Prototyp einer sprachgesteuerten Software-Entwicklungsumgebung

Pascal Witkowski, Prof. Dr.-Ing. Markus Dahm

Hochschule Düsseldorf, Fachbereich Medien

pascalwitkowski@gmx.de, markus.dahm@hs-duesseldorf.de

Zusammenfassung

Diese Arbeit befasst sich mit der Evaluation von Sprache als Eingabemethode für Entwicklungsumgebungen (IDEs). Zur Untersuchung der Fragestellung, ob Sprache die klassischen Eingabemethoden Tastatur und Maus ergänzen oder gar ersetzen kann, wurde für eine IDE eine Sprachsteuerung konzipiert. Dies ermöglicht eine barrierearme Bedienung. Der sprachgesteuerte Funktionsumfang kann beliebig erweitert werden. Ein Hilfefenster und ein Sprachassistent dienen der Unterstützung des Nutzers. Eine heuristische Evaluierung zeigt, dass Nutzer zunächst Gewöhnungszeit für die Sprachsteuerung benötigen, diese aber auf Dauer Zeit spart. Eine Spracherkennung mit höherer Erkennungsgenauigkeit kann die Effizienz und Usability noch steigern.

1 Motivation und Fragestellung

Die Verfügbarkeit an Rechenleistung und Netzwerkverbindungen verschiedener Devices hat zu einer ständig wachsenden Verbreitung von Applikationen geführt. Dabei ergaben sich neue Usability-Herausforderungen: Displays und Tastaturen sind klein und für manche schwierig zu handhaben. Bei der mobilen Nutzung ergeben sich Situationen, bei denen Augen und Hände nicht verfügbar sind, z. B. in einem Automobil. Spracherkennung ist eine Möglichkeit, den entgegenzuwirken (Neustein, 2010, S. 20). Sie bietet vielen beeinträchtigten Menschen eine Alternative (Neustein, 2010, S. 23, 32f.). Die natürliche Sprache liegt dem Menschen am nächsten und Sprechen ist potentiell schneller als die Eingabe per Tastatur (Neustein, 2010, S. 32). Als Fragestellung wird untersucht, ob Sprachsteuerung für eine IDE im Vergleich zu klassischen Eingabemethoden effizienter ist, oder nur als Ergänzung dienen sollte. Es wurde ein Konzept entwickelt, das Spracheingaben in Systemreaktionen der IDE *IntelliJ IDEA* umwandelt. Es wurde implementiert und hinsichtlich der Usability sowie der Fragestellung evaluier

Veröffentlicht durch die Gesellschaft für Informatik e. V. 2018 in
R. Dachselt, G. Weber (Hrsg.):
Mensch und Computer 2018 – Tagungsband, 02.–05. September 2018, Dresden.
 https://doi.org/10.18420/muc2018-mci-0334

2 Konzept

Die Aufnahme von Spracheingaben kann in der IDE durch ein *Sprachsteuerungsmenü* aktiviert und deaktiviert werden. Start und Ende einer Äußerung werden automatisch detektier So kann der Nutzer ohne weiteren Aufwand und frei sprechen. Das Ergebnis ist die Transkrip tion des Gesagten. Die Auswertung geschieht durch die Erfassung von Schlüsselwörtern und der Bildung von Key-Value -Paaren. Wenn alle Paare einer Systemreaktion vollständig sind, kann diese ausgeführt werden. Werden dem System nicht genügend Informationen zur Verfügung gestellt, so kann es unvollständige Key-Value-Paare durch gezieltes Nachfragen füllen. Dieser Vorgang heißt *Slot-Filling* (Euler, 2006, S. 155). Der Nutzer kommuniziert mit einem Assistenten, um ihm das Gefühl zu geben, mit einem intelligenten Kommunikationspartner zu interagieren. Angelehnt an der IDE wird ihm der Name IntelliJay verliehen. Er gibt kontextbezogene Anweisungen und Feedback. Die Mensch-Computer-Kommunikation ist grundsätzlich *nutzer-initiativ*. IntelliJay leitet jedoch den Slot-Filling-Prozess im obigen Fall. Der Kommunikationsverlauf wird sichtbar protokolliert, sodass er immer nachvollzogen werden kann.

