



Projektteam

Bresemmler, Eduard

Cavallaro, Angelo

Gröne, Adrian

Kasarca, Hüseyin

Kinzelmann, Daniel

Inhaltsverzeichnis

Abkürzungsverzeichnis	III
Abbildungsverzeichnis	IV
Tabellenverzeichnis	V
1 Vorstellung des Projekt-Teams	1
2 SAPlexa – Die betriebliche Sprachassistentz für SAP	2
2.1 Beschreibung von SAPlexa	2
2.2 Technische Anforderungen an SAPlexa	2
2.3 Einsatzbereich der Applikation	3
3 Organisation des Projekts	4
3.1 Projektbezogene Anwendung von SCRUM	4
3.2 Zeitmanagement	7
4 Meilensteine der (agilen) Softwareentwicklung	9
4.1 Konzeption des Front-Ends	9
4.1.1 Entwurf der Menüführung	10
4.1.2 Ergonomie und Erprobung von Schlüsselbegriffen	15
4.1.3 Informationsbeschaffung bei der Groz-Beckert KG	19
4.2 Konzeption des Back-Ends	21
4.2.1 Java-Perspektive	21
4.2.2 SAP-Perspektive	25
4.2.3 SAP Java Connector – Die Schnittstelle	32
5 Zukünftige Optimierungs- und Erweiterungsmöglichkeiten	34
6 Reflexion und Fazit	35
Literaturverzeichnis	36
Anhang (Quellcodes)	37

Abkürzungsverzeichnis

BAPI	Business Application Programming Interface
BEDAT	Komponententyp: Bestelldatum
CHAR	Datentyp: Zeichenfolge
DATS	Datentyp: Datum im Format JJJJMMDD
EBELN	Komponententyp: Einkaufsbelegnummer
EBELP	Komponententyp: Einkaufsbelegposition
EKET	SAP-Datenbanktabelle: Lieferplaneinteilung
EKKO	SAP-Datenbanktabelle: Einkaufsbelegkopf
EKPO	SAP-Datenbanktabelle: Einkaufsbelegposition
i.d.R.	in der Regel
LGORT	Komponententyp: Lagerort
LIFNR	Komponententyp: Lieferantennummer
MATNR	Komponententyp: Materialnummer
MENGE	Komponententyp: Bestellmenge zur Position
NUMC	Datentyp: Numerischer Text
o.g.	oben genannt
QUAN	Datentyp: Mengenfeld
TXZ01	Komponententyp: Material – Beschreibung
WEMNG	Komponententyp: Wareneingangsmenge (Gelieferte Menge)

Abbildungsverzeichnis

Abbildung 4.1 Menüführung als Ablauflogik.....	10
Abbildung 4.2 Hauptmenü SAPlexa	11
Abbildung 4.3: Hauptmenü SAPlexa - Bestellnummer übergeben	12
Abbildung 4.4: MIGO-Übersicht SAPlexa	12
Abbildung 4.5: MIGO-Übersicht SAPlexa - Bild anzeigen.....	13
Abbildung 4.6: MIGO-Übersicht SAPlexa – Materialbeleg erzeugen	14
Abbildung 4.7 Kardinalitäten zwischen EKKO, EKPO und EKET.....	26
Abbildung 4.8 Tabellentypen - Strukturen - Datentypen.....	27
Abbildung 4.9 Quellcode ZE268_GETPROPOSALLIST	31
Abbildung 4.10 Java-seitiger Funktionsaufruf auf den Baustein ZE268_GETPROPOSALLIST ..	32

Tabellenverzeichnis

Tabelle 5.1 Relevante Datenbanktabellen in HANA.....	25
Tabelle 5.3 Definierte RFC-Funktionsbausteine in SAP	30

1 Vorstellung des Projekt-Teams

Für das Projekt, welches im Rahmen des Moduls „Projektstudium“ im 5. Semester in der Fakultät Informatik vorgesehen ist, wurde die Auswahl der Projektmitglieder auf Studenten gelegt, die aus unterschiedlichen Studiengängen kommen. Damit verteilten sich die Aufgabengebiete nach Spezialisierung und Fachkenntnis, wie im Folgenden aufgezeigt.

Name	Studiengang	Schwerpunkt im Projekt
Bresemmler, Eduard	Technische Informatik	Kommunikation und Organisation SCRUM-Master
Cavallaro, Angelo	Wirtschaftsinformatik	SAP ABAP
Gröne, Adrian	IT-Security	Spracherkennung
Kasarca, Hüseyin	Wirtschaftsinformatik	JAVA GUI
Kinzelmann, Daniel	IT-Security	Spracherkennung

Das Projekt-Team behandelte das Thema SAPlexa als erste Projektgruppe. SAPlexa wurde somit ohne jegliche vorherigen Vorentwicklungsarbeiten von Grund auf neu konzipiert und implementiert. Daraus ergaben sich Freiheiten in der Implementierung, die allerdings wider Erwarten nicht in jedem Fall einen Vorteil darstellten.

Für die Bearbeitung der gestellten Projektaufgabe war es demnach wichtig, strukturiert vorzugehen und die Anforderungen, sowohl technischer, als auch organisatorischer Art zu kennen. Nach diesem Prinzip soll diese Ausarbeitung ebenfalls arbeiten. Die grundlegenden Anforderungen und Ziele des Projekts sollen daher im nächsten Schritt näher beleuchtet werden.

2 SAPlexa – Die betriebliche Sprachassistentin für SAP

2.1 Beschreibung von SAPlexa

SAPlexa - die sprachgesteuerte Applikation, die SAP unterstützt.

Im Rahmen des Projektstudiums wurde die Aufgabe gestellt eine sprachgesteuerte Applikation zu entwickeln, die mit SAP kompatibel ist und kommunizieren kann. Daher der Name SAPlexa. Dieser setzt sich aus SAP und „Alexa“, der Sprachsteuerung von Amazon, zusammen.

SAPlexa soll den Lagermitarbeitern in aller erster Linie durch die Sprachsteuerung eine Erleichterung beim Wareneingang bringen. Durch diese Applikation muss der Lagerist nicht mit schmutzigen Fingern am Display per Touchscreen die Eingabe vornehmen, sondern kann per einfachen Sprachkommandos die App bedienen und den Wareneingang verbuchen. Des Weiteren kann er alle relevanten Informationen über die eingegangenen Bestellungen einsehen und darüber verfügen.

2.2 Technische Anforderungen an SAPlexa

Die grundlegende Funktion der Sprachsteuerung ist, dass sie es dem Mitarbeiter ermöglicht mithilfe seiner Stimme Eingaben in SAP vorzunehmen. So soll das sich oft wiederholende Eintippen von Informationen vermieden werden.

Die Anforderungen an die Sprachsteuerung sind, dass sie die gesprochenen Schlüsselwörter (Keywords) des Mitarbeiters erkennt und Hintergrundgeräusche (in Maßen) nicht zu Fehlerkennungen führen. Die Benutzeroberfläche soll alle wichtigen Informationen darstellen, den Mitarbeiter jedoch nicht zu viele bzw. unnötige Informationen zeigen. Die Anforderung ist also nur die wirklich wichtigen Informationen anzuzeigen. Die Sprachsteuerung soll den Mitarbeiter entlasten und einfach zu bedienen sein, deshalb sollten die Schlüsselwörter einfach und möglichst selbsterklärend sein, um Missverständnisse vorzubeugen. Die Ergonomie der Schlüsselwörter und somit auch die Häufigkeit der notwendigen Spracheingabe sind ein wichtiger Teil der Implementierung.

2.3 Einsatzbereich der Applikation

Die Sprachsteuerung soll im Wareneingang eingesetzt werden. Dort kommen die bestellten Waren an, werden erfasst und weiterverarbeitet bzw. verbucht, z.B. in ein Lager. Der Grund, weshalb die Sprachsteuerung im Wareneingang eingesetzt wird, ist der, dass dort die eingegangene Ware im SAP System verbucht werden muss. Da bei größeren Firmen täglich mehrere hunderte Bestellungen eintreffen, müssen die Mitarbeiter sehr oft so ziemlich das Gleiche eintippen, was auf Dauer sehr erschöpfend, frustrierend und zeitaufwendig sein kann. Mithilfe der Sprachsteuerung soll dies vereinfacht werden, da nun die Mitarbeiter die Eingaben per Stimme erledigen können und der Buchungsvorgang übersichtlicher gestaltet wird.

3 Organisation des Projekts

3.1 Projektbezogene Anwendung von SCRUM

SCRUM ist ein Rahmenwerk zur agilen Produktentwicklung. Nach dem Original SCRUM-Guide von Ken Schwaber und Jeff Sutherland gibt es drei Rollen, fünf Ereignisse und drei Artefakte.

Rollen	
Product Owner	Er ist für das Produkt, das entstehen soll, verantwortlich.
SCRUM Master	Er ist für die Einhaltung der SCRUM-Regeln und die Optimierung des SCRUM zuständig.
Team	Entwicklungsteam; zuständig für das Erstellen des Produktes.

Ereignisse	
Sprint	Ein Zyklus, der immer gleichlang sein sollte und maximal 4 Wochen dauert.
Sprint-Planungssitzung	Sie findet immer am Anfang eines Sprints statt und legt die Sprint-Ziele fest.
Daily SCRUM	Ein tägliches Meeting von etwa 15 Minuten, um sich zu synchronisieren und die nächsten 24 Stunden zu planen.
Sprint Review	Es findet immer am Ende jedes Sprints statt und dient dazu das freizugebende Produktinkrement zu überprüfen.
Sprint-Retrospektive	Es findet immer am Ende jedes Sprints statt und dient dazu Maßnahmen zu beschließen zukünftige Sprints zu optimieren.

Artefakte	
Product Backlog	Es dient der Pflege der Anforderungen durch den Product Owner.
Sprint Backlog	Ein Plan des Entwicklungsteams, dass die Anforderungen im nächsten Sprint beschreiben.
Produktinkrement	Das einsatzfähige Produkt, das am Ende eines Sprints freigegeben wird.

Am Anfang des Projektstudiums haben wir uns darauf geeinigt den SCRUM auf unser Projekt anzuwenden. Folgende Rollen haben wir dabei im Kollektiv verteilt:

Rollen	
Product Owner und Projektleiter	Herr Professor Bernd Stauß
SCRUM Master	Bresemmler Eduard
Team	Cavallaro Angelo, Gröne Adrian, Kasarcia Hüseyin, Kinzelmann Daniel, Bresemmler Eduard

PRODUCT BACKLOG

- API Speech to Text
- API zur Selektion und Darstellung der offenen Positionen zur Bestellnummer
- Funktion Buchen Wareneingang zur Bestellposition
- Menüführung und Ergonomie

Unseren Sprint haben wir auf 3 Wochen festgesetzt, konnten es aber nicht immer einhalten. Einen Daily SCRUM haben wir nicht gehabt, da es durch die verschiedenen Vorlesungspläne nicht möglich war, stattdessen hatten wir jeden Dienstag einen Weekly SCRUM.

