Formel des Musters neu berechnet werden und so diese Aufgabe dem Nutzer abgenommen werden. Der bereits fertige Algorithmus wird aber dennoch im Programm belassen, falls in Zukunft der anfängliche Ansatz wieder benutzt werden soll.

Name	Vorkommen	Gewichtung
var1	2	5
v2	2	7
c	1	4
up	1	3

Tabelle 4.1.: Markierungen bei „Sum up var1, v2 and c and divide it by var1 and v2.“

Ein Spezialfall, der bei der Variablenmarkierung vorkommt, ist in Abbildung 4.5 dargestellt. In diesem Satz hat der Nutzer den englischen Artikel „a“ in seiner Eingabe benutzt. Deshalb wird an dieser Stelle keine Variablenmarkierung durch das System vorgenommen. Um herauszufinden, wann es sich um einen Artikel und wann es sich um einen variablen Namen handelt, werden zwei Kontrollmechanismen benutzt. Zuerst wird eine Überprüfung durchgeführt, ob auf den Artikel eine Kombination von Adjektiven und ein Nomen folgt. Zusätzlich wird überprüft, ob vor oder nach dem Artikel ein Wort, dass vorher als Operator identifiziert wurde, steht. Falls auf das untersuchte Wort Adjektive und ein Nomen folgen und es nicht Teil eines Operators sein kann, wird keine Markierung als Variable vorgenommen.

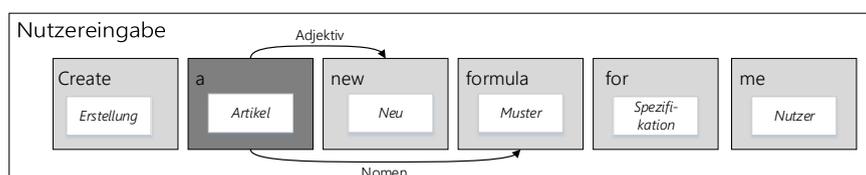


Abbildung 4.5.: Behandlung von Artikeln in der Vorverarbeitung

4.2.2. Kontextinterpretation

Nachdem die sprachliche Analyse das Identifizieren von allen mathematischen Formeln in der Nutzereingabe übernommen hat, wird die Kontextinterpretation ausgeführt. Die Kontextinterpretation stammt ursprünglich aus der Arbeit von Voigt [Voi14], wurde aber so angepasst, dass sie auch mit der Mustergenerierung umgehen kann. Bei den vorgenommenen Änderungen wurde immer darauf geachtet, dass der ursprüngliche Aufbau erhalten bleibt und dieselben Ergebnisse wie zuvor erzielt werden. Die Aufgabe der Kontextinterpretation besteht darin, die Verbindung zwischen den Fragen des Systems und den Antworten des Nutzers herzustellen. Dabei wird wie in Abbildung 4.6 vorgegangen. Zunächst wird im Kontext nachgeschaut, ob es sich bei der letzten erzeugten Systemausgabe um eine Frage gehandelt hat. Sollte dies nicht der Fall sein, wird die Kontextinterpretation direkt verlassen und die Verarbeitung mit dem nächsten Modul fortgesetzt. Wurde eine Frage gefunden, wird anschließend überprüft was für einen Antworttyp erwartet wird. Im Ablauf der Mustergenerierung werden folgende Antworttypen erwartet:

- **Formel** - Hier wird vom Benutzer eine mathematisch interpretierbare Eingabe erwartet.
- **Name** - Das System erwartet, dass der Nutzer entweder ein Wort zur Benennung seines Musters, oder ein Benennungsschlüsselwort zusammen mit einem geeigneten Namen eingibt.
- **Ja/Nein** - Wurde eine Ja/Nein-Frage gestellt, wird an dieser Stelle die entsprechende Antwort erwartet.
- **Zahl** - Das System erwartet vom Nutzer eine Zahl als Eingabe. Dies wird zum Beispiel bei einer Auswahlfrage (siehe 4.2.2) erwartet.
- **Schlüsselwörter** - Im Rahmen der Mustergenerierung wurde eine Auswahlfrage bezüglich des Änderns des Musters hinzugefügt (siehe 4.2.2). Diese Frage erwartet als Antwort eines der Schlüsselwörter *Formel*, *Name*, *beides*, *nichts* oder *keines von beidem*.

Ist nun ein passender Antworttyp zu einer Frage gefunden, wird die von der Frage induzierte Aufgabe entsprechend der Nutzerantwort durchgeführt.

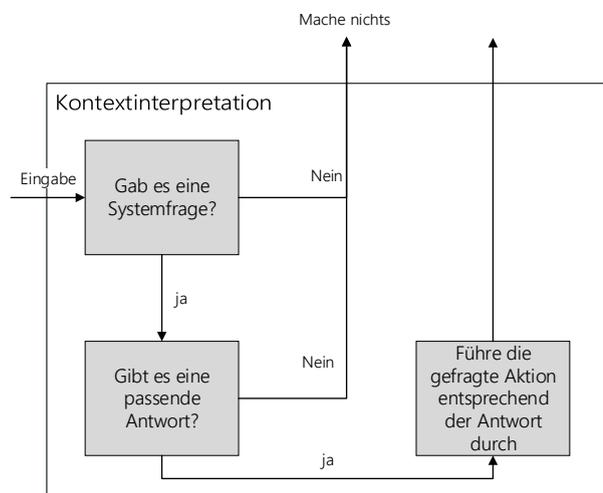


Abbildung 4.6.: Ablauf der Kontextinterpretation

Beantwortung einer Frage zum Speichern

Diese Frage wird automatisch an den Nutzer gerichtet, sobald ein komplett definiertes Muster vorliegt. Die Frage erwartet als Antwort des Nutzers entweder ein klares *Ja*, *Nein* oder keine Reaktion. Keine Reaktion seitens des Nutzers ist zulässig, da die Mustergenerierung nicht entscheiden kann, ob der Erstellungsprozess bereits abgeschlossen ist. Ein Muster ist erst komplett, sobald es über einen identifizierten Namen und eine korrekt ausführbare Formel verfügt. Beantwortet der Nutzer diese Frage mit einem *Ja* wird automatisch versucht das Muster der Datenbank hinzuzufügen. Daraufhin wird eine Systemantwort generiert, die dem Nutzer mitteilt, ob das Hinzufügen erfolgreich war. Bei einem *Nein* als Antwort reagiert das System mit einer Änderungsfrage.

Beantworten einer Formelfrage

Stellt das System eine Formelfrage, wird vom Nutzer eine *Formel* als Antwort erwartet. Diese Frage löst im System keine bestimmte Reaktion aus, muss aber abgefangen werden, damit die iterative Ausgabe von Hilfetexten in der Dialogmanagement-Einheit funktioniert (siehe 4.3.3). Das Hinzufügen der Formel übernimmt das Modul der Musterinterpretation.

Beantworten einer Namensfrage

An dieser Stelle wird als Antwort vom Benutzer ein *Name* erwartet. Auch hier erfolgt, wie bei der Formelfrage, innerhalb der Kontextinterpretation keine bestimmte Reaktion des Systems. Es findet lediglich eine Überprüfung statt, ob ein passender Name gefunden wurde oder nicht. Abhängig davon werden Hilfetexte in der Dialogmanagement-Einheit (siehe 4.3.3) erzeugt. Das Hinzufügen des Namens zur Mustervorlage übernimmt anschließend das Modul der Musterinterpretation.

Beantworten einer Änderungsfrage

Über diese Systemfrage kann ein Nutzer die einzelnen Bestandteile seines Musters zurücksetzen lassen und von neuem eingeben. Dabei gibt es vier Reaktionsmöglichkeiten der Kontextinterpretation auf die Antwort des Nutzers. Wurde das Schlüsselwort *Formel* gefunden wird automatisch die Formel und alle dazugehörigen Parameter, zum Beispiel der Zähler für die Anzahl an Eingabeparametern, zum Ausgangszustand zurückgesetzt. Der Nutzer kann vom System über ein *Namen*-Schlüsselwort die Benennung seines Musters zurücksetzen lassen. Es wird auch erkannt, wenn der Nutzer beides gleichzeitig, oder keins von beidem ändern möchte. In jedem Fall wird automatisch eine passende Antwort an den Nutzer zurückgegeben.

Beantworten einer Auswahlfrage

Eine Auswahlfrage wird generiert, wenn bei der mathematischen Interpretation Mehrdeutigkeiten entstanden sind. Diese Art der Frage war bereits in der Arbeit von Voigt vorhanden, wurde aber angepasst. In der ursprünglichen Version beschränkte sich die Auswahl auf zwei Formeln, selbst wenn noch mehr Möglichkeiten gefunden wurden. Um die Arbeit mit der Mustergenerierung besser zu gestalten wurde die maximale Anzahl an Formeln auf fünf erhöht. Darüber hinaus wurde die Textausgabe überarbeitet, um das Erkennen der einzelnen Formeln und deren Auswahl übersichtlicher zu machen (siehe Tabelle 4.3). Der Nutzer trifft mittels einer *Zahl* eine Auswahl und das System wird dann mit der gewählten Formel weiter arbeiten.

4.2.3. Musterinterpretation

Das Modul der Musterinterpretation ist das Kernstück der Erweiterung zur Mustergenerierung. In diesem Modul werden die Intentionen des Nutzers gesammelt, um schrittweise ein neues Muster zu erstellen. Der grobe Ablauf der Musterinterpretation ist in Abbildung 4.7 zu sehen. Während der Abarbeitung werden zusätzlich Aussagen und Fragen für die Dialogmanagement-Einheit erzeugt. Hierbei werden immer alle möglichen Aussagen und Fragen gesammelt und an den Dialogmanager weitergegeben, der dann selbständig entscheidet, welche für den Nutzer im Moment am wichtigsten sind. Der aktuelle Fortschritt im Erstellungsprozess wird in einer so genannten Mustervorlage festgehalten. In dieser Vorlage werden alle aus den Nutzereingaben ermittelten Informationen gesammelt. Da immer dieselben Informationen benötigt werden, kann aus der Vorlage automatisch abgelesen werden, was zur Vollständigkeit des Musters noch fehlt. Die beschriebene Vorlage kann nur innerhalb der Musterinterpretation verändert werden. In Tabelle 4.2 wurde der Eintrag der dritten binomischen Formel erstellt.

Jedes neue Muster benötigt einen eindeutigen Namen, damit es von JustLingo später benutzt werden kann. Dieser Name wird in der Mustervorlage im Namensfeld abgespeichert. Zusätzlich wird die von JustLingo aus den Nutzereingaben generierte Formel, die für die Tabellenkalkulationssoftware verständlich ist, abgespeichert. Die Formel wird dabei in zwei verschiedenen Schreibweisen gespeichert. Die erste Schreibweise enthält dabei exakt die vom Nutzer gewählte Variablen-Benennung und wird im „Formel (Eingabe)“-Feld eingetragen. Diese wird immer dann als Repräsentation der Formel verwendet, wenn sie an den Nutzer ausgegeben werden soll. Die andere Schreibweise der Formel entspricht der in den Datenbanken verwendeten Darstellung. In dieser Rohform kann die Ausgabe dem Nutzer aber nicht angezeigt werden, da sie eine unübersichtliche Darstellung der Parameter besitzt. Sie ist im „Formel(Datenbank)“-Feld in Tabelle 4.2 beispielhaft dargestellt. Zusätzlich zu Formel und Name wird die Anzahl an benutzen Variablen gespeichert. Diese Anzahl wird bei jeder Änderung, die der Nutzer vornimmt, vom System berechnet. Diese Information ist nötig, damit bei Benutzung des Musters die Parameter korrekt zugeordnet werden können. Die Musterinterpretation wurde passend zu den benötigten Informationen in zwei Module aufgeteilt: die Namensinterpretation und die Formelinterpretation (siehe Abbildung 4.7). Es werden bei jeder Nutzereingabe immer beide Module durchlaufen um alle Änderungen jederzeit zu erfassen.

Eigenschaft	Inhalt
Name	Binom3
Formel (Eingabe)	Produkt(Summe(a;b);Summe(a;Negativ(b)))
Formel (Datenbank)	Produkt(Summe(%1;%2);Summe(%1;Negativ(%2)))
Variablen Anzahl	2

Tabelle 4.2.: Mustervorlage der dritten binomischen Formel

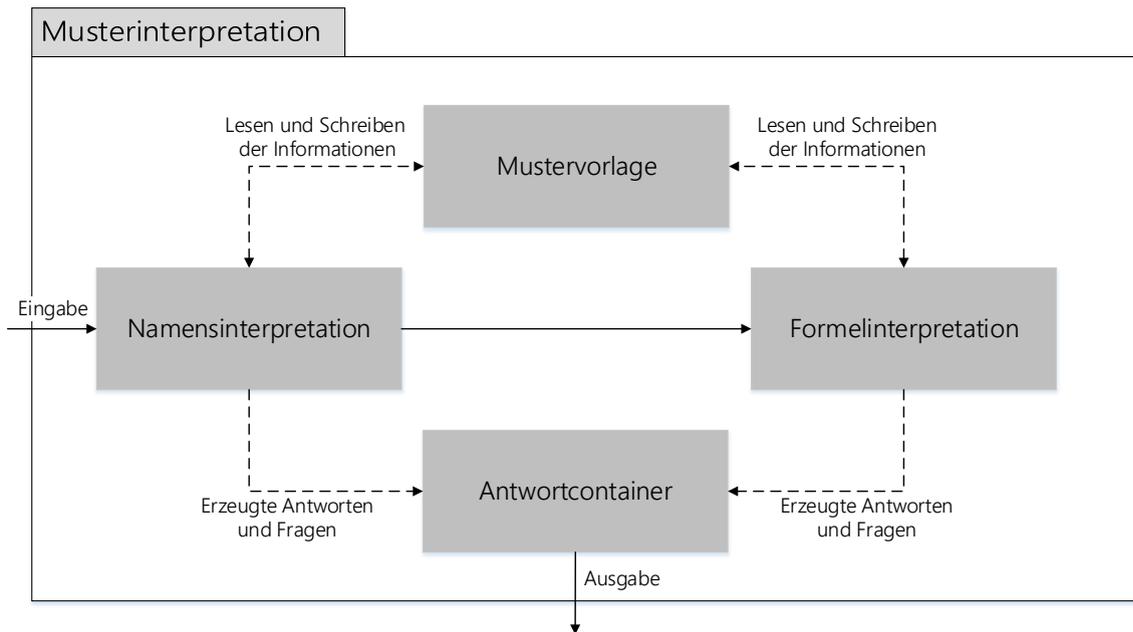


Abbildung 4.7.: Aufbau der Musterinterpretation

Namensinterpretation

Die Namensinterpretation ist dafür zuständig jegliche Äußerungen des Nutzers, die sich auf das Hinzufügen und Ändern des Namens der neuen Formel beziehen, zu erkennen und umzusetzen. Das Vorgehen erfolgt dabei immer nach dem in Abbildung 4.8 aufgezeichneten Ablauf. Im Folgenden wird der Ablauf der Namensinterpretation beschrieben, detaillierte Informationen zur Answererzeugung finden sich in Kapitel 4.3.1.

Zuerst wird die gesamte Eingabe des Benutzers nach Schlüsselwörtern, die auf eine Namensbenennung hinweisen durchsucht. Dazu gehören zum Beispiel: *call*, *define* und *name*. Wurde ein solches Schlüsselwort nicht gefunden wird überprüft, ob die Mustervorlage für das Namensfeld bereits einen gültigen Eintrag besitzt. Falls dies nicht der Fall ist, wird eine Frage nach dem Namen des Musters erstellt und dem Antwortcontainer (siehe Kapitel 4.3.1) hinzugefügt, anderenfalls wird die Namensinterpretation verlassen und mit der Formelinterpretation fortgefahren.

Wurde ein Benennungsschlüsselwort gefunden, wird die Nutzereingabe nach einem passenden Namen durchsucht. Hierbei werden nur Wörter betrachtet, die dem Programm unbekannt sind. Dazu zählen alle Wörter, zu denen kein Eintrag in der Wortdatenbank besteht. Falls dies scheitert, also kein passender Namen gefunden wurde, wird der Benutzer darüber informiert, dass er dem System zwar einen Namen mitteilen wollte, aber dieser entweder nicht verstanden wurde oder bereits belegt ist. Bei erfolgreicher Namenssuche wird noch verglichen ob es sich um eine Umbenennung, oder um eine Erstbenennung des Musters handelt. Zu beiden Fällen werden spezifische Antworten für den Benutzer erstellt, die ihm genau mitteilen, welche Änderungen vorgenommen wurden. Diese werden in den Antwortcontainer abgelegt und später an den Dialogmanager übergeben.

Alle gefundenen Namensänderungen werden sofort in der Mustervorlage gespeichert. Da Neu- und Umbenennungen zu jeder Zeit während des Erstellungsprozesses möglich sind, wurde auf ein explizites Nachfragen seitens des Systems, speziell bei Umbenennungen, verzichtet. Sollte der Nutzer sich aus Versehen vertippt haben, oder das System die Äußerung missverstanden haben, kann er dies bei der nächsten Eingabe umgehend korrigiert werden.

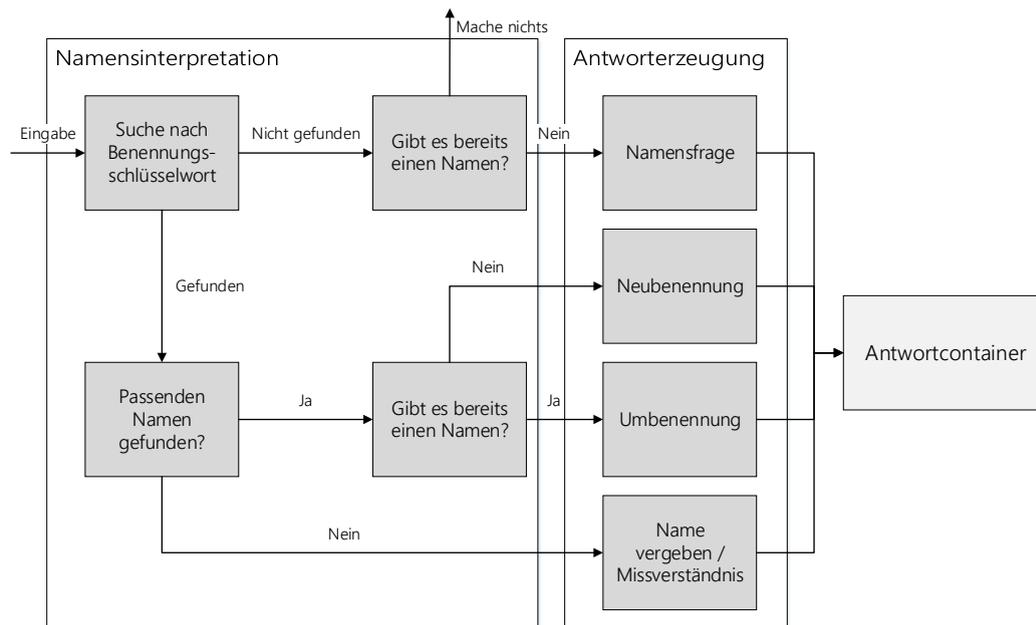


Abbildung 4.8.: Übersicht der Namensinterpretation

Formelinterpretation

In der Formelinterpretation wird die gesamte Eingabe des Nutzers nach Formeln durchsucht. Dazu wird das Ergebnis des vorher ausgeführten Moduls, der Sprachanalyse, benutzt. Die Sprachanalyse sucht in der Eingabe nach allen Interpretationsmöglichkeiten. Dazu gehört nicht nur das Zuordnen von Operanden zu Operationen, sondern auch das Auflösen von Mehrdeutigkeiten und das Erkennen von Fehlern in der Nutzereingabe. In der Formelinterpretation (siehe Abbildung 4.9) wird zuerst anhand der von der Sprachanalyse vorgenommenen Zuordnungen überprüft, ob die Nutzereingabe überhaupt mathematisch interpretierbar ist. Sollte dies nicht der Fall sein, wird in der Mustervorlage überprüft, ob früher bereits eine Formel eingetragen wurde. Wurde keine Formel gefunden wird mittels der Antworterzeugung eine Formelfrage für den Nutzer erstellt und im Antwortcontainer abgelegt. Sollte bereits eine Formel vorhanden sein, wird keine Frage erstellt und die Formelinterpretation direkt verlassen.

