Abstract

Ziel der vorliegenden Masterarbeit war es, ein Konzept und Implementierung eines Sprachassistenten zur Unterstützung des Vertriebsprozesses zu entwickeln. Dies wurde am Beispiel der SAP Cloud for Customer umgesetzt. Dazu wurden sowohl Analysen der Herausforderungen von Sprachinteraktion als auch darauf aufbauenden Designrichtlinien für die Umsetzung solcher Anwendungen angestellt und anwendungsspezifische use cases und Anforderungen erarbeitet. Darüber hinaus bietet die Arbeit eine Analyse technologischer Aspekte, wie einem cross-platform Framework und einem Vergleich verschiedener Plattformen. Aufbauend darauf wird auf die praktische Umsetzung dieser Aspekte eingegangen, evaluiert, und Ansätze für zukünftige Weiterentwicklungen des hier entstandenen Prototyps herausgearbeitet.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation und Zielstellung

Sprachassistenten wie Amazon Alexa oder Google Assistant sind seit einiger Zeit in aller Munde und können ihren Marktanteil stetig ausbauen, was beispielsweise in der Verbreitung von smart speakern sichtbar wird [18]. Im Vertriebsbereich, beispielsweise in der Kommunikation mit der SAP Cloud for Customer, wird jedoch hauptsächlich mit graphischen Oberflächen gearbeitet. Da insbesondere Angestellte im Vertrieb an unterschiedlichen Orten arbeiten, welche eine Interaktion mit einer graphischen Oberfläche nicht erlauben, wie etwa im Auto auf dem Weg zu einem Kunden, bietet sich hierbei die Möglichkeit der Sprachinteraktion mit dem System an.

Ein Vertriebsmitarbeiter kann sich so zum Beispiel bereits im Auto auf dem Weg zum Kunden über den Kunden, den Termin, offene Angebote, o. ä. informieren, womit diese Zeit effektiv nutzbar wird. Im Hinblick darauf, dass die Kommunikation via Sprache aufgrund verschiedener Herausforderungen (siehe 3.1) nur situativ der graphischen Oberfläche vorzuziehen ist, sollte eine bestehendes System mit herkömmlichen Interkationsmöglichkeiten um diesen Kommunikationskanal erweitert werden. Hierfür bietet sich die SAP Cloud for Customer als cloud basierte Plattform mit einer Schnittstelle für Datenzugriffe an.

1.2 Überblick der Arbeit

Relevante Konzepte und Begrifflichkeiten für die folgende Arbeit werden innerhalb der Grundlagen (siehe 2) erläutert, welches sowohl auf Termino-

logien der Sprachanwendungen, als auch die der SAP Cloud for Customer eingeht. Danach werden verwandte Arbeiten (siehe 3) aus der Wissenschaft und Praxis untersucht, welche einen Bezug zu dem im Kontext der Arbeit behandelten Anwendungsfall haben.

Im Zuge der Analyse (siehe 4) werden technologische Aspekte in der Entwicklung einer Sprachanwendung untersucht, was eine Basis für die Entscheidungen im Umsetzungsprozess bilden. Zusätzlich wird innerhalb der Untersuchung des Anwendungsfalls (siehe 5) die gewünschten use cases bzw. Anforderungen erarbeitet, definiert, sowie ein exemplarisches Kontextszenario vorgestellt.

Aufbauend darauf wird das Konzept der entwickelten Anwendung erarbeitet (siehe 6). Dies beinhaltet sowohl die klarere Abgrenzung der implementierten Informationen, als auch das Konzept von Dialog- und Programmabläufen für die erarbeiteten use cases. In der Umsetzung (siehe 7) wird dieses grundlegende Konzept mit den gewählten Technologien implementiert und die grundlegende Funktionsweise des Systems erläutert. Im Zuge der Zusammenfassung (siehe 8) wird die vorliegende Arbeit resümiert und herausgearbeitet, welche daran anknüpfende Aspekte mit zukünftigen Arbeiten adressiert werden können.

2 Grundlagen

Folgendes Kapitel soll eine Grundlage an Kenntnissen schaffen, welche für das Verständnis des restlichen Teils dieser Arbeit, und dem in diesem Kontext entwickelten skill, notwendig sind. Zuerst werden hierbei relevante Begriffe in der Entwicklung von Sprachanwendungen erklärt. Danach wird die *Speech Synthesis Markup Language* (SSML) mit einigen beispielhaften Features vorgestellt. Darüber hinaus wird knapp die Funktion der SAP Cloud for Customer erklärt, und einige der für diese Arbeit relevanten Datenobjekte erläutert.

2.1 Begriffe in der Entwicklung von Sprachanwendungen

Die Kommunikation via natürlicher Sprache mit einem System unterscheidet sich in einigen Aspekten deutlich von den zur Zeit omnipräsenten graphischen Oberflächen. Demnach haben sich für die Entwicklung von Anwendungen mit dieser Art der Eingabe einige neue Konzepte und Begriffe entwickelt und durchgesetzt. Auch wenn die genauen Bezeichnungen dieser Konzepte sich teilweise zwischen den hier primär betrachteten Assistenten (Amazon Alexa und Google Assistant/Dialogflow) unterscheiden, so sind die Konzepte der grundsätzlichen Funktionen ähnlich aufgebaut. Im Folgenden sollen diese Konzepte kurz erläutert werden und eine für den Rest der Arbeit einheitliche Begriffsbasis erzeugt werden.

2.1.1 Skill

Ein *skill* bündelt eine Menge an Funktionalitäten in eine Anwendung für einen Assistenten [9]. Dies ist vergleichbar mit Apps im Kontext mobiler Betriebssysteme wie beispielsweise Android oder iOS. Ähnlich wie dort, gibt es auch hier oft die Möglichkeit, neue skills zu entwickeln und diese den Nutzern anzubieten. Dies passiert meistens über einen zentralen Shop wie den Amazon skill Shop [6] oder das Google Assistant directory [32].

Ein skill beinhaltet Einstellungen und Meta-Informationen über die Anwendung, wie beispielsweise den *invocation name*, mit welchem den skill identifiziert und aufgerufen werden kann. Zusätzlich dazu beinhaltet er *intents* (siehe 2.1.2) und den zur Erfüllung dessen notwendigen Code, bzw. eine Referenz auf den cloud basierten Service, welcher diesen Code enthält [9]. Bei Google Assistant wird das *natural language understanding* ausgelagert, indem ein Dialogflow agent verknüpft werden kann, welcher dann intents und eine Referenz auf den Code enthält [22]. Um weiterhin eine einheitliche Begriffs-Basis zu wahren, wird auch dies in dem Begriff skill zusammengefasst.

2.1.2 Intent

Ein *intent* ist eine Repräsentation des Wunsches eines Nutzers und beinhaltet die Aktionen, die dafür umzusetzen sind. Ein Beispiel dafür wäre die Ausgabe des Wetters. Dieser intent kann durch den Nutzer aufgerufen werden, wodurch die Funktionalität der Abfrage und Ausgabe des Wetters angestoßen wird. [7]

2.1.3 Phrase

Phrases (bei Alexa *utterances*) sind Formulierungen, welche von dem Nutzer geäußert werden, um bei dem skill einen intent auszulösen, oder um auf eine Frage des skills zu antworten [11]. Ein Beispiel für den oben etablierten intent wäre “Wie ist das Wetter?”. Da es in natürlichen Sprachen viele verschiedene Arten gibt etwas auszudrücken, ist es empfehlenswert eine

möglichst hohe Anzahl an phrases für einem intent anzugeben. Eine alternative Formulierung für den Beispiel-intent wäre beispielsweise “Gib mir einen Wetterbericht”.

Die Menge der phrases, welche einem intent zuzuordnen sind, beschränkt sich jedoch nicht nur auf die in der Entwicklung definierte Menge, sondern wird mit der Hilfe von machine learning erweitert. So kann ein intent mit der einzigen phrase “I want pizza” auch automatisch durch “Get a pizza” oder “Order pizza” aufgerufen werden. Das bedeutet, dass im Entwicklungsprozess nicht alle möglichen phrases, welche erkannt werden sollen, angegeben werden müssen. Da der machine learning Prozess diese definierte Menge als *training phrases* verwendet, müssen jedoch ausreichend viele vorhanden sein, damit die künstliche Intelligenz das Gesagte dem richtigen intent zuweisen kann. Laut Dialogflow sind hier 20 oder mehr angemessen, um eine hinreichend zuverlässige intent-Zuweisung gewährleisten zu können. [27]

2.1.4 Input

Ein *input*, bei Dialogflow *entity* und bei Alexa *slot* genannt, ist ein Argument für einen intent, welches zusätzliche variable Information zu dem Anliegen des Nutzers liefert [23, 10]. Sie sind zu vergleichen mit Parametern einer Funktion von herkömmlichen Programmiersprachen. Ein Beispiel für den Wetter-Abfrage-intent wäre, wenn der Nutzer den Tag der Wetterprognose in der phrase angeben könnte. Da es nicht möglich ist, für jeden Tag eine phrase oder intent zu erstellen, kann hierfür ein input verwendet werden. Einige Beispiel-phrases dafür könnten also wie folgt aussehen:

- “Wie ist das Wetter {tag}”
- “Wie ist das Wetter am {tag}”
- “Wie war das Wetter {tag}”
- “Wie war das Wetter am {tag}”
- ...

Der input ist hier *tag* und kann mit allen möglichen Tagen gefüllt werden. Die möglichen Werte, welche ein input annehmen kann, werden durch *input types* definiert. Durch fest definierte input types kann sichergestellt

werden, dass der übergebenen Wert im intent behandelt werden kann. Im Fall des Wetterprognosen-intents handelt es sich bei dem *tag* input type um einen häufig von allen möglichen skills verwendeten input types. Für diese Art der inputs stellen Plattformen wie Dialogflow oder Alexa vordefinierte input types zu Verfügung, welche dem erstellten input zugewiesen werden können.

In dem Falle des Datums wären das beispielsweise *@sys.date* für Dialogflow bzw. *AMAZON.DATE* bei Alexa. Diese haben den Vorteil, dass sie viele verschiedene Ausdrucksweisen zulassen, dem intent aber in einem einheitlichen Format übergeben werden [26, 15]. Folgende Aussagen können beispielsweise mit Hilfe der oben genannten input types behandelt werden:

- "Wie ist das Wetter Freitag?"
- "Wie ist das Wetter am ersten Fünften?"
- "Wie war das Wetter gestern?"

Alternativ dazu, können auch neue input types mit eigenen Werten definiert werden. Wenn der Wetter-skill also zulassen will, dass ein Nutzer fragen kann, ob er einen Regenschirm, Sonnencreme oder Wintermantel benötigt, kann ein input type *benoetigtesItem* mit diesen Werten erstellt werden. Dann können inputs in phrases diesem individuellen input type zugewiesen werden.

Wenn ein input für die Funktionalität des intents zwingende gesetzt sein muss, so bieten Plattformen wie Alexa oder Dialogflow hierfür die Möglichkeit, diese als *required* zu setzen. Dadurch können die Systeme automatisch nach nicht bereits gelieferten input-Wert fragen, bevor die Anfrage an den intent weitergeleitet wird. [8, 25]

2.1.5 Speech und Reprompt

Der als Antwort ausgegebene Text wird im Folgenden *speech* genannt. Wenn ein Nutzer nach der Ausgabe der speech eine bestimmte Zeit nichts sagt, so kann ein *reprompt* ausgegeben werden, welche beispielsweise etwaige

Unklarheiten erklärt [14]. Eine Beispielhafte Interaktion könnte folgendermaßen aussehen:

- Nutzer: "Wetter"
- Assistant: "Willst du einen Überblick über das Wetter heute in Dresden?"
- ...
- Assistant: "Sag ja oder nein!"
- Nutzer: "Ja"
- Assistant: "Es wird bis zu 30 Grad in Dresden."

2.1.6 Session

Um eine mehrzügige und kontextsensitive Konversation und das Speichern von Daten bzw. Status des Dialogs zu verwalten, benötigt es eine *session*. Die session wird beim Starten des skills durch den invocation name geöffnet. Solange diese geöffnet ist, kann der Nutzer mit dem skill kommunizieren ohne erneut den invocation name aussprechen zu müssen.

Innerhalb des skill codes kann dann nach jeder Anfrage entschieden werden, ob die session geschlossen werden soll, was den skill beendet. Die session kann auch beendet werden, indem der Nutzer eine bestimmte Zeit keine Antwort liefert. Zusätzlich zur Aufrechterhaltung der Kommunikation mit dem skill können in der session weitere Daten gespeichert werden, welche für spätere Anfragen an den skill relevant sind. Diese Daten werden, solange die session nicht beendet wurde, bei jeder folgenden Anfrage an den skill geschickt, wo sie ausgelesen und verwendet werden können. In dem Wetterskill könnte beispielsweise gespeichert werden, welche Städte seit dem starten des skills von dem Nutzer abgefragt wurden, oder in welchem Zustand (bzw. *state*) er sich in dem Dialog befindet.

Es ist darauf zu achten, dass bei jedem Startvorgang eine neue session erzeugt wird, und bei jedem Beenden diese mit den Daten zerstört wird. Für eine persistente Speicherung der Daten ist session data also nicht geeignet. Um Daten über eine längere Zeit zu speichern wird eine andere Möglichkeit der persistenten Datenspeicherung benötigt, in welchen dann

für die einzelnen Nutzer relevante Daten über sessions hinaus gespeichert werden können. [13]

2.2 Speech Synthesis Markup Language

Speech Synthesis Markup Language (kurz *SSML*) ist eine auf XML basierende Markup Sprache. Sie bietet ein Format, mit welchem Sprachausgabe bzw. Sprachsynthese modelliert werden kann. Beispiele für Modifikationen der Synthese wären etwa Lautstärke, Pausen, Tonhöhe, Betonung , Sprache, Geschwindigkeit und so weiter [65]. Plattformen wie Google Assistant und Amazon Alexa unterstützen viele grundlegende SSML Elemente, welche zur Sprachmodifikation genutzt werden können [16, 36]. Folgendes zeigt ein für Google Assistant und Alexa gültiges SSML:

```
<speak>
    Max Muster <break time="500ms"/>
        <say-as interpret-as="ordinal">1</say-as>
        name is
        <emphasis level="moderate">Max</emphasis>.

        Telephone number is
        <say-as interpret-as="cardinal">012345</say-as>.

        Height is
        <say-as interpret-as="unit">6 foot</say-as>.

        Birthday is
        <say-as interpret-as="date" format="yyyymmdd">
            1960-09-10
        </say-as>.

        <prosody rate="slow" pitch="50%">
            This is slower and lower.
        </prosody>
</speak>
```

Obiges Beispiel zeigt, wie SSML Elemente Kontext und weitere Informationen zusätzlich zu dem Text bereitstellen, welche die Ausgabe beeinflussen. Das *break* Element in Zeile 2 bewirkt, dass eine Pause, deren Länge mit dem *time* Attribut angegeben werden kann, an der Stelle eingefügt wird. Mit dem *emphasis* Element (Zeile 5) kann ein wichtiger Abschnitt hervorgehoben werden. Hierbei gibt das *level* Attribut mit Werte wie *strong*, *moderate* oder *reduced* die Art der Betonung an.

Das *say-as*-Element in Zeile 3 mit dem Attribut *interpret-as* als *ordinal*, führt dazu, dass *1* nicht "*one*" ausgesprochen wird, sondern "*first*". Zeile 8 zeigt, wie eine mehrstellige Zahl Ziffer für Ziffer ausgeben werden kann, was beispielsweise für Postleitzahlen oder Telefonnummern hilfreich ist. Auch ein korrekt formuliertes Datum kann aus einem einfachen Text ausgegeben werden, wie Zeile 14 darstellt. In Zeilen 18 bis 20 wird die Prosodie des auszugebenden Texts moduliert, indem die Sprache langsamer und tiefer als der Standard ausgegeben wird.

Amazon bietet für Alexa zusätzlich dazu eigene Elemente an, wie den sogenannten *effect*, mit welchem beispielsweise ein Text geflüstert ausgegeben werden kann. Hierzu folgendes Beispiel: [16]

```
1 <speak>
2     I want to tell you a secret.
3     <amazon:effect name="whispered">
4         I am not a real human.
5     </amazon:effect>.
6     Can you believe it?
7 </speak>
```

2.3 SAP Cloud for Customer

Bei der SAP Cloud for Customer handelt es sich um eine cloud basierte Plattform, welche vor allem den Vertrieb einer Firma unterstützten soll. Neben der Verwaltung von Kunden, Aufträgen usw. bietet die Cloud for Customer einige weitere Features, wie Schnittstellen und Synchronisation mit Diensten wie Gmail, Outlook oder anderen SAP Systemen. Des weiteren

gibt es dem Nutzer die Möglichkeit, von vielen Geräten, etwa mit Apps, auf die Cloud for Customer zuzugreifen. [59]

Im Folgenden sollen kurz einige, für die weitere Arbeit relevante Features der Cloud for Customer vorgestellt werden. Zusammengefasst sollen diese Features dafür sorgen, dem Vertriebsteam dabei zu helfen mehr Verträge in kürzerer Zeit abschließen zu können.