Im Folgenden sind unsere Termine mit den zusammengehörenden Informationen zu sehen:

Datum	Tätigkeit	Backlogliste
08.10.19	Allgemeine Vorstellung des Teams und des Projektes. Klärung wichtiger Fragen und Entscheidungen.	
15.10.19	Erklärung der Prozesse und Klärung von offenen Fragen. Backlogliste angefertigt	1: API Speech2Text funktionsfähig gestalten 2: API zur Selektion und Darstellung von offenen Positionen zur Bestellnummer 3: Funktion „Buchen Wareneingang“ zur Bestellposition
22.10.19	Überarbeitung der Backlogliste Wiederholung SAP (Rahmenbedingungen für SAPLEXA) Weekly SCRUM	1: Einheitliche Entwicklungsumgebung definieren 2: Auswahl der Speech2Text API 3: Realisierung API Speech2Text auf Englisch 4: Anwendungs-Framework GUI einstellen 5: Projekte lauffähig zusammenführen
29.10.19	Weekly SCRUM	

Datum	Tätigkeit	Backlogliste
05.11.19	Weekly SCRUM	
12.11.19	Sprint Review durch Präsentation Klärung von offenen Fragen Backlogliste erstellt Weekly SCRUM	1: Selektieren der Bestellung in SAP 2: Lauffähiger Java-Connector 3: Ergonomie 4: Vorbereitung auf die Exkursion
19.11.19	Erklärung und die ersten Schritte in ABAP Weekly SCRUM	
26.11.19	Exkursion zu Groz-Beckert KG	
03.12.19	Nachbesprechung der Exkursion Weekly SCRUM Exceptionhandling ABAP-Code für Prefix angefangen	
10.12.19	Sprint Review Backlogliste erstellt Weekly SCRUM Klärung offener Fragen	1: Buchung des Materialbelegs im ABAP 2: Liste an offenen Positionen zur selektierten Reihenfolge über Funktionsbaustein anzeigen 3: Materialbildidentifikation mit Vergrößerung in der Detailansicht 4: Eingabebereitschaft visuell darstellen, Rückmeldung vom Programm 5: Exceptionhandling 6: Mengenanpassung bei Teillieferung 7: Planung der Struktur und Inhalte der Präsentation
17.12.19	Weekly SCRUM Besprechung der PowerPoint-Präsentation für die Vorstellung des Projektes am 14.01.20 ABAP-Baustein für das Buchen angefangen	
07.01.20	Weekly SCRUM	

3.2 Zeitmanagement

Das Zeitmanagement im Team wurde nach der im SCRUM-Prinzip beschriebenen Vorgehensweise strukturiert und organisiert. Somit ergaben sich in nahezu regelmäßigen Abständen definierte Arbeitsblöcke, in denen ausgewählte Ziele hinsichtlich der Implementierung oder auch der Recherchearbeit verfolgt wurden. Auf Grundlage einer realistischen Aufwandsabschätzung anhand eines Punktesystems konnte die termingerechte Fertigstellung in den allermeisten Fällen garantiert werden. Die Vordefinition einer konkreten Zielerreichung (Definition-Of-Done) stellte ein zentrales Merkmal der Organisation dar.

Als Dead-Line für das Projekt lässt sich terminlich die Vorstellung der Projekte am Dienstag, den 14. Januar 2020, nennen. Eine funktionierende und zuverlässige Sprachassistentz zur erfolgreichen Buchung eines (fiktiven) Wareneingangs stellte für die Vorstellung des Projekts das Hauptziel dar. Als Startzeitpunkt für das Projekt lässt sich rückblickend der Dienstag, den 15. Oktober 2019, identifizieren. Der Zeitrahmen umfasst demnach ganze 14 Wochen.

Jahr	KW	Zeitraum
		Tätigkeit
2019	42	14. Oktober – 20. Oktober
		15. Oktober Sprint Meeting 1°: - Festlegung des Projektziels - Besprechung der Projekt-Organisation - Rollenverteilung
	43	21. Oktober – 27. Oktober
		22. Oktober Sprint Meeting 2°: - Prozedur einer Wareneingangsbuchung im SAP wiederholen - Besprechung von Beispielsanfragen für die Sprachassistentz - Vorentscheidungen und Recherche
	44	28. Oktober – 03. November
		Recherche und Auswahl einer S2T API Implementierungs- und Testphase
	45	04. November – 10. November
		Implementierungs- und Testphase
	46	11. November – 17. November
		12. November Sprint Meeting 3°: - Feststellung der Zielerreichung (Review) - Festlegung weiterer Projektziele im Backlog
	47	18. November – 24. November
		Vorbereitung zur Exkursion bei Groz-Beckert Implementierungs- und Testphase

Jahr	KW	Zeitraum
		Tätigkeit
	48	25. November – 01. Dezember
		26. November Informationsbeschaffung: Exkursion bei Groz-Beckert KG
	49	02. Dezember – 08. Dezember
		03. Dezember Internes Team-Meeting: Nachbereitung der gewonnen Erkenntnisse aus der Exkursion
	50	09. Dezember – 15. Dezember
		10. Dezember Sprint Meeting 4°: - Besprechung von Verbesserungspotenzialen der GUI und UX - Beschränkung der visualisierten GUI-Inhalte aus Relevanzgründen
	51	16. Dezember – 22. Dezember
		Auswahl und Erprobung optimierter Keywords Implementierungs- und Testphase
	52	23. Dezember – 29. Dezember
		Weihnachtsferien Implementierungs- und Testphase
2020	1	30. Dezember – 05. Januar
		Weihnachtsferien Implementierungs- und Testphase
	2	06. Januar – 12. Januar
		Besprechung der Präsentationsinhalte der kommenden Projekt- Vorstellung Finale Implementierungs- und Testphase
	3	13. Januar – 19. Januar
		14. Januar Dead-Line: Vorstellung des Projekts

In der obigen Zeitabfolge sind nur Meetings aufgeführt, die durch physische Anwesenheit an der Hochschule stattfanden. Virtuelle Meetings, die im Fortgang des Projektfortschritts immer relevanter wurden, sind nicht visualisiert.

4 Meilensteine der (agilen) Softwareentwicklung

4.1 Konzeption des Front-Ends

Das Front-End stellt die visuelle Kommunikation zwischen dem Lageristen und SAPlexa dar. Im Fall SAPlexa besteht das Front-End aus einer GUI¹, die als graphische Schnittstelle zwischen Mensch und dem technischen System agiert. Dabei ist es einem Lageristen nicht von großer Bedeutung wie eine GUI selbst programmiert ist, sondern viel mehr wie diese optisch aufgebaut ist. Man spricht hierbei von der *Nutzersicht der GUI*, die man anhand folgender Fragestellungen näher beschreiben kann:

- ❖ **Werden alle Elemente auf der Oberfläche erkannt?**
Sind die Kontraste dieser Elemente stark genug? Werden die Oberflächenbereiche deutlich genug voneinander abgesetzt? Sind alle Beschriftungen lesbar?
- ❖ **Sind die Abläufe in meinem Arbeitsumfeld logisch und folgerichtig?**
Sind diese Abläufe schnell zu lernen (Lernbarkeit)? Ist die Oberfläche selbsterklärend?
- ❖ **Ist die Oberfläche konsistent in Aussehen und Verhalten?**
- ❖ **Wie schnell kann ich mit der GUI arbeiten?**
- ❖ **Ist die Oberfläche auf mein Arbeitsumfeld angepasst?**

Um Antworten auf diese Fragestellungen zu finden, ist es wichtig sich in die Lage eines Lageristen zu versetzen. Es ist von großer Bedeutung zu erkennen was dem Lageristen für die Buchung eines Wareneingangs wichtig ist. Welche Informationen werden benötigt und auf welche Informationen kann verzichtet werden um die Oberfläche ansprechend und überschaubar zu halten.

Ein weiteres wichtiges Merkmal, das leicht in Vergessenheit gerät, ist die Wahl der Schriftgröße. Da ein Lagerist auch mal mit größeren Lieferungen rechnen muss und dadurch etwas distanzierter vor dem Bildschirm steht, ist eine angemessene, große Schriftgröße ein wichtiges Merkmal, um eine hohe Nutzerfreundlichkeit zu garantieren.

¹GUI: Graphical User Interface

4.1.1 Entwurf der Menüführung

Rückblickend auf die Nutzersicht der graphischen Oberfläche, betrachten wir in den folgenden Absätzen wie der Lagerist durch die GUI geführt wird. Wir nehmen an, dass eine Lieferung beim Lageristen ankommt und nun über SAPlexa der Prozess zur Wareneingangsbuchung startet.

Die Ablauflogik und Menüführung wird allem Voran in der nachstehenden Abbildung visualisiert.

