Großer Beleg

SAR-Interaktion mittels KI-konfigurierbarer Adaptionsmodelle

eingereicht von

Konstantin Herold
geboren am 31.07.1998 in Aschaffenburg

Technische Universität Dresden

Fakultät Informatik
Institut für Angewandte Informatik
Lehrstuhl Mensch-Computer-Interaktion

Betreuer:
David Gollasch, M. Sc.

Hochschullehrer:
Prof. Dr. rer. nat. habil. Gerhard Weber

Eingereicht am 20. Februar 2023

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Angewandte Informatik Professur für Mensch-Computer-Interaktion

Aufgabenstellung für den Großen Beleg

Name des Studenten: **Konstantin Herold**
Studiengang: **Diplom Informatik**
Immatrikulationsnummer: **4688610**

Thema:

SAR-Interaktion mittels KI-konfigurierbarer Adaptionsmodelle

SAR Interaction by Means of AI-Configurable Adaptation Models

Zielstellung

Kontext. Entlang der aktuell stark bearbeiteten Forschungsfelder der Künstlichen Intelligenz und der natürlich-sprachlichen Interaktion entwickelt sich der Bereich der Robotik fernab von Industrierobotern in schnellem Tempo weiter. Insbesondere die Bereiche der Sozialen Assistenzroboter (SARs) und Mensch-Roboter-Interaktion bieten allerdings noch zahlreiche offene technische und auch anwendungsbezogene Fragen.

Ein zunehmender Bedarf an SARs erschließt sich im Umfeld von Menschen mit Beeinträchtigungen, Senioren sowie im Altenpflegebereich. SARs könnten hier eingesetzt werden, um die Selbständigkeit und Selbstbestimmtheit der Nutzenden zu fördern. Gleichzeitig können solche Assistenzsysteme dazu beitragen, den personell angespannten Pflegebereich zu entlasten.

Bisherige SARs verfügen noch nicht über einen breiten Funktionsumfang, um verschiedentliche Anwendungsfälle abzudecken und die Entwicklung von Software für SARs ist mangels plattformübergreifender Lösungen teuer. Eine besondere Herausforderung ist außerdem die Umsetzung einer gebrauchstauglichen Mensch-Roboter-Interaktion. Idealerweise wird diese benutzendenzentriert entwickelt und erlaubt eine automatische Adaption an die Anforderungen der Nutzenden, den Anwendungskontext und sich dynamisch ändernde Faktoren.

Ein Ansatz zur Lösung der beschriebenen Probleme kann die Entwicklung einer gemeinsamen Anwendungsarchitektur, basierend auf den Methoden aus der Softwarevariabilität, sein. Diese umschließt neben den implementierten Anwendungsfällen auch die Interaktion und ermöglicht daher die erforderliche Adaptivität.

(Fortsetzung Rückseite)

Fachbetreuer: David Gollasch, M.Sc.
Beginn am: 25.04.2022
Einzureichen am: 12.09.2022 (20 Wochen)

Dresden, 08.04.2022

Prof. Dr. rer. nat. habil. Gerhard Weber
verantwortlicher Hochschullehrer

Technische Universität Dresden
Fakultät Informatik
Institut für Angewandte Informatik
Professur Mensch-Computer-Interaktion

Projektziel. Der Fokus dieser Arbeit soll auf der konzeptionellen Ebene des Adaptionsmechanismus für im Kontext (s. o.) erwähnte SARs liegen. Dafür wurden am Lehrstuhl bereits zahlreiche Themen bearbeitet, welche die Grundlage dafür bilden, wie die Interaktion mit einem SAR bezüglich unterschiedlicher Modalitäten (v. a. Sprache, Gesten, Emotionen) gestaltet werden sollte. Es wurden auch bereits Implementierungsfragen bearbeitet, bspw. wie die Aufgabenausführung eines Roboters variabel gestaltet werden kann und in welcher Reihenfolge Aufgaben ausgeführt werden sollten. Die Grundlage der Anpassbarkeit der Interaktion bildet die Idee, eine Abbildung zwischen den Bedarfen und Präferenzen der Nutzenden und den implementierten Interaktionsmöglichkeiten herzustellen, die auf Konzepten der Softwarevariabilität basiert. Hierbei wird ein sog. Variabilitätsmodell benötigt, welches das Spektrum der Nutzendenanforderungen (Bedarfe und Präferenzen) wie auch die verfügbaren Interaktionsmöglichkeiten in sich vereint. Zur Laufzeit wird dieses Modell konfiguriert, sodass sich aus dieser Konfiguration ableiten lässt, wie die Interaktion mit dem SAR zu gestalten ist. Dieser Konfigurationsprozess trifft eine Auswahl an Interaktionsmöglichkeiten unter Berücksichtigung der Gültigkeit des Modells (Regeln innerhalb des Variabilitätsmodells) einerseits und den vorliegenden Bedarfen und Präferenzen des Nutzenden (Regeln, die durch den Nutzenden induziert werden) andererseits. Der dem Konfigurationsprozess zugrundeliegende Regelsatz basiert u. a. auf bestehenden Richtlinien zur Interaktion sowie auf individuellen Einstellungen, die Nutzende vornehmen können sollen. Dieser Konfigurationsprozess ist im Sinne einer symbolischen KI angelegt. *Ziel dieser Arbeit soll die konzeptionelle Ausarbeitung eines geeigneten Variabilitätsmodells sowie eines zugehörigen Konfigurationsregelsatzes sein.* Zudem soll das entwickelte Modell auf Eignung überprüft werden. Hierfür soll das Modell auf die Interaktion mittels Sprache gefaltet und prototypisch in Form eines Sprachassistenten so umgesetzt werden, dass eine Evaluation mit Nutzenden möglich wird.

Schwerpunkte:

- Einarbeitung in die folgenden Themengebiete inklusive Analyse des aktuellen Forschungsstandes; insbes. vertraut machen mit den bestehenden Arbeiten am Lehrstuhl.
    - Wichtige Engineering-Aspekte: Softwarevariabilität, Software-Produktlinien, Feature-Modelle und Laufzeit-Konfiguration
    - Mensch-Roboter-Interaktion: Soziale Assistenzroboter im Umfeld älterwerdender Menschen (hinsichtlich kognitiver Beeinträchtigungen sind Menschen mit leichter Demenz eine interessante Zielgruppe)
    - Symbolische Künstliche Intelligenz, Regelbasierte Abläufe
    - Bestehende Guidelines: Richtlinien zur Interaktion mit Menschen mit Beeinträchtigungen sowie mit älteren Menschen (insbes. der ISO Guide 71 als Basis)
    - Sprachinteraktion: Entwicklung von Skills für Sprachassistenten (VUIs und CUIs)
- Systematisches und analytisches Zusammentragen von Nutzendenanforderungen und Interaktionsmöglichkeiten sowie Konsolidierung dieser zu einem konfigurierbaren Variabilitätsmodell.
- Entwurf eines für die symbolische KI nötigen Regelsatzes.
- Umsetzung des Variabilitätsmodells und eines Konfigurationsmechanismus inkl. des erarbeiteten Regelsatzes als Softwarebaustein zur Laufzeit-Konfiguration der Interaktion eines Sprachassistenten (Einsatz von MyCroft.ai [Agentenarchitektur] und RASA [VUI und Dialogmanagement] empfohlen).
- Evaluation und Auswertung des erarbeiteten Modells mittels des Prototyps sowie Dokumentation der Ergebnisse in geeigneter, wissenschaftlicher Form

Erklärung

Ich erkläre, dass ich die vorliegende Arbeit mit dem Titel SAR-*Interaktion mittels KI-konfigurierbarer Adaptionsmodelle* selbständig unter Angabe aller Zitate angefertigt und dabei ausschließlich die aufgeführte Literatur und genannten Hilfsmittel verwendet habe.

Dresden, 20. Februar 2023

Konstantin Herold

Abstract

Durch die, in den nächsten Jahren voraussichtlich weiter steigende Zahl an älteren Menschen nimmt die Belastung des Pflegepersonals und der Angehörigen zu. Um hier Abhilfe zu schaffen, eignen sich Soziale Assistenzroboter (SAR), welche zu Beginn der Arbeit vorgestellt und eingeordnet werden. Diese Arbeit zeigt, dass sich altersbedingt jedoch unterschiedliche Einschränkungen bei der Nutzung von SAR ergeben können, durch die die Mensch-Roboter-Interaktion (MRI) beeinträchtigt wird. Nach einer grundlegenden Vorstellung der Software-Variabilität, Feature-Modellen und regelbasierter Künstliche Intelligenz (KI) werden recherchierte Nutzeranforderungen zusammengefasst. Damit der SAR die Interaktion an den Nutzer anpassen kann, wird erarbeitet, dass zunächst Feature-Modelle benötigt werden, die einerseits die Einschränkungen der Benutzer und andererseits die möglichen Interaktionsvarianten, die der SAR anbietet, gespeichert werden können. Um die, für einen spezifischen Nutzer passende Variante der Interaktion zu finden, braucht man sowohl einen Regelsatz, der die beiden zuvor genannten Modelle logisch verbindet, als auch einen Algorithmus, der es ermöglicht, das Feature-Modell der Interaktionen auf Grundlage der eigenen Einschränkungen zu konfigurieren. Auf Grundlage der erarbeiteten Anforderungen konnten in dieser Arbeit sowohl zwei Feature-Modelle, als auch ein Regelsatz abgeleitet werden, welche anschließend in Form eines Prototyps auf Korrektheit überprüft worden.

Inhaltsverzeichnis

1 Motivation

Es wird erwartet, dass sich die Zahl an älteren Menschen in den nächsten Jahren verdoppeln wird. Laut einem Report der United Nations [Uni20], wird die Zahl an Menschen über 65 Jahren von aktuell 727 Millionen auf über 1,5 Milliarden innerhalb der nächsten 30 Jahre steigen. Durch die zunehmende Zahl wird damit gerechnet, dass es zu Platz- und Personalmangel in Altenheimen und anderen Pflegeeinrichtungen kommen kann [MS16]. Bevor hier neue Stellen geschaffen werden, kommt es zu einer höheren Arbeitsbelastung der bisherigen Pflegekräfte. Aus diesem Grund muss an die Entlastung des Personals gedacht werden. Dabei können assistierende Systeme wie beispielsweise Roboter zur Unterstützung eingesetzt werden. So kann durch deren Nutzung sowohl die Arbeit in Pflegeeinrichtungen erleichtert werden, als auch dafür gesorgt werden, dass ältere Menschen länger ohne fremde Hilfe oder Unterstützung in der eigenen Wohnung leben können und somit das Patientenaufkommen in Pflegeeinrichtungen geringer ist. Dabei gibt es Roboter, die bei Aktivitäten des alltäglichen Lebens helfen können, aber auch Soziale Assistenzroboter (SAR). Diese unterstützen die Benutzer auf der einen Seite und interagieren mit ihnen auf der anderen Seite, sodass er als Freund oder Assistent dient [FM05]. Es gibt aktuell eine große Bandbreite an SAR, die für unterschiedliche Anwendungen und Nutzergruppen konzipiert wurden [Hee10].
Das Problem hierbei ist jedoch, dass bisher existierende SAR noch nicht über einen breiten Funktionsumfang verfügen, um unterschiedliche Anwendungsfälle abzudecken. Ein weiteres Hindernis ist, dass die Nutzer dieser Systeme auch auf Grund ihres fortgeschrittenen Alters verschiedene Einschränkungen haben, die dazu führen, dass es zu stark variierenden Nutzeranforderungen im Bezug auf die Interaktionsmöglichkeiten zwischen Nutzer und Roboter kommt [Lie21].
Wenn eine Lösung gefunden werden kann, wie SAR die Interaktion adaptiv an die Anforderungen und Wünsche der Benutzer anpassen können, wäre es möglich, diese vermehrt im Altenpflegebereich einzusetzen und somit das Personal zu entlasten.
Um dieses Ziel zu erreichen stellt sich die folgende Frage:

Forschungsfrage: Wie sehen die Regelsätze und der Konfigurationsmechanismus aus, die notwendig sind, damit sich ein SAR während der Laufzeit an die Einschränkungen und damit verbundenen Anforderungen eines Nutzers anpassen kann?

Ziel der Arbeit: Das Ziel dieser Arbeit ist die Entwicklung eines Adaptionsmechanismus für SAR. Dafür sollen zunächst Variabilitätsmodelle verwendet werden, um einerseits die Anforderungen der Nutzer und andererseits die Interaktionsmöglichkeiten abzubilden. Anhand dieser Modelle soll das System während der Laufzeit konfiguriert werden und somit auf eine Gestaltung der Interaktion mit dem SAR geschlossen werden. Dafür ist die Erstellung eines Regelsatzes, der den Konfigurationsprozess steuert, notwendig. Ein weiteres Teilziel der Arbeit ist die Überprüfung auf Eignung des entwickelten Modells. Dazu ist es notwendig, Teile des Modells in Form eines Prototyps zu entwickeln. Der Prototyp soll hierbei ein Sprachassistent sein, der bei einer technischen Evaluation genutzt werden kann.

Aufbau der Arbeit: Die Arbeit beginnt mit einer Darstellung der benötigten Grundlagen. Dazu wird zunächst auf unterschiedliche Robotertypen eingegangen, um schließlich SAR zu definieren. Anschließend werden (altersbedingte) Einschränkungen und dadurch resultierende Anforderungen aufgelistet. Im nächsten Schritt wird auf die Software-Variabilität eingegangen. Dort werden die Software-Produktlinien-Entwicklung und verschiedene Arten von Feature-Modellen vorgestellt. Abgeschlossen werden die Grundlagen mit einer Vorstellung der symbolischen KI durch regelbasierte Abläufe.
Danach werden die Anforderungen an eine SAR-Interaktion analysiert und strukturiert gesammelt. Es wird gezeigt, wie sich Nutzer- und Interaktionsdaten in Form eines Variabilitätsmodells speichern lassen.
Darauf folgt die Ausarbeitung des Konzepts, welches die Erstellung eines Adaptionsmodells für die SAR-Interaktion beschreibt. Dieses Konzept wird aus den beiden beschriebenen Modellen, der Definition eines Regelsatzes und des dazu gehörenden Konfigurationsmechanismus bestehen.
Im darauf folgenden Kapitel wird das Konzept teilweise durch einen Prototyp eines Sprachassistenten realisiert, damit es letztlich durch eine technische Evaluation ausgewertet werden kann.
Nach einer Zusammenfassung wird die Arbeit mit einer Diskussion, einem Fazit und einem Ausblick auf mögliche weitere Arbeiten abgeschlossen.

2 Altersbedingte Einschränkungen im Bezug auf die Interaktion mit SAR und Software-Variabilität

Dieses Kapitel dient dazu, dem Leser Grundlagen über Themen, die für diese Arbeit relevant sind zu vermitteln. Dabei wird zunächst auf Roboter und im speziellen auf SAR eingegangen. Es folgt eine Darstellung möglicher Einschränkungen, die Nutzer eines SARs haben können. Generelle Aspekte zur Software-Variabilität in Form von Software-Produktlinien und Feature-Modellen als Variabilitätsmodell folgen darauf. Der Teil wird mit einer Vorstellung der regelbasierten Künstlichen Intelligenz abgeschlossen.

2.1 Sozialer Assistenzroboter (SAR)

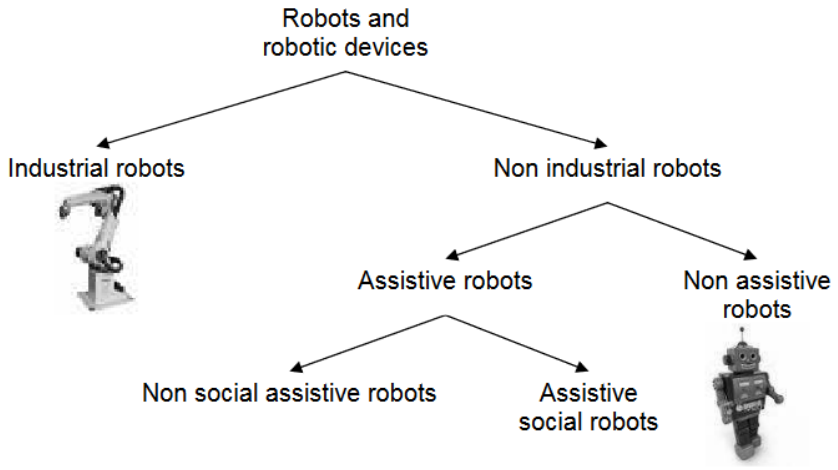
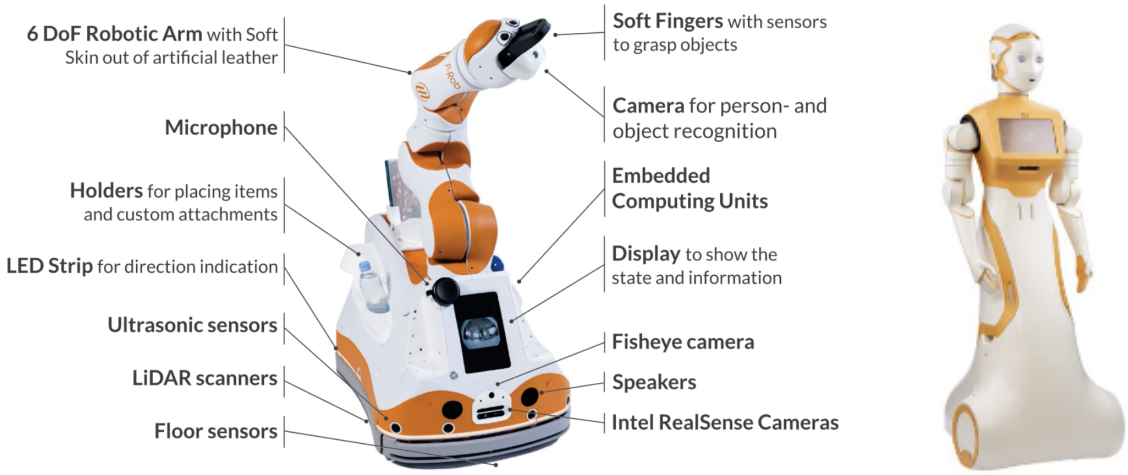


Abbildung 2.1 – Kategorisierung von Robotern nach Heerink [Hee10]

Heerink kategorisiert Roboter anhand ihres Aufgabenfeldes [Hee10]. Diese Kategorisierung ist in Abbildung 2.1 zu sehen. An erster Stelle wird zwischen Industrierobotern, welche für industrielle Automatisierungsanwendungen eingesetzt werden [Sta12] und Robotern, die nicht industriell genutzt werden unterschieden. Die zweite Gruppe wird im Anschluss erneut unterteilt. Zum einen in nicht-assistive Systeme, wie beispielsweise Spielzeugroboter, und Assistenzroboter (AR). Letztere sollen nach Feil-Seifer und Mataric [FM05] sowohl bei körperlichen Einschränkungen helfen, als auch menschlichen Benutzern Hilfe und Unterstützung durch Interaktion ohne körperlichen Kontakt bieten. In der Richtlinie 8373 [Sta12] werden AR unter der Kategorie Service Roboter de-

finiert. Ein Service Roboter ist demnach ein Roboter, welcher für einen Menschen oder ein Gerät eine nützliche Aufgabe ausführt und dabei nicht für industrielle Automatisierungsaufgaben, wie zum Beispiel die Herstellung, Inspektion oder Montage, eingesetzt wird.
Bei Service Robotern wird zwischen den Persönlichen Service Robotern und Professionellen Service Robotern unterschieden. Erstere werden für unkommerzielle Zwecke eingesetzt und können von Laien bedient werden. Der Anwendungszweck der zweiten Gruppe hingegen liegt im kommerziellen Gewerbe. Hier wird meistens ein Trainer, welcher im Vorfeld über die Interaktion mit dem Roboter geschult wurde, benötigt.
Heerink [Hee10] unterscheidet AR in sozial und nicht-sozial. Die Aufgabe der Nicht-Sozialen Assistenzroboter ist es, den Benutzer bei körperlichen Aufgaben, wie zum Beispiel einer Rehabilitation, zu unterstützen. SAR können laut Heerink als soziale Entitäten angesehen werden, mit denen der Benutzer kommunizieren kann und die mit Menschen interagieren können. Sie können in Form eines Service-Roboters genutzt werden, indem sie den Benutzer bei alltäglichen Aufgaben unterstützen und ihm so zu einem unabhängigeren Leben verhelfen. Als Companion-Roboter bauen sie zum Nutzer ein freundschaftliches Verhältnis auf und können dadurch unter Umständen seinen Gesundheitsstatus besser überwachen.
Feil [FM05] sieht den SAR als Schnittmenge eines AR und eines Sozialen Interaktiven Roboters (SIR). Ein SIR wird durch Fong [FND02] als Roboter, mit welchem der Nutzer wie mit einem Menschen interagieren kann, definiert. Der Roboter soll dadurch als Partner, Assistent oder Freund dienen. Die Ziele der AR und SIR werden bei einem SAR kombiniert. So soll er dem menschlichen Nutzer einerseits Hilfe anbieten, was Aufgabe des AR ist. Zusätzlich soll es möglich sein, eine menschenähnliche Interaktion zwischen Mensch und Roboter durchzuführen, was in den Aufgabenbereich eines SIR fällt.
Matari [MS16] beschreibt SAR als Roboter, die den Nutzer nicht nur körperlich, sondern zusätzlich sozial unterstützen. Er vergleicht ihn mit einem Lehrer oder Trainer, der dem Benutzer Motivation, Hilfestellungen, Unterstützung und Training (beispielsweise für soziale Interaktion) anbietet. Hauptsächlich geschieht die Unterstützung jedoch nicht körperlich, sondern durch Interaktion mit dem Roboter. Durch die zu erwartende zunehmende Zahl an älteren Personen [Uni20] sieht Matari eine steigende Notwendigkeit von SAR, um Menschen bei täglichen Aufgaben zu unterstützen, ihnen dabei zu helfen, gesund und aktiv zu leben und Isolationen zu vermeiden [MS16].

Der Roboter Lio, welcher auf Abbildung 2.2a zu sehen ist, wurde entwickelt, um Aufgaben in Pflegeheimen oder in der häuslichen Pflege zu automatisieren. Der Roboter besitzt eine mobile Plattform, mit deren Hilfe er sich bewegen kann. Auf der Plattform ist ein Roboterarm befestigt, sodass es möglich ist, Gegenstände sowohl vom Boden, als auch von größeren Höhen, wie beispielsweise einem Tisch zu erreichen. Durch den Roboterarm kann der Roboter einerseits Aktionen ausführen und ihn andererseits als Eingabeinstrument verwenden. Am Ende des Arms sind Greifer befestigt, durch die der Roboter die Möglichkeit besitzt, Objekte zu greifen. [Miš+20]
ARI (Abbildung 2.2b) ist ein SAR, der durch sein menschenähnliches Aussehen die Akzeptanz an Roboter fördern soll. Zu seinen Aufgaben zählen unter anderem die Unterhaltung der Benutzer, eine Erinnerungsfunktion an Termine oder die Einnahme wichtiger Medikamente und die Überprüfung von Vitalparametern. [Coo+20]



(a) Lio **(b)** ARI

Abbildung 2.2 – Aufbau des Roboters Lio [Miš+20] und Roboter ARI [Coo+20]

2.2 Altersbedingte Einschränkungen

Der ISO-Guide 71 [Sta14] gibt an, dass Einschränkungen im Alter zunehmen können. Daneben beschreibt er, dass Einschränkungen einerseits temporär (wie beispielsweise ein Knochenbruch) und andererseits dauerhaft (wie zum Beispiel Demenz) auftreten können. Zusätzlich wird angegeben, dass Einschränkungen sichtbar oder nicht sichtbar sein können.
Auch wenn es einzelne Einschränkungen gibt, durch die eine problemlose Interaktion mit SAR auf den ersten Blick nicht beschränkt wird, kann es vorkommen, dass es durch die Kombination mit anderen, kleineren Einschränkungen zu Problem führen kann.
Der Guide unterscheidet zwischen sensorischen, körperlichen und kognitiven Einschränkungen.

2.2.1 Sensorische Einschränkungen

Unter sensorischen Einschränkungen werden die Beeinträchtigungen zusammengefasst, die die korrekte Wahrnehmung der Umwelt verändern. Relevant sind diese Einschränkungen im Bezug auf SAR, dass aufgrund ihrer die Ausgabe von Informationen vom Roboter an den Menschen gestört werden kann und möglicherweise Alternativen angeboten werden müssen.

Sehvermögen

Unter das Sehvermögen, welches zu den sensorischen Fähigkeiten des Menschen gezählt wird, fällt die korrekte Wahrnehmung von Licht, Farbe, Kontrast, Form und Größe von Objekten. Schwierigkeiten können daher auftreten, wenn der Fokus zwischen nahen und weit entfernten Objekten gewechselt wird, zwischen unterschiedlichen Farben unterschieden werden muss oder sich an unterschiedliche Lichtverhältnisse gewöhnt werden muss. Personen, die eine Sehschwäche haben, können auch empfindlicher auf Blendungen und blinkendes beziehungsweise flackerndes Licht reagieren.
Der ISO-Guide 71 empfiehlt, bei einer hohen Sehschwäche und bei blinden Menschen auf alternative Interaktionskonzepte auszuweichen. Möglich wäre beispielsweise eine auditive oder taktile

Benutzung.
Das visuelle User Interface (UI) könnte durch Anpassung der Objektgröße, der Helligkeit und des Kontrastes verändert werden. Dabei muss jedoch darauf geachtet werden, dass die Person durch eine zu hohe Helligkeit nicht geblendet wird. Helle Schrift auf dunklem Untergrund stellte sich als gelungenes Design heraus. Auch ein vergrößerter Zeilenabstand und die Verwendung einer klar erkennbaren Schriftart ohne Serifen wirkte sich positiv auf die Wahrnehmung aus. [Sta14]
Ältere Menschen können Probleme beim Scharfstellen naher Objekte haben, sodass sie die Inhalte eines nahegelegenen Bildschirms nicht sehen können [AA04].
Sabev et al. beschreiben, dass blinde und sehbehinderte Menschen akustische Hilfsmittel wie Screenreader verwenden, die den Inhalt des Bildschirms vorlesen. Sehbehinderte Menschen nutzen des Weiteren Bildschirmlupen, durch die es möglich wird, relevante Informationen vergrößert darzustellen [SGB20]. Auch Strantz [Str21] verweist auf die Möglichkeit, den visuell dargestellten Text durch eine Audio-Ausgabe verfügbar zu machen. Außerdem gibt er den Punkt an, dass Farben möglichst unterscheidbar gewählt werden sollen. Durch einen hohen Kontrast zwischen Text- und Hintergrundfarbe ist es somit für ältere Menschen einfacher, geschriebene Informationen zu lesen. Auch Erkennungsprobleme in Situationen mit wenig Licht lassen sich so vermeiden. Arch und Abou-Zhara geben an, dass die Kontrastempfindlichkeit im Alter um über 80% abnehmen kann [AA04]. Der ISO-Guide 71 empfiehlt hier bestimmte Farbkombinationen, die sich positiv auf die Erkennbarkeit auswirken [Sta14].

Hörvermögen

Ebenfalls zu den sensorischen Fähigkeiten zählt das Hörvermögen. Neben der reinen Wahrnehmung der Sprache wird hier auf die korrekte Erkennung unterschiedlicher Tonhöhen, Lautstärken und der Ortung eines auditiven Signals geachtet.
Bei einer Beeinträchtigung des Hörvermögens ist es beispielsweise nicht mehr möglich bestimmte (meist hohe) Frequenzbereiche zu hören und es kann zu Lokalisierungsschwierigkeiten kommen. Probleme können ebenfalls bei niedrigen Lautstärken auftreten. Dies wird unter anderem durch hohe Lautstärken in der Umgebung oder in Situationen, in denen mehr als eine Person spricht, verstärkt. Auch plötzliche Änderungen der Lautstärke führen zu Schwierigkeiten.
Neben einem multimodalen Interaktionskonzept, welches neben auditiven Elementen visuelle und taktile Features unterstützt, wurde hier eine angemessene Lautstärke und Frequenzhöhe vorgeschlagen, die anhand der Umgebung variiert. Zusätzlich zu der automatischen Regulierung bei Veränderungen der Umgebungslautstärke, wird eine Einstellung vorgeschlagen, durch die sich der Nutzer seine Wunschlautstärke und bevorzugte Frequenz auswählen kann. [Sta14]
Pascual et al. haben eine Übersicht aufgestellt, mit deren Hilfe Menschen nach ihrem Hörvermögen eingestuft werden können. Bei einer *leichten* Schwerhörigkeit kann man Geräusche, die lauter als 21 - 40 dB sind wahrnehmen. Bei *mäßiger* liegt die Schwelle bei 41 - 70 dB und bei *schwerer* bei 71 - 90 dB. *Hochgradiger* Hörverlust oder auch *Gehörlosigkeit* liegt dann vor, wenn nur noch Geräusche, die lauter als 90 dB sind wahrgenommen werden können. Für diese Zielgruppe empfehlen Pascual et al. Lippenlesen oder Gebärdensprache. [PRG14]
Für Menschen mit Gehörlosigkeit oder Schwerhörigkeit raten Cavender und Ladner zum Einsatz von Untertiteln bei Video- oder Audioinhalten [CL08].