2.3.1 Lead

In der Cloud for Customer können sogenannte *leads* angelegt werden, welche für das Interesse eines (potentiellen) Kunden an einem Produkt oder Service steht. Ein solcher lead kann beispielsweise auf einer Messe entstehen, auf welcher ein Mitarbeiter einer anderen Firma Interesse an einem Produkt äußert und seine Visitenkarte übergibt. Anhand dieser Informationen kann dann ein lead erstellt werden. Dieser kann zudem auch zu einem späteren Zeitpunkt mit weiteren Daten gefüttert werden oder qualifiziert werden, je nachdem wie vielversprechend das Interesse eingeschätzt wird. [56]

2.3.2 Opportunity

Ein lead kann sich zu einer *opportunity* entwickeln, womit der sogenannte sales cycle gestartet wird. Hierbei können sämtliche relevante Informationen dazu gebündelt werden, wie etwa die entsprechenden Produkte, Partner, Konkurrenten, Umfragen und so weiter. Diese Informationen können für die Evaluation der Wichtigkeit und Wahrscheinlichkeit des Zustandekommens des Auftrags Aufschluss geben. [57]

2.3.3 Sales Quote

Aus der Opportunity entwickelt sich im Optimalfall wiederum ein konkretes Angebot, die sogenannte *sales quote*. Auch hier bündelt die sales quote alle für diesen Status des sale cycles relevanten Informationen wie den

zugehörigen (potentiellen) Kunden, einen Preis, Rabatte und so weiter. Dieses Angebot kann so dem Kunden vorgelegt werden. [58]

2.3.4 Sales Order

Wenn der Kunde eine sales quote akzeptiert, kann der Status auf *completed* gesetzt werden. Damit ist der für diese Arbeit relevante Teil des sales cycle Prozesses abgeschlossen, und die sales quote wird zu einer *sales order*, welche dann beispielsweise an ein weiteres System zum Verwalten von Bestellungen exportiert wird. [58]

2.3.5 Account

Wie bereits erwähnt können opportunities, sales quotes usw. mit einem sogenannten *account* vermerkt werden, welcher einen (potentiellen) Kunden, Partner usw. im System darstellt. Dabei handelt es sich um ein anderes Business, während Einzelpersonen als *individual customers* dargestellt werden. Der Account bündelt alle relevanten Informationen der entsprechenden Firma wie Adresse, Hauptansprechpartner und so weiter. Darüber hinaus kann via Verknüpfung zusätzlich auch auf die account-spezifischen leads, opportunities, sales quote und sales orders zugegriffen werden. [54]

2.3.6 Activity

Damit der Vertriebsmitarbeiter mit dem Kunden den sales cycle durchlaufen kann, benötigt es weitere Aktionen, welche in der SAP Cloud for Customer verwaltet werden können. Dazu gehören *phone calls*, *tasks*, *e-mails*, *appointments* und *visits*. Zusammen genommen werden diese als *activities* bezeichnet. [55]

Wenn also beispielsweise ein Kunde in der Firma anruft, kann der bearbeitende Mitarbeiter diesen Anruf mit einer Notiz, was das Anliegen war, in der Cloud for Customer ablegen und dies mit dem Account verknüpfen. Er

kann dann zusätzlich für einen Vertriebsmitarbeiter ein appointment erstellen. Dieses appointment enthält wiederum alle für das Meeting relevanten Informationen. Der Vertriebsmitarbeiter kann dann von dem appointment aus auf den account und davon auf den phone call zugreifen. Alternativ könnte der phone call auch direkt als *related item* mit dem appointment verknüpft werden.

2.3.7 Architektur

Die SAP Cloud for Customer stellt für verschiedene Business Prozesse und Aktionen Datenobjekte bereit, welche alle dafür relevanten Informationen bereithalten. Diese haben darüber hinaus einen hohen Verknüpfungsgrad, wodurch wichtige Informationen, welche sich nicht direkt in dem Datenobjekt selbst befinden, durch Verlinkung erreichbar sind.

3 Verwandte Arbeiten

Folgendes Kapitel betrachtet verwandte Arbeiten sowohl aus der wissenschaftlichen und konzeptionellen, als auch aus wirtschaftlichem und praktischem Kontext. Diese bilden zusätzlich zu der Grundlage und der Analyse im darauf folgenden Kapitel die Basis des in dieser Arbeit entstandenen skills. Hierbei sollen zuerst die Herausforderungen der Kommunikation mit einer Sprachanwendung herausgearbeitet werden. Darauf aufbauend sollen aus diesen Herausforderungen entstandene Designheuristiken vorgestellt werden, welche bei der Entwicklung zu beachten sind.

3.1 Herausforderungen bei der Entwicklung von Sprachanwendungen

Eine im Vergleich zu herkömmlichen graphischen Benutzeroberflächen sehr unterschiedliche Art der Mensch-Computer-Interaktion via Sprachassistenten, wirft dementsprechend im Design-Prozess spezifische zu überwindende Problemstellungen auf. Demnach entstehen auch für diese Art der Kommunikation per Sprache eigene Designheuristiken, welche bei der Entwicklung entsprechender skills zu beachten sind. Shneiderman [61] erarbeitete diesbezüglich spezielle Herausforderungen, welche bei der Sprachinteraktionen auftreten. Diese werden im Folgenden erläutert.

3.1.1 Präsentation von Möglichkeiten

Eine gut gestaltete graphische Oberfläche kann dem Nutzer die ihm offenen Optionen und die Struktur der Anwendung auf einen Blick veranschauli-

chen. Diese angebotenen Funktionen und deren Befehle dem Nutzer eines skills beizubringen ist eine der größten Herausforderungen in deren Designprozess.

Während bei skills mit geringem Umfang und vielen einmaligen oder neuen Benutzern es empfehlenswert ist, alle Optionen per speech direkt auszugeben und den Dialog stark durch das System leiten zu lassen, ist dies bei komplexeren, freieren und alltäglich genutzten Anwendungen nicht praktikabel. Hier ist es wichtig, sowohl neuen Nutzern eine Anleitung zu geben, ohne damit jedoch den erfahreneren Nutzern, welche über Funktionsumfang und Interaktion Bescheid wissen, einzuschränken.

Einem erfahrenen Nutzer soll es also möglich sein, durch alternative Wege der Interaktion, schneller die gewünschte Funktionalität auszulösen oder Information abzufragen. Zudem erkennt der skill im Optimalfall einen neuen Nutzer, und passt die Interaktion dementsprechend an. [61]

3.1.2 Reaktion auf Fehler

Des Weiteren muss davon ausgegangen werden, dass der Nutzer eine fehlerhafte Eingabe macht, oder das System den Nutzer falsch versteht. Demnach sollte vor allem bei kritischen Informationen dem Nutzer die Möglichkeit gegeben werden, den Fehler zu entdecken und diesen gegebenenfalls zu korrigieren. [61]

3.1.3 Intentzuweisung

Eine weitere Fehlerquelle ist die falsche Zuordnung des Gesagten zu der auszulösenden Funktionalität. Um dieses Problem zu minimieren, können für unterschiedliche Funktionalitäten (bzw. intents) nur eine oder wenige eindeutige phrases zugeordnet werden. Dies würde jedoch dazu führen, dass einige nicht zugeordnete phrases gar keiner Funktionalität zugeordnet werden können. Die Herausforderung besteht demzufolge darin, viele Variationen anzubieten, diese jedoch so eindeutig zu formulieren, dass die Intentzuweisung zuverlässig funktioniert. [61]

3.1.4 Dialoganpassung und Feedback

Die kontextuell richtige Reaktion auf eine Eingabe oder einen Zustand ist eine weitere Herausforderung des Designprozesses eines skills. Durch die optimalerweise sehr freie Eingabe des Nutzers können im System dementsprechend viele verschiedene Zustände entstehen, welche eventuell den Dialog verändern. [61] Je nach Anzahl von optionalen oder verpflichtenden inputs wird die Anpassung immer komplexer und das zu liefernde kontextuelle Feedback relevanter. Hierbei ist es das Ziel dem Nutzer möglichst viel Freiheit zu bieten, ihm jedoch mit Hilfe von informativen Feedback Auskunft über den Zustand des skills zu liefern.

3.1.5 Sprachausgabe

Auch die Sprachausgabe stellt Herausforderungen, welche im Design zu beachten sind. Zum einen ist im Vergleich zu einer graphischen Darstellung die Ausgabe per Sprache von vor allem langen Texten deutlich zeitaufwändiger. Des weiteren ist sie zeitlich vergänglich: Während ein Nutzer bei graphisch dargestellten Texten bei Unklarheiten oder irrelevanten Informationen zurück oder vorwärts springen kann, ist dies hier nicht möglich. [61]

Des weiteren wirft es Fragen bezüglich der Privatsphäre auf, vor allem bezüglich der Ausgabe von vertraulichen oder privaten Informationen. Zudem ist das Suchen nach Informationen oder Stichworten, welche bei graphischen Oberflächen meist ganz einfach per Suchfunktion möglich ist, hier nicht so komfortabel umsetzbar. [61]

3.2 Designheuristiken bei der Entwicklung von Sprachanwendungen

Als Resultat der Herausforderungen (siehe 3.1) im Umgang mit einer Sprachanwendung können Designheuristiken abgeleitet werden, welche die Entwicklung eines intuitiv bedienbaren skills unterstützen. Diese Richtlinien

gibt es beispielsweise direkt von den Entwicklern der Plattformen, wie den Alexa Design Guide [1], welcher praktische Hinweise für das Design eines skills definiert.

Der Alexa Design Guide setzt als Grundsatz, dass der Design-Prozess komplett mit einem Voice-first und nutzerzentrierten Gedanken durchlaufen werden sollte, anstatt lediglich die Struktur und Aufbau einer möglicherweise bereits bestehenden graphischen Oberfläche zu kopieren. Es soll darauf geachtet werden, auf wie viele verschiedene Weisen ein Nutzer eine bestimmte Forderung ausdrücken kann. Die in Anbetracht dessen entstandenen Regeln aufgeteilt in die vier Ziele der *Adaptabilität*, *Personalisierbarkeit*, *Verfügbarkeit* und *Nachvollziehbarkeit* werden im Folgenden genauer untersucht.

3.2.1 Adaptabilität

Durch die Herausforderungen des Überblicks über die verfügbaren Optionen (siehe 3.1.1) und der möglicherweise fehlerhaften Intentzuweisung (siehe 3.1.3) ist es von elementarer Wichtigkeit, dass sich der skill dem Nutzer anpasst, wodurch eine intuitive Benutzung ermöglicht wird.

Erreicht wird dies beispielsweise durch das Zuweisen vieler unterschiedlicher Beispiel-phrases zu intents, welche möglichst viele Arten abdecken, wie ein Nutzer einen Wunsch ausdrücken kann. Es soll auch beachtet werden, dass eventuell in einer phrase zusätzlich gelieferte Informationen direkt aufgenommen werden und der Dialog entsprechend angepasst wird (siehe 3.1.4). Im Gegenzug soll der skill bei fehlenden Informationen direkt nachfragen, um alle notwendigen Informationen für die Ausführung des intents zu erlangen, wie etwa durch automatisches input filling (siehe 2.1.4).

Wenn ein Nutzer beispielsweise sagt “Stelle den Wecker auf morgen 18 Uhr” hat das System alle benötigten Werte. Es soll dem Nutzer jedoch auch möglich sein “Stelle den Wecker” zu sagen, worauf die Anwendung mit “Auf wie viel Uhr soll ich den Wecker stellen?” antworten soll.

Zusätzlich sollte es dem Nutzer möglich sein, seine Eingaben zu korrigieren (siehe 3.1.2), falls diese falsch verstanden wurden, oder der Benutzer

sich umentscheiden sollte. Dies passiert meistens beispielsweise durch erneute Wiedergabe der Eingabe und einer Möglichkeit dies zu ändern, wie folgendes Beispiel zeigt:

- Nutzer: “Erinnere mich morgen um 8 Uhr an Wäsche waschen”
- Assistant: “Okay, ich erinnere dich morgen um 8 Uhr morgens an Wäsche waschen”
- Nutzer: “Nein, morgen 8 Uhr abends”
- Assistant: “Okay, werde dich um 8 Uhr abends erinnern”

Des Weiteren soll auf Fehler zum Beispiel bei einer nicht möglichen Intentzuweisung hilfreich reagiert werden. Anstatt lediglich auszugeben, dass man den Nutzer nicht richtig verstanden habe, könnte beispielsweise eine Hilfe ausgegeben werden, welche intents mit welchen Werten möglich sind.

Als weitere Hilfestellung sollten reprompts ausgegeben werden, wenn der skill den Nutzer nach einer Eingabe fragt, dieser jedoch nicht antwortet. Dieser soll optimalerweise genauer erklären, welche Möglichkeiten der Nutzer hat.

Zudem soll dem Nutzer, wenn benötigt, auf Anfrage Hilfe angeboten werden. Sie soll dem Nutzer beispielsweise erklären, wie ein intent ausgelöst werden kann. Wichtig ist hierbei, dass diese Hilfe kontextuell sein sollte, sich also je nach Zustand des skills anpassen sollte. Bei einer Bestellungsabwicklung eines Online-Shops kann beispielsweise gesagt werden, dass man die Bestellung bestätigen kann, während bei einer Produktsuche die möglichen Filter aufgelistet werden können. [2]

3.2.2 Personalisierbarkeit

Zur Personalisierung eines skills gehört beispielsweise das Speichern bestimmter Daten über den Nutzer und die Anpassung des skills anhand dieser Daten. Dazu gehört unter anderem das Erkennen von neuen bzw. wiederkehrenden Nutzern. Anhand dessen ist es zum Beispiel möglich, eine angepasste Wilkommensnachricht ausgegeben werden. Für neue Nutzer könnte hier eine Auflistung der Features ausgegeben werden, während dem erfahreneren Nutzer eine schneller Interaktion durch eine kürzere

Nachricht ermöglicht wird (siehe 3.1.1). Bei einem Wetter-skill könnte dies folgendermaßen aussehen:

- Nutzer: “Starte Wetter skill”
- Assistant: “Willkommen zu der Wetter skill. Du kannst nach der Temperatur, der Luftfeuchtigkeit oder dem Sonnenaufgang für bestimmte Tage fragen.”
- ...
- Nutzer: “Starte Wetter skill”
- Assistant: “Willkommen zurück! Was kann ich für dich tun?”

Zusätzlich dazu kann ein skill mit weiteren gespeicherten Informationen personalisiert werden, wie beispielsweise durch das Einbinden des Namens des Nutzers in die Sprachausgabe des skills. Dies dient dazu, den Nutzer “vertrauter” mit dem skill bzw. dem Assistenten zu machen.

Es ist auch hilfreich verschiedene Formulierungen der Sprachausgabe eines intents zu erstellen, und diese je nach Benutzer auszuwählen. Dabei kann beispielsweise je nach Häufigkeit der Nutzung immer kürzere Ausgaben getätigt werden, da der Nutzer bereits vertraut mit dem skill und dessen Funktionalitäten ist.

Um das Nutzererlebnis weiterhin zu steigern kann der skill zusätzlich nach Beenden und erneutem Starten fragen, ob der Nutzer dort weiter machen möchte, wo er bei der vorherigen Sitzung aufgehört hat. Dies kann dem Nutzer einige Schritte ersparen, um wieder zu dem gleichen Stand zu kommen. Zusätzlich dazu können skills die aktuelle Position des Nutzers abfragen und nutzen, um den skill weiter zu personalisieren. [4]

3.2.3 Verfügbarkeit

Ein weiterer Grundsatz des Alexa Design Guides ist die Verfügbarkeit. Umgesetzt steht dies für eine sogenannte horizontale Struktur des skills, was bedeutet, dass möglichst viele Funktionen direkt verwendbar sind, anstatt diese an zu viele Bedingungen zu knüpfen, wie etwa eine komplexe menüartige Struktur, wie sie in graphischen Oberflächen zu finden ist. Der skill sollte viele verschiedene Wege anbieten, eine Aktion im System

auslösen zu können. Dies nimmt sich der beschriebenen Herausforderung an, dass der Nutzer nicht genau weiß, wann er was tun kann (siehe 3.1.1), indem so viele Möglichkeiten offen gehalten werden, wie möglich.

Zum Grundsatz der Verfügbarkeit gehört zusätzlich die effiziente Formulierung von reprompts, welche (bei Alexa) acht Sekunden nach der ersten Sprachausgabe bei keiner Antwort ausgegeben wird. Wenn darauf weitere acht Sekunden nicht geantwortet wird, wird die session geschlossen. Es wird davon ausgegangen, dass der Nutzer die Frage nicht richtig verstanden hat, weswegen man die Frage umformulieren bzw. genauer erläutern oder die Interaktionsmöglichkeiten genauer aufzeigen sollte. Folgendes Beispiel zeigt, wie man das Gespräch mit reprompts besser leiten kann, als durch ein erneutes Wiedergeben der Ausgangsfrage.

- Assistant: "Willkommen zum Wetter skill, wie kann ich dir helfen?"
- 8 Sekunden Pause
- Assistant: "Du kannst nach der Temperatur, der Luftfeuchtigkeit oder dem Sonnenaufgang für bestimmte Tage fragen."

Es ist auch elementar, dass, vor allem bei einer Suche, der skill alle Informationen, welche der Nutzer liefert, aufnimmt, und die fehlenden notwendigen Informationen kurz und eindeutig abgefragt werden. Wenn der Nutzer bei einem Bestellvorgang beispielsweise zwischen Express- oder Overnight-Versand wählen sollte, könnte ein Nutzer auf die Frage "Willst du Express Versand oder Overnight Versand?" Mit "Ja" antworten. Eine eindeutigere Formulierung wäre beispielsweise "Wir bieten Express oder Overnight Versand an. Welche Art willst du?", wodurch die Antworten "Ja" oder "Nein" logisch ausgeschlossen werden.

Die Verfügbarkeit des Systems verbessert sich auch durch das effiziente Design der Sprachausgabe (siehe 3.1.5). Eine kurze und prägnante Antwort des Systems überfordert den Nutzer nicht nur nicht mit unnötigen Informationen, sondern ist zusätzlich auch schneller für erneute Eingaben verfügbar. Der skill sollte beispielsweise versuchen herauszufinden, was der Nutzer möchte, falls das Gesagte nicht eindeutig ist. Ein Beispiel ist die Frage nach dem Wetter in "Neustadt", da es verschiedene Städte mit diesem Namen gibt. Anstatt hier alle Städte aufzulisten, kann per Abfrage des

Standpunkts des Nutzers eine Vermutung gemacht werden, wie folgendes Beispiel zeigt:

- Nutzer: “Wie ist das Wetter in Neustadt?”
- Assistant: “Ich habe acht Städte mit diesem Namen gefunden. Meinst du Neustadt im Landkreis Marburg?”