Ist die Nutzereingabe mathematisch interpretierbar wird als nächstes untersucht, ob mindestens eine der gefundenen Möglichkeiten zu einem fehlerfreien Ergebnis führt. Gibt es nur eine Möglichkeit, wird diese in der Mustervorlage gespeichert und eine Antwort erzeugt, die dem Nutzer mitteilt, dass eine bestimmte Formel zu seinem Muster hinzugefügt wurde. Gibt es jedoch mehr als eine korrekte Interpretation, werden dem Nutzer alle zusammengesetzten Formeln ausgegeben. Anschließend wird ihm eine Entscheidungsfrage gestellt. Hierbei werden aus Gründen der Lesbarkeit alle Formeln zeilenweise aufgelistet und mit einem Index versehen, mittels dessen der Nutzer seine Auswahl treffen kann. Es werden darüber hinaus höchstens die fünf wahrscheinlichsten Möglichkeiten ausgegeben. Ein beispielhafter Dialogverlauf wird in Tabelle 4.3 gezeigt. Der Nutzer kann daraufhin eine der beiden Formeln direkt über die Zahlen auswählen. Sollte keine der vorgeschlagenen Möglichkeiten die richtige sein, kann der Nutzer die Frage ignorieren und seine Anfrage neu formulieren. Sollten jedoch nur unvollständig oder fehlerbehaftet Formeln gefunden

werden, wird eine dieser Formeln zusammen mit dem gefundenen Fehler ausgegeben. Der Benutzer erhält daraufhin die Möglichkeit seine Eingabe zu verändern, sei es über den Austausch einzelner Operanden oder komplettes Umformulieren.

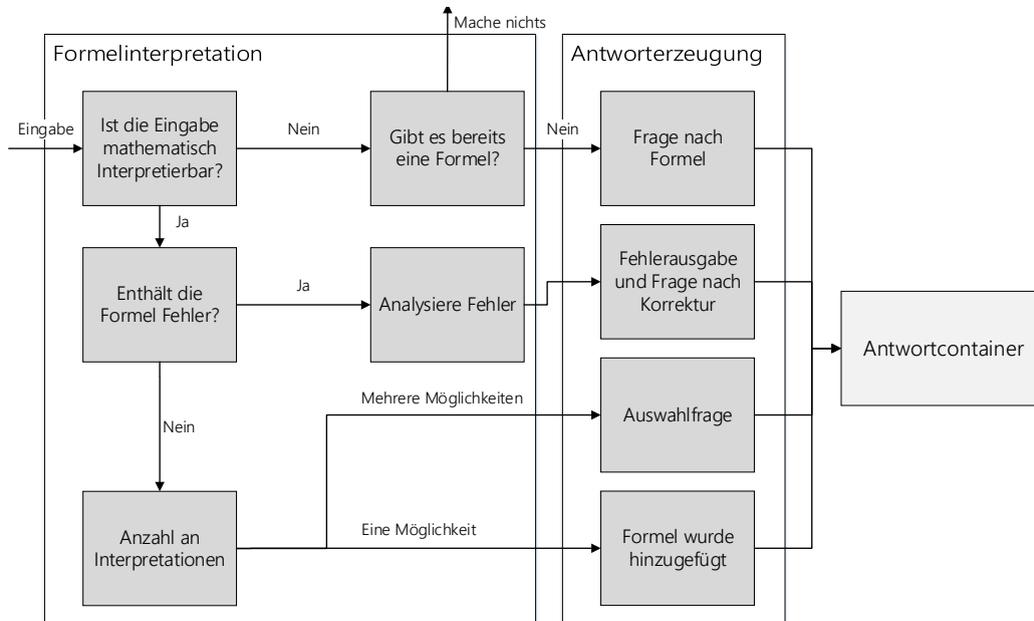


Abbildung 4.9.: Übersicht der Formelinterpretation

Nutzereingabe

The formula is a minus b divided by c minus d.

Systemausgabe

I found 2 possible formulas:

1. $((a - b) / (c - d))$

2. $((a - (b / c) - d))$

Which one should I use?

Tabelle 4.3.: Beispiel einer nicht eindeutig interpretierbaren Formel

4.3. Dialogmanagement

Die Dialogmanagement-Einheit steuert die Dialogabläufe und Dialogentscheidungen, die während der Benutzung der Mustergenerierung getroffen werden. Die Entscheidungen sind dabei von zwei Faktoren abhängig:

1. Die von der Musterinterpretation erstellten Antworten, die als Antwortcontainer übergeben werden.
2. Der interne Zustand in dem sich die Dialogmanagement-Einheit befindet.

Der allgemeine Ablauf ist in Abbildung 4.10 skizziert. Als erstes wird anhand der Zustandswahl entschieden, mit welchem der Dialogmanagement-Modulen fortgefahren werden soll. Wenn das generelle Dialogmanagement gewählt wird findet eine Antwortauswahl unabhängig von vorherigen Aussagen statt (siehe Kapitel 4.3.2). In den speziellen Modulen wiederum finden Kontext abhängige Antwortentscheidungen statt (siehe Kapitel 4.3.3). Um in den speziellen Management-Modulen auch auf Kontext unabhängige Äußerungen des Nutzers reagieren zu können wird im Anschluss immer ein Rückfall-Management durchgeführt (siehe Kapitel 4.3.3). Falls alle Interpretationen scheitern sollten, wird dem Nutzer am Ende eine „Ich habe die Eingabe nicht verstanden“ Antwort ausgegeben. Allerdings ist dieser Fall sehr unwahrscheinlich, da die Systeminitiative stark erhöht wurde und fast bei jedem Durchlauf eine Rückfrage an den Nutzer ausgegeben wird.

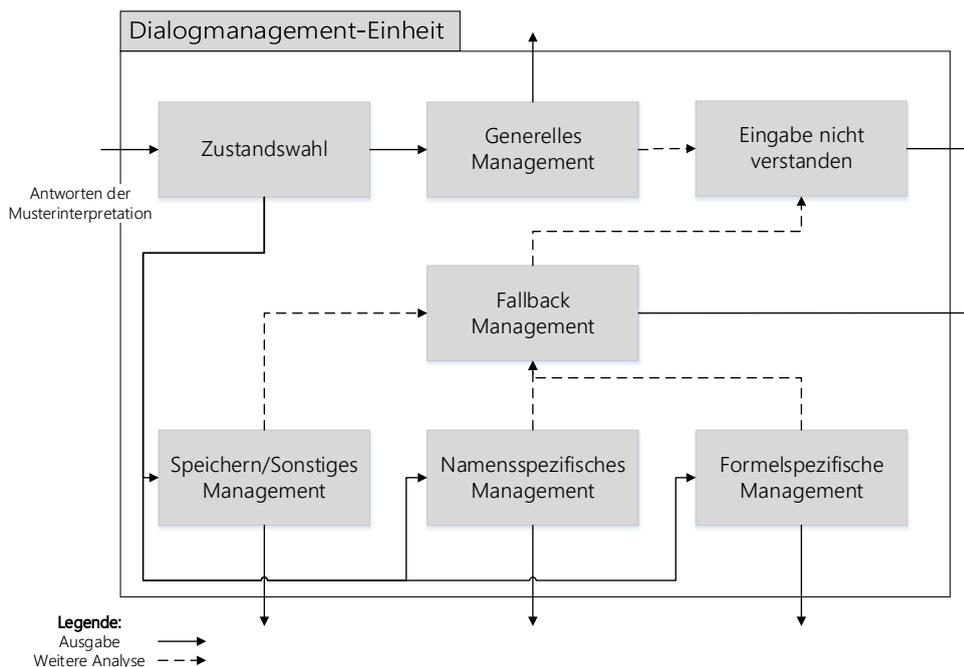


Abbildung 4.10.: Übersicht der Dialogmanagement-Einheit

4.3.1. Answererzeugung

Der Prototyp von JustLingo verfügte bereits über eine Answererzeugung. Diese war jedoch statisch umgesetzt und konnte deswegen nicht direkt an die Mustergenerierung angepasst werden. Aus diesem Grund wurde die Answererzeugung in dieser Arbeit komplett überarbeitet. Dabei wurde darauf geachtet, dass die Answererzeugung von der Answerausgabe entkoppelt ist. Durch diese Änderung wurde die Erstellungslogik von der Auswahl und

Ausgabe getrennt. Das Klassendiagramm der neuen Antwortlogik ist auszugsweise in Abbildung 4.11 zu sehen. Bei der Implementierung wurde dabei nach dem Entwurfsmuster der abstrakten Fabrik vorgegangen. Dadurch kann für jede Erweiterung, auch für zukünftige, unabhängig die Antworterzeugung umgesetzt werden. Als Schablone für die konkreten Implementierungen dient die so genannte *AbstractAnswerGeneration*. Sie enthält den Verweis auf die Antwortdatenbank und gibt allen Klassen die Methode „buildAnswerSegments“ vor. Innerhalb dieser Methode wird die Erzeugung der Antwortsegmente umgesetzt. Die Daten, die die einzelnen Klassen dazu brauchen können über den jeweils unabhängigen Konstruktor übergeben werden. In Abbildung 4.11 wurden als Beispiel die drei Implementierungen *GenericAnswers*, *PatternNameAnswers* und *PatternSingleFormulaAnswers* eingezeichnet. Bei den *GenericAnswers* handelt es sich um Antworten, die das System auf bestimmte Schlüsselwörter erwidert. Dazu gehören zum Beispiel:

- *Help*: Das System wird dem Nutzer darauf einen kurzen Hinweistext ausgeben, in dem steht, wozu JustLingo benutzt werden kann.
- *Hello*: Es wird dem Nutzer eine Willkommensnachricht ausgegeben.
- *Thank you*: Das System reagiert darauf, wenn sich der Nutzer bedankt.

Es handelt sich also hauptsächlich um eine Ansammlung an Floskeln um ein Gespräch einzuleiten. In der *PatternNameAnswers*-Klasse handelt es sich hingegen um die speziellen Reaktionen des Systems, die bei der Namensfindung eines Musters zustande kommen. Dazu gehören das erfolgreiche Hinzufügen eines Namens, das Ändern eines Namens und Hilfetexte bei Misserfolg (vergleiche 4.3.3).

Ebenso handelt es sich bei der *PatternSingleFormulaAnswers*-Klasse um eine speziell auf die Mustererweiterung zugeschnittene Umsetzung. Diese Klasse erstellt alle Systemausgaben, wenn erfolgreich eine Formel zum Muster hinzugefügt wurde. In der Abbildung handelt es sich dabei nur um einen kleinen Auszug. Innerhalb dieser Arbeit wurden insgesamt 17 verschiedene Antworterzeugungen, die für die jeweiligen Bereiche angepasst wurden, erstellt.

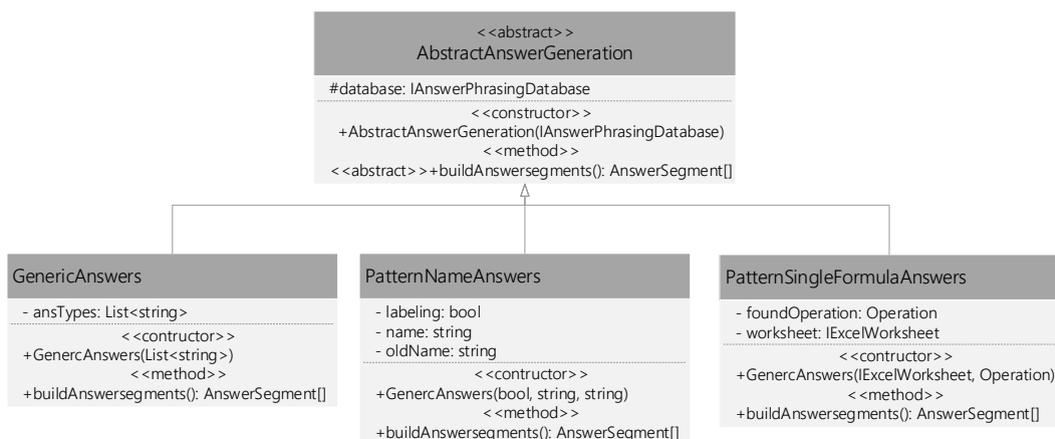


Abbildung 4.11.: Auszug des Antworterzeugungs-Klassendiagrammes

Antwortsegment

Alle Antworten und Fragen, die JustLingo erzeugt, werden in Form von Antwortsegmenten abgespeichert. Jedes dieser Segmente besitzt einen eindeutigen Eintrag in der Datenbank (vergleiche Tabelle 4.4) und wird über seinen Namen identifiziert. Jedes Antwortsegment besitzt darüber hinaus noch ein *Typen*-Feld. Dort wird vermerkt, ob es sich bei dieser Antwort um eine Frage, oder um einen Namen handelt. Bei Fragen gibt es zusätzlich noch ein

AsksFor-Feld. Hier steht zum einen, welche Reaktion vom Nutzer auf diese Frage erwartet wird und zum anderen welche Aktion bei einer passenden Antwort ausgeführt werden soll. Im *Phrasing*-Feld steht an welcher Stelle in der Datenbank die möglichen Formulierungen (vergleiche Tabelle 4.5) zu diesem Antwortsegment zu finden sind.

In Tabelle 4.5 ist ein Auszug aus der Datenbank für mögliche Formulierungen einer Namensfrage abgebildet. Bei einem Datenbankenaufruf wird dabei zufällig eine der Möglichkeiten ausgewählt, die der Nutzer dann erhält. Durch diese zufällige Auswahl soll erreicht werden, dass nicht immer dieselben Phrasen wiederholt werden und so die Interaktion mit dem System menschlicher wirkt.

Eigenschaft	Inhalt
Name	MissingNameQuestion
Type	Question
AsksFor	Type: Name Action: AddName
Phrasing	*PatternNameQuestion

Tabelle 4.4.: Inhalt der Namensfrage

Index	PatternNameQuestion
1	Can you tell me the name of your pattern
2	Would you mind telling me the name
3	May I aks you for the name
4	You need to tell me the name

Tabelle 4.5.: Phrasen der Namensfrage

Antwortcontainer

Der bereits in der Musterinterpretation benutzte Antwortcontainer ist eine Datenstruktur, die gewählt wurde, um alle in einem Durchgang erzeugten Antwortsegmente zwischenspeichern. Der in der Musterinterpretation erstellte Antwortcontainer stellt die eine Hälfte der im Dialogmanager berücksichtigten Antworten dar. Als zweite Eingabe wird immer der aktuelle Kontext in Betracht gezogen.

4.3.2. Generelles Dialogmanagement

Das generelle Dialogmanagement stellt den Ausgangspunkt der gesamten Antwortauswahl innerhalb dieser Arbeit dar. Als Eingabeparameter wird der von der Musterinterpretation erzeugte Antwortcontainer übergeben. Aus diesem Antwortcontainer werden zuerst alle Antwortsegmente, die Informationen zu in der Verarbeitung vorgenommenen Änderungen enthalten, in den Ausgabecontainer überschrieben. Sollten es zwei oder mehr Aussagen sein wird an dieser Stelle aufgehört, um den Benutzer nicht mit zu viel Text abzuschrecken. Ansonsten wird absteigend, wie in Abbildung 4.12 aufgezeichnet, nach möglichen Systemfragen, die dem Nutzer gestellt werden können, gesucht. Dabei wird mit der Frage nach dem Namen des Musters angefangen. Im Anschluss wird versucht die zugehörige Formel vom Benutzer zu erfahren. Sobald beide Informationen gefunden wurden, fragt das System den Nutzer automatisch, ob der sein Muster nun abspeichern möchte. Je nach gewählter Frage findet ein Zustandswechsel zu den beschriebenen spezifischen Management-Modulen statt. Unabhängig von den im Diagramm eingezeichneten Systemfragen, gibt es noch zusätzliche Fragen mit erhöhter Priorität. Diese werden bevorzugt gewählt und wurden aus Gründen der Übersichtlichkeit und weil sie nicht unmittelbar zum Ablauf der Dialogentscheidungen beitragen, nicht in die Abbildung aufgenommen. Dazu gehören zum Beispiel die bereits beschriebenen Auswahlfragen, wenn es bei der Formeleingabe zu Mehrdeutigkeiten kommt.

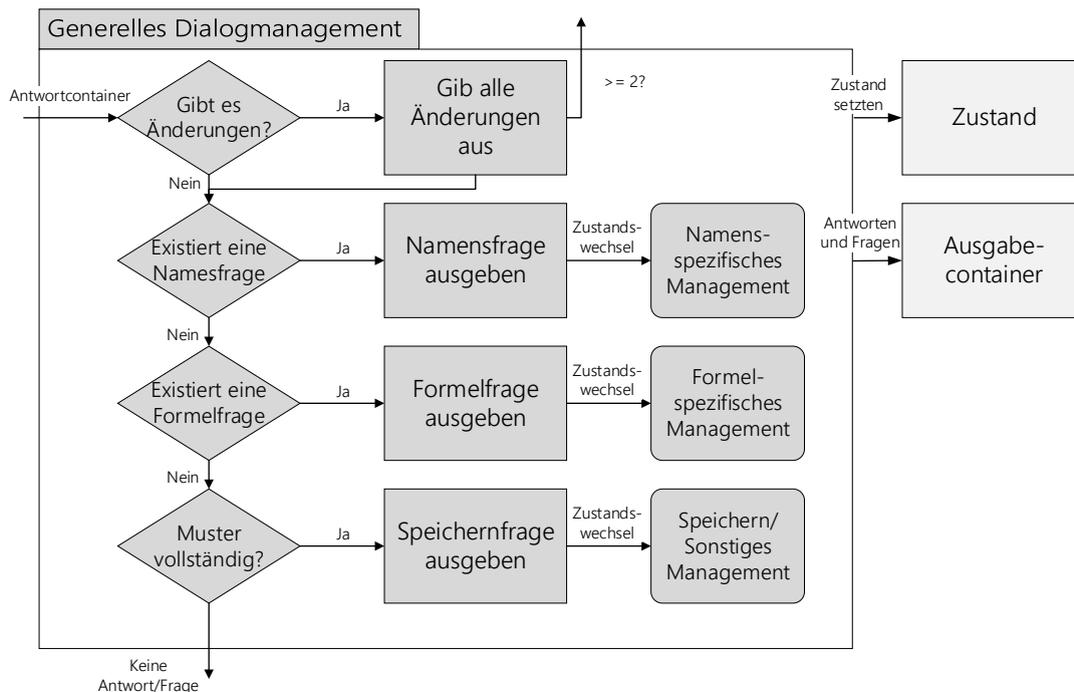


Abbildung 4.12.: Ablauf des generellen Dialogmanagements

4.3.3. Spezielles Dialogmanagement

Um detailliertere Antworten und Hilfen an den Nutzer ausgeben zu können, wurden die speziellen Dialogauswahlverfahren ausgelagert. Um in einen der im Folgenden beschriebenen Zustände zu gelangen, muss vorher von der generellen Interpretation eine zugehörige Frage an den Nutzer ausgeben werden. Sowohl das namensspezifische als auch das formelspezifische Auswahlverfahren verfügen dabei über eine iterative Umsetzung, die es erlaubt dem Nutzer immer präzisere Fragen zu stellen.

Namensspezifisches Dialogmanagement

Befindet sich das Dialogmanagement in diesem Zustand werden spezielle Ausgaben zur Namensfindung des Musters ausgegeben. Dazu wird vom Dialogsystem durch gezielte Fragen versucht das vom Nutzer gewählte Wort in der Eingabe zu isolieren. Das Dialogsystem geht automatisch in diesen Zustand über, sobald im Gespräch eine Namensfrage an den Nutzer ausgegeben wurde. Diese erste Systemfrage ist noch sehr allgemein und soll den Nutzer nur mitteilen was zur Vollendung seines Musters benötigt wird:

„Please tell me the name of your pattern.“

Sollte der Nutzer daraufhin keine passende Antwort an das System richten, wird iterativ versucht den gewählten Namen in der Eingabe einzugrenzen:

„Please tell me only the name for your pattern“.