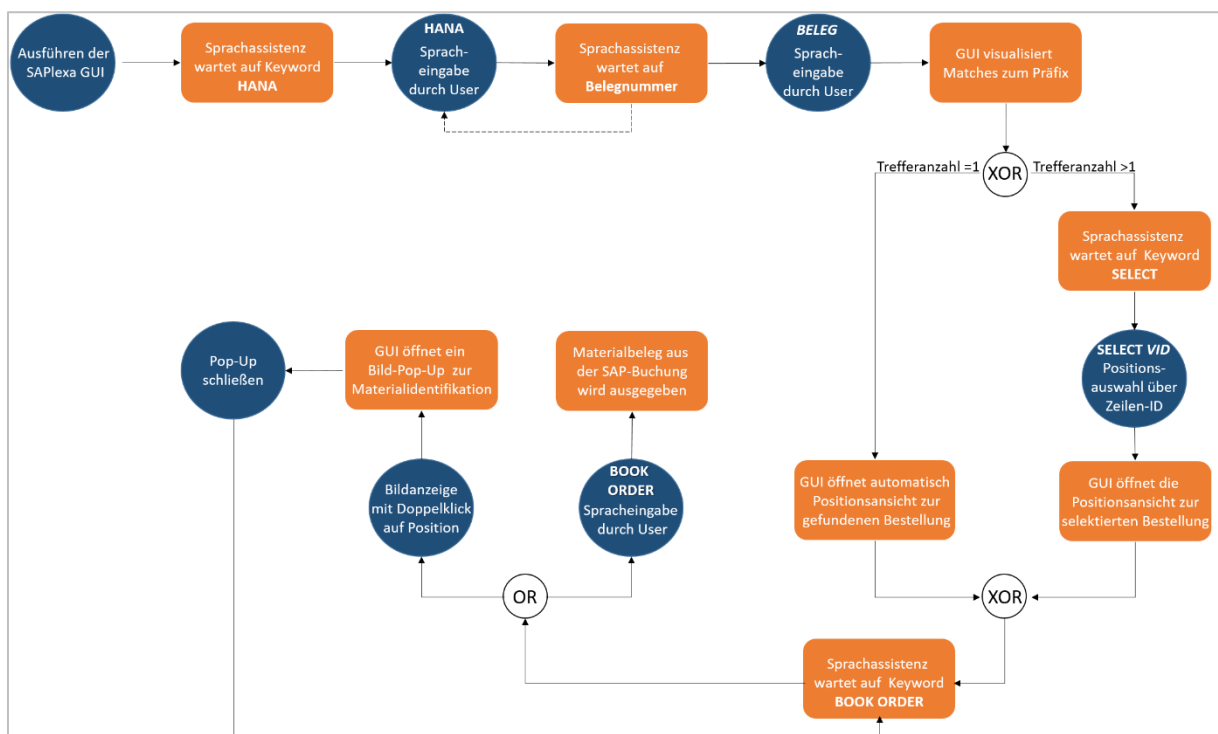


Abbildung 4.1 Menüführung als Ablauflogik

SAPiPlexa Settings

Ordered list

Looking for OID:

Say "HANA" to enter an order id

Say "HANA" for initiating the voice command.

Voice ID	Bestellnummer	Lieferant	Belegdatum

Pick out

Der Lagerist startet nun über das Kommando „**HANA**“ die Spracheingabe und kann die Bestellnummer auf dem Lieferschein an SAPlexa übergeben. Dabei wird die aktuell übergebene Ziffer der Bestellnummer im oberen linken Bereich angezeigt. Wenn die Bestellnummer übergeben wurde, bestätigt man die Eingabe über das Kommando „**OKAY**“.

11

SAPlexa Settings

Ordered list

Looking for OID: Now, say SELECT and choose your entry. Confirm with OKAY.

SAP is listening ...

Voice ID	Bestellnummer	Lieferant	Belegdatum
0	4500000666	0000120000	Tue Oct 23 00:00:00 CES...
1	4500000166	LF1-143	Wed Nov 07 00:00:00 CE...
2	4500000266	LF1-126	Mon Nov 19 00:00:00 CE...
3	4500000366	LF1-133	Tue Dec 04 00:00:00 CET ...
4	4500000466	LF1-103	Tue Apr 09 00:00:00 CES...
5	4500000566	LF1-185	Sun May 19 00:00:00 CES...
6	4500000660	LF1-183	Mon Nov 25 00:00:00 CE...
7	4500000661	LF1-183	Mon Nov 25 00:00:00 CE...
8	4500000662	LF1-002	Wed Nov 27 00:00:00 CE...
9	4500000663	LF1-182	Wed Nov 27 00:00:00 CE...
10	4500000664	LF1-183	Wed Nov 27 00:00:00 CE...
11	4500000665	LF1-183	Wed Nov 27 00:00:00 CE...
12	4500000666	LF1-182	Thu Nov 28 00:00:00 CET...
13	4500000667	LF1-182	Fri Nov 29 00:00:00 CET ...
14	4500000668	LF1-182	Thu Dec 05 00:00:00 CET...
15	4500000669	LF1-182	Thu Dec 05 00:00:00 CET...
16	4500000766	LF1-218	Tue Jan 14 00:00:00 CET ...

Pick out

Abbildung 4.3: Hauptmenü SAPlexa - Bestellnummer übergeben

Die zweite Möglichkeit besteht darin die ganze Bestellnummer an SAPlexa zu übergeben und dies direkt über das Kommando „**OKAY**“ zu bestätigen. Somit gelangt man ohne Umwege in die MIGO-Übersicht von SAPlexa.

MIGO Overview

General Information

Receipt ID: **4500000662**

Issued on: **Wed Nov 27 00:00:00 CET 2019**

Distributor Details

Distributor/Vendor: **<Lieferantname>**

Distributor ID: **LF1-002**

Position	Material	Description	Amount delivered	Amount ordered	Stock	Image
<input type="checkbox"/> 00010	R1-002	Rahmen 002	527	1000	RM00	DoubleClick me

Vorgesetzten anrufen

Say "Book Order" for adding a unit to stock

Abbildung 4.4: MIGO-Übersicht SAPlexa

Die MIGO-Übersicht enthält alle wichtigen Informationen für den Lageristen bezüglich einer Lieferung. Im oberen Bereich der MIGO-Übersicht werden die allgemeinen Daten zur Lieferung und die Details über den Lieferanten angezeigt. Dadurch kann der Lagerist die Lieferung schnell zuordnen.

Im Zentrum der MIGO-Übersicht werden alle Bestellpositionen zu der selektierten Lieferung aufgelistet. Dabei wird neben den allgemeinen Informationen wie dem gelieferten Material und der Produktbeschreibung auch die gelieferte Menge, die bestellte Menge und der Lagerort angezeigt. Dadurch kann der Lagerist auf einem Blick erkennen ob das richtige Produkt geliefert wurde und ob die gelieferte Menge der bestellten Menge entspricht. Zusätzlich bietet SAPlexa noch die Möglichkeit per Doppelklick ein Bild zu der ausgewählten Bestellposition anzuzeigen um das Produkt schneller zu identifizieren.

MIGO Overview

General Information

Receipt ID

4500000662

Issued on

Wed Nov 27 00:00:00 CET 2019

Distributor Details

Distributor/Vendor

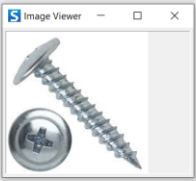
<Lieferantname>

Distributor ID

LF1-002

Position	Material	Description	Amount delivered	Amount ordered	Stock	Image
<input type="checkbox"/> 00010	R1-002	Rahmen 002	527	1000	RM00	Doubleclick me

Image Viewer



Say "Book Order" for adding a unit to stock

Vorgesetzten anrufen

Abbildung 4.5: MIGO-Übersicht SAPlexa - Bild anzeigen

Nach der Überprüfung der Bestellpositionen kann diese nun gebucht werden. Im unteren Bereich der MIGO-Übersicht wird angezeigt auf welches Kommando SAPlexa lauscht, um einen Wareneingang zu buchen. Ist nur eine Bestellposition aufgelistet, kann über das Kommando „**BOOK ORDER**“ die Bestellposition gebucht werden. Falls mehrere Bestellpositionen vorhanden sind, müssen diese explizit über das Kommando „**SELECT**“ in Kombination mit der Spaltennummer an SAPlexa übergeben werden und mit dem Kommando „**OKAY**“ bestätigen.

Hat der Lagerist die gewünschte Bestellposition gebucht, wird ein Materialbeleg zu der Buchung erzeugt und angezeigt.

MIGO Overview

General Information
Receipt ID 4500000662
Issued on Wed Nov 27 00:00:00 CET 2019

Distributor Details
Distributor/Vendor <Lieferantname>
Distributor ID LF1-002

Position	Material	Description	Amount delivered	Amount ordered	Stock	Image
<input type="checkbox"/> 00010	R1-002	Rahmen 002	527	1000	RM00	DoubleClick me

Material Receipt :5000000785

Ok

Say "Book Order" for adding a unit to stock

Vorgesetzten anrufen

Abbildung 4.6: MIGO-Übersicht SAPlexa – Materialbeleg erzeugen

4.1.2 Ergonomie und Erprobung von Schlüsselbegriffen

ERGONOMIE

Unter Ergonomie versteht man die wechselseitige Anpassung zwischen Menschen und ihren Arbeitsbedingungen. Auch in der Softwareentwicklung sollte auf die Ergonomie der Anwendung geachtet werden. Dabei sollte man folgende Kriterien/Eigenschaften im Blick behalten:

Attribut	Beschreibung
Konsistenz	Der Nutzer sollte auf bekannte Muster und Funktionen vertrauen können.
Verfügbarkeit	Features und Funktionen sollten nicht den Fluss der Software blockieren.
Verständlichkeit	Ähnliche Funktionen/Tools sollten in gleichen Sektionen zu finden sein und lassen sich auch ähnlich bedienen.
Automatisierung wiederholter Aufgaben	Repetitive Aufgaben sollten automatisiert werden oder optional automatisierbar sein.
Umgehende Rückmeldung	Der Nutzer sollte sowohl über fehlgeschlagene als auch durch erfolgreiche Aktionen informiert werden. (Fehler sollten „aufdringlich“ gezeigt werden z.B. Popup, Erfolge nur „nebenbei“ z.B. Statusziele)
Selbsterklärung	Durch Namen oder visuelle Elemente sollte sich der Sinn bzw. die Auswirkung einer Funktion erahnen lassen.
Anpassbarkeit	Der Nutzer sollte Optionen nach persönlicher Präferenz einstellen können
Fehlertoleranz	Fehler sollten vom Verursacher rückgängig gemacht bzw. korrigiert werden können
Erwartungskonformität	Funktionen sollten korrekt betitelt werden und nicht unvorhersehbare Auswirkungen mit sich bringen
Höflichkeit	Der Nutzer sollte auf Fehler, bei Aufforderungen oder einfachen Statusmeldungen immer Höflich auf seine Situation aufmerksam gemacht werden.

Gerade bei Benutzereingaben sollte darauf geachtet werden, dass es zu möglichst wenig Missverständnissen kommt und die Eingaben für die vorgesehenen Auftrittsumstände angemessen ausgewählt werden. Ferner sollte ebenfalls überlegt werden, wie komplex die Nutzereingaben sein können, sollen und dürfen. Generell sollten häufig auftretende Eingaben möglichst kurzgehalten werden, um Zeit und Aufwand bei der Eingabe gering zu halten. Beim Fall der Spracherkennung sollte ein gesunder Kompromiss zwischen für den Arbeiter angenehm aufzusagen, als auch von der Spracherkennungssoftware schnell und zuverlässig erkennbar sein.