Außerdem ist darauf zu achten, dass bei umfangreichen Listen die Anzahl der items angesagt werden sollte, wie obiges Beispiel zeigt. Um die Ausgabe zu kürzen, können zudem auch nur die ersten Elemente gesprochen werden, und der Nutzer gefragt werden, ob dieser mehr hören möchte. Zudem ist darauf zu achten, dass lediglich relevante Informationen der Listenelemente ausgegeben werden, wie beispielsweise der Titel. Je nachdem wie lange die Ausgabe eines items benötigt, sollen höchstens fünf Elemente auf einmal ausgegeben werden. Diese Anzahl sollte sich beispielsweise bei längeren Titeln verringern.

Damit der Nutzer versteht, dass es sich bei der Ausgabe um eine Liste handelt, soll zu Beginn eine Einleitung gegeben werden, welche beispielsweise mit der Anzahl kombiniert werden kann, wie “Hier ist die Wettervorhersage der nächsten drei Tage”. Zwischen den items sollte dann eine Pause eingelegt werden, damit der Nutzer diese voneinander abgrenzen kann. Dies ist mit SSML (siehe 2.2) einfach möglich, womit auch die Länge der Pause je nach Komplexität verändert werden kann. Als Richtwert der Dauer gelten 350ms bis 400ms.

Zuletzt soll auf allgemeine Fragen wie “Kann ich sonst noch was tun?” nach dem Ausführen einer Aufgabe des skills vermieden werden, da solche offenen Fragen eine hohe kognitive Belastung für den Nutzer darstellen. Es kann außerdem frustrierend für den Nutzer sein, die Session immer manuell beenden zu müssen, wenn er die gewünschte Aktion durchgeführt hat. Deshalb sollte bei einer abgeschlossenen Aufgabe der skill von selbst die Session beenden. [3]

3.2.4 Nachvollziehbarkeit

Um es dem Nutzer zu ermöglichen leicht zu verstehen, welche Informationen das System möchte und was gerade passiert, ist es wichtig sich deutlich auszudrücken und *mit* statt *zu* dem Nutzer zu sprechen.

Um das erreichen zu können, sollte der Grundsatz *Schreib es, wie du es sagen würdest!* umgesetzt werden. Geschriebene Sprache unterscheidet sich in Punkten wie Förmlichkeit oder Informationsdichte von Gesagtem. Demnach soll es vermeidet werden geschriebene Texte in einem skill als Ausgabe wiederzuverwenden. Im Designprozess sollte sich eher gefragt werden, wie man etwas gesprochen formulieren würde. Als Hilfestellung kann man sich das System als eine Person vorstellen. Falls das Gesagte für eine Person unnatürlich klingt, muss der Text verändert werden.

Dazu gehört beispielsweise auch, dass nicht zu viel Jargon-Sprache verwendet wird, und vor allem nicht von dem Nutzer erwartet wird. Stattdessen soll das Gesagte eher kolloquialen Charakter haben, was durch eine kurze prägnante Ausdrucksweise, Schmelzwörter oder einer Kadenz, welche sich an gesprochener Sprache orientiert, umgesetzt werden kann.

Da zu lange Antworten des Systems, auch durch die zeitliche Vergänglichkeit des Gesagten (siehe 3.1.5), für den Nutzer schwerer zu verstehen und merken sind, ist das kurz halten in der Sprachausgabe von elementarer Wichtigkeit. Als Test hierfür, kann der sogenannte *one-breath test* durchgeführt werden.

Bei dem *one-breath-test* soll der komplette Text vorgelesen werden können, ohne Luft holen zu müssen. Wenn dies nicht gelingt, sollte möglichst versucht werden, den Text zu kürzen. Ausnahmen gelten hierfür etwa für Listen mit aufeinander folgenden Punkten. Hierbei sollten die einzelnen Punkte den *one-breath-test* bestehen.

Die Ausgabe des Assistenten sollte zudem kontextuell relevant sein. Das bedeutet, dass beispielsweise die Vorschläge an Aktionen, die dem Nutzer gegeben werden, in einer dem Kontext entsprechend sinnvollen Reihenfolge ausgegeben werden. Dadurch wird dem Nutzer die Arbeit abgenommen, selbst herausfinden zu müssen, was zum aktuellen Zeitpunkt die für ihn relevanteste Aktion oder Information ist.

Während Varianz der Sprachausgabe abhängig vom Nutzer zu der Personalisierung (siehe 3.2.2) des skills beitragen kann, kann diese auch unabhängig davon eingesetzt werden, um durch Abwechslung im Gespräch, dieses natürlicher wirken zu lassen. Dabei ist besonders ein Schwerpunkt auf die Texte zu legen, welche voraussichtlich besonders häufig ausgegeben werden. Dazu gehören beispielsweise die Willkommens- bzw. End-Nachrichten.

Dies gilt auch für sogenannte *discourse makers*, also Begriffe, welche einen Satz strukturieren, wie "Okay" oder "Alles klar". Die Verwendung dieser Begriffe wird explizit empfohlen, um dem Nutzer das Einordnen des Gesagten zu erleichtern. Begriffe wie "Okay" drücken ganz einfach aus, dass das System den Nutzer verstanden hat, und / oder eine Aktion erfolgreich durchgeführt hat, wie folgendes Beispiel zeigt:

- Nutzer: "Setze Waschmittel auf meine Einkaufsliste!"
- Assistant: "Alles klar, habe Waschmittel auf deine Einkaufsliste gesetzt"

Im Zusammenspiel sollen diese Regeln letztendlich dafür sorgen, dass sich die Konversation für den Nutzer natürlich anfühlt, als würde er sich mit einem anderen Menschen unterhalten. [5]

4 Analyse

Dieses Kapitel untersucht verschiedene konkrete Technologien und deren Vor- bzw. Nachteile. Ergebnisse dieser Analysen bieten zusammen mit der Analyse des Anwendungsfalls (siehe 5) und dem daraus entstandenen Konzept (siehe 6) die Grundlage für die Implementierung (siehe 7).

4.1 Voice Assistants - ein Vergleich

Im Folgenden sollen exemplarisch einige ausgewählte voice assistant Plattformen hinsichtlich verschiedener Gesichtspunkte untersucht werden. Ausgewählt wurden hier neben den bereits erwähnten und weit verbreiteten Amazon Alexa und Google Assistant auch Mycroft AI und Snips.

Generell kann die steigende Akzeptanz und Nachfrage nach voice assistant Systemen anhand der rapide ansteigenden Verkaufszahlen von sogenannten *smart speakern*, Lautsprechern mit integriertem voice assistant, erkannt werden. Laut Strategy Analytics [18] wurden weltweit im Jahr 2018 86,2 Millionen smart speaker verkauft. Alleine im vierten Quartal 2018 wurden mit 38.5 Millionen mehr smart speaker verkauft als im gesamten Jahr zuvor.

In diesem vierten Quartal 2018 betrug der Anteil der verkauften Amazon Geräte 35,6%, während Google 29,9% erreichte. Alibaba, als Hersteller mit den drittmeisten Verkaufszahlen kommt auf einen Wert von 7.3% [48]. Dies zeigt die Dominanz von Amazon und Google, zumindest innerhalb der Kategorie der smart speaker.

Zusätzlich zu der Interaktion via smart speaker gibt es darüber hinaus die Möglichkeit voice assistants auf anderen Plattformen oder Systemen einzubetten. Hinsichtlich des in dieser Arbeit betrachteten Situation des

reisenden Vertriebsmitarbeiter sind hierbei im besonderen mobile Plattformen relevant, wie etwa die Möglichkeit der Interaktion via Smartphone App. Google Assistant und Alexa bieten hierfür Apps für iOS [20, 19] und Android [35, 34] an. Zusätzlich dazu ist der Google Assistant bei neueren Android Versionen direkt in das Betriebssystem integriert [33].

Mycroft, als open source voice assistant, bietet zusätzlich zu ihrem smart speaker Mark 1, die Möglichkeit der Installation auf einem Raspberry Pi und Linux [53]. Eine Anwendung für Android ist bereits verfügbar, befindet sich zum aktuellen Zeitpunkt jedoch noch in der proof of concept Phase und ist nicht bereit für gemeinen Gebrauch [50].

Snips zeichnet sich durch die Unabhängigkeit der Cloud aus. Während vorher behandelte voice assistants den code und die Verarbeitung der Daten innerhalb der cloud erledigen, wird dies bei Snips lokal auf dem Gerät erledigt. Lediglich das Training der *Spoken Language Understanding* (SLU) Komponenten findet auf deren Servern statt. Bei Snips kann der Nutzer sich einen neuen Assistenten erstellen, mit vordefinierten oder eigenen skills ausstatten, und diesen auf verschiedene Geräte installieren [21]. Dieser Assistent kann dann auf einem Raspberry Pi, Android, iOS oder Linux installiert werden [63].

Ein Vorteil dieses Vorgehens ist, dass Snips im Gegensatz zu den anderen voice assistants offline funktioniert [21]. In Anbetracht des Anwendungsfalls, bei dem eine Internetverbindung zur Kommunikation mit der SAP Cloud for Customer nötig ist, ist dieser Punkt jedoch zu vernachlässigen.

Zusätzlich zu der Funktionalität auch ohne Internetverbindung, bringt die komplett lokale Verarbeitung der Logik den Vorteil des Schutzes der Privatsphäre, da zur Interpretation des Gesagten keine Daten das Gerät verlassen. Snips kann deshalb als *private by design* bezeichnet werden. [21]

In diesem Kontext ist jedoch auch in Erwägung zu ziehen, dass eine komplett lokale Verarbeitung der Spracherkennung, Konvertieren zu Text, Intentzuweisung und Umwandeln von Text zu Sprache, Einfluss auf die Performanz des Systems hat, besonders wenn diese in von der Leistung eingeschränkt sind. In einer Arbeit von Sridhar et al. [64] wurden beispielsweise diese Prozesse testweise unterschiedlich auf das Endgerät oder die Cloud

verteilt. Das Ergebnis hierbei war, dass cloud services durchschnittlich die Verarbeitung auf Endgeräten zeitlich unterbieten.

Mycroft hingegen funktioniert ähnlich wie beispielsweise Alexa auch mit einem eigenen Mycroft back end, welches Anfragen des Nutzers bekommt, das Gesagte in Text umwandelt und dies einem intent zuweist. Diese Informationen werden dann in einer anonymen request an den skill geschickt, welcher die Antwort an das Mycroft back end zurück gibt. Bezüglich Datenschutz bzw. Privatsphäre speichert Mycroft nicht grundsätzlich Nutzerdaten, sondern bietet die Möglichkeit des opt-ins an [52]. Zudem ist Mycroft open source, was der Transparenz der Plattform zugute kommt [29].

Es ist jedoch auch zu beachten, dass es innerhalb der Features der Technologien Unterschiede gibt. Dies kann exemplarisch anhand von vordefinierten Input Typen betrachtet werden. Zu diesem Zeitpunkt bietet Snips lediglich zehn verschiedene vordefinierte input types [62]. Mycroft bietet derzeit nativ lediglich die Möglichkeit Datum und oder Zeit und eine Zahl aus einem Text zu extrahieren [51].

Im Vergleich dazu bieten Dialogflow [26] und Alexa [15] beide jeweils über 50 nutzbare vordefinierte input types an. Darunter befinden sich unter anderem ein input type für einen beliebigen Text (bzw. Suchbegriffe) oder Städte bzw. Adressen.

Es ist zu beachten, dass der im Kontext dieser Arbeit erarbeite skill wegen der Verbindung zur SAP Cloud for Customer sowieso eine Verbindung zum Internet benötigt. Im Hinblick darauf, der großen Verbreitung der Plattformen und dem Angebot von inputs wie Adressen oder Suchbegriffe, welche im Zusammenhang mit dem Vertriebskontext obligatorisch sind, wurde die für die Implementierung in Betracht zu ziehenden Plattformen auf Amazon Alexa oder Google Assistant beschränkt.

4.2 Jovo - Ein Voice App Framework

Aufbauend auf der Wahl der Plattformen stellt sich die Frage der Technologie, welche sich am besten zur Umsetzung eignet. Hierfür gibt es für verschiedene Plattformen eigene dafür konzipierte SDKs, wie etwa den

Alexa Skills Kit SDK, welcher zu diesem Zeitpunkt für Node.js, Java und Python angeboten wird [12]. Dies bietet den Vorteil, dass die Funktionalität auf die Plattform zugeschnitten ist.

Alternativ kann ein third party Framework verwendet werden, welches weitere Tools, Features und Schnittstellen bereit stellt. Jovo, ein Node.js Voice App Framework macht es zusätzlich möglich, einen Multi-Plattform skill zu entwickeln, womit zugleich Amazon Alexa und Google Assistant mit einer codebase bedient werden können [47].

Im Hinblick auf den grundsätzlich ähnlichen Aufbau einer Alexa und Google Assistant skills mit intents, input types, inputs und phrases, bietet sich eine universelle Definition dieser an. Jovo stellt hierzu ein eigenes *language model* zur Verfügung, in welchem diese einmalig definiert werden können. Dieses language model wird dann von Jovo in die plattformspezifischen language models umgewandelt. [43]

Dieser build Prozess und weitere Funktionalitäten, wie deployment in den plattformabhängigen back ends oder Starten des Servers lassen sich auch einmalig für beide Plattformen mittels eines command line interfaces umsetzen. [43]

Zusätzlich verfügt Jovo über ein natives state management und intent routing, welches es einfach und übersichtlich ermöglicht, Kontext, bzw. den Status des Dialogs in das intent handling miteinzubeziehen. Darüber hinaus verfügt es auch über verschiedene Arten der Integrationen von beispielsweise Datenbanken, welche genutzt werden können, um Nutzerdaten zu speichern. [43]

Bezüglich Technologien, welche eine Lösung für mehrere Plattformen oder Systeme anbieten, ist jedoch auch immer zu hinterfragen, ob und wie viel Möglichkeiten im Vergleich zu plattformspezifischen, nativen Frameworks oder libraries verloren geht, oder ob eine universelle, cross-platform Lösung unproportional komplexer ist, und somit eventuell mehr Aufwand erzeugt als die Entwicklung zweier eigenständiger Anwendungen.

Diesbezüglich besteht bei Jovo die Möglichkeit ausgewählte intents oder Features für die Plattformen unterschiedlich zu implementieren [44]. Außerdem können bestimmte Features, welche nur auf einer der Plattformen

möglich sind, wie beispielsweise die Alexa Reminder API [45], auch innerhalb von Jovo verwendet werden. Dies gilt auch für Features, welche in der Umsetzung der Plattformen sehr unterschiedlich sind. Ein Beispiel hierfür ist die visuelle Ausgabe [40, 38], welche in Jovo plattformspezifisch umgesetzt werden kann.

Angesichts der hohen Verbreitung der beiden von Jovo unterstützten Plattformen, den zusätzlich angebotenen Tools und Integrationen, der Möglichkeit von plattformspezifischen Behandlung ausgewählter Funktionalitäten und der Unterstützung von bestimmten individuellen Features der einzelnen Plattformen, wurde Jovo und somit ein cross-platform skill für die Umsetzung gewählt.

5 Anwendungsfall

Folgendes Kapitel beschäftigt sich mit der Analyse des Anwendungsfalls und der daraus entstehenden Spezifikation der Anforderungen. Um die Qualität der Anforderungen sicherzustellen wurde hierfür ein iterativer Designprozess durchgeführt.

Im Folgenden wird zuerst der Ablauf dieses Prozesses genauer beschrieben. Danach wird der Status Quo eines exemplarischen Arbeitsablauf im Vertriebsbereich erläutert. Zudem werden die use cases vorgestellt, welche sich daraus entwickelt haben. Neben den funktionalen Anforderungen (use cases) wird darüber hinaus auf die nicht-funktionalen Anforderungen des skills eingegangen. Zuletzt wird mit Hilfe eines Kontextszenarios ein exemplarischer Ablauf eines Tages eines Vertriebsmitarbeiters mit dem entworfenen skill dargestellt.

5.1 Ablauf der Anforderungsanalyse

Um einem Vertriebsmitarbeiter in Konsequenz den Alltag erleichtern zu können, muss großen Wert auf die Entwicklung der Anforderungen gelegt werden, welche dann durch den skill umgesetzt werden sollen. Deshalb wurden hierfür mehrere Gespräche mit einem Vertriebsmitarbeiter, welcher über praktische Erfahrung im Umgang mit Kundengesprächen und Systemen wie der SAP Cloud for Customer verfügt, durchgeführt.

Hierbei wurde ein iterativer Designprozess angewendet, welcher aus zwei Interviews bestand. Im ersten dieser Gespräche wurden grundsätzliche Punkte besprochen, wie etwa der aktuelle Alltag eines Vertriebsmitarbeiters

der Firma, welche Informationen für Vertriebsmitarbeiter generell von besonderer Relevanz sind und welche Features wünschenswert wären, bzw. was im aktuellen Ablauf verbessert werden könnte.

Auf den Erkenntnissen des ersten Interviews aufbauend wurden erste Versionen der use cases entwickelt. Um garantieren zu können, dass diese die Probleme und Wünsche von Vertriebsmitarbeitern abdecken, wurde als zweiter Iterationsschritt ein weiteres Interview durchgeführt. Dabei wurden die entwickelten use cases vorgestellt und eventuell modifiziert bzw. ergänzt. Des weiteren wurden Prioritäten der einzelnen use cases weiter präzisiert.

5.2 Status Quo

Im Zuge des ersten Interviews wurde nach einem exemplarischen Tagesablauf eines Vertriebsmitarbeiters im Außendienst gefragt. Dieser wurde so beschrieben, dass sich der Mitarbeiter einen Überblick über den Tag am Laptop verschafft und sich optimalerweise auf den ersten Besuch vorbereitet, indem er die dafür benötigten Informationen am Laptop via graphischem Interface beim System abfragt. Es sei hierbei nicht ausgeschlossen, dass diese Vorbereitungsphase, zum Beispiel bei eventuellem Zeitdruck, nicht zufriedenstellend ausgeführt werden kann.

Danach setzt sich der Mitarbeiter meist alleine in ein Auto und fährt zum Kunden. Im Gespräch mit dem Kunden gilt es, die vorher abgerufenen Informationen, wie etwa offene Angebote, letzte Bestellungen usw. abrufen und den Kunden eventuell darauf ansprechen zu können. Nach dem Gespräch fährt der Mitarbeiter wieder zurück oder eventuell zum nächsten Termin. Wichtig ist hierbei, dass neue durch das Gespräch entstandene Informationen, Aufträge oder ähnliches, danach auch im System vermerkt werden müssen.