Das Dialogsystem erwartet auf diese Frage den Namen als ein isoliertes Wort. Sollte der Nutzer dieser Aufforderung nachkommen, kann jetzt eine genaue Aussage darüber getroffen werden, ob das gewünschte Wort noch zur Verfügung steht oder nicht. Deshalb wird im zweiten Durchlauf dem Nutzer bei einem Misserfolg direkt mitgeteilt, dass sein gewählter Name nicht mehr zur Verfügung steht und ein Anderer gewählt werden muss:

„The name you told me seems to be already in use, can you tell me another one?“

Sollte nun im dritten Versuch immer noch kein passender Name gefunden worden sein, wird zur Vermeidung einer „Dead End Situation“ (vergleiche [HG04]) an dieser Stelle zurück zum generellen Dialogmanagement gewechselt und dem Nutzer mitgeteilt:

„Let's try something else“.

Eine solche Sackgasse im Dialog ist für den Nutzer meist sehr frustrierend, da ein Programm in einer solchen Situation endlos dieselbe Ausgabe erzeugt und dies oft nur durch einen Neustart des gesamten Systems verlassen werden kann. Der Wechsel zurück zum namensspezifischen Dialogmanagement kann zu einem späteren Zeitpunkt erfolgen, denn zur Komplettierung des Musters wird zwingend ein Name benötigt.

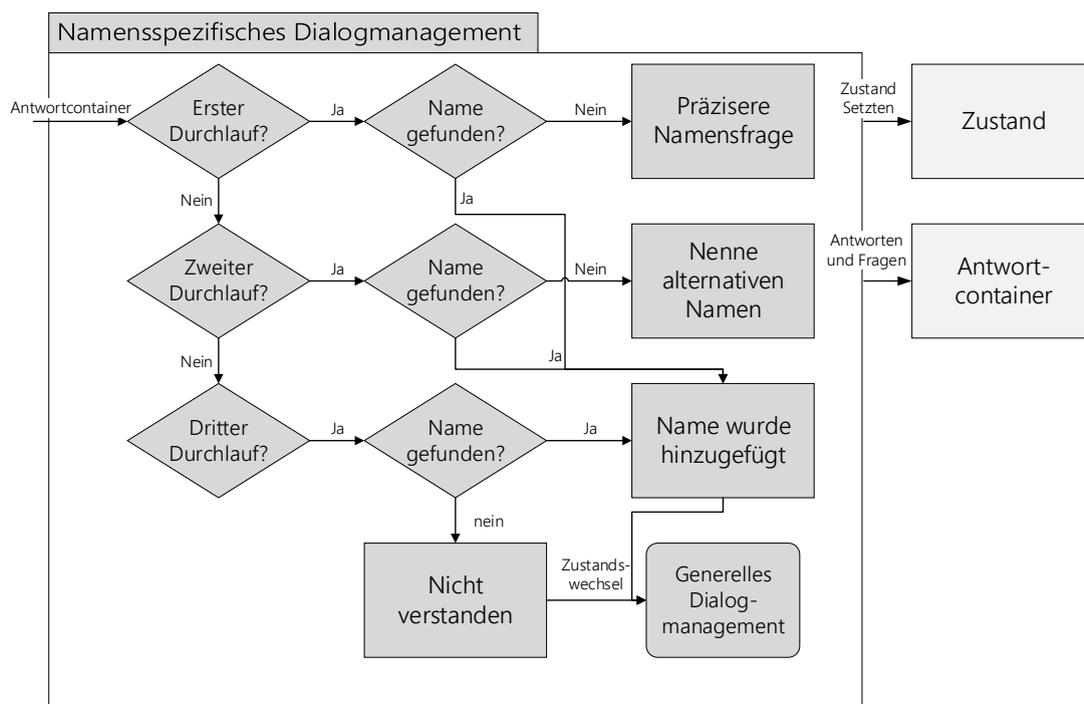


Abbildung 4.13.: Ablauf des Namensspezifischen Dialogmanagement

Formelspezifisches Dialogmanagement

Ebenso, wie für die Namensfindung, gibt es zur Formelerfassung auch einen speziellen Zustand der Dialogmanagement-Einheit. Auch bei Auswahl dieses Zustandes wurde vorher eine Frage nach der Formel des neuen Musters an den Nutzer ausgegeben. Im ersten Durchlauf wird deshalb bereits eine mathematisch interpretierbare Eingabe erwartet (siehe Abbildung 4.14). Sollte keine passende Antwort vom Nutzer erfolgt sein, wird zunächst ein Hilfetext bezüglich der Benutzung erstellt und ausgegeben. In diesem Hilfetext wird dem Nutzer erklärt, dass er zur Definition seiner neuen Funktion als Funktionsparameter am besten die Kleinbuchstaben a bis z verwenden soll. Im zweiten Durchlauf wird bei Misserfolg eine Beispielingabe ausgegeben:

„I still did not understand you, here is an example for an easy formula: „JustLingo use the sum of a, b and c for my pattern.“ Can you tell me your formula one more time?“

Sollte auf diese Frage immer noch keine passende Äußerung des Nutzers erfolgen, findet wie zuvor ein Zustandswechsel zurück zum generellen Dialogmanagement statt, um eine Sackgasse im Dialog zu vermeiden. Da zur Vollendung des Musters eine Formel ebenso wichtig wie ein Name ist, kann die Eingabe zu einem späteren Zeitpunkt erneut aufgenommen werden.

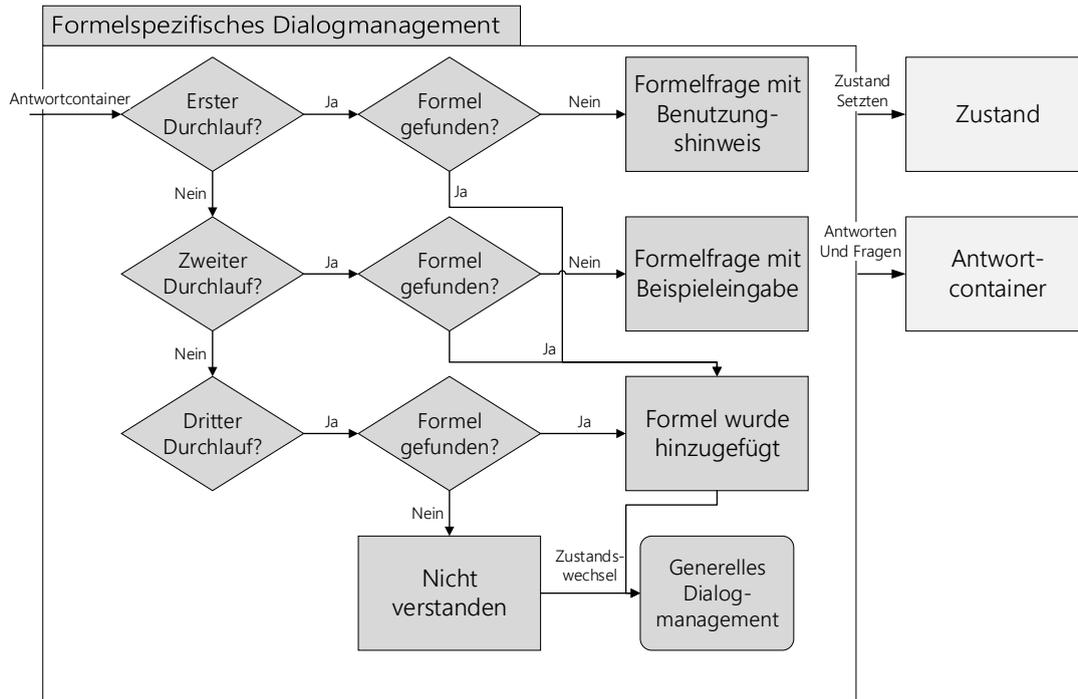


Abbildung 4.14.: Ablauf des Formelspezifischen Dialogmanagement

Speichern/Sonstiges Dialogmanagement

Sobald das neue Muster vollständig ist, wird das System den Nutzer automatisch fragen, ob er sein Muster speichern möchte. Diese Aufforderung erfolgt in Form einer Ja/Nein-Frage. Sollte der Nutzer weiterhin Änderungen an seinem Muster vornehmen wollen, kann er diese Frage ignorieren und mit weiteren Anweisungen fortfahren. Erst wenn er auf die Frage zum Speichern mittels eines klaren Ja oder Nein antwortet findet der Zustandsübergang zu dieser speziellen Dialogauswahl (siehe Abbildung 4.15) statt. Dieses Modul der Dialogmanagement-Einheit ist im Gegensatz zu den den beiden Anderen nicht iterativ umgesetzt. Stattdessen wird die Auswahl anhand der im vorherigen Durchlauf an den Nutzer ausgegebenen Frage getroffen. Wurde eine Frage zum Speichern gestellt, kann der Nutzer diese entweder bejahen und es werden alle nötigen Datenbankeneinträge erstellt sowie dem Nutzer eine Nachricht über Erfolg oder Misserfolg ausgegeben. Sollte das Speichern erfolgreich gewesen sein wird anschließend die Dialogmanagement-Einheit und die Mustervorlage in den Ausgangszustand zurückversetzt werden. Zuletzt wird die Pipeline zur Mustergenerierung verlassen, damit der Nutzer sein neues Muster sofort im mathematischen Modus benutzen kann. Bei einem Misserfolg findet hingegen nur ein Zustandswechsel zum generellen Dialogmanagement statt.

Wurde die Frage zum Speichern vom Benutzer mit einem Nein beantwortet, nimmt das System an, dass der Nutzer etwas an seinem Muster ändern möchte. Dazu wird eine passende Auswahlfrage ausgegeben. Der Nutzer hat nun die Möglichkeit über passende Schlüsselwörter (vergleiche 4.2.2) eine Auswahl zu treffen. Je nach Eingabe wird dem Nutzer mitgeteilt, welche Parameter seines Musters zurückgesetzt worden sind und anschließend

direkt mit einer Frage zur Bestimmung dieses Parameters fortgefahren. Zusätzlich findet ein Zustandswechsel statt um korrekt auf folgende Nutzereingaben zu reagieren. Sollte die Frage zum Ändern verneint worden sein, wird anschließend gefragt, ob das Muster komplett verworfen werden soll. Bei dieser Frage erwartet der Dialogmanager wieder eine einfache Ja- oder Nein-Antwort. Bei einem Ja wird der gesamte Fortschritt verworfen und die Pipeline zur Mustergenerierung verlassen. Bei einem Nein als Antwort finden keinerlei Änderungen statt und der nächste Dialogentscheidungsschritt findet wieder im generellen Modus statt.

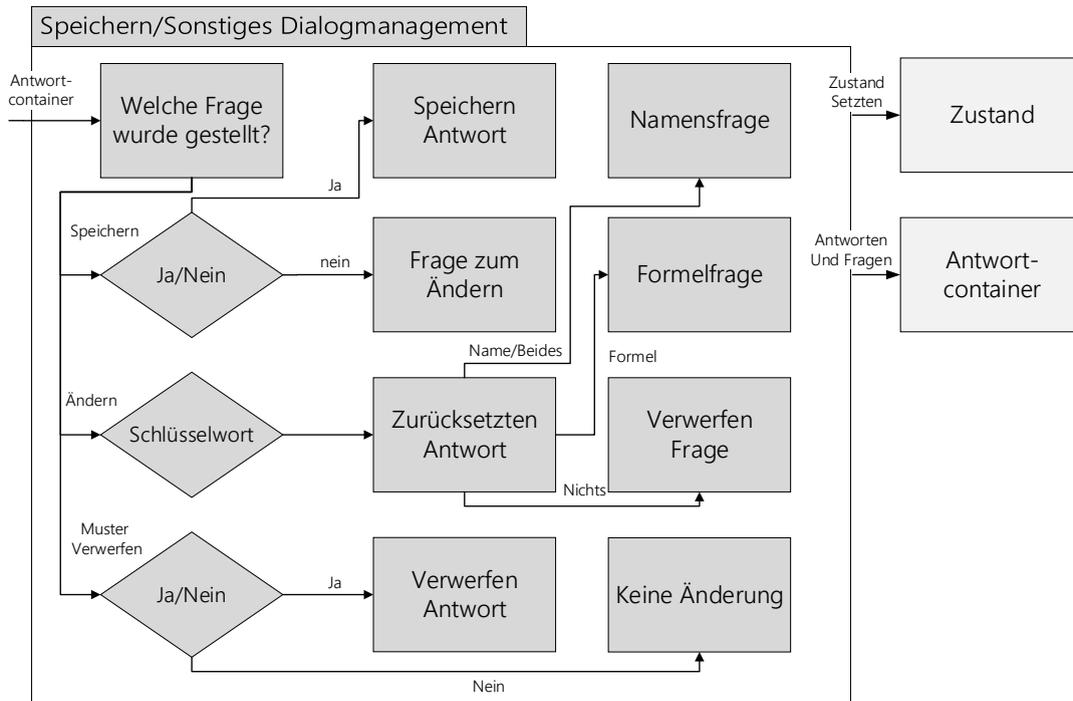


Abbildung 4.15.: Ablauf des Speichern/Sonstiges Dialogmanagement

Rückfall Dialogmanagement

Solang sich die Dialogmanagement-Einheit in einem der speziellen Zustände befindet, wird zusätzlich noch eine Rückfall-Auswahl durchgeführt. Dadurch wird ermöglicht, dass in jedem Zustand auch Nutzereingaben erkannt werden, die Kontext fremd sind. An dem in Tabelle 4.6 gezeigten Dialogauszug befindet sich das Dialogsystem nach der Frage „Can you tell me the name of your pattern?“ im namensspezifischen Zustand. Wenn die in Abbildung 4.13 gezeigte Ausführung stattfindet, wird keine Antwort erzeugt und deshalb im Anschluss das Rückfall-Management ausgeführt. Dort wird erkannt, dass der Benutzer JustLingo eine Formel diktiert hat und die passende Antwort ausgegeben. Anschließend findet ein Zustandswechsel zum generellen Antwortmanagement statt.

Systemantwort

Can you tell me the name of your pattern?

Nutzereingabe

The formula is a plus c.

Systemantwort

Ok, I added the formula ($a + c$) to your pattern.

Tabelle 4.6.: Auszug eines kontextunabhängigen Dialogverlaufes

4.3.4. Kontextspeicherung

Nachdem durch die Dialogmanagement-Einheit eine Antwortauswahl getroffen wurde, muss diese noch zusammen mit der zugehörigen Nutzereingabe im aktuellen Kontext abgespeichert werden. Ansonsten würden die Rückbezüge auf früher gestellte Fragen und Aussagen aufgelöst werden können. Um mit der Mustergenerierung zu funktionieren muss während der Erstellung ein selektives Speichern erfolgen. Denn schon bei ersten Tests hat sich herausgestellt, dass sonst das schrittweise Aufbauen von mathematischen Formeln nicht korrekt funktioniert, da wie im Beispiel in Abbildung 4.16 gezeigt wird, immer ein Rückbezug auf die vorherige Eingabe durchgeführt wird. Der Nutzer erklärt in diesem Beispiel JustLingo die Formel seines neuen Musters. Dabei fängt er mit der einfachen Teilformel „ $Sum(a,b,c)$ “ an. Im Anschluss wird diese Formel im Kontext vermerkt und abgespeichert. Bei seiner nächsten Äußerung teilt der Nutzer JustLingo mit, wie seine Formel erweitert werden soll. Dabei erkennt JustLingo eine Mehrdeutigkeit und stellt dem Nutzer eine Auswahlfrage. An dieser Stelle werden immer alle verschiedenen Formelzusammensetzungen im Kontext gespeichert. Im nächsten Schritt rechnet JustLingo mit einer Auswahlentscheidung für eine Formel. Sollten alle Formeln der Auswahlentscheidung nicht der Intention des Nutzers entsprechen und dieser deshalb dem System mitteilen, dass die Formeln verworfen werden sollen, tritt ein bisher nicht berücksichtigtes Problem auf: Will der Nutzer nun erneut versuchen eine Eingabe zu machen, die sich auf seinen bisherigen Fortschritt bezieht, wird der Rückbezug in der vorherigen Version von JustLingo falsch interpretiert. Statt wieder auf der Summe aufzubauen, werden stattdessen die eigentlich verworfenen Ergebnisse der Auswahlfrage benutzt. Diese hat der Nutzer aber durch seine Eingaben eigentlich verworfen. Deshalb wird innerhalb der Mustergenerierung in die Kontextspeicherung eingegriffen um dies zu verhindern und der verworfene Kontexteintrag als ungültig markiert. Zusätzlich findet in der Mustergenerierung nach jedem Durchlauf ein Abgleich zwischen Kontext und Mustervorlage statt. Dadurch wird erreicht, dass der Nutzer sich in all seinen Äußerungen immer auf die aktuelle Formel in der Mustervorlage beziehen kann. Zusätzlich werden über diesen Abgleich auch alle Referenzen als ungültig markiert, wenn der Nutzer die Formel seines Musters zurücksetzt.

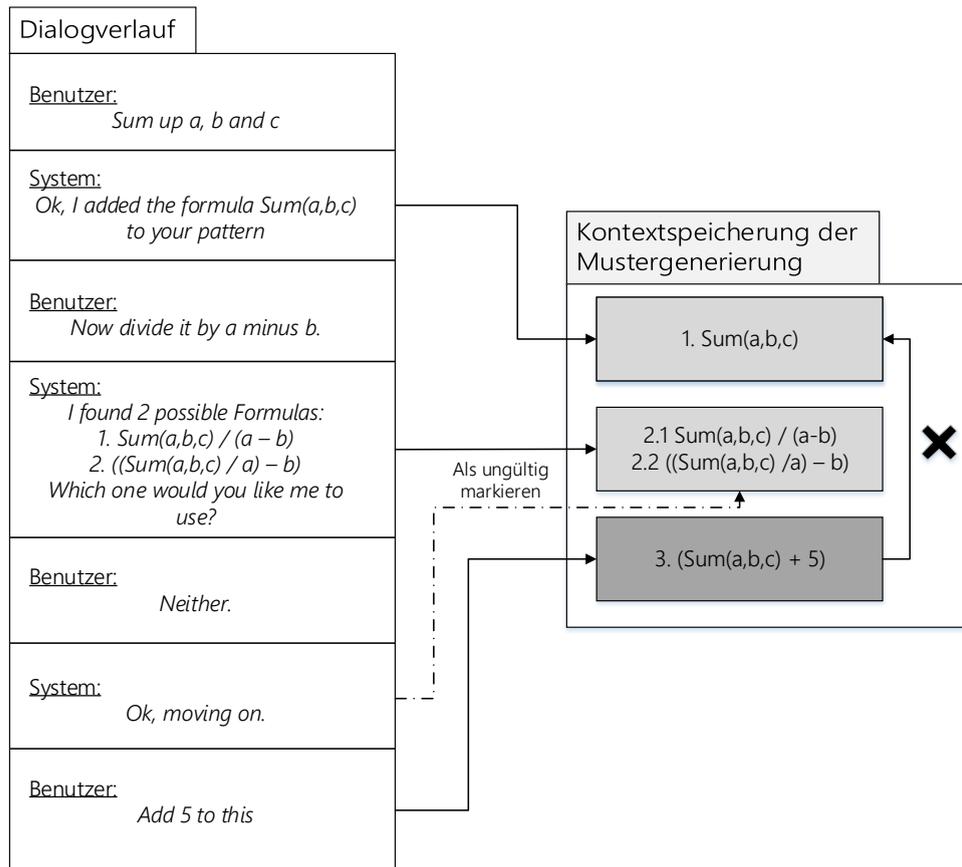


Abbildung 4.16.: Beispieldialog mit Kontextspeicherung

4.4. Datenbanken

Bis jetzt wurde beschrieben, wie der Ablauf zum Sammeln der nötigen Daten zur Erstellung eines neuen Musters abläuft. Deshalb soll nun abschließend der Ablauf zum Speichern eines Musters betrachtet werden. Dabei wird zuerst auf den allgemeinen Aufbau der Datenbanken eingegangen und anschließend, wie das Schreiben in sie abläuft.