2.2.2 Körperliche Einschränkungen

Die Fähigkeit zur körperlichen Interaktion mit Objekten, wie die Benutzung eines Touchscreens oder ähnlichen haptischen Interaktionsmöglichkeiten, können durch körperliche Beeinträchtigungen erschwert werden. [Sta14]
Arch und Abou-Zhara benennen Probleme dieser Art, die im Alter auftreten können, zum Beispiel durch Arthritis, Parkinson oder ähnliche Krankheiten. Den betroffenen Personen fällt es in diesen Situationen schwer, eine Maus oder andere Zeigegeräte zu bedienen. [AA04]
Ziman et al. erwähnen Schwierigkeiten bei präzisen Aktionen, wie dem genauen Treffen eines kleinen Buttons [Zim17].
Gröger teilt körperliche Einschränkungen in zwei Gruppen ein. Personen mit leichten motorischen Beeinträchtigungen sind nach ihm weiterhin in der Lage ein Touchdisplay zu bedienen. Die Bedienelemente müssen hierbei aber groß genug gestaltet werden. Personen mit schweren motorischen Beeinträchtigungen können kein Touchdisplay mehr benutzen. [Grö21]
Durch körperliche Einschränkungen der Sprachfähigkeit wird die Artikulation, die gesprochene Geschwindigkeit und die Lautstärke eingeschränkt. Dadurch kann es zu Schwierigkeiten bei der Interaktion mit Systemen kommen, welche über eine Spracheingabe verfügen. [Sta14]

2.2.3 Kognitive Einschränkungen

Zu den kognitiven Fähigkeiten zählt die Möglichkeit, Informationen zu verarbeiten und zu behalten. Probleme kann es beispielsweise dann geben, wenn ein Nutzer nicht weiß, wie er mit dem SAR interagieren soll, der SAR Informationen zu schnell ausgibt oder zu komplizierte Formulierungen verwendet, sodass der Benutzer diese nicht verarbeiten kann. Einer Studie von Lindenberger et al. nach, nimmt neben der Denkfähigkeit auch das Gedächtnis sowie die Wortflüssigkeit mit steigendem Alter ab [Lin+10]. Durch Erkrankungen, die sich auf das Gedächtnis auswirken, wie Demenz oder Alzheimer, kann es vorkommen, dass Nutzer bestimmte Funktionen falsch oder gar nicht nutzen können. Das Interaktionskonzept muss folglich so aufgebaut sein, dass es ohne Anleitung, sondern intuitiv zu bedienen ist. [Sta14]

2.3 Software-Variabilität

Dieses Unterkapitel stellt zunächst die Software-Produktlinien-Entwicklung vor und zeigt anschließend, wie sich variable Software in Form eines Feature-Modells darstellen lässt. Dabei wird neben klassischen Feature-Modellen auch auf Erweiterungen durch Kardinalitäten oder Attribute eingegangen.

2.3.1 Software-Produktlinien

Pohl et al. [PBL05] definieren Software-Produktlinien (SPL) als Paradigma, um Software durch Plattformen und Massenanpassungen zu entwickeln. Dazu sollen wiederbenutzbare Teile entwickelt werden.
Linden et al. [LSR07] beschreiben ebenfalls SPL. Bei diesen wird zwischen Gemeinsamkeiten, Variabilität und produktspezifischen Eigenschaften unterschieden. Gemeinsamkeiten sind Teile, die bei allen Produkten aus der Produktlinie verwendet werden. Durch die Variabilität werden die

Teile beschrieben, die nur bei einigen, aber nicht bei allen Produkten vorkommen. Produktspezifische Eigenschaften kommen in absehbarer Zukunft nur bei einem einzelnen Produkt vor.
Zusätzlich nennen Linden et al. [LSR07] Vorteile von Software-Produktlinien. So wirkt sich die Verwendung von SPL positiv auf den Preis des Produktes und auf die benötigte Entwicklungszeit aus, da Teile aus existierenden Produkten übernommen werden können. Durch die Wiederverwendung von bekannten Elementen in der Benutzeroberfläche steigt die Benutzbarkeit und damit die Qualität des Produktes, da die Nutzer durch einheitliche UIs diese bereits durch andere Produkte kennen. Die Qualität steigt auch aus dem Grund, weil eine ausgereifte und bewährte Basis verwendet werden kann, was Fehler reduziert. Ein Nachteil der SPL-Entwicklung ist, dass bei einer kompletten Neuentwicklung erst eine wiederverwendbare Basis erstellt werden muss.

2.3.2 Feature-Modelle

Kang hat in einem Report [Kan+90] den Nutzen und Aufbau von Feature-Modellen erläutert. Sie werden dazu genutzt, um die, für einen Kunden relevanten, Eigenschaften und Funktionen darzustellen. Das Modell soll gemeinsame und variable Funktionen der einzelnen Produkte abbilden. Dabei können Konfigurationen eines Softwareprodukts durch Auswahl bestimmter Features zusammengestellt werden.
Zur anschaulicheren Erklärung der Eigenschaften eines Feature-Modells, wird in den Abbildungen 2.3 und 2.4 wie bei Kang [Kan+90] und Czarnecki und Eisenecker [CE99] beispielhaft ein Auto als variables Produkt betrachtet.
Die einzelnen Features sind baumartig strukturiert. Generell gilt die Regel, dass bei Auswahl eines bestimmten Features auch der zugehörige Eltern-Knoten gewählt werden muss.
Es gibt unterschiedliche Möglichkeiten der Beziehung zwischen einem Eltern- und einem Kind-Knoten.

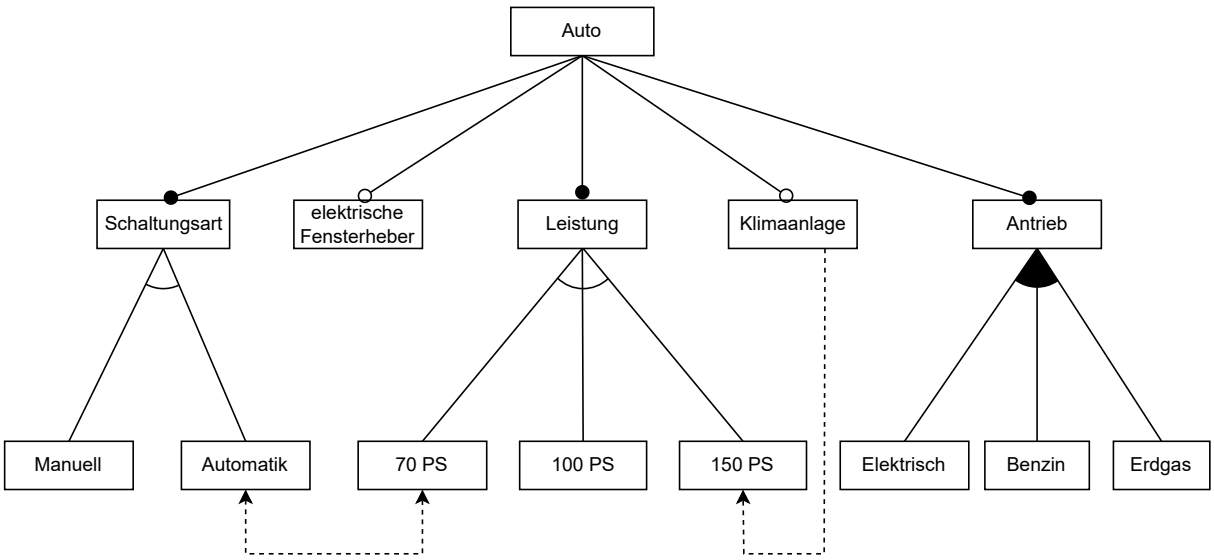


Abbildung 2.3 – Feature-Modell mit den Erweiterungen von Czarnecki und Eisenecker. Angelehnt an [CE99]

Obligatorisches Feature: Ein Feature ist obligatorisch, wenn es bei Auswahl des Eltern-Knotens auch ausgewählt werden muss. Sobald der Eltern-Knoten für eine Konfiguration gewählt wurde, müssen alle obligatorischen Kind-Knoten ebenfalls ausgewählt werden [TBK09]. Im Diagramm werden sie nach Kang nicht speziell gekennzeichnet, sondern nur durch eine einfache Linie zwischen Eltern-Knoten und Kind-Knoten dargestellt. Czarnecki und Eisenecker [CE99] hingegen nennen obligatorische Elemente *Mandatory Features* und stellen diese durch eine Linie, welche am Kind-Knoten einen gefüllten Kreis besitzt dar, wie es in Abbildung 2.3 zu sehen ist. Diese Kennzeichnung wird in dieser Arbeit übernommen.
Im Modell aus Abbildung 2.3 sind die Features *Schaltungsart*, *Leistung* und *Antrieb* obligatorisch. Bei der Konfiguration eines Autos müssen folglich diese drei Feature gewählt werden.

Optionales Feature: Ein optionales Feature kann ausgewählt werden, wenn der Eltern-Knoten ausgewählt wurde. Dies ist aber nicht verpflichtend. Grafisch dargestellt werden sie sowohl von Kang, als auch von Czarnecki und Eisenecker als Linie, die einen ungefüllten Kreis am Kind-Knoten besitzt. Im Modell auf Abbildung 2.3 sind die *Klimaanlage* und die *elektrischen Fensterheber* optionale Features. Ein Auto kann demnach entweder mit oder ohne Klimaanlage beziehungsweise mit oder ohne elektrischen Fensterhebern bestellt werden.

XOR-Oder-Gruppe: Kang stellt des Weiteren alternative Features vor. Aus der Gruppe der Kind-Knoten darf in diesem Fall nur ein Feature ausgewählt werden. Diese XOR-Relation wird durch einen unausgefüllten Winkel am Eltern-Knoten dargestellt. Die Auswahl genau eines Kind-Knotens kann als Spezifikation des Eltern-Knotens angesehen werden. Im Beispiel kann entweder eine manuelle Schaltung oder ein Automatikgetriebe gewählt werden. Diese Wahl spezifiziert das Feature *Schaltungsart*.

OR-Oder-Gruppe: Während es im klassischen Modell von Kang nur alternative Funktionen gibt, stellen Czarnecki und Eisenecker sogenannte *Oder-Features* vor. Aus der Menge der Kind-Knoten muss mindestens ein Feature gewählt werden. Es wäre aber auch möglich, mehrere Features auszuwählen. Dargestellt wird diese Relation durch einen ausgefüllten Winkel am Eltern-Knoten. Im beispielhaften Feature-Modell auf Abbildung 2.3 handelt es sich bei den Kind-Knoten unter dem Feature *Motor* um eine OR-Oder-Gruppe. Ein Auto muss einen Motor als Antrieb besitzen. Dieser kann entweder elektrisch oder durch Verbrennung eines Kraftstoffs angetrieben werden. Bei Hybrid-Fahrzeugen ist aber auch eine Kombination mehrerer Antriebsarten möglich.

Benötigt Relation: Neben Beziehungen zwischen Eltern- und Kind-Knoten, kann man auch Verbindungen auf anderen Ebenen kennzeichnen. So kann es vorkommen, dass ein Feature A nur dann in die Konfiguration aufgenommen werden kann, wenn ein anderes Feature B ebenfalls aufgenommen wird. Feature A hängt in diesem Fall von Feature B ab. Diese Verbindung wird durch einen gestrichelten Pfeil von Feature A nach Feature B (mit Spitze an B) gekennzeichnet. [GFd98] [BTR05]. Im Beispiel auf Abbildung 2.3 hängt die Klimaanlage von der Leistung des Autos ab. Sie kann nur dann hinzugefügt werden, wenn die Leistung 150 PS beträgt.

Ausgeschlossen Relation: Zwei Features können sich auf der anderen Seite auch gegenseitig ausschließen. Wenn Feature A für die Konfiguration gewählt wurde, kann Feature B nicht mehr ausgewählt werden. Dies wird durch einen gestrichelten Pfeil von Feature A nach Feature B, welcher

an beiden Enden eine Spitze besitzt, gekennzeichnet. Im Beispiel liegt diese Relation zwischen dem Automatikgetriebe und einer Leistung von 70 PS vor.

Kardinalitätsbasierte Feature-Modelle

Während bei den bisher vorgestellten Varianten von Feature-Modellen die Darstellung rein grafisch erfolgte, ist es bei kardinalitätsbasierten Feature-Modellen durch Angabe konkreter Zahlen einfacher möglich, Informationen darzustellen.
Czarnecki et al. [CHE05] und Benavides et al. [BTR05] erläutern kardinalitätsbasierte Feature-Modelle genauer. Es wird zwischen zwei unterschiedlichen Arten von Kardinalitäten unterschieden. Die Feature-Kardinalität beschreibt ein spezielles Feature im Diagramm, während die Gruppen-Kardinalität die Auswahl einzelner Features von einer ganzen Gruppe genauer definiert.

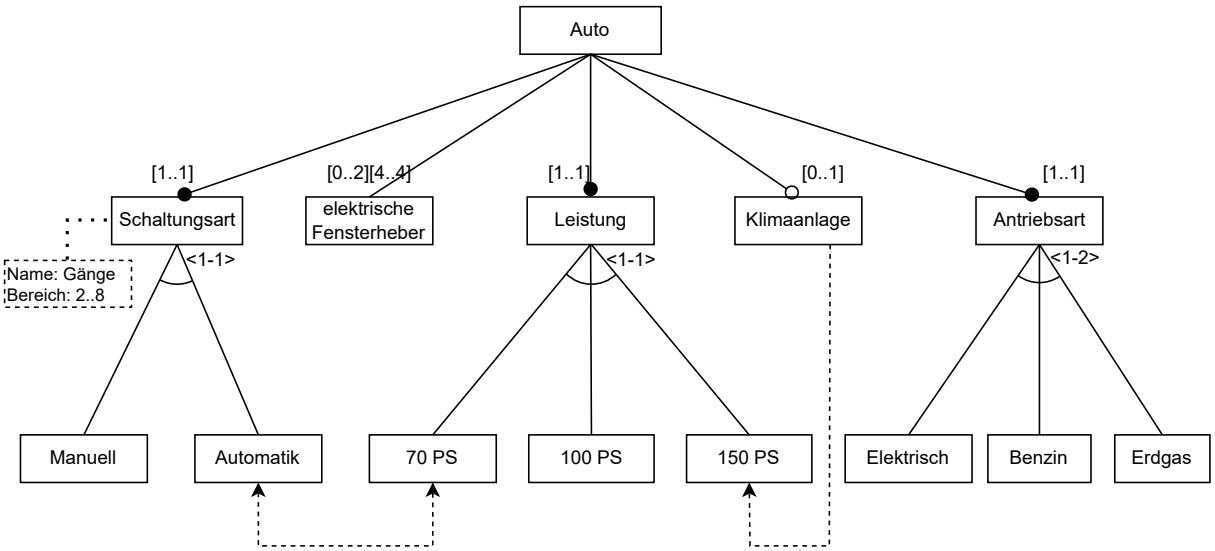


Abbildung 2.4 – Kardinalitätsbasiertes Feature-Modell nach Czernecki [CHE05]

Feature-Kardinalität: Bei den bisher vorgestellten Varianten der Feature-Modelle war es nur möglich auszudrücken, ob ein Feature gewählt werden muss (obligatorisches Feature) oder, ob es wählbar ist (optionales Feature). Eine Aussage, wie häufig das Feature gewählt werden kann war somit nicht möglich.
Die Feature-Kardinalität ermöglicht dies. Durch Angabe von Intervallen kann festgelegt werden, wie häufig ein Feature minimal beziehungsweise maximal ausgewählt werden muss. Die Darstellung erfolgt hierbei durch eckige Klammern der Form $[a..b]$ und steht dabei dafür, dass das Feature mindestens a-mal und maximal b-mal gewählt werden muss. Auch eine Verkettung mehrerer Intervalle ist möglich. Dabei ist zu beachten, dass die obere Schranke des linken Intervalls kleiner sein muss, als die untere Schranke des folgenden Intervalls. Möchte man beliebig viele Features zulassen, so gibt man als obere Schranke ein $*$ im Intervall an.
Obligatorische Features lassen sich durch die Kardinalität $[1..1]$ und optionale mit $[0..1]$ darstellen.
Im beispielhaften Feature-Modell in Abbildung 2.4 wurde den bisherigen obligatorischen und

optionalen Features die entsprechende Kardinalität hinzugefügt. Das Feature *elektrischer Fensterheber* wurde angepasst. Durch Verwendung der Feature-Kardinalität ist es so möglich auszudrücken, dass ein Auto zum einen ohne elektrischen Fensterheber ausgeliefert werden kann. Konfigurationen mit einem oder zwei elektrischen Fensterheber(n) sind auch möglich. Wenn mehr elektrische Heber gewünscht sind, müssen jedoch 4 bestellt werden; eine Konfiguration mit nur 3 elektrischen Fensterhebern ist nicht möglich.

Gruppen-Kardinalität: Eine Einschränkung der bisher vorgestellten Modelle war, dass es bei Feature-Gruppen lediglich zwei Auswahlvarianten gab. So war es entweder möglich, exakt ein Feature auszuwählen (XOR) oder mindestens eines (OR). Eine weitere Limitierung gab es nicht.
Durch die Gruppen-Kardinalität wird das möglich. Auch hier wird mit Hilfe eines Intervalls angegeben, wie viele Features aus der Gruppe gewählt werden können. Die Notation erfolgt hier mit spitzen Klammern in der Form $\langle a - b \rangle$. Das bedeutet, dass mindestens a und maximal b Features aus der Gruppe gewählt werden müssen.
Die bisher verwendeten Gruppentypen lassen sich durch die kardinalitätsbasierte Darstellung ebenfalls abbilden. Eine Gruppe mit alternativen Features kann somit durch die Kardinalität $\langle 1 - 1 \rangle$ und eine Gruppe mit Oder-Features durch $\langle 1 - N \rangle$ geschrieben werden, wobei N die Anzahl aller Features aus der jeweiligen Gruppe ist.
Im Feature-Modell in Abbildung 2.4 wurden die OR-Oder-Gruppen und XOR-Oder-Gruppen durch die neu definierte Gruppen-Kardinalität ersetzt. Die Feature-Gruppe für die Antriebsart wurde weiter spezialisiert. So ist es durch Einführung der neuen Notation einfacher möglich einzuschränken, dass mindestens eine aber gleichzeitig maximal 2 Antriebsarten für ein Auto gewählt werden müssen.

Feature-Modelle mit Attributen

Benavides et al. fassen verschiedene Publikationen zu Erweiterungen von Feature-Modellen durch Attribute zusammen [BSR09]. Sie verweisen auf die Notationsvariante von Banavides et al. [BTR05], bei der die Attributdetails in einem separaten, gestrichelten Kasten neben dem Feature notiert sind. Ein Feature und ein zugehöriges Attribut sind durch eine gepunktete Linie verbunden. Zu den Attributdetails gehört neben dem Namen ein Bereich, in dem sich die möglichen Werte des Attributes befinden können. In einem konfigurierten Modell wird außerdem der gewählte Wert für das Attribut gespeichert.
Attribute können von anderen Attributen abhängen. Dies wird durch Angabe einer Formel realisiert. Zur verbesserten Übersicht werden in dieser Arbeit zwei Attribute, die voneinander abhängen durch eine doppelte, gestrichelte Linie gekennzeichnet.
Im beispielhaften Diagramm aus Abbildung 2.4 besitzt das Feature *Schaltungsart* das Attribut *Gänge*. Ein Fahrzeug kann zwischen 2 und 8 Gänge haben.

2.4 Symbolische Künstliche Intelligenz

Garnelo [GS19] beschreibt, dass ein System, welches symbolische Künstliche Intelligenz verwendet, durch die Ausführung von logischen Regeln funktioniert. Durch diese Regeln werden bestimmte Beziehungen zwischen Objekten beschrieben. Vorteil einer symbolischen KI sei das bessere Verständnis der verwendeten Regeln, da diese einen sprachähnlichen Charakter aufweisen.

Außerdem erwähnt er die leichtere Wiederverwendbarkeit und Nutzung existierender Regeln aufgrund ihrer deklarativen Natur. Letztlich neigen symbolische Darstellungen dazu abstrakt zu sein, wodurch Generalisierungen möglich wären.
Bosse definiert symbolische Künstliche Intelligenz so, dass komplexe Entscheidungen auf einen, für den Menschen einfacher verständlichen Regelsatz reduziert werden [Bos20]. Flasinki beschreibt in seinem Buch grundlegende Konzepte der Künstlichen Intelligenz. [Fla16] So gibt er an, dass das Wissen eines Modells eines intelligenten Systems in symbolischer Art und Weise gespeichert wird. Diese Speicherung kann beispielsweise durch Graphen, logische Formeln oder symbolische Regeln geschehen.
In einem regelbasierten Ansatz wird das Wissen in Form von Regeln definiert. Eine Regel ist hierbei wie folgt definiert: Wenn Bedingung A eintritt, dann führe Aktion X aus. Die Aktion X ist hierbei entweder eine Folgerung, die auf Grundlage der Regel geschlossen wird oder eine reale Handlung, die sich auf die Umgebung des Systems auswirkt.
Flasinski beschreibt des Weiteren den generellen Aufbau eines regelbasierten Systems. [Fla16] Im Arbeitsspeicher werden Informationen über Fakten des Systems oder aus der Umgebung des Systems gespeichert. Die einzelnen Regeln sind als logische Sätze definiert. Dabei kann eine Regel auch aus verschiedenen Unterbedingungen bestehen. Die Regeln sind in der Regelbasis gespeichert. Logische Schlussfolgerungen werden durch eine Inferenzmaschine, welche auf den Arbeitsspeicher und die Regelbasis zugreift, getroffen. Dabei durchläuft die Maschine unterschiedliche Phasen. Zu Beginn wird geprüft, welche Regeln zu den Fakten aus dem Arbeitsspeicher passen und diese anschließend in eine Konfliktmenge gespeichert. In der Konfliktlösungs-Phase wählt die Maschine eine Regel oder eine Sequenz an Regeln aus der Konfliktmenge aus, die ausgeführt werden sollen. In der letzten Phase werden die ausgewählten Regeln angewendet und das System geht wieder in die erste Phase über. [Fla16]

2.5 Zusammenfassung

Im bisherigen Teil der Arbeit wurden die Grundlagen für SAR erläutert und diese in unterschiedliche Robotergruppen eingeordnet. Darauf folgend wurde eine Auswahl an altersbedingten Einschränkungen vorgestellt. Dabei wurden die Beeinträchtigung in drei Kategorien unterteilt. Sensorische Einschränkungen schränken die Ausgabe, körperliche die Eingabe und kognitive die Verarbeitung von Informationen des SAR ein. Zum Thema der Software-Variabilität wurden Software-Produktlinien und Feature-Modelle zur Darstellung möglicher Varianten vorgestellt. Dort wurden auch kardinalitätsbasierte Feature-Modelle und solche mit Attributen erwähnt. Letztlich wurde die regelbasierte Künstliche Intelligenz behandelt.
Folgend soll nun erarbeitet werden, welche Anforderungen sich an eine Adaption der Interaktion von SAR ergeben.

3 Analyse der Anforderungen zur KI-basierten Adaption der Interaktion mit SAR

Nach der Vorstellung der Grundlagen, in der SAR eingeordnet, Software-Variabilität, Software-Produktlinien und Feature-Modelle erklärt und die regelbasierte Künstliche Intelligenz erläutert wurden, ist das Ziel dieses Kapitels, die Anforderungen zu analysieren, die dafür notwendig sind, dass sich SAR während der Laufzeit an die jeweiligen Nutzer und deren Einschränkungen anpassen können. Die Grundlage hierfür bietet das von Kephart und Chess vorgestellte MAPE-K-Prinzip [KC03]. Das vorgestellte Modell beschreibt die eigene Anpassung eines adaptiven Elementes einer Software innerhalb von vier Phasen. Neben diesen Phasen gibt es eine Wissens-Datenbank (**K**: Knowledge), die unter anderem Informationen über den aktiven Nutzer aber auch über die Umgebung, in der sich das System befindet, enthält. Die erste Phase ist das Beobachten (**M**: Monitor) der adaptiven Software und deren Umgebung. In der zweiten Phase werden die soeben gewonnenen Informationen in Kombination mit den bekannten Daten aus der Wissens-Datenbank analysiert (**A**: Analyse) und entschieden, ob eine Adaption durchgeführt werden muss. Sollte dies der Fall sein, werden in der dritten Phase Ausführungspläne erstellt, die die Adaption beschreiben (**P**: Plan). In der vierten und letzten Phase wird einer der Pläne anhand einer Gewichtung ausgewählt und ausgeführt (**E**: Execute). Nach einem erfolgreichen Durchlauf der Adaption wird erneut mit der ersten Phase begonnen.

Für den **K**-Aspekt des MAPE-K-Prinzips ist es wichtig, relevante Informationen zu speichern. So müssen die möglichen Einschränkungen der Nutzer, die altersbedingt auftreten können betrachtet und gespeichert werden. Zudem ist eine Übersicht über die Interaktionsmöglichkeiten zwischen Roboter und Nutzer notwendig. Auch eine Verknüpfung von Einschränkungen zur Interaktion wird benötigt.

3.1 Anforderungen älterer Menschen an SAR

In Abschnitt 2.2 wurden mögliche Einschränkungen, welche im steigenden Alter auftreten können bereits nach ihrer Art kategorisiert. Anhand dieser Cluster werden im Folgenden Anforderungen der einzelnen Personengruppen gesammelt.

3.1.1 Anforderungen bei sensorischen Einschränkungen

Wie bereits in Abschnitt 2.2 werden sensorische Einschränkungen in zwei Bereiche getrennt. Zum einen Anforderungen, die aus Beeinträchtigungen des Sehvermögens resultieren und zum anderen jene, deren Ursprung in vermindertem Hörvermögen liegt.

Sehvermögen
Sowohl Gröger [Grö21], als auch Fuchs [Fuc21] unterteilen den Abschnitt über das Sehvermögen in Teilabschnitte. So betrachten sie Menschen, deren Sehschärfe reduziert ist, die komplett blind sind oder die eine Farbfehlsichtigkeit aufweisen. Sabev et al. empfehlen für Blinde anstatt eines Displays die Verwendung einer alternativen akustischen Ausgabe. Sofern das Display durch eine andere Anforderung nicht zwingend benötigt wird, soll dieses zusammen mit der Toucheingabe deaktiviert werden. Für Menschen mit einer Sehschwäche kann eine vergrößerte Darstellung der visuellen Informationen hilfreich sein. [SGB20] Auch ein größerer Zeilenabstand stellt sich laut Guide 71 als hilfreich heraus [Sta14]. Einer Umfrage von Loitsch nach, brauchen die meisten blinden Nutzer eine sprachliche Textausgabe oder eine Braillezeile. [Loi18] Eine Anpassung des Sprechers und der Sprechgeschwindigkeit bei auditiven Ausgaben wird von den meisten Befragten gewünscht. Bei Nutzern mit einer Sehschwäche wird häufig eine vergrößerte Darstellung der visuellen Inhalte gefordert. Der Faktor der Skalierung variiert dabei bei jedem Menschen. Es sollte somit möglich sein, im User-Profil eine Präferenz für die Schriftgröße einzustellen. Falls diese vom Nutzer nicht mitgeteilt wurden, sollen Standardwerte verwendet werden. Für Menschen ohne Seheinschränkungen eignet sich beispielsweise eine Schriftgröße von 12 Punkten. Der Deutsche Blinden- und Sehbehindertenverband (DBSV) hat in einer Broschüre vorgeschlagen, dass die Schriftgröße für Menschen mit Sehbehinderungen bis zum Faktor 1,75 erhöht werden sollte. Aus diesem Grund wird eine Größe von 21 als Normalwert verwendet, sofern der Nutzer keine Präferenz angegeben hat. Auch die Präferenzen für den Zeilenabstand sollte sich der Nutzer personalisiert festlegen lassen können. Der DBSV schlägt hier einen Wert von 120 % vor. [Seh19]
Strantz [Str21] verweist auf die Nutzung eines geeigneten Kontrastverhältnisses und Verwendung möglichst unterscheidbarer Farben für den Gebrauch eines Systems durch farbenblinde Nutzer. Von der Verwendung eines globalen Farbschemas, welches für alle Nutzergruppen passend ist raten Chua et al. ab, da Menschen ohne Farbenblindheit bestimmte Farben semantisch bereits vordefiniert haben. [Chu+15] Es ist demnach notwendig, für jede Art der Farbfehlsichtigkeit ein eigenes Farbkonzept zu verwenden, welches unterscheidbare Farben aufweist. Diese Idee vertritt auch Fuchs und bietet als Farbkorrekturen entweder einen Grün-/Rotfilter oder einen Rot-/Grünfilter, als auch einen Blau-/Gelbfilter an, der bei Bedarf jeweils genutzt werden kann [Fuc21]. In den Web Content Accessibility Guidelines (WCAG) wird erläutert, dass Farbe nicht als einziges Unterscheidungsmerkmal genutzt werden soll [WCA]. In Diagrammen könnte sich beispielsweise bei zwei Linien nicht nur die Farbe, sondern auch die Struktur unterscheiden.