5.3 Relevante Daten im Kundentermin

Um den Funktionsumfang des zu implementierenden skills weiter zu präzisieren, wurden innerhalb der Interviews konkret nach den für einen im Außendienst arbeitenden Mitarbeiter relevanten Informationen, besonders in der direkten Vorbereitung auf ein Kundengespräch, gefragt. Hierbei wurden des appointments, accounts, opportunities, sales quotes und sales orders im System als wichtigste Datenobjekte genannt. Im Folgenden werden diese genauer betrachtet.

Die Repräsentation eines Termins (appointments siehe 2.3.6) im System, mit den darin enthaltenen Informationen ist der zentrale Punkt der Betrachtung eines Vertriebsmitarbeiters in der Vorbereitung eines Kundengesprächs. Neben den grundsätzlichen Informationen, wie etwa der Titel oder Uhrzeit des Termins sind hier unter anderem die Gesprächsteilnehmer relevant. Zudem wird ein hoher Wert auf Protokollierung gelegt, womit nachvollziehbar ist, was zu besprechen ist, welche Fragen aufgekommen sind oder was in Zukunft zu erledigen ist. Demnach soll das Erstellen einer Notiz über eine freie Texteingabe möglich sein.

Zusätzlich kann das appointment einem Kunden (account siehe 2.3.5) zugewiesen werden, welcher zusätzlich weitere möglicherweise relevante Information für den anstehenden Termin enthält. Dazu gehören beispielsweise Kontaktinformationen, wie der Hauptansprechpartner, Adresse usw., aber auch wiederum Referenzen auf appointments oder andere Datenobjekte welche für den account von Relevanz sind.

Zuletzt sind für den Termin eventuell leads (siehe 2.3.1), opportunities (siehe 2.3.2), sales quotes (siehe 2.3.3) oder sales orders (siehe 2.3.4) im Bezug zum account von Relevanz. Während vergangene Bestellungen zwar Gegenstand des Gesprächs sein können, liegt das Hauptaugenmerk im Kundengespräch eines Vertriebsmitarbeiters jedoch auf dem Verkauf, was bedeutet, dass die sales quotes in diesem Kontext einen besonders hohen Stellenwert einnehmen. Hierbei sind dann vor allem der Titel, die Summe, der Status und evtl. die Produkte von Relevanz.

5.4 Wünschenswerte Skill-Features

Aus der Analyse der für den Mitarbeiter relevanter Daten (siehe 5.3) lässt sich das Bedürfnis nach dem Abfragen dieser Informationen herleiten. Zusätzlich dazu sind das Erstellen oder Bearbeiten einiger für den Anwendungskontext angemessenen Daten wünschenswert, wie beispielsweise das Erstellen eines neuen appointments, oder das Verändern der Uhrzeit dessen. Darüber hinaus wurden innerhalb der Interviews weitere wünschenswerte Features erarbeitet, welche im Folgenden beschrieben werden.

Eines dieser Features ist die Ausgabe eines Briefings, was bedeutet, dass der skill auf Anfrage eine Zusammenfassung relevanter Informationen der Datenobjekte liefern soll. Im Kontrast zu der Abfrage jeder einzelnen interessanter Information, wie das einzelne Nachfragen nach Titel, Zeit und Ort eines Termins, soll das System von selbst abhängig vom Kontext entscheiden, welche der Daten den Nutzer interessieren können, und diese auszugeben. Das bedeutet jedoch gleichzeitig, dass nicht zu viele, in dem Kontext eventuell nicht relevante, Informationen aufgelistet werden sollen.

Dazu gehört auch, dass eventuell Daten durch das System ausgewertet werden, und je nach Ergebnis der Nutzer beispielsweise auf etwas aufmerksam gemacht werden soll. Denkbar wäre hier beispielsweise, dass der Nutzer darauf aufmerksam gemacht wird, wenn der Kunde auf ein Angebot, welches abgelaufen ist, noch nicht reagiert hat. Durch das direkte Ansprechen dessen durch den skill, soll sicher gestellt werden, dass sehr wichtige Sachverhalte oder Informationen bei dem Nutzer ankommen, ohne dass dieser explizit danach fragen muss.

Des Weiteren besteht der Wunsch, den Nutzer zu ermutigen Protokoll zu führen, es ihm also so einfach wie möglich zu machen, Notizen zu beispielsweise einem Termin hinzuzufügen. Konkret bedeutet dies, dass es möglich sein soll innerhalb eines Befehls eine Notiz dem vergangenen, aktuellen oder kommenden Termin hinzuzufügen.

5.5 Use Cases

Im Folgenden werden die für den praktischen Teil dieser Arbeit erstellten use cases vorgestellt, welche anhand der Interviews und Gesprächen mit Vertriebsmitarbeitern verschiedener Firmen entwickelt wurden. Diese daher entstandenen use cases wurden so ausgewählt, dass sie dem Szenario des Vertriebsmitarbeiters auf dem Weg zum Kunden entsprechen. Dementsprechend wurden die hierfür primär relevanten Datenobjekte des appointments, accounts und sales quotes als Gegenstand der use cases ausgewählt.

Es ist zu beachten, dass für jeden der use cases die Grundvoraussetzung gilt, dass der Nutzer Zugriff auf den skill hat, eine Internetverbindung besteht und der skill mit einer vorhandenen SAP Cloud for Customer verbunden ist. Die folgenden use cases wurden anhand der use case Definition von Moser [49] definiert.

5.5.1 Appointments abfragen

Vorbedingung

- Keine

Auslöser

- Nutzer fragt nach appointments für einen bestimmten Tag
- Nutzer fragt nach den nächsten n appointments
- Nutzer fragt nach appointments eines Kunden

Normaler Ablauf

1. Das System fragt appointment in der Cloud ab
2. Das System sagt die Anzahl der gefundenen appointments an
3. Das System gibt stückweise appointments aus
4. Das System gibt evtl. weitere Möglichkeiten der Interaktion aus

Alternativer Ablauf

Es werden keine appointments gefunden:

1. Das System fragt appointments in der Cloud ab
2. Das System gibt aus, dass es keine appointments gefunden hat
3. Das System beendet die session

Es wird nur ein appointment gefunden:

1. Das System gibt eine genauere Beschreibung des appointments aus
2. Das System gibt evtl. weitere Optionen aus
3. Das gefundene appointment kann bearbeitet, gelöscht oder gespeichert werden

Nachbedingung Erfolg

- Die gefundenen appointments sind in der session gespeichert und können selektiert werden (siehe 5.5.2)

5.5.2 Appointment abfragen

Vorbedingung

Für Abfrage per Titel:

- temporär gespeicherte appointments (siehe 5.5.1)

Auslöser

- Nutzer fragt nach bestimmtem appointment via Titel
- Nutzer fragt nach letztem / aktuellen / nächsten appointment

Normaler Ablauf

1. Das System sucht das passende appointment, je nach Eingabe
2. Das System fragt alle relevanten Informationen über das appointment in der Cloud ab
3. Das System gibt einen detaillierten Bericht über das appointment aus
4. Das System gibt evtl. weitere Möglichkeiten aus

Alternativer Ablauf

Es wird kein appointment gefunden:

1. Das System sucht nach dem entsprechenden appointment
2. Das System gibt aus, dass es kein appointment gefunden hat
3. Das System beendet die session

Nachbedingung Erfolg

- Das gewählte appointment wurde selektiert und kann weiterhin bearbeitet werden (siehe 5.5.4)

5.5.3 Appointment erstellen

Vorbedingung

- keine

Auslöser

- Nutzer äußert Wunsch ein neues appointment zu erstellen

Normaler Ablauf

1. Das System fragt nach den fehlenden verpflichtenden Informationen (Zeit, Thema, usw.)
2. Der Nutzer stellt Informationen bereit
3. Das System erstellt das appointment temporär
4. Das System teilt mit, dass dieses appointment nun bearbeitet, gespeichert oder gelöscht werden kann
5. Der Nutzer speichert das appointment

Alternativer Ablauf

Es können Informationen nicht verstanden werden:

1. Das System fragt automatisch erneut nach einem gültigen Wert
2. Der Nutzer liefert einen gültigen Wert

Es gibt bereits ein appointment zu dieser Zeit:

1. Das System gibt aus, dass zu diesem Zeitpunkt bereits ein appointment besteht
2. Das System bietet eine alternative Zeit an
3. Der Nutzer lehnt dies ab, oder bestätigt dies

Nachbedingung Erfolg

- Das erstellte appointment wurde lokal und in der cloud erstellt
- Das erstellte appointment kann weiterhin bearbeitet werden (siehe 5.5.4)

5.5.4 Appointment bearbeiten

Vorbedingung

Optional:

- Ausgewähltes appointment (siehe 5.5.2)

- Erstelltes appointment (siehe 5.5.3)

Auslöser

- Nutzer äußert welches Attribut des appointments einen neuen Wert bekommen soll

Normaler Ablauf

1. Das System fragt eventuell nach, ob es den Wert richtig verstanden hat
2. Der Nutzer bestätigt
3. Das System speichert die Änderung lokal
4. Das System gibt evtl. die geänderte Information erneut aus
5. Der Nutzer macht evtl. weitere Veränderungen
6. Der Nutzer sagt, dass er speichern möchte
7. Das System speichert das appointment in der Cloud

Nachbedingung Erfolg

- Das appointment wurde lokal verändert
- Das appointment wurde in der Cloud verändert

5.5.5 Appointment löschen

Vorbedingung

Optional:

- Ausgewähltes appointment (siehe 5.5.2)
- Erstelltes appointment (siehe 5.5.3)

Auslöser

- Nutzer äußert dass er ein appointment löschen möchte

Normaler Ablauf

1. Das System fragt, ob das appointment wirklich gelöscht werden soll
2. Der Nutzer bestätigt dies
3. Das System löscht das appointment in der Cloud (wenn vorhanden)
4. Das System löscht das appointment lokal
5. Das System gibt Rückmeldung über die Löschung

Nachbedingung Erfolg

- Das lokal gespeicherte appointment ist gelöscht
- Das appointment in der Cloud ist gelöscht

5.5.6 Account abfragen

Vorbedingung

- keine

Auslöser

- Nutzer fragt nach account mit einem bestimmten Namen oder Nummer
- Nutzer fragt nach dem account eines bestimmten appointments
- Nutzer fragt nach dem account einer sales quote

Normaler Ablauf

1. Das System bezieht relevante Informationen von der Cloud
2. Das System gibt eine kurze Zusammenfassung des accounts aus
3. Das System gibt evtl. weitere Möglichkeiten aus

Nachbedingung Erfolg

- Der account ist lokal gespeichert
- Der Nutzer kann weitere Informationen des accounts abfragen

5.5.7 Sales Quotes abfragen

Auslöser

- Nutzer fragt nach den sales quotes eines accounts

Normaler Ablauf

1. Das System bezieht relevante Informationen von der Cloud
2. Das System gibt stückweise die gefundenen sales quote titles aus
3. Das System gibt evtl. weitere Optionen aus

Alternativer Ablauf

Es wurden keine sales quotes gefunden:

1. Das System gibt zurück, dass für diesen account keine sales quotes gibt
2. Das System beendet die session

Es wird nur ein Angebot gefunden:

1. Das System gibt eine detailliertere Beschreibung der gefundenen sales quote aus
2. Das System gibt evtl. weitere Optionen aus
3. Die sales quote wird in der session gespeichert
4. Der Nutzer kann weitere Attribute der sales quote abfragen

Nachbedingung Erfolg

- Die gefundenen sales quotes sind in der session gespeichert und können selektiert werden (siehe 5.5.8)

5.5.8 Sales Quote abfragen

Vorbedingung

Für die Auswahl per Titel:

- temporär gespeicherte sales quotes (siehe 5.5.7)

Auslöser

- Nutzer fragt nach bestimmter sales quote

Normaler Ablauf

1. Das System bezieht relevante Informationen von der Cloud
2. Das System gibt eine detaillierte Beschreibung der sales quote
3. Das System bietet bei Auffälligkeiten Hinweise bzw. eine Analyse des Angebots
4. Das System gibt evtl. weitere Optionen aus

5.6 Use case Diagramm

Abbildung 5.1 stellt die visuelle Repräsentation der umzusetzenden use cases dar und verdeutlicht die Beziehung dieser use cases zueinander. Die dichte Vernetzung der einzelnen Punkte zeigt die verschiedenen Wege, wie ein Nutzer zu einem use case gelangen kann. Durch eine Vernetzung von beispielsweise dem appointment mit dem account, ist es dem Nutzer möglich Informationen zu einem account über ein appointment, eine sales quote, oder via direkter Anfrage zu bekommen. Die Möglichkeit des Nutzers

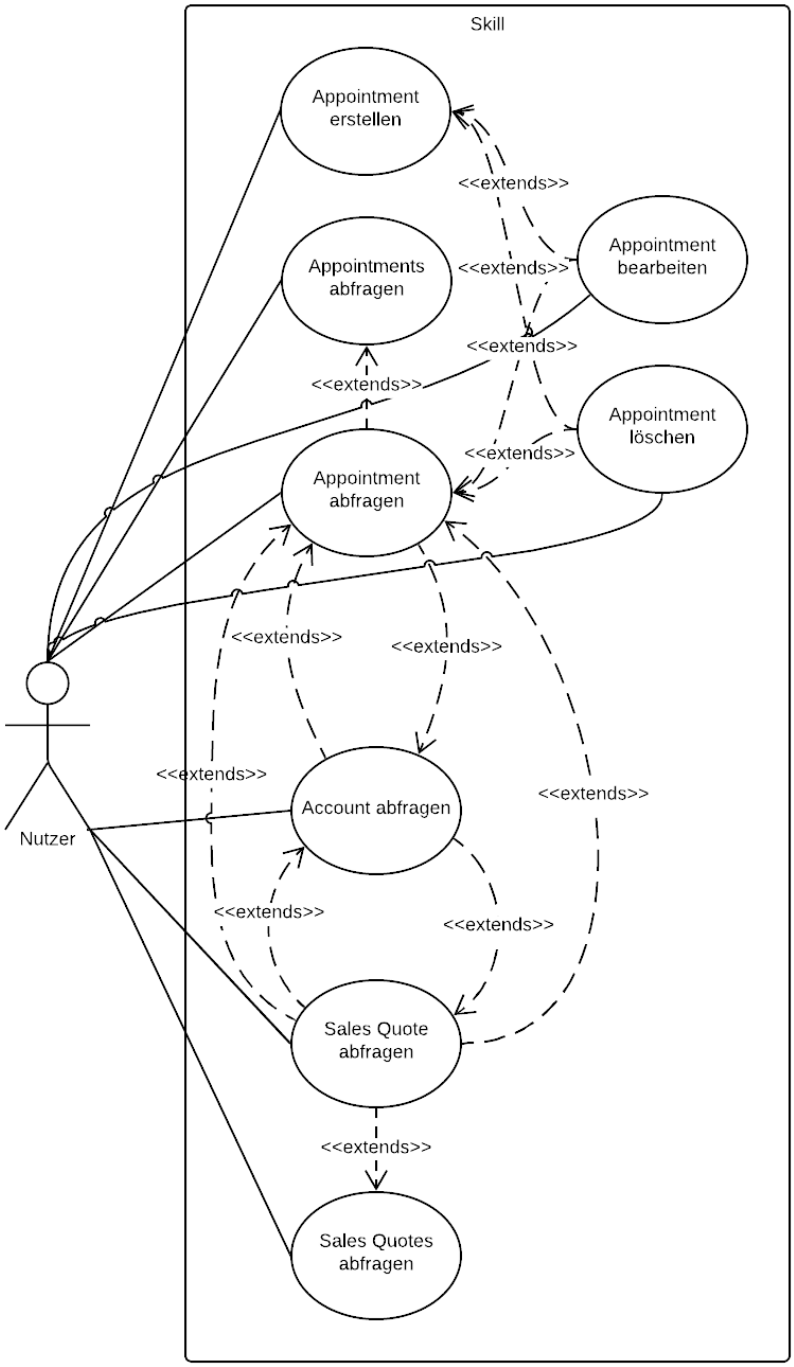

Abbildung 5.1: Use Case Diagramm

direkt auf alle der dargestellten use cases zuzugreifen, repräsentiert die horizontale Struktur (siehe 3.2.3), auf welche im skill Design zu achten ist.

5.7 Nicht-funktionale Anforderungen

Bei der Entwicklung von skills ist neben dem vorangehend beschriebenen fest definierten Funktionsumfang auch ein besonderes Augenmerk auf die Qualität dessen Umsetzung zu legen. Damit soll sicher gestellt werden, dass die Erwartungen des Nutzers erfüllt werden können. Im Zuge dessen wird im Folgenden auf die einzelnen für diese Arbeit relevanter Betrachtungspunkte der Norm ISO/IEC 25010 eingegangen, welche das Ziel der Evaluation von Software Qualität hat. [37]

Im Bezug auf die Charakteristik *Functional Suitability* ist sicherzustellen, dass die definierten use cases (siehe 5.5) vollständig im skill implementiert wurden. Darüber hinaus haben die ausgegebenen Informationen denen der SAP Cloud for Customer zu entsprechen.

Des weiteren ist die sogenannte *Performance Efficiency* zu beachten. Dies bedeutet, dass der skill unter geeigneten Voraussetzungen, wie etwa einer ausreichend schnellen Verbindung zum Internet, welche die Kommunikation mit dem System nicht unverhältnismäßig verlangsamt, keine spürbar unnatürlich lange Zeit für eine Antwort auf einen Befehl des Nutzers benötigen darf. Um dies zu erreichen und möglichst wenig Ressourcen zu benötigen, sollen dafür nur benötigte Attribute der Datenobjekte von der SAP Cloud for Customer angefragt und verarbeitet werden.

