Dies ist der Titel der Bachelorarbeit und bitte ohne Rechtschreibfehler

Bachelorarbeit

für die Prüfung zum

Bachelor of Science

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Heidenheim

von

Michael Lustig

Septemper 2017

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	1234510, TINF2014MI
Ausbildungsbetrieb	Firma GmbH, Firmenort
Erstgutachter	Dipl.-Ing. (FH) Peter Pan
Zweitgutachter	Prof. Dr. Rolf Assfalg

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einführung

Wir leben in einer Zeit in der der Einsatz von Technik im Alltag aus unserem Leben nicht mehr weg zu denken ist. Technologien wie selbstfahrende Autos, die vor wenigen Jahren noch als Science Fiction klassifiziert wurden, sind nun realisierbar. In Fabriken werden selbst komplexere Tätigkeiten von Robotern übernommen und zu Hause gehört das Staubsaugen der Vergangenheit an, weil dies eine programmierte Maschine zuverlässig übernehmen kann.
In den verschiedensten Situationen erlebt unser Leben eine Bereicherung und Erleichterung durch neue Technologien. So auch im Gesundheitswesen. In Asien werden beispielsweise Roboter zur Altenpflege eingesetzt weil es dort wie auch in Deutschland an Pflegekräften mangelt. Hier in Deutschland ist der Einsatz moderner Technik im Gesundheitswesen allerdings noch nicht üblich, gäbe es doch so viele denkbare Fälle in denen kranken Menschen durch mehr Forschung auf diesem Gebiet sehr geholfen werden könnte.
Das Projekt hinter dieser Studienarbeit soll einen Teil dazu beitragen, das Leben kranker, genauer hötgeschädigter, Menschen zu erleichtern.

Hörgeschädigte Menschen haben ein kleines defiziet, welches allerdings dazu führt, dass Ihnen das Teilnehmen am sozialleben sehr erschwert wird: Ihr Gehör funktioniert nicht. Angefangen beim Schulbesuch, wo ein Gebärdesprachendolmetscher von nöten ist, damit sie dem Untericht folgen können über Dialoge XXXXXXX

Weil im Moment an einer Brille entwickelt wird, die mit einer Kamera bestückt ist, welche das gesehene Bild aufzeichnet, live verarbeitet und dem Träger erfasste Texte vorliest und ihn über vor ihm stehende Personen informiert, war die Idee nicht weit, diese Konvertierung in die entgegengesetzte Richtung vorzunehmen: Der von einem Mobiltelefon aufgenommene Ton wird auf das Vorkommen von Sprachbausteinen untersucht und die erkannte Sprache in Text konvertiert. Der Text wird auf einer mit dem Mobiltelefon verbundene AR-Brille ausgegeben.
So kann ein hörgeschädigter jedem Vortrag, jeder Unterichtsstunde problemlos folgen. Auch das kommunizieren mit Menschen, die der Gebärdensprache nicht mächtig sind, wird vereinfacht, da diese einfach in das Microphone sprechen müssen. Durch die Integration des Konvertierers in ein Mobiltelefon, würde es sogar möglich, dass ein Hörgeschädigter nun Telefonieren könnte, da die Stimme des Anrufers direkt verarbeitet werden könnte und dem Anwender live als Text auf die AR-Brillengläser projekziert werden können.

2 Struktur

Die Applikation ist untergliedert in zwei Hauptbestandteile. Einen Recorder, welcher die Tonaufnahme durchführt und die erkannte Sprache in Text umwandelt, und ein Printer, welcher sich zu einer Augmenter Reality Brille verbindet und den vom Recorder erkannten Text forformatiert und darauf ausgibt. Das Bindeglied zwischen diesen beiden Komponenten ist das User Interface mit Klassenname MainActivity. Die MainActivity wird durch Usereingaben gesteuert. Sie erstellt bei bedarf neue Recorder oder Printer Instanzen, startet beziehungsweise stoppt die Aufnahme bzw die AR-Ausgabe und informiert den AR-Printer über neu verfügbare Ausgabetexte. Diese drei Komponenten werden im folgenden näher erleutert.