Hörvermögen
Anhand der Kategorierung der Hörschwäche von Pascual et al. [PRG14] in leichte, mäßige und schwere Schwerhörigkeit, wie sie in Abschnitt 2.2 vorgestellt wurde, muss die Ausgabelautstärke eines auditiven Systems angepasst werden. Der Nutzer muss hier Präferenzen festlegen, wie laut die Sprachausgabe mindestens sein sollte. Da die Wahrnehmung der Lautstärke frequenzabhängig ist, wird dieser Wert nicht in dB, sondern in Phon angegeben. Der Standardwert für Menschen ohne Hörbeeinträchtigungen von 60 Phon entspricht bei einer Frequenz von 1000 Hz 60 dB, was ungefähr der Schalldruckpegel eines normalen Gesprächs ist. Bei einer Frequenz von 100 Hz hingegen muss der Schalldruckpegel bereits fast 80 dB betragen, damit es gleich laut wirkt. [Myö19] Damit die Schmerzgrenze von 140 dB nicht überschritten wird, sollte die Ausgabelautstärke 100 Phon nicht überschreiten. Falls ein Nutzer mit einer Höreinschränkung keine Mindestlautstärke angegeben hat, sollte ein möglichst hoher Wert genommen werden, damit auch Men-

schen mit einer schweren Schwerhörigkeit die Ausgabe verstehen. Falls dieser Wert dem Nutzer zu hoch ist, kann er ihn nach seinen Wünschen reduzieren. Daher wird ein Standardwert von 90 Phon vorgeschlagen.
Die Schwierigkeit, im zunehmenden Alter hohe Frequenzen wahrzunehmen [LY07], muss unter dem Aspekt betrachtet werden, dass sich die akustische Text- und Tonausgabe an den hörbaren Bereich des Nutzers anpasst. Auch hier ist die Angabe durch den Nutzer erforderlich. Das Standard-Intervall (bei keinen Einschränkungen) von 100 Hz bis 8000 Hz wird, sobald dem System mitgeteilt wird, dass der Nutzer eine Höreinschränkung hat, auf 200 Hz bis 2000 Hz angepasst.
Der Guide 71 beschreibt des Weiteren, dass die Lautstärke des Systems zusätzlich an die Umgebungslautstärke angepasst werden muss. Dazu sollte die Systemlautstärke um eine, vom Nutzer festgelegte Differenz höher, als der Hintergrund sein [Sta14]. Bei Menschen ohne Höreinschränkung wird hier 5 Phon; bei jenen mit einer Einschränkung 10 Phon als Standardwert genommen.
Veljanovska et al. erwähnen die Verwendung der Gebärdensprache, schränken diesen Vorschlag aber ein, weil nicht jeder gehörloser Mensch dieser Sprache mächtig ist [Vel+20]. Der SAR sollte diese Option nur dann wählen, wenn der Nutzer angegeben hat, dass er durch Gebärdensprache kommunizieren kann. Flavia et al. haben die Erkennung von Gebärden mittels Künstlicher Intelligenz untersucht [Fla+22]. So ist es möglich, sowohl komplexe Gesten, wie eine Gebärde, aber auch leichtere Gesten zu erkennen. Statt einer akustischen Ausgabe sollte Cavender et al. nach auf visuellen Ersatz in Form von Untertiteln geachtet werden [CL08]. Anstatt oder neben der Sprachausgabe kann dazu auch eine Textausgabe über einen Bildschirm erfolgen.

3.1.2 Anforderungen bei körperlichen Einschränkungen

Gröger unterteilt körperliche Einschränkung in ihre Stärke. [Grö21] Bei leichten motorischen Einschränkungen ist demnach weiterhin eine Nutzung des Touchscreens möglich, wenn die Bedienelemente auf dem Bildschirm groß genug gestaltet sind. Außerdem kann eine Gestensteuerung verwendet werden, sofern die verwendeten Gesten einfacher Natur sind. Der Guide 71 erwähnt neben der bereits genannten Schwierigkeit, Aktionen, die eine hohe Genauigkeit fordern durchzuführen, dass gleichzeitige Doppelbewegungen schwerer sind, als eine einfache Bewegung. Auf die Verwendung von Multi-Touch-Gesten sollte demnach verzichtet werden. [Sta14] Hier variieren die genauen Anforderungen an die Größe der Bedienelemente von Nutzer zu Nutzer. Als Vorschlag wird hier erwähnt, dass die Buttons auf dem Display im besten Fall quadratisch sind, sodass sie einfacher getroffen werden können. Eine Mindestgröße von 50 Quadratzentimetern, was etwa so groß ist, wie die Hälfte des Displays eines gängigen Smartphones, wirkt sinnvoll, da sie einerseits groß genug ist, damit sie trotz motorischer Einschränkungen getroffen werden kann und andererseits auf den meisten Bildschirmen mehr als eine Auswahloption zulässt. Für Menschen ohne Beeinträchtigungen müssen die Eingabemöglichkeiten nicht so groß gestaltet werden. Als Mindestgröße werden an dieser Stelle ungefähr 3 Quadratzentimeter genannt, was in etwa so groß, wie ein Icon auf einem Smartphone ist.
Bei stärkeren Einschränkungen hingegen kann der Roboter nicht mehr normal via Toucheingabe gesteuert werden. Möglich wäre aber weiterhin eine Eingabe, bei der nur darauf geachtet wird, ob der Bildschirm berührt wurde; die Position ist dabei nicht relevant. Gröger verweist hier auf die Nutzung einer Sprachsteuerung. Des Weiteren erwähnt er, dass Menschen mit motorischen Einschränkungen, die zusätzlich nicht sprechen können auf einen Eye-Tracker angewiesen sind. Diese Anforderung wird in dieser Arbeit anders interpretiert und lediglich betrachtet, dass ein

nicht sprechender Mensch keine Spracheingabe benutzen kann. Für Menschen, die nicht stumm sind und deren Sprache lediglich eingeschränkt ist, schlägt der Guide 71 eine Sprachsteuerung vor, die statt auf den gesprochenen Text auf bestimmte Faktoren, wie beispielsweise die Lautstärke oder Tonhöhe achtet.

3.1.3 Anforderungen bei kognitiven Einschränkungen

Gollasch und Weber haben in einer Umfrage ermittelt, welche Anforderungen an Voice User Interfaces (VUIs) für ältere Personen relevant sind. [GW21] Auch wenn dabei nur auf VUI und nicht direkt auf Menschen mit kognitiven Einschränkungen eingegangen wurde, können einige Ergebnisse für diese Arbeit verwendet werden.
Ältere Menschen präferierten demnach eine schrittweise Sprachausgabe gegenüber komplexen Dialogen. Auch bei der Gestaltung der Fragen finden sie einfache Ja/Nein-Fragen besser als komplexe. Eine klare Kennzeichnung, wann das System eine Eingabe durch den Nutzer erwartet, wurde mit steigendem Alter eher gewünscht. Ebenfalls erwähnten die Nutzer die Wahl der Sprache. Neben der normalen Sprachausgabe mit vielen Informationen und längeren Sätzen ist auch eine Verwendung von kurzen, einfachen Sätzen mit leicht verständlichen Worten wünschenswert. Die genannten Aspekte vereinfachen die Interaktion, was vor allem für Menschen mit kognitiven Beeinträchtigungen hilfreich ist.
Der Guide 71 [Sta14] nennt des Weiteren die Möglichkeit, bei mehrstufigen Prozessen, die in einzelne Teilschritte zerlegt wurden, dem Benutzer stets eine Rückmeldung zu geben, an welcher Stelle er sich aktuell im Prozess befindet. Gerade für Menschen mit Einschränkungen im Gedächtnis ist dieses Vorgehen hilfreich. Laut der von Gollasch und Weber durchgeführten Umfrage wird dieses Verhalten des VUI nicht von allen älteren Personen gefordert. Für Menschen mit Gedächtniseinschränkungen wäre es jedoch eine wichtige Funktion.
Durch Einschränkungen des Kurz- und Langzeitgedächtnisses kann es bei der Interaktion mit Robotern oder Computern dazu führen, dass sich Benutzer nicht daran erinnern können, wie sie mit unterschiedlichen Benutzeroberflächen interagieren können [AA04]. Hier wäre eine dauerhafte Erklärung der UI hilfreich.
Durch Einschränkungen der Sprache kann es vorkommen, dass Menschen nicht in der Lage sind, passende Worte zu ihren Gedanken zu finden. Auch das Verständnis von geschriebener und gesprochener Sprache kann beeinträchtigt sein [Sta14]. Dadurch ergeben sich unterschiedliche Möglichkeiten, welche Interaktionsvarianten nicht nutzbar sind. Bei Menschen, die gesprochene Sprache kognitiv nicht verarbeiten können, kann die Sprachausgabe nicht genutzt werden. Nutzer mit Beeinträchtigungen beim Verständnis von geschriebener Sprache können die Textausgabe über den Bildschirm nicht verwenden. Analog dazu kann die Spracheingabe von Menschen mit kognitiven Sprachstörungen nicht genutzt werden.

3.1.4 Strukturelle Sammlung der recherchierten Anforderungen

Die soeben aus unterschiedlichen Quellen erarbeiteten Anforderungen, die sich durch altersbedingte Einschränkungen ergeben, werden nun strukturiert zusammengefasst, indem sie anhand der Einschränkung aufgelistet werden. Dabei wird jedoch nur erläutert, welche Interaktion aufgrund einer Einschränkung genutzt werden muss beziehungsweise überhaupt nicht genutzt werden kann. Darüber hinaus wäre es für Nutzer möglich eine andere, nicht genannte Interaktionsmöglichkeit zu nutzen.

1. Ohne Einschränkungen
   - 1.1. Anpassung der Lautstärke auf ein Minimum bei der Verwendung einer akustischen Ausgabe. Dieses Minimum kann vom Benutzer festgelegt werden. Sollte keine Angabe erfolgt sein, wird ein Wert von 50 Phon verwendet.
   - 1.2. Die Lautstärke darf die Schmerzgrenze von 140 dB nicht überschreiten und sollte daher maximal 100 Phon betragen.
   - 1.3. Anpassung der Lautstärke an die Umgebungsgeräusche. Die Systemlautstärke (in Phon) muss um eine festgelegte Differenz (in Phon) höher als der gemessene Schalldruckpegel in dB sein. Die Differenz kann der Nutzer selbst bestimmen. Standardmäßig ist diese bei 5 Phon.
   - 1.4. Anpassung der Tonhöhe an den wahrnehmbaren Bereich des Nutzers. Der Bereich kann durch ein, zum Nutzer passendes Intervall durch Angabe des Minimum und Maximums konfiguriert werden. Bei keiner Angabe wird der Systemstandard von 100 Hz bis 8000 Hz verwendet.
   - 1.5. Verwendung von Knöpfen mit einer Mindestgröße von 3 cm^2 auf dem Touchscreen.
2. Einschränkungen im Sehvermögen
   - 2.1. Eingeschränkte Sehschärfe
     - 2.1.1. Verwendung einer akustischer Ein- und Ausgabe in Form einer Sprachsteuerung.
     - 2.1.2. Vergrößerte Darstellung des ausgegebenen Textes. Der Nutzer kann eine minimale Schriftgröße festlegen und jederzeit ändern. Als Standardwert wird 21 Pt. verwendet.
     - 2.1.3. Nutzung eines größeren Zeilenabstands, welcher vom Nutzer als Prozentsatz der Schriftgröße festgelegt werden kann. Bei keiner Angabe wird ein Abstand von 120% der aktuellen Schriftgröße verwendet.
     - 2.1.4. Anpassung der Schaltflächen und Objekte auf dem Display, sodass der vergrößerte Text mit dem erhöhten Zeilenabstand angezeigt werden kann.
   - 2.2. Blindheit
     - 2.2.1. Verwendung einer akustischen Ein- und Ausgabe in Form einer Sprachsteuerung.
     - 2.2.2. Sofern der Nutzer im User-Profil hinterlegt hat, dass er Braille lesen kann, soll die Ausgabe taktil über eine Braillezeile erfolgen. Falls der Nutzer keine Angabe gemacht hat, nimmt das System an, dass er der Brailleschrift nicht mächtig ist und schaltet die taktile Ausgabe aus.
     - 2.2.3. Sofern es zu keinem Widerspruch durch andere Anforderungen kommt, sollen das Display und sonstige visuelle Ausgabegeräte deaktiviert werden.
     - 2.2.4. Sofern es zu keinem Widerspruch durch andere Anforderungen kommt, soll die Toucheingabe deaktiviert werden.
   - 2.3. Farbfehlsichtigkeit
     - 2.3.1. Falls eine optische Ausgabe über ein Display erfolgt, soll dort die farbliche Darstellung auf eine passende Farbpalette umgestellt werden, die der Nutzer gut erkennen kann.

2.3.2. Sofern eine optische Ausgabe über ein Display erfolgt, sollen farblich unterschiedliche Objekte zusätzlich durch ihre Struktur unterscheidbar gestaltet werden. Die Farbe darf nicht mehr alleiniges Unterscheidungsmerkmal sein.

3. Einschränkungen im Hörvermögen
   3.1. Schwerhörigkeit
      3.1.1. Anpassung der Lautstärke auf ein Minimum bei der Verwendung einer akustischen Ausgabe. Dieses Minimum kann vom Benutzer festgelegt werden. Sollte keine Angabe erfolgt sein, wird ein Wert von 90 Phon verwendet.
      3.1.2. Anpassung der Lautstärke an die Umgebungsgeräusche. Die Systemlautstärke (in Phon) muss um eine festgelegte Differenz (in Phon) höher als der gemessene Schalldruckpegel in dB sein. Die Differenz kann der Nutzer selbst bestimmen. Standardmäßig liegt diese bei 10 Phon.
      3.1.3. Anpassung der Tonhöhe an den wahrnehmbaren Bereich des Nutzers. Der Bereich kann durch ein, zum Nutzer passendes Intervall durch Angabe des Minimum und Maximums konfiguriert werden. Bei keiner Angabe wird der Systemstandard von 200 Hz bis 2000 Hz bei Menschen mit Schwerhörigkeit verwendet.
      3.1.4. Redundante oder erweiterte Informationsmöglichkeit in der Form einer Textausgabe über einen Bildschirm durch Verwendung der optischen Ausgabe.
      3.1.5. Nutzung von Gebärden- oder Gestenerkennung zur Nutzereingabe.
   3.2. Gehörlosigkeit
      3.2.1. Verwendung des Bildschirms zur optischen Ein- und Ausgabe der Informationen.
      3.2.2. Nutzung von Gebärden- oder Gestenerkennung zur Nutzereingabe.
      3.2.3. Sofern es zu keinem Widerspruch durch andere Anforderungen kommt, soll die akustische Ausgabe deaktiviert werden.

4. Körperliche Einschränkungen
   4.1. Leichte körperliche Einschränkungen
      4.1.1. Verwendung einer größeren Eingabemöglichkeit auf dem Touchscreen durch quadratische Knöpfe mit einer Mindestgröße von 50 cm^2.
      4.1.2. Verwendung einfacher haptischer Zeigegesten zur Eingabe. Damit sind reine Armbewegungen gemeint, bei der keine Fingerkoordination benötigt wird.
      4.1.3. Verzicht auf komplizierte Multi-Touch-Gesten.
   4.2. Schwere körperliche Einschränkungen
      4.2.1. Knöpfe strecken sich über den kompletten Bildschirm. Zu jedem Zeitpunkt ist maximal ein Knopf auf dem Display verfügbar. Somit ist es nicht mehr relevant wo, sondern nur dass das Display berührt wurde.
      4.2.2. Verwendung einer Sprachsteuerung zur Ein- und Ausgabe.
      4.2.3. Verzicht auf komplizierte Multi-Touch-Gesten.

5. Kognitive Einschränkungen
   5.1. Allgemeine kognitive Einschränkungen
      5.1.1. Verwendung einfacher Sprache durch Verzicht auf komplizierte Begriffe oder Reduzierung der Satzlänge bei Dialogen.
      5.1.2. Verwendung schrittweiser Dialogverarbeitung. Dem Nutzer wird nur eine Frage oder Aufgabe gleichzeitig gestellt. Erst, wenn er diese beantwortet oder gelöst hat, wird mit dem nächsten Punkt fortgefahren.
      5.1.3. Präferierte Verwendung von Ja/Nein-Fragen. Sofern es möglich ist, stellt der SAR seine Fragenformulierung so um, dass der Nutzer nur mit *Ja* oder *Nein* antworten muss.
   5.2. Einschränkungen des Gedächtnisses
      5.2.1. Stetige Rückmeldung und Bestätigung der aktuellen Stufe in einem mehrschrittigen Prozess. Gerade in einer schrittweisen Dialogverarbeitung, wie in Anforderung 5.1.2. beschrieben, sollte der Nutzer über den aktuellen Stand des Dialogs informiert werden. Dazu gehört die Auskunft über bereits abgeschlossene und noch kommende Teilaufgaben.
      5.2.2. Dauerhafte Erklärung der Benutzerschnittstelle. Dem Nutzer sollte stets erklärt werden, welche Aktion zu welchem Ergebnis führen wird.
6. Einschränkungen der Sprache
   6.1. Sofern es zu keinem Widerspruch durch andere Anforderungen kommt: Keine Verwendung der Spracheingabe bei Sprachstörungen.
   6.2. Sofern es zu keinem Widerspruch durch andere Anforderungen kommt: Keine Verwendung der Sprachausgabe bei Problemen, gesprochene Sprache zu verstehen.
   6.3. Wenn trotz Problemen, gesprochene Sprache zu verstehen eine Sprachausgabe aktiviert wird, sollte eine natürliche Stimme verwendet werden.
   6.4. Sofern es zu keinem Widerspruch durch andere Anforderungen kommt: Keine Verwendung der Textausgabe bei Problemen, geschriebene Sprache zu verstehen.

3.2 Einschränkungen der Nutzer erfassen

Um für jeden Nutzer eine möglichst passende Interaktionskonfiguration zu finden, ist es primär notwendig, die Einschränkungen jedes Nutzers zu ermitteln und zu speichern.
Dazu ist es zunächst erforderlich, dass der Roboter einen neuen Nutzer, zu dem bisher noch keine Informationen gespeichert wurden, erkennen kann. Dieser Aspekt wurde bereits von Thile [Thi19] bearbeitet und ist nicht Bestandteil dieser Arbeit. Dort wird auf verschiedene Möglichkeiten der Nutzererkennung eingegangen. Beispielsweise erkennt der Roboter neue Nutzer durch Stimm- oder Gesichtserkennung, anhand des Fingerabdrucks oder der Venen innerhalb der Hand.
Auch die Art der Eingabe der relevanten Nutzerdaten wurde bereits von Fuchs [Fuc21] behandelt. Hier wird zwischen impliziter und expliziter Erkennung unterschieden. Bei der ersteren erkennt der Roboter mit Hilfe von Sensoren automatisch Daten über den Benutzer. Bei der expliziten Erkennung erfolgt die Eingabe der relevanten Informationen durch Befragung des Nutzers oder

Dritte. Da dieses Thema ebenfalls bereits bearbeitet wurde, wird es in dieser Ausarbeitung nicht weiter beleuchtet.

Es wird für diese Arbeit folglich angenommen, dass die Informationen über alle relevanten Einschränkungen des Benutzers bereits vorliegen oder vom Nutzer selbst eingegeben wurden. Ein wichtiger Aspekt ist die Art, wie diese Daten gespeichert werden.
Durch ein Feature-Modell, wie es in Abschnitt 2.3.2 vorgestellt wurde, ist es möglich, alle zulässigen Einschränkungen, die ein Benutzer haben kann zu sammeln. Zulässig bedeutet in diesem Fall, dass bestimmte Einschränkungs-Kombinationen unlogisch sind und nicht betrachtet werden müssen. Für jeden Benutzer des SAR wird ein konfiguriertes Feature-Modell gespeichert, welches die Einschränkungen des selbigen beschreibt. Dieses Feature-Modell wird im Folgenden als FM_{USER} bezeichnet.
Während der Laufzeit des SAR kann es vorkommen, dass der Roboter entweder implizit erkennt, dass die realen Einschränkungen des Nutzers nicht mit den Einschränkungen, die im konfigurierten Modell gespeichert sind, übereinstimmen oder sich die Konfiguration durch explizite Eingabe von aktualisierten Daten ändert.

3.3 Mögliche Varianten der Interaktion

Um die Interaktion mit dem Roboter so, wie in Abschnitt 3.1 beschrieben, anzupassen, müssen verschiedene Interaktionsvarianten zur Verfügung gestellt werden. Um diese unterschiedlichen Varianten gesammelt zu notieren, bietet sich erneut ein Feature-Modell an. Dieses Modell wird mit den Informationen, die in den Abschnitten 2.2 und 3.1 gesammelt wurden, erstellt. Dabei sollte vor allem auf die unterschiedlichen Möglichkeiten der Ein- und Ausgabe von Informationen eingegangen werden. Aus den Anforderungen ging bereits hervor, dass folgende Ein- beziehungsweise Ausgaben benötigt werden:

- Eingabe
  - Haptische Eingabe über einen Touchbildschirm
  - Akustische Eingabe über eine Sprachsteuerung
  - Motorische Eingabe über eine Gestenerkennung
- Ausgabe
  - Optische Ausgabe über einen Bildschirm
  - Akustische Ausgabe über eine Sprachsteuerung
  - Haptische Ausgabe über eine Braillezeile

Der SAR kann aus diesem Feature-Modell eine gültige Konfiguration auswählen, um das Interaktionskonzept zwischen ihm und dem Benutzer anzupassen.
Dieses Feature-Modell wird im Folgenden als FM_{SYSTEM} bezeichnet.

3.4 Entwicklung des Regelsatzes

Durch die soeben vorgestellten Feature-Modelle ist es möglich, die Einschränkungen verschiedener Nutzer zu speichern und zu beurteilen, welche Arten der Interaktion bei dem SAR zugelassen sind. Durch die beiden Modelle wird aber nicht eingeschränkt, dass eine konkrete Interaktionsvariante von einem Benutzer mit bestimmten Einschränkungen nicht genutzt werden kann. Wenn beispielsweise das für Nutzer A konfigurierte Feature-Modell FM_{USER} aussagt, dass dieser blind ist, sollten die Konfigurationen von FM_{SYSTEM}, die eine visuelle Interaktion erlauben, gesperrt werden.
Somit ist es notwendig, Regeln zu speichern, die das Feature-Modell der Einschränkungen FM_{USER} und das Lösungsmodell FM_{SYSTEM} miteinander verbinden und somit nur die Konfigurationen von FM_{SYSTEM} gültig sind, die der Nutzer trotz seiner Einschränkungen verwenden kann.

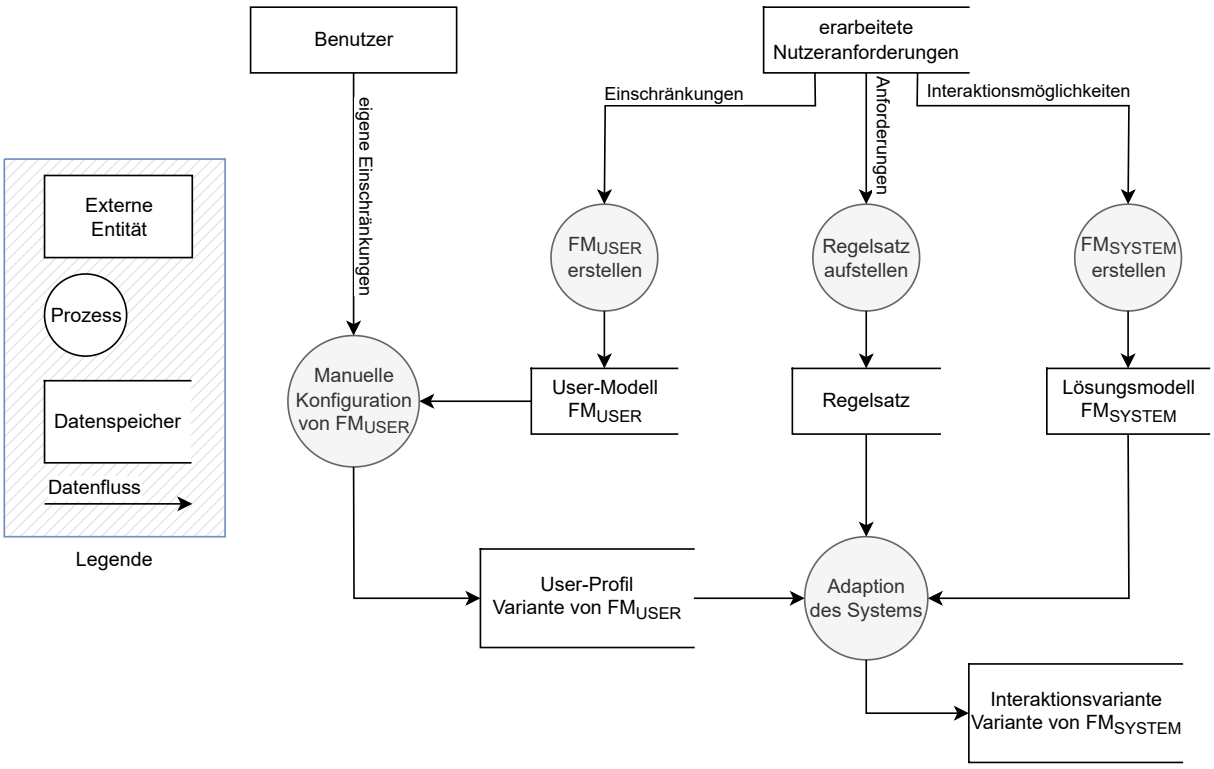


Abbildung 3.1 – Geplanter Adaptionsablauf der Interaktion

Abbildung 3.1 beschreibt, wie anhand der individuellen Anforderungen aufgrund unterschiedlicher Einschränkungen eine passende Interaktionsvariante ausgewählt wird. Die Grundlage hierfür sind die soeben herausgearbeiteten Nutzeranforderungen aus Abschnitt 3.1. Diese sind so definiert, dass sie anhand von Einschränkungen Interaktionsmöglichkeiten implizieren oder ausschließen; teilweise mit Betrachtung der Umgebungsvariablen. Die Einschränkungen lassen sich, wie in Abschnitt 2.2 definiert, kategorisieren, sodass ein Feature-Modell (FM_{USER}) konstruiert werden kann, welches als User-Modell dient. Auch die verschiedenen Möglichkeiten der Interaktion lassen sich somit in einem weiteren Feature-Modell (FM_{SYSTEM}) strukturiert sammeln, sodass

ein Lösungsmodell aufgebaut werden kann.
Die Anforderungen selbst lassen sich formal als Regeln der Form $E \rightarrow A$ darstellen, welche in einem Regelsatz gespeichert werden müssen.
E ist hierbei eine Aussage über die Nutzereinschränkung. Dies kann entweder eine einzelne Einschränkung und demnach ein Feature von FM_{USER} sein, eine Nutzervariable oder eine Kombination beider. Nutzervariablen sind Angaben, die der Benutzer festlegen kann, um das System an seine persönlichen Anforderungen besser anpassen zu können.

A stellt hier eine mögliche Aussage dar, die erfüllt werden muss, wenn E wahr ist. Das kann entweder eine konkrete Wahl oder der Ausschluss eines oder mehrerer Features des Lösungsmodells oder Vergleiche verschiedener Attribute der beiden Modelle sein. Formal wird A als ein Element der Menge M_A definiert, wobei M_A die kleinste Menge ist, für die die folgende Bedingungen gelten:

- Wenn F ein Feature aus $\text{FM}_{\text{SYSTEM}}$ ist, dann $F \in M_A$
- Wenn $F \in M_A$, dann $\neg F \in M_A$
- Wenn $F_1, F_2 \in M_A$, dann $(F_1 \wedge F_2) \in M_A$ und $(F_1 \vee F_2) \in M_A$
- Wenn $T_1, T_2 \in M_T$, dann $T_1 \circ T_2 \in M_A$ wobei $\circ \in \{=, <, \leq, >, \geq\}$

Die Menge M_T wiederum enthält alle Terme, die sich aus Attributwerten der Feature-Modelle und den natürlichen Zahlen bilden lassen und ist formal durch die kleinste Menge, die die folgenden Bedingungen erfüllt, definiert:

- Wenn T ein Attributwert aus FM_{USER} ist, dann $T \in M_T$
- Wenn T ein Attributwert aus $\text{FM}_{\text{SYSTEM}}$ ist, dann $T \in M_T$
- Wenn $T \in \mathbb{N}$ ist, dann $T \in M_T$
- Wenn $T_1, T_2 \in M_T$, dann $T_1 \circ T_2 \in M_T$ wobei $\circ \in \{+, -, \cdot, :\}$

Die Anforderungen an das System werden nun per manueller Eingabe der Einschränkungen des Benutzers an den Roboter übergeben. Dieser konfiguriert das Feature-Modell (FM_{USER}), damit das User-Profil als konfiguriertes Feature-Modell, welches alle Einschränkungen des Benutzers enthält, gespeichert werden kann. Aus diesem Profil wird im nächsten Schritt (Adaption des Systems) das Lösungsmodell konfiguriert. Für diese Konfiguration wird der aus den Nutzeranforderungen erstellte Regelsatz zur Hilfe genommen. Final wird somit ein konfiguriertes Feature-Modell $\text{FM}_{\text{SYSTEM}}$ ausgegeben, welches eine Variante der Interaktion darstellt.

3.5 Adaption durch Konfiguration des Feature-Modells der Interaktion

Nach dem vorgestellten MAPE-K-Prinzip, muss der Roboter zunächst den aktuellen Zustand beobachten und analysieren, ob die Regeln, die die Gültigkeit einer Konfiguration von $\text{FM}_{\text{SYSTEM}}$ im Bezug auf die aktuelle Konfiguration des Einschränkungs-Feature-Modells und der Umgebung

beschreiben, erfüllt sind. Wenn mindestens eine Regel nicht erfüllt wurde, muss versucht werden, eine andere Konfiguration von FM_{SYSTEM} zu finden.
Die für Menschen gut lesbaren Feature-Modelle müssen zur Verarbeitung durch den Roboter zunächst in Formeln übertragen werden. Felfernig et al. haben mögliche Konfigurationsprozesse von Feature-Modellen zusammengetragen [Fel+21]. Zunächst ist es notwendig, jedes Feature (und Attribut) mit seiner Domäne aufzulisten. Da Felfernig et al. normale Feature-Modelle verwendeten, ist bei ihnen die Domäne stets die Menge $\{0; 1\}$. Bei der Verwendung von kardinalitätsbasierten Modellen und Attributen variieren die Domänen jedoch. Als Domäne für Attribute wird deren definierter Bereich, wie in Abschnitt 2.3.2 vorgestellt, verwendet.
Eine Variante eines Feature-Modells wird durch Zuordnung jedes Features und Attributs zu einem, zur jeweiligen Domäne passenden Wert, gespeichert.