Compability nimmt, aufgrund der Kommunikation mit der SAP Cloud for Customer, in dieser Arbeit ebenfalls einen besonders hohen Stellenwert ein. Da der skill als Sprach-Schnittstelle der SAP Cloud for Customer dienen soll, ist das korrekte Abfragen und Senden von Informationen von und zu dieser Grundvoraussetzung. Das Vergewissern dieses Punkts ist somit also gleichzeitig Teil der funktionalen Anforderungen, also der use cases.

Die Anforderungen bezüglich der Charakteristik *Usability* sind hinsichtlich der schweren Orientierung der Möglichkeiten durch den Nutzer (siehe

3.1.1) und einer Vielfalt an Fehlern durch Missverständnisse, falsche Eingaben usw. (siehe 3.1.2, 3.1.3), ein weiterer wichtiger Betrachtungspunkt. Deswegen muss durch eine Hilfestellung für jeden Zustand des Skills die Möglichkeit zum Erlernen der Funktionen gegeben sein und das System auf viele verschiedene Formulierungen (10 phrases oder mehr) ein und des selben Befehls vorbereitet sein (siehe 3.2.1). Dies stärkt neben der Usabilty zusätzlich die *Reliability* des Systems.

Durch das Verwenden von mehreren Formulierung ein und des selben Inhalts (siehe 3.2.4), vor allem für oft verwendete intents, soll der Dialog zusätzlich unterhaltsamer für den Nutzer gestaltet werden.

Da der im Kontext dieser Arbeit entwickelte skill nur ein Subset der Informationen und Funktionen, welche die SAP Cloud for Customer bietet, abdeckt, ist zusätzlich die *Maintainability* sicherzustellen. Dies geschieht durch Modularität. Konkret bedeutet dies, dass der code durch strukturierten Aufbau das Hinzufügen weiterer Funktionalitäten und Daten leicht ermöglicht.

Im Zuge dessen gibt es für die Datenobjekte der Cloud for Customer repräsentative models, welche nach einem bestimmten Schema aufgebaut sind. Mit Vorbild dieses Schemas können weitere Datenobjekte implementiert werden. Zusätzlich dazu sind intents, welche sich auf diese beziehen, in separaten handlern umzusetzen, was das Erweitern des Funktionsumfangs durch Erstellen eines neuen handlers und Einfügen in das erstellte Schema ermöglichen soll. Durch die strikt getrennte Trennung von Logik (handler), Input (language model), output und Datenobjekte (models) wird das einzelne analysieren, testen und evtl. modifizieren der einzelnen Komponenten bzw. Module ermöglicht.

Da die *Portability* im Bereich der skill-Entwicklung zu einem großen Teil von der Wahl der Plattform (wie Google Assistant, oder Amazon Alexa) eingeschränkt wird, soll der skill nicht nur für ein, sondern für zwei Plattformen entwickelt werden, was es dem Nutzer ermöglichen soll, den selben Funktionsumfang (siehe 5.5) unabhängig von dem von ihm präferierten angebotenen Plattform, zu bekommen. Im Zuge dessen soll der skill sowohl für Alexa-Geräte, als auch für Google Assistant nutzbar sein.

5.8 Kontextszenario

Folgendes Szenario beschreibt einen beispielhaften Ausschnitt eines Arbeitsalltags eines Vertriebsmitarbeiters einer Firma mit dem SAP Cloud for Customer voice assistant skill:

Martin Müller, ein Vertriebsmitarbeiter einer mittelgroßen Hardware Firma fährt mit dem Auto in das Büro. Während der Fahrt startet er den SAP Cloud for Customer skill mit der Alexa App seines Smartphones und fragt "What are my appointments for today?" um sich einen Überblick zu verschaffen. Der Assistent listet ihm die Termine geordnet nach Uhrzeit auf. Um zu erfahren, wann und wo der erste Termin ist, fragt Martin "Tell me more about sales meeting with SAP.", woraufhin der Assistent antwortet, von wann bis wann und mit welchem account das Meeting ist. Danach kommt Martin im Büro an, interagiert per Webanwendung mit der SAP Cloud for Customer und fügt dem Sales Meeting mit SAP die Notiz hinzu, den Kunden nach Zufriedenheit mit der Firma zu fragen.

Rechtzeitig setzt sich Martin nun in das Auto um zu dem Sales Meeting mit SAP zu fahren und nutzt die Zeit der Fahrt, um sich per skill über relevante Informationen bezüglich des Termins zu informieren. Dafür sagt er "Give me the notes for my next appointment" wodurch ihm alle Notizen des nächsten Termins ausgegeben werden. Um zu erfahren, welche Teilnehmer Martin zu erwarten hat, fragt er "tell me who will be there." und bekommt eine Liste der von der externen Firma teilnehmenden Mitarbeiter.

Da Martin vorher per Webanwendung sales quotes, welche er mit dem Kunden besprechen möchte, mit dem appointment verlinkt hat, fragt er zusätzlich "Give me the sales quotes", wodurch ihm eine Liste von sales quotes mit den Titeln ausgegeben wird. Martin möchte genauer wissen, um welche Produkte es bei der sales quote mit dem Titel "computers for new office" geht und fragt demnach den skill "Give me the products for computers for new office". Martin möchte bezüglich des Treffpunkts den Hauptansprechpartner des Kunden SAP kontaktieren, und fragt diesen bei dem skill mit "What is the main contact for SAP" ab, wodurch er den Namen geliefert bekommt und diesen anruft.

Nach dem Meeting setzt sich Martin wieder in sein Auto und protokolliert wichtige Notizen über aufgekommene Gesprächsthemen in die Notizen des Termins, indem er “Add a note about time was not sufficient to the last appointment”. Dies bekommt er nochmal vorgelesen, bestätigt dies und speichert es in der SAP Cloud for Customer.

Er möchte zusätzlich einen weiteren Termin erstellen, in welchem weitere Fragen zu neuen Angeboten geklärt werden. Dazu sagt er “Create a new appointment”. Er liefert dann auf Nachfrage des Systems Informationen wie Titel und Zeitpunkt des Termins. Daraufhin synchronisiert Martin das erstellte appointment mit der SAP Cloud for Customer, indem er “Save that” sagt. Er fügt zudem mit dem Kommando “The account should be SAP” den Kunden zum Termin hinzu und fügt mit“Add Max Mustermann to the attendees” den Gesprächspartner hinzu.

6 Konzept

6.1 Datenobjekte

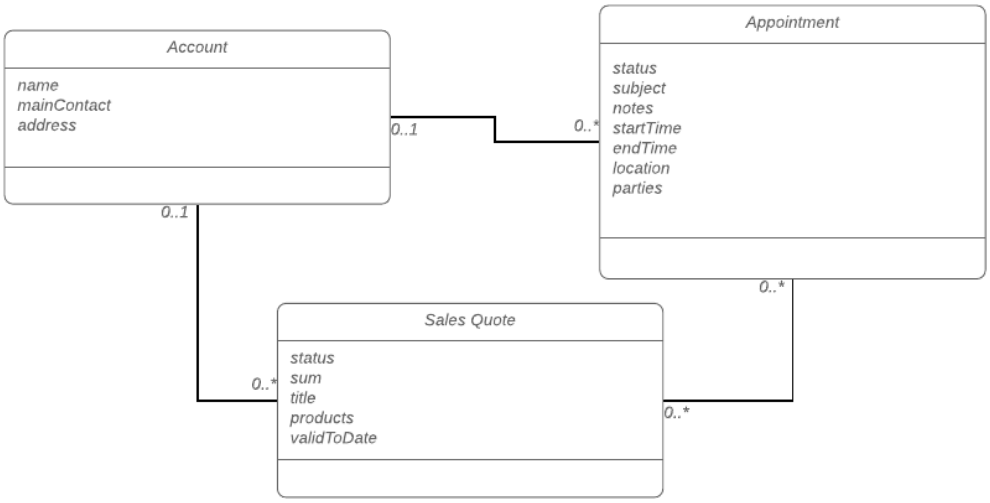

Abbildung 6.1: Datenobjekte des SAP Cloud for Customer Skills

Abbildung 6.1 zeigt die im skill implementierten Informationen. Diese Attribute der Datenobjekte account, appointment und sales quote aus der SAP Cloud for Customer stellen ein Subset der kompletten Datenmenge dar, welche darin zu finden sind. Diese Informationen werden jeweils für die einzelnen accounts, appointments oder sales quotes abgefragt und können im skill ausgegeben, evtl. bearbeitet oder gelöscht werden.

Konzeptuell sind hier besonders die Verbindungen dieser Datenobjekte zueinander wichtig. Dies bedeutet, dass die diese miteinander verknüpft

sind. Somit kann sich der Nutzer beispielsweise Informationen zu einem appointment anhören, von dort aus den account abfragen, und weiter zu dessen letzten sales quotes springen. Dies ist direkt möglich, ohne bereits gesagte Informationen, wie den Namen des accounts, wiederholen zu müssen.

6.2 Dialogführung

Im Folgenden werden für die innerhalb der Anforderungsanalyse definierten use cases (siehe 5.5), die zugehörigen Konzepte der Dialogführung und interne Prozesse des im Kontext dieser Arbeit entwickelten skills detailliert vorgestellt. Dafür werden Flow Charts verwendet. Hierbei sind die Elemente, welche mit der Spracheingabe des Nutzers zusammenhängen blau eingefärbt, während systeminterne Prozesse, Entscheidungen und Ausgaben grau gefärbt sind.

6.2.1 Appointments abfragen

Abbildung 6.2 zeigt, dass es verschiedene Wege der Abfrage von appointments gibt. Dazu gehören phrases wie “What appointments do I have today?”, “What is on my schedule?” oder “What are my next 4 appointments?”. Wenn kein Datum oder Anzahl genannt wurde, werden die appointments des aktuellen Tages abgefragt. Alternativ dazu kann der Nutzer bei zuvor abgefragtem account die Funktion mit einer phrase wie “Give me the appointments for that” auslösen. Die weiteren Schritte sind abhängig von der Anzahl der gefundenen appointments. Wurden keine gefunden, wird dies ausgegeben und die Session wird beendet. Wenn mehrere gefunden werden, folgt eine Ausgabe in Einheiten von fünf und eine Abfrage, ob weitere appointments ausgegeben werden sollen.

Zusätzlich dazu befindet sich der skill nun im AppointmentsState, welcher dem Nutzer neue Interaktionsmöglichkeiten bietet. Es sind dem Nutzer nun also zusätzlich zu den globalen (*stateless*) intents die des AppointmentsState zugänglich. Wurde nur ein appointment gefunden, werden darüber mehr

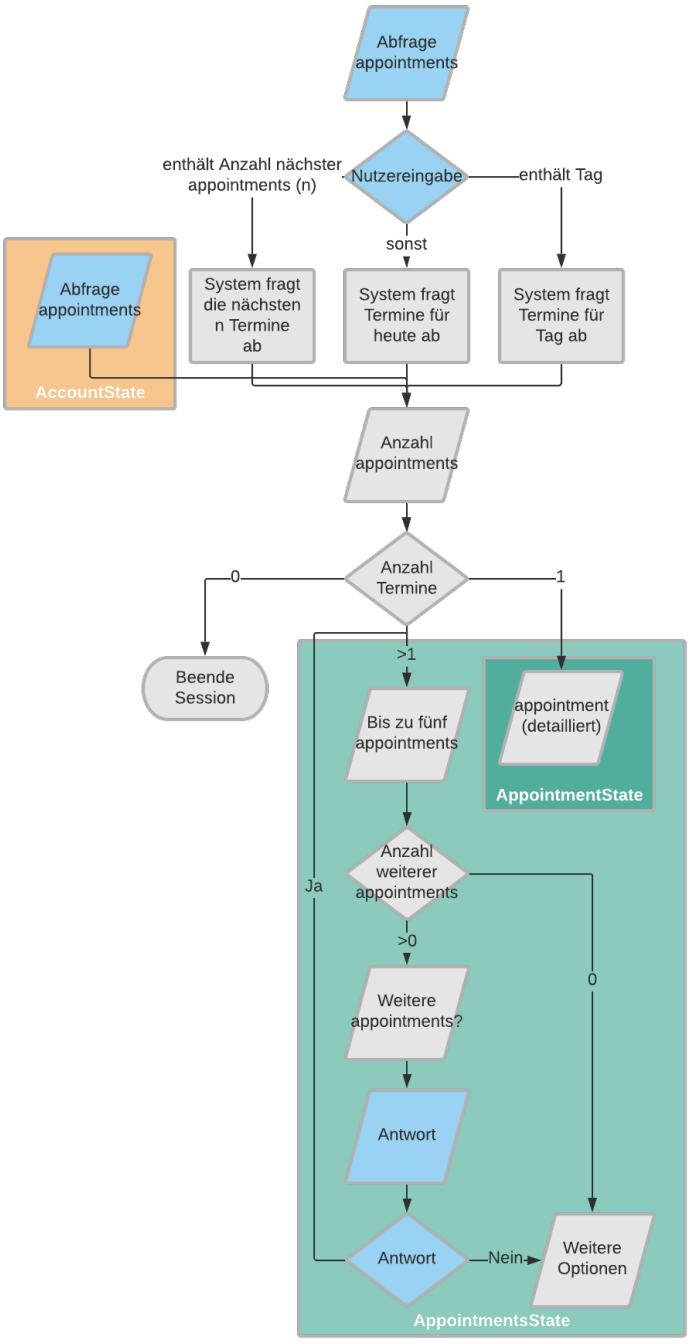

Abbildung 6.2: Appointments abfragen - Ablauf

Informationen ausgegeben und der Dialog wird direkt in den Appointment-State geleitet, welcher sich innerhalb des AppointmentsState befindet. Damit kann das appointment beispielsweise direkt bearbeitet (siehe 6.2.5) werden, ohne dies manuell selektieren zu müssen.

6.2.2 Appointment abfragen

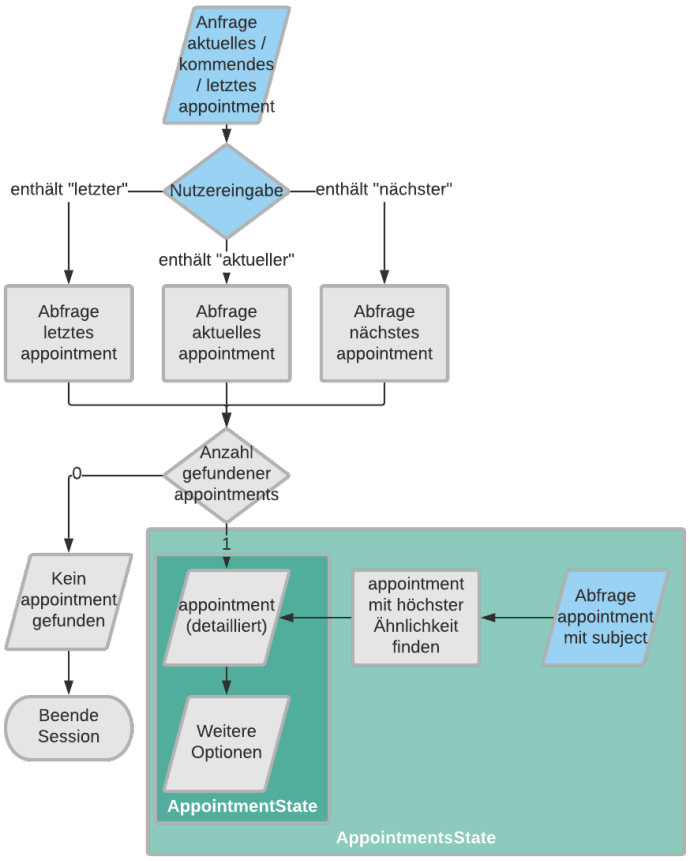

Abbildung 6.3: Appointment abfragen - Ablauf

Abbildung 6.3 zeigt den Ablauf des use cases der Abfrage eines einzelnen appointments. Es ist hierbei möglich aus dem globalen state das letzte, ak-

tuelle oder nächste appointment abzufragen, welche, wenn diese gefunden wurden, ausgegeben werden. Wenn hierbei keines gefunden wird, soll die Interaktion beendet werden. Falls der Nutzer vorher bereits appointments abgefragt hat (siehe 6.2.1) und sich somit innerhalb des AppointmentsState befindet, kann dieser auch mit einem Satz wie "Tell me the details about sale meeting with SAP" die vorher abgefragten appointments mit Hilfe des subjects "Business meeting with SAP" filtern. Hierbei sollen die subjects der Auswahl der appointments nach dem minimalen Unterschied zu der Eingabe durchsucht werden. Der so gefundene Termin wird dann ausgegeben und der state wird auf AppointmentState gesetzt.

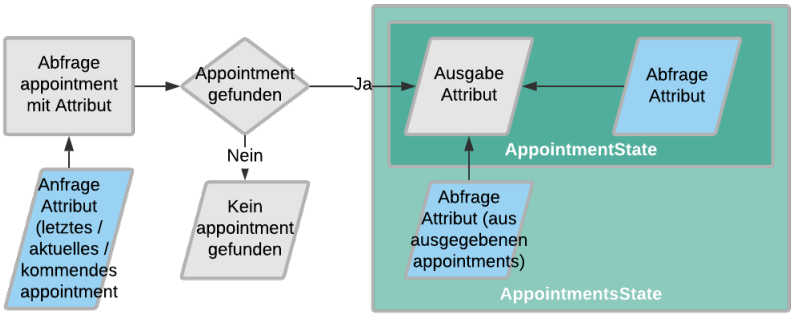

Abbildung 6.4: Appointmentattribut abfragen - Ablauf

Vor allem für erfahrenere Benutzer, welche genau wissen, welche Attribute sie benötigen, soll es ermöglicht werden, diese einzeln abzufragen (siehe Abbildung 6.4). Auch hierbei soll es möglich sein aus allen states heraus das gefragte Attribut abfragen zu können. Global kann beispielsweise die phrase "Give me the location for the current appointment" erkannt werden, welcher das appointment abfragt, die Information ausgibt und den state AppointmentState setzt, womit der Nutzer nun nur noch "Give me the status" sagen muss, um den Status abfragen zu können.