4.4.1. Aufbau der Datenbanken

An dieser Stelle wird nun erklärt, wie die einzelnen Datenbanken aufgebaut sind und wie ein neues Muster abgespeichert wird. Zusätzlich zu der Antwortdatenbank, die alle Systemreaktionen enthält, verfügt JustLingo noch über drei weitere Datenbanken.

Wörter-Datenbank

In der Wörter-Datenbank befinden sich alle Einträge für Wörter und Synonyme. Diese Datenbank enthält den gesamten Wortschatz, über den JustLingo verfügt. Diese Datenbank verfügt im aktuellen Stand über mehr als 400 Wörter. Ein Datenbankeneintrag für das Wort *formula* aus der Wörter-Datenbank wurde in Tabelle 4.7 dargestellt. Im *Substitution*-Feld wird vermerkt, durch welches Wort das Eingabewort ersetzt wird. Diese Ersetzung findet statt, damit die Implementierung nur für repräsentierende Wörter umgesetzt werden muss, aber JustLingo dennoch alle Synonyme, die für ein Wort existieren, erkennen kann. Im *Meaning*-Feld steht, welche Bedeutung dieses Wort im Satz hat. Im Beispiel wird das Wort „formula“ als Muster markiert. Zuletzt enthält die Datenbank noch eine

Information darüber um welche Wortart es sich handelt. Dabei wird zwischen Adjektiven, Substantiven, Verben und Bindungswörtern unterschieden.

Argument	Ausprägung
Name	formula
Substitution	pattern
Meaning	Pattern
Type	noun

Tabelle 4.7.: Datenbankeneintrag der Wörter-Datenbank

Befehls-Datenbank

Die Befehls-Datenbank enthält alle für das System wichtigen Informationen zu einem Operator. In Tabelle 4.8 ist der Datenbankeneintrag für die Wurzel Funktion als Beispiel dargestellt. Ein solcher Datenbankeneintrag enthält zum einen Informationen zur Operation selber, den Namen und um welchen Typ von Operation es sich handelt. Zusätzlich sind Informationen zu den Parametern, die von dieser Operation benutzt werden, abgespeichert. Dazu gehören wie viele Eingaben diese Funktion entgegennimmt, ob diese Parameter zwingend benötigt werden und welche Eigenschaft für diese Parameter gelten muss. Dabei wird zwischen *Single* und *Any* unterschieden. Zu Parametern mit der Eigenschaft *Single* gehören Zellenangaben und Zahlen. Zahlen können auch Ergebnisse von anderen Operationen sein, was ineinander verschachtelte Funktionen erlaubt. Zuletzt werden noch Informationen zur Repräsentation eines Operators gespeichert. Dazu gehören die Excel[®]-Formel der Operation in der jeweiligen Sprache und eine natürlichsprachliche Dialogausgabe für den Nutzer. Um eine zukünftige Erweiterung für andere Sprachen möglichst einfach zu machen wird die Formel und die Ausgabe einer Operation passend mit den in C# verwendeten „Culturenames“ gespeichert [Cor14].

Operation:	
Key	root
Type	Operation
Parameter:	
Anzahl	1
Use	required
Type	single
Formula:	
en-US	SQRT
de-DE	WURZEL
Output:	
en-US	the root of %1

Tabelle 4.8.: Datenbankeneintrag der Befehls-Datenbank

Syntax-Datenbank

In der Syntax-Datenbank finden sich für die Benutzung von Operatoren weitere wichtige Informationen. Die Einträge sind vor allem für die Identifizierung und Anordnung von Parametern wichtig. Als Beispiel wird nun der folgende Satz betrachtet:

„*JustLingo please subtract A1 from B2.*“

Nachdem die Funktion *difference* markiert wurde, wird in der Syntax-Datenbank nach zugehörigen Syntaxmustern gesucht. Dabei wird die Regel „%2 from %1“ gefunden. Diese sagt im Bezug auf die Anordnung der Parameter aus, dass sie zur korrekten Auswertung vertauscht werden müssen. Da die natürlichsprachliche Formulierung dieser Operation zuerst den abzuziehenden Betrag und dann erst den Ausgangswert nennt. Zusätzlich stehen in der Syntax-Datenbank Informationen zu den einzelnen Parametern. Bei einer Subtraktion dürfen zum Beispiel beide Parameter nur vom Typ *Single* sein. Dies bedeutet, dass es sich entweder um Zahlen oder Zellen handeln muss. Es dürfen keine Spalten, Zeilen oder Bereiche als Eingabeparameter übergeben werden.

4.4.2. Speichern eines Musters

Nachdem in Zusammenarbeit mit dem Nutzer das Sammeln der Informationen für ein Muster abgeschlossen wurden, erfolgt ein automatisches Speichern aller Informationen in den Datenbanken. Zur korrekten Funktion des Musters müssen Einträge in allen aufgezählten Datenbanken vorgenommen werden. Beim Erstellen wurde darauf geachtet, dass sowohl für englische, als auch deutsche Versionen der Tabellenkalkulationssoftware die korrekten Daten in die Datenbank übertragen werden. Der genaue Ablauf des Hinzufügens wird im Folgenden genauer beschrieben.

Bestandteile eines Musters

Zur Erstellung eines Musters werden, wie bereits erwähnt, drei Informationen benötigt. Der Name, eine Formel und die Anzahl an Parametern, die das Muster benutzt. Aus diesen Informationen lässt sich ein vollständiger Datensatz zusammenbauen.

Da das Muster später über den Namen identifiziert werden soll, ist dessen Eindeutigkeit nötig, damit ein passender Verweis in der Wörter-Datenbank erstellt werden kann. Dieser Eintrag sieht bei jedem Muster gleich aus. Der Name wird gleichzeitig als Name und „Substitution“ verwendet (vergleiche Tabelle 4.7). Jedes neu erstellte Muster wird darüber hinaus als Substantiv des Typs Operator klassifiziert.

Nachdem ein erfolgreiches Eintragen in die Wörter-Datenbank abgeschlossen wurde, wird als nächstes der Verweis in der Befehls-Datenbank erstellt. Dabei wird ein Datenbankeintrag wie in Tabelle 4.8 erzeugt, wobei sich der Schlüssel der Operation aus dem Namen des Musters ergibt. Im Parameterfeld wird die beim Erstellungsprozess gefundene Parameteranzahl eingetragen. Dieser Wert kann bei der aktuellen Umsetzung zwischen null und 26 liegen. Bei Funktionen, die keine Parameter nehmen handelt es sich um statische Funktionen. Diese können entweder wie zum Beispiel die *PI*-Funktion in der Tabellenkalkulationssoftware konstant den Wert von Pi zurückgeben, oder es handelt sich um vom Nutzer erstellte Funktionen, die mit statischen Zellenangaben arbeiten. Sollte die Anzahl größer als null sein, wird zu jedem Parameter vermerkt um welchen Typ es sich handelt. Dabei wird auf die Vererbung der Operatoren-Eigenschaften geachtet (siehe 4.4.2). Um die selbst generierten Muster während dem Arbeiten möglichst übersichtlich darstellen zu können wird als *Formula* der Name des Musters in Großbuchstaben gespeichert. Diese Schreibweise wird während des Mustergenerierungsprozesses benutzt, damit der Nutzer beim zusammensetzen mehrerer neuer Muster besser erkennen kann, wie seine Formel funktioniert

(vergleiche Tabelle 4.9). Dabei wurde die Diskriminante als separate Funktion im Voraus definiert. Als letztes wird der Datenbankeintrag in die Syntax-Datenbank durchgeführt. Dabei wird in der Mustergenerierung immer das Standardmuster zur Parameterzuordnung verwendet. Da davon ausgegangen wird, dass der Nutzer bei der Benutzung später immer die Parameter in derselben Reihenfolge übergeben wird, da sonst unterschiedliche Ergebnisse bei der Auswertung der Funktion erzielt würden. Das Standardmuster entspricht der Form:

Calculate **Funktionsname** of Parameter₁, Parameter₂,
Parameter₃, ..., Parameter_{n-1} and Parameter_n

Darstellung	
Ohne Kurzschreibweise:	(b - WURZEL(POTENZ(b;2) - PRODUKT(4;a;c))) / PRODUKT(2;a)
Mit Kurzschreibweise:	(b - DISKRIMINANTE(a;b;c)) / (PRODUKT(2;a))

Tabelle 4.9.: Funktionsdarstellung der Mitternachtsformel in der Mustergenerierung

Eigenschaften der Excel[®]-Operatoren

Während der Entwicklung der Mustergenerierung haben sich zwei Probleme beim Erstellen der neuen Muster ergeben, die bei der Planung zuerst nicht berücksichtigt wurden. Im Folgenden werden diese zuerst beschrieben und anschließend gezeigt, wie sie gelöst wurden.

Operatoren Darstellung

Die Funktionen der von der Tabellenkalkulationssoftware verwendeten Formelsprache lassen sich in drei Kategorien bezüglich ihrer Darstellung aufteilen:

1. **Reine Formeldarstellung:** Dazu gehören alle Funktionen, die nur über eine Formeldarstellung verfügen. Ein Beispiel hierfür ist die Mittelwertfunktion: *MITTELWERT()*.
2. **Reine Zeichendarstellung:** Bestimmte Funktionen lassen sich in der Tabellenkalkulationssoftware nur mit mathematischen Zeichen darstellen. Dazu gehören beispielsweise die Differenz (-) und das Dividieren (/).
3. **Gemischte Darstellung:** Diese Funktionen können sowohl durch Formeln, als auch durch Zeichen repräsentiert werden. Dazu gehört die Funktion zum Addieren, die entweder als *SUMME()* oder als (+) dargestellt werden kann.

Funktionen der ersten oder zweiten Kategorie müssen in der Mustererzeugung nicht gesondert betrachtet werden, da es für beide nur jeweils eine, unveränderliche, Darstellungsmöglichkeit existiert. Bei Funktionen der dritten Kategorie musste eine Sonderbehandlung vorgenommen werden, da die sprachliche Analyse von JustLingo wenn möglich die Zeichendarstellung präferiert. Im Prozess der Mustergenerierung wird jedoch mit Variablen gearbeitet, weil an dieser Stelle noch nicht gesagt werden kann, welcher Parametertyp bei der späteren Benutzung verwendet wird. Die Zeichendarstellung einer Funktion kann nur zusammen mit Parametern des Typs „Single“, also Zellen und Zahlen, verwendet werden. Falls eine Darstellung in der Zeichenschreibweise nicht möglich ist, wird die Funktion automatisch in Formeldarstellung ausgegeben (siehe Abbildung 4.17). In der Mustergenerierung wurde diese Auswahl so gelöst, dass dem Nutzer immer die reine Zeichenschreibweise

ausgegeben wird, da diese vor allem bei sehr langen Formeln übersichtlicher ist. Zum Speichern in die Datenbank wird jedoch die Formelschreibweise genommen, da sie alle Arten von Eingaben zulässt und beim Erstellungsprozess dieser Informationsverlust verhindert werden soll.

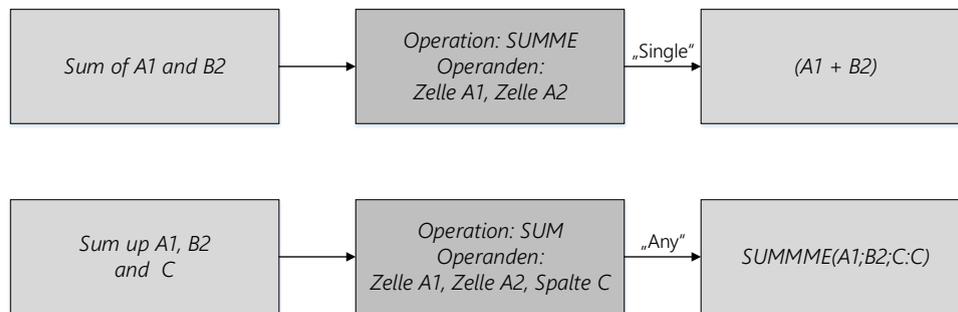


Abbildung 4.17.: Beispiel der Formelauflösung

Vererbung der Eigenschaften

Neben der Darstellung der Formel wird bei der Mustergenerierung auf die Vererbung der Eigenschaften der einzelnen Operatoren geachtet. In den Abbildungen 4.18 und 4.19 ist beispielhaft der Vererbungsbaum für die Mitternachtsformel aufgezeichnet. Dabei steht in jedem Knoten des Graphen die angewendeten Funktionen und deren zulässige Parametertypen. Zuerst soll der Aufbau der Diskriminantenfunktion betrachtet werden. Auf der untersten Ebene des Schaubildes befinden sich die einzelnen Parametern aus denen sich die Funktion aufbaut. Dazu gehören die Variablen a , b und c sowie die Zahl 2. Um eine Aussage über den erlaubten Typ der Parameter treffen zu können muss nun rekursiv überprüft werden, zu welchen Operationen diese zugeordnet werden. Wie zu sehen ist, sind Variable b und die Zahl 2 Teil der Funktion zum Potenzieren. Da die Funktion Potenzieren aber nur Parameter des Typs *Single* zulässt folgt automatisch, dass Variable b vom Typ *Single* sein muss. Die Variablen a und c sind dahingegen Bestandteil einer Multiplikation, die auch Parameter des Typs *Any* erlaubt. Daher muss für diese beiden Variablen keine Einschränkung getroffen werden.

In der nächsthöheren Stufe ist zu sehen, dass die Multiplikation und die Potenz Teil einer Differenz sind. Die Differenz erlaubt nur Eingaben des Typs *Single*. Dies stellt aber kein Problem dar, da Funktionen in Excel[®] von innen nach außen ausgewertet werden. Deswegen wird die Multiplikation vor der Differenz ausgewertet. Und das Ergebnis einer Multiplikation ist eine Zahl und Zahlen sind immer vom Typ *Single*. Zuletzt wird auf das Ergebnis der Differenz noch die Wurzelfunktion angewendet.

Bei der Erstellung der Mitternachtsformel wird exakt auf dieselbe Weise Verfahren. Auf unterster Ebene steht hier die Diskriminantenfunktion, die drei Parameter entgegen nimmt. Da bei der Vererbung der Operatoren in der Diskriminantenfunktion bereits festgestellt wurde, dass die Variable b höchstens vom *Single* sein darf, wird diese Festlegung direkt an die Mitternachtsformel weitervererbt. Für die Variable a und c bestehen durch diese Unterfunktion bis jetzt keine Einschränkungen, weshalb die einzelnen Bestandteile der Mitternachtsformel untersucht werden müssen, um eine korrekte Aussage treffen zu können. Nachdem die gesamte Auswertung abgeschlossen ist, ergibt sich, dass durch die Mitternachtsformel keine weiteren Einschränkungen in Bezug auf die Parameter getroffen werden muss. Diese Analyse wird durchgeführt, damit das Dialogsystem bei Auftreten von Fehlern in der mathematischen Benutzung präzise Aussagen treffen kann und den Nutzer auf Fehl-

benutzung hinweisen kann. Wenn nämlich die Nutzung der Variable b nicht eingeschränkt wäre, würde Excel bei der Berechnung keinen Wert zurückliefern, wenn unzulässige Operanden übergeben werden. Das Dialogsystem würde dies bemerken, könnte aber nur die Aussage treffen, dass während der Berechnung Fehler aufgetreten sind.

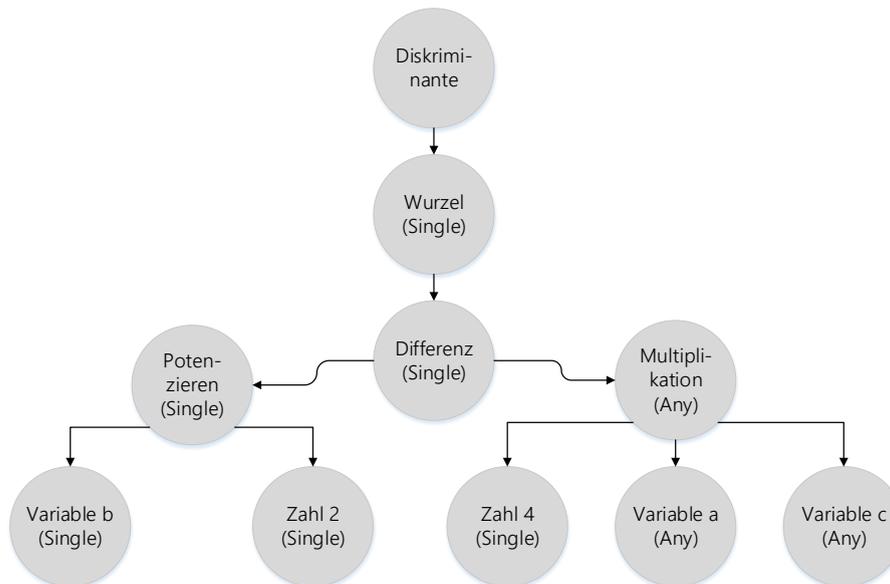


Abbildung 4.18.: Operatorenvererbung in der zugehörigen Diskriminantenfunktion

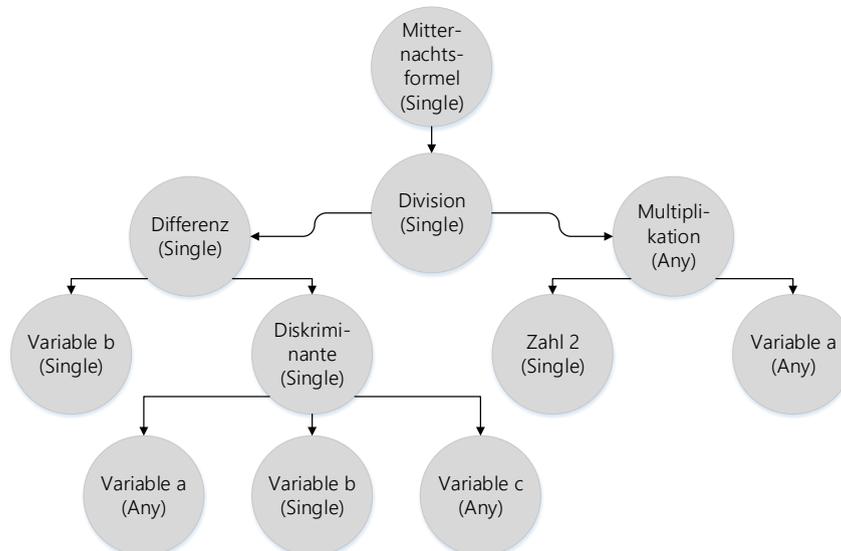


Abbildung 4.19.: Operatorenvererbung in der Mitternachtsformel

5. Evaluation

In den vorherigen Kapiteln wurde der theoretische Aufbau der Mustergenerierung und deren technische Umsetzung ausführlich erklärt. Um nun aber eine möglichst objektive Aussage über die Güte der umgesetzten Erweiterung treffen zu können, wurde eine Studie, die das Erstellen neuer Muster evaluiert, entworfen. Da der Lernprozess des Systems über natürliche Sprache abläuft ist keinesfalls voraussehbar, wie gut die Erkennung in einem praxisnahen Umfeld funktioniert. Deshalb wurden freiwillige Teilnehmer gesucht, die das Arbeiten mit der Mustergenerierung testen sollten. Um die Studie möglichst objektiv zu halten wurden dabei den einzelnen Testpersonen nur erklärt, worum es sich beim System JustLingo handelt, und was evaluiert werden soll. Als Hilfe haben die Probanden lediglich eine Vokabelliste (siehe Anhang B) erhalten, auf der einige wenige englische Übersetzungen für deutsche Worte angegeben waren. Der genaue Ablauf der Studie wird nun im Folgenden beschrieben. Darüber hinaus werden erkannte Mängel und Verbesserungsmöglichkeiten, die während der Durchführung aufgefallen sind, beschrieben. Zuletzt werden die aus der Evaluierung gewonnenen Erkenntnisse dargestellt und entschieden wie gut die geplanten Ziele umgesetzt wurden.