Die Abhängigkeiten zwischen Features oder Attributen, welche im Feature-Modell als Kanten dargestellt werden, müssen in der Form von aussagenlogischen Formeln gespeichert werden. Felfernig et al. haben zusammengefasst, wie sich unterschiedliche Beziehungen in Formeln abbilden lassen. [Fel+21] In Tabelle 3.1 sind diese Erkenntnisse gemeinsam mit eigenen Überlegungen zusammengefasst. Dabei wird jeder Abhängigkeit eine eindeutige ID zugeordnet.
Felfernig et al. schließen bei allen Feature-Modellen die leere Konfiguration aus (Abhängigkeit a_0). In dieser Arbeit symbolisiert jedoch die leere Variante von FM_{USER}, dass der Benutzer keine Einschränkungen hat. Daher wird diese Regel beim User-Modell nicht verwendet. Außerdem schreiben Felfernig et al. a_0 als $A = 1$. In dieser Arbeit wurde darauf geachtet, dass jede Formel entweder als Implikation ($\rightarrow$) oder als Äquivalenz ($\leftrightarrow$) dargestellt wird.
Abhängigkeit a_1 beschreibt ein obligatorisches Feature. Alternativ lässt sich die Formel $A \leftrightarrow B$ als $A \rightarrow B$ und $A \leftarrow B$ schreiben. Die erste Richtung fordert, dass aufgrund des obligatorischen Features bei der Auswahl von A ebenfalls B gewählt werden muss. Die andere Richtung ergibt sich dadurch, dass A der Eltern-Knoten von B ist und bei dessen Auswahl ebenfalls benötigt wird.
Optionale Features, wie sie in a_2 dargestellt sind, besitzen lediglich die Formel, die die soeben beschriebene Elternbeziehung beschreibt.
Die Abhängigkeit a_3^O zeigt eine OR-Oder-Gruppe, bei der mindestens ein Element gewählt werden muss. In der Notation von Felfernig et al., wie sie in a_3^O verwendet wurde, fehlt jedoch die Darstellung der Elternbeziehungen.
a_3^X zeigt die XOR-Oder-Gruppe, bei der genau ein Feature gewählt werden muss. Die Formel wird bereits bei zwei möglichen Features relativ lang. Bei einer Gruppe mit mehr möglichen Features oder der Verwendung einer Gruppenkardinalität würde sie noch länger und unübersichtlich werden.
Aus diesem Grund wurden die beiden Abhängigkeiten a_3^O und a_3^X zu a_3 zusammengefasst. Bei Wahl des Eltern-Knotens A muss die `Oder`-Funktion erfüllt werden. `Oder(liste: List, min: int, max: int)` ist *wahr*, wenn aus der Liste `liste` mindestens `min` und maximal `max` Features gewählt wurden. Durch diese Notation lassen sich Oder-Gruppen mit einer größeren Anzahl an Features übersichtlich notieren und zusätzlich Gruppenkardinalitäten abbilden.
Die Benötigt-Relation wird durch eine Implikation der Form $B \rightarrow C$, wie in a_4 zu sehen, gekennzeichnet.
Letztlich wird die Ausgeschlossen-Relation betrachtet. Felfernig et al. stellen diese durch eine negierte Und-Beziehung in der Form $\neg(B \wedge C)$ dar. Aufgrund der Konvention, dass eine Abhängigkeit stets entweder eine Implikation oder eine Äquivalenz sein muss, wurde diese Formel in

zwei Formeln der Form $B \rightarrow \neg C$ und $C \rightarrow \neg B$ übersetzt. Es sind hier zwei Formeln notwendig, da es sich bei der Ausgeschlossen-Relation um eine zweiseitige Beziehung handelt.

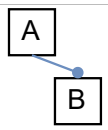
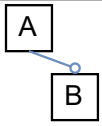
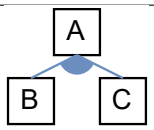
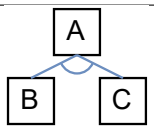
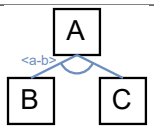
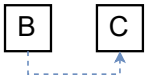
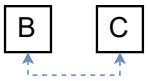
ID	Beschreibung	Abbildung	Formel
a_0	verpflichtete Wurzel		$\top \rightarrow A$
a_1	obligatorisches Feature		$A \leftrightarrow B$
a_2	optionales Feature		$B \rightarrow A$
a_3^O	OR-Oder-Gruppe		$A \rightarrow (B \vee C)$
a_3^X	XOR-Oder-Gruppe		$(B \leftrightarrow (\neg C \wedge A)\wedge$ $(C \leftrightarrow (\neg B \wedge A)$
a_3	Oder-Gruppe der Kardinalität $\langle a-b \rangle$		$A \rightarrow \text{Oder}([B, C], a, b)$ $B \rightarrow A$ $C \rightarrow A$
a_4	Benötigt-Relation		$B \rightarrow C$
a_5	Ausgeschlossen-Relation		$B \rightarrow \neg C$ $C \rightarrow \neg B$

Tabelle 3.1 – Notation der Abhängigkeiten eines Feature-Modells als Formeln

Somit ist es möglich, ein Feature-Modell maschinenlesbar zu notieren. Als nächsten Schritt ist es notwendig, anhand einer teilweisen Initialkonfiguration auf weitere Features eines Modells zu schließen. Hubaux hat die featureorientierte Konfiguration von Feature-Modellen vorgestellt. [Hub12] Er beschreibt sie als interaktiven Prozess, bei dem das Modell stets auf Korrektheit überprüft werden muss. Beginnend mit einer manuellen Entscheidung, werden initial Features gewählt oder nicht gewählt. Diese Initialkonfiguration erfolgt in dieser Arbeit durch Beachtung des Regelsatzes, der das Verhältnis zwischen Feature-Modell FM_{USER} (User-Modell) und Feature-Modell $\text{FM}_{\text{SYSTEM}}$ (Lösungsmodell) beschreibt. Hubaux schildert, dass anhand der manuell getrof-

fenen Entscheidung das System das Feature-Modell selbst weiter füllt. Dazu werden alle Regeln angewendet und dadurch ein Feature entweder für benötigt (1) oder für nicht verwendbar (0) erklärt. Da es durch die neu gewonnenen Erkenntnisse zu weiteren Schlussfolgerungen kommen kann, wird dieser Prozess solange wiederholt, bis keine neuen Features mehr gefunden wurden.

Sollte es zu dem Fall kommen, dass keine Konfiguration gültig ist, muss entweder eine Alternative gewählt werden oder dem Nutzer (beziehungsweise Pflegepersonal) mitgeteilt werden, dass der SAR für ihn nicht nutzbar ist.
Bei einer Anpassung des Interaktionskonzepts muss der Benutzer in geeigneter Weise darüber informiert werden. Dazu muss die neue Interaktion benutzt werden.

3.6 Zusammenfassung

In diesem Kapitel wurden zunächst die Anforderungen der Nutzer an SAR gesammelt und beschrieben, sodass diese in strukturierter Form dargestellt werden konnten. Durch die dort erworbenen Erkenntnisse und den Grundlagen aus Kapitel 2 wurde anschließend der Plan zur Erstellung zweier Variabilitätsmodelle vorgestellt, um unterschiedliche altersbedingte Einschränkungen und mögliche Interaktionskonzepte strukturiert und einfach verständlich zu präsentierten.
Die geplante Verbindung dieser beiden Modelle wurde darauf folgend durch Erläuterung eines Regelsatzes und des Ablauf der Adaption vorgestellt. Letztlich wurde die Umwandlung der grafischen Notation eines Feature-Modells und des Regelsatzes in aussagenlogische Formeln sowie die featureorientierte Konfiguration, bei der das Lösungsmodell schrittweise auf Grundlage des Regelsatzes und User-Profils konfiguriert wird, vorgestellt.
Als nächstes sollen die in diesem Kapitel vorgestellten Feature-Modelle und der Regelsatz passend zu den in Abschnitt 3.1 erarbeiteten Anforderungen konstruiert werden.

4 Konzept zur Erstellung eines Adaptionsmodells für die SAR-Interaktion

Nachdem die Anforderungen, die für eine Adaption der Interaktion von SAR während der Laufzeit notwendig sind, gesammelt wurden, soll sich dieses Kapitel damit beschäftigen, die zuvor genannten Modelle zu erzeugen. Dazu werden zwei Feature-Modelle aufgestellt. Eines zeigt die möglichen Einschränkungen der Nutzer. Das andere bildet jedes wählbare Interaktionsmodell, welches der Roboter anbietet, ab. Zur Verbindung beider Modelle muss auf Grundlage von Abschnitt 3.1 ein Regelsatz entworfen werden. Letztlich ist die Vorstellung eines Konfigurationsmechanismus notwendig, der darstellt, wie die Adaption abläuft.

4.1 Ausarbeitung der Anforderungen je Nutzer

In Abschnitt 3.1 wurden unterschiedliche Anforderungen von Menschen mit altersbedingten Einschränkungen erfasst. In diesem Teil soll es darum gehen, ein passendes User-Modell zu erzeugen, damit alle in Abschnitt 3.1 genannten Einschränkungen erfasst und gespeichert werden können. Dazu wird ein Feature-Modell, welches im Folgenden als FM_{USER} bezeichnet wird, erstellt, durch das Einschränkungen des Benutzers in einer Variante gespeichert werden können. Die grobe Struktur dieses Feature-Modells orientiert sich, wie bereits die Abschnitte 2.2 und 3.1, an dem ISO-Guide 71 und trennt Nutzereinschränkungen zunächst in *sensorisch*, *körperlich* und *kognitiv*. Das in Abbildung 4.1 entworfene Feature-Modell ordnet alle Einschränkungen, die in den Anforderungen genannt wurden, ihrer Kategorie zu. Dabei ist zu erwähnen, dass die leere Variante auch gültig ist. Für Menschen ohne Beeinträchtigungen ist diese Variante das passende User-Profil.
Einschränkungen des Sehens können entweder eine *reduzierte Sehschärfe*, eine *Farbfehlsichtigkeit* oder eine komplette *Erblindung* sein. Sollte ein Nutzer blind sein, so werden die anderen beiden Features der Gruppe durch eine *ausgeschlossen Relation* verboten. Sensorische Einschränkungen, die das korrekte Hören spezialisieren sind einerseits die *Schwerhörigkeit* und die komplette *Gehörlosigkeit*. Beide Features schließen sich logischerweise gegenseitig aus. Dies erfolgt durch die *ausgeschlossen Relation* und durch die Angabe der Gruppenkardinalität.
Körperliche Einschränkungen können entweder *leicht* oder *schwer* sein. Eine Variante, bei der sowohl leichte, als auch schwere motorische Einschränkungen gewählt wurden, ist ungültig. Außerdem kann der Benutzer angeben, dass er eine Sprachstörung hat, die nicht kognitiv, sondern motorisch bedingt ist.
Kognitive Einschränkungen kann der Nutzer weiter spezifizieren; muss er aber nicht. Daher wurde hier die Gruppenkardinalität $\langle 0 - 3 \rangle$ gewählt. Möglich wäre die Nennung von *Gedächtnisproblemen*, *Spracheinschränkungen*, deren Herkunft kognitiv zu belegen sind und eine *reduzierte Intelligenz*. Beeinträchtigungen der Sprache lassen sich erneut unterteilen in eine *kognitive Sprachstörung*, bei der der Nutzer kognitiv nicht in der Lage ist, korrekt zu sprechen. Andererseits ist es möglich, dass der Benutzer *gesprochene* oder *geschriebene* Sprache (oder beides) nicht korrekt

aufnehmen kann.

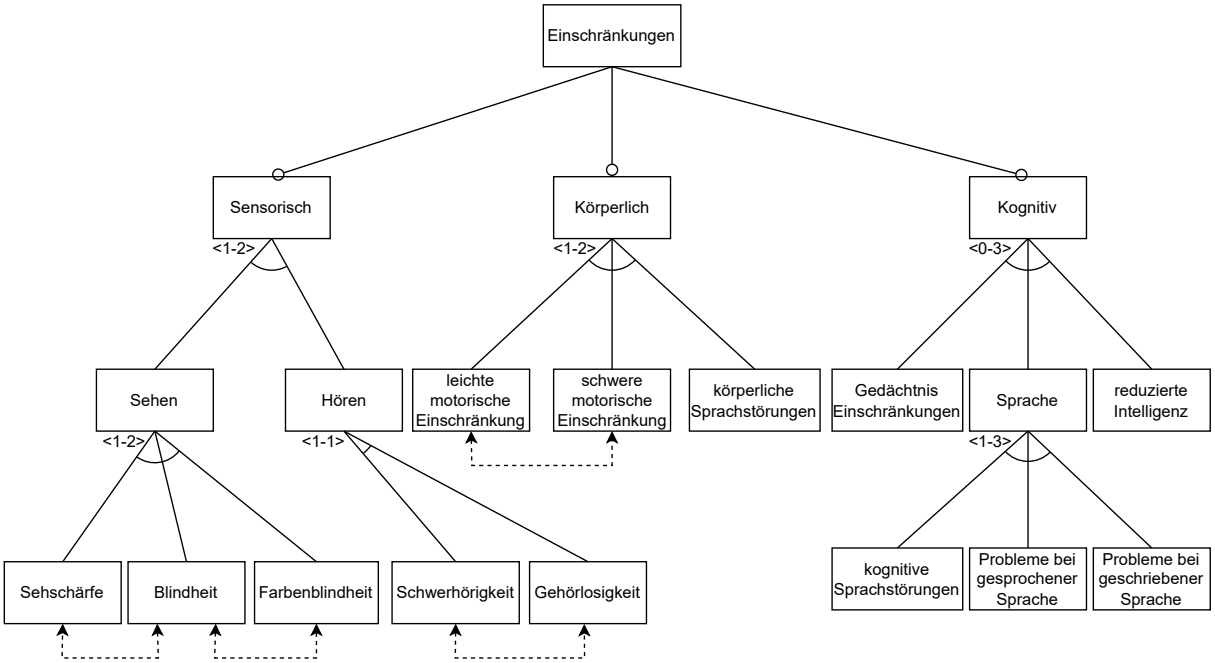


Abbildung 4.1 – Feature-Modell FM_{USER} zur Darstellung der Nutzereinschränkungen

4.2 Ausarbeitung der Interaktionsmöglichkeiten

Durch die in Abschnitt 3.1 zusammengefassten Anforderungen können Aussagen über mögliche Interaktionsvarianten von SAR getroffen werden. Alle dort erwähnten Möglichkeiten der Interaktion werden in einem großen Feature-Modell FM_{SYSTEM} strukturiert gespeichert. Eine Variante dieses Modells stellt dabei eine Möglichkeit dar, wie verschiedene Interaktionskonzepte gleichzeitig verwendet werden können.
Aus Gründen der Übersichtlichkeit wurde FM_{SYSTEM} in Teildiagramme unterteilt. Abbildung 4.2 zeigt, dass sich die Interaktion in die Gebiete *Eingabe*, *Ausgabe* und *Verarbeitung* teilen lässt, die jeweils in den Abbildungen 4.3, 4.4 und 4.5 zu sehen sind.

Das Feature-Modell FM_{SYSTEM} zeigt hierbei lediglich die in Abschnitt 3.1 erwähnten Interaktionsmöglichkeiten, die die **meisten** SAR abdecken. Es kann jedoch sein, dass es vereinzelt Roboter gibt, die weitere Features für die Interaktion besitzen, die in diesem Diagramm nicht dargestellt sind.

4.2.1 Interaktionmöglichkeiten der Eingabe

In Teildiagramm $FM^{Eingabe}_{SYSTEM}$ (zu sehen in Abbildung 4.3) werden unterschiedliche Möglichkeiten der *Eingabe* von Informationen dargestellt. Diese kann multimodal erfolgen. Durch Verwendung einer Oder-Gruppe an dieser Stelle im Modell, muss mindestens eine Art gewählt werden. Eine Kombination aus verschiedenen Eingabemöglichkeiten ist aber auch denkbar.

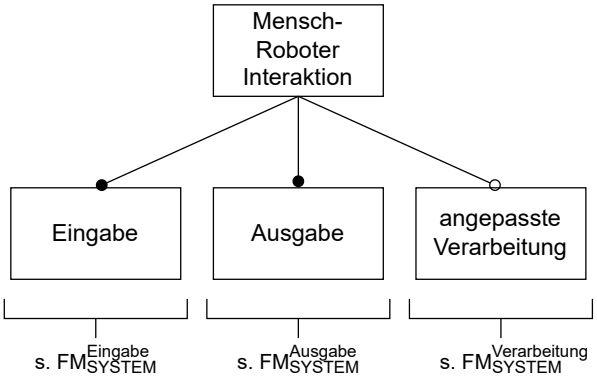


Abbildung 4.2 – Oberste Struktur von FM_{SYSTEM}

Die haptische Eingabe erfolgt durch die Navigation über einen *Touchscreen*. Hierbei muss die Größe der Navigationselemente auf dem Screen durch Angabe ihrer Fläche als Attribut gewählt werden. Somit kann die Eingabemöglichkeit problemlos vergrößert werden. Für Menschen mit starken körperlichen Einschränkungen kann es trotz vergrößertem Design aber zu Komplikationen führen, sodass die Spanne der möglichen Werte bis hoch zu $600\,cm^2$ geht, damit eine Einstellung gewählt werden kann, bei der der komplette Bildschirm bedeckt wird. Hierbei ist es unwichtig, an welcher Stelle der Bildschirm gedrückt wurde. Registriert wird lediglich, ob er vom Benutzer berührt wurde.
Das Attribut hat eine Abhängigkeit zum Attribut der Schriftgröße, welche im nächsten Abschnitt erläutert wird.

Die *akustische* Eingabe ist durch das obligatorische Feature *Eingabeart* unterteilt. Zum einen steht die Verarbeitung der vom Nutzer gesprochenen Sprache zur Verfügung. Da diese Eingabevariante für Menschen mit einer Sprachbehinderung nicht optimal nutzbar ist, kann auch zu einer nicht textbasierten Eingabe durch Summen gewechselt werden. Diese beiden Varianten schließen sich aber gegenseitig aus, sodass dies im Diagramm durch Alternative Features gekennzeichnet ist.
Das Feature *Bestätigung* ist optional und bedeutet, dass die akustische Eingabe ohne zusätzliche Bestätigung funktionieren kann. Dabei erkennt der Roboter automatisch, wann die Eingabe des Benutzers beendet ist. Bei Nutzern, die für ihre gesprochenen Eingaben länger brauchen und dabei möglicherweise längere Pausen machen, könnte die Eingabe zu früh beendet und falsch interpretiert werden. Um das zu verhindern könnte eine Bestätigung erwartet werden. So ist es möglich, die Eingabe durch ein bestimmtes Keyword oder durch Berühren des Bildschirms zu beenden. Für die zweite Art der Bestätigung muss jedoch der Touchscreen aktiviert sein. Auch eine Kombination aus beiden Bestätigungen ist denkbar.

Eine weitere Eingabemöglichkeit, die sich gerade für Menschen mit körperlichen Einschränkungen anbietet, ist die Verwendung von *Gesten*. Bei *motorischen Gesten* ist keine so genaue Koordination der Arme und Finger notwendig, wie bei der Eingabe über einen Touchscreen. Sollte es dennoch zu Problemen führen, kann durch die Verwendung eines *Eye-Trackers* die Bewegung der Augen verfolgt werden und so Informationen aufgenommen werden. Da es sich hier um eine

Oder-Gruppe handelt, ist auch eine Verwendung beider Gestenarten möglich.

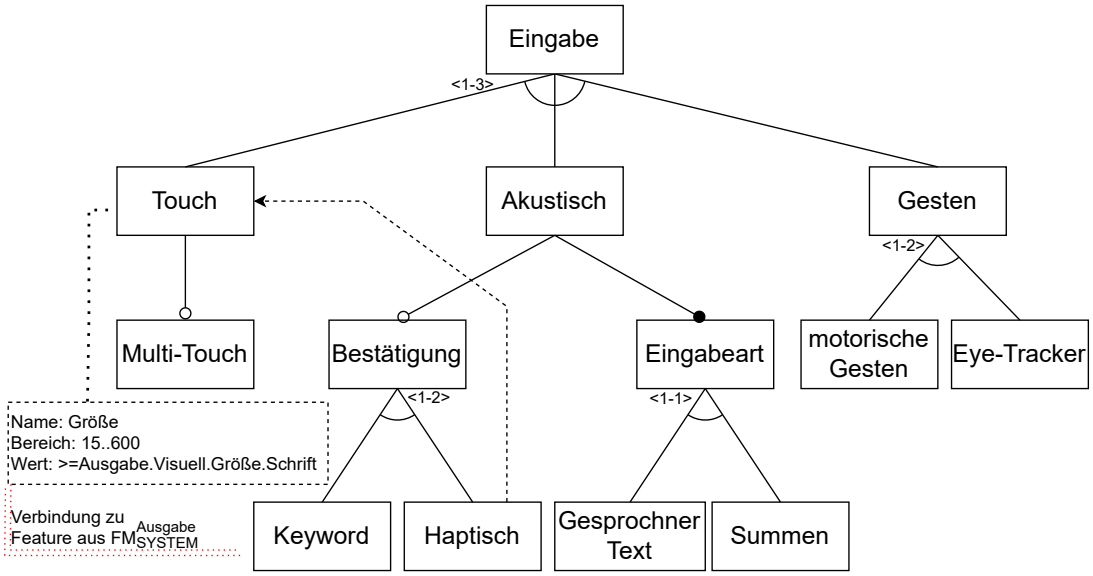


Abbildung 4.3 – $FM^{Eingabe}_{SYSTEM}$ zur Beschreibung der Interaktion zwischen Mensch und Roboter

4.2.2 Interaktionmöglichkeiten der Ausgabe

Auch die Übergabe der Informationen vom Roboter an den Menschen (*Ausgabe*) ist multimodal möglich. Deren Teildiagramm ist in Abbildung 4.4 zu sehen. Minimal muss eine Ausgabemöglichkeit gewählt werden. Eine Kombination aus den drei angebotenen Möglichkeiten ist aber machbar.

Für blinde oder sehbehinderte Nutzer ist die Ausgabe über eine *Braillezeile* denkbar. Dort wird der auszugebene Text in Brailleschrift erstellt, sodass der Benutzer mit Hilfe seines Fingers den Text lesen kann.

Durch die *akustische* Ausgabe wird die Weitergabe von Informationen durch gesprochenen Text oder Töne beschrieben. Die *Ausgabeart* wird im Modell durch eine Oder-Gruppe dargestellt. So ist es möglich, zu einer unkomplizierten Ausgabe in Form bestimmter Töne zu wechseln, wenn die Sprachausgabe für den Benutzer zu komplex wird. Eine hybride Verwendung beider Varianten ist aber auch denkbar. Im Falle einer Sprachausgabe kann zwischen einer computergenerierten *Text To Speech* (*TTS*)-Stimme und einer eingesprochenen, *natürlichen* Stimme gewählt werden. Die akustische Ausgabe umfasst des Weiteren drei Attribute. Einerseits wird der *Frequenzbereich* durch Angabe der Minimal- und Maximalfrequenz gespeichert. Diese können Werte zwischen 0 und 8000 Hz annehmen. Die Minimalfrequenz muss logischerweise kleiner oder gleich der Maximalfrequenz sein. Die *Ausgabelautstärke* des Systems wird in Phon im Bereich zwischen 10 und 100 gespeichert.

Letztlich steht die *visuelle* Ausgabe über einen Monitor zur Verfügung. Unterteilt wird diese durch zwei obligatorische Features. Ein Feature beschreibt das *Aussehen* der Ausgabe. Über ein Attribut wird hier der Wert für die Helligkeit angegeben. Prozentual darf diese zwischen einem Prozent

(sehr dunkel) und 100 % (sehr hell) liegen.
Für Menschen mit Sehbehinderungen ist ein *hoher Kontrast* hilfreich, der optional gewählt werden kann. Für farbenblinde Nutzer ist eine optionale *Modifikation der Farben* praktisch. Diese wird durch eine Oder-Gruppe erneut unterteilt. Zum einen steht die Auswahl von gut unterscheidbaren Farben zur Verfügung. Alternativ kann auch die *Ergänzung von farblichen Kennzeichnungen durch strukturelle Eigenschaften* gewählt werden. Dabei wird ein rot gefärbter Strich beispielsweise gepunktet dargestellt. Beide Varianten können aber auch gleichzeitig verwendet werden.
Neben dem Aussehen der Ausgabe kann auch ihre *Größe* angepasst werden. Dazu steht auf der einen Seite die Anpassung der Schriftgröße und auf der anderen Seite die Variation des Zeilenabstandes zur Verfügung. Die Schriftgröße wird durch die Punktzahl, welche zwischen 10 und 70 Pt liegen darf, beschrieben. Der Zeilenabstand wird als Prozentsatz der Schriftgröße gespeichert und darf zwischen 50 % und 500 % liegen. Wenn der Touchscreen als Eingabemöglichkeit gewählt wurde, muss dessen Größe mindestens so groß, wie die Schriftgröße in Punkten sein.

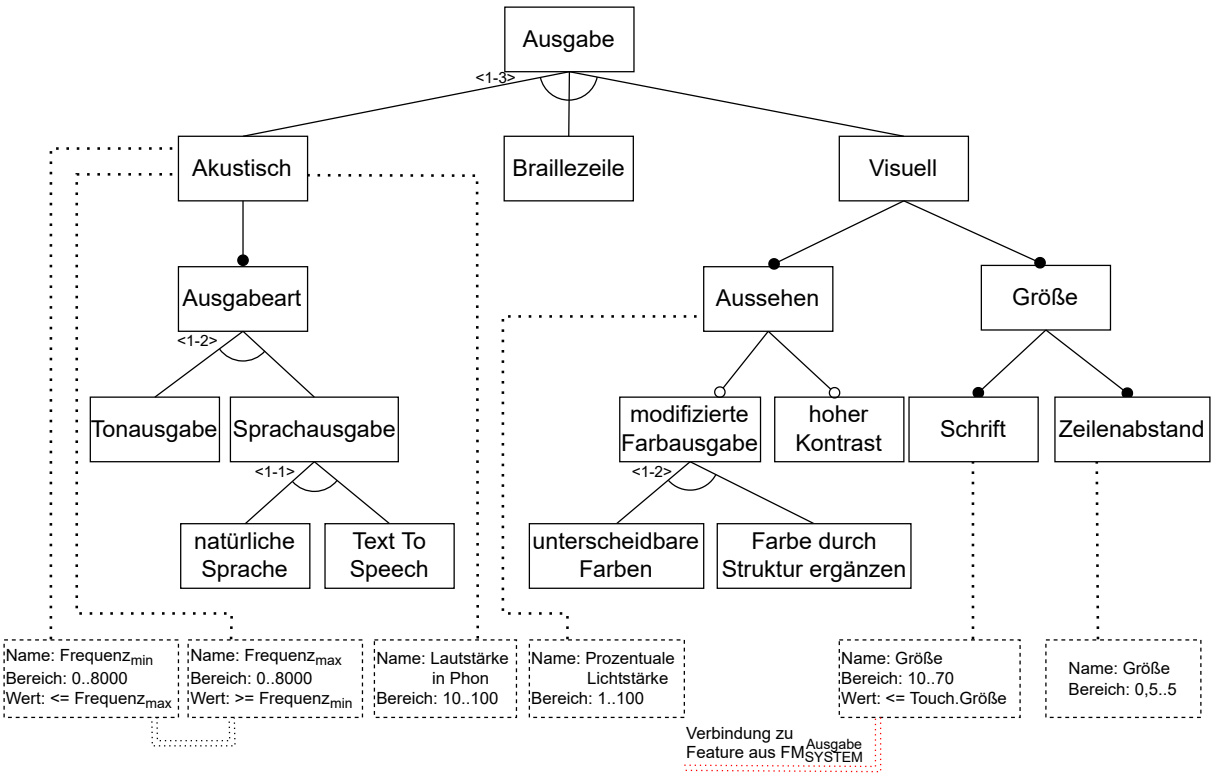


Abbildung 4.4 – FM$^{Ausgabe}_{SYSTEM}$ zur Beschreibung der Interaktion zwischen Roboter und Mensch

4.2.3 Interaktionmöglichkeiten der Verarbeitung

Neben der Ein- und Ausgabe wird durch dieses Feature die Art der Verarbeitung und Erzeugung der auszugebenden Informationen definiert. Das Teildiagramm FM$^{Verarbeitung}_{SYSTEM}$, welches in Abbildung 4.5 zu sehen ist, beschreibt, wie die Informationen für den Nutzer aufbereitet werden, sodass sie für ihn verständlich sind.
Für Menschen mit kognitiven Einschränkungen sollte die Ausgabe weniger komplex aufgebaut sein. So ist es vorteilhaft, eine *leichte Sprache* zu verwenden. Das kann entweder die Verwendung

von *kurzen Sätzen* oder die *Vermeidung komplizierter Begriffe* bedeuten. Dialoge können aber auch so aufgebaut sein, dass der SAR die Nutzereingabe stets wiederholt und *bestätigt*. Komplizierte Aktionen mit mehreren Schritten können bei Bedarf so geändert werden, dass diese *schrittweise* ausgeführt werden. Zusätzlich gibt es die Variante, dass der Roboter, sofern es möglich ist, die Fragen so umformuliert, dass der Nutzer lediglich mit *Ja* oder *Nein* antworten kann.
Für Menschen mit Gedächtnisproblemen bietet der SAR die Option an, dass er die Möglichkeiten der Schnittstelle zu jeder Zeit *erklärt*, damit der Nutzer weiß, wie er interagieren kann.