2.1 Main Activity

Wie bereits erwähnt ist die MainActivity das Zentrum der Applikation, ein UML-Klassendiagramm ist in XXX abgebildet.
Die bestandteile der MainActivity:

1. Ein Spinner zur Wahl einer verfügbaren Sprachkonvertierungsoption und ein beschreibendes Textfeld
2. Ein Spinner zur Wahl eines verfügbaren AR-Printers und ein beschreibendes Textfeld
3. Ein Button zum Starten bzw. Stoppens einer Konvertierung
4. Ein SideView Menü, welches Einstellungsmöglichkeiten enthält wie zum Beispiel das Festlegen der gesprochenen Sprache
5. Eine Recorder Instanz vom Typ IRecorder. IRecorder ist ein interface, welches von allen Recorder-Implementierungen verwendet wird, um eine einheitlich API zu gewährleisten. Dieses Interface wird im Abschnitt XXX näher erleutert.
6. Eine AR-Printer Instanz vom Typ IPrinter. Iprinter ist ebenfalls ein Interface, welches von allen AR-Printern zur Gewährleistung einer einheitlichen API implementiert wird und wird in Abschnitt XXX näher erleutert.
7. Eine boolsche Hilfsvariable, die Auskunft darüber gibt, ob aktuell ein Konvertierungsvorgang im Gange ist.

8. Zwei Integer-Variablen, die Auskunft über den aktuell gewählten Recorder bzw Printer geben.

Zu 1. Und 2.:
Die beiden Spinner, enthalten alle aktuell supporteten Printer bzw Recorder. Der Inhalt der Spinner wird dynamisch genereiert, hierbei spielt die Klasse Constants eine tragende Rolle.
!!Genaueres über die initialisierung eines spinner arrays!!
Jedem der beiden Spinner ist eine OnItemSelectListener Instanz zugeordnet. Der OnItemSelectListener, ruft im Falle des Printer Spinners die Methode setPrinterMode(int mode, String description) und im Falle des Recorder Spinners die Methode setRecorderMode(int mode, String description) auf. Diese beiden Methoden initialisieren den gewünschten AR-Printer beziehungsweise den Recorder/Converter. Die Funktionalität dieser Methoden wird XXXX näher beschrieben.
Zu 3.:
Der OnClickListener der Buttons fragt die Hilfsvariable (7) ab und ruft abhängig von deren Wert die Methoden IRecorder.startRecording() und IPrinter.startPrinting() oder die Methode notifyRecorderStop() der eigenen Klasse auf.
4. Menü: XXXX

Methoden der Klasse MainActivity:

onCreate() und initialize_components():
In der onCreate()-Methode, welche beim start der Applikation ausgeführt wird, werden zunächst all Ihre bestandteile initialisiert durch die initialize_components()-Methode.
setPrinterMode(int mode, String description)
Diese Methode prüft zuerst, ob im Moment ein Konvertierungsvorgang im Gange ist, durch das abfragen der boolschen Hilfsvariable isRecording. Im Falle einer laufenden Konvertierung wird eine Meldung ausgegeben, die den User darauf hinweißt, dass der AR-Printer während einer Konvertierung nicht gewechselt werden kann. Sonst wird nichts weiter unternommen. Ist keine Konvertierung im Gange wird überprüft, ob der aktuelle Wert in der Hilfsvariable PRINTER_MODE mit dem Wert des ausgewählten Printers in der übergebenen Variable mode übereinstimmt, denn auch dann muss keine weitere Aktion ausgeführt werden.
Wurde ein anderer Printer gewählt, als der gerade verwendete, wird die Methode generatePrinter(int mode, String description) der Klasse PrinterFactory aufgerufen. Die Methode generatePrinter prüft durch ein Switch Statement, welcher ARPrinter vom User gewünscht ist und gibt eine Instanz des gewünschten Printers and die aufrufende Methode zurück. Ist der übergebene Integer-Wert der Factory unbekannt, wird standardmäßig eine Instanz

der Klasse TextPrinter erzeugt. Dieser erzeugt ein TextFeld in der MainActivity, in dem Speech to Text Conversion Ergebnisse ausgegeben werden.
setRecorderMode(int mode, String description):
Diese Methode hat den gleichen Ablauf, wie die Methode setPrinterMode(), mit dem einzigen Unterschied, dass die Methode generateRecorder der Klasse RecorderFactory aufgerufen wird, welche eine IRecorder Instanz zurück gibt. Auch die Recorder Factory hat für unbekannte Eingaben einen Standardwert. Sie gibt einen TextFieldRecorder zurück, welcher ebenfalls ein Textfeld erzeugt und die Eingaben and den gewählten AR-Printer sendet. Dieser Recorder verfehlt zwar den Sinn der Speech to Text Conversion, ist aber eine Erleichterung um die Funktionalität eines einzelnes AR-Printer Moduls zu testen.

receiveResult()
notifyStopRecord()

2.2 Constants Klasse

In der Klasse Constants werden globale Applikationskonstanten abgelegt. So gibt es für jede IRecorder-Implementierung einen Integer Wert, welcher den Recorder repräsentiert. Außerdem gibt es für jeden Recorder eine beschreibende String-Variable. Eine Map, welche beim start der Anwendung erzeugt wird, bringt die String-Werte mit den jeweiligen Int-Werten in Verbindung. Analog dazu besteht die selbe Struktur für Instanzen des Typs IPrinter, die unterstützten AR-Printer verwalten zu können.