ERPROBUNG VON SCHLÜSSELBEGRIFFEN

Bei der Erprobung von Schlüsselbegriffen sollten die oben genannten Punkte der Ergonomie im Hinterkopf behalten werden. Jedoch spielt bei den Schlüsselbegriffen die zuverlässige Erkennung in Speech2Text (S2T) Software eine wichtigere Rolle. Dabei sollten die Begriffe sorgfältig ausgewählt werden. Um die Fehlerquote zu senken, hilft es, sich vorab Gedanken zu machen, was der S2T-Software helfen könnte Wörter besser zu differenzieren. Jedoch darf ein Erproben der Begriffe in einem Umfeld, welches dem zukünftigen Einsatzgebiet ähnelt, nicht vernachlässigt werden. Dabei muss sowohl die Fehlerquote, als auch die Ergonomie betrachtet werden, wobei eine niedrige Fehlerquote automatisch die Ergonomie verbessert.

Im Folgenden wird am Beispiel SAPlexa die Erprobung und das Ersetzen der Schlüsselbegriffe erklärt.

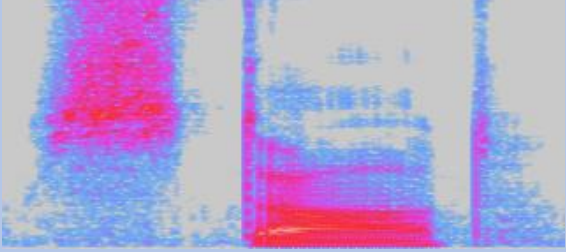
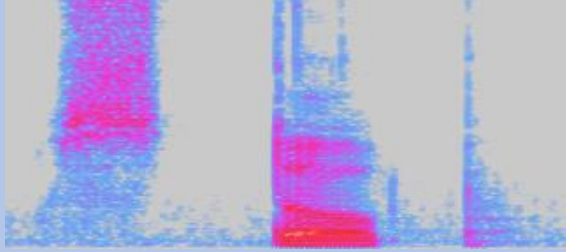
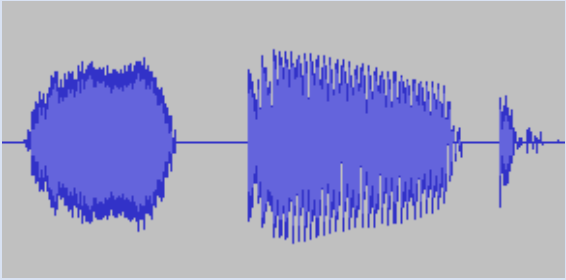
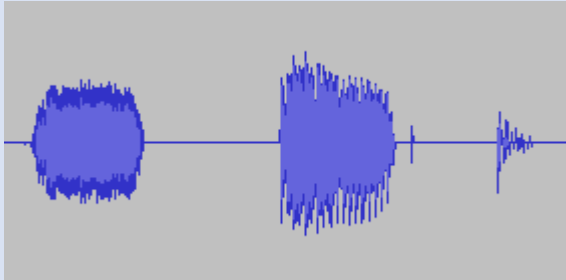
Zu Beginn der Entwicklung wurden die Schlüsselbegriffe „Start Number“ und „End Number“ genutzt, um das Zusammensetzen einer neuen Ziffernkette zu beginnen bzw. zu beenden. Schnell wurde dabei klar, dass das Wort „Number“ weder zur besseren Erkennung noch zur Ergonomie beiträgt und wurde damit schnell verworfen. Damit wechselten wir zu neuen ergonomischeren Schlüsselwörter wie „Start“ und „Stop“ (jeweils in englischer Aussprache, da ein englisches Spracherkennungsmodell verwendet wurde).

In den im Entwicklungsprozess üblichen Unit-Test fiel eine häufige Fehlerkennung der beiden Worte auf, was wiederum die Ergonomie der Sprachbedienung durch eine hohe Fehlerquote verschlechterte. Beim genaueren Betrachten der Begriffe über eine Audioverarbeitungssoftware ist zu erkennen, dass die Wörter sich in verschiedenen audiovisuellen Formen kaum unterscheiden.

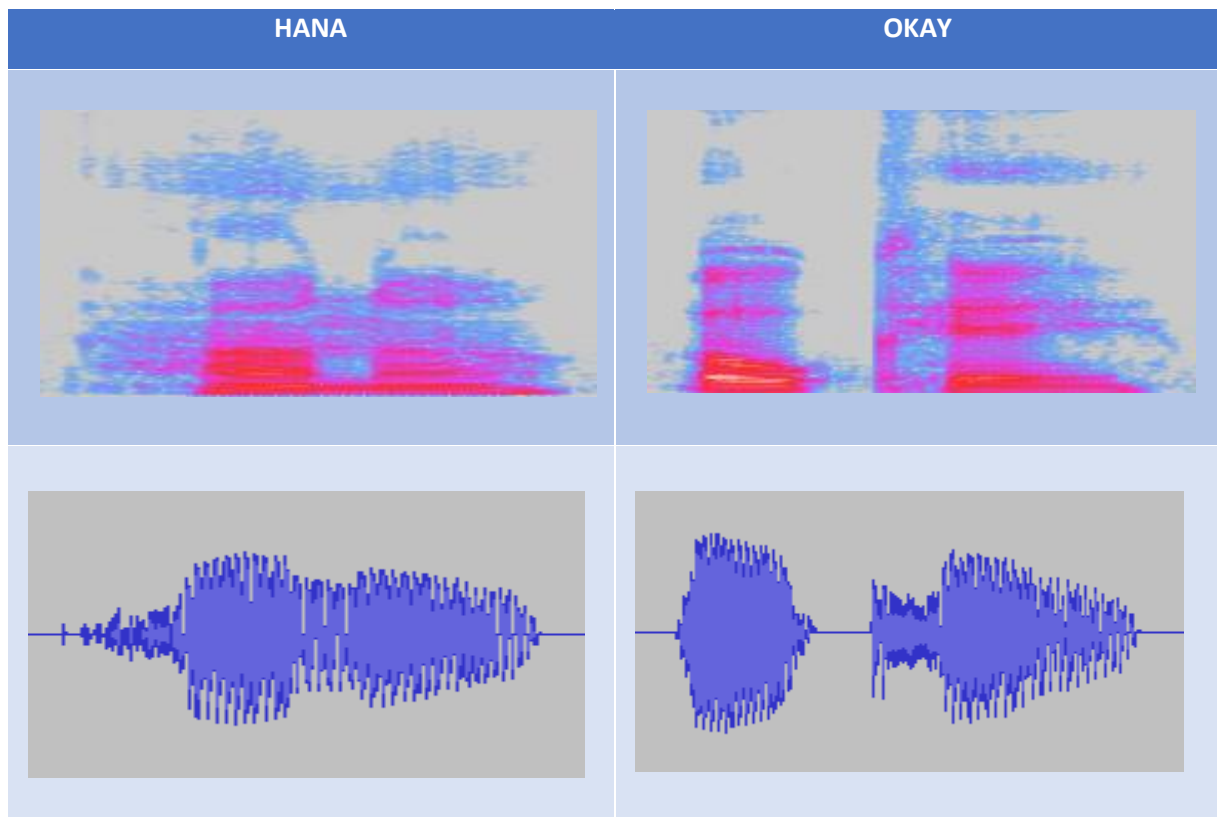
In den untenstehenden Tabellen sind zwei verschiedene Arten zur Visualisierung von Audiodaten.

Die obige Zeile der Tabellen zeigt ein sogenanntes Spektrogramm oder speziell bei Audiodaten auch Sonagramm genannt, welches die Energiedichte der Daten in bestimmten Frequenzbereichen über einen Zeitraum aufzeigt. Die darauffolgende Zeile zeigt die eher übliche Waveform (Wellenform).

Die folgende Tabelle stellt die bei SAPlexa sich als problematisch herausgestellten Begriffe gegenüber.

START	STOP
	
	

Die Aufnahmen wurden mit einem für den Privatgebrauch sehr gutem Mikrofon in einer ruhigen Umgebung aufgenommen. Somit sind wenig Verunreinigungen wie Hintergrundrauschen in den Audiosamples enthalten. Es sind zwar Unterschiede zu erkennen, jedoch sind diese eher über den Zeitraum zu erkennen und können bei weniger optimalen Bedingungen sowie anderer Aussprache noch weniger erkennbare Unterschiede aufweisen.



In der obigen Tabelle stehen sich die „neuen“ ausgetauschten Schlüsselbegriffe „Hana“ und „Okay“ der Anwendung gegenüber. Es sind direkt mehrere Unterschiede zu erkennen, was sich auch in der Fehlerkennungsrate der Speech2Text-Anwendung widerspiegelt. Die Begriffe sind recht kurz geblieben, was sich zusammen mit der geringeren Fehlerquote positiv auf die Ergonomie der Applikation auswirkt. Eventuell könnte die Fehlerkennung durch Wörter mit drei bis fünf Silben nochmals gesenkt werden.

4.1.3 Informationsbeschaffung bei der Groz-Beckert KG

Im Laufe des Projektstudiums sind viele Fragen aufgekommen. Als wir uns mit dem SAP-Prozess, den Wareneingang, beschäftigt haben ist uns aufgefallen, wie viele Informationen grundsätzlich geliefert werden. Wir mussten uns überlegen, welche Informationen relevant für den Lageristen im Wareneingang sind und welche hingegen weniger relevant sind. Die besten Antworten auf unsere Fragen hätte nur jemand liefern können, der täglich damit zu tun hat. Unser Projektleiter organisierte deswegen eine Exkursion zur Groz-Beckert KG.

Nach einem professionellen Empfang und einer herzlichen Begrüßung begann die Exkursion mit einem kleinen Einblick in das Entwicklergebäude, etwas Basis-Wissen über das Unternehmen selbst und was es genau macht.

Unsere Exkursion führte uns als nächstes zum Warenausgang. Uns ist recht schnell aufgefallen, dass der Geräuschpegel äußerst angenehm war und ideal, um mit einer sprachgesteuerte Applikation zu arbeiten. Zu unserem Erstaunen setzte das Unternehmen im Warenausgang bereits eine sprachgesteuerte Applikation, namens Lydia, ein. Lydia ist eine Pick-per-Voice Applikation. Sie ist ein nicht lernendes System was bedeutet, dass den Arbeitern die Aussprache antrainiert werden muss und Dialekte abgelegt werden müssen. Lydia sagt dem Arbeiter wohin er gehen muss. Das Lager war vollständig gekennzeichnet mit Umlauten gefolgt von Ziffern. Das Lager funktioniert als eine chaotische Lagerhaltung. Das einzulagernde Produkt erhält keinen vorbestimmten Lagerplatz, es wird vielmehr auf beliebige freie Stellen eingelagert.