Bestehen abgefragte appointments (siehe 6.2.1) kann der Nutzer die Anfrage beispielsweise mit Hilfe des subjects formulieren. Es kann also mit der phrase "Tell me the location of sale meeting with SAP" der Ort abgefragt werden. Auch hier befindet sich der Nutzer nach der Ausgabe des Orts im state AppointmentState, von wo aus weitere Modifikationen oder Abfragen

für diesen Termin gemacht werden können.

6.2.3 Appointment erstellen

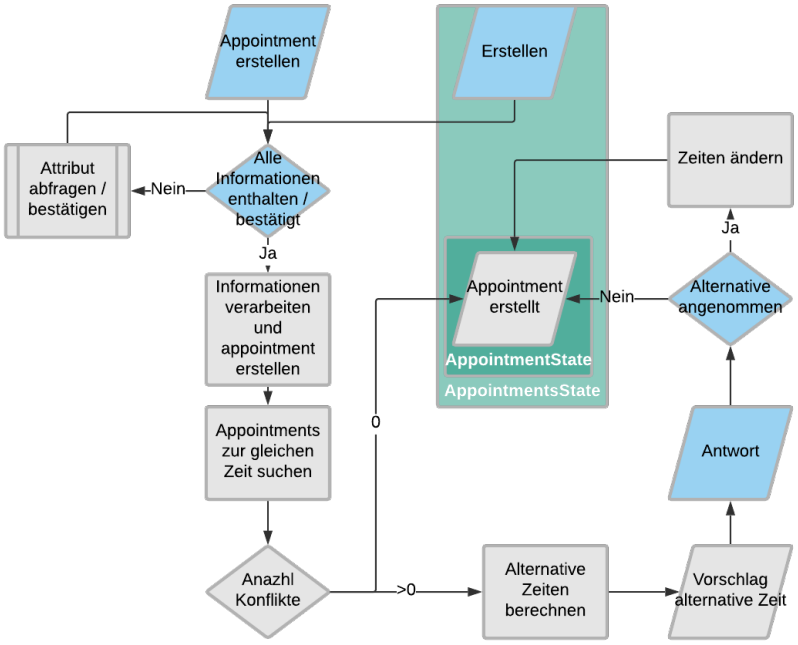

Abbildung 6.5: Appointment erstellen - Ablauf

Den Programmablauf des Erstellens eines appointments zeigt Abbildung 6.5. Es ist ohne bestimmten state möglich, mit einer phrase wie “Create a new appointment about customer meeting with SAP” die Funktionalität anzustoßen. Zusätzlich dazu soll es in dem AppointmentsState möglich sein, die Aktion mit z. B. “Create a new one” zu starten. Dass der Nutzer ein appointment erzeugen möchte, wird in dem Fall aus dem Kontext, also dem state, erschlossen. Danach fragt das System automatisch nach allen relevanten Informationen (subject, startTime, endTime). Des weiteren soll für das subject die Möglichkeit gegeben werden, diese Eingabe zu korrigieren.

Anschließend wird die SAP Cloud for Customer nach appointments durchsucht, welche sich mit den Zeiten des erstellten überschneiden. Werden diese gefunden wird eine alternative Uhrzeit vorgeschlagen. Der Nutzer kann diese vorgeschlagene Zeit annehmen, wodurch diese verändert wird, oder ablehnen. In beiden Fällen bestätigt das System dies und leitet den Dialog in den AppointmentState, von welchem weitere Attribute bearbeitet werden können (siehe 6.2.5).

6.2.4 Appointment löschen

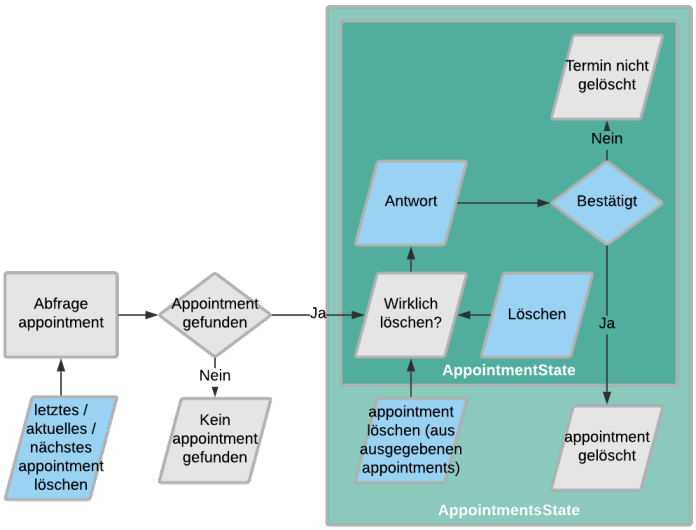

Abbildung 6.6: Appointment löschen - Ablauf

Wie Abbildung 6.6 zeigt, soll auch das Löschen eines appointments in jeder Situation bzw. in jedem state möglich sein. Konzeptuell verhält es sich dazu ähnlich zu der Abfrage eines einzelnen appointments (siehe 6.2.1). Es soll also global möglich sein mit einer phrase wie “Delete the current appointment” das aktuelle appointment zu löschen. Wurden bereits einige appointments abgefragt, können diese durch Nennen des subjects gelöscht werden, wie z.B. “Delete the customer meeting with SAP appointment”.

Wenn der Nutzer sich bereits im AppointmentState befindet, reicht die phrase “delete” aus um den Löschvorgang zu starten.

Um sicherzustellen, dass der Nutzer nicht versehentlich etwas löscht, wird zuvor durch das System gefragt, ob das appointment mit dem subject wirklich gelöscht werden soll. Abhängig von der Antwort befindet sich der Nutzer danach entweder noch im AppointmentState oder im AppointmentsState, von wo aus weitere Termine bearbeitet, gelöscht usw. werden können.

6.2.5 Appointment bearbeiten

Einzelne Attribute können mit Sätzen wie “Update the subject to Sales meeting with SAP” innerhalb des AppointmentState bearbeitet werden (siehe Abbildung 6.7). Für bestimmte Attribute, wie subject oder notes, soll das System vor dem Bearbeiten des appointments nachfragen, ob der Wert des inputs richtig verstanden wurde. Wenn der Nutzer dies bestätigt, wird das appointment vorerst lokal modifiziert. Nach beliebig vielen Veränderungen dieser Art, ist es möglich das appointment mit einer phrase wie “Save it to the cloud” zu speichern. Wurde dies nicht gemacht und der Nutzer ruft einen intent außerhalb des AppointmentState auf, fragt das System, ob die Veränderungen zuvor gespeichert oder verworfen werden sollen. Sobald der Nutzer sich für eine der beiden Optionen entschieden hat, kann der gewünschte intent aufgerufen werden.

Alternativ zu dem Aufruf der Funktionalität im AppointmentState gibt es auch hier die Möglichkeit, für relative (letzter/aktueller/nächster) appointments, bestimmte Attribute zu verändern, welche im AppointmentState münden. Auch kann bei vorheriger Abfrage von appointments (siehe 6.2.1) direkt Attribute eines spezifischen appointments verändert werden.

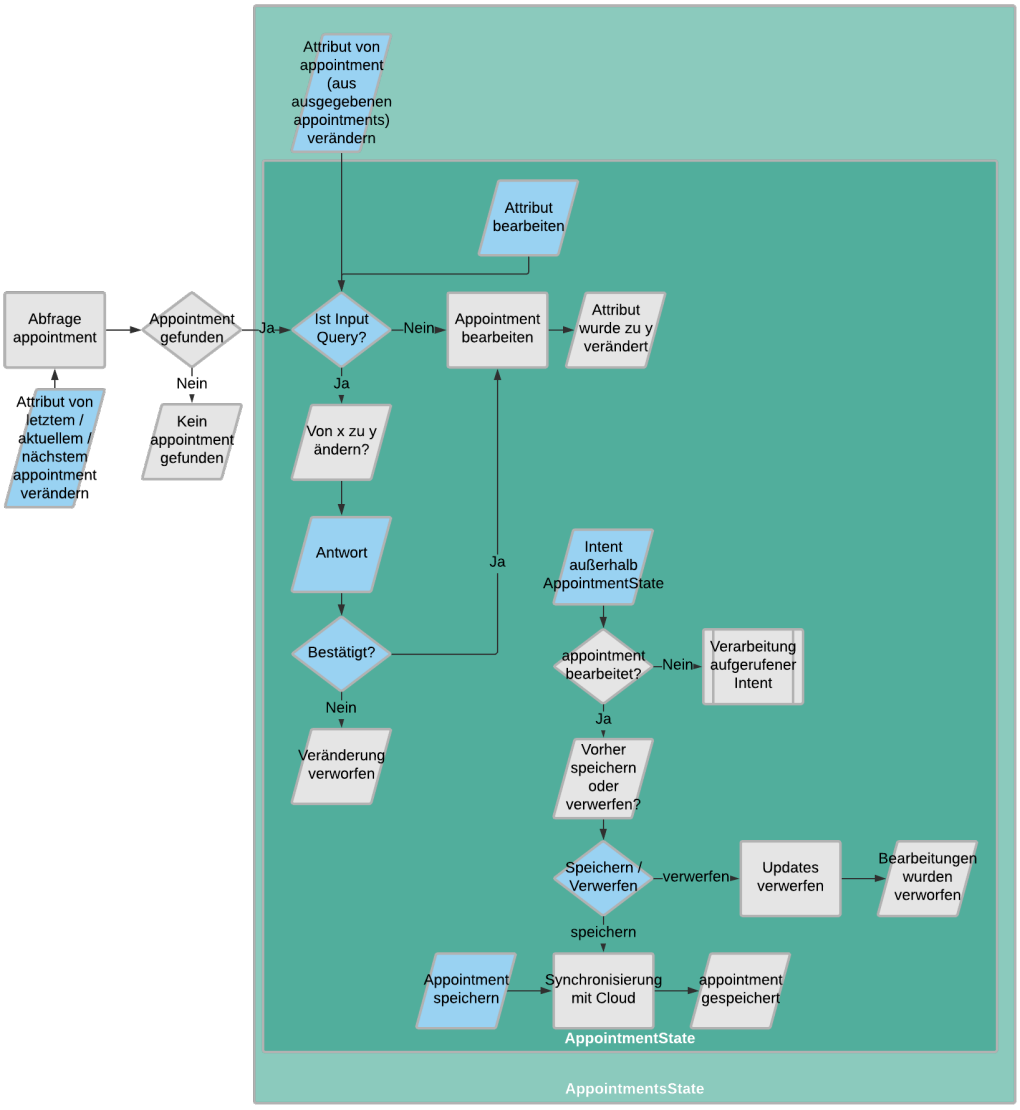

Abbildung 6.7: Appointment bearbeiten - Ablauf

6.2.6 Account abfragen

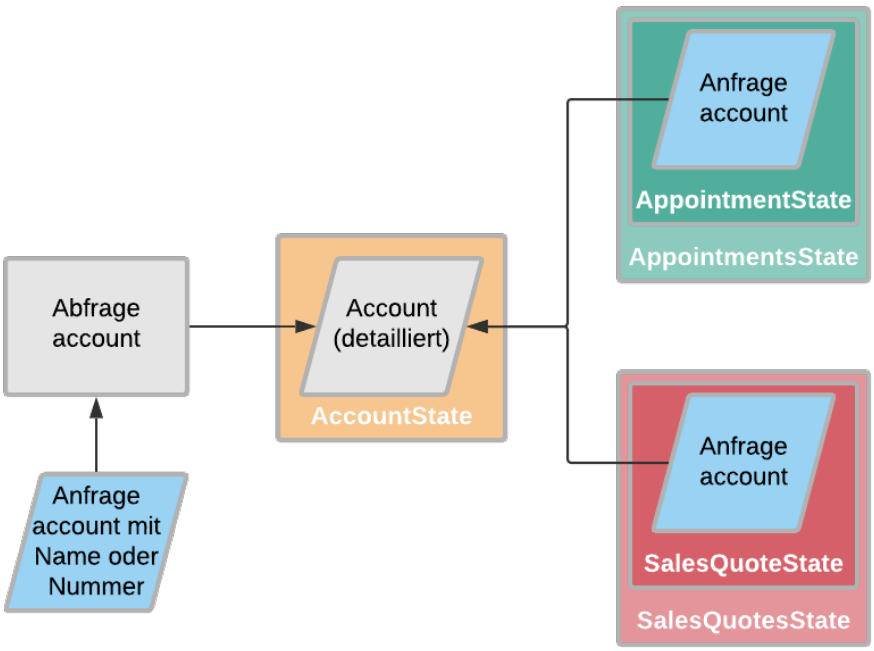

Abbildung 6.8: Account abfragen - Ablauf

Abbildung 6.8 zeigt die Möglichkeit der Abfrage eines accounts. Hier sollen name, main contact und address des accounts ausgegeben werden. Des weiteren gelangt der Kunde somit in den AccountState, welcher weitere Möglichkeiten der Interaktion bietet. Dies kann global durch eine phrase wie “Tell me about the account SAP” oder, für erfahrenere Nutzer, “Account number 10001” erreicht werden. Falls der Nutzer sich im AppointmentState oder SalesQuoteState befindet, kann dieser mit einer phrase wie “Tell me the account for that” diese Informationen abfragen.

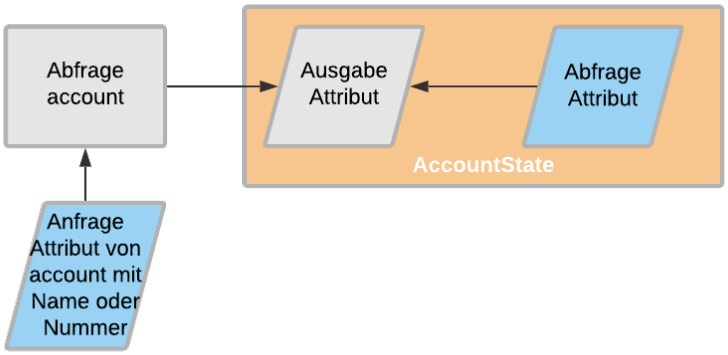

Abbildung 6.9: Accountattribut abfragen - Ablauf

Um die Ausgabe auf das Nötigste zu beschränken, gibt es auch hier die Möglichkeit Attribute einzeln abzufragen (siehe Abbildung 6.9). Dies ist einerseits innerhalb des AccountState möglich, beispielsweise mit einer phrase wie "Who is the main contact?". Global muss der Name oder Nummer des accounts angegeben werden, wie z. B. bei der phrase "Give me the main contact for SAP".

6.2.7 Sales Quotes abfragen

Das Abfragen von sales quotes (siehe Abbildung 6.10) ist konzeptionell ähnlich zu der Abfrage von appointments gestaltet (siehe 6.2.1). Hier ist es möglich global durch eine phrase wie "List me the sales quotes for SAP" eine Liste der zutreffenden sales quotes ausgegeben zu bekommen. Je nach Anzahl der gefundenen sales quotes verändert sich hier, analog zu der Abfrage der appointments, der state und die session. Auch hier werden die gefundenen items, falls diese eine Grenze überschreiten, stückweise ausgegeben, wodurch der Nutzer Kontrolle über die Anzahl der auszugebenden sales quotes hat.

Alternativ zu der globalen Abfrage gibt es gemäß der Verknüpfung der Datenobjekte (siehe 6.1) die Möglichkeit im AccountState oder AppointmentState durch eine phrase wie "Give me the sales quotes" die gleiche Funktionalität

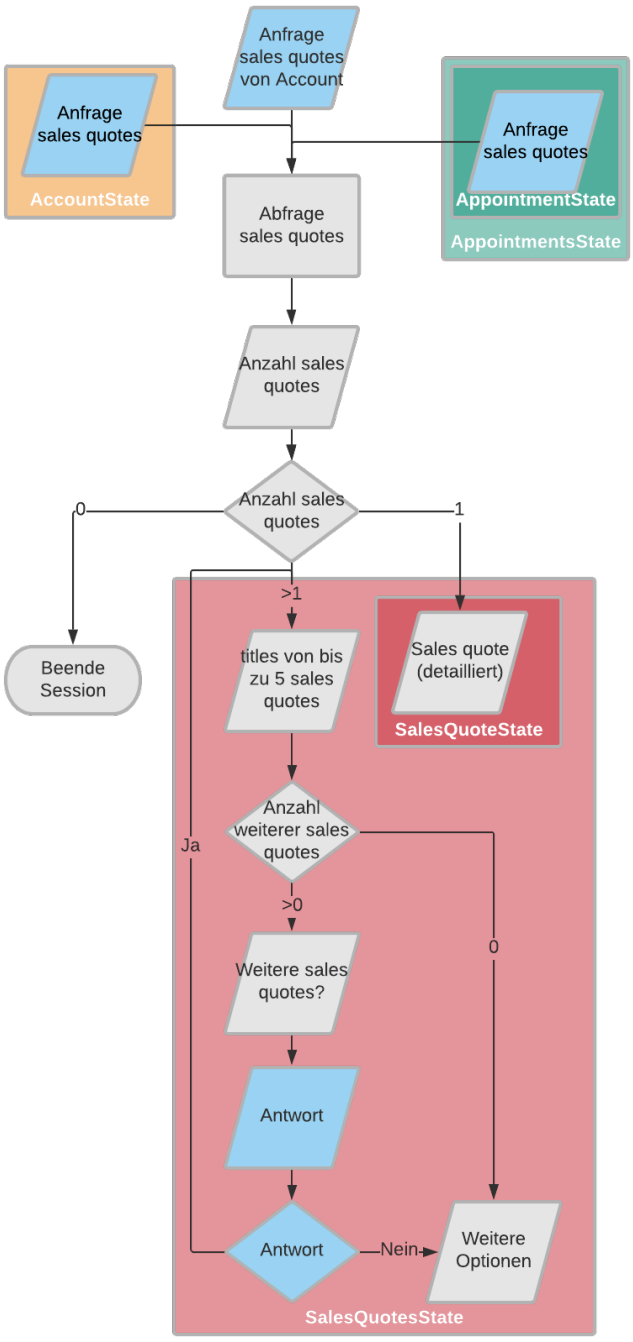

Abbildung 6.10: Sales quotes abfragen - Ablauf

aufzurufen. Im Falle des AppointmentState werden hierbei die mit dem entsprechenden appointment verlinkten sales quotes ausgegeben.