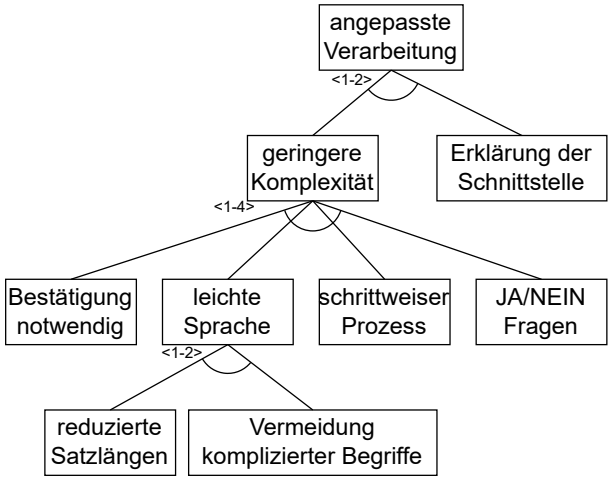


Abbildung 4.5 – $FM_{SYSTEM}^{Verarbeitung}$ zur Beschreibung, wie Interaktion verarbeitet wird

4.3 Entwicklung eines Regelsatzes

Auf Grundlage der in Abschnitt 3.1 erarbeiteten Anforderungen wird in diesem Abschnitt ein Regelsatz aufgestellt, der jede Anforderung in eine oder mehrere Regeln abbildet. Zur Darstellung der Regeln wurden Implikationen verwendet. In Tabelle 4.1 sind die Regeln c_1 bis c_{41} des Regelsatzes $C_{ANFORDERUNGEN}$ zu sehen, der die erarbeiteten Anforderungen in eine logische, maschinenlesbare Form bringt.

Zunächst wird auf die Notation der Regeln eingegangen. Neben den aussagenlogischen Ausdrücken „$\top$“ für *True*, „$\rightarrow$“ für die Implikation, „$\wedge$“ für eine logische Und-Verknüpfung und „$\neg$“ für die Negation, treten die Zeichen „$\geq$“, „$\leq$“ und „$=$“ für größer-gleich, kleiner-gleich und Gleichheitsbeziehungen zwischen Attribut- und Variablenwerten und „+“ für die Addition auf. Axiome in den Formeln, die innerhalb von eckigen Klammern stehen, symbolisieren Features oder Attribute aus dem Feature-Modell FM_{SYSTEM}. Die übrigen Axiome stammen entweder aus dem Feature-Modell FM_{USER} oder sind Nutzervariablen, die der Nutzer an seine persönlichen Bedürfnisse anpassen kann. Für diese Variablen sind in Tabelle 4.2 Standardbelegungen festgelegt, die verwendet werden sollen, sofern der Nutzer keine eigenen Angaben gemacht hat. Da sich Features, wie *Akustisch* aus den Feature-Modellen $FM_{SYSTEM}^{Eingabe}$ und $FM_{SYSTEM}^{Ausgabe}$ bei reiner Angabe des Feature-Namens nicht auseinanderhalten lassen, wird an einigen Stellen der Pfad zum Feature angegeben. So meint das Axiom [Ausgabe, Akustisch] das Feature *Akustisch* aus dem Feature-Modell $FM_{SYSTEM}^{Ausgabe}$ von Abbildung 4.4.

ID	Formel	Prio	Nr.
c_1	$\top \rightarrow$ [Lautstärke] $\geq$ Lautstärke$^{\text{min}}_{\text{USER}}$	1	1.1.
c_2	$\top \rightarrow$ [Lautstärke] ≤ 100 Phon	1	1.2.
c_3	$\top \rightarrow$ [Lautstärke] $\geq$ Lautstärke$_{\text{UMGEBUNG}}$ + Lautstärke$^{\text{add}}_{\text{USER}}$	1	1.3.
c_4	$\top \rightarrow$ [Tonhöhe$_{\text{min}}$] $\geq$ Tonhöhe$^{\text{min}}_{\text{USER}}$	1	1.4.
c_5	$\top \rightarrow$ [Tonhöhe$_{\text{max}}$] $\leq$ Tonhöhe$^{\text{max}}_{\text{USER}}$	1	1.4.
c_6	$\top \rightarrow$ [Eingabe, Touch, Größe] $\geq 3\,\text{cm}^2$	1	1.5.
c_7	Sehschwäche $\rightarrow$ [Eingabe, Akustisch] $\wedge$ [Ausgabe, Akustisch]	1	2.1.1.
c_8	Sehschwäche $\rightarrow$ [Schriftgröße] $\geq$ Schriftgröße$^{\text{min}}_{\text{USER}}$	1	2.1.2.
c_9	Sehschwäche $\rightarrow$ [Zeilenabstand] $\geq$ Zeilenabstand$^{\text{min}}_{\text{USER}}$	1	2.1.3.
c_{10}	Blindheit $\rightarrow$ [Eingabe, Akustisch] $\wedge$ [Ausgabe, Akustisch]	1	2.2.1.
c_{11}	Blindheit $\wedge$ Braille$_{\text{USER}} = 1 \rightarrow$ [Ausgabe, Braillezeile]	1	2.2.2.
c_{12}	Blindheit $\rightarrow \neg$[Ausgabe, Visuell]	0,5	2.2.3.
c_{13}	Blindheit $\rightarrow \neg$[Eingabe, Touch]	0,5	2.2.4.
c_{14}	Farbfehlsichtigkeit $\wedge$ [Ausgabe, Visuell] $\rightarrow$ [unterscheidbare Farben]	1	2.3.1.
c_{15}	Farbfehlsichtigkeit $\wedge$ [Ausgabe, Visuell] $\rightarrow$ [Ergänzung durch Struktur]	1	2.3.2.
c_{16}	Schwerhörigkeit $\rightarrow$ [Lautstärke] $\geq$ Lautstärke$^{\text{min}\prime}_{\text{USER}}$	1	3.1.1.
c_{17}	Schwerhörigkeit $\rightarrow$ [Lautstärke] $\geq$ Lautstärke$_{\text{UMGEBUNG}}$ + Lautstärke$^{\text{add}\prime}_{\text{USER}}$	1	3.1.2.
c_{18}	Schwerhörigkeit $\rightarrow$ [Tonhöhe$_{\text{min}}$] $\geq$ Tonhöhe$^{\text{min}\prime}_{\text{USER}}$	1	3.1.3.
c_{19}	Schwerhörigkeit $\rightarrow$ [Tonhöhe$_{\text{max}}$] $\leq$ Tonhöhe$^{\text{max}\prime}_{\text{USER}}$	1	3.1.3.
c_{20}	Schwerhörigkeit $\rightarrow$ [Ausgabe, Visuell]	1	3.1.4.
c_{21}	Schwerhörigkeit $\rightarrow$ [Eingabe, Gesten]	1	3.1.5.
c_{22}	Gehörlosigkeit $\rightarrow$ [Eingabe, Touch] $\wedge$ [Ausgabe, Visuell]	1	3.2.1.
c_{23}	Gehörlosigkeit $\rightarrow$ [Eingabe, Gesten]	1	3.2.2.
c_{24}	Gehörlosigkeit $\rightarrow \neg$[Ausgabe, Akustisch]	0,5	3.2.3.
c_{25}	leichte körperl. Einschr. $\rightarrow$ [Eingabe, Touch, Größe] $\geq 50\,\text{cm}^2$	1	4.1.1.
c_{26}	leichte körperl. Einschr. $\rightarrow$ [Eingabe, Gesten]	1	4.1.2.
c_{27}	leichte körperl. Einschr. $\rightarrow \neg$[Eingabe, Touch, Multi-Touch]	0,5	4.1.3.
c_{28}	schwere körperl. Einschr. $\rightarrow$ [Eingabe, Touch, Größe] = Bildschirm$_{\text{Größe}}$	1	4.2.1.
c_{29}	schwere körperl. Einschr. $\rightarrow$ [Eingabe, Akustisch]	1	4.2.2.
c_{30}	schwere körperl. Einschr. $\rightarrow$ [Ausgabe, Akustisch]	1	4.2.2.
c_{31}	schwere körperl. Einschr. $\rightarrow \neg$[Eingabe, Touch, Multi-Touch]	0,5	4.2.3.
c_{32}	kognitive Einsch. $\rightarrow$ [leichte Sprache]	1	5.1.1.
c_{33}	kognitive Einsch. $\rightarrow$ [schrittweise Dialogverarbeitung]	1	5.1.2.
c_{34}	kognitive Einsch. $\rightarrow$ [Ja/Nein-Fragen]	1	5.1.3.
c_{35}	Gedächtnisstörung $\rightarrow$ [Bestätigung notwendig]	1	5.2.1.
c_{36}	Gedächtnisstörung $\rightarrow$ [Erklärung der UI]	1	5.2.2.
c_{37}	Sprachstörung $\rightarrow \neg$[Eingabe, Akustisch]	0,5	6.1.
c_{38}	Probl. bei gespr. Sprache $\rightarrow \neg$[Ausgabe, Akustisch]	0,5	6.2.
c_{39}	Probl. bei gespr. Sprache $\wedge$ [Ausgabe, Akustisch] $\rightarrow$ [natürliche Sprache]	1	6.3.
c_{40}	Probleme bei geschriebener Sprache $\rightarrow \neg$[Ausgabe, Visuell]	0,5	6.4.
c_{41}	Braille$_{\text{USER}} = 0 \rightarrow \neg$[Ausgabe, Braillezeile]	0,5	

Tabelle 4.1 – Regelsatz $C_{\text{ANFORDERUNGEN}}$ zur Verbindung beider Feature-Modelle

Da der Regelsatz im Prinzip lediglich eine Übersetzung der Anforderungen in logische Formeln ist, wird an dieser Stelle nicht im Detail auf jede einzelne Regel eingegangen, sondern nur Besonderheiten erwähnt. Tabelle 4.1 enthält in der Spalte „Nr." die zu der Regel gehörende Anforderung aus Abschnitt 3.1.
Die Formeln c_1 bis c_6 beschreiben die Anforderungen 1.1. bis 1.5., die für alle Nutzer, unabhängig ihrer Einschränkungen, gelten müssen. Daher ist der linke Teil der Implikation „$\top$", sodass der rechte Teil immer erfüllt werden muss.
Regel c_{11} besitzt auf der linken Seite der Implikation eine Und-Beziehung. Für diese Regel muss sowohl das Feature *Blindheit* in FM_{USER} ausgewählt als auch die Nutzervariable $\text{Braille}_{\text{USER}}$, die angibt, dass der Nutzer Braille lesen kann, auf `1` gesetzt worden sein, damit die rechte Seite der Regel benötigt wird.
Die letzte Regel c_{41} ist keiner Anforderung direkt zugeordnet. Sie beschreibt, dass für Nutzer, die keine Brailleschrift lesen können, die Braillezeile deaktiviert sein sollte.
Die Regel c_{38} fordert für Menschen mit Problemen bei gesprochener Sprache die Deaktivierung der akustischen Ausgabe. Sollte es jedoch zu dem Fall kommen, dass Regel c_{38} aufgrund einer anderen Regel ignoriert wird, fordert Regel c_{39}, dass eine natürliche Stimme statt TTS verwendet wird. Die Priorität der meisten Regeln liegt bei `1`. Das bedeutet, dass diese bei der Konfiguration auf jeden Fall gewählt werden müssen. Für die Regeln, die ein bestimmtes Feature ausschließen und die Form $A \rightarrow \neg B$ aufweisen, wurde die Priorität auf `0,5` reduziert. Genauere Details zur Priorität und dem Konfigurationsalgorithmus liefert Abschnitt 4.4. Konkret betrifft dies im Regelsatz $\text{C}_{\text{ANFORDERUNGEN}}$ die Regeln c_{12}, c_{13}, c_{24}, c_{27}, c_{31}, c_{37}, c_{38}, c_{40} und c_{41}.
Die Regeln c_{16} bis c_{19} ähneln den Regeln c_1 und c_3 bis c_5. Sie sind jedoch nur aktiv, wenn der Nutzer angegeben hat, dass er schwerhörig ist. Sobald dies der Fall ist, werden andere Nutzervariablen (gekennzeichnet durch $\prime$) für die vier Ungleichungen verwendet. Das gleiche gilt für die Regeln c_6 und c_{25}.

ID	Formel	Nr.
c_A	$\text{Lautstärke}^{\text{min}}_{\text{USER}} = 50\,\text{Phon}$	1.1.
c_B	$\text{Lautstärke}^{\text{add}}_{\text{USER}} = 5\,\text{Phon}$	1.3.
c_C	$\text{Tonhöhe}^{\text{min}}_{\text{USER}} = 100\,\text{Hz}$	1.4.
c_D	$\text{Tonhöhe}^{\text{max}}_{\text{USER}} = 8000\,\text{Hz}$	1.4.
c_E	$\text{Schriftgröße}^{\text{min}}_{\text{USER}} = 21$	2.1.2.
c_F	$\text{Zeilenabstand}^{\text{min}}_{\text{USER}} = 120\%$	2.1.3.
c_G	$\text{Braille}_{\text{USER}} = False$	2.2.2.
c_H	$\text{Lautstärke}^{\text{min}\prime}_{\text{USER}} = 90\,\text{Phon}$	3.1.1.
c_I	$\text{Lautstärke}^{\text{add}\prime}_{\text{USER}} = 10\,\text{Phon}$	3.1.2.
c_J	$\text{Tonhöhe}^{\text{min}\prime}_{\text{USER}} = 200\,\text{Hz}$	3.1.3.
c_K	$\text{Tonhöhe}^{\text{max}\prime}_{\text{USER}} = 2000\,\text{Hz}$	3.1.3.
c_L	$\text{Bildschirm}_{\text{Größe}} = 300\,\text{cm}^2$	4.2.1.

Tabelle 4.2 – Standardwerte für die in Tabelle 4.1 verwendeten Nutzervariablen

4.4 Ausarbeitung eines Konfigurationsmechanismus

Abbildung 4.6 zeigt den Kreislauf der Adaption und die damit verbundene Konfiguration des Lösungsmodells auf Grundlage des User-Profils.
Dabei wird zunächst geprüft, ob die Einschränkungen des aktiven Nutzers dem SAR bekannt sind. Sollte dies noch nicht der Fall sein, muss das User-Profil durch Konfiguration des User-Modells FM_{USER} zunächst erstellt werden.
Mit einer zum Nutzer passenden Variante dieses Feature-Modells wird im nächsten Schritt der komplette Regelsatz $C_{ANFORDERUNGEN}$, der die Verbindung zwischen FM_{USER} und FM_{SYSTEM} beschreibt, angewendet. Zur besseren Lesbarkeit wird dieser Regelsatz im Diagramm durch C_{ANFO} abgekürzt. Dabei werden für jede aussagenlogische Formel aus dem Regelsatz die bekannten Variablen eingesetzt. Eine Variable ist bekannt, wenn sie entweder aus der eben erwähnten Variante von FM_{USER} stammt, eine Aussage über die Umgebung beschreibt oder eine Nutzervariable ist. Anhand der teilweise ausgefüllten Formeln muss anschließend eine Schlussfolgerung über die restlichen Variablen erfolgen, damit die Regel als *wahr* abgebildet werden kann.
Eine Regel kann entweder eine Implikation der Form $A \rightarrow B$ oder eine Äquivalenz der Form $A \leftrightarrow B$ sein. Bei der Implikation muss B *wahr* sein, sobald A *wahr* ist. Sollte A *falsch* sein, so ist der Wert von B nicht relevant. Tabelle 4.3 zeigt diesen Zusammenhang. Bei der Äquivalenz müssen entweder beide Seiten der Regel *wahr* oder beide Seiten *falsch* sein, wie es Tabelle 4.4 zeigt. Der Regelsatz $C_{ANFORDERUNGEN}$ enthält ausschließlich Implikationen. In Abschnitt 3.5 wurde jedoch gezeigt, wie sich die Abhängigkeiten innerhalb von Feature-Modellen als Regeln darstellen lassen. Für den, in Abbildung 4.6 dargestellten Adaptionsalgorithmus, stellt C_{SYSTEM} den Regelsatz dar, der die Abhängigkeiten innerhalb des Feature-Modells FM_{SYSTEM} beschreibt.

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Tabelle 4.3 – Wahrheitstabelle Implikation

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Tabelle 4.4 – Wahrheitstabelle Äquivalenz

Nach der Anwendung des Regelsatzes $C_{ANFORDERUNGEN}$ ergeben sich Schlussfolgerungen, die zur Konfiguration weiterer Features und Attribute genutzt werden können und in der Liste $L_{INITIAL}$ gespeichert werden. In dieser Liste wird demnach eine Initialkonfiguration für FM_{SYSTEM} gespeichert, indem Features des Lösungsmodell als *zwingend benötigt* (`1`) und *nicht verwendbar* (`0`) gekennzeichnet werden. Zu Features, die nicht in dieser Liste enthalten sind, kann noch keine Aussage getroffen werden.
Sollte bereits in dieser Liste ein Widerspruch enthalten sein, wird (wenn möglich) die Regel mit der niedrigsten Priorität ungleich `1` aus $C_{ANFORDERUNGEN}$ entfernt und $L_{INITIAL}$ durch Anwendung des bearbeiteten Regelsatzes $C_{ANFORDERUNGEN}$ gefüllt. Dies geschieht solange, bis entweder kein Widerspruch mehr enthalten ist und der Algorithmus fortgesetzt werden kann oder keine Regel mehr gelöscht werden kann und die Adaption ohne Erfolg abbricht.
Der nächste Schritt im Algorithmus ist die, von Hubaux [Hub12] beschriebene, teilweise Konfiguration von FM_{SYSTEM} durch die Initialkonfiguration $L_{INITIAL}$. In einem zyklischen Verfahren wird

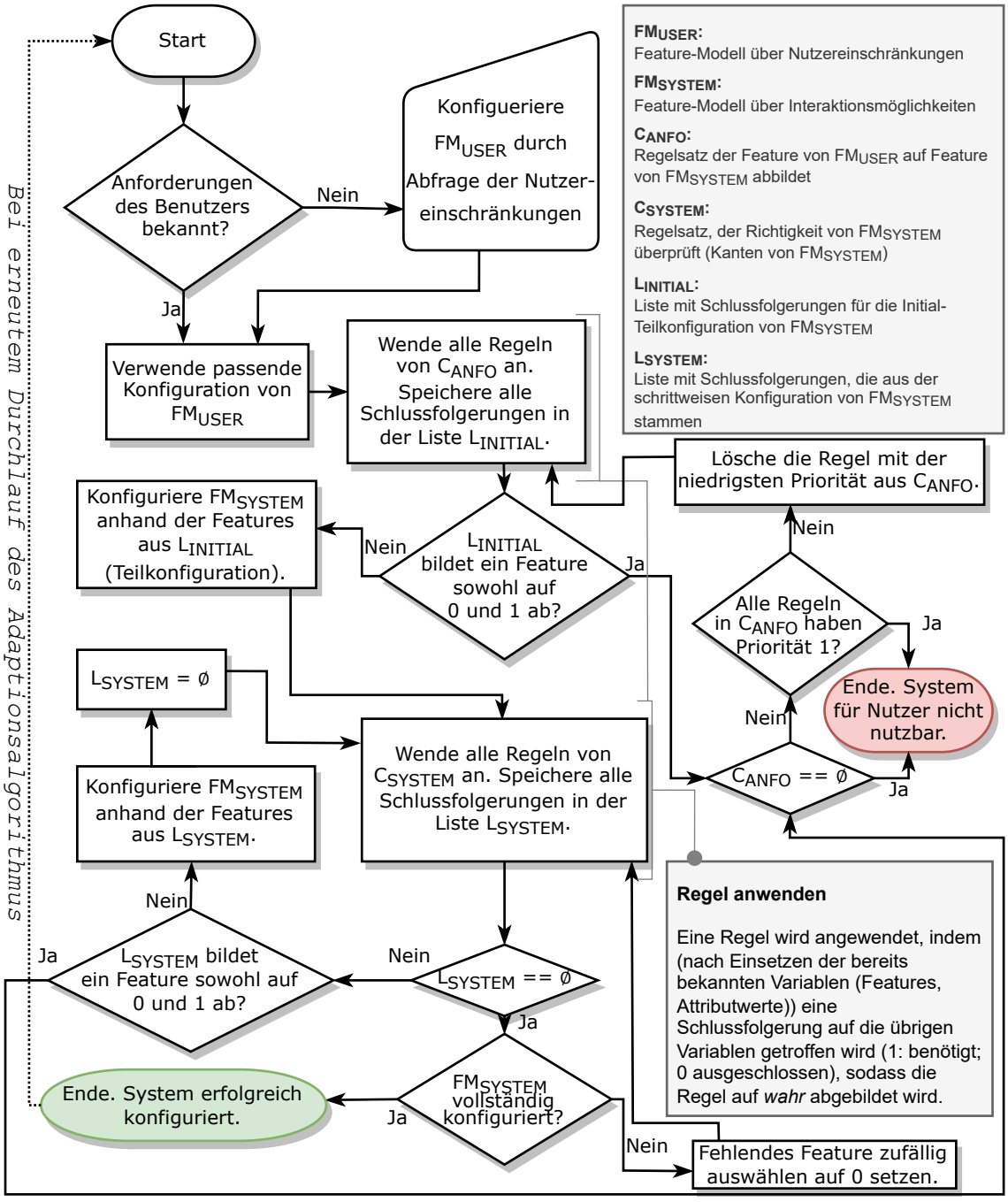


Abbildung 4.6 – Ablaufdiagramm der Adaption

nun der Regelsatz C_{SYSTEM} (der die Korrektheit von FM_{SYSTEM} beschreibt) angewendet. Im gleichen Verfahren, wie oben beschrieben, werden auf Grundlage der bereits gewählten Features Schlussfolgerungen getroffen, damit jede Regel auf *wahr* abgebildet werden kann. Diese Folgerungen werden in einer weiteren Liste namens L_{SYSTEM} gespeichert. Vor der weiteren Konfiguration wird erneut geprüft, ob es in L_{SYSTEM} einen Widerspruch gibt. Sofern dies der Fall ist, wird im gleichen Verfahren, wie oben beschrieben, versucht, die Regel mit der niedrigsten Priorität aus $C_{ANFORDERUNGEN}$ zu löschen und erneut bei der Anwendung des modifizierten Regelsatzes $C_{ANFORDERUNGEN}$ begonnen.
Sollte die Auswahl in Ordnung sein, wird das Lösungsmodell entsprechend konfiguriert und die Liste L_{SYSTEM} geleert. Da sich dadurch Änderungen ergeben können, durch die es bei der Anwendung von Regelsatz C_{SYSTEM} zu weiteren Schlussfolgerungen kommen kann, wird dieses Verfahren so lange wiederholt, bis die Liste L_{SYSTEM} direkt nach der Regelsatz-Anwendung keine neuen Schlussfolgerungen mehr enthält.
Sollte das Feature-Modell nun noch nicht vollständig konfiguriert worden sein, wird ein zufälliges, noch nicht konfiguriertes Feature ausgewählt und auf `0` konfiguriert. Dabei müssen aber alle Regeln der beiden Regelsätze $C_{ANFORDERUNGEN}$ und C_{SYSTEM} auf *wahr* abgebildet werden. Im Anschluss muss das soeben vorgestellte Verfahren zur Anwendung des Regelsatzes C_{SYSTEM} erneut durchgeführt werden. Diese zufällige Auswahl mit angeschlossener Anwendung des Regelsatzes wird so lange wiederholt, bis alle Features erfolgreich konfiguriert wurden.

4.5 Zusammenfassung

In diesem Kapitel wurde zunächst ein Feature-Modell vorgestellt, welches mögliche Einschränkungen der Nutzer beschreibt. Dieses wird als User-Modell verwendet. Eine auf den Nutzer passende, konfigurierte Version des Feature-Modells stellt das User-Profil dar. Auch die Interaktionsmöglichkeiten mit dem SAR werden in Form eines Features-Modells gespeichert, welches als Lösungsmodell bezeichnet wird. Beide Feature-Modelle lassen sich durch Darstellung ihrer Kanten in Form von aussagenlogischen Regeln maschinenlesbar darstellen.
Die Anforderungen, die ältere Menschen mit bestimmten Einschränkungen an die Interaktion haben, werden in Form eines Regelsatzes, der zwischen den beiden vorgestellten Feature-Modellen vermittelt, beschrieben.
Die Konfiguration des Lösungsmodells um eine passende Variante der Interaktion zu finden erfolgt durch featureorientierte Konfiguration, bei der zunächst anhand des Regelsatzes auf eine Initialkonfiguration auf Grundlage des User-Profils geschlossen wird. Die restlichen Features, die so noch nicht konfiguriert wurden, werden durch logische Schlüsse und letztlich zufällig gesetzt.
Um das erarbeitete Konzept auf Korrektheit zu überprüfen ist es jetzt notwendig, einen Prototyp zu entwickeln, durch den Testfälle geprüft werden können.

5 Prototypische Umsetzung des Konzepts

Im vorherigen Abschnitt wurde das entwickelte Konzept zur Adaption der Interaktion von SAR während der Laufzeit vorgestellt. Zur Korrektheitsprüfung dieses Konzepts wird in diesem Abschnitt eine prototypische Umsetzung durchgeführt.

5.1 Auswahl der verwendeten Bausteine

Ziel dieses Abschnittes ist es, passende Bausteine für die Entwicklung des Prototyps zu finden. Der Prototyp soll bei dieser Arbeit ein Sprachassistent werden. Bereits in der Aufgabenstellung wurde hierfür RASA als Dialogmanagement vorgeschlagen. Mit RASA lässt sich auf einfache Art und Weise ein Chatbot entwickeln. Die Ein- und Ausgabe an RASA erfolgt jedoch über die Tastatureingabe und Textausgabe über den Bildschirm. Zur Übersetzung der vom Nutzer gesprochenen Inhalte und des von RASA ausgegebenen Textes ist eine Agentenarchitektur notwendig.
Zur Erstellung des Prototyps wurden zwei unterschiedliche Architekturen betrachtet und deren Vor- und Nachteile gesammelt.

MyCroft.ai ist ein Open-Source-Sprachassistent. Die Open-Source-Verfügbarkeit und Unabhängigkeit von externen Firmen ist ein klarer Vorteil. Ebenso ist es möglich, MyCroft komplett ohne Internetverbindung zu nutzen. Es gibt jedoch keine offiziellen Versionen von MyCroft.ai für Windows oder Android, sondern nur inoffizielle Veröffentlichungen aus der Community. Daher musste zur Einrichtung von MyCroft.ai zunächst ein MyCroftCore auf einem Raspberry Pi installiert werden. Dafür war wiederum externe Hardware (Lautsprecher und Mikrofon) notwendig. Wenn der MyCroftCore aktiviert wurde, ist es möglich, über eine von der Community erstellte App MyCroft zu nutzen. Diese App ist jedoch nicht direkt auf Deutsch verfügbar. Die Umwandlung von Text zu Sprache (TTS) klang in den getesteten Beispielen sehr unnatürlich. In MyCroft ist es möglich, eigene Skills zu entwickeln. Diese Entwicklung erfolgt direkt über die Konsole auf dem initialisierten MyCroftCore und ist dadurch etwas unübersichtlich.

Amazons *Alexa* ist ein Sprachassistent, der die Spracherkennung und Sprachsynthese nicht lokal, sonderen in der Cloud durchführt. Die erzeugte Stimme klingt dadurch natürlicher und menschenähnlich. Die Skill-Entwicklung ist sehr übersichtlich aufgebaut und erfolgt online in einem Web-Interface. Alexa kann unkompliziert in allen verfügbaren Sprachen entweder auf einem Smartphone oder auch direkt im Browser verwendet werden. Eine Initialisierung eines lokalen Servers ist, anders als bei MyCroft, nicht notwendig. Das hat den Vorteil, dass einerseits keine zusätzliche Hardware (Raspberry Pi) benötigt wird und der Laptop für RASA verwendet werden kann. Nachteil von Alexa ist, dass die Software nicht Open-Source ist, ständig eine aktive Internetverbindung benötigt wird und Kritiker Bedenken äußern, dass persönliche Daten an Amazon weitergeleitet werden.

	MyCroft.ai		**Alexa**
⊕	Open-Source	⊖	Proprietär
⊕	ohne Internetverbindung nutzbar	⊖	Internetverbindung notwendig
⊕	Datensicherheit	⊖	Bedenken bei persönlichen Daten
⊖	zusätzliche Hardware für MyCroftCore notwendig	⊕	Funktioniert über Cloud
⊖	unnatürliche Sprachausgabe	⊕	menschenähnliche Sprachausgabe
⊖	unübersichtliche Skillentwicklung	⊕	einfache Skillentwicklung
⊖	eingeschränkte Verfügbarkeit auf Android	⊕	Support auf allen von Alexa unterstützten Geräten
⊖	eingeschränkte Verfügbarkeit auf deutsch	⊕	in vielen Sprachen verfügabr

Tabelle 5.1 – Vor- und Nachteile beider Agentenarchitekturen

Aufgrund der genannten Vorteile wurde bei der Wahl der Architektur, trotz der Bedenken, die den Schutz personenbezogener Daten betreffen, Amazons Alexa gewählt. Ein weiterer Beweggrund war, dass bereits Erfahrung bei der Skill-Entwicklung für Alexa vorlag, die für den Prototyp genutzt wurde.