Lydia basiert auf ein Wiederholssystem und funktioniert wie folgt:

Sobald die Bestellnummer aufgesagt wird, schickt Lydia den Arbeiter zu einem gekennzeichneten Regalplatz. Der Lagerist wiederholt die Kennzeichnung des Regales und Lydia sagt, ob der Mitarbeiter an der richtigen Stelle steht. Wenn man falsch steht, weist Lydia darauf hin und wiederholt die Kennzeichnung. Der Mitarbeiter kann sich nun korrigieren. An der richtigen Position angekommen nennt Lydia die Menge des herauszunehmenden Produktes. Ist die Bestellung vollzählig, kann es weiter zur Sammelstation gehen.

Vom Warenausgang ging es nun in ein Meeting-Zimmer indem wir unser Projekt mit Hilfe einer PowerPoint-Präsentation vorgestellt haben. An der Präsentation hat zusätzlich der Leiter des Wareneingangs teilgenommen. Nach der Vorstellung unseres Projektes, haben wir über Verbesserungsvorschläge und allgemeinen Input gesprochen, welche wir im Wareneingang mit dem Leiter des Wareneingangs und den Mitarbeitern näher besprechen werden.

Im Wareneingang angekommen war es erstmals, wie im Warenausgang, ruhig und der Geräuschpegel angenehm. Das täuschte wie es sich herausgestellt hatte, denn es war Mittagszeit und nur ein paar Mitarbeiter waren noch an den Arbeitsplätzen. Normalerweise geht es laut zu, was sich definitiv als Problem für die sprachgesteuerte Applikation SAPlexa darstellen würde.

Das Auspacken ist manuell zu erledigen. Man nimmt sich zufällig ein Paket aus dem Sammelbehälter raus und scannt den Strichcode oder tippt die Bestellnummer ein. Sind beide unerkennbar beschädigt, muss ein Blick in das Paket für Klarheit sorgen, andernfalls landet das Paket im „Ware in Klärung“-Regal. Zehn bis Fünfzehn Prozent der Ware wird auch direkt am LKW entgegengenommen, wo eine sprachgesteuerte Applikation sehr helfen würde. In einem Gespräch mit dem Leiter des Wareneingangs und den Mitarbeitern haben wir viele Informationen erhalten, welche uns weitergeholfen haben. Von diversen Komplikationen mit der Identifizierung der Ware im Paket bis hin zu Hilfestellungen, wie Bilder der Ware, die zur Identifizierung helfen. Eine sprachgesteuerte Applikation ist bei den Mitarbeitern sehr wünschenswert.

Die Exkursion war eine gelungene Unternehmung, die uns Fortschritt und Wissen um die Lage im Wareneingang und mehr eingebracht hat. Vor Allem im Thema der Gestaltung und Weiterentwicklung der GUI und auf die Ergonomie war die Exkursion eine große Hilfe. In den Meilensteinen der (agilen) Softwareentwicklung wurde dazu bereits eingegangen.

4.2 Konzeption des Back-Ends

4.2.1 Java-Perspektive

4.2.1.1 Auswahl der Speech-to-Text API – CMUSphinx

Bevor mit der Erstellung der SAP Sprachsteuerung begonnen werden konnte, musste zuerst ein grober Überblick über die zur Verfügung stehenden Speech-to-Text APIs verschafft werden, um eine passende Speech-to-Text API auszuwählen, die die Ansprüche an das Projekt erfüllt. Nach etwas Recherche wurden schon ein paar mögliche Kandidaten gefunden. Es wurde nun versucht diese miteinander zu vergleichen, um eine Speech-to-Text API zu finden, die für das Projekt verwendet werden kann. Folgende Speech-to-Text APIs wurden genauer betrachtet:

„Google Speech-to-Text API“ über die Google Cloud. Die Vorteile dieser Speech-to-Text Lösung sind eine hohe Genauigkeit und eine der geringsten Fehlerquote im Bereich der Spracherkennung. Außerdem erkennt sie viele verschiedene Sprachen schon standardmäßig. Leider kostet die Nutzung der Speech-to-Text API, da sie wie oben schon erwähnt über die Google Cloud läuft. Auch wäre der Datenschutz durch die Cloud Lösung vermutlich etwas schwerer zu bewerkstelligen.

Auch Microsoft hat eine Speech-to-Text API auf dem Markt – die „Microsoft Cognitive Services“. Diese Speech-to-Text API hat jedoch so ziemlich die gleichen Vorteile aber auch Probleme, wie die von Google bereitgestellte Lösung, da sie auch über die Cloud arbeitet und somit nicht unentgeltlich zur Verfügung steht.

Durch die Betrachtung der beiden oben genannten möglichen Lösungen entstand die Erkenntnis, dass eine Cloud Lösung nicht optimal ist, da diese immer mit Kosten verbunden ist. Es wäre besser eine Offline Speech-to-Text API zu verwenden, da so auch ein Internet-Zwang entfällt. Es wurde bei der weiteren Suche nun nach Offline Varianten gesucht und auch drei weitere Kandidaten gefunden.

„IBM Watson“ und „Speechmatics“ sind oft benutzte Offline Speech-to-Text APIs, jedoch kosten beide leider auch Geld, was bei einer professionellen Anwendung kein großes Problem darstellt, da diese Varianten oft eine sehr gute Qualität aufweisen, einen guten Support bieten und auch regelmäßig geupdatet werden. Bei einem Projektstudium kommt dies jedoch nicht wirklich infrage.

Eine weitere Offline Speech-to-Text API ist die Open Source Variante „CMUSphinx“, welche von der „Carnegie Mellon University (CMU)“ veröffentlicht wurde und auch regelmäßig mit Updates versorgt wird. Sie wird oft bei Projekten an Universitäten und allgemein zu Lehrzwecken eingesetzt, da sie eine kostenlose Offline-Lösung, eine gute Qualität mit geringer Fehlerquote, die Dokumentation gut und ausführlich ist und viele Tutorials zu ihrer Nutzung im Internet zur Verfügung stehen. Außerdem hatten zwei Personen in der

Projektgruppe schon etwas Erfahrung mit „CMUSphinx“, da sie bei dem „Makeathon“ an der Hochschule eingesetzt wurde und die beiden Gruppenmitglieder dort teilgenommen hatten.

Nach der Sichtung dieser verschiedenen Speech-to-Text APIs, wurde sich für „CMUSphinx“ als Speech-to-Text API entschieden, da diese Variante kostenlos ist, eine gute Dokumentation besitzt und die bereits (etwas) vorhandene Erfahrung praktisch ist.

4.2.1.2 Speech-to-Text – Implementierung

Bevor mit der eigentlichen Programmierung gestartet werden konnte, wurden zuerst ein paar Online Tutorial Videos angeschaut, um sich einen Überblick über die Sprachsteuerung zu verschaffen. Nachdem man sich mit den Grundfunktionen der Sprachsteuerung und der Bibliotheken auseinandergesetzt hat. Anhand von ein paar Dokumentationen zur Verwendung von CMUSphinx, wurde dann das erste Grundgerüst der Sprachsteuerung programmiert.

Vor dem weiteren Programmieren musste noch eine weitere Rahmenbedingung geklärt werden und zwar, ob man eine gesamte Sprache erkennen möchte, also z.B. Deutsch oder Englisch, oder ob man mithilfe einer sogenannten „Grammatik“ nur ausgewählte Wörter erkennen möchte. In einer Grammatik kann man die Wörter, die man erkennt haben möchte reinschreiben, um False-Positives zu vermeiden, indem man die zu erkennende „Grammatik“ eingrenzt. Im Rahmen unseres Projektes wurden folgenden Wörter in die „Grammatik“ einbezogen:

HANA:

Wird zum Start der Spracherkennung benutzt. Nachdem die Anwendung das Schlüsselwort „HANA“ erkennt, beginnt sie nun nur noch auf die folgenden beschriebenen Ziffern zu reagieren.

Zero; one; two; three; four; five; six; seven; eight; nine:

Die Spracherkennung erkennt die gesagten einzelnen Ziffern und fügt sie in die Anwendungslogik ein und zeigt sie auch in der GUI auf.

Select:

„Select“ wählt die mit dem Schlüsselwort verbundene Eingabe aus, um weiter im Menüfluss voranzuschreiten.

Okay:

Mit dem Schlüsselwort „Okay“ kann man eine Eingabe bestätigen und zum nächsten Menu bzw. zur nächsten Eingabe voranschreiten.

Book Order:

Mit diesem Schlüsselbegriff wird am Ende der Wareneingang im SAP System gebucht.

Es wurde bewusst nur eine kleine Menge an Schlüsselworten angelegt, um eine einfachere Bedienung zu gewährleisten. So wie in 4.1.2 beschrieben wurde, haben wir die

Schlüsselworte so angepasst, dass sie von der Spracherkennung gut voneinander unterschieden werden können.

Im folgenden Absatz wird der Ablauf der Spracherkennung beschrieben.

Am Anfang des Spracherkennungszyklus wartet die Anwendung auf den Begriff „HANA“ um eine neue Suche nach Bestellnummern zu initialisieren. Nach der erfolgreichen Erkennung des Schlüsselwortes können nun die Ziffern der Bestellnummer genannt werden. Die Anwendungslogik erkennt die Ziffern und wandelt sie in Integer Zahlen um, um sie verarbeiten und auf der GUI anzeigen zu können. Nach jeder erkannten Zahl werden die passenden Bestellnummern auf der GUI angezeigt, damit der Benutzer bei mehreren angezeigten Bestellnummern gegebenenfalls die passende Nummer mit dem Befehl „Select“ auswählen kann. Nach Bestätigung der richtigen Bestellnummer wird die in 4.1.1 beschriebene Detailansicht aufgerufen. Daraufhin kann der gesamte Wareneingang mit „Book Order“ gebucht werden.

4.2.1.3 (Auswahl der) Java Libraries – SWT

SAPlexa wurde in der Entwicklungsumgebung Eclipse entwickelt und als Programmiersprache wurde Java verwendet. Somit gab es mehrere Möglichkeiten eine GUI zu implementieren. Die Entscheidung fiel dabei auf Java SWT, da für die Entwicklung von SAPlexa keine komplexen Frameworks notwendig waren, sondern diese einfach gehalten werden kann. Außerdem konnte mit Java SWT besser und schneller entwickelt werden, da das Entwicklungsteam bereits fortgeschrittene Kenntnisse hatte.