5.1. Aufbau und Durchführung

Um den Aufbau besser nachvollziehen zu können findet sich im Anhang A dieser Arbeit eine Kopie des Fragebogens. Der Aufbau der Studie unterteilt sich in drei wesentliche Bestandteile. Zuerst wurden dem Teilnehmer allgemeine Fragen zur Erfahrung im Umgang mit Excel[®] gestellt. Dabei wurde zuerst nach dem Umfeld in dem Excel[®] benutzt wird gefragt. Zusätzlich sollte die Testperson ihren Erfahrungsstand im Umgang mit Excel[®] selbst einschätzen und angeben, wie viele Jahre sie schon mit dieser Software gearbeitet hat. Es wurden auch personenbezogene Daten erhoben. Darunter fällt die Muttersprache und der aktuelle Berufsstand der Person. Zusätzlich wurde noch gefragt, ob die Testpersonen schon einmal mit JustLingo gearbeitet, oder an einer früheren Studie teilgenommen hatten.

Im nächsten Schritt der Studie wurden die eigentlichen Testaufgaben zum Arbeiten mit der neuen Mustererweiterung gestellt. Dabei sollten die Testpersonen zuerst an den Umgang mit der Mustergenerierung herangeführt werden. Dieser Teil der Evaluation fließt am Ende auch in die Auswertung mit ein und erfolgte anhand der Erstellung der dritten binomischen Formel. Der Erstellungsprozess wurde dabei in fünf logische Schritte unterteilt, sodass der Nutzer für den im Anschluss folgenden, dritten Teil der Studie weiß, wie die Aufgaben zu lösen sind. Das Vorgehen setzt sich aus den folgenden Schritten zusammen:

1. Mitteilen eines Erweiterungswunsches.
2. Benennen des neuen Musters.
3. Mitteilen des ersten Formelteiles.
4. Erweitern der Formel über Rückbezüge im Dialog.
5. Abspeichern des nun vollständigen Musters.

Hierbei wurde bei den einzelnen Schritten genannt, warum die jeweiligen Informationen zur Erstellung nötig sind, und wie diese dem Dialogsystem erklärt werden können. Dazu wurden bei jedem Aufgabenteil zwei deutsche Beispielsätze genannt, die der Testperson einen Eindruck davon machen sollten, wie die Mustergenerierung zu benutzen ist. Die Beispiele wurden absichtlich nur auf deutsch vorgegeben, damit innerhalb dieser Aufgabe nicht jeder Nutzer dieselben Eingaben an das Dialogsystem liefert. Zur Erfolgskontrolle wurden alle Eingaben vom System protokolliert. Zusätzlich sollte der Nutzer auf dem mitgelieferten Ausdruck der Studie bei den jeweiligen Schritten ankreuzen, ob ein Erfolg erzielt werden konnte. Darüber hinaus konnte der Nutzer Anmerkungen jeglicher Art zu jeder Teilaufgabe hinzufügen. Die meisten Teilnehmer haben diese Möglichkeit genutzt und Vorgehensweisen, Hinweise, Verbesserungsvorschläge, Wünsche, Lob und auftretende Probleme dokumentiert

Im letzten Teil der Studie sollten die Testpersonen selbständig JustLingo zwei mathematische Funktionen beibringen. Zu Beginn wurde der Testperson das Bild eines rechtwinkligen Dreiecks gezeigt (siehe Abbildung 5.1).

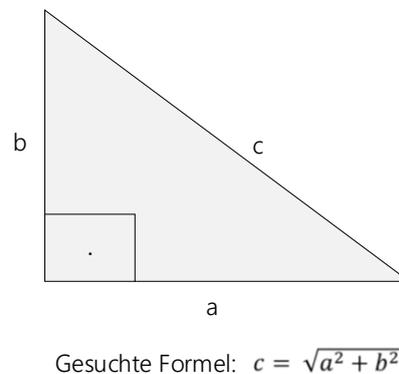


Abbildung 5.1.: Satz des Pythagoras

Die Aufgabe dazu lautet eine Formel zur Berechnung der Hypotenuse zu finden. Die Testpersonen sollte daher zuerst schriftlich angeben, welche Formel sie nun JustLingo zur Lösung dieses Problems beibringen wollen. Vorgesehen war dabei, dass der Satz des Pythagoras benutzt werden soll. Um die Aufgabenstellung aber möglichst offen zu halten, wurde den Teilnehmern keine genauen Hinweise gegeben. Zur Datenerhebung sollte anschließend der Name der neuen Funktion und die dem System beigebrachte Formel notiert werden. Abschließend sollte noch vermerkt werden, ob die Erstellung erfolgreich durchgeführt werden konnte.

Um die Grenzen des Systems zu testen, sollten die Teilnehmer als zweites mathematisches Problem JustLingo die Mitternachtsformel, zur Lösung von quadratischen Gleichungen, erklären. Als Hilfestellung wurde den Probanden jedoch zuerst ein neues Werkzeug in die

Hand gelegt. Es wurde anhand eines unabhängigen Beispiels erklärt, wie man mathematische Formeln in mehrere Teilformeln zerlegen kann. Gleichzeitig sollte mit dieser Zerlegung getestet werden, ob das Arbeiten mit den neuen Funktionen dem Nutzer Probleme bereitet und ob solche Nutzerformeln erfolgreich zur Erstellung weiterer Muster verwendet werden können. Auch bei der Lösung dieses mathematischen Problems wurden den Testpersonen keine weitere Hilfe geleistet, sodass bei der Auswertung ein möglichst breites Spektrum an unterschiedlichen Eingabesätzen und Herangehensweisen erfasst werden konnte. Die Teilnehmer sollten anschließend alle Teilformeln, die sie zur Mustergenerierung verwendet haben, auf dem Ausdruck der Evaluation vermerken.

Die Durchführung der Studie fand abgesehen von der bereits beschriebenen Datenerhebung ausschließlich am Computer statt. Während der Bearbeitung wurden die Teilnehmer, um Fragen und Unklarheiten zu klären, betreut. Die betreuende Person beschränkte sich bei den Hilfestellungen nur auf Aussagen, die keinerlei Hinweise zur Lösung eines Problems enthielten. Zusätzlich bekam jede Testperson eine Liste mit Vokabeln. Auf dieser Liste standen ausgewählte Wörter für bestimmte mathematische Funktionen. Es waren dabei keinerlei Hinweise zur Benutzung dieser Worte in Sätzen angegeben. Die Vokabelliste diente vor allem dem Zweck das Fehlschlagen von Eingaben aufgrund von Rechtschreibfehlern zu verhindern. Die Bearbeitung der Studie durch die einzelnen Teilnehmer fand nacheinander statt.

5.2. Auswertung der erhobenen Daten

Allgemeine Daten

Insgesamt haben sich 10 Teilnehmer dazu bereit erklärt an der Evaluation dieser Bachelorarbeit teilzunehmen. Bei allen Probanden handelte es sich um Studenten aus dem Fachbereich Informatik. Es wurde 9 mal deutsch und einmal kasachisch als Muttersprache angegeben. 7 der 10 Teilnehmer gaben an, dass Sie mit Excel[®] hauptsächlich privat arbeiten. 3 benutzen die Tabellenkalkulationssoftware zusätzlich bei der Arbeit oder im Rahmen ihrer Ausbildung. Exakt die Hälfte der Teilnehmer hat schon einmal an einer Studie zum Dialogsystem JustLingo teilgenommen, die andere Hälfte hatte noch nie von JustLingo gehört.

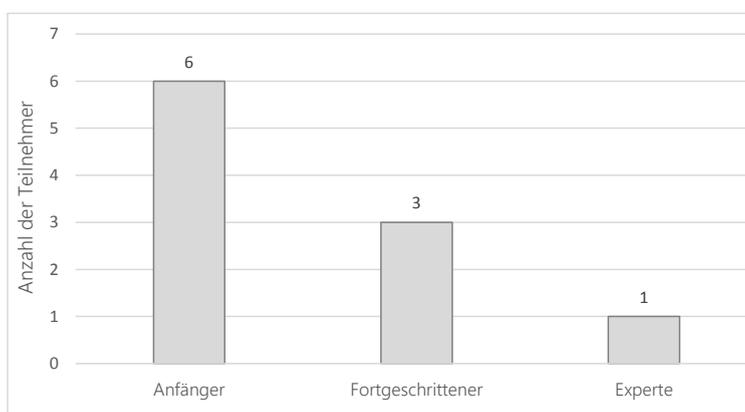


Abbildung 5.2.: Selbsteinschätzung der Erfahrung

Die meisten der Teilnehmer (6) haben sich ihrer eigenen Einschätzung nach als Anfänger im Umgang mit Excel[®] eingestuft. Drei weitere als Fortgeschritten und ein Teilnehmer hat sich selbst als Excel[®]-Experte bezeichnet, wie in Abbildung 5.2 zu sehen ist. In Abbildung 5.3 ist dargestellt, wie die Teilnehmer auf die Frage: „Wie häufig arbeiten Sie mit Excel[®]?“ geantwortet haben. 60% der Teilnehmer arbeiten höchstens einmal im Monat oder noch seltener mit Excel[®]. Nur ein Drittel der Befragten arbeitet mindestens einmal je Woche mit der Tabellenkalkulationssoftware. Zusätzlich sagte ein Teilnehmer aus, dass er noch nie zuvor mit Excel[®] gearbeitet hat. In Abbildung 5.4 ist dargestellt seit wie vielen Jahren die Teilnehmer bereits mit Excel[®] Erfahrung gesammelt haben. An dieser Stelle fällt auf, dass 60% der Teilnehmer bereits seit mindestens 9 Jahren mit Excel[®] zu tun hatten. Insgesamt ergibt sich eine durchschnittliche Erfahrung von 8 Jahren bezogen auf alle Teilnehmer.

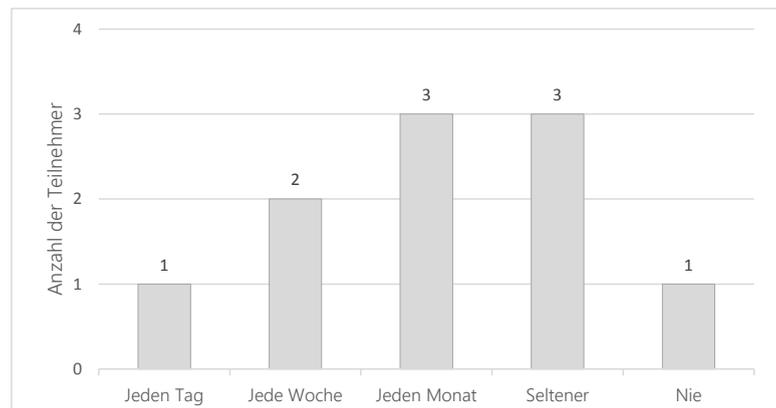


Abbildung 5.3.: Häufigkeit der Excel[®] Nutzung

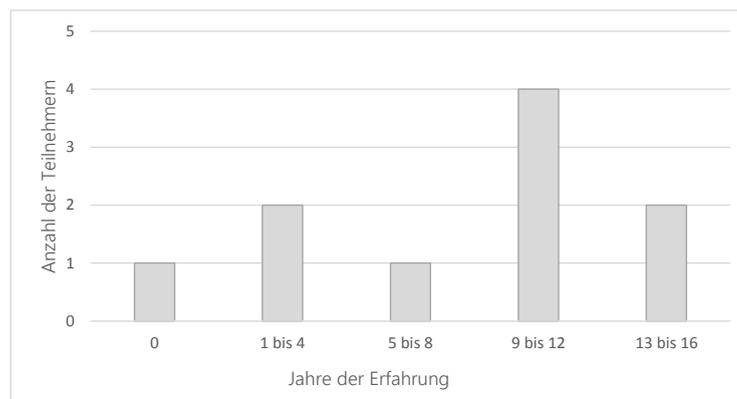


Abbildung 5.4.: Erfahrungslevel im Umgang mit Excel[®] gemessen in Jahren

Unterstütztes Arbeiten

Zur Einführung und Gewöhnung an die Mustererweiterung wurde ein aus mathematischer Sicht recht einfaches Beispiel, die dritte binomische Formel, gewählt. Zur genaueren Erfolgsanalyse wurde in Abbildung 5.5 für jeden Aufgabenteil einzeln überprüft, ob er von den Teilnehmern erfolgreich, teilweise erfolgreich oder nicht erfolgreich ausgeführt werden konnte.

Alle Teilnehmer konnten dabei ihren Wunsch zur Systemerweiterung, die Benennung ihres Musters und den ersten Teil der binomischen Formel dem Dialogsystem erklären. Dagegen konnten nur 2 von 10 Teilnehmern dem Dialogsystem die komplette Formel beibringen. Bei der Mehrheit wurde die Formel falsch erkannt und statt $(a + b) * (a - b)$ (vergleiche 5.6) wurde $((a + b) * a) - b$ (vergleiche 5.7) ausgegeben. Als Ursache für diese falsche Zuordnung wurde bei der Auswertung zuerst angenommen, dass es am allgemeinen Vorgehen der sprachlichen Analyse liegt. Alle Nutzer bei denen die Zuordnung fehlschlug benutzen Formulierungen wie in Schaubild 5.6. Da aber für das Muster „product“ kein Datenbankeintrag im Zusammenhang mit der Syntax „multiply %1 with %2“ existiert, wurde die Formel nicht korrekt erkannt. Bei Fehlen eines Musters ordnet die sprachliche Analyse die Parameter in Leserichtung den Operatoren zu. Dabei wird der beschriebenen Multiplikation zuerst die Referenz auf die vorher eingegebene Formel $(a + b)$ und anschließend die Variable a als zweiter Parameter zugeordnet. Die sprachliche Analyse findet in der restlichen Eingabe aber nur noch den Differenz Operator und die Variable b als Parameter. Da aber in der Datenbank vermerkt ist, dass eine Differenz immer zwei Parameter benötigt, wird angenommen, dass der Nutzer implizit auf das Ergebnis der vorher gefundenen Multiplikation verweist und diese deshalb der Differenz zugeordnet.

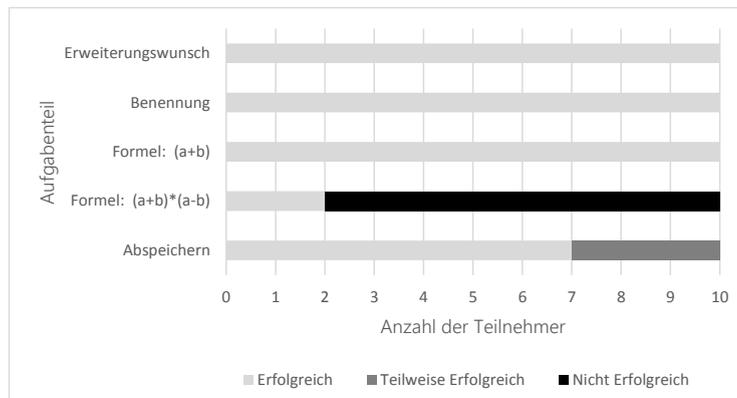


Abbildung 5.5.: Erfolg beim bearbeiten der ersten Teilaufgabe

Der finale Schritt der Mustererweiterung, das Speichern in die Datenbanken, wurde unabhängig von der Vollständigkeit und Korrektheit der Formel durchgeführt. Das Schreiben in die Datenbank erfolgte bei jedem Teilnehmer. Jedoch wurde in drei Fällen keine korrekte Antwort vom System ausgegeben, weshalb diese nur als Teilerfolg gewertet werden können. Nach der Frage zum Speichern konnte es passieren, dass die Bestätigung durch das System nicht dem Nutzer ausgegeben wurde und stattdessen das Dialogsystem die Antwort „Excuse me, I’m having trouble understanding you.“ ausgegeben hat. Dies hat bei den Teilnehmern zu Verwirrung geführt, da an dieser Stelle der Mustererstellungprozess verlassen wurde und es für den Probanden so aussah, als wäre sein gesamter Fortschritt verloren gegangen.

Teilnehmer A
<i>a plus b</i>
System
<i>Your pattern now uses the formula (a+b).</i>
Teilnehmer A
<i>multiply this with a minus b</i>
System
<i>I added (((a+b)*a)-b) to your pattern.</i>

Abbildung 5.6.: Nicht erkannter Syntax „with“

Teilnehmer B
<i>add a to b</i>
System
<i>I added(a+b) to your pattern.</i>
Teilnehmer B
<i>multiply this by a minus b</i>
System
<i>I found 2 possible formulas: 1. ((a+b)*(a-b)) 2. (((a+b)*a)-b) Which one should I use?</i>

Abbildung 5.7.: Erkannter Syntax „by“

Selbstständiges Arbeiten

Im ersten Teil des selbständigen Arbeitens sollte dem System der Satz des Pythagoras beigebracht werden. Jedem Teilnehmer gelang es dabei ohne Probleme den Erweiterungswunsch zu äußern und das neue Muster mit dem selbst ausgewählten Namen zu benennen (siehe Abbildung 5.8). Beim Umsetzen dieser Formel waren die Teilnehmer deutlich engagierter als im ersten Teil der Studie. Viele Probanden haben mehr als einmal von neuem begonnen, ihr Muster JustLingo zu erklären, falls vorherige Versuche nicht erfolgreich waren. Dabei kam es bei zwei Nutzern zu einem kompletten Absturz des Programmes, da sie zum Quadrieren einer Zahl das Schlüsselwort *Square* benutzt haben. Wie sich herausstellte, existierte in der Wörterdatenbank ein Eintrag zu diesem Schlüsselwort, dass auf die Operation *square* in der Befehlsdatenbank verwies. Dieser Datenbankeneintrag existierte allerdings nicht, was beim Auslesen zu einer Ausnahme im System führte. Beide Teilnehmer konnten nach einem Neustart von Excel[®] mit einer anderen Wortwahl ihr Muster erfolgreich vollenden. Die Aufsichtsperson wollte anschließend den fehlerhaften Datenbankeneintrag manuell reparieren. Hat sich dann aber nach mehreren Fehlversuchen dazu entschieden mit Hilfe der Mustergenerierung JustLingo zu erklären, wie die Funktion *Square* zu benutzen ist. Zwei weitere Teilnehmer konnten danach erfolgreich mit dieser neuen Funktion den Satz des Pythagoras zum Wortschatz von JustLingo hinzufügen. Eine Testperson hat nach nur einem Versuch aufgegeben und eine inkorrekte Formel abgespeichert. Das Speichern des Musters wurde wie in der Aufgabe zuvor jedesmal korrekt ausgeführt, jedoch wurde bei einem Teilnehmer wieder eine inkorrekte Antwort ausgegeben.