5.2 Entwicklung des Prototyps

Zunächst mussten die beiden Dienste Alexa und RASA verbunden werden. Dazu wurde eine von RASA bereit gestellte Anleitung aus dem Internet als Hilfe verwendet[1]. Abbildung 5.1 zeigt diese Verbindung durch Visualisierung der Schritte, die zwischen Ein- und Ausgabe geschehen.
Die Worte eines Nutzers werden mit der Hilfe von Amazon Web Services (AWS) innerhalb eines entwickelten Skills dem `DummyIntent` zugeordnet. Dieser ist so gestaltet, dass er alle Eingaben an Alexa auffängt und den kompletten, vom Nutzer gesprochenen Text in einem Slot sichert. Die Weitergabe der von AWS erkannten Information erfolgt im JSON-Format. Als Endpunkt dient hier der auf dem Computer gehostete RASA-Core.
Dieser ist über `localhost:5005` erreichbar. Damit AWS einfach auf den lokalen Rechner zugreifen kann, wird *ngrok* verwendet, womit der gehostete Server ohne Verwendung von Portweiterleitung, virtuellem privatem Netzwerk (VPN) oder dynamischem Domain-Namen-System (DNS) erreichbar gemacht wird.
Aus der übergebenen JSON-Datei wird der Slot ausgelesen, in welchem die komplette Nachricht des Benutzers abgespeichert war. Im RASA Core wird nun eine passende Aktion gewählt und der Text der Antwort in JSON-Form gebracht, damit dieser wieder an AWS geschickt werden kann. Dort wird die JSON-Datei interpretiert und daraus eine gesprochene Ausgabe erzeugt.

5.3 Funktionen des Prototyps

Die Grundfunktion des Prototyps ist es, anhand unterschiedlicher Nutzereinschränkungen auf eine passende Interaktionsvariante zu schließen, wie es im Kapitel 4 beschrieben wurde.

[1]Für weitere Details siehe: `https://rasa.com/blog/connect-your-rasa-ai-assistant-to-amazon-alexa/`

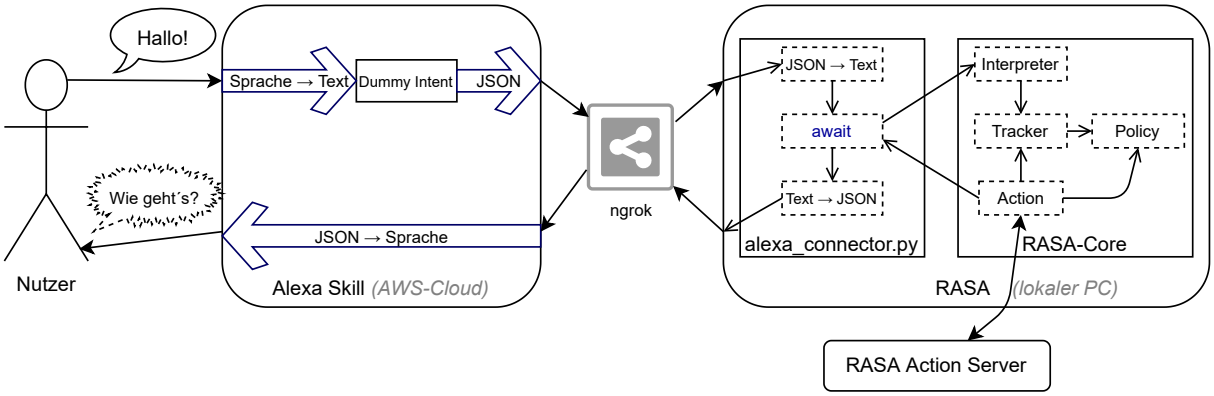


Abbildung 5.1 – Prozess der Dialogverarbeitung

Die Nutzereingabe wird hierfür durch den RASA-Core zu einem passenden Intent zugeordnet. Der Prototyp ist in der Lage, die folgenden Funktionen auszuführen:

1. **Benutzeränderung**: Damit der Prototyp von mehr als einer Person genutzt werden kann, ist es möglich, durch Äußerungen wie „*Ich bin ein neuer Nutzer.*“ den aktuellen Benutzer des Prototyps zu ändern. Alle folgenden Aktionen werden dann auf Grundlage der zu dem neuen Nutzer gespeicherten Informationen ausgeführt. Beim Start des Prototyps wird diese Aktion automatisch ausgeführt und der Prototyp erwartet die Eingabe des Namen des Nutzers.

2. **Einschränkung setzen**: Für jeden Nutzer kann dem Prototyp individuell angegeben werden, welche Einschränkungen er hat. Dabei muss die Einschränkung aus dem Feature-Modell FM_{USER} des vorgestellten Konzepts stammen. Eine Beeinträchtigung lässt sich beispielsweise mit den Worten „*Ich bin blind.*“ oder „*Deaktiviere Gehörlosigkeit.*“ aktivieren beziehungsweise deaktivieren.
   Im Anschluss an die Änderung der Nutzereinschränkungen werden zunächst Rückschlüsse über weitere Einschränkungen des Nutzers durch Konfiguration von FM_{USER} gezogen. Anhand dieser Informationen wird die Adaption, wie in Abbildung 4.6 dargestellt, durchgeführt und somit versucht, eine passende Interaktionsvariante zu finden. Der Prototyp speichert zuletzt diese Variante für den aktiven Nutzer ab.

3. **Testaktion:** Um die Anpassungen der Adaption an einem Beispiel zu zeigen wurde ein Testintent erstellt, dessen Formulierungen an die Interaktionsvariante angepasst sind. So ist hier eine schrittweise Dialogführung oder eine Reduzierung der Satzlängen bei Bedarf möglich. Diese Testaktion ist durch „*Unterhalte mich.*“ aufrufbar und stellt einen möglichen Dialog eines SAR dar.

4. **Adaption durchführen**: Zu Testzwecken kann der Prototyp auch ohne Änderung der Einschränkungen durch „*Starte die Adaption.*“ eine Adaption durchführen.

5. **Datei laden und schreiben**: Die Datei, in der Benutzerdaten gesichert werden, wird innerhalb der in Punkt 2 beschriebenen Aktion gespeichert. Sollte es trotzdem gewünscht sein, diese während eines anderen Zeitpunkts zu laden oder zu speichern, kann dies durch „*Laden.*“ oder „*Speichern.*“ erfolgen.

6. **Übersicht erstellen**: Um zu überprüfen, ob die Einschränkungen des Benutzers korrekt aufgenommen wurden, kann über „*Nenne die gespeicherten Einschränkungen.*“ eine Übersicht ausgegeben werden, die alle Beeinträchtigungen eines Nutzers ausgibt. Gleiches gilt für die, durch die Adaption erarbeitete Interaktionsvariante, welche durch „*Interaktion zeigen.*“ aufgerufen werden kann.

Nach der Vorstellung der Funktionen, die durch den Prototyp abgedeckt sind, wird im folgenden Abschnitt darauf eingegangen, wie diese umgesetzt wurden. Dazu wird zunächst beleuchtet, welche Klassen und Strukturen zur Speicherung eines Feature-Modells und des Regelsatzes notwendig sind.

5.3.1 Verwendete Klassen und Strukturen

Grundlegend für den Prüf- und Konfigurationsprozess ist die aktuelle `Auswahl` der Features und Attribute. Das wurde mit einem `Dictionary` von Python gelöst, welches bei den meisten Funktionen als Parameter übergeben wird. Dabei bedeutet die Zuordnung einer `1`, dass das Feature gewählt wurde und der `0`, dass das Feature nicht gewählt wurde. Für Attribute wird der jeweilige Attributwert gespeichert. Im Beispiel 5.1 ist die Auswahl von `Gehörlosigkeit` und explizite Nicht-Auswahl von `Blindheit` gefordert. Dem Attribut `Lautstärke` wurde der Wert `50` zugeordnet.

```
1    {
2        "Gehörlosigkeit": 1,
3        "Blindheit": 0,
4        "Lautstärke": 50
5    ]
```

Listing 5.1 – Dictionary zur Speicherung der Feature-Auswahl

In Abbildung 5.2 sind die wichtigsten Klassen und deren Beziehungen, die zur Darstellung einer Regel notwendig sind, abgebildet. Das unterste Strukturelement des Feature-Modells wird durch die Klasse `Feature` realisiert. Neben normalen Features gibt es ein `NegiertesFeature`, welches sich gegenteilig zum normalen `Feature` verhält. Ein `UndFeature` besteht aus einer Liste weiterer Features, die alle gewählt werden müssen. Bei `OderFeatures` muss aus der Liste der Features mindestens `minimum` und maximal `maximum` Features gewählt werden. Letztere sind für die Darstellung der Oder-Gruppen eines Feature-Modells notwendig. Letztlich gibt es mit dem `AttributFeature` die Möglichkeit eine (Un-)Gleichung ($=, \leq, <, \geq, >$) zwischen Attributen darzustellen. Durch die Klasse `AttributKette` ist es daneben möglich, Attribute durch mathematische Operationen miteinander zu verbinden. Die Klassen `NegiertesFeature`, `UndFeature`, `OderFeature` und `AttributFeature` erben von der Klasse `Feature`.
Auf nächsthöherer Ebene befindet sich die Klasse `Regel`. Diese besitzt eine `linkeSeite` und eine `rechteSeite` des Typs `Feature`. Aufgrund der Vererbung kann demnach aus den fünf Klassen ausgewählt werden.

In Tabelle 3.1 wurde gezeigt, wie sich welche Abhängigkeit (Kante) innerhalb eines Feature-Modells als Formel schreiben lässt. Wie sich diese Formeln nun durch die soeben vorgestellten Klassen darstellen lassen, zeigt Tabelle 5.2. Grundlage jeder Formel ist die Kasse `Regel`. Diese benötigt zwei Parameter des Typs `Feature`, wodurch die linke und rechte Seite einer Implikation beschrieben wird. Die Formel $\top \rightarrow A$ sagt aus, dass das Feature A immer gewählt werden muss. Für diesen

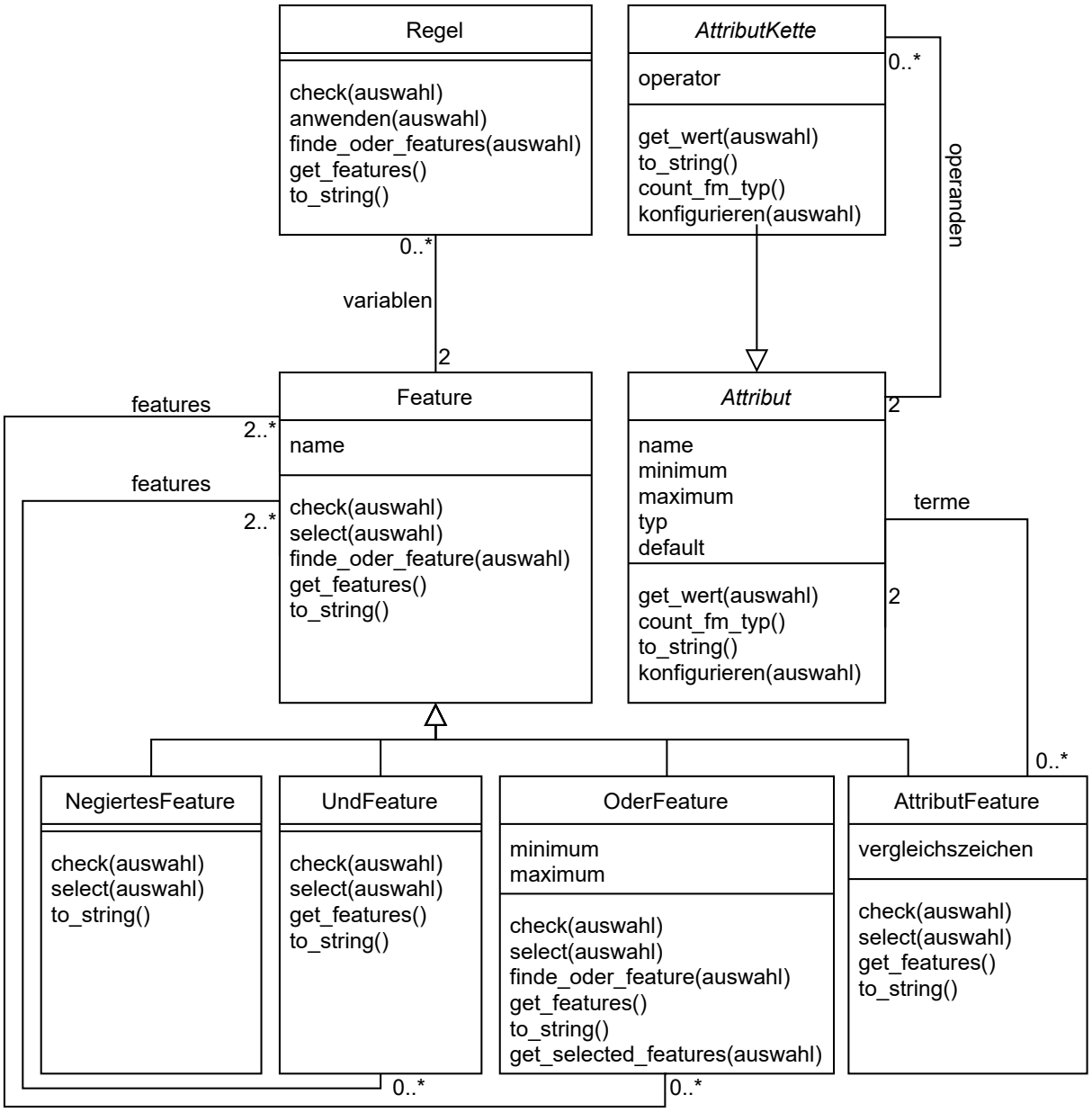


Abbildung 5.2 – Klassendiagramm zur Regeldefinition

Fall kann das `DummyFeature`[2] verwendet werden, welches immer *True* ergibt.
Während eine normale Implikation durch `Regel` einfach dargestellt werden kann, müssen für eine Äquivalenz der Form $A \leftrightarrow B$ zwei Regeln mit vertauschten Argumenten aufgestellt werden.
Negierte Teile einer Formel können mit der Hilfe von `NegiertesFeature` abgebildet werden. Im Prototyp wurde dabei jedoch nur die Negation eines einzelnen (normalen) Features umgesetzt. Die Formel $A \rightarrow \neg(B \wedge C \wedge D)$ müsste beispielsweise durch das De-morgansche Gesetz in die Formel $A \rightarrow (\neg B) \vee (\neg C) \vee (\neg D)$, welche die gleiche Bedeutung hat, übersetzt werden.
Eine Oder-Gruppe eines Feature-Modells wird durch `OderFeature` mit Angabe einer Liste aller

[2] Zur besseren Übersicht wurde diese Klasse nicht im Klassendiagramm dargestellt.

möglichen Features, der Minimalzahl und Maximalzahl ermöglicht. Analog dazu wird eine Konjunktion ($\wedge$) innerhalb einer Formel durch `UndFeatures` und Angabe aller Features im Prototyp gespeichert. Konjunktionen treten in den verwendeten Feature-Modellen nicht auf; innerhalb des Regelsatzes sind diese jedoch möglich.
Da sich im Prototyp nur Implikationen darstellen lassen, muss für eine (Un-)Gleichung, die durch Attribute erfüllt werden muss erneut das `DummyFeature` verwendet werden. Dem `AttributFeature` werden anschließend zwei Parameter vom Typ `Attribut` übergeben. Der zweite Parameter ist hierbei eine `AttributKette`, mit der die Summe $F_2 + U$ dargestellt wird. `AttributKette` nimmt als Parameter zwei `Attribut`-Elemente und eine Funktion, die angewendet werden soll.

Formel	Verwendung	Code
$\top \rightarrow A$	a_0	`Regel(DummyFeature('true'), Feature('A'))`
$A \rightarrow B$	a_2, a_3, a_4	`Regel(Feature('A'), Feature('B'))`
$A \leftrightarrow B$	a_1	`Regel(Feature('A'), Feature('B'))` `Regel(Feature('B'), Feature('A'))`
$A \rightarrow \neg B$	a_5	`Regel(Feature('A'), NegiertesFeature('B'))`
$A \rightarrow \text{Oder}([B, C], a, b)$	a_3	`Regel(Feature('A'), OderFeatures(` `[Feature('B'), Feature('C')], a, b)`
$A \rightarrow B \wedge C$	Regelsatz	`Regel(Feature('A'), UndFeatures(` `[Feature('B'), Feature('C')])`

$F_1 \leq F_2 + U$	Regelsatz	`f1=Attribut('F1', 1, 10, 1)`
		`f2=Attribut('F2', 1, 10, 1)`
		`u=Attribut('U', 1, 10, 2)`
		`Regel(DummyFeature('true'), AttributFeature(`
		`f1, AttributKette(f2, u, addition), '< ='))`

Tabelle 5.2 – Darstellung logischer Abhängigkeiten im Prototyp

Ein Feature-Modell wird durch Angabe aller in ihm enthaltenen Kanten initialisiert. Die Kanten sind in diesem Fall Instanzen der Klasse `Regel`. Zusätzlich zu den Kanten wird für ein Feature-Modell eine Menge aller enthaltenen Features gespeichert. Die Features werden dabei entweder bei der Initialisierung angegeben oder durch die Funktion `kanten_to_features` ermittelt, indem die Features jeder Regel/Kante zu der Menge hinzugefügt werden.

5.3.2 Prüfung eines Regelsatzes

Die Funktion `validieren(regelsatz, auswahl)` prüft, ob eine Auswahl `auswahl` einen Regelsatz `regelsatz` erfüllt. Dazu wird zunächst geprüft, ob die Auswahl gültig ist. Sollte sie einen Wert auf `-1` zuweisen oder `None` sein, ist sie ungültig. Danach wird jede Regel des Regelsatzes einzeln geprüft. Jede Regel muss erfüllt sein, damit *True* zurückgegeben werden kann; anderenfalls wird *False* ausgegeben.
Eine Regel gilt als erfüllt, wenn entweder die linke Seite *False* ist oder beide Seiten *True* sind. Die Spalte „Rückgabe von `check(auswahl)`“ der Tabelle 5.3 zeigt, wie die einzelnen Features auf Korrektheit überprüft werden. Ein `Feature` muss zwingend auf `1` gesetzt worden sein. Keine Zuordnung ist nicht zulässig und liefert *False* zurück. Analog verhält sich ein `NegiertesFeature`. Hier

muss `0` gespeichert sein, damit *True* zurückgegeben wird. `UndFeature` und `OderFeature` besitzen jeweils eine Liste, welche `Feature` enthält. Bei einem `UndFeature` müssen alle Features aus der Liste beim Aufruf von `check(auswahl)` *True* liefern. Bei einem `OderFeature` hingegen muss die Anzahl der Features, die bei `check(auswahl)` *True* ergeben größer oder gleich `minimum` und kleiner oder gleich `maximum` sein. Die bei einem `AttributFeature` hinterlegte (Un-)Gleichung muss erfüllt sein, damit dort *True* zurückgegeben wird.

Klasse	Rückgabe von `check(auswahl)`	Änderung bei `select(auswahl)`
`Feature`	$\begin{cases} True & \text{, Feature in } \texttt{auswahl} \\ & \text{mit } \texttt{1} \text{ gespeichert} \\ False & \text{, sonst} \end{cases}$	$\begin{cases} 1 & \text{, Feature nicht in } \texttt{auswahl} \\ -1 & \text{, Feature in } \texttt{auswahl} \\ & \text{mit } \texttt{0} \text{ gespeichert} \end{cases}$
`Negiertes Feature`	$\begin{cases} True & \text{, Feature in } \texttt{auswahl} \\ & \text{mit } \texttt{0} \text{ gespeichert} \\ False & \text{, sonst} \end{cases}$	$\begin{cases} 0 & \text{, Feature nicht in } \texttt{auswahl} \\ -1 & \text{, Feature in } \texttt{auswahl} \\ & \text{mit } \texttt{1} \text{ gespeichert} \end{cases}$
`Und Feature`	$\begin{cases} True & \text{, für alle Features gilt:} \\ & \texttt{check()} == True \\ False & \text{, sonst} \end{cases}$	`select()` für jedes Feature
`Oder Feature`	$\begin{cases} True & \text{, } \texttt{min} \leq \text{Features mit} \\ & \texttt{check()} == True \leq \texttt{max} \\ False & \text{, sonst} \end{cases}$	`-/-`
`Attribut Feature`	$\begin{cases} True & \text{, wenn (Un-)Gleichung} \\ & \text{erfüllt} \\ False & \text{, sonst} \end{cases}$	Liste der möglichen Werte für das unbekannte Attribut, sodass die (Un-)Gleichung erfüllt ist

Tabelle 5.3 – Rückgabe der Funktionen des Prototyps

5.3.3 Konfiguration anhand eines Regelsatzes

Durch die Funktion `konfigurieren(regelsatz, auswahl)`, die als Pseudocode in Listing 5.2 dargestellt ist, werden der Auswahl `auswahl` solange Features hinzugefügt, bis alle Regeln des Regelsatzes `regelsatz` erfüllt sind. Dazu werden alle Regeln des Regelsatzes durch die Funktion `select(auswahl)` angewendet. Die Spalte „Änderung bei `select(auswahl)`“ von Tabelle 5.3 zeigt, wie welches Feature die Auswahl editiert. Bei einem normalen `Feature` wird eine `1` in `auswahl` geschrieben. Sollte dort jedoch bereits eine `0` gespeichert sein, würde das heißen, dass das Feature einerseits benötigt wird und andererseits nicht verwendet werden darf. Dieser Widerspruch wird mit dem Wert `-1` gekennzeichnet. Ein `NegiertesFeature` verhält sich analog dazu. Der Unterschied ist, dass statt einer `1` eine `0` gespeichert wird und der Widerspruch dann auftritt, wenn bereits eine `1` gespeichert wurde. Für ein `UndFeature` wird `select(auswahl)` für jedes gespeicherte Feature aus der Liste angewendet. Ein `OderFeature` gibt bei Aufruf von `select(auswahl)` direkt `auswahl` zurück, ohne etwas zu ändern. Das hat den Grund, dass die

`OderFeatures` eines Feature-Modells nach dem übrigen Konfigurationsprozess ausgewählt werden. Bei einem `AttributFeature` werden alle möglichen Zahlen, die der unbekannte Attributwert annehmen kann in einer Liste gespeichert. Die Änderungen, die sich dadurch ergeben, werden in `auswahl` geschrieben.
Nachdem alle Regeln angewendet wurden, wird geprüft, ob dadurch Änderungen in `auswahl` auftreten. Sollte dies der Fall sein, werden erneut alle Regeln angewendet. Dieses Vorgehen wird solange wiederholt, bis es zu keinen Neuerungen kommt (vgl. Zeile 1 - 3).
Erst wenn alle übrigen Features konfiguriert wurden, werden alle Kombinationen betrachtet, wie `OderFeatures` konfiguriert werden können, sodass die jeweiligen Gruppenkardinalitäten stimmen. Diese werden in einer Liste gespeichert, aus der anschließend eine zufällige Kombination gewählt wird. Falls es durch die gewählte Kombination zu keinem Widerspruch kommt, wird diese angenommen. Anderenfalls wird solange eine weitere zufällige Kombination gezogen, bis entweder eine passende Konfiguration gewählt wurde oder die Liste leer ist und somit keine passende Konfiguration gefunden werden konnte. (Zeilen 5 - 11).
Zuletzt wird in den Zeilen 13 - 14 für alle Attribute ein konkreter Wert gesetzt. Während des Konfigurationsprozesses wurden hier teilweise Listen mit möglichen Werten gespeichert. Eine finale Variante muss jedoch einen konkreten Wert für jedes Attribut besitzen. Daher wird aus den Listen ein zufälliger Wert ausgesucht.

```
1     do:
2         alle Regeln anwenden und in 'auswahl' speichern
3     while 'auswahl' hat sich seit dem letzten Schleifendurchgang geändert
4
5     speichere alle Kombinationen, wie OderFeatures konfiguriert werden können in einer Liste
6     do:
7         Liste der Kombinationen ist leer?
8             ja: ungültig; return None
9             nein: wähle eine beliebige Konfiguration aus und lösche sie aus der Liste
10    while ausgewählte Konfiguration verletzt Regel
11    nutze die ausgesuchte Konfiguration
12
13    wenn für ein Feature eine Liste gespeichert wurde
14        wähle einen zufälligen Wert aus
```

Listing 5.2 – Pseudocode des Konfigurationsprozesses

5.3.4 Speicherung der Daten

Bei einem Neustart von RASA würden die in der vergangenen Sitzung gespeicherten Slots verloren gehen. Daher werden alle relevanten Informationen in einer JSON-Datei (siehe Listing 5.3) auf dem Rechner, auf dem der RASA-Action-Server läuft, gespeichert. Für jeden eingepflegten Nutzer werden dort die gespeicherten Einschränkungen unter dem Schlüssel `einschränkungen` und die Nutzerpräferenzen unter dem Schlüssel `variablen` gespeichert. Beim Setzen einer Variable wird automatisch das Feature-Modell FM_{USER} konfiguriert und die dadurch gewonnenen Erkenntnisse unter `alle_einschränkungen` gespeichert. Nach einer erfolgreichen Adaption werden die Rückschlüsse, die dort gewonnen wurden unter `interaktion` gesichert. In dem gezeigten Beispiel hat der Nutzer *Hans* die Einschränkung *Blindheit*, präferiert eine *schrittweise Verarbeitung* und hat angegeben, dass er *Braille* lesen kann. Das System schließt daraus auf weitere Features und erkennt beispielsweise, dass der Nutzer *keine Farbenblindheit* und *keine Sehschwäche* haben kann.

Durch die Adaption wird eine *akustische Ausgabe* über die Sprachsteuerung und die Aktivierung einer *Braillezeile* gefolgert. Die angegebene Präferenz zur *schrittweisen Verarbeitung* wird ebenfalls beachtet.

```
{
"hans": {
    "name": "Hans",
    "einschränkungen": ["blindheit"],
    "variablen": {
        "verarbeitung komplexität schrittweise verarbeitung": 1,
        "uservalue_braille": 3
    },
    "alle_einschränkungen": {
        "blindheit": 1,
        "sehen": 1,
        "sehschärfe": 0,
        "farbfehlsichtigkeit": 0,
        "sensorisch": 1,
        "einschränkungen": 1
    },
    "interaktion": {
        "ausgabe akustik": 1,
        "ausgabe braille": 1,
        "verarbeitung komplexität schrittweise verarbeitung": 1,
        ... # weitere Schlussfolgerungen über Interaktion
    }
},
... # weitere registrierte Nutzer
}
```

Listing 5.3 – JSON-Datei mit gespeicherten Nutzerdaten

5.3.5 Adaption nach Änderung einer Einschränkung

In diesem Teilabschnitt soll es darum gehen, welche Schritte im Hintergrund des Prototyps geschehen, wenn ein Nutzer, wie in Funktion 2 beschrieben, eine seiner Einschränkungen ändert.
Die Funktion überprüft erst, ob die neue Einschränkung valide ist und fügt diese bei Erfolg hinzu, wie es in Listing 5.4 dargestellt ist. Zunächst wird hierfür geprüft, ob die Einschränkung, die geändert werden soll, bekannt und in FM_{USER} enthalten ist. Es folgt eine Prüfung, ob die Einschränkung, die hinzugefügt oder entfernt werden soll, bereits aktiv ist oder deaktiviert wurde. Dazu wird die Liste `einschränkungen` aus den in Listing 5.3 gezeigten Nutzerdaten verwendet.
Je nachdem, ob eine Aktivierung oder Deaktivierung einer Einschränkung gewünscht ist, wird die Beeinträchtigung zu der Liste der bereits bekannten Einschränkungen des Benutzers hinzugefügt oder gelöscht.
Wie in Abschnitt 5.3.3 beschrieben, wird die Auswahl der für den aktiven Nutzer relevanten Einschränkungen anhand der Regeln, die die Kanten von FM_{USER} abbilden, konfiguriert. Diese Konfiguration wird im Anschluss, wie in Abschnitt 5.3.2 erläutert auf Korrektheit überprüft. Sofern die Konfiguration valide ist, wird mit der Adaption des Systems fortgefahren. Bei einer invaliden Konfiguration hingegen werden die Änderungen rückgängig gemacht und der Vorgang mit einem Widerspruch beendet.

```
# Legende zur Erläuterung der Attribute
#
# einschränkung: Einschränkung, die geändert werden soll
# aktivierung: True, wenn Einschränkung deaktiviert werden soll; False sonst
# user.einschränkungen: Gespeicherte Einschränkungen des Nutzers
#

if 'einschränkung' nicht aus 'fm_user':
    Abbruch

if 'aktivierung' and 'einschränkung' bereits aktiviert:
    Abbruch

if not 'aktivierung' and 'einschränkung' bereits deaktiviert:
    Abbruch

if aktivierung:
    'einschränkung' zu 'user.einschränkungen' hinzufügen
else:
    'einschränkung' aus 'user.einschränkungen' entfernen

'user.einschränkungen' für 'fm_user' konfigurieren -> 'konfiguration'

if 'konfiguration' valide:
    adaption mit 'konfiguration' durchführen
else:
    Widerspruch
    Revidieren der Änderungen
```

Listing 5.4 – Pseudocode zur Einschränkungsänderung

Der nächste Schritt ist die Adaption des Systems an die aktualisierten Nutzeranforderungen, welche in Listing 5.5 als Pseudocode notiert ist. Dazu werden Nutzereinschränkungen (Blindheit, Gehörlosigkeit, ...) und Nutzerpräferenzen (Wunsch der akustischen Ausgabe, Mindestlautstärke, ...) kombiniert.
Innerhalb einer Endlosschleife wird zunächst eine Initialkonfiguration für FM_{SYSTEM} anhand der Nutzeranforderungen durch Verwendung des Regelsatzes $C_{ANFORDERUNGEN}$ und der in Abschnitt 5.3.3 vorgestellten Funktion erstellt. Aus dieser Erstkonfiguration wird durch erneute Anwendung der Konfiguration aus Abschnitt 5.3.3 zusammen mit dem Regelsatz C_{SYSTEM} eine Interaktionsvariante erarbeitet.
Sollte diese Variante valide sein, wird sie übernommen und das System entsprechend angepasst. Anderenfalls wird, wenn möglich, eine niedrig priorisierte Regel aus $C_{ANFORDERUNGEN}$ gelöscht und die Schleife erneut durchlaufen. Sollte keine Regel mehr entfernt werden können, bricht die Adaption erfolglos ab.

```
    # Legende zur Erläuterung der Attribute
    #
    # user_einschränkung: Einschränkung des Nutzer
    # user_präferenzen: Präferenzen des Nutzer an die Interaktion
    # c_anforderungen: Regelsatz
    # c_system: Kanten von fm_system
    #

    'user_präferenzen' zu 'user_einschränkungen' hinzufügen

    while True:
        'user_einschränkungen' für 'c_anforderungen' konfigurieren -> 'konfiguration'
        'konfiguration' für 'c_system' konfigurieren -> 'konfiguration_final'

        if 'konfiguration_final' valide:
            Verwendung der neuen Variante 'konfiguration_final'
            return
        else:
            if keine Regel mehr löschbar:
                return Widerspruch
            Regel mit niedrigstem Gewicht aus 'c_anforderungen' löschen
```

Listing 5.5 – Pseudocode zur Adaption

5.4 Inbetriebnahme des Prototyps

In diesem Abschnitt soll es darum gehen, wie der Prototyp in Betrieb genommen und verwendet werden kann.
Zunächst muss der RASA-Core auf dem Rechner gestartet werden. Dies geschieht über Eingabe der Befehle innerhalb des Verzeichnisses `prototyp/rasa_prototyp`. Bei der ersten Verwendung müssen hierfür zunächst Python 3.9 installiert und die erforderlichen Pakete mittels des Befehls `pip install -r requirements.txt` installiert werden. Falls das Modell von RASA noch nicht trainiert wurde, muss dies durch `rasa train` geschehen. Ab sofort kann der RASA-Core mit dem Befehl `rasa run` gestartet werden. Möchte man den Prototyp direkt in der Konsole ohne Anbindung an Alexa nutzen, bietet `rasa shell` dafür die Möglichkeit.