6.2.8 Sales quote abfragen

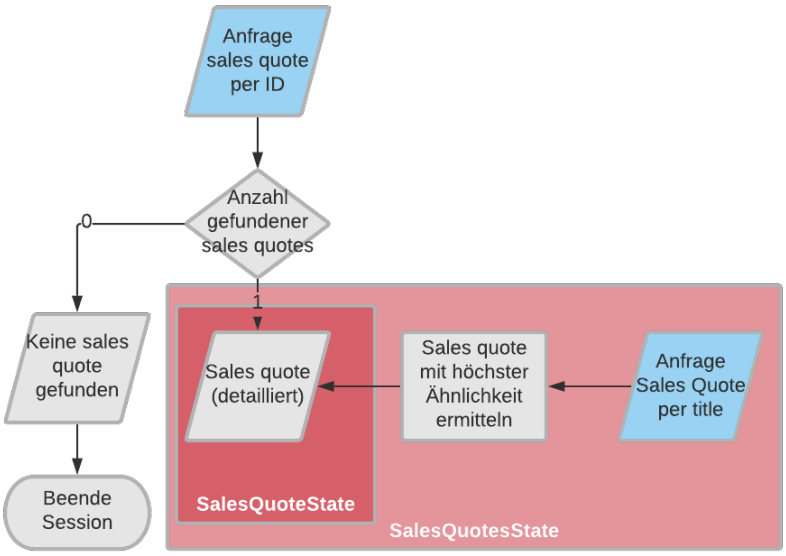

Abbildung 6.11: Sales quote abfragen - Ablauf

Die Abfrage einer einzelnen sales quote verhält sich ähnlich zu dem Abfragen eines appointments (siehe 6.2.1). Hier kann beispielsweise mit Hilfe der ID eine sales quote abgefragt werden. Wenn der Nutzer bereits sales quotes abgefragt hat (siehe 6.2.6), kann man analog zu der Auswahl eines appointments, eine sales quote mit der Nennung des titles abfragen. Daraufhin wird ein Bericht über die gewählte sales quote ausgegeben. Wenn das Attribut validTo der sales quote abgelaufen ist, wird dies durch das System zusätzlich ausgegeben.

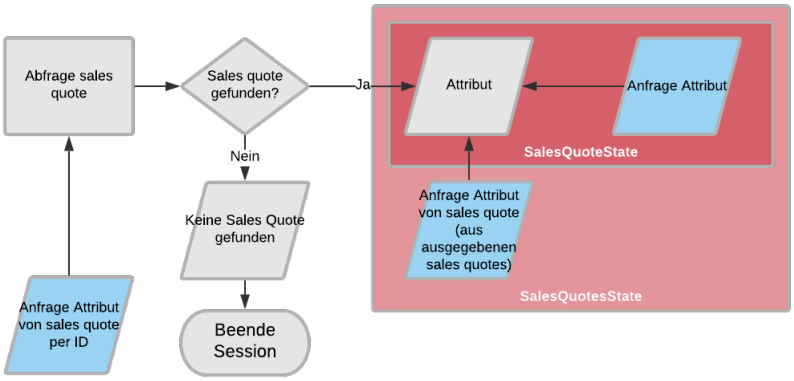

Abbildung 6.12: Sales quote Attribut abfragen - Ablauf

Auch das Abfragen eines einzelnen Attributs verhält sich analog zu der Abfrage eines Attributs eines appointments (siehe 6.2.2). Dies ist sowohl global, als auch von innerhalb des SalesQuotesState und SalesQuoteState möglich.

7 Umsetzung

Wie bereits innerhalb der Analyse (siehe 4.2) erwähnt, wurde die Umsetzung des erarbeiteten Konzepts (siehe 6) mit dem cross-platform Framework Jovo durchgeführt. Dieser auf Jovo basierende skill bezieht und sendet Daten von bzw. an eine Schnittstelle der SAP Cloud for Customer. Ein Überblick über die Funktionsweise des skills und bei der Entwicklung entstandenen Herausforderungen werden im Folgenden beschrieben.

7.1 Kommunikation mit der SAP Cloud for Customer

Die SAP Cloud for Customer bietet als Schnittstelle zum Lesen und Manipulieren von Daten die SAP Hybris Cloud for Customer OData API [60] an. Bevor hierauf Zugriffe gemacht werden können, muss eine Authentifizierung stattfinden, für welche es gemäß dem SAP Cloud for Customer OData API Developer's Guide [31] mehrere Möglichkeiten gibt.

Im Hinblick auf den Schwerpunkt der Arbeit wurde hierfür die *basic authentication* gewählt, welche mit jeder Anfrage an die Schnittstelle ein verschlüsseltes Paar aus Benutzername und Passwort schickt. Diese Paar ist derzeit in einer Konstante des Jovo back ends gespeichert. Im Zuge einer Weiterentwicklung könnte diesbezüglich die Authentifizierung auf das Endgerät ausgelagert werden.

Bei gelungener Authentifizierung können mit gängigen HTTP Requests Abfragen und Manipulationen der Daten der SAP Cloud for Customer vorgenommen werden. Hierzu bietet die Schnittstelle verschiedene *query options* an, womit man unter anderem das Format, die Anzahl oder die

Reihenfolge der Ergebnisse festlegen kann. Zudem ist es möglich die Ergebnisse abhängig von Attributwerten zu filtern oder die übermittelten Attribute einzuschränken, um ausschließlich benötigte Werte abfragen zu können. [31]

7.2 Language Model

```
{
  "name": "AddNoteToLastAppointmentIntent",
  "phrases": [
    "add a note about {note} to the last appointment",
    "add a note to the last appointment",
  ],
  "inputs": [
    {
      "name": "note",
      "type": {
        "alexa": "AMAZON.SearchQuery",
        "dialogflow": "@sys.any"
      },
      "dialogflow": {
        "required": true,
        "prompts": [
          {
            "lang": "en",
            "value": "What should I note?"
          }
        ],
        "isList": true
      }
    }
  ]
}
```

Intents können als JSON Objekt innerhalb des Jovo language models erzeugt

werden, welches im build Prozess zu den plattformspezifischen language models umgewandelt wird. Diese generierten language models werden dann auf den jeweiligen back ends der Plattformen abgelegt. Hierbei können, wie das Beispiel zeigt, Attribute wie *name* und *phrases* plattformunabhängig definiert werden. Der input type muss plattformabhängig angegeben werden, sofern es sich um einen vordefinierten input type handelt, wie in dem Falle eines beliebigen Textes bzw. einer Suchanfrage (*AMAZON.SearchQuery* bzw. *@sys.any*). [42]

Es ist hier auch möglich bestimmte inputs als *required* zu setzten, was das System beim Aufrufen des intents ohne einen Wert dieses inputs automatisch nach diesem fragen lässt. Dies unterscheidet sich strukturell jedoch zwischen den Plattformen, und muss somit individuell für die entsprechende Plattform umgesetzt werden.

Obiges Beispiel zeigt lediglich die Umsetzung der required inputs für Dialogflow, da Nachfragen der input Werte bei Alexa mit dem sogenannten interaction model separat behandelt werden und demnach nicht im intent definiert werden. Dieses interaction model kann jedoch in einem extra dafür angelegten Objekt des Jovo language models angelegt werden [39]. Somit befindet sich das komplette language model samt plattformspezifischen Informationen in einem einzigen JSON Dokument und kann von dort verwaltet werden.

7.3 Handler

Die dem intent zugehörigen Prozesse, wie das Bearbeiten von Daten, Abfragen und Modifikationen auf dem Server oder das Erzeugen der Antwort des intents findet bei Jovo in den sogenannten handlers statt, welche Funktionen beinhalten. Intents werden diesen Funktionen per Namen zugewiesen [46]. Wird also der intent *AddNoteToLastAppointmentIntent* erkannt, wird die Funktion mit selbigem Namen im handler aufgerufen.

Es besteht darüber hinaus die Möglichkeit intents im *AlexaHandler* bzw. *GoogleAssistantHandler* zu implementieren, falls die auszuführende Logik zwischen den Plattformen unterschiedlich ist [46]. Ein Beispiel hierfür, aus dem hier entstandenen skill, ist der intent *CreateAppointmentIntent*, welcher

aufgrund von des Füllens von inputs individuell implementiert werden musste (siehe 7.7).

Um eine übersichtliche Struktur zu gewährleisten, wurde der cross-platform handler zusätzlich noch semantisch aufgeteilt. In diesem Zuge wurden für jedes betrachtete Datenobjekt (siehe 6.1) ein eigener handler erzeugt (*appointmentHandler*, *salesQuoteHandler* und *accountHandler*). Auch umfangreiche states, wie etwa den *AppointmentsState*, wurden in eigene Dateien ausgelagert und entsprechend der Struktur importiert. Allgemeine, den anderen handlers nicht zuzuordnende, intents werden von dem *standardHandler* bearbeitet.

7.4 States und Routing

Wie bereits in der Analyse (siehe 4.2) erwähnt, verfügt Jovo über eine state machine, welche es ermöglicht auf den gleichen intent abhängig von dem state unterschiedlich zu reagieren. Dies ist beispielsweise bei dem intent *GetLocationIntent* nötig, welchem unter anderem die phrase "Where is it" zugewiesen ist. Je nachdem in welchem Kontext (bzw. state) der Nutzer diese Frage stellt, kann die location eines appointments oder die address eines accounts gemeint sein. Aus diesem Grund wurden die states aus dem Konzept (siehe 6) in gleicher Struktur in den handler übernommen. Die praktische Umsetzung zeigt das folgende reduzierte Beispiel:

```
1  app.setHandler({
2    AccountState: {
3      GetLocationIntent(){
4          // Ausgabe account.address
5      }
6    },
7    AppointmentsState: {
8      AppointmentState: {
9        GetLocationIntent(){
10         // Ausgabe appointment.location
11       }
12     }
```

```
13    }
14  });
```

Analog zu diesem Beispiel wird gemäß der Anforderungen (siehe 5.7) für jeden State eine *HelpIntent*-Funktion implementiert, welche kontextuelle Hilfe ausgibt.

Innerhalb der Funktionen ist es beliebig möglich, den state zu manipulieren. Dementsprechend wird beispielsweise nach der Abfrage von appointments immer in den *appointmentsState* geleitet. Darüber hinaus ist es auch möglich innerhalb einer Anfrage zu einem anderen intent zu *routen*. Dies ermöglicht beispielsweise eine einfache Implementierung der horizontalen Struktur (siehe 3.2.3) indem *shortcut intents* erstellt werden, welche zu dem primären intent leiten. Für das erstellen einer note zu einem appointment sieht dieses Verfahren exemplarisch wie folgt aus:

```
1  app.setHandler({
2    AddNoteToLastAppointmentIntent(){
3      // ...
4      this.toStateIntent(
5          'AppointmentsState.AppointmentState',
6          'AddNoteIntent'
7      );
8    },
9    AppointmentsState: {
10     AddNoteToSpecificAppointmentIntent(){
11         // ...
12         this.toStateIntent(
13             'AppointmentsState.AppointmentState',
14             'AddNoteIntent'
15         );
16     },
17     AppointmentState: {
18       AddNoteIntent(){
19         // Auslesen und eintragen der eingegebenen Note
20       }
21     }
22   }
```

```
23 });
```

Es ist jedoch zu beachten, dass dieses routing sich auf dem Jovo back end abspielt. Daraus folgt, dass automatische Nachfragen (siehe 7.2) durch die Plattform des gerouteten (primären) intents übersprungen werden. Hat *AddNoteIntent* also einen als required gesetzten input mit dem Namen *note* und wird beim direkten Aufruf dieses intents automatisch erfragt, müssen auch *AddNoteToLastAppointmentIntent* und *AddNoteToSpecificAppointmentIntent* diesen input als required gesetzt haben.

7.5 Model

Prozesse und Informationen bezüglich der Datenobjekte der SAP Cloud for Customer werden innerhalb des Jovo skills in models gebündelt. Diese models (*Appointment*, *Account*,...) können Informationen von der Schnittstelle (siehe 7.1) abfragen, diese importieren und (wenn implementiert) manipulieren bzw. Daten mit der SAP Cloud for Customer synchronisieren. Sie referenzieren sich gemäß des Konzepts gegenseitig (siehe 6.1), was das wechseln der states z.B. eines appointments zu dessen sales quotes ermöglicht. Models werden innerhalb der handler verwendet und werden in der session (siehe 2.1.6) gespeichert, um sie über mehrere Anfragen hinweg adressieren zu können.

7.6 Output

Ähnlich wie bei nach Model View Controller basierten graphischen Anwendungen, wird auch für diesen skill die Ausgabe strukturell abgegrenzt von dem handler (Controller) definiert, was den Vorteil der einfachen Modifikation oder Austausch der einzelnen Komponenten ermöglicht. Hilfreich ist dies beispielsweise bei für die Unterstützung verschiedener Sprachen. Jovo Applikationen nutzen hierfür nativ *i18next* [41], um den Inhalt von der Logik zu trennen.

Dies wird über ein JSON Dokument verwaltet, welches beliebig strukturiert werden kann und die auszugebenden Werte mit Variablen und evtl. mehreren Formulierungen einer auszugebenden Information enthält. Um auch hier Texte des gleichen intents in unterschiedlichen Kontexten individuell definieren zu können und eine grundsätzliche Struktur beizubehalten, ist die Struktur des Dokuments denen der handler nachempfunden. Innerhalb des handlers gibt es also die Möglichkeit, auf die Ausgaben des aktuellen intents mitsamt dessen states zuzugreifen. Folgendes Beispiel zeigt exemplarisch die Struktur:

```
"appointment": "appointment",
"appointment_plural": "appointments",
"GetAppointmentsIntent": {
  "speech": "You have got {{count}} $t(appointment) for
    <say-as interpret-as=\"date\" format=\"mdy\">
    {{day}}</say-as>!
    The subjects are: {{subjects}}!"
},
"AppointmentsState": {
    "AppointmentState": {
        "SaveIntent": {
            "speech": ["Okay, I saved that appointment",
                    "Saved that"]
        }
    }
}
```

Die Ausgabe von *GetAppointmentsIntent* zeigt, wie mit *count* oder *day* eine Variable verwendet werden kann, welche im handler übergeben wird. Auch zeigt es die Möglichkeit der Verlinkung auf andere Werte ($t(appointment)) welches automatisch abhängig von *count* Singular oder Plural von *appointment* in den Text einfügt. SSML (siehe 2.2) wird hierbei verwendet, um eine korrekte Ausgabe des Datums sicherzustellen. Das *speech*-Attribut des hier abgebildeten *SaveIntent* zeigt zwei verschiedene Variationen von phrases, von welchen automatisch innerhalb des handlers ein zufälliges gewählt wird, um die Interaktion abwechslungsreicher und natürlicher zu gestalten (siehe 3.2.4).

Zusätzlich zu dieser Struktur der intents mit entsprechenden Ausgaben, gibt es weitere Attribute mit Texten, welche über einen einzelnen intent hinaus relevant sind, wie etwa Möglichkeiten, die der Nutzer in einem gewissen state hat. Diese Optionen bzw. Hinweise, was ein Nutzer zu tun hat, werden entsprechend von Nutzerdaten ausgegeben. Wenn ein Nutzer den skill also zum ersten mal nutzt, bekommt dieser dadurch mehr Hilfestellung, während ein erfahrener Nutzer diese nicht mehr explizit ausgegeben bekommt, womit die Anpassungsfähigkeit des skills gesteigert wird (siehe 3.1.1).

7.7 Herausforderungen

Während die Implementierung mit Jovo für Amazon Alexa und Google assistant für die meisten use cases und Features problemlos plattformunabhängig umzusetzen war, traten an einigen Stellen trotzdem Herausforderungen auf. Ein Beispiel hierfür ist das Aufrufen der Nachfrage fehlender Inputs eines intents innerhalb des handlers auf dem back end der Plattform, welches beispielsweise für das Erstellen eines appointments benötigt wird. Jovo bietet hierfür lediglich für Alexa nativ die Funktionalität an, dies zu erledigen [39].

Ein zum Alexa dialog interface ähnliches Feature wird auch bei Dialogflow angeboten [28], ist aber zum aktuellen Zeitpunkt nicht in Jovo integriert. Deshalb wurde das handling für diesen intent für Alexa mit dem dialog interface und für Dialogflow mit eigens dafür angelegten states und durch Jovo erzeugten Nachfragen erstellt. Laut Aussagen der Jovo Entwickler [30], könnte eine abstrakte Möglichkeit des automatischen Füllens von inputs in zukünftigen Versionen des Frameworks implementiert werden.

Des weiteren kam es bei Tests innerhalb der Entwicklungsphase teilweise zu falsch zugewiesenen intents (siehe 3.1.3). Dies kam besonders häufig bei intents mit beliebigen Text inputs vor, wie etwa *SelectQueryIntent* mit welchem man mit einer phrase wie “tell me about {query}” nach einem beliebigen appointment oder sales quote fragen kann. Wenn nun der intent *GetDetailIntent* unter anderem die phrase “tell me about {detail}” hat, und der Nutzer diesen mit einer in den dessen phrases nicht vorzufinden

leicht unterschiedlichen Variation davon (z.B. "Tell me about location of this") aufrufen möchte, kann dies zu einer Zuordnung zu *SelectQueryIntent* führen.

Um dem entgegenzuwirken wurde für intents mit hoher Verwechslungsgefahr besonders darauf Wert gelegt, viele Variationen von phrases zu definieren, und phrases mit beliebigen Texten nicht zu generisch zu gestalten, um so falsche Zuordnungen zu vermeiden. Eine weitere Möglichkeit fehlerhafte Zuweisungen zu minimieren wäre die kontextuelle Einschränkung der erkennbaren intents auf der Ebene der Intentzuweisung. Mit Hilfe von states ist bereits die kontextuelle Behandlung von intents möglich, jedoch geschieht dies erst nachdem das Gesagte einem eventuell falschem intent zugewiesen wurde. Dialogflow ermöglicht es mit Hilfe von *input* und *output contexts* diese Einschränkung auf Intenterkennungsebene umzusetzen [24].