4.2.1.4 Eingehen von technischer Schuld

Ein Punkt, der leider aus zeittechnischer Perspektive nicht bewerkstelligt werden konnte, ist das sogenannte „Training“ (oder eine „Acoustic Model Adaptation“) der Sprachsteuerung. „CMUSphinx“ bietet nämlich die Möglichkeit mithilfe von „.wav“ Dateien „trainiert“ zu werden, also z.B. spezielle Wörter wie „SAPlexa“ zu lernen, da dieses ja kein gewöhnliches Wort einer bekannten Sprache ist oder eine ganze neue Sprache zu lernen falls diese noch nicht standardmäßig von „CMUSphinx“ unterstützt wird. Es ist auch möglich einen Dialekt zu lernen, was z.B. bei „Groz Beckert“ (auf der Alb, wo manchmal auch nicht unbedingt hochdeutsch gesprochen wird) ein Vorteil oder sogar eine Voraussetzung sein könnte. Die .wav Dateien können dann z.B. die Aufnahmen der Stimme einer Person enthalten, die ein bestimmtes oder mehrerer bestimmte Worte (oder auch Sätze bzw. längere Texte für eine neue Sprache) wiederholen, um die Stimme bzw. den Dialekt oder eine neue Sprache zu erlernen. Das Trainieren von CMUSphinx ist jedoch nicht einfach und erfordert auch, wenn man es richtig machen möchte, viele Dateien und aufgenommene Stimmen von Personen. Das Trainieren selbst ist auch etwas komplizierter und hätte leider in dem Projektzeitraum keinen Platz mehr gefunden.

Ein weiterer Punkt, der leider nicht mehr in den zeitlichen Ablauf untergebracht werden konnten, war der Abbruch einer Buchung im GUI oder ein Sprung zurück auf die vorherige Seite. Dies wäre natürlich noch ein sehr wichtiger Punkt, der unerlässlich für die Menüführung ist, da ein Abbruch einer Buchung oder eine Korrektur eine Eingabe oft gebraucht wird. Bei einer Sprachsteuerung kann dies vor Allem vorkommen, wenn sie in einem Bereich eingesetzt wird, in dem es viele Hintergrundgeräusche gibt.

4.2.2 SAP-Perspektive

Als angemeldeter SAP-Benutzer erfolgt die Wareneingangsbuchung über den Transaktionsbefehl MIGO. In unserem Fall handelt es sich um eine Wareneingangsbuchung zu einer zuvor elektronisch hinterlegten und getätigten Bestellung beim Lieferanten. Unter Angabe der Bestellnummer kann somit im Anschluss die Buchung aller gelieferten Positionen vorgenommen werden. Das anschließende Unterkapitel soll diesen Vorgang aus Sicht der Datenbank-Transaktionen erläutern.

4.2.2.1 Technischer Vorgang einer Wareneingangsbuchung im SAP

Im Hintergrund des o.g. Buchungsprozesses spielen in SAP HANA 3 systemeigene Datenbanktabellen eine wesentliche Rolle. Diese sind im Folgenden mit deren Funktionsbezeichnungen aufgeführt.

Name	Kurzbeschreibung der Datenbanktabelle
	Informationsgehalt
EKKO	Einkaufsbelegkopf
	Die EKKO-Tabelle enthält übergeordnete Informationen zu den im System hinterlegten Einkaufsbelegen. Derartige Informationen können beispielsweise Belegnummern, Zeitstempel zum Anlagezeitpunkt, Lieferantenummer oder auch der Buchungskreis selbst sein.
EKPO	Einkaufsbelegposition
	Die EKPO-Tabelle enthält im Vergleich zur EKKO-Tabelle nähere Informationen zu den im Einkaufsbeleg hinterlegten Positionen. Hierin werden Informationen zu Materialnummern, Lagerort, Werk oder auch Bestellmenge gelistet.
EKET	Lieferplaneinteilungen
	Als Nachweis für den Warenverkehr dient die Datenbanktabelle EKET. Im Rahmen dieses Projekts steht die darin enthaltene Information zu bereits gelieferten Positionen im Mittelpunkt. Damit wird u. A. die Verwaltung von Teillieferungen ermöglicht.

Tabelle 4.1 Relevante Datenbanktabellen in HANA

Der Bestellbeleg beinhaltet i.d.R. mindestens eine Bestellposition. Hierbei besteht die Möglichkeit den Wareneingang zu den einzelnen Bestellpositionen sowohl nach Eintreffen einer einmaligen, vollständigen Lieferung, als auch nach Eintreffen mehrmaliger Teillieferungen zu verbuchen. Die Beziehungen zwischen den o.g. Tabellen lassen sich demnach als folgende Kardinalitäten beschreiben:



Abbildung 4.7 Kardinalitäten zwischen EKKO, EKPO und EKET²

Der Zugriff auf die genannten Datenbanktabellen soll über sog. Funktionsbausteine erfolgen, die von außerhalb des SAP-Systems zugänglich sind. Um die Funktionsweise dieser Funktionsbausteine besser zu illustrieren, werden im nächsten Unterkapitel zunächst die relevanten Datenstrukturen aus dem SAP System vorgestellt, welche für die Datenübermittlung verwendet werden sollen.

² EKKO-Tabelle, Böselager.

4.2.2.2 Relevante Datenstrukturen im ABAP Dictionary

Als relevante Datenstrukturen werden im Rahmen dieser Ausarbeitung derartige Strukturen verstanden, die im Rahmen der Datenübermittlung zwischen dem SAP-System und der GUI-Anwendung verwendet werden. Dazu gehören im Wesentlichen die (Zeilen-)Strukturen als solches und die Tabellentypen. Diese können im ABAP Dictionary unter der Transaktion SE11 neu definiert und systemintern in der ABAP Programmierung verwendet werden. Durch die Definition neuer, flacher Datenstrukturen soll die Datenverarbeitung effizienter, zweckmäßiger und übersichtlicher gestaltet werden. Nicht zuletzt können sich daraus Performance-Verbesserungen ergeben, die mit der Vermeidung von entbehrlichen Datenabfragen und -übertragungen einhergehen. Aus diesem Grund wurde bewusst auf bereits vordefinierte Systemstrukturen verzichtet.

Tabellentypen können sich auf eine vordefinierte Struktur beziehen, welche die Zeilenstruktur für den Tabellentyp vorgibt. Programminterne ABAP-Objekte, die demnach auf einen Tabellentypen verweisen, sind interne Tabellen des entsprechenden Zeilentyps aus der Struktur.³ Die anschließende Abbildung soll die Beziehungen zwischen den einzelnen Elementen und deren Bezug zu vordefinierten Datentypen visualisieren. Dabei ist noch hinzuzufügen, dass im Sinne der Übersichtlichkeit alle referenzierten Komponententypen (in der Abbildung als Spalte bezeichnet), auf welche sich die einzelnen Komponenten der Struktur primär beziehen, ausgeblendet wurden. Stattdessen wird der zugrundeliegende Datentyp visualisiert. Die vollständigen Bezeichnungen zu den Akronymen aus den Typdefinitionen können im Abkürzungsverzeichnis nachgeschlagen werden.

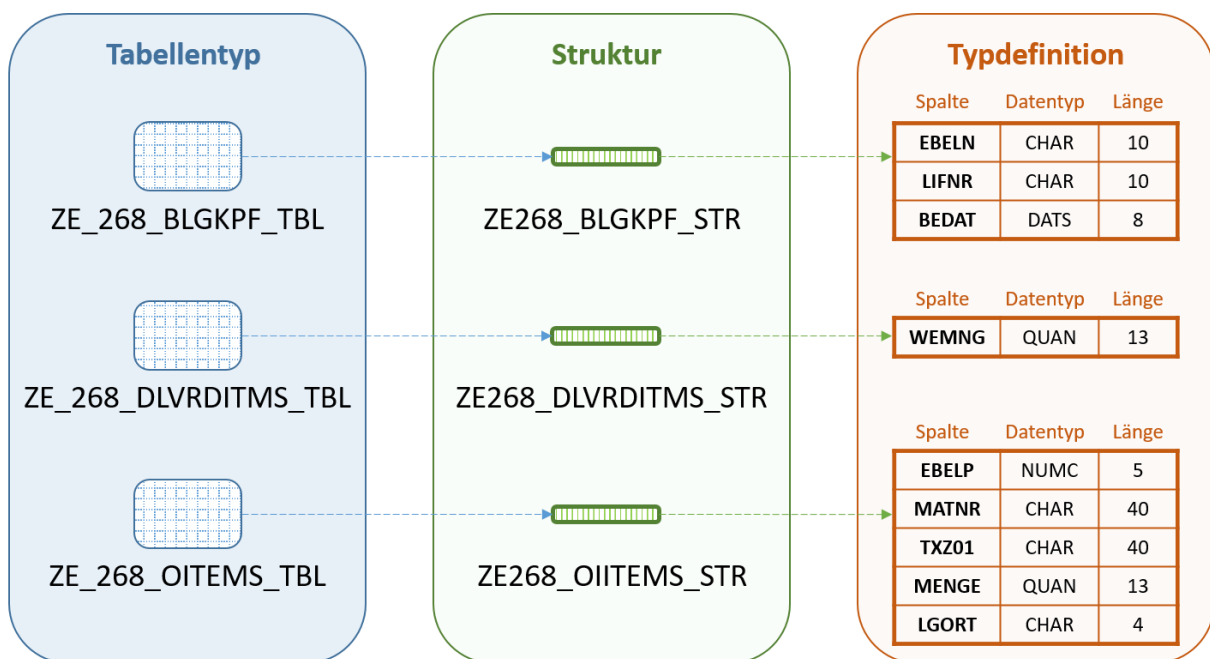


Abbildung 4.8 Tabellentypen - Strukturen - Datentypen

³ Tabellentypen, SAP.

Mithilfe dieser erzeugten Strukturen werden im nächsten Schritt die Ausgabe-Parameter der Funktionsbausteine definiert.

Unter Verwendung von Mengen- oder auch Währungskomponenten muss zusätzlich ein Referenzfeld als Einheitenschlüssel hinterlegt werden, welches die Mengen- bzw. Währungseinheit spezifiziert. Dies wird in der Einzelfeldpflege der Strukturkomponente vorgenommen.⁴

⁴ Mengenfelder, SAP.

