PROTOTYP EINER SPRACHGESTEUERTEN SOFTWARE-ENTWICKLUNGSUMGEBUNG

PASCAL WITKOWSKI, MARKUS DAHM

Hochschule Düsseldorf
University of Applied Sciences
HSD

HOCHSCHULE DÜSSELDORF

Fachbereich Medien
Faculty of Media

MOTIVATION

Spracherkennung bietet vielen beeinträchtigten Menschen eine Alternative.

Die natürliche Sprache liegt dem Menschen am nächsten.

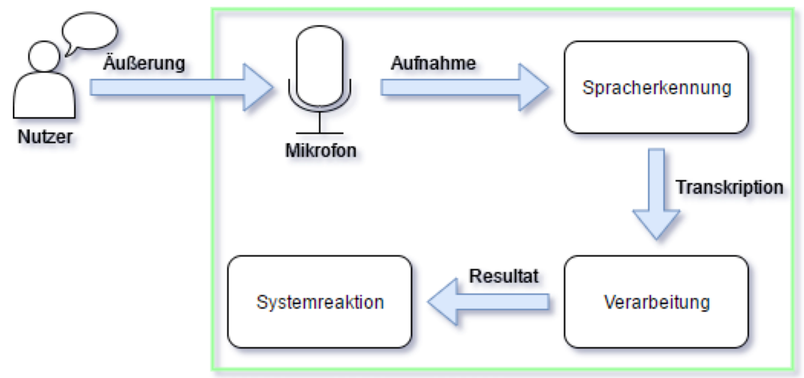
Sprechen ist potentiell schneller als die Eingabe per Tastatur.

Bei mobiler Nutzung sind Augen und Hände nicht immer verfügbar, z. B. in einem Automobil. Spracherkennung ist eine Möglichkeit, dem entgegenzuwirken.

FRAGESTELLUNG

Ist Sprachsteuerung für eine IDE im Vergleich zu klassischen Eingabemethoden effizienter, oder sollte sie nur als Ergänzung dienen?

KONZEPT



- De- und Aktivierung der **Sprachaufnahme** erfolgt durch Sprachsteuerungsmenü.
  - Start und Ende **automatisch detektiert**
  - **Ergebnis**: Transkription des Gesagten
  - **Auswertung**: Erfassung von Schlüsselwörtern und Bildung von **Key-Value-Paaren**. Wenn alle Paare einer Systemreaktion vollständig sind, kann diese ausgeführt werden. Werden dem System nicht genügend Informationen geliefert, kann es unvollständige Key-Value-Paare durch gezieltes Nachfragen füllen (**Slot-Filling)**
- Nutzer kommuniziert mit intelligentem **Assistenten (IntelliJay)**
  - Kontextbezogene Anweisungen & Feedback
  - Kommunikation grundsätzlich **nutzer-initiativ**. IntelliJay leitet jedoch Slot-Filling
  - **Kommunikation** sichtbar protokolliert
- Neben der Ausführung von Menüpunkten und weiteren Befehlen soll **Code-Diktation** möglich sein
  - Problem: Äußerungen manchmal mehrdeutig, gewünschte Aktion nicht immer eindeutig
  - ➢ **Diktier- und Befehlsmodus**. Nutzer ist im Diktiermodus, wenn Editor fokussiert ist. Durch „dictation“ / „command“ kann der Nutzer manuell wählen
- **Hilfefenster** neben Editor zeigt alle für aktuellen Modus relevante Äußerungen. So kann der Nutzer die Sprachsteuerung erlernen

DIKTIERMODUS

Spracheingaben werden in aktuelle Datei eingefügt.

- Bestimmte Wörter / Wortsequenzen werden **ersetzt**. Beispiel: „comment“ → / /
- Ersetzendes Wort: **Diktat-Wert.** Können auch durch **Kurzversionen** oder **semantische Beschreibungen** erzeugt werden: „curly“ / „block“ → `{ }`
- **Cursor** wird nach Einfügen an bestimmte Stelle gesetzt: `if(|)`. Wird grundsätzlich hinter dem Diktat-Wert und einem Leerzeichen platziert

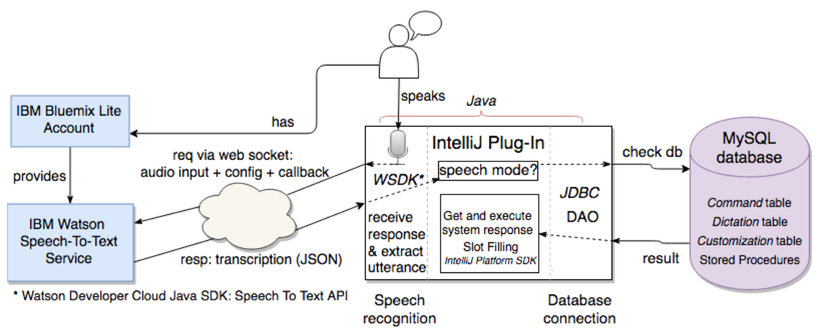
BEFEHLSMODUS

Unter den Befehlsmodus fallen nicht die Code-Eingabe betreffende Interaktionen.