Im letzten Teil der Evaluation sollten die Teilnehmer JustLingo die Mitternachtsformel beibringen. Dabei sollten sich die Teilnehmer für eine der beiden Möglichkeiten entscheiden:

1. $midnight_1(a; b; c) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$
2. $midnight_2(a; b; c) = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$

Wie bei den Aufgaben zuvor ist es jedem Teilnehmer problemlos gelungen die Mustergenerierung zu starten und dem Muster einen treffenden Namen zu geben (siehe Abbildung 5.9). Beim übergeben der Formel konnten zwei Probanden nur Teilerfolge erzielen

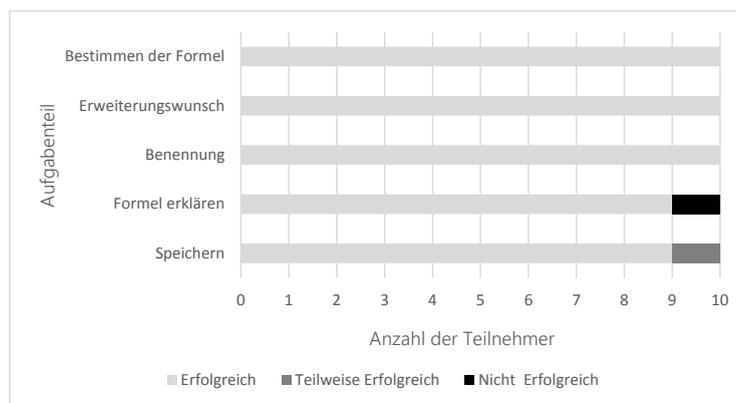


Abbildung 5.8.: Erfolg beim Erklären des Satz des Pythagoras

und ein weiterer konnte überhaupt keine Formel hinzufügen. Dieser Teilnehmer hat anschließend sein Muster auch nicht gespeichert. Die beiden Probanden, die ihre Formel nur teilweise dem System erklären konnten haben ihre Formel zuerst in zwei Teilformeln zerlegt. Dabei konnten sie JustLingo erfolgreich eine Funktion zur Berechnung der Wurzel beibringen. Als Bestätigung teilte ihnen JustLingo mit, dass die neue Formel von nun an so verwendet werden kann: $Diskriminante(a;b;c)$. Bei dieser Bezeichnung handelt es sich nur um ein repräsentatives Beispiel, da jeder Teilnehmer sowohl die Benennung, als auch die benutzten Variablen selber festgelegt hat. Diese Schreibweise hat jedoch mehrere Nutzer dazu verleitet im Anschluss diese genau so, mit Klammern und Semikola, einzugeben. JustLingo kann solche Eingaben jedoch nicht verarbeiten. Stattdessen muss wie bei jeder bisherigen Benutzung eine natürlichsprachliche Eingabe in ganzen Sätzen verwendet werden.

Der eigentliche Grund für die Ausgabe in der Schreibweise der Excel[®]-Formeln war jedoch ein anderer. Der Nutzer sollte darüber informiert werden, in welcher Reihenfolge die neue Funktion ihre Eingabeparameter erwartet. Bei vorherigen Tests kam es bei einigen Formulierungen von Funktionen zu dem Problem, dass die Parameter a und b in vertauschter Reihenfolge erwartet wurden. Dieser Umstand hängt mit der erzeugten Formel zusammen. Alle Parameter werden nach der Erkennung von links nach rechts durch Platzhalter ersetzt. Abhängig davon welcher Parameter zuerst beim Durchsuchen der Formel gefunden wurde, konnte sich die Reihenfolge ändern.

Wie in Abbildung 5.10 zu sehen ist, haben die Teilnehmer der Evaluation unterschiedliche Ansätze zur Erstellung ihrer Formel benutzt. 6 der Teilnehmer haben versucht die Formel direkt zu erklären. 2 der 6 Teilnehmer haben es sogar in nur einem einzigen Satz erfolgreich geschafft die gesamte Mitternachtsformel hinzuzufügen. Von den restlichen Teilnehmern dieser Gruppe hat einer am Ende eine inkorrekte Formel abgespeichert und ein weiterer konnte gar keine Formel erstellen. Insgesamt 3 Teilnehmer haben, wie vorgesehen, ihre Formel in zwei Teilformeln aufgeteilt. Einer dieser Teilnehmer hat bei seiner Formulierung dennoch vergessen die Wurzel in seine Teilformel einzubeziehen, weshalb seine Eingabe in der Auswertung nur als Teilerfolg gewertet wurde. Ein Proband hat seine Mitternachtsformel sogar in 3 Bausteine aufgeteilt und jeden davon erfolgreich JustLingo erklärt.

Nach der Evaluation wurden alle Eingaben, die während der Studie gesammelt wurden betrachtet und kategorisiert. Insgesamt wurden 641 Eingaben an JustLingo gerichtet (siehe Tabelle 5.1). Davon führten 515 dieser Eingaben zu einer korrekten Interpretation des Systems. Die restlichen 126 Eingaben führten zu Fehlinterpretationen (siehe Tabelle 5.2).

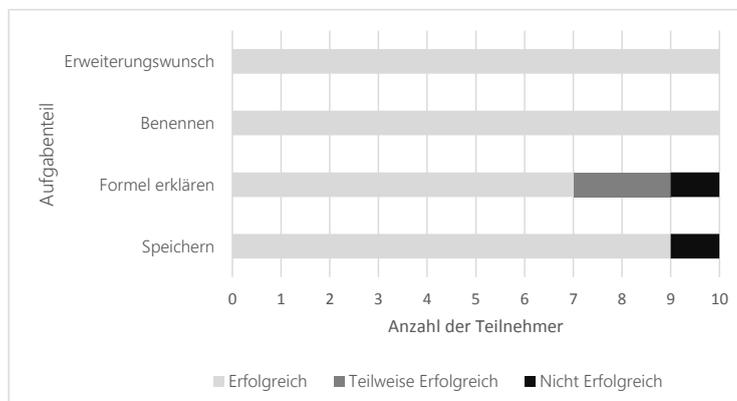


Abbildung 5.9.: Erfolg beim Erklären der Mitternachtsformel

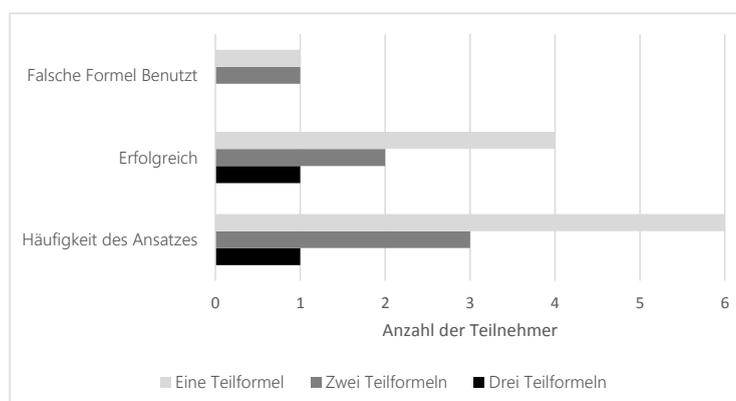


Abbildung 5.10.: Erfolg abhängig von der Anzahl der Benutzten Teilformeln

38 dieser Fehlinterpretationen wurden dabei alleine vom System verursachte. Dazu gehört das falsche Auflösen von Referenzen und das fehlerhafte Zuweisen von Parametern. Darüber hinaus wurde 10 mal nicht die richtige Systemantwort an den Nutzer ausgegeben. Die Benutzung des fehlerhaft in der Datenbank vermerkten Operators *square* führte insgesamt 3 mal zum Systemabsturz. Die in JustLingo implementierte Autokorrektur verursachte bei der Benennung von Mustern drei mal Probleme. Dabei wollte ein Teilnehmer sein Muster mit einem Namen benennen, der sich nur in einem Zeichen von einer bereits vorhandenen Operation unterschied. JustLingo hat dies als Tippfehler interpretiert und daraufhin eine Korrektur vorgenommen. Dadurch wurde der Name aber als Operator erkannt und konnte somit nicht als Name für das neue Muster verwendet werden. Die restlichen 88 Fehlinterpretationen sind durch unvorhergesehene Nutzereingaben ausgelöst worden. Den größten Teil davon macht die Benutzung von „Zeichenschreibweisen“ aus. Damit ist gemeint, dass in der Eingabe mathematische Symbole ($*$, $+$, $-$ und $/$) benutzt wurden. JustLingo wurde jedoch so entwickelt, dass nur Eingaben in Satzform erkannt werden. Einige Benutzer haben vermutlich unabsichtlich deutsche Wörter bei Ihren Eingaben benutzt. Dies führte in einem Fall dazu, dass eine Formel den Namen *ist* statt dem Namen *Binom* durch die Eingabe „The name ist binom“ erhalten hat. Der Teilnehmer hat dies aber nicht bemerkt und hat den Namen so belassen. Bei der Benutzung bestimmter Operationen kam es durch

die Benutzung von falschen Präpositionen bei der Eingabe zu Fehlern. Ein Beispiel hierfür ist der Satz:

„subtract this with 4 times a times c“

Hierbei konnte die Differenz nicht richtig aufgelöst werden, da es *subtract this from...* heißen muss. Ein Teilnehmer hat versucht einen Operator bei seiner Eingabe implizit anzugeben. Das System kann mit dieser Schreibweise allerdings nicht umgehen:

„multiply a to a and b to b.“

Nicht zuletzt haben mehrere Nutzer Formeln ohne Parameter angegeben. Dies wurde durch die Fehlbenutzung der neu hinzugefügten Muster verursacht. Für die Teilnehmer war nach der Systemausgabe zur Benutzung des neuen Musters nicht ersichtlich, dass sie weiterhin die Parameter angeben müssen.

Beschreibung	Wert
Gesamtzahl der Eingaben	641
Korrekt Ausgewertet	515 (80.34%)
Fehlinterpretierte Eingaben	126 (19,66%)

Tabelle 5.1.: Übersicht der ausgeführten Eingaben

Fehlerbeschreibung	Wert
Systemfehler	
Gesamt	38 (5,92%)
Auflösung von Referenzen	5
Fehlerhafte Parameterzuweisung	17
Falsche Antwort ausgegeben	10
Square-Datenbankeneintrag	3
Autokorrektur	3
Eingabefehler	
Gesamt	88 (13,74%)
Zeichenschreibweise	50
Deutsche Eingaben	8
Falsche Präpositionen	6
Systemausgaben ignoriert	16
Implizite Operatoren Nutzung	1
Formeln ohne Parameter	7

Tabelle 5.2.: Auflistung der fehlinterpretierte Eingaben

Weitere Anmerkungen

- Dadurch dass zu Beginn der Formeleingabe alle Referenzen entfernt werden, konnte es passieren, dass Rückbezüge innerhalb einer Eingabe, die an dieser Stelle zulässig gewesen wären, auch entfernt wurden.
- Einmal dauerte die Abarbeitung einer Nutzereingabe mehr als 10 Sekunden. Der Nutzer hat daraufhin eine weitere Eingabe getätigt, was dazu führte, dass die erste Eingabe verworfen wurde.

- Für viele Teilnehmer war nicht klar, dass sie die einzelnen Bestandteile ihres Musters jederzeit überschreiben können. Viele haben Ihre falschen Muster gespeichert um neu Anfangen zu können.
- Wenn ein vollständiges Muster vorliegt befindet sich das Dialogsystem in einer Schleife, da versucht wird vom Benutzer entweder die Erlaubnis zum Speichern zu erhalten, oder ein Änderungswunsch zu erhalten. Diese Schleife kann jederzeit durch eine Eingabe des Nutzers verlassen werden, indem er zum Beispiel seine Formel editiert, oder den Namen verändert. Die meisten Teilnehmer waren sich dessen nicht bewusst und fanden sich augenscheinlich in einer Endlosschleife gefangen.

Auffälligkeiten

Die zur Einführung ausgewählte dritte binomische Formel wurde als das am einfachsten umzusetzende Muster eingestuft. Diese Aufgabe hat den Teilnehmern aber am meisten Problemen bereitet.

Bei den selbständigen Aufgaben haben sich viele Teilnehmer an eine Lösung schrittweise herangetastet, indem sie beim Übergeben der Formel viele verschiedene Eingabemöglichkeiten ausprobiert haben. Der einzige Teilnehmer, der sich als Experte in Bezug auf die Nutzung der Tabellenkalkulationssoftware bezeichnet hat, hat die Aufgaben im Vergleich mit den anderen Teilnehmern eher schlecht gelöst. Dahingegen war der einzige Teilnehmer der noch nie mit Excel[®] gearbeitet hat, beim Erklären seiner Muster sehr erfolgreich.

5.3. Erkenntnisse

Die in dieser Studie erzielten Ergebnisse sind insgesamt als sehr befriedigend einzustufen. Die meisten Teilnehmer konnten bis auf die Einführungsaufgabe alle Funktionen JustLingo beibringen. Die Benutzung der Mustergenerierung erschien aber einigen Teilnehmern etwas gewöhnungsbedürftig. Viele hätten sich gewünscht, dass man jede der Eingaben direkt rückgängig machen kann, da dies mühseliges Eingeben von bereits funktionierenden Teilen erspart. Dies widerspräche allerdings der Idee dem System neues Wissen im Gespräch beizubringen. Wenn man einem anderen Menschen etwas erklärt, ist es auch nicht möglich mitten in der Erklärung das eben Gesagte rückgängig zu machen. Viele Nutzer wünschen sich darüber hinaus die Möglichkeit mathematische Funktionen mit den zugehörigen Zeichen an JustLingo zu übergeben. In der aktuellen Implementierung war dies nicht vorgesehen, könnte aber in zukünftigen Arbeiten durchaus umgesetzt werden.

Einige der in dieser Studie entstandenen Fehler sind auf die Benutzung der sprachlichen Analyse zurückzuführen. Diese war in der ursprünglichen Version zwar durchaus auch für das Eingeben komplizierterer Funktionen ausgelegt, wurde aber in der damaligen Evaluation nicht auf die Benutzung ebendieser getestet. Die ursprüngliche Version befasst sich vor allem damit die Qualität der Systemausgaben und das Erkennen von Zusammenhängen in Bezug auf mehrere Sätze zu testen. Dabei kam es nie zu kompliziert zusammengesetzten Excel[®]-Formeln. Für die Evaluation der Mustergenerierung macht es allerdings wenig Sinn nur einfache Funktionen von den Teilnehmern an JustLingo erklären zu lassen, da für diese meist schon ein Befehl existiert. Darüber hinaus hätte dann keine echte Aussage über die Qualität der Mustergenerierung getroffen werden können. Dieser Umstand macht das Ergebnis der Evaluation durchaus überraschend, da zum Beispiel gerade die Excel[®]-Formel für die Mitternachtsformel sehr lang und kompliziert ist.

6. Verwandte Arbeiten

6.1. Natural Language Programming System - NLC

Ende der 70er Jahre entwickelten Ballard und Biermann das System NLC, das es Nutzern erlaubt Tabellen und Matrizen mittels textuellen Eingaben zu bearbeiten [BB79]. NLC erkennt hierbei nur Anfragen, die sich auf mathematische Operationen zur Bearbeitung der Daten beziehen. Die Entwickler mussten bei ihrem Ansatz weitere Einschränkungen vornehmen, damit NLC auf den damaligen Computern schnell und effizient arbeiten konnte. So müssen beispielsweise alle Nutzereingaben im Imperativ stehen, da NLC am Anfang einer Eingabe stets ein Verb erwartet. Das System führt im Gegensatz zu JustLingo auch keinen Dialog mit dem Nutzer: Nach der Auswertung einer Eingabe werden alle benutzten Eingabeparameter sowie alle durch die Ausführung des Befehls veränderten Parameter vom System markiert, damit überprüft werden kann, ob eine Anfrage erfolgreich ausgeführt wurde. JustLingo hingegen informiert den Benutzer textuell über alle Ergebnisse und versucht etwaige Ungenauigkeiten und Fehler in der Nutzereingabe durch Nachfragen zu beheben.

Diese Basisversion des NLC Systems sollte durch eine Erweiterung verbessert werden. Diese Erweiterung sah vor, dem Nutzer die Möglichkeit zu bieten, dem System neue Worte beizubringen [BBH79]. Diese Wortdefinitionen sind nach demselben Prinzip wie in dieser Arbeit aufgebaut: Zuerst muss der Nutzer dem System einen Namen für sein neues Verb mitteilen. Anschließend muss der Nutzer mitteilen aus welchen bereits bekannten Operationen sich das neue Verb zusammensetzt. Diese Erweiterung wurde für NLC geplant, aber nie umgesetzt.

Ogleich das Vorgehen demselben Muster wie in dieser Arbeit entspricht, verfügt NLC nicht über ein Dialogsystem, das dem Nutzer mit gezielten Fragen und hilfreichen Hinweisen beim Erstellen seiner neuen Funktionalität unterstützt. Allerdings war im NLC System geplant es dem Nutzer zu ermöglichen Schleifen mit Worten zu beschreiben. Dies sollte dem Nutzer beispielsweise ermöglichen dem System eine Prozedur zu beschreiben und diese automatisch für alle Spalten der Matrix wiederholen zu lassen. Eine solche Erweiterung zur Programmierung in natürlicher Sprache ist im aktuellen Stand von JustLingo nicht möglich, für zukünftige Arbeiten aber denkbar.

6.2. Pegasus

Pegasus ist ein an der Universität Darmstadt entwickeltes Programm, das natürlichsprachliche Äußerungen von Nutzern in ausführbaren Programmcode übersetzt. Pegasus akzeptiert dabei Eingaben in verschiedenen Sprachen und übersetzt diese automatisch in Programmiercode [KM06]. Dabei ist das Vorgehen in drei Stufen unterteilt. Zuallererst werden die Nutzeräußerungen in die so genannte „*idea notation*“ überführt. Mit Hilfe dieser Notation versucht Pegasus die Eingabe zu formalisieren. Eine „*idea notation*“ besteht aus mehreren „*atomic ideas*“. Beispielsweise assoziiert Pegasus mit dem Wort „Tisch“ ein „festes“ Objekt mit einer „horizontalen Fläche“. In der nächsten Stufe wird den jeweiligen Ideen, unabhängig voneinander, eine Bedeutung zugeordnet. Dann werden die einzelnen entstandenen Bausteine zusammengesetzt und in ausführbaren Java-Code übersetzt.

Auch wenn sich Pegasus und diese Arbeit in ihrem erstrebten Ziel unterscheiden finden sich im methodischen Vorgehen Ähnlichkeiten. Die Eingaben werden in Sätze unterteilt, welche unabhängig voneinander interpretiert werden. Anschließend wird wie bei Pegasus zielgerichtet nach Zusammenhängen gesucht, die zur Komplettierung eines Musters nötig sind. Wenn alle nötigen Informationen vorhanden sind, wird eine Formel erzeugt, die von der Tabellenkalkulationssoftware interpretiert werden kann.

Der Unterschied zwischen den beiden Systemen besteht vor allem in der Umsetzung. So wurde bei Pegasus im Gegensatz zu dieser Arbeit kein Dialogsystem verwendet. Pegasus übersetzt die Nutzereingabe stattdessen direkt in fertigen Quelltext. Zum besseren Verständnis werden zu den erzeugten Programmblöcken als Kommentar die Eingabe, die diese erzeugt haben, hinzugefügt. So kann der Nutzer anschließend selbständig die Übersetzung nachvollziehen und nach Ungenauigkeiten und Missverständnissen suchen.

6.3. Metafor

Ein häufiges Problem bei der objektorientierten Softwareprogrammierung ist das gute Planen und Erstellen der Klassen und Methoden. Metafor soll diesen Erstellungsprozess unterstützen indem es aus Geschichten und Beschreibungen eines Programmierers automatisch Klassen und Methoden erstellt. [LL05] Die Entwickler von Metafor verfolgten dabei zwei Ziele: Einerseits soll Metafor Anfängern helfen objektorientiert zu programmieren und andererseits bereits erfahrenen Programmierern beim Erstellen von gut geplanten Klassen helfen.

Bei der Interpretation der Nutzereingaben verfolgt Metafor ein bestimmtes Muster. Gefundene Subjekte im Text werden als Klassen interpretiert, dazugehörige Adjektive werden als Eigenschaften einer Klasse realisiert. Zuletzt werden gefundene Verben als Methoden von Klassen erkannt. Hierbei erkennt Metafor auch komplexe Konstrukte. So werden etwa Aufzählungen als Listen erkannt und Parameter automatisch Methoden zugeordnet.

Wenngleich diese Arbeit und Metafor unterschiedliche Ziele verfolgen ist das prinzipielle Vorgehen beim Erreichen dieser vergleichbar. Metafor sucht zuerst nach möglichen Namen für Klassen und erstellt sie. Im nächsten Schritt werden die Eigenschaften eines solchen Objektes aus der Nutzereingabe ermittelt.