Damit eigene Aktionen in der von Python-Funktionen durch RASA ausgeführt werden können, muss der RASA-Action-Server gestartet werden. Dafür muss in einer weiteren Konsole im Verzeichnis `prototyp/rasa_prototyp/actions` der Server durch Eingabe von `rasa run actions` aufgesetzt werden.

Durch Ausführung dieser Punkte ist es möglich, den Prototyp als Chatbot innerhalb der Konsole zu nutzen. Wenn jedoch die Verwendung als Sprachassistent über Alexa gewünscht ist, muss zunächst *ngrok* aktiviert werden, um die unkomplizierte Weiterleitung zum eigenen `localhost` zu ermöglichen. Dafür muss im Verzeichnis `prototyp/rasa_ngrok` der Befehl `ngrok.exe http 5005`[3] ausgeführt werden. In der Konsole erscheint unter dem Punkt `Forwarding` eine Uniform

[3] Unter Mac und Linux lautet der Befehl `ngrok http 5005`.

Resource Locator (URL), die im Alexa-Skill hinterlegt werden muss. Der URL muss zusätzlich der Adresspfad `/webhooks/alexa_assistant/webhook` angehängt werden.

Auf die Konfiguration des Alexa-Skills wird in dieser Anleitung nicht weiter eingegangen . Es wird nur erwähnt, dass unter `prototyp/alexa_schema.json` das verwendete Schema abgelegt ist. In der Amazon-Developer-Konsole muss für den Alexa Skill der passende Endpoint[4], wie im vorherigen Textabschnitt erläutert, eingefügt werden.

Nach Start des Prototyps lädt dieser im Hintergrund bereits die Datei, in der die Nutzerdaten der letzten Sitzung abgespeichert wurden. Der Prototyp erwartet vom Benutzer, dass dieser seinen Namen eingibt. Falls man im Laufe einer Sitzung den Namen ändern möchte, ist das durch „*Benutzer ändern.*“ und anschließender Eingabe des Namens möglich. Eine Einschränkung kann für einen Benutzer beispielsweise durch „*Aktiviere Blindheit*“ aktiviert und durch „*Deaktiviere Blindheit*“ deaktiviert werden. Dabei wird automatisch eine Adaption durchgeführt. Diese kann jedoch bei Bedarf auch nach einer Aufforderung durch den Nutzer mittels „*Adaption*“ erfolgen. Um unterschiedliche Formulierungen zu testen, die der Prototyp aufgrund der gegeben Einschränkungen folgert, kann mit „*Unterhalte mich.*“ ein Dialog gestartet werden. Sollte der Nutzer keine kognitiven Einschränkungen haben, erwartet der Prototyp anschließend Anweisungen wie „*Ich möchte einen Witz zum Thema Spannung.*“ Für Menschen mit kognitiven Einschränkungen wechselt der Prototyp auf eine Dialogstruktur mit Ja/Nein-Fragen, um die Komplexität zu senken.

5.5 Zusammenfassung

In diesem Kapitel wurde die prototypische Umsetzung des erarbeiteten Konzepts vorgestellt. Dazu wurde zu Beginn auf die Wahl der verwendeten Komponenten eingegangen. Es wurde entschieden, eine Kombination aus RASA und Alexa zu verwenden. Nachdem die Verbindung der beiden Architekturen erläutert wurde, wurden die Funktionen vorgestellt. Dabei wurde im Detail zum einen auf die Speicherung eines Feature-Modells und dessen Varianten und zum anderen auf die Konfiguration eines Feature-Modells anhand eines Regelsatzes und einer Initialkonfiguration eingegangen. Letztlich wurde erklärt, mit welchen Befehlen der Prototyp in Betrieb genommen werden kann.
Mit Hilfe des soeben Prototyps, soll das in Kapitel 4 vorgestellte Konzept auf Korrektheit überprüft werden.

[4] Dieser ist von der Form `https://0150-2003-ao-c738-8rli-40ew-j8i1-3bae-19f2.eu.ngrok.io/webhooks/alexa_assistant/webhook`.

6 Auswertung des Prototyps durch Evaluation

Im vorherigen Kapitel wurde auf Grundlage des erarbeiteten Konzepts ein Prototyp aufgesetzt, mit dessen Hilfe nun eine Evaluation durchgeführt wird, um die Korrektheit des Konzepts zu beweisen. Ziel soll es sein, durch geeignete Testfälle zu zeigen, dass die in 3.1 definierten Anforderungen durch das aufgestellte Konzept erfüllt werden.

6.1 Planung der Evaluation

Um zu überprüfen, ob sich das Konzept wie erwartet verhält, indem die Interaktion des SAR an die spezifischen Einschränkungen des Benutzers angepasst wird, werden unterschiedliche Testfälle betrachtet. Dabei wird pro Testfall eine andere Einschränkung aus dem User-Modell FM_{USER} betrachtet. Zusammen mit der leeren Konfiguration ergeben sich somit 14 verschiedene Varianten. Hinzu kommt der Testfall $T'_{1,3}$, der die Nutzung des Systems durch einen blinden Nutzer, der keine Blindenschrift lesen kann, beschreibt. Für jede einzelne Variante muss geprüft werden, wie sich das System durch Wahl einer bestimmten Interaktionsmethode anpassen würde. Soweit dies möglich ist, wird hierfür der im vorherigen Abschnitt entwickelte Prototyp verwendet. Da es sich bei diesem lediglich um einen Sprachassistenten handelt, können nicht alle Interaktionsmöglichkeiten, die im Feature-Modell FM_{SYSTEM} vorgestellt wurden, getestet werden. Damit dennoch für diese Testfälle eine aussagekräftige Folgerung geschlossen werden kann, gibt der Prototyp für diese Fälle per Textausgabe über den Bildschirm aus, inwiefern sich die Ein- und Ausgabemöglichkeiten verändern würden. Benötigt wird dieses Vorgehen beispielsweise bei der Braillezeile oder bei der Gestenerkennung.

Neben den soeben vorgestellten 15 Testfällen, bei welchen keine oder nur eine gleichzeitige Einschränkung betrachtet wurde, gibt es noch weitere Fälle, die durch Kombination mehrerer Einschränkungen auftreten können. FM_{USER} lässt fasst 3000 verschiedene Kombinationen aus Einschränkungen zu. Bedingt durch diese Vielzahl, ist ersichtlich, dass nicht jede einzelne Kombination auf Korrektheit überprüft werden kann. Es werden daher nur einzelne Kombinationen betrachtet, die als relevant erachtet werden. In Tabelle 6.1 sind alle 26 Testfälle aufgelistet. Diese wurden nach der Anzahl der gewählten Einschränkungen sortiert.
$T_{0,1}$ ist der Testfall, bei dem der Nutzer keine Einschränkungen hat. Die Testfälle $T_{1,1}$ bis $T_{1,14}$ betrachten die Fälle, wenn ein Nutzer genau eine Einschränkung hat.
Im Folgenden wird darauf eingegangen, wie Testfälle ausgewählt wurden, die problematische Einschränkungskombinationen testen sollen. Bei Betrachtung des Adaptionsalgorithmus aus Abbildung 4.6 erkennt man, dass es zwei mögliche Fälle gibt, wann eine Adaption nicht alle Regeln von $C_{ANFORDERUNGEN}$ erfüllt oder ohne Erfolg komplett abbricht. Dies ist der Fall, wenn entweder $L_{INITIAL}$ oder L_{SYSTEM} einen Widerspruch enthält und ein Feature gleichzeitig aktivieren und deaktivieren möchte. Demnach müssen zur Auswahl relevanter Testfälle, die mögliche Probleme des Konzepts aufzeigen können, die Regeln aus $C_{ANFORDERUNGEN}$ genauer betrachtet werden, die eine

Negation enthalten. Die Negation bedeutet, dass ein bestimmtes Feature zwingend ausgeschlossen wird. Durch diesen Ausschluss könnte es zu einem Konflikt kommen, wenn eine andere Regel die gleichzeitige Aktivierung fordert. Aus diesem Grund werden die 9 Regeln c_{12}, c_{13}, c_{24}, c_{27}, c_{31}, c_{37}, c_{38}, c_{40} und c_{41}, welche Negationen enthalten, genauer betrachtet und auf ihrer Grundlage Testfälle generiert. Regel c_{41} hängt nicht von den Einschränkungen der Benutzer, sondern von einer Nutzervariable ab und wird hier aus diesem Grund nicht weiter betrachtet.
Die Regeln c_{12} und c_{13} fordern das Ausschalten des Touchdisplays bei Blindheit. Dieses wird jedoch bei den Regeln c_{20} und c_{22} gefordert. Dadurch ergeben sich die Testfälle $T_{2,1}$ und $T_{2,2}$, in denen *Blindheit* in Kombination mit *Schwerhörigkeit* beziehungsweise *Gehörlosigkeit* betrachtet wird. Auch Regel c_{40} verlangt die Deaktivierung der visuellen Ausgabe. Daher wird der Testfall $T_{3,1}$ erzeugt, der die Kombination der drei Einschränkungen *Blindheit*, *Probleme bei geschriebener Sprache* und *Gehörlosigkeit* beschreibt.
c_{24} und c_{38} fordern bei gehörlosen Nutzern beziehungsweise Menschen, die Probleme haben, gesprochene Sprache zu verstehen, die Deaktivierung der akustischen Ausgabe, welche jedoch bei den Regeln c_7, c_{10} und c_{30} explizit gefordert wird. Aus diesen Zusammenhängen wurden die Testfälle $T_{2,3}$ (*Gehörlosigkeit* und *Sehschwäche*) und $T_{2,4}$ (*Probleme bei gesprochener Sprache* und *Blindheit*) mit je zwei gleichzeitigen Einschränkungen und die Kombination $T_{4,1}$ mit der Verknüpfung der vier Einschränkungen *Gehörlosigkeit*, *Probleme bei gesprochener Sprache*, *Blindheit* und *schwerer körperlicher Einschränkungen* erzeugt.
Die beiden Regeln c_{27} und c_{31} erfordern das Abschalten der Multi-Touch-Funktionalität, welche jedoch von keiner anderen Regel betrachtet wird. Aus diesem Grund ergeben sich hier keine weiteren Testfälle.
Letztlich verbleibt Regel c_{37}, welche die Deaktivierung einer akustischen Eingabe für Menschen mit Sprachstörungen (Probleme selbst zu sprechen) anordnet. Diese wird hingegen bei den Regeln c_7, c_{10} und c_{29} erwartet. Dadurch ergeben sich die weiteren Testfälle $T_{2,5}$, $T_{2,6}$ und $T_{3,2}$, durch die zwei Kombinationen mit zwei Einschränkungen und eine Kombination mit drei gleichzeitigen Einschränkungen abgebildet werden.

Als letzte Testfälle werden zwei Kombinationen betrachtet, bei der möglichst viele Einschränkungen gleichzeitig auftreten. Aufgrund der *Ausgeschlossen Relationen* innerhalb von FM_{USER} ist es nicht möglich, alle Features gleichzeitig zu wählen. Daher wurden zwei Varianten zufällig gewählt. Dadurch ergeben sich die Testfälle $T_{9,1}$ und $T_{10,1}$.

6.2 Dokumentation während der Durchführung der Evaluationen

Um für jeden der 25 Testfälle die Erfüllung jeder Anforderung zu überprüfen, werden mithilfe des `SetzeEinschränkung`-Intents des Prototyps für jeden Testfall die geforderten Einschränkungen des Nutzers aktiviert und nicht geforderte deaktiviert.
Aus 25 Testfällen und 38 Anforderungen ergeben sich 988 einzelne Prüfpunkte, die zur besseren Übersicht in die zwei Tabellen 6.2 und 6.3 aufgeteilt wurden. Dabei bildet die erste Tabelle die Testfälle, die keine oder nur eine Einschränkung betrachten und die zweite die übrigen Testfälle ab.
Die Notation ist so gewählt, dass die Kennzeichnnung mit „$\circ$“ bedeutet, dass die Anforderung aufgrund der getesteten Einschränkung nicht relevant ist. Das Zeichen „$\surd$“ heißt, dass die Anforderung relevant und erfüllt ist. Mit „$\times$“ wurden Prüfpunkte gekennzeichnet, die nicht erfüllt

ID	Beschreibung
$T_{0,1}$	keine Einschränkungen
$T_{1,1}$	Sehschwäche
$T_{1,2}$	Farbfehlsichtigkeit
$T_{1,3}$	Blindheit
$T'_{1,3}$	Blindheit *ohne Kenntnis der Brailleschrift*
$T_{1,4}$	Schwerhörigkeit
$T_{1,5}$	Gehörlosigkeit
$T_{1,6}$	leichte körperliche Einschränkung
$T_{1,7}$	schwere körperliche Einschränkung
$T_{1,8}$	körperliche Sprachstörung
$T_{1,9}$	Gedächtnisstörung
$T_{1,10}$	kognitive Sprachstörung
$T_{1,11}$	Probleme bei gesprochener Sprache
$T_{1,12}$	Probleme bei geschriebener Sprache
$T_{1,13}$	reduzierte Intelligenz
$T_{2,1}$	Blindheit ◇ Schwerhörigkeit
$T_{2,2}$	Blindheit ◇ Gehörlosigkeit
$T_{2,3}$	Gehörlosigkeit ◇ Sehschwäche
$T_{2,4}$	Probleme bei gesprochener Sprache ◇ Blindheit
$T_{2,5}$	kognitive Sprachstörung ◇ Sehschwäche
$T_{2,6}$	körperliche Sprachstörung ◇ schwerere körperliche Einschränkungen
$T_{3,1}$	Blindheit ◇ Probleme bei geschriebener Sprache ◇ Gehörlosigkeit
$T_{3,2}$	kognitive Sprachstörung ◇ Blindheit ◇ schwerere körperliche Einschränkungen
$T_{4,1}$	Gehörlosigkeit ◇ Probleme bei gesprochener Sprache ◇ Blindheit ◇ schwerere körperliche Einschränkungen
$T_{9,1}$	Blindheit *ohne Kenntnis der Brailleschrift* ◇ Schwerhörigkeit ◇ schwere körperliche Einschränkung ◇ körperliche Sprachstörung ◇ Gedächtnisstörung ◇ kognitive Sprachstörung ◇ reduzierte Intelligenz ◇ Probleme bei gesprochener Sprache ◇ Probleme bei geschriebener Sprache
$T_{10,1}$	Sehschwäche ◇ Farbfehlsichtigkeit ◇ Gehörlosigkeit ◇ reduzierte Intelligenz ◇ Probleme bei gesprochener Sprache ◇ Probleme bei geschriebener Sprache ◇ leichte körperliche Einschränkung ◇ körperliche Sprachstörung ◇ kognitive Sprachstörung ◇ Gedächtnisstörung

Tabelle 6.1 – Herausgearbeitete Testfälle für die Evaluation

wurden.

Testfall	$T_{0,1}$	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$	$T'_{1,3}$	$T_{1,4}$	$T_{1,5}$	$T_{1,6}$	$T_{1,7}$	$T_{1,8}$	$T_{1,9}$	$T_{1,10}$	$T_{1,11}$	$T_{1,12}$	$T_{1,13}$
1.1.	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
1.2.	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
1.3.	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
1.4.	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
1.5.	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
2.1.1.	○	√	○	○	○	○	○	○	○	○	○	○	○	○	○
2.1.2.	○	√	○	○	○	○	○	○	○	○	○	○	○	○	○
2.1.3.	○	√	○	○	○	○	○	○	○	○	○	○	○	○	○
2.1.4.	○	√	○	○	○	○	○	○	○	○	○	○	○	○	○
2.2.1.	○	○	○	√	√	○	○	○	○	○	○	○	○	○	○
2.2.2.	○	○	○	√	√	○	○	○	○	○	○	○	○	○	○
2.2.3.	○	○	○	√	√	○	○	○	○	○	○	○	○	○	○
2.2.4.	○	○	○	√	√	○	○	○	○	○	○	○	○	○	○
2.3.1.	○	○	√	○	○	○	○	○	○	○	○	○	○	○	○
2.3.2.	○	○	√	○	○	○	○	○	○	○	○	○	○	○	○
3.1.1.	○	○	○	○	○	√	○	○	○	○	○	○	○	○	○
3.1.2.	○	○	○	○	○	√	○	○	○	○	○	○	○	○	○
3.1.3.	○	○	○	○	○	√	○	○	○	○	○	○	○	○	○
3.1.4.	○	○	○	○	○	√	○	○	○	○	○	○	○	○	○
3.1.5.	○	○	○	○	○	√	○	○	○	○	○	○	○	○	○
3.2.1.	○	○	○	○	○	○	√	○	○	○	○	○	○	○	○
3.2.2.	○	○	○	○	○	○	√	○	○	○	○	○	○	○	○
3.2.3.	○	○	○	○	○	○	√	○	○	○	○	○	○	○	○
4.1.1.	○	○	○	○	○	○	○	√	○	○	○	○	○	○	○
4.1.2.	○	○	○	○	○	○	○	√	○	○	○	○	○	○	○
4.1.3.	○	○	○	○	○	○	○	√	○	○	○	○	○	○	○
4.2.1.	○	○	○	○	○	○	○	○	√	○	○	○	○	○	○
4.2.2.	○	○	○	○	○	○	○	○	√	○	○	○	○	○	○
4.2.3.	○	○	○	○	○	○	○	○	√	○	○	○	○	○	○
5.1.1.	○	○	○	○	○	○	○	○	○	○	√	√	√	√	√
5.1.2.	○	○	○	○	○	○	○	○	○	○	√	√	√	√	√
5.1.3.	○	○	○	○	○	○	○	○	○	○	√	√	√	√	√
5.2.1.	○	○	○	○	○	○	○	○	○	○	√	○	○	○	○
5.2.2.	○	○	○	○	○	○	○	○	○	○	√	○	○	○	○
6.1.	○	○	○	○	○	○	○	○	○	√	○	√	○	○	○
6.2.	○	○	○	○	○	○	○	○	○	○	○	○	√	○	○
6.3.	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
6.4.	○	○	○	○	○	○	○	○	○	○	○	○	○	√	○

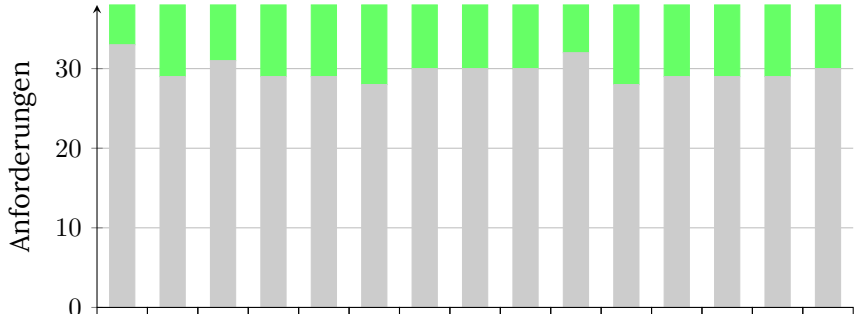
Tabelle 6.2 – Ergebnis der Testfälle $T_{0,1}$ bis $T_{1,13}$

Testfall	$T_{2,1}$	$T_{2,2}$	$T_{2,3}$	$T_{2,4}$	$T_{2,5}$	$T_{2,6}$	$T_{3,1}$	$T_{3,2}$	$T_{4,1}$	$T_{9,1}$	$T_{10,1}$
1.1.	√	√	√	√	√	√	√	√	√	√	√
1.2.	√	√	√	√	√	√	√	√	√	√	√
1.3.	√	√	√	√	√	√	√	√	√	√	√
1.4.	√	√	√	√	√	√	√	√	√	√	√
1.5.	√	√	√	√	√	√	√	√	√	√	√
2.1.1.	○	○	√	○	√	○	○	○	○	○	√
2.1.2.	○	○	√	○	√	○	○	○	○	○	√
2.1.3.	○	○	√	○	√	○	○	○	○	○	√
2.1.4.	○	○	√	○	√	○	○	○	○	○	√
2.2.1.	√	√	○	√	○	○	√	√	√	√	○
2.2.2.	√	√	○	√	○	○	√	√	√	√	○
2.2.3.	×	×	○	√	○	○	×	√	×	×	○
2.2.4.	√	×	○	√	○	○	×	√	×	√	○
2.3.1.	○	○	○	○	○	○	○	○	○	○	√
2.3.2.	○	○	○	○	○	○	○	○	○	○	√
3.1.1.	√	○	○	○	○	○	○	○	○	√	○
3.1.2.	√	○	○	○	○	○	○	○	○	√	○
3.1.3.	√	○	○	○	○	○	○	○	○	√	○
3.1.4.	√	○	○	○	○	○	○	○	○	√	○
3.1.5.	√	○	○	○	○	○	○	○	○	√	○
3.2.1.	○	√	√	○	○	○	√	○	√	○	√
3.2.2.	○	√	√	○	○	○	√	○	√	○	√
3.2.3.	○	×	×	○	○	○	×	○	×	○	×
4.1.1.	○	○	○	○	○	○	○	○	○	○	√
4.1.2.	○	○	○	○	○	○	○	○	○	○	√
4.1.3.	○	○	○	○	○	○	○	○	○	○	√
4.2.1.	○	○	○	○	○	√	○	√	√	√	○
4.2.2.	○	○	○	○	○	√	○	√	√	√	○
4.2.3.	○	○	○	○	○	√	○	√	√	√	○
5.1.1.	○	○	○	√	√	○	√	√	√	√	√
5.1.2.	○	○	○	√	√	○	√	√	√	√	√
5.1.3.	○	○	○	√	√	○	√	√	√	√	√
5.2.1.	○	○	○	○	○	○	○	○	○	√	√
5.2.2.	○	○	○	○	○	○	○	○	○	√	√
6.1.	○	○	○	○	×	×	○	×	○	×	×
6.2.	○	○	○	×	○	○	○	○	×	×	×
6.3.	○	○	○	√	○	○	○	○	√	√	√
6.4.	○	○	○	○	○	○	×	○	○	×	×

Tabelle 6.3 – Ergebnis der Testfälle $T_{2,1}$ bis $T_{10,1}$

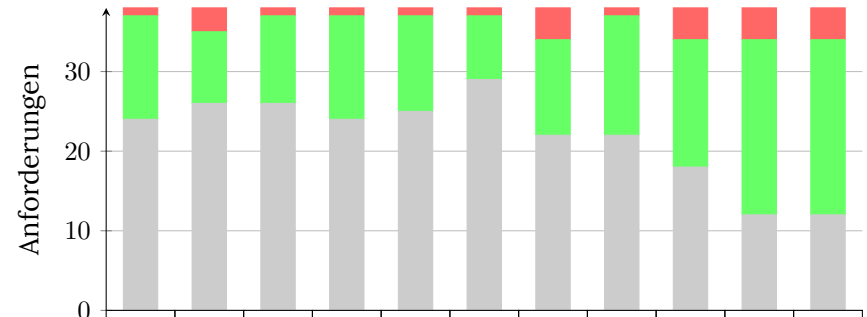
6.3 Auswertung der Evaluationen

Die im vorherigen Abschnitt ermittelten Daten der Evaluation aus den Tabellen 6.2 und 6.3 werden nun anschaulicher gestaltet. Dafür werden für jeden Testfall gezählt, wie häufig eine Anforderung nicht relevant war, erfüllt oder nicht erfüllt wurde. Diese Zusammenfassungen werden in den Abbildungen 6.1 und 6.2 in Form zweier Säulendiagramme dargestellt. Zusätzlich zur grafischen Darstellung enthalten die Abbildungen eine Datentabelle, sodass exakte Werte direkt abgelesen werden können.



	$T_{0,1}$	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$	$T'_{1,3}$	$T_{1,4}$	$T_{1,5}$	$T_{1,6}$	$T_{1,7}$	$T_{1,8}$	$T_{1,9}$	$T_{1,10}$	$T_{1,11}$	$T_{1,12}$	$T_{1,13}$
nicht erfolgreich	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
erfolgreich	5	9	7	9	9	10	8	8	8	6	10	9	9	9	8
irrelevant	33	29	31	29	29	28	30	30	30	32	28	29	29	29	30

Abbildung 6.1 – Auswertung Testfälle $T_{0,1}$ bis $T_{1,13}$



	$T_{2,1}$	$T_{2,2}$	$T_{2,3}$	$T_{2,4}$	$T_{2,5}$	$T_{2,6}$	$T_{3,1}$	$T_{3,2}$	$T_{4,1}$	$T_{9,1}$	$T_{10,1}$
nicht erfolgreich	1	3	1	1	1	1	4	1	4	4	4
erfolgreich	13	9	11	13	12	8	12	15	16	22	22
irrelevant	24	26	26	24	25	29	22	22	18	12	12

Abbildung 6.2 – Auswertung Testfälle $T_{2,1}$ bis $T_{10,1}$

6.4 Interpretation der Ergebnisse

Abbildung 6.1 betrachtet lediglich die Testfälle, in denen keine oder nur eine gleichzeitige Einschränkung getestet wurde. Hier erkennt man, dass keine der Säulen einen roten Anteil besitzt und somit jede Anforderung entweder erfüllt wurde oder für den Testfall keine Relevanz hatte.
Die Testfälle, die in Abbildung 6.2 dargestellt sind, wurden in der Vorbereitung der Evaluation so gewählt, dass es in jedem Fall zu Konflikten führt. Der Prototyp wurde durch diese Testfälle auf Konfliktsituationen getestet, bei der es keine mögliche Variante gibt, für die alle Anforderungen erfüllt sind. Demnach wurde bereits erwartet, dass jeder Testfall mindestens eine Anforderung nicht erfüllen wird.
Prinzipiell lässt sich sagen, dass bei höherer Zahl der gleichzeitigen Einschränkungen auch die Anzahl an nicht erfüllten Anforderungen steigt. So werden in den Testfällen $T_{2,1}$, $T_{2,3}$, $T_{2,4}$, $T_{2,5}$, $T_{2,6}$ und $T_{3,2}$ jeweils nur eine Anforderung nicht erfüllt. Bei den beiden Testfällen $T_{9,1}$ und $T_{10,1}$, bei welchen je eine Variante mit möglichst vielen gleichzeitigen Einschränkungen betrachtet wurde, wurden jeweils vier Anforderungen nicht erfüllt.
Nicht nur die Zahl der gleichzeitigen Einschränkungen eines Nutzers, sondern auch die Art der Beeinträchtigung sind für die Zahl der nicht erfüllten Anforderungen relevant. So weisen die Testfälle $T_{2,2}$, $T_{2,6}$ und $T_{4,1}$ drei beziehungsweise vier Fehler auf. Diese Testfälle haben die Gemeinsamkeit, dass hier eine gleichzeitige *Blindheit* und *Gehörlosigkeit* betrachtet wurde. Diese Kombination fordert jeweils eine gleichzeitige Aktivierung und Deaktivierung der visuellen und akustischen Ausgabe und führt daher vermehrt zu Konflikten.
Betrachtet man Tabelle 6.3 aber genauer, sieht man, dass durch die Testfälle $T_{2,1}$ bis $T_{10,1}$ nur folgende 6 Anforderungen nicht erfüllt wurden: 2.2.3., 2.2.4., 3.2.3., 6.1., 6.2. und 6.4.. All diese Einschränkungen beginnen mit der Klausel, dass diese nur dann greifen sollen, wenn es ansonsten zu keinem Widerspruch führt und besitzen eine Priorität, die kleiner als 1 ist. Da es in allen durchgeführten Testfällen aus Tabelle 6.3 ansonsten zu Widersprüchen geführt hätte, ist es daher in Ordnung, dass die Bedingungen dieser 6 Anforderungen nicht immer umgesetzt wurden.