- **Befehle** leiten Systemreaktion ein (z. B. Klassenerstellung wird mit Befehl „create“ eingeleitet)
- **Befehlsobjekte** liefern zu verwendenden Gegenstand, wie eine Klasse oder ein Package
- Manche Befehl-Befehlsobjekt-Paare benötigen Zusatzinformationen: **Properties.** (z. B. Klassenerstellung: eine Property für Name und Zielordner)
- **Datenmodell** definiert, welche Befehle mit welchen Objekten gesprochen werden können und welche Properties benötigt werden
- In zwei Fällen keine eindeutige Systemreaktion. IntelliJay beginnt **Slot-Filling**:
  - Befehl vom Nutzer festgelegt, jedoch wird Objekt benötigt. IntelliJay zeigt mögliche Objekte. Nutzer wählt Objekt
  - Properties werden nacheinander abgefragt. Nutzer versieht sie mit Wert.
- **Resultat:** Befehl-Befehlsobjekt-Paar mit ggf. Properties und deren Werten. Wird mit anschließend ausgeführter Implementierung einer Systemreaktion verknüpft

IMPLEMENTIERUNG

- Umsetzung als Plug-In für **IntelliJ IDEA**
- Spracherkennung: **IBM Watson Speech-To-Text** Service der IBM Bluemix Plattform
- Prüfung der Transkriptionen über JDBC anhand des als **MySQL-Datenbank** umgesetzten Datenmodells
  - Speziell erstellte Tabellen für beide Modi
  - Aus Resultat wird Systemreaktion ermittelt



HEURISTISCHE EVALUIERUNG

- **DIN EN ISO 9241-110** als Heuristiken
- Durchführung mit **Domain-Experten**
  - Aufgabenszenario
  - Beliebig Zeit für vorevaluatives Training
- Während und nach der Bearbeitung wurden erfasste Usability-Probleme notiert und kategorisiert

ERGEBNISSE

\+

- **Verwendung von Schlagworten im Diktiermodus** ein Vorteil gegenüber klassischen Eingabemethoden, da beliebige Konstrukte innerhalb von nur zwei Sekunden Antwortzeit eingefügt werden → in vielen Fällen **Zeitersparnis**
- **Verwendung semantischer Begriffe** entscheidender Vorteil der Sprachsteuerung → Nutzer kann abstrahiert von Programmiersprache Code einfügen
- Hilfefenster **gut gruppiert**. Erleichtert Einprägung der Interaktionsmöglichkeiten. Bietet gute **Lernförderlichkeit**. Wurde auf Dauer teilweise gar nicht mehr benötigt
- Dateierstellung als **schneller** empfunden, da eine automatische, einheitliche Namensgebung garantiert wird

\-

- Teils **suboptimale Qualität** der Transkriptionen
- Schlagworte werden umgewandelt, wenn man sie als **Bezeichner** verwendet
- Beim Löschen oder Öffnen muss bei inkorrekter Transkription der Befehl **erneut genannt** werden → verringert Effizienz und Zufriedenheit des Nutzers. Wiederholung des Dateinamens wäre günstiger

FAZIT

Der Prototyp kann Spracheingaben auswerten, um sowohl die IDE zu steuern, als auch Code zu diktieren.

Laut Evaluierung liegt das Potenzial der Sprachsteuerung beim Diktieren von Quellcode vor allem in der **Verringerung des Diktieraufwands** durch kurze Spracheingaben, die vordefinierte Konstrukte einfügen.

Durch semantisches Diktieren ist der Nutzer **näher am Konzept der Programmierung und abstrahiert von der Programmiersprache**. Bei Weiterentwicklung könnte so auch Quellcode mehrerer Programmiersprachen diktiert werden.

Durch die **einfache Erweiterbarkeit** der auslösbaren Systemreaktionen könnte bei weiterer Bearbeitung des Projektes auf Tastatur und Maus weitestgehend verzichtet werden.

Eine **sprecherabhängige Spracherkennung** würde die Usability höhere Erkennungsgenauigkeit steigern.

LITERATUR

Euler, S. (2006). *Grundkurs Spracherkennung*. Wiesbaden: Vieweg.

Neustein, A. (2010). *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*. Springer Science+Business Media.