Häufige Probleme bei der Verarbeitung von Spracheingaben sind Mehrdeutigkeiten und Unklarheiten. In dieser Arbeit werden sie mittels des vorliegenden Dialogsystems durch Fragen gelöst. Da bei der Entwicklung von Metafor aber ein anderer Ansatz gewählt wurde, der kein Dialogsystem beinhaltet, entfällt diese Nutzerinteraktion. Metafor kann deshalb jede Eingabe nur für sich alleine gestellt interpretieren und bietet keine Möglichkeit an, vom Nutzer zusätzliche Informationen anzufordern. Als größter Unterschied zwischen den beiden Ansätzen kann die geplante Umsetzung identifiziert werden. Metafor erwartet vom Nutzer die Eingaben in Form eines Textes, der alle wichtigen Informationen enthält. Wo-

hingegen in der vorliegenden Arbeit das Gespräch mit dem Nutzer gesucht wird und das Sammeln von Informationen über mehrerer kurze Eingaben erfolgt.

6.4. VoiceTone[®]

Die Telefongesellschaft AT&T hat speziell für Geschäftskunden das System VoiceTone[®] entwickelt. VoiceTone[®] ist ein System das automatisierte Telefongespräche durchführt und dabei auf die Äußerungen des Anrufers reagiert. Mit Hilfe von VoiceTone[®] sollen konventionelle Lösungen, bei denen der Fortschritt im Dialog mit dem System meist über Tastendrucke funktioniert haben, abgelöst werden.

VoiceTone[®] nimmt gesprochene Äußerungen eines Nutzers entgegen und wandelt diese in Text um. Dieser Text wird normalisiert und semantisch klassifiziert. Danach wird die Intentionen, die der Nutzer mit seiner Eingabe vermitteln wollte, aus dem Text extrahiert. Zuletzt wird anhand der gefundenen Information eine passende Antwort ausgegeben. Das System VoiceTone[®] verfügt über ein so genanntes „active learning framework“ [GTHT⁺06], das während des normalen Betriebes teilweise automatisch das System erweitert. Dabei werden mit Hilfe der vorliegenden Daten neue Varianten für Anfragen zu bestimmten Nutzerintentionen gesammelt. Dadurch wird versucht eine möglichst breite Palette an verschiedenen Formulierungen eines Sachverhaltes zu sammeln und dem System verständlich zu machen. So arbeitet das Modul mit einem so genannten „Confident“-Wert, der zu jedem neu erkannten Wort automatisch berechnet wird. Sollte der besagte Wert zu niedrig sein, oder das Modul die Nutzeräußerung nicht verstehen, muss ein Mensch die Markierungen überprüfen und gegebenenfalls verändern.

VoiceTone[®] sucht während der Benutzung ständig nach neuen Formulierungen für bereits bekannte Funktionen. In dieser Arbeit wird im Vergleich dazu nur nach neuen Funktionen gesucht, wenn der Nutzer das System explizit dazu auffordert. Die automatische Suche nach neuen Wörtern findet dabei bei VoiceTone[®] im Hintergrund statt, ohne das der Nutzer dies mitbekommt. Die Benutzer des Systems sind also nur indirekt an der Erweiterung beteiligt. Stattdessen wird der Erweiterungsprozess ständig von einer dritten Person überwacht, die alle vom System gefundenen neuen Formulierungen anschaut und bei Bedarf bearbeitet. Da es sich bei VoiceTone[®] um einen Telefondienst handelt, der zum Beispiel bei Banken zum Einsatz kommt, wird dem Anrufenden absichtlich nicht die Möglichkeit angeboten, Erweiterungen der Funktionalitäten vorzunehmen. Ein Anrufer sollte selbstverständlich nicht in der Lage sein an Kontodaten dritter Personen zu gelangen.

6.5. Mensch-Roboter Interaktion und Kommunikation

Ein weiterer Bereich der Forschung, der sich mit der Kommunikation zwischen Mensch und Maschine beschäftigt ist die Robotik. Schon seit mehreren Jahren forscht das Karlsruher Institut für Technologie im Bereich humanoider Roboter. Das Ziel dieser Forschung ist es den Dialog zwischen Mensch und Roboter so natürlich wie nur möglich zu gestalten.

Die Sprachdomäne, für die der Roboter des Instituts für Anthropomatik ausgelegt wurde, ist dabei wie bei JustLingo stark eingeschränkt. Der Roboter wurde dazu konzipiert in einer Küche Aufgaben zu verstehen und auszuführen [SFG⁺04]. Hierbei kann es sich zum einen um Aufgaben wie das Einräumen einer Spülmaschine handeln. Zum anderen verfügt der Roboter auch über die Möglichkeit dem Nutzer einfache Rezepte für Gerichte zu erklären [GW05]. Im Gegensatz zu JustLingo interpretiert der Roboter jedoch bei Nutzeranfragen nicht nur die sprachlichen Eingaben. Die Erkennung wurde multimodal ausgelegt, sodass über eingebaute Kameras auch Gesten, wie zum Beispiel das Zeigen mit der Hand auf ein bestimmtes Objekt, bei der Bearbeitung einer Nutzeranfrage mit einbezogen werden [HG04]. Diese zusätzlichen Informationsquellen sind für JustLingo, in der aktuellen

Version, nicht von Bedeutung, da das Auswählen bestimmter Bereiche in der Tabellenkalkulationssoftware entweder umschrieben oder mittels eines Mausklicks vom Nutzer an das System übermittelt werden.

In weiterführenden Arbeiten wurde der Roboter um die Möglichkeit erweitert automatisch neue Begriffe zu lernen. Ein Nutzer kann durch Handzeichen und Erklärungen dem Roboter neue, ihm unbekannte, Objekte benennen, sodass der Roboter bei folgenden Äußerungen des Nutzers mit diesen umgehen kann. Dabei konnte bei einem Test mit elf verschiedenen Personen in 32 von 42 Dialogen das unbekannte Objekt erfolgreich durch den Roboter erkannt werden [FGHK06]. Bei dieser Art des Lernens handelt es sich jedoch nicht wie in dieser Arbeit um die Erweiterungen von Funktionalitäten, sondern um das Lernen neuer Begriffe, mit denen in folgenden Dialogen zwischen Mensch und Roboter umgegangen werden kann.

Es sind auch Gemeinsamkeiten zwischen den beiden Systemen vorhanden, so besitzen beide ein Dialogsystem und beide stellen dem Nutzer Fragen. Mit diesen Fragen soll dem System geholfen werden unvollständige Äußerungen des Nutzers zu komplettieren. Und durch hilfreiche Hinweise soll der Nutzer beim Erreichen seiner Intentionen unterstützt werden.

6.6. SmartSynth

SmartSynth ist ein weiteres Computersystem, das natürliche Sprache in ausführbaren Programmcode überführt. Dabei wurde als Anwendungsgebiet die automatisierte Skript-Programmierung für Handys ausgewählt [LGS13]. Diese Skripts sollen dem Nutzer ermöglichen, Abläufe an seinem Handy zu automatisieren. Beispielsweise kann der Nutzer ein Skript beschreiben, das beim Empfang einer Textnachricht, während der Nutzer am Autofahren ist, automatisch eine Antwort an den Absender schickt, in der steht: „I am driving“.

Die Verarbeitung der Nutzereingabe ist innerhalb des SmartSynth Systems in mehrere Stufen unterteilt. Im ersten Schritt wird das Identifizieren der einzelnen Komponenten des Skriptes in den Nutzereingaben vorgenommen. Dabei wird zuerst ein Zuordnungsalgorithmus eingesetzt, der die erhaltenen Beschreibungen zu den verschiedenen Bausteinen eines Skriptes zuordnet. Die Zuordnungen können dabei einerseits auf Literale verweisen, aber auch Schnittstellen anderer Programme können erkannt werden. Sollten in diesem Schritt Teile der Eingabe nicht verstanden worden sein, wird der Nutzer von SmartSynth dazu aufgefordert, diese überarbeitet, neu einzugeben. War eine Erkennung erfolgreich werden anschließend die Bestandteile in eine so genannte „*dataflow relation*“ überführt. Diese verbildlicht den Ablauf des Skriptes. Sollte SmartSynth eine nicht vollständige Definition der „*dataflow relation*“ feststellen, wird zunächst versucht diese automatisch zu komplettieren. Sollten dabei mehrere, als gleich wahrscheinlich angenommene Lösungen gefunden werden, initiiert SmartSynth einen Dialog mit dem Nutzer, um die Mehrdeutigkeiten aufzulösen. Schlussendlich wird die erkannte „*dataflow relation*“ automatisch in die Skriptsprache geparsed.

Der beschriebene Ablauf weist viele Parallelen zur Vorgehensweise der Mustergenerierung auf. Die gefundenen Eingaben werden zunächst auch in eine Mustervorlage abgespeichert um die nötigen Daten zu Erstellung zu sammeln. Sollte es dabei zu Missverständnissen kommen, wird der Nutzer automatisch darauf hingewiesen und darf seine Eingabe verändert neu an das System übergeben. In der Mustergenerierung können beim Zuordnen der Formeln aus den natürlichsprachlichen Eingaben auch Mehrdeutigkeiten entstehen. Diese werden dann auch dem Nutzer mitgeteilt und dieser muss eine Entscheidung treffen. Bei einem vollständig vorliegenden Muster werden am Ende der Mustererstellung automatisch

Datenbankeneinträge erstellt. Als Abgrenzung zwischen den beiden Ansätzen ist vor allem das Anwendungsgebiet zu sehen. So versucht SmartSynth Skripte zu erstellen, die anschließend vom benutzten Handy verwendet werden. Dahingegen wird in der vorliegenden Arbeit eine automatisierte Erweiterung des Dialogsystems selbst verfolgt.

6.7. Mercury Flight Reservation System

Das Flugbuchungssystem Mercury ist ein telefonisch erreichbarer Dienst, der Nutzern beim Organisieren von Reisen mit dem Flugzeug unterstützen soll [SP00]. Dazu bietet es dem Nutzer an, seine Reisewünsche in natürlicher Sprache an das System zu äußern, dass daraufhin nicht nur Antworten gibt, sondern auch Rückfragen stellt. Mercury verfolgt dabei einen zielorientierten Ansatz und fragt den Benutzer schrittweise nach den nötigen Informationen. Dabei handelt es sich um Ziel- und Startflughäfen, sowie Abreise- oder gewünschte Ankunftszeit. Das System kann dem Nutzer auch in Echtzeit den Preis seines Flugplanes nennen und anschließend alle Informationen mittels einer E-Mail zuschicken.

Um zwischenspeichern, welche Informationen Mercury zur Komplettierung der aktuellen Nutzeranfrage noch benötigt wurde ein Zustandsdiagramm verwendet. Jeder Zustand besitzt vier Variablen, die gesetzt sind, wenn der Nutzer die passende Information an das Dialogsystem weitergegeben hat. Mercury hilft dem Nutzer nun nach und nach durch gezielte Nachfragen alle nötigen Daten zu nennen. Eine Anfrage gilt als abgeschlossen, wenn das Dialogsystem sich in einem Zustand befindet, bei dem alle Variablen gesetzt sind.

Das System Mercury wurde als verwandte Arbeit ausgewählt, da diese Art des Sammelns von Informationen mit dem dieser Arbeit viele Parallelen aufweist. Wenn der Nutzer eine Anfrage zur Mustererweiterung an JustLingo stellt, wird automatisch eine Variable angelegt, die den Fortschritt des Erstellungsprozesses speichert. Danach werden nach und nach alle noch benötigten Informationen über Nachfragen und hilfreiche Tipps vom Nutzer eingeholt. Erst wenn das Muster vollständig ist, werden dem Nutzer noch einmal alle Informationen, die aus seinen Äußerungen extrahiert werden zur Überprüfung angezeigt. Nun kann sich der Nutzer entscheiden ob er dies so übernehmen möchte, oder ob das System den Fortschritt verwerfen soll.

Als Abgrenzung zwischen den beiden Ansätzen ist vor allem der Umfang der erkannten Eingaben zu nennen, sowie die Komplexität der Informationen, die aus der Nutzereingabe erkannt werden sollen. Bei Mercury handelt es sich hierbei nur um Uhrzeiten, Namen von Fluggesellschaften, ein Datum und um Namen von Flughäfen. Bei dieser Arbeit wiederum sollen mathematische Formeln erkannt werden und zu völlig neuen, dem System bis dahin unbekannt Funktionen zusammengesetzt werden.

7. Zusammenfassung

Diese Arbeit beschäftigt sich mit der Erweiterung des Dialogsystems JustLingo. JustLingo erlaubt es die von Microsoft[®] entwickelte Tabellenkalkulationssoftware Excel[®] mit natürlicher Sprache zu bedienen. Dazu werden natürlichsprachliche Eingaben in die von Excel[®] benutzte Formelsprache übersetzt. Die erzeugten Formeln können vom Benutzer für Berechnungen in der geöffneten Tabelle verwendet werden. Motiviert durch die schwere Erweiterbarkeit JustLingos wurde in dieser Arbeit eine Ergänzung vorgenommen, die das automatische Erstellen neuer Formeln während des Betriebes zulässt. Die Erweiterbarkeit wurde dabei über das bereits bestehende Dialogsystem realisiert. Dabei erklärt der Nutzer dem System im Dialog, welche Eigenschaften eine neue Funktionalität auszeichnet.

JustLingo verfügt über eine eigene Darstellung von Operationen, anhand derer die Erkennung der natürlichsprachlichen Äußerungen erfolgt. Diese Darstellung wird als ein Muster bezeichnet. Ein Muster besteht dabei aus drei Bestandteilen: Einem eindeutigen Namen, einer Excel[®]-Funktion und Regeln, die angeben, wie das Muster in natürlicher Sprache verwendet werden kann. Bei der Erzeugung der neuen Muster innerhalb dieser Arbeit mussten alle diese Bestandteile automatisch vom System erzeugt werden. Dabei wurde das Modul zur Sammlung der Informationen eines Musters in zwei logische Module aufgeteilt. Das erste Modul wurde auf die Identifizierung des Namens spezialisiert und das zweite Modul auf das Erkennen mathematischer Formeln. Um die Benutzung der Mustergenerierung möglichst intuitiv zu machen, wird die Erstellung der Anwendungsregeln automatisch vom System übernommen. Sollte das Muster komplett sein, erzeugt das System automatisch die nötigen Datenbankeneinträge.

Das Lernen der neuen Muster funktioniert dabei über Erklärungen des Nutzers. Das System unterstützt den Nutzer dabei durch Hinweise und Fragen, solange bis entweder der Prozess durch den Nutzer abgebrochen, oder erfolgreich ein Muster erstellt wurde. Dieser Ansatz des Lernens soll für den Nutzer möglichst menschlich erscheinen, da er so gestaltet ist, als würde er einem Bekannten etwas erklären. Das Dialogsystem wird den Nutzer immer sofort darauf hinweisen, wenn es etwas nicht verstanden hat. Die für die Dialogentscheidungen zuständige Einheit wurde dabei in ein separates Modul implementiert. Dieses verfügt über mehrere Zustände, die alle auf einen Teil der Mustererweiterung spezialisiert sind. Dazu gehört ein genereller Modus, der den Ausgangspunkt bildet. Von diesem Modus kann ein Wechsel zu drei weiteren Zuständen stattfinden: Ein auf die Namenssuche spezialisierter Zustand, einer der den Nutzer beim Eingeben von Formeln unterstützt und ein Zustand, der gewählt wird, sobald das Muster gespeichert werden kann.

Um eine abschließende Aussage über die Qualität der implementierten Mustergenerierung treffen zu können wurde im Rahmen der Evaluation eine Studie durchgeführt. Dabei wurde festgestellt, dass die Erweiterung insgesamt gut funktioniert. Die Teilnehmer hatten keine Probleme mit dem Starten der Mustergenerierung sowie dem Hinzufügen eines Namens zum neuen Muster. Beim Beschreiben der Formel kam es teilweise zu Fehlinterpretationen der Nutzereingabe. Diese wurden einerseits durch das System selbst und andererseits durch unerwartete Nutzereingaben, für die das System nicht ausgelegt ist, verursacht. Insgesamt wurden 641 Nutzereingaben während der Evaluation ausgeführt. Davon führten 515 zu einer korrekten Reaktion des Systems. Von den 126 aufgetretenen Fehlern wurden 38 durch das System verursacht. Dazu gehören falsch interpretierte Formeln, falsche Systemantworten und 3 Totalabstürze, die durch einen fehlerhaften Datenbankeneintrag verursacht wurden. Weitere 88 Fehler wurden durch unerwartete Eingaben ausgelöst. Dabei kam es vor, dass Nutzer sich vertippten, deutsche Eingaben an das System richteten oder versucht haben ihre Formeln mit mathematischen Zeichen anzugeben. Die Darstellung über Zeichen ist aber in der aktuellen Version von JustLingo nicht möglich.

Diese Arbeit zeigt, dass es möglich ist für bereits bestehende Dialogsystem Erweiterungen zum selbständigen lernen zu entwickeln. Auch wenn in dieser Arbeit nur Erweiterungen für eine begrenzte Sprachdomäne möglich sind motiviert sie den Versuch selbstlernende Dialogsystem für komplexere Anwendungsgebiete zu entwickeln. Im Ausblick werden im Folgenden einige weiterführende Ideen und laufende Projekte am System JustLingo beschrieben (siehe Kapitel 8).

8. Ausblick

Parallel zu dieser Erweiterung finden zwei weitere Arbeiten an dem Dialogsystem JustLingo statt. Eines der Projekte setzt bei der Interaktion zwischen Nutzer und JustLingo an. In der aktuellen Version unterstützt JustLingo hauptsächlich textuelle Eingaben. Die während der ersten Version von JustLingo eingebaute Sprachsteuerung erfüllt nur sehr bedingt ihren Zweck. Sie weist große Mängel vor allem beim Erkennen von Spalten, Zellen und Zeilenangaben auf. Gerade aber eine Sprachsteuerung würde Vorteile gegenüber textuellen Eingaben bringen. So würden zum Beispiel auftretende Probleme durch Tippfehler entfallen. Darüber hinaus können vor allem längere Eingaben schneller an das Dialogsystem vorgelesen, als getippt werden. Aus diesem Grund wurde eine grundsätzliche Neuentwicklung der Sprachsteuerung als Bachelorarbeit ausgeschrieben. Durch diese Arbeit sollen die bestehenden Mängel behoben werden. Darüber hinaus wurden aber auch Neuerungen vorgenommen. So kann JustLingo dem Nutzer etwa die erzeugten Ausgaben vorlesen.

Die zweite Erweiterung von JustLingo beschäftigt sich mit der Erstellung eines Lernsystems. Da JustLingo mit natürlicher Sprache arbeitet, kommt es bei den Nutzereingaben sehr häufig zu Mehrdeutigkeiten. Vor allem beim benutzen von komplexeren mathematischen Formeln können bei einer Nutzereingabe schnell eine zweistellige Anzahl an Interpretationsmöglichkeiten gefunden werden. Beim Auftreten eines solchen Falles versucht JustLingo dem Nutzer die verschiedenen Möglichkeiten mittels einer Auswahlfrage auszugeben. Der Nutzer kann sich dann für eine dieser Formeln entscheiden. Bei der Evaluation der Arbeit von Voigt ist dabei aufgefallen, dass Benutzer immer denselben Eingabestil benutzen und bei Auswahlfragen sehr oft dieselbe Entscheidung treffen. Das aktuelle in der Implementierung befindliche Lernsystem soll solche häufig auftretende Entscheidungen, bei der der Nutzer immer dieselben Antworten fällt, erkennen und die gewählte Antwort merken. Sollte dann bei zukünftigen Eingaben wieder ein solcher Fall auftreten übernimmt das System automatisch die Entscheidung für den Nutzer, da es sich das Verhalten gemerkt hat. Zusätzlich soll das Lernsystem mit der Mustergenerierung eng verbunden werden. Das Lernsystem soll bei Erkennung einer unbekanntenen Eingabe dem Nutzer automatisch vorschlagen dieses zum Wortschatz von JustLingo hinzuzufügen. Dazu soll automatisch in den Zustand der Mustergenerierung gewechselt und der Erstellungsprozess gestartet werden.