Amazon Alexa skills hingegen haben ein inhärent stateless Design [17], womit bei der Intentzuweisung keine Einschränkungen gemacht werden können und zum aktuellen Zeitpunkt das in Betracht ziehen von Kontext bzw. state lediglich innerhalb der handler möglich ist. Dieses Designprinzip entspricht dem Ziel des horizontal aufgebauten immer erreichbaren skills (siehe 3.2.3), während Dialogflow eine eventuell eher eingeschränkte bzw. geleitete aber robustere Dialogführung ermöglicht. Aus dem Grund und dem Ziel möglichst wenig Features plattformabhängig umzusetzen, wurde für diesen skill der Ansatz von Amazon Alexa für beide Plattformen umgesetzt.

Da der skill es dem Nutzer ermöglichen sollte beispielsweise nach accounts zu fragen, oder einen der SAP Cloud for Customer vermerkten contact (Nutzer) einem appointment als participant zuzuweisen, ist eine Erkennung dieser contacts und accounts zu implementieren. Ein Ansatz hierfür ist das Verwenden von vordefinierten input types der Plattformen, wie etwa einem beliebigen Text, um einen account zu suchen, oder inputs für Namen, um einen contact zu finden. Diese sind jedoch nicht speziell auf die Daten der SAP Cloud for Customer angepasst, können also auch mit Werten gefüllt werden, welche dort nicht zu finden sind.

Alternativ kann hierfür auch ein selbst definierter input type erstellt werden, welcher mit den möglichen Werten der Firmennamen bzw. Kontaktnamen

befüllt ist, und somit bereits bei dem Füllen des inputs nur valide Werte annehmen kann und die Erkennung durch Einschränkung der Möglichkeiten robuster und genauer wird.

Angesichts der Tatsache, dass accounts und contacts innerhalb der Cloud for Customer über die Zeit gesehen größtenteils konstant bleiben, wurde letztere Möglichkeit gewählt. Dafür wurde ein Skript geschrieben, welche die Werte aus der SAP Cloud for Customer bezieht und so formatiert, dass es in das Jovo language model (siehe 7.2) eingefügt werden kann. Da die input Werte für das Training des back ends der Plattformen bereits fest stehen müssen, muss bei Veränderung dieser Daten darauf geachtet werden, dass das language model mit Hilfe des Skripts erneut angepasst werden muss, und der deployment Prozess für die back ends der jeweiligen Plattformen auch wiederholt werden muss.

8 Zusammenfassung

Aus der Retrospektive im Anschluss der praktischen Umsetzung mit vorhergehender Recherche und Analyse ergeben sich Schlussfolgerungen und ein Ausblick für die Zukunft. Das beinhaltet zum sowohl eine Evaluation der in der Analyse und Implementierung getroffenen Entscheidungen, sowie einen konkreten Vorschlag für die Weiterentwicklung des entstandenen skills in weiteren Arbeiten.

8.1 Fazit

Ausgehend von dem Wunsch, die SAP Cloud for Customer für Situationen, in denen eine Interaktion mit einer graphischen Oberfläche nicht möglich ist, durch das Bereitstellen der Sprachinteraktion zu erweitern, wurde im Kontext dieser Arbeit ein skill entwickelt. Um ein nutzerfreundliches Design dieses neuen user interfaces gewährleisten zu können, wurden innerhalb der verwandten Arbeiten Herausforderungen und sich daraus herleitende Designheuristiken erarbeitet.

Unter anderem anhand der Erkenntnisse daraus, und zusätzlichen Aspekten wie Verbreitung verschiedener voice assistant Plattformen, wurden innerhalb der Analyse (siehe 4) Entscheidungen bezüglich Plattformen und des zu verwendenden Frameworks getroffen. Hierbei wurde eine Implementierung einer cross-platform Anwendung mit dem Jovo Framework gewählt. Mit Hilfe von Interviews mit Vertriebsmitarbeiter wurde darüber hinaus der anwendungsspezifische Aspekt der Arbeit untersucht und ein Konzept basierend auf den in den Interviews herausgearbeiteten Schwerpunkten erstellt. Dieses Konzept wurde innerhalb eines weiteren Gesprächs evaluiert und angepasst.

Innerhalb der Implementierung traten Herausforderungen auf, welche bereits in den verwandten Arbeiten (siehe 3) erarbeitet wurden, wie die Schwierigkeiten innerhalb der Intentzuweisung. Darüber hinaus traten Probleme technologischer Natur auf, was dazu führte, dass einzelne Funktionen trotz cross-platform Framework plattformspezifisch umgesetzt werden mussten. Im Hinblick darauf, dass sich der Großteil der Funktionalitäten mit Hilfe von Jovo plattformunabhängig umsetzen ließen, ist die Wahl des Frameworks in Retrospektive jedoch gerechtfertigt.

8.2 Ausblick

Wie innerhalb der Definition der use cases (siehe 5.5) und dem Konzept (siehe 6) bereits erwähnt, bildet der hier entstandene skill lediglich einen Bruchteil der in der SAP Cloud for Customer angebotenen Funktionen und Informationen ab. Es war Ziel der Arbeit hiervon die relevantesten herauszuarbeiten und diese in einer Struktur zu implementieren, welche die Erweiterung des Systems mit weiteren Funktionen und Datenobjekten ermöglicht und begünstigt, also eine Skalierbarkeit sicherstellt. Im Hinblick darauf würde es sich anbieten inkrementell weitere Datenobjekte hinzuzufügen, intents, states und Verlinkungen zu bestehenden Datenobjekten herzustellen, und somit weitere use cases umzusetzen, bis der skill eine komplette Schnittstelle der SAP Cloud for Customer darstellt.

Eine tiefergehende Evaluation des skills und dessen Konzepts durch weitere Interviews und Tests mit Angestellten aus dem Vertriebsbereich wäre als weiterer Schritt zudem denkbar. Auch technologischer Aspekte, wie etwa die Authentifizierung an der OData Schnittstelle der SAP Cloud for Customer oder die Optimierung der Performanz, welche im Hinblick auf den Schwerpunkt der Arbeit nicht umfassend behandelt wurden, sind für die zukünftige Weiterentwicklung des skills in Betracht zu ziehen.

Literatur

[1] Amazon. *Alexa Design Guide*. URL: https://developer.amazon.com/docs/alexa-design/get-started.html (besucht am 02. 06. 2019) (siehe S. 16).

[2] Amazon. *Alexa Design Guide - Be Adaptable*. URL: https://developer.amazon.com/docs/alexa-design/adaptable.html (besucht am 08. 06. 2019) (siehe S. 17).

[3] Amazon. *Alexa Design Guide - Be Available*. URL: https://developer.amazon.com/docs/alexa-design/available.html (besucht am 08. 06. 2019) (siehe S. 20).

[4] Amazon. *Alexa Design Guide - Be Personal*. URL: https://developer.amazon.com/docs/alexa-design/personal.html (besucht am 08. 06. 2019) (siehe S. 18).

[5] Amazon. *Alexa Design Guide - Be Relatable*. URL: https://developer.amazon.com/docs/alexa-design/relatable.html (besucht am 08. 06. 2019) (siehe S. 22).

[6] Amazon. *Alexa Skill Shop*. URL: https://www.amazon.de/b?ie=UTF8&node=10068460031 (besucht am 26. 05. 2019) (siehe S. 4).

[7] Amazon. *Alexa Skills Kit Glossary - Intents*. URL: https://developer.amazon.com/docs/ask-overviews/alexa-skills-kit-glossary.html#intent (besucht am 26. 05. 2019) (siehe S. 4).

[8] Amazon. *Alexa Skills Kit Glossary - Required Slot*. URL: https://developer.amazon.com/docs/ask-overviews/alexa-skills-kit-glossary.html#required-slot (besucht am 27. 05. 2019) (siehe S. 6).

[9] Amazon. *Alexa Skills Kit Glossary - Skill*. URL: https://developer.amazon.com/docs/ask-overviews/alexa-skills-kit-glossary.html#skill (besucht am 26. 05. 2019) (siehe S. 4).

[10] Amazon. *Alexa Skills Kit Glossary - Slot*. URL: https://developer.amazon.com/docs/ask-overviews/alexa-skills-kit-glossary.html#slot (besucht am 27. 05. 2019) (siehe S. 5).

[11] Amazon. *Alexa Skills Kit Glossary - Utterance*. URL: https://developer.amazon.com/docs/ask-overviews/alexa-skills-kit-glossary.html#utterance (besucht am 26. 05. 2019) (siehe S. 4).

[12] Amazon. *Alexa Skills Kit SDKs*. URL: https://developer.amazon.com/de/docs/sdk/alexa-skills-kit-sdks.html (besucht am 20. 08. 2019) (siehe S. 26).

[13] Amazon. *Manage the Skill Session and Session Attributes*. URL: https://developer.amazon.com/docs/custom-skills/manage-skill-session-and-session-attributes.html (besucht am 10. 06. 2019) (siehe S. 8).

[14] Amazon. *Request and Response JSON Reference - Response Object*. URL: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#response-object (besucht am 28. 05. 2019) (siehe S. 7).

[15] Amazon. *Slot Type Reference*. URL: https://developer.amazon.com/docs/custom-skills/slot-type-reference.html (besucht am 27. 05. 2019) (siehe S. 6, 25).

[16] Amazon. *Speech Synthesis Markup Language (SSML) Reference*. URL: https://developer.amazon.com/de/docs/custom-skills/speech-synthesis-markup-language-ssml-reference.html (besucht am 28. 05. 2019) (siehe S. 8, 9).

[17] Amazon. *Tips on State Management at Three Different Levels*. URL: https://developer.amazon.com/blogs/alexa/post/648c46a1-b491-49bc-902d-d05ecf5c65b4/tips-on-state-management-at-three-different-levels (besucht am 27. 08. 2019) (siehe S. 67).

[18] Strategy Analytics. *2018 Global Smart Speaker Sales Reached 86.2 Million Units on Back of Record Q4*. URL: https://news.strategyanalytics.com/press-release/devices/strategy-analytics-2018-global-smart-speaker-sales-reached-862-million-units (besucht am 30. 06. 2019) (siehe S. 1, 23).

[19] Apple. *App Store - Amazon Alexa*. URL: https://apps.apple.com/de/app/amazon-alexa/id944011620 (besucht am 30. 06. 2019) (siehe S. 24).

[20] Apple. *App Store - Google Assistant*. URL: https://apps.apple.com/de/app/google-assistant/id1220976145 (besucht am 30. 06. 2019) (siehe S. 24).

[21] Alice Coucke u. a. »Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces«. In: *CoRR* abs/1805.10190 (2018). arXiv: 1805.10190. URL: http://arxiv.org/abs/1805.10190 (siehe S. 24).

[22] Dialogflow. *Agents overview*. URL: https://dialogflow.com/docs/agents (besucht am 26. 05. 2019) (siehe S. 4).

[23] Dialogflow. *Entities Overview*. URL: https://dialogflow.com/docs/entities (besucht am 27. 05. 2019) (siehe S. 5).

[24] Dialogflow. *Input and output contexts*. URL: https://cloud.google.com/dialogflow/docs/contexts-input-output (besucht am 27. 08. 2019) (siehe S. 67).

[25] Dialogflow. *Slot Filling*. URL: https://dialogflow.com/docs/concepts/slot-filling (besucht am 27. 05. 2019) (siehe S. 6).

[26] Dialogflow. *System Entities*. URL: https://cloud.google.com/dialogflow-enterprise/docs/reference/system-entities (besucht am 27. 05. 2019) (siehe S. 6, 25).

[27] Dialogflow. *Training Phrases*. URL: https://dialogflow.com/docs/intents/training-phrases (besucht am 26. 05. 2019) (siehe S. 5).

[28] Dialogflow. *Webhook for slot filling*. URL: https://cloud.google.com/dialogflow/docs/fulfillment-webhook-slot-filling (besucht am 27. 08. 2019) (siehe S. 66).

[29] GitHub. *Mycroft*. URL: https://github.com/MycroftAI (siehe S. 25).

[30] Github. *jovo-framework - An abstraction layer for slot filling*. URL: https://github.com/jovotech/jovo-framework/issues/160 (siehe S. 66).

[31] Github. *SAP Cloud for Customer OData API Developer's Guide*. URL: https://github.com/SAP/C4CODATAAPIDEVGUIDE (siehe S. 59, 60).

[32] Google. *Assistant directory*. URL: https://developers.google.com/actions/distribute/directory (besucht am 26. 05. 2019) (siehe S. 4).

[33] Google. *Google Assistant auf dem Smartphone oder Tablet verwenden*. URL: https://support.google.com/assistant/answer/7172657?co=GENIE.Platform%5C%3DAndroid&hl=de (siehe S. 24).

[34] Google. *Google Play - Amazon Alexa*. URL: https://play.google.com/store/apps/details?id=com.amazon.dee.app (siehe S. 24).

[35] Google. *Google Play - Google Assistant*. URL: https://play.google.com/store/apps/details?id=com.google.android.apps.googleassistant&hl=de (siehe S. 24).

[36] Google. *SSML*. URL: https://developers.google.com/actions/reference/ssml (siehe S. 8).

[37] ISO. *ISO/IEC 25010:2011*. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en (besucht am 21. 06. 2019) (siehe S. 41).

[38] Jovo. *Amazon Alexa - Visual Output*. URL: https://www.jovo.tech/docs/amazon-alexa/visual-output (siehe S. 27).

[39] Jovo. *Dialog Interface*. URL: https://www.jovo.tech/docs/amazon-alexa/dialog-interface (siehe S. 61, 66).

[40] Jovo. *Google Assistant - Visual Output*. URL: https://www.jovo.tech/docs/google-assistant/visual-output (siehe S. 27).

[41] Jovo. *i18n*. URL: https://www.jovo.tech/docs/output/i18n (siehe S. 64).

[42] Jovo. *Jovo Language Model*. URL: https://www.jovo.tech/docs/model#jovo-language-model (siehe S. 61).

[43] Jovo. *Jovo-framework - An abstraction layer for slot filling*. URL: https://www.jovo.tech/features (siehe S. 26).

[44] Jovo. *Platforms*. URL: https://www.jovo.tech/docs/platforms (siehe S. 26).

[45] Jovo. *Reminders API*. URL: https://www.jovo.tech/docs/amazon-alexa/reminders (siehe S. 27).

[46] Jovo. *Routing*. URL: https://www.jovo.tech/docs/routing (siehe S. 61).

[47] Jovo. *The Framework for Voice App Development*. URL: https://www.jovo.tech (siehe S. 26).

[48] Bret Kinsella. *Amazon Increases Global Smart Speaker Sales Share in Q4 2018, While Google's Rise Narrows the Gap and Apple Declines*. Hrsg. von voicebot.ai. URL: https://voicebot.ai/2019/02/20/amazon-increases-global-smart-speaker-sales-share-in-q4-2018-while-googles-rise-narrows-the-gap-and-apple-declines/ (besucht am 30. 06. 2019) (siehe S. 23).

[49] Christian Moser. »User Experience Design«. In: *User Experience Design: Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 1–22. ISBN: 978-3-642-13363-3. DOI: 10.1007/978-3-642-13363-3_1. URL: https://doi.org/10.1007/978-3-642-13363-3_1 (siehe S. 32).

[50] Mycroft. *Mycroft for Android*. URL: https://mycroft.ai/documentation/android/ (siehe S. 24).

[51] Mycroft. *mycroft.util package*. URL: https://mycroft-core.readthedocs.io/en/stable/source/mycroft.util.html# (siehe S. 25).

[52] Mycroft. *Usability vs. Privacy*. URL: https://mycroft.ai/blog/usability-vs-privacy-keeping-things-in-balance/ (siehe S. 25).

[53] Mycroft. *Which Mycroft platform is best for me?* URL: https://mycroft.ai/get-mycroft/ (siehe S. 24).

[54] SAP. *Accounts and Individual Customers*. URL: https://help.sap.com/viewer/24765b551a014b779b95c7b07d8e9079/1905/en-US/3100774df47a450983233f2615693452.html (siehe S. 11).

[55] SAP. *Activities*. URL: https://help.sap.com/viewer/24765b551a014b779b95c7b07d8e9079/1905/en-US/2653c85bb18e46d3a178bf3c9b1814fd.html (siehe S. 11).

[56] SAP. *Leads*. URL: https://help.sap.com/viewer/24765b551a014b779b95c7b07d8e9079/1905/en-US/1b85145f32ba4f7584f45886efac803b.html (siehe S. 10).

[57] SAP. *Opportunities*. URL: https://help.sap.com/viewer/24765b551a014b779b95c7b07d8e9079/1905/en-US/ce6be44fedd447bbbe013def2262eca5.html (siehe S. 10).

[58] SAP. *Sales Quotes*. URL: https://help.sap.com/viewer/24765b551a014b779b95c7b07d8e9079/1905/en-US/b8df192cb7aa420d8e1f9bb7eeb88782.html (siehe S. 11).

[59] SAP. *SAP Cloud for Customer*. URL: https://www.sap.com/germany/products/cloud-customer-engagement.html (siehe S. 10).

[60] SAP. *SAP Hybris Cloud for Customer OData API*. URL: https://help.sap.com/viewer/26fdb8fadd5b4becb5c858d92146d0e0/1708/en-US/6c0a463cc9ca450cbd01a9a5057ce682.html (siehe S. 59).

[61] Ben Shneiderman u. a. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 6th. Pearson, 2016, S. 311–335. ISBN: 013438038X, 9780134380384 (siehe S. 13–15).

[62] Snips. *Slot Types*. URL: https://docs.snips.ai/ (besucht am 30. 06. 2019) (siehe S. 25).

[63] Snips. *Snips Platform*. URL: https://docs.snips.ai/articles/platform/dialog/slot-types (besucht am 19. 08. 2019) (siehe S. 24).

[64] Smruthi Sridhar und Matthew E Tolentino. »Evaluating Voice Interaction Pipelines at the Edge«. In: *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE. 2017, S. 248–251 (siehe S. 24).

[65] W3C. *Speech Synthesis Markup Language (SSML) Version 1.1*. URL: https://www.w3.org/TR/speech-synthesis11/#S3.1.8.1 (besucht am 28. 05. 2019) (siehe S. 8).