2.3 Interface für AR-Glass-Printer

2.4 Interface für Speech-To-Text-Konvertierer

3 Graphical User Interfaces

3.1 dummy

4 Canvas Animationen

4.1 Dummy

5 AR-Glass-Printer Implementierungen

5.1 Die Klasse AbstractPrinter

5.2 Support der GlassUp AR Brille

5.3 Der TextField Printer

6 Speech-To-Text-Konvertierung

6.1 Der Android Speech Recognition Client

6.2 Speech-To-Text Konvertierung durch die Google Cloud API

Die Google Streaming Speech Cloud API definiert in ihrer Spezifikation sowohl die Art der Authentifizierung, als auch die verschiedenen Zugriffsmöglichkeiten. Für Java empfiehlt sich eine Authentifizierung über die Oauth2 Methode und die Kommunikation über Google Remote Procedure Calls (grpc).

Der Begriff Remote Procedures kommt aus dem Bereich verteilter Systeme, bei denen ein Programm auf einem Client-Rechner auf einem Server laufende Programmteile ausführen will.
Um dies zu realisieren, verfügt der Client über einen sogenannten method stub. Dieser Stub ist eine Dummymethode mit gleicher Signatur wie die produktive auf dem Server laufende Methode. Die Dummymethode ruft über einen bestehenden Channel zum Server dann die produktive Methode mit den übergebenen Parametern auf und wartet dann auf den Rückgabewert der Servermethode. Hat sie den Wert erhalten, gibt sie Ihn zurück an das aufrufende Unterprogramm. Der eine Remote Procedure ausführende Client kann beziehungsweise muss sich somit nicht selbst um die Client-Server-Kommunikation kümmern, wie es beim REST-Modell der Fall ist, und bekommt im Bestfall nicht einmal mit, dass die aufgerufene Methode extern ausgeführt wurde.

Google bietet für das RPC Konzept eine library ein, welche als Dependency zum Projekt hinzugefügt werden muss. Für die Speech API wird die library com.google.cloud.speech.v1beta1, die RPC Komponenten befinden sich im Package com.google.cloud.speech.v1beta1.SpeechGrpc. Der SpeecGrpc Stub kann über die static Methode SpeechGrpc.newStub(ManagedChannel channel) erzeugt werden. Diese Methode gibt eine Instanz der Klasse SpeechGrpc.SpeechStub zurück.

6.3 Der TextFieldRecorder / Konvertierer

Anhang

(Beispielhafter Anhang)

A. Assignment

B. List of CD Contents

C. CD

B. Auflistung der Begleitmaterial-Archivdatei

Die Archivdatei wurde zusammen mit der Online-Version dieser Ausarbeitung auf die Lernplattform hochgeladen.

```
⊢ Literature/
|    ⊢ Citavi-Project(incl pdfs)/   ⇒ Citavi (bibliography software) project with
|    |                               almost all found sources relating to this report.
|    |                               The PDFs linked to bibliography items therein
|    |                               are in the sub-directory 'CitaviFiles'
|    |    – bibliography.bib        ⇒ Exported Bibliography file with all sources
|    |    – Studienarbeit.ctv4      ⇒ Citavi Project file
|    |    ⊢ CitaviCovers/           ⇒ Images of bibliography cover pages
|    |    ⊢ CitaviFiles/            ⇒ Cited and most other found PDF resources
|    ⊢ eBooks/
|    ⊢ JournalArticles/
|    ⊢ Standards/
|    ⊢ Websites/
|
⊢ Presentation/
|    –presentation.pptx
|    –presentation.pdf
|
⊢ Report/
     – Aufgabenstellung.pdf
     – Studienarbeit2.pdf
     ⊢ Latex-Files/ ⇒ editable LaTeX files and other included files for this report
          ⊢ ads/                  ⇒ Front- and Backmatter
          ⊢ content/              ⇒ Main part
          ⊢ images/               ⇒ All used images
          ⊢ lang/                 ⇒ Language files for LaTeX template
```