Bei einer durchgeführten Studie zur Erhebung von Nutzerbedürfnissen beim Arbeiten mit der Tabellenkalkulationssoftware Excel[®] wurde zusätzlich ein Bedarf zur Hilfe beim Erstellen von Diagrammen festgestellt (siehe [Voi]). Viele der Teilnehmer haben angegeben, dass sich beim Erstellen von Diagrammen Hilfe wünschen würden. Eine Erweiterung von JustLingo zum Erstellen von Diagrammen würde den aktuellen Funktionsumfang nahtlos ergänzen. Bis jetzt besitzt JustLingo die Fähigkeiten den Nutzer beim Auswerten von Daten zu unterstützen. Mit einer dialoggestützten Diagrammerstellung könnten diese Daten dann anschließend visualisiert werden. Diese Art der Erweiterung würde das Dialogsystem vor neue Herausforderungen stellen, denn das exakte Beschreiben von Diagrammen in natürlicher Sprache ist sehr komplex.

Eine weitere Überlegung, die während des Planens dieser Arbeit auftrat, ist das Erzeugen von so genannten Makros in Excel[®]. Über die Makros können in der Tabellenkalkulationssoftware kleinere Programme erstellt werden, die bestimmte Berechnungsabläufe automatisch ausführen. Zur Programmierung dieser Makros sind allerdings Kenntnisse im Umgang mit Visual Basic nötig. Deshalb wäre eine Erweiterung von JustLingo, die aus natürlicher Sprache einfache Makros erstellen kann wünschenswert.

Literaturverzeichnis

- [BB79] Bruce W. Ballard und Alan W. Biermann: *Programming in Natural Language: „NLC“ As a Prototype*. In: *Proceedings of the 1979 Annual Conference*, ACM '79, Seiten 228–237, New York, NY, USA, 1979. ACM, ISBN 0–89791–008–7. <http://doi.acm.org/10.1145/800177.810072>.
- [BBH79] Alan W. Biermann, Bruce W. Ballard und Anne M. Holler: *A System for Natural Language Computation*. SIGLASH Newsl., 12(1):6–16, März 1979, ISSN 0036-147X. <http://doi.acm.org/10.1145/1041361.1041362>.
- [Cor14] Microsoft Corporation: *National Language Support (NLS) API Reference*, 2014. <http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx>, Besucht 5. Dezember 2014.
- [FGHK06] Christian Fügen, Petra Gieselmann, Hartwig Holzapfel und Florian Kraft: *Natural Human Robot Communication*. In: *2nd Intl. Workshop on Human-Centered Robotic Systems, Munich, Germany*, 2006.
- [Fow06] Martin Fowler: *Passive View*, 2006. <http://martinfowler.com/eaDev/PassiveScreen.html>, Besucht 1. Dezember 2014.
- [KIT13] Karlsruher Institut für Technologie Institut für Programmstrukturen und Datenorganisation (IPD): *JustLingo*, 2013. <http://ps.ipd.kit.edu/JustLingo.php>, Besucht 1. Dezember 2014.
- [GTHT⁺06] N. Gupta, G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi und M. Gilbert: *The AT&T spoken language understanding system*. Audio, Speech, and Language Processing, IEEE Transactions on, 14(1):213–222, Jan 2006, ISSN 1558-7916.
- [GW05] Petra Gieselmann und Alex Waibel: *What makes human-robot dialogues struggle*. In: *Proceedings 9th workshop on the semantics and pragmatics of dialogue (DIALOR)*. Nancy, France, 2005.
- [HG04] H. Holzapfel und P. Gieselmann: *A way out of dead end situations in dialogue systems for human-robot interaction*. In: *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, Band 1, Seiten 184–195 Vol. 1, Nov 2004.
- [Kü12] Andreas Kühnel: *Visual C# 2012: Das umfassende Handbuch: Spracheinführung, Objektorientierung, Programmier Techniken*. Galileo Computing, 2012, ISBN 3836219972.
- [KM06] Roman Knöll und Mira Mezini: *Pegasus: First Steps Toward a Naturalistic Programming Language*. In: *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '06, Seiten 542–559, New York, NY, USA, 2006. ACM, ISBN 1–59593–491–X. <http://doi.acm.org/10.1145/1176617.1176628>.

- [LGS13] Vu Le, Sumit Gulwani und Zhendong Su: *SmartSynth: Synthesizing Smartphone Automation Scripts from Natural Language*. In: *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '13*, Seiten 193–206, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-1672-9. <http://doi.acm.org/10.1145/2462456.2464443>.
- [LL05] Hugo Liu und Henry Lieberman: *Metafor: Visualizing Stories As Code*. In: *Proceedings of the 10th International Conference on Intelligent User Interfaces, IUI '05*, Seiten 305–307, New York, NY, USA, 2005. ACM, ISBN 1-58113-894-6. <http://doi.acm.org/10.1145/1040830.1040908>.
- [NFK⁺00] S. Narayanan, G. Di Fabbrizio, C. Kamm, J. Hubbell, B. Buntschuh, P. Ruscitti und J. Wright: *Effects of dialog initiative and multimodal presentation strategies on large directory information access*. In: *In ICSLP*, Seiten 16–20, 2000.
- [Ort14] Charles L. Ortiz: *The Road to Natural Conversational Speech Interfaces*. Internet Computing, IEEE, 18(2):74–78, Mar 2014, ISSN 1089-7801.
- [SFG⁺04] R. Stiefelhagen, C. Fugen, R. Gieselmann, H. Holzapfel, K. Nickel und A. Waibel: *Natural human-robot interaction using speech, head pose and gestures*. In: *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, Band 3, Seiten 2422–2427 vol.3, Sept 2004.
- [SP00] Stephanie Seneff und Joseph Polifroni: *Dialogue Management in the Mercury Flight Reservation System*. In: *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems – Volume 3, ANLP/NAACL-ConvSyst '00*, Seiten 11–16, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. <http://dx.doi.org/10.3115/1117562.1117565>.
- [Voi] Phillip Voigt: *Studie zu Nutzerbedürfnissen in Microsoft Excel*. <http://mphsoftware.de/justlingo/study/de-DE/>, Besucht 13. Dezember 2014.
- [Voi14] Philipp Voigt: *Datenverarbeitung in Tabellenkalkulationen durch das natürlichsprachliche Dialogsystem JustLingo*. Bachelorarbeit, Karlsruher Institut für Technologie, 2014.
- [Wal00] Douglas Walton: *The Place of Dialogue Theory in Logic, Computer Science and Communication Studies*. *Synthese*, 123(3):327–346, 2000, ISSN 0039-7857. <http://dx.doi.org/10.1023/A%3A1005237527730>.

Abbildungsverzeichnis

2.1. Systemaufbau des JustLingo-Prototyps (<i>Quelle:</i> [Voi14])	7
3.1. Aufbau des Addieren-Musters	11
3.2. Übersicht des Systemaufbaus	12
4.1. Benutzeroberfläche des Dialogsystems	16
4.2. Model-View-Presenter	17
4.3. Verarbeitungsablauf der Mustererweiterung-Einheit	18
4.4. Annotation der Eingabe: „It uses the sum of a and b.“	19
4.5. Behandlung von Artikeln in der Vorverarbeitung	20
4.6. Ablauf der Kontextinterpretation	21
4.7. Aufbau der Musterinterpretation	24
4.8. Übersicht der Namensinterpretation	25
4.9. Übersicht der Formelinterpretation	26
4.10. Übersicht der Dialogmanagement-Einheit	27
4.11. Auszug des Antworterzeugungs-Klassendiagrammes	28
4.12. Ablauf des generellen Dialogmanagements	30
4.13. Ablauf des Namensspezifischen Dialogmanagement	31
4.14. Ablauf des Formelspezifischen Dialogmanagement	32
4.15. Ablauf des Speichern/Sonstiges Dialogmanagement	33
4.16. Beispieldialog mit Kontextspeicherung	35
4.17. Beispiel der Formelauflösung	39
4.18. Operatorenvererbung in der zugehörigen Diskriminantenfunktion	40
4.19. Operatorenvererbung in der Mitternachtsformel	40
5.1. Satz des Pythagoras	42
5.2. Selbsteinschätzung der Erfahrung	43
5.3. Häufigkeit der Excel [®] Nutzung	44
5.4. Erfahrungslevel im Umgang mit Excel [®] gemessen in Jahren	44
5.5. Erfolg beim bearbeiten der ersten Teilaufgabe	45
5.6. Nicht erkannter Syntax „with“	46
5.7. Erkannter Syntax „by“	46
5.8. Erfolg beim Erklären des Satz des Pythagoras	47
5.9. Erfolg beim Erklären der Mitternachtsformel	48
5.10. Erfolg abhängig von der Anzahl der Benutzten Teilformeln	48

Tabellenverzeichnis

3.1. Beispieldialog einer Mustererstellung	10
4.1. Markierungen bei „Sum up var1, v2 and c and divide it by var1 and v2.“ . .	20
4.2. Mustervorlage der dritten binomischen Formel	23
4.3. Beispiel einer nicht eindeutig interpretierbaren Formel	26
4.4. Inhalt der Namensfrage	29
4.5. Phrasen der Namensfrage	29
4.6. Auszug eines kontextunabhängigen Dialogverlaufes	33
4.7. Datenbankeneintrag der Wörter-Datenbank	36
4.8. Datenbankeneintrag der Befehls-Datenbank	36
4.9. Funktionsdarstellung der Mitternachtsformel in der Mustergenerierung . . .	38
5.1. Übersicht der ausgeführten Eingaben	49
5.2. Auflistung der fehlinterpretierte Eingaben	49

9. Anhang

A. Fragebogen der Evaluation zur Bachelorarbeit

Die Teilnahme an der Studie ist freiwillig und anonym. Um die Anonymität auch bei handschriftlichen Antworten zu gewährleisten können Sie Ihr Schriftbild verändern. Die Aufgaben, die Sie in dieser Studie bekommen, sollen am PC gelöst werden. Zu Auswertungszwecken werden daher Ihre Eingaben und die zugehörigen Ausgaben des Systems anonym aufgezeichnet. Zur Zuordnung der Aufzeichnung zu diesem Fragebogen benötigen wir das aktuelle Datum und die Uhrzeit.

Bitte tragen Sie Datum und Uhrzeit ein:

I. Allgemeine Fragen

Zunächst benötigen wir eine Angaben zu Ihrer Person, sowie zu Ihrer persönlichen Erfahrung im Umgang mit der Tabellenkalkulationssoftware Microsoft® Excel®.

1. Bitte nennen Sie uns Ihren aktuellen Berufsstand.

- Schüler Student Berufstätig Nicht berufstätig

2. Welchen Beruf üben Sie aus?/ Welches Studienfach belegen sie? *(falls zutreffend)*

3. Bitte nennen Sie uns Ihre Muttersprache.

- Englisch Deutsch Andere

4. Seit wie vielen Jahren benutzen Sie Microsoft® Excel®?

5. Wie häufig arbeiten Sie mit Microsoft® Excel®?

- Jeden Tag Jede Woche Jeden Monat Seltener Nie

6. In welchem Umfeld benutzen Sie Microsoft® Excel®? *(Mehrfachnennungen möglich)*

- Privat Arbeit Ausbildung Sonstige

7. Schätzen Sie bitte Ihren persönlichen Kenntnisstand im Umgang mit Microsoft® Excel® ein.

- Anfänger Fortgeschrittener Experte

8. Bitte teilen Sie uns mit, ob Sie schon einmal mit JustLingo gearbeitet haben.

- Ja Nein Weis nicht

I. Einführung in den Umgang mit der automatischen Mustergenerierung

Bei dem bereits erwähnten System JustLingo handelt es sich um ein natürlichsprachliches Dialogsystem, das Sie bei der Arbeit mit Excel unterstützt. JustLingo interpretiert Ihre **englischen** Eingaben und erzeugt daraus **mathematische Formeln** für Microsoft® Excel®. Bisher hat JustLingo nur Formeln verstanden, die vorher von den Entwicklern eingebaut wurden. Um die Benutzung an die individuellen Bedürfnisse der Nutzer anzupassen, wurde eine Erweiterung vorgenommen, die das **automatisierte hinzufügen neuer Formeln** erlaubt. In der vorliegenden Studie soll diese Erweiterung evaluiert werden.

Schrittweises Vorgehen zum Erstellen einer Benutzerformel

Im folgenden Abschnitt werden Sie Schritt für Schritt angeleitet, wie Sie JustLingo eine neue Formel hinzufügen können. Das Ziel dieser Einführung ist es JustLingo die dritte binomische Formel beizubringen, die Sie anschließend mit dem Namen „Binom“ benutzen können. Sie dürfen uns bei jedem Teilschritt gerne Anmerkungen zur Benutzung, oder hilfreiche Hinweise in den vorgesehenen Zeilen mitteilen.

Dritte binomische Formel: $Binom(a; b) = (a + b) * (a - b)$

1. Wunsch zur Erweiterung mitteilen

Zunächst müssen Sie JustLingo mitteilen, dass Sie eine neue Formel erstellen möchten. Dazu können Sie Sätze wie „JustLingo erstell mir bitte ein neues Muster“ (en: Pattern) oder „Ich möchte eine neue Formel hinzufügen“. Wenn Ihre Anfrage erfolgreich war, wird JustLingo Ihnen sofort eine Rückmeldung geben. Kreuzen Sie nun bitte an, ob dies erfolgreich war.

Ja

Nein

Anmerkungen:

2. Benennung der Formel

Da Sie nun den Erstellungsprozess gestartet haben, sollten Sie dem System zunächst den Namen Ihrer neuen Formel nennen. Dazu können Sie Sätze wie „Der Name ist Binom“ oder „JustLingo nenne meine Formel Binom“ benutzen. Geben Sie nun Ihre Formulierung ein und benennen Sie Ihre neue Formel mit dem Namen **Binom**. Kreuzen Sie anschließend an, ob Ihre Anfrage erfolgreich erkannt wurde.

Ja

Nein

Anmerkungen:

3. Formel mitteilen

Da Ihre neue Funktion nun einen Namen besitzt müssen Sie JustLingo nur noch mitteilen wie die Berechnungsvorschriften aussehen. Da im Moment noch nicht bekannt ist, welche Werte bei der späteren Benutzung der Formel verwendet werden, sollten Sie Variablen als Parameter benutzen. Das System wird automatisch die **Kleinbuchstaben von a-z als Variablen erkennen**. In dieser Einführung wird die Formel in Leserichtung schrittweise übergeben.

Diktieren Sie nun den ersten Teil Ihrer Formel „**a + b**“. Benutzen Sie dazu ganze Sätze als würden Sie es einem Bekannten erklären. Ein Beispiel wäre: „addiere a und b“ oder „berechne die Summe von a und b“.
JustLingo wird Ihnen bei Erfolg mitteilen, dass die Formel erkannt wurde und sie hinzugefügt wurde.
Bei Misserfolg wird das System Ihnen automatisch Hinweise geben, die Sie unterstützen sollen. Kreuzen Sie anschließend an, ob Ihre Eingabe erkannt wurde.

 Ja

 Nein

Anmerkungen:

1. Formel erweitern

Wenn Ihre Eingabe erkannt wurde, wird JustLingo Sie sicher gefragt haben ob Sie Ihre Formel abspeichern möchten. Ignorieren Sie diese Frage vorerst und Teilen Sie JustLingo den Rest Ihrer Formel mit.

Dabei sollten Sie darauf achten, dass Sie sich in Ihrer Eingabe auf die Vorherige beziehen.

Multiplizieren Sie nun zu Ihrer bisherigen Formel mit „**(a – b)**“. Eine Beispieleingabe ist „multipliziere dies mit a minus b“. Kreuzen Sie anschließend an, ob Ihre Formel erfolgreich erkannt wurde.

 Ja

 Nein

Anmerkungen:

2. Abspeichern einer Formel

Da Sie nun Ihre Formel vervollständigt haben, können Sie diese nun zum Wortschatz hinzufügen lassen. Beantworten Sie dazu die Frage zum Speichern bejahend. Und Tragen Sie bitte die Excel®-Formel, die JustLingo für Ihre neue Funktion benutzt in die untenstehende Tabelle ein.

Excel®-Formel Ihrer neuen Funktion:

Anmerkungen:

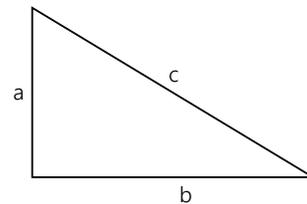
I. Eigenständige Aufgaben

Da Sie nun schon Erfahrung im Umgang mit JustLingo gesammelt haben sollen Sie dem System abschließend Eigenständig zwei Funktionen beibringen. Dabei sollen Sie sich selber Namen sowie Funktionen überlegen, die Sie Benutzen wollen. Bitte notieren Sie Ihr Vorgehen an den jeweiligen Stellen. Sie dürfen jeder Zeit Hinweise und Anmerkungen zu den Aufgaben aufschreiben.

Geometrische Aufgabe

In dieser Aufgabe sollen Sie eine Formel zur Berechnung der Länge der Hypotenuse (Schaubild: c) dieses rechtwinkligen Dreiecks angeben. Bitte geben Sie jetzt die mathematische Formel, die Sie zur Berechnung benutzen wollen an:

c = _____



Bringen Sie nun JustLingo die von Ihnen definierte Formel bei und halten Sie das Ergebnis in den folgenden Zeilen fest:

Name Ihrer neuen Funktion:

Excel®-Formel Ihrer neuen Funktion:

Konnten Sie JustLingo die Formel erfolgreich beibringen?

Ja

Nein

Falls Ihre Antwort Nein lautet, erläutern Sie bitte kurz auf an welcher Stelle die Erstellung fehlgeschlagen ist.

Anmerkungen:

Abschließend : „Mitternachtsformel“

Zum Schluss sollen Sie JustLingo eine komplexere Formel beibringen. Aber zunächst geben wir Ihnen einen Hinweis zum Arbeiten mit JustLingo. Viele kompliziertere Formeln lassen sich einfach in mehrere Stücke unterteilen. Nehmen Sie beispielsweise an sie würden folgendes realisieren wollen: $f(x) = \frac{x+2}{\sqrt{x^2+25}}$.

Hier empfiehlt sich eine Unterteilung in folgende Schritte:

- Zuerst erstellen wir „fquotient“ mit der Formel: $fquotient(x) = \sqrt{x^2 + 25}$
- Danach Erstellen wir eine zweite Formel „fcomplete“ : $fcomplete(x) = \frac{x+2}{fquotient(x)}$

Ebenso können Sie auch beim Erstellen mit JustLingo verfahren, denn neben elementaren Grundoperationen können Sie auch früher erzeugte Formeln mit einbeziehen. Bitte bringen Sie JustLingo nun die „Mitternachtsformel“ zur Lösung von quadratischen Gleichungen bei.

Zur Vereinfachung entscheiden Sie sich für einen der beiden Fälle:

$$1. \quad \mathit{midnight}(a; b; c) = \frac{-b + \sqrt{b^2 - 4*a*c}}{2*a}$$

$$2. \quad \mathit{midnight}(a; b; c) = \frac{-b - \sqrt{b^2 - 4*a*c}}{2*a}$$

Bitte tragen Sie hier alle Benutzen Teilformeln, sowie das Endergebnis ein. Sollten Sie kein komplettes oder ein inkorrektes Ergebnis erhalten, schreiben Sie bitte dennoch Ihre Teilformeln und Teilschritte auf.

Benutzte Formeln:

Anmerkungen:

B. Vokabelliste zur Evaluation

Deutsch	Englisch
Muster	Pattern, sample
Formel	Formula
Addieren	Add, plus, sum up
Subtrahieren	subtract, minus
Multiplizieren	Multiply, "times", product
Dividieren	Divide
Potenzieren	Potentiate
Runden	Round
Rückbezüge	It, this, that
Benennen	Call, name
Wurzel	Root
Zählen	Count
Mittelwert	Mean
Cosinus	Cosine
Sinus	Sine
Zelle	Cell
Variable	Variable
Spalte	Column
Reihe	Row