4.2.2.3 Implementierung der Funktionsbausteine

Im SAP ABAP Kontext dienen Funktionsbausteine zur Kapselung bzw. Auslagerung von Programmcode, um einen globalen Zugriff auf dessen Funktionalität im SAP-System zu gewährleisten.⁵ Unterschieden werden folgende Arten von Funktionsbausteinen:

- Normaler Funktionsbaustein
- Remote fähiger Funktionsbaustein – (Remote Function Call = RFC)
- Verbuchungsbaustein
 - Start sofort
 - Start sofort – nicht nachverbuchbar
 - Start verzögert
 - Sammellauf

Wie die jeweiligen Bezeichnungen richtig vermuten lassen, wird der Zugriff auf die relevanten Informationen aus Sicht eines Fremdsystems (hier: JAVA-GUI Programm) nur über den o.g. RFC-Baustein möglich, da dieser von außerhalb zugänglich sind und damit eine wichtige Voraussetzung für eine Schnittstelle zwischen SAP und JAVA erfüllt. Zu den übrigen Bausteinarten lässt sich vereinfacht darstellen, dass diese keinen Zugriff aus Fremdsystemen gewähren und sich demnach nicht für unser Projekt eignen. Normale Funktionsbausteine stehen primär für die Kapselung von Programmcode, wie z.B. einem Einheiten-Umrechner, wohingegen Verbuchungsbausteine die Manipulation von Datenbanktabellen fokussieren und deren eigenen SAP-seitigen Prüfmechanismus (Logical Unit of Work, kurz: LUW) zur Wahrung der Datenkonsistenz auslösen.⁶

⁵ Funktionsbausteine, SAP.

⁶ Verbuchungsfunktionsbaustein, SAP.





Funktionsbausteinname	Funktion	Importparameter	Wert	Typisierung
		Exportparameter		
 ZE268_GETPROPOSALLIST	Sucht Belegköpfe zu einem festgelegten Präfix.	I_PREFIX	Präfix Zahlenfolge	String
		E_ORDERLIST	Passende Belegköpfe	ZE_268_BLGKPF_TBL
 ZE268_GETORDERITEMS	Gibt alle Positionen zu einer Bestellnummer aus.	I_ORDERID	Bestellnummer	EBELN
		E_ITEMLIST	Positionsliste	ZE_268_OTIMES_TBL
 ZE268_GETOPENPOSITION	Gibt bereits gelieferte Positionsmenge zur Bestellung aus.	I_ORDERID	Bestellnummer	EBELN
		E_POSLIST	Positionsmengen	ZE_268_DLVRDITMS_TBL
 ZE270_GMVT	Bucht den Wareneingang und gibt den Materialbeleg zurück.	I_ORDERID I_POSNR I_MENGE I_UOM	Bestellnummer Positionsnummer Erfassungsmenge Mengeneinheit	EBELN EBELP ERFMG ERFME
		E_MATDOC	Materialbelegnr.	MBLNR

Tabelle 4.2 Definierte RFC-Funktionsbausteine in SAP

Funktionsbausteine bieten grundsätzlich die Möglichkeit zur Pflege von optionalen Import- bzw. Export-Parametern für den Datenaustausch mit dem aufrufenden Programm. Das SAP-System selbst stellt zahlreiche Funktionsbausteine, wie z.B. die BAPI Bausteine, zur sofortigen Verwendung bereit. Da sich die Quellcode-Programmierung der Funktionsbausteine in unserem Fall überwiegend auf relativ simple Open-SQL-Statements beschränkt, wurden für das SAPlexa-Projekt eigene, zweckorientierte Funktionsbausteine für den Informationsaustausch definiert. Die RFC-Bausteine sind in programmlogischer Reihenfolge in obiger Tabelle 4.2 visualisiert. Den neu hinzukommenden Typisierungen ERFMG (MENGE), ERFME (Mengeneinheitsschlüssel bzw. Referenzfeld zu ERFMG) und MBLNR (EBELN) liegt jeweils dieselbe Datenstruktur der in Klammer stehenden Datenelementen zugrunde.

```

1  FUNCTION ZE268_GETPROPOSALLIST .
2  "-----
3  " "Lokale Schnittstelle:
4  " IMPORTING
5  "     VALUE(I_PREFIX) TYPE STRING
6  " EXPORTING
7  "     VALUE(E_ORDERLIST) TYPE ZE_268_BLGKPF_TBL
8  "-----
9
10
11  DATA L_PREFIX TYPE STRING.
12
13
14  CONCATENATE '%' I_PREFIX '%' INTO L_PREFIX.
15
16  SELECT EBELN, LIFNR, BEDAT FROM EKKO
17  INTO CORRESPONDING FIELDS OF TABLE @E_ORDERLIST
18  WHERE EKKO~EBELN LIKE @L_PREFIX.

```

Abbildung 4.9 Quellcode ZE268_GETPROPOSALLIST

Um die Nutzung der Sprachsteuerung, und damit einhergehend die Eingabe der Bestellnummer, ergonomisch zu halten, wurde der Suchmechanismus nach Bestellbelegen im betroffenen Funktionsbaustein mit zwei Wildcards (siehe: %-Symbole) versehen. Damit kann der Benutzer in Kürze zu seinem gewünschten Beleg gelangen ohne dabei eine vorgegebene Abfragestruktur einhalten zu müssen. Der Vorteil hieraus liegt in der Zusammensetzung bzw. Nummernkreisstruktur der Bestellnummer. Bei einer feststehenden Nummernlänge für Bestellbelege von 10 Ziffern kann damit erreicht werden, dass lediglich eine überschaubare Anzahl an Endziffern diktiert werden müssen.

Die Ablauflogik der JAVA Programmierung bewirkt also, dass nach jeder schrittweisen Zahleneingabe bzw. -ansage durch den Benutzer eine neue Anfrage an das SAP System gesendet wird. Die hierfür zuständige Open SQL Abfrage beinhaltet dabei die gesuchte Zahlenfolge als String-Parameter und übergibt alle Einträge aus der EKKO-Tabelle an das aufrufende JAVA Programm, welche die entsprechende Zahlenfolge beinhalten. Die Programmierung der SAP-Schnittstelle steht im nachfolgenden Unterkapitel im Fokus.

Hinweis

Alle vollständigen Quellcodes zu den Programmierungen aller Bestandteile dieses Projekts sind im Anhang aufgeführt.

4.2.3 SAP Java Connector – Die Schnittstelle

Die Implementierung einer Schnittstelle zwischen Java und SAP wird durch den SAP-eigenen Java Connector realisiert. Für das Java Programm wird sowohl die physische JAR-Bibliothek des Java Connectors, als auch ein gültiges Benutzerkonto zur Anmeldung am SAP System benötigt. Alle zur Anmeldung benötigten Benutzer- und Serverinformationen werden in einer ausgelagerten JCoDestination-Datei in UTF-8 formatierter Textform gelistet.

Das folgende Abbild der Textdatei zeigt die zugrundeliegende Konfigurationen für das erfolgreiche Verbinden des Java Connectors mit unserem gehosteten SAP-Server aus Magdeburg. Unter Anderem werden darin Sprache, Mandant, Anmeldedaten des Users und Serverspezifikation festgelegt.

```
jco.client.lang=de
jco.client.client=202
jco.client.passwd=SAPLEXA
jco.client.user=RFC_SAPLEXA
jco.client.sysnr=95
jco.client.ashost=/H/cloud.ucc.ovgu.de/S/3299/H/a95z.2.ucc.md
jco.destination.peak_limit=10
jco.destination.pool_capacity=3
```

Nach einem erfolgreichen Ping-Test und Verbindungsaufbau stehen nun für das Ausführen der im SAP-System definierten Funktionsbausteine zahlreiche Java-Funktionen zur Verfügung. Im Rahmen dieser Ausarbeitung wird dabei nur auf die für dieses Projekt praktisch relevanten Funktionen eingegangen. Unter der Annahme, die Variable **repository** sei vom Typ JCoRepository, **function** vom Typ JCoFunction und **d** vom Typ JCoDestination, kann ein möglicher Java-seitiger Funktionsaufruf wie folgt aussehen.

```
30 public Collection<Order> getProposalList(String prefix) throws JCoException {
31     function = repository.getFunction("ZE268_GETPROPOSALLIST");
32     JCoParameterList input = function.getImportParameterList();
33     input.setValue("I_PREFIX", prefix);
34     function.execute(d);
35
36     JCoParameterList valuelist = function.getExportParameterList();
37     JCoTable jcotable = valuelist.getTable(0);
38
39     for (int i = 0; i < jcotable.getNumRows(); i++) {
40         jcotable.setRow(i);
41         ord = new Order();
42         ord.setOrderID(jcotable.getString("EBELN"));
43         ord.setDistributorID(jcotable.getString("LIFNR"));
44         ord.setOrderDate(jcotable.getDate("BEDAT"));
45
46         orderlist.add(ord);
47     }
48     return orderlist;
49 }
```

Abbildung 4.10 Java-seitiger Funktionsaufruf auf den Baustein ZE268_GETPROPOSALLIST

Die Funktionsweise des Zugriffs auf einen definierten Funktionsbaustein aus SAP sollte in obiger Abbildung selbsterklärend sein. Um die Datenströme zu klassifizieren werden die eigens hierfür kreierten Objektklassen „Order“ und „Item“ mit deren zugehörigen Felder instanziiert. Durch das schleifenweise Abfragen der Ergebniszeilen aus der exportierten (hier) internen Tabelle des Funktionsbausteins werden die einzelnen Objektinstanzen erzeugt, gefüllt und einer Collection zugewiesen, um die darauf anschließende Datenverarbeitung zu strukturieren.

Die tatsächliche Buchung des Wareneingangs greift erstmalig auf vordefinierte Bausteine in SAP zurück, den sogenannten BAPI-Bausteinen (Business Application Programming Interface). Durch die Verwendung der Bausteine können weitere für die Buchung benötigte Bestellinformationen abgefragt werden. Die Programmierung aus dem Funktionsbaustein ZE270_GMVT nutzt die nachstehenden BAPI-Funktionen.

- **BAPI_PO_GETDETAIL**

Dieser BAPI Baustein ermöglicht die Ausgabe einer detaillierten Auflistung von Bestellpositionen, die innerhalb einer ausgewählten Bestellung aufgeführt sind.⁷ Als notwendiger Importparameter hierfür ist die Belegnummer zu erwähnen.

- **BAPI_GOODSMVT_CREATE**

Bei diesem Baustein handelt es sich um ein universelles BAP-Interface, welches die Buchung einer Warenbewegung auf Datenbankebene vornimmt.⁸

- **BAPI_TRANSACTION_COMMIT**

Die Manipulation von Daten durch BAPI-Bausteine erfordert eine abschließende COMMIT Anweisung, um alle vorgenommen Änderungen final zu speichern. Grundsätzlich hängt dies mit der zuvor erwähnten LUW (SAP-Datenbankmechanismus zur Wahrung der Datenkonsistenz).⁹ Die Ausführung dieses BAPI-Bausteins umfasst dabei mehr als eine simple inline Commit-Work Anweisung. Weitergehende Details hierzu werden aus Gründen unzureichender Relevanz im Rahmen dieser Ausarbeitung ausgeblendet.