6.5 Zusammenfassung

Die technische Evaluation hat in 25 unterschiedlichen Testfällen gezeigt, dass das Konzept in den 14 Fällen, die ohne Konflikte eine mögliche Interaktionsvariante finden, diese ausgibt. Für die 11 Testfälle, die zu Problemen führten, hat das System Schritt für Schritt einzelne Anforderungen ignoriert, sodass dennoch eine möglichst passende Variante zur Verfügung gestellt werden konnte. Insgesamt wurden dabei alle Anforderungen erfüllt, da nur die Regeln weggelassen wurden, die dies anhand ihrer Formulierung auch zulassen.
Nach diesem Test auf Eignung folgt ein abschließender Teil, der die Arbeit zusammenfasst und bewertet.

7 Zusammenfassung

Nachdem das Konzept evaluiert und ausgewertet wurde, soll dieses Kapitel die bearbeiteten Punkte zusammenfassen und in einer Diskussion offene Fragen, die durch das vorgestellte Konzept nicht beachtet wurden zeigen. Anschließend wird die Forschungsfrage aufgegriffen und entschieden, ob diese durch die vorliegende Arbeit beantwortet wurde. Abschließend soll dieses Kapitel einen Blick auf mögliche weitere Arbeiten zu dem bearbeiteten Thema für die Zukunft werfen.

7.1 Zusammenfassung der Arbeit

Zur Entlastung des Pflegepersonals ist der Einsatz von SAR denkbar. Da deren Zielgruppe aufgrund verschiedener Einschränkungen unterschiedliche Anforderungen aufweist, sollte sich die Interaktion zwischen SAR und Benutzer an den jeweiligen Nutzer adaptiv anpassen. Anhand dieser Informationen wurde eine Forschungsfrage erstellt und Teilziele, die die Arbeit erfüllen sollte, definiert.
Die Arbeit begann mit der Vorstellung der Grundlagen, indem zunächst unterschiedliche Arten an Robotern vorgestellt und SAR anhand verschiedener Kategorisierungen eingeordnet wurden. Darauf folgte eine Vorstellung unterschiedlicher Einschränkungen älterer Menschen. Diese wurden in *sensorisch*, *körperlich* und *kognitiv* unterteilt. Um die Variabilität eines Systems abzubilden wurden Software-Produktlinien und Feature-Modelle als Variabilitätsmodell vorgestellt. Für Feature-Modelle wurden außerdem durch Kardinalitäten und Attribute zwei Erweiterungskonzepte gezeigt. Eine kurze Beleuchtung der symbolischen Künstlichen Intelligenz schloss das Grundlagenkapitel ab.
Um zu wissen, welche Schritte zur Erstellung eines KI-basierten Adaptionsalgorithmus zur Interaktionsanpassung notwendig sind, wurden in Kapitel 3 zunächst die Anforderungen, die ältere Menschen an die Interaktion mit SAR stellen, herausgearbeitet und strukturiert gesammelt. Anschließend wurde erläutert, wie die Einschränkungen der Nutzer erfasst und gespeichert werden sollen. Es wurde sich dafür entschieden, dass alle möglichen Einschränkungen in der Form eines Feature-Modells (FM_{USER}) gesammelt werden. Für jeden Nutzer wird nun eine Variante dieses Feature-Modells gespeichert, die beschreibt, welche Einschränkungen ein Nutzer aufweist. Auch die verschiedenen Interaktionsmöglichkeiten sollen durch ein Feature-Modell (FM_{SYSTEM}) beschrieben werden. Beide Feature-Modelle sollen durch einen Regelsatz miteinander verbunden werden, sodass anhand einer Variante von FM_{USER} auf eine Konfiguration von FM_{SYSTEM} geschlossen werden kann. Das Kapitel wurde mit der Vorstellung, wie die Kanten innerhalb eines Feature-Modells als aussagenlogische Formeln dargestellt werden können, abgeschlossen. Dort wurde auch darauf eingegangen, wie ein Feature-Modell anhand einer teilweisen Initialkonfiguration weiter konfiguriert werden kann, sodass weiterhin alle Regeln erfüllt sind.
Es folgte die Ausarbeitung eines an die zuvor erarbeiteten Anforderungen passendes Konzepts. Dabei wurde zunächst das Feature-Modell FM_{USER} erstellt. Wie auch bei der Vorstellung der möglichen Einschränkungen aus dem Grundlagenkapitel, wurden die Beeinträchtigungen hier in die

Kategorien *sensorisch*, *körperlich* und *kognitiv* unterteilt. Neben FM_{USER} wurde auch FM_{SYSTEM} aufgestellt, indem alle Interaktionsmöglichkeiten, die in den Anforderungen genannt wurden, beachtet wurden. Zur besseren Übersicht wurde dieses Modell in Teildiagramme unterteilt, die jeweils die *Eingabe*, die *Ausgabe* und die *Verarbeitung* beschreiben. Als nächstes wurde der Regelsatz aufgestellt, der beschreibt, welche Interaktionsvariante bei bestimmten Einschränkungen verwendet werden muss oder sollte. Dazu wurden 41 Regeln aufgestellt, die die beiden Feature-Modelle miteinander verbinden sollen. Durch das Ablaufdiagramm aus Abbildung 4.6 wurde der überlegte Adaptionsalgorithmus vorgestellt, der anhand der Benutzereinschränkungen und des Regelsatzes das Feature-Modell FM_{SYSTEM} konfigurieren und somit eine passende Variante der Interaktion finden sollte.
Der nächste bearbeitete Punkt war die Erstellung eines Prototyps, der das Konzept abbilden sollte. Zunächst wurde hier die Entscheidung für Alexa in Verbindung mit RASA getroffen, diese begründet und die Verbindung der beiden Systeme erläutert. Darauf folgend wurden die möglichen Funktionen des Prototyps aufgezeigt und die Architektur beschrieben, die dafür notwendig war, die Funktionen umzusetzen. Der Prototyp lässt eine Änderung aller aus FM_{USER} bekannten Einschränkungen zu und führt bei einer Änderung automatisch eine Adaption der Interaktion durch. Mithilfe einer Testaktion lassen sich unterschiedliche Interaktionsvarianten testen.
Durch den entwickelten Prototyp konnte eine technische Evaluation durchgeführt werden, bei der in 25 Testfällen alle 38 Anforderungen überprüft wurden. Dabei wurde festgestellt, dass sich das System bei einfachen Testfällen, die nur wenige Einschränkungen testen, wie erwartet verhält und alle Anforderungen erfüllt werden. Bei schwierigeren Testfällen, die so gewählt wurden, dass ein Konflikt auftreten wird und mindestens eine Anforderung nur teilweise erfüllt wird, war zu erkennen, dass die Anzahl der nicht erfüllten Anforderungen bei größerer Zahl an gleichzeitig getesteten Einschränkungen ebenfalls steigt. Man konnte außerdem sehen, dass die Kombination der beiden Einschränkungen *Blindheit* und *Gehörlosigkeit* ebenfalls zu größeren Problemen führte. Die Anforderungen, die während der durchgeführten Testfälle nicht erfüllt wurden, waren jedoch so konzipiert, dass diese nur in dem Fall greifen sollten, wenn es ansonsten zu keinem Widerspruch führen würde. Aus diesem Grund war es in Ordnung, dass diese Anforderungen nicht erfüllt wurden. Somit wurde gezeigt, dass das Konzept wie erwartet funktioniert und anhand der Einschränkungen die Interaktion mit dem Nutzer passend ändert.

7.2 Diskussion der Ergebnisse

Wie in der Evaluation zu sehen war, erfüllt das entwickelte Konzept die Anforderungen. Auch in Konfliktfällen erkennt der Algorithmus, dass er einzelne Forderungen des Regelsatzes ignorieren soll, um dem Nutzer trotz eines Widerspruchs eine dennoch funktionierende Variante der Interaktion zur Verfügung zu stellen. Dafür wurden einzelnen Regeln eine niedrigere Priorität zugewiesen, sodass der SAR diese Stück für Stück ignorieren kann, bis die erarbeitete Lösung keinen Fehler aufweist. Dadurch wird es möglich, dass ein SAR von unterschiedlichen Menschen benutzt werden kann, die verschiedene Einschränkungen haben. Auch die Akzeptanz von Robotern in Pflegeeinrichtung könnte durch adaptiv angepasste Interaktionen steigen und somit das bisherige Pflegepersonal entlasten.

Es ergeben sich aber gleichzeitig auch Fragen, die durch diese Arbeit noch nicht geklärt wurden. So wird hier zunächst auf die betrachteten Einschränkungen eingegangen.

In dieser Arbeit wurde nur der Teil der möglichen Einschränkungen eines Menschen betrachtet, der im Zusammenhang mit der MRI für relevant gehalten wurde. Das Konzept lässt sich hier aber an neue, noch nicht betrachtete Anforderungen anpassen. Dafür müsste die neue Einschränkung in das Feature-Modell FM_{USER} eingefügt werden. Die Anforderungen, die sich mit der neuen Einschränkung ergeben, müssen in den Regelsatz $C_{ANFORDERUNGEN}$ aufgenommen werden. Falls sich durch die neuen Anforderungen Änderungen im Modell FM_{SYSTEM} ergeben, müssen diese auch hier nachgetragen werden. Eine Änderung des Adaptionsalgorithmus ist nicht notwendig. Es bleibt an dieser Stelle jedoch die Frage: *Sollten weitere Einschränkungen, als die verwendeten, beachtet werden?*
Des weiteren hat die Arbeit eine Einschränkung als booleschen Wert gesehen. Ein Nutzer kann somit entweder eine Sehbeeinträchtigung haben oder nicht. Über den Grad der jeweiligen Einschränkung wurde kein Wert gespeichert. Es bleibt hier die offene Frage im Raum: *Wie könnte das Konzept angepasst werden, um zu speichern und zu beachten, wie stark eine Einschränkungen ausgeprägt ist?*
Der letzte Punkt, der die Einschränkungen des Nutzers thematisiert, ist, wie die Erhebung der Beeinträchtigungen erfolgen sollte. Die Arbeit hat angenommen, dass diese per manuelle Eingabe durch den Nutzer oder durch externes Personal erfolgen soll. Eine automatische Erkennung der Einschränkungen oder eines Teils der Beeinträchtigungen wäre aber ebenfalls denkbar. Es bleibt die Frage: *Wie sollte dem System eine Nutzereinschränkung übergeben werden?*

Ebenfalls kann über den Adaptionsalgorithmus diskutiert werden. Ein mögliches Problem ergibt sich, sobald der verwendete Algorithmus auf einen Widerspruch stößt. In diesem Fall wird die Regel mit dem niedrigsten Gewicht ignoriert. Es wird dabei jedoch nicht überprüft, ob diese Regel wirklich dafür verantwortlich war, dass das System keine passende Konfiguration gefunden hat. Dadurch könnte es vorkommen, dass Regeln unnötigerweise nicht betrachtet werden. *Wie könnte der erarbeitete Konfigurationsmechanismus in dieser Hinsicht verbessert werden?*
Ein weiterer Punkt in diesem Zusammenhang ist die Konfiguration einer Oder-Gruppe. Die aktuelle, zufällige Auswahl einer Konfiguration könnte in ein deterministisches Verfahren geändert werden. Hier stellt sich die Frage: *In welcher Form könnte die Konfiguration einer Oder-Gruppe innerhalb eines Feature-Modells verbessert werden?*

Letztlich werden Aspekte des Prototyps diskutiert. Während das Konzept einen SAR betrachtete, wurde der Prototyp nur als Sprachassistent gestaltet. Die Adaption konnte hierbei zum Beispiel durch die Art der Formulierung oder die Sprecherwahl erkannt werden. Gerade die Aktivierung der Braillezeile oder der Gestenerkennung ist durch den Prototyp schlecht darstellbar. Mit Hilfe der Textausgabe von Alexa wurden aber Debug-Werte bei jedem Dialogschritt ausgegeben, um die Veränderung der Adaption zu zeigen.
Die Speicherung eines Feature-Modells innerhalb des Prototyps erfolgt durch Angabe einer Liste aller Features und aller Regeln. Somit kann das Modell zwar gespeichert und verwendet werden; eine Speicherung als baumartige Struktur, wie sie Seidl in seiner Dissertation [Sei15] vorgestellt hat, wäre aber sauberer. Auch eine Trennung von Struktur und Daten wäre eine mögliche Verbesserung. Aktuell wird eine Negation durch `NegiertesFeature('A')` gespeichert. Es müssen daher für das Feature `'A'` zwei Objekte erzeugt werden, um es positiv und negativ abzubilden. Eine bessere Variante wäre `Negation(Feature('A'))`, wobei `Negation` ein Strukturelement ist.

7.3 Fazit

Die Forschungsfrage *Wie sehen die Regelsätze und der Konfigurationsmechanismus aus, die notwendig sind, damit sich ein SAR während der Laufzeit an die Einschränkungen und damit verbundenen Anforderungen eines Nutzers anpassen kann*?, welche in Kapitel 1 aufgestellt wurde, soll in diesem Abschnitt beantwortet werden.
Um diese Frage beantworten zu können, war es zunächst notwendig, die möglichen Einschränkungen und damit verbundenen Anforderungen an die Interaktion mit einem SAR zu recherchieren und zusammenzufassen. Dieser Schritt wurde in Abschnitt 3.1 erledigt.
Es konnte durch Abbildung 3.1 gezeigt werden, wie aus diesen Anforderungen die beiden Feature-Modelle FM_{USER} (siehe Abbildung 4.1) und FM_{SYSTEM} (siehe Abbildungen 4.2, 4.3, 4.4 und 4.5) und ein Regelsatz, welcher die beiden Modelle verbindet, erstellt werden können. Der Regelsatz (siehe Tabelle 4.1) ist hierbei eine Übersetzung der definierten Anforderungen in maschinenlesbare Formeln, die auf Features und Attribute der beiden Feature-Modelle zugreifen.
Die beiden Feature-Modelle und der Regelsatz stellen hierbei den ersten Teil des in Kapitel 4 erarbeiteten Konzepts dar. Somit ist es möglich, jede Kombination aus Einschränkungen eines Menschen durch eine Variante von FM_{USER} abzubilden. Ebenso kann man durch eine Variante von FM_{SYSTEM} die Variabilität der Interaktion des SAR zeigen.

Somit wurde der erste Teil der Forschungsfrage, der die Konstruktion eines Regelsatzes forderte, beantwortet. Der zweite Teil der Frage, der einen Konfigurationsmechanismus verlangt, kann mit dem in Abbildung 4.6 dargestellten Adaptionsalgorithmus beantwortet werden. Diese Übersicht zeigt einen Mechanismus, der versucht, anhand des definierten Regelsatzes und der gespeicherten Nutzereinschränkungen als Variante von FM_{USER} auf eine gültige Konfiguration von FM_{SYSTEM} zu schließen.
Dabei wird anhand der gewählten und nicht gewählten Features von FM_{USER} durch Verwendung des erarbeiteten Regelsatzes eine Initialkonfiguration für FM_{SYSTEM} geschaffen. Durch Nutzung der von Hubaux vorgestellten featureorientierten Konfiguration eines Feature-Modells [Hub12] wird durch iteratives Anwenden aller Regeln, die die Beziehungen innerhalb von FM_{SYSTEM} zeigen, das Modell vollständig konfiguriert und somit eine mögliche Interaktionsvariante ausgegeben. Bei einem Misserfolg werden nach und nach Regeln, die eine niedrige Priorität haben, entfernt, bis das System entweder eine valide Lösung findet oder erfolglos abbricht.

7.4 Ausblick

Die Diskussion in Abschnitt 7.2 zeigt, dass offene Fragen existieren, die die vorliegende Arbeit erweitern würden.
Es wurde bereits ein umfassender Teil der möglichen Einschränkungen beachtet. Dabei wurde sich an die Strukturierung des Guide 71 gehalten. Eine mögliche Verbesserung des erarbeiteten Konzepts wäre eine Ergänzung des aktuellen Modells durch weitere Einschränkungen, die ein Nutzer haben kann.
Die Stärke der Einschränkung wurde in dieser Arbeit nicht weiter beachtet. Um diesen Aspekt zu betrachten, könnte dem Modell FM_{USER} eine Feature-Kardinalität hinzugefügt werden, die aussagt, wie stark die jeweilige Einschränkung bei einem Nutzer ausgeprägt ist. Der Regelsatz müsste außerdem angepasst werden, um nicht nur, wie bisher, die Auswahl eines Features, sondern auch

dessen Kardinalität zu beachten und anhand derer eine passende Interaktionsvariante zu finden.

Der erarbeitete Konfigurationsmechanismus löscht bei Misserfolg die Regel des Regelsatzes, die die niedrigste Priorität aufweist. Ein interessanter Gedanke wäre es, diese Logik in einer zukünftigen Arbeit so zu ändern, dass zunächst geprüft wird, aufgrund welcher Regeln das System keine passende Variante findet und einen Widerspruch meldet. Aus diesen Regeln müsste das System intelligent die Regeln aussuchen, die ignoriert werden sollen. Hierbei bleibt die Frage, wie diese Auswahl erfolgen sollte. Einerseits könnte wieder eine Variante verwendet werden, die jeder Regel eine Priorität zuweist, anhand derer aussortiert wird. Auch möglich wäre aber ein Verfahren, das die Regeln so löscht, dass möglichst wenige Regeln nicht beachtet werden.

Abkürzungsverzeichnis

AR	Assistenzroboter	3
AWS	Amazon Web Services	40
DBSV	Deutscher Blinden- und Sehbehindertenverband	14
DNS	Domain-Namen-System	40
KI	Künstliche Intelligenz	I
MRI	Mensch-Roboter-Interaktion	I
SAR	Sozialer Assistenzroboter	I
SIR	Sozialer Interaktiver Roboter	4
SPL	Software-Produktlinie	7
TTS	Text To Speech	30
UI	User Interface	6
URL	Uniform Resource Locator	49
VPN	virtuelles privates Netzwerk	40
VUI	Voice User Interface	16
WCAG	Web Content Accessibility Guidelines	14

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Literatur

[AA04] Andrew Arch und Shadi Abou-Zhara. “How Web Accessibility Guidelines Apply to Design for the Ageing Population”. In: *World Wide Web Consortium* (2004) (siehe S. 6 f., 16).

[Bos20] Stefan Bosse. *Einführung in die Künstliche Intelligenz*. Aug. 2020 (siehe S. 12).

[BSR09] David Benavides, Sergio Segura und Antonio Ruiz-Cort´es. *Automated Analysis of Feature Models: A Detailed Literature Review*. 2009 (siehe S. 11).

[BTR05] David Benavides, Pablo Trinidad und Antonio Ruiz-Cortes. “Automated Reasoning on Feature Models”. In: *Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE*. 2005. URL: https://doi.org/10.1007/978-3-642-36926-1_29 (siehe S. 9–11).

[CE99] Krzysztof Czarnecki und Ulrich W. Eisenecker. “Components and generative programming (invited paper)”. In: ACM *SIGSOFT Software Engineering Notes* 24.6 (Sep. 1999), S. 2–19. DOI: 10.1145/318774.318779. (Besucht am 10. 02. 2023) (siehe S. 8 f.).

[CHE05] Krzysztof Czarnecki, Simon Helsen und Ulrich Eisenecker. “Formalizing cardinality-based feature models and their specialization”. In: *Software Process: Improvement and Practice* 10.1 (März 2005), S. 7–29. DOI: 10.1002/spip.213 (siehe S. 10).

[Chu+15] Soon Hau Chua, Haimo Zhang, Muhammad Hammad, Shengdong Zhao, Sahil Goyal und Karan Singh. “ColorBless: Augmenting Visual Information for Colorblind People with Binocular Luster Effect”. In: ACM *Transactions on Computer-Human Interaction* 21.6 (Jan. 2015), S. 1–20. DOI: 10.1145/2687923 (siehe S. 14).

[CL08] Anna Cavender und Richard E. Ladner. “Hearing Impairments”. In: (2008) (siehe S. 6, 15).

[Coo+20] Sara Cooper, Alessandro Di Fava, Carlos Vivas, Luca Marchionni und Francesco Ferro. “ARI: the Social Assistive Robot and Companion”. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2020, S. 745–751. DOI: 10.1109/RO-MAN47096.2020.9223470 (siehe S. 4 f.).

[Fel+21] Alexander Felfernig, Viet-Man Le, Andrei Popescu, Mathias Uta, Thi Ngoc Trang Tran und Müslüm Atas. “An Overview of Recommender Systems and Machine Learning in Feature Modeling and Configuration”. In: *Proceedings of the 15th International Working Conference on Variability Modelling of Software-Intensive Systems*. Feb. 2021, S. 1–8. DOI: 10.1145/3442391.3442408 (siehe S. 23).

[Fla+22] B Judy Flavia, P Kamali Priya, Nayan Chandravanshi und M Vaishnavi Amirtham. “Hand Gesture Recognition using AI Algorithm for Hearing Impaired”. In: *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*. Sep. 2022, S. 1069–1072. DOI: 10.1109/ICIRCA54612.2022.9985748 (siehe S. 15).

[Fla16] Mariusz Flasiński. *Introduction to Artificial Intelligence*. ISBN: 978-3-319-40020-4. Springer International Publishing, 2016 (siehe S. 12).

[FM05] D. Feil-Seifer und M.J. Mataric. "Defining socially assistive robotics". In: *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.* ISSN: 1945-7901. Juni 2005, S. 465–468. DOI: 10.1109/ICORR.2005.1501143 (siehe S. 1, 3 f.).

[FND02] Terrence Fong, Illah Nourbakhsh und Kerstin Dautenhahn. "A Survey of Socially Interactive Robots: Concepts, Design, and Applications". In: (Nov. 2002), S. 58. URL: http://hdl.handle.net/2299/3821 (siehe S. 4).

[Fuc21] Julian Fuchs. *Entwicklung von Nutzerprofilen für adaptive Assistenzroboter*. Mai 2021 (siehe S. 14, 19).

[GFd98] Martin L. Griss, John Favaro und Massimo d'Alessandro. "Integrating feature modeling with the RSEB". In: *Proceedings. Fifth International Conference on Software Reuse (Cat. No.98TB100203)*. Juni 1998, S. 76–85. DOI: 10.1109/ICSR.1998.685732 (siehe S. 9).

[Grö21] Tim Gröger. *Konzeption eines Konfigurationsprozesses zur adaptiven Interaktion mit Assistenzrobotern*. Jan. 2021 (siehe S. 7, 14 f.).

[GS19] Marta Garnelo und Murray Shanahan. "Reconciling deep learning with symbolic artificial intelligence: representing objects and relations". In: *Current Opinion in Behavioral Sciences*. Artificial Intelligence 29 (Jan. 2019), S. 17–23. DOI: 10.1016/j.cobeha.2018.12.010 (siehe S. 11).

[GW21] David Gollasch und Gerhard Weber. "Age-Related Differences in Preferences for Using Voice Assistants". In: *Mensch und Computer 2021*. Sep. 2021, S. 156–167. DOI: 10.1145/3473856.3473889 (siehe S. 16).

[Hee10] Marcel Heerink. "Assessing acceptance of assistive social robots by aging adults". Diss. Universiteit van Amsterdam, Nov. 2010. URL: http://hdl.handle.net/11245/1.327918 (siehe S. 1, 3 f.).

[Hub12] Arnaud Hubaux. "Feature-based Configuration: Collaborative, Dependable, and Controlled". Diss. University of Namur, Jan. 2012. URL: http://hdl.handle.net/2078.2/105420 (siehe S. 24, 35, 62).

[Kan+90] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak und A. S. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Techn. Ber. Defense Technical Information Center, Nov. 1990. URL: https://apps.dtic.mil/sti/citations/ADA235785 (siehe S. 8).

[KC03] J.O. Kephart und D.M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (Jan. 2003), S. 41–50. DOI: 10.1109/MC.2003.1160055 (siehe S. 13).

[Lie21] Robert Liefke. *Altersspezifische Strategien zur Umsetzung multimodaler CUIs*. Dez. 2021 (siehe S. 1).

[Lin+10] Ulman Lindenberger, Jacqui Smith, Karl Ulrich Mayer und Paul B. Baltes. *Die Berliner Altersstudie*. ISBN: 978-3-05-004508-5. 2010 (siehe S. 7).

[Loi18] Claudia Loitsch. "Designing Accessible User Interfaces for All by Means of Adaptive Systems". Diss. Technische Universität Dresden, Feb. 2018. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-319846 (siehe S. 14).

[LSR07] Frank van der Linden, Klaus Schmid und Eelco Rommes. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. ISBN: 978-3-540-71436-1. Springer, 2007 (siehe S. 7 f.).

[LY07] Xz Liu und D Yan. "Ageing and hearing loss". In: *The Journal of Pathology* 211.2 (2007), S. 188–197. DOI: 10.1002/path.2102 (siehe S. 15).

[Miš+20] Justinas Mišeikis, Pietro Caroni, Patricia Duchamp, Alina Gasser, Rastislav Marko, Nelija Mišeikienė, Frederik Zwilling, Charles de Castelbajac, Lucas Eicher, Michael Früh und Hansruedi Früh. "Lio-A Personal Robot Assistant for Human-Robot Interaction and Care Applications". In: *IEEE Robotics and Automation Letters* 5.4 (Okt. 2020), S. 5339–5346. DOI: 10.1109/LRA.2020.3007462 (siehe S. 4 f.).

[MS16] Maja J Matari und Brian Scassellati. "Socially Assistive Robotics". In: *Handbook of Robotics*. 2. Aufl. Bd. 73. Springer, 2016, S. 1973–1993. URL: https://doi.org/10.1007/978-3-319-32552-1_73 (siehe S. 1, 4).

[Myö19] Aleksi Myöhänen. *Loudness Compensation*. 2019 (siehe S. 14).

[PBL05] Klaus Pohl, Günter Böckle und Frank van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. 1. Aufl. ISBN: 978-3-540-24372-4. Springer, 2005 (siehe S. 7).

[PRG14] Afra Pascual, Mireia Ribera und Toni Granollers. "Impact of web accessibility barriers on users with hearing impairment". In: *Proceedings of the XV International Conference on Human Computer Interaction*. Sep. 2014, S. 1–2. DOI: 10.1145/2662253.2662261 (siehe S. 6, 14).

[Seh19] Deutscher Blinden- und Sehbehindertenverband e.V. *Leserlich*. Dez. 2019. URL: www.leserlich.info (besucht am 10. 02. 2023) (siehe S. 14).

[Sei15] Christoph Seidl. "Integrated management of variability in space and time in software families". Diss. Technische Universität Dresden, Juli 2015. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-218036 (siehe S. 61).

[SGB20] Negoslav Sabev, Galya Nikolova Georgieva-Tsaneva und Galina Bogdanova. "Research, analysis, and evaluation of web accessibility for a selected group of public websites in Bulgaria". In: *Journal of Accessibility and Design for All* 10.1 (Mai 2020), S. 124–160. DOI: 10.17411/jacces.v10i1.215 (siehe S. 6, 14).

[Sta12] International Organization for Standardization. *Robots and robotic devices - Vocabulary*. ISO 8373. März 2012 (siehe S. 3).

[Sta14] International Organization for Standardization. *Guide for addressing accessibility in standards*. ISO 71. Dez. 2014 (siehe S. 5–7, 14–16).

[Str21] Adam Strantz. "Using Web Standards to Design Accessible Data Visualizations in Professional Communication". In: *IEEE Transactions on Professional Communication* 64.3 (Sep. 2021), S. 288–301. DOI: 10.1109/TPC.2021.3091784 (siehe S. 6, 14).

[TBK09] Thomas Thüm, Don Batory und Christian Kästner. "Reasoning about edits to feature models". In: *2009 IEEE 31st International Conference on Software Engineering*. Mai 2009, S. 254–264. DOI: 10.1109/ICSE.2009.5070526 (siehe S. 9).

[Thi19] Vincent Julius Thiele. *Nutzererkennung zur Individualisierung der Mensch-Roboter-Interaktion*. de. März 2019 (siehe S. 19).

[Uni20] United Nations Department of Economic and Social Affairs. *World Population Ageing 2020*. en. ISBN: 978-92-1-148347-5. New York: United Nations Publication, 2020 (siehe S. 1, 4).

[Vel+20] Kostandina Veljanovska, Natasha Blazheska-Tabakovska, Blagoj Ristevski und Snezana Savoska. "User Interface for e-learning Platform for Users with Disability". In: *ISGT 2020 Information Systems and Grid Technologies* (2020). URL: `http://eprints.uklo.edu.mk/id/eprint/5828` (siehe S. 15).

[WCA] WCAG. *Understanding Success Criterion 1.4.1: Use of Color | WAI | W3C*. URL: `https://www.w3.org/WAI/WCAG21/Understanding/use-of-color.html` (besucht am 19. 02. 2023) (siehe S. 14).

[Zim17] Randall Ziman. *Factors Affecting Seniors' Perceptions of Voice User Interfaces*. Dez. 2017. URL: `http://hdl.handle.net/11603/7721` (siehe S. 7).