Als Export-Parameter übergibt der ZE270_GMVT Baustein den gebuchten Materialbeleg als numerischen String an das aufrufende Programm zurück. Geeignete Prüfmechanismen vor der Buchung eines Falschbelegs werden im aufrufenden Programm (hier: Java Programm) implementiert.

⁷ BAPI_PO_GETDETAIL, CONSULT.

⁸ BAPI_GOODSMVT_CREATE, Wiki SAP.

⁹ BAPI_TRANSACTION_COMMIT, Wiki SAP.

5 Zukünftige Optimierungs- und Erweiterungsmöglichkeiten

SAPlexa hört, versteht und antwortet visuell, wie man es sich wünscht und vorstellt, doch wie bei jeder Neuentwicklung gibt es stets Optimierungs- und Entwicklungsbedarf. Im Folgenden werden einige Punkte aufgezählt, die verbessert oder erweitert werden können, um SAPlexa zu einem noch besseren und unterstützenden Assistenten zu machen:

- ❖ Qualitätsprüfstand
 - In der realen Welt wird die bestellte Ware auf die Qualitätsstandards des jeweiligen Unternehmens geprüft. Dieser Schritt wird in SAPlexa nicht berücksichtigt.
- ❖ Dynamisierung der Buchung
 - Momentan wird durch den Sprachbefehl „Book Order“ nur eine Einheit der jeweiligen Position gebucht. Dies könnte man ändern indem man durch neue Sprachbefehle die Wahl hat entweder alles oder nur eine gewünschte Menge zu buchen.
- ❖ Ausweitung der Sprachassistentz
 - SAPlexa befasst sich mit dem Prozess im Wareneingang. Weitere Prozesse, die durch SAPlexa unterstützt werden können, sind z.B. der Warenausgang oder die Warenumlagerung.
- ❖ Spezifikation von Lieferungen
 - Die Bestellliste in SAPlexa unterscheidet Teillieferungen nicht von Gesamtlieferungen. Teillieferungen sollten dementsprechend gekennzeichnet und als Teillieferungen gebucht werden, damit der Lagerist und das Unternehmen keine fehlerhaften Zahlen aufweisen.

6 Reflexion und Fazit

Um das Projekt erfolgreich und praxisrelevant durchführen zu können, haben wir sehr früh auf einen direkten Bezug mit der betroffenen Zielgruppe gesetzt. Der Besuch bei der Groz-Beckert KG war damit ein zentraler und wichtiger Baustein des Projekts.

Die agile SCRUM-Methode half uns jederzeit, uns an sich anpassende Änderungen und Anforderungen anzupassen.

Nach relativ schneller Einigung auf die zu verwendeten Technologien konnten wir trotz technischer Schuld zielstrebig die Implementierung umsetzen.

Die Kommunikation im Team verlief immer reibungslos. Zu unserem Erfreuen konnten wir von den verteilten Know-How's aus den verschiedenen Studienbereichen – also der Wirtschaftsinformatik oder auch der IT-Security – profitieren.

Grundsätzlich konnten termingerechte Abgaben in über 90 % der Fälle garantiert werden. Aufgrund von Zeitmangel konnte die ausgiebig gefüllte Backlog-Liste nicht vollständig umgesetzt. Daraus ergeben sich u.A. die im vorhergehenden Kapitel „Zukünftige Optimierungs- und Erweiterungsmöglichkeiten“. Berücksichtigt man allerdings die Kürze der Zeit, erscheint uns das Projekt in unseren Augen als gelungen und erfolgreich. Die Programmierung ist zwar von der Marktreife noch weit entfernt, dennoch konnten wir eine lauffähige und (wenn auch nicht zu hundert Prozent) stabile Sprachassistentz für das SAP System ins Leben rufen, die sogar in der Praxis einen hohen Stellenwert einnehmen kann.

Literaturverzeichnis

BAPI_GOODSMVT_CREATE [Online] / Verf. Wiki SAP // Goods Movements with BAPI. - 06. Januar 2020. - <https://wiki.scn.sap.com/wiki/display/ERPSCM/Goods+Movements+with+BAPI>.

BAPI_PO_GETDETAIL [Online] // CONSULT. - 06. Januar 2020. - https://www.consolut.com/s/sap-ides-zugriff/d/e/doc/E-BAPI_PO_GETDETAIL/.

BAPI_TRANSACTION_COMMIT [Online] / Verf. Wiki SAP. - 06. Januar 2020. - https://wiki.scn.sap.com/wiki/display/ABAP/BAPI_TRANSACTION_COMMIT+versus+COMMIT+WORK.

Eclipse User Interface Guidelines [Online] / Verf. al. N. Edgar et // Official Eclipse Wiki. - 2016. - 02.. Februar 2020. - http://wiki.eclipse.org/User_Interface_Guidelines.

EKKO-Tabelle [Online] / Verf. Böselager Gerrit // ERP-Yourself. - 26. Dezember 2019. - <https://www.eryyourself.net/de/sap-tabellen/EKKO.html>.

Funktionsbausteine [Online] / Verf. SAP. - 26. Dezember 2019. - https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenabap_functions.htm.

Java-Entwicklung mit Eclipse 3.2 [Buch] / Verf. Daum B.. - Heidelberg : D.Punkt Verlag, 2006. - Bd. 4. Edition.

Mengenfelder [Online] / Verf. SAP. - 26. Dezember 2019. - https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-de/abenddic_quantity_field.htm.

Mobile Informationssysteme [Online] / Verf. Becker Peter // Fachhochschule Rhein-Sieg. - 2011. - 03.. Februar 2020. - <http://www2.inf.h-brs.de/~pbecke2m/mobis2/script.pdf>.

Professional Java Native Interfaces with SWT/JFace [Buch] / Verf. Guojie Jackwind Li. - [s.l.] : Wrox, 2009. - Bd. 1. Edition.

Tabellentypen [Online] / Verf. SAP. - 26. Dezember 2019. - https://help.sap.com/doc/abapdocu_751_index_html/7.51/de-DE/abenddic_table_types.htm.

Verbuchungsbausteine [Online] / Verf. SAP. - 26. Dezember 2019. - https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenupdate_function_module_glosry.htm.

Anhang (Quellcodes)

Code 1 Funktionsbaustein ZE268_GETOPENPOSITION

```
FUNCTION ZE268_GETOPENPOSITION.  
  *"--  
  *"--Lokale Schnittstelle:  
  *"-- IMPORTING  
  *"--     VALUE(I_ORDERID) TYPE EBELN  
  *"-- EXPORTING  
  *"--     VALUE(E_POSLIST) TYPE ZE_268_DLVRDITMS_TBL  
  *"--  
  
  SELECT EKET~WEMNG FROM EKET INTO CORRESPONDING FIELDS OF TABLE E_  
  POSLIST WHERE EKET~EBELN = I_ORDERID.  
  
ENDFUNCTION.
```

Code 2 Funktionsbaustein ZE268_GETORDERITEMS

```
FUNCTION ZE268_GETORDERITEMS.  
  *"--  
  *"--Lokale Schnittstelle:  
  *"-- IMPORTING  
  *"--     VALUE(I_ORDERID) TYPE EBELN  
  *"-- EXPORTING  
  *"--     VALUE(E_ITEMLIST) TYPE ZE_268_OITEMS_TBL  
  *"--  
  ** Retrieve Order Positions  
  
  SELECT * FROM EKPO INTO CORRESPONDING FIELDS OF TABLE E_ITEMLIST  
  WHERE EKPO~EBELN = I_ORDERID.  
  
ENDFUNCTION.
```

```
FUNCTION ZE268_GETPROPOSALLIST .
*"-----
*" "Lokale Schnittstelle:
*"  IMPORTING
*"    VALUE(I_PREFIX) TYPE  STRING
*"  EXPORTING
*"    VALUE(E_ORDERLIST) TYPE  ZE_268_BLGKPF_TBL
*"-----

DATA L_PREFIX TYPE STRING.

CONCATENATE '%' I_PREFIX '%' INTO L_PREFIX.
SELECT EBELN, LIFNR, BEDAT FROM EKKO
      INTO CORRESPONDING FIELDS OF TABLE @E_ORDERLIST
      WHERE EKKO~EBELN LIKE @L_PREFIX.

ENDFUNCTION.
```

```

FUNCTION ZE270_GMVT.
*"-----
*""Lokale Schnittstelle:
*"  IMPORTING
*"      VALUE(I_ORDERID) TYPE  EBELN
*"      VALUE(I_POSNR)  TYPE  EBELP
*"      VALUE(I_MENGE)  TYPE  ERFMG
*"      VALUE(I_UOM)    TYPE  ERFME
*"  EXPORTING
*"      VALUE(E_MATDOC) TYPE  MBLNR
*"-----

Data: l_goodsmvt_header type BAPI2017_GM_HEAD_01,
      lt_goodsmvt_item type TABLE OF BAPI2017_GM_ITEM_CREATE
          WITH HEADER LINE,
      lt_return type TABLE OF BAPIRET2,
      lt_items type TABLE OF BAPIEKPO.

CALL FUNCTION 'BAPI_PO_GETDETAIL'
  EXPORTING
    PURCHASEORDER           = i_orderid
  TABLES
    PO_ITEMS                = lt_items
  .

l_goodsmvt_header-pstng_date = sy-datum.
l_goodsmvt_header-doc_date  = sy-datum.
lt_goodsmvt_item-entry_qnt  = I_MENGE.
lt_goodsmvt_item-entry_uom  = I_UOM.
lt_goodsmvt_item-move_type  = '101'.
lt_goodsmvt_item-plant      = 'HD00'.
lt_goodsmvt_item-stge_loc   = 'RM00'.
lt_goodsmvt_item-material   = 'R1-002'.
lt_goodsmvt_item-po_number  = I_ORDERID.
lt_goodsmvt_item-po_item    = I_POSNR.
lt_goodsmvt_item-mvt_ind    = 'B'.

APPEND lt_goodsmvt_item.

CALL FUNCTION 'BAPI_GOODSMVT_CREATE'
  EXPORTING
    GOODSMVT_HEADER           = l_goodsmvt_header
    GOODSMVT_CODE             = '01'
  IMPORTING
    MATERIALDOCUMENT          = E_MATDOC
  TABLES
    GOODSMVT_ITEM             = lt_goodsmvt_item
    RETURN                    = lt_return
  .

CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'.

ENDFUNCTION.

```