Codieren ist eine grundlegende Nutzertätigkeit in einer IDE, weshalb neben der sprachgesteuerten Ausführung von Menüpunkten und weiteren Befehlen das Diktieren von Code möglich sein soll. Dadurch kommt es vor, dass Äußerungen mehrere Bedeutungen haben können und die gewünschte Aktion nicht eindeutig ist. Deshalb wurden zwei Modi definiert: Der *Diktier-* und *Befehlsmodus*. Dazwischen wird automatisch oder manuell gewechselt. Automatisch heißt, dass der Nutzer sich im Diktiermodus befindet, sobald der Editor fokussiert ist. Durch die Spracheingabe „dictation“ bzw. „command“ kann der Nutzer den Modus manuell wählen. Zur Unterstützung des Nutzers wird ein *Hilfefenster* neben dem Editor platziert. Dort werden alle für den aktuellen Modus relevanten Äußerungen angezeigt. So kann der Nutzer die Sprachsteuerung verfolgen und gut erlernen.

2.1 Diktiermodus

Im Diktiermodus werden Spracheingaben in die aktuelle Code-Datei eingefügt. Es müssen Schlagworte erkannt werden, die durch andere Zeichen ersetzt werden, ehe sie hinzugefügt werden. Teile der Transkription wie „comment“ werden nicht als das Wort, sondern hier als zwei Querstriche als Kommentar eingefügt. Hierfür wird der Begriff *Diktat-Wert* eingeführt. Bei den Schlagworten kann es sich auch um Wortsequenzen handeln. Der Diktat-Wert von „curly brackets“, „curly“ sowie von „block“ ist „{ }“. Diktat-Werte können so auch durch Kurzversionen oder semantische Beschreibungen erzeugt werden. Bei manchen Schlagworten wie „if“ wird zudem definiert, dass sich der Cursor nach Einfügen des Diktat-Wertes an einer bestimmten Stelle befindet: „if(|)“. Grundsätzlich befindet sich der Cursor hinter dem Diktat-Wert und einem Leerzeichen. Die Transkriptionen werden vor dem Einfügen nach Diktat-Werten geprüft. Dies geschieht Wort für Wort, damit der Nutzer natürlich sprechen kann.

2.2 Befehlsmodus

Unter den Befehlsmodus fallen nicht die Code-Eingabe betreffende Interaktionen. Zunächst werden die Begriffe *Befehl* und *Befehlsobjekt* eingeführt. *Befehle* leiten eine Systemreaktion ein. Die Erstellung einer Klasse wird mit dem Befehl „create“ eingeleitet. Zusätzlich wird die Information benötigt, *was* erstellt wird. Dies liefern *Befehlsobjekte* , welche den zu verwendenden Gegenstand, wie eine Klasse oder ein Package, bestimmen. Bei manchen Befehl-Befehlsobjekt-Paaren werden Zusatzinformationen benötigt: *Properties*. Bei der Erstellung einer Klasse ist eine Property der Name und der Zielordner. In einem Datenmodell wird definier welche Befehle mit welchen Objekten gesprochen werden können und welche Properties be nötigt werden. So ist dem Prototyp bekannt, ob aus einer Transkription eine Systemreaktio ermittelt werden kann. Dies ist der Fall, wenn in der Transkription ein Befehl und ein jeweili ges Objekt vorliegt und benötigte Properties einen Wert besitzen. Manche Befehle reiche allein für die Auslösung einer Systemreaktion aus.

In zwei Fällen resultiert aus einer Transkription keine eindeutige Systemreaktion und IntelliJa beginnt das Slot-Filling *Im ersten Fall* ist der Befehl vom Nutzer festgelegt, jedoch wird das Objekt benötigt. IntelliJay legt dem Nutzer alle Objekte des Befehls dar, der nun eine Äußerung tätigen muss, bis diese mit einem Objekt übereinstimmt. *Im zweiten Fall* werden die Properties nacheinander abgefragt. Wenn diese vom Nutzer mit einem Wert versehen wurden, wird eine Aktion ausgelöst. Als *Resultat* liegt ein Befehl-Befehlsobjekt-Paar mit ggf. Properties und deren Werten vor. Dieses wird mit einer anschließend ausgeführten Implementierung einer Systemreaktion verknüpft. Nacheinander werden alle Befehle mit dem aus dem Resultat abgeglichen. Der Befehl, der im Resultat ist, wird ausgewählt. Wenn dieser mit Objekten in Beziehung steht, werden diese auch nacheinander mit dem Resultat abgeglichen.

3 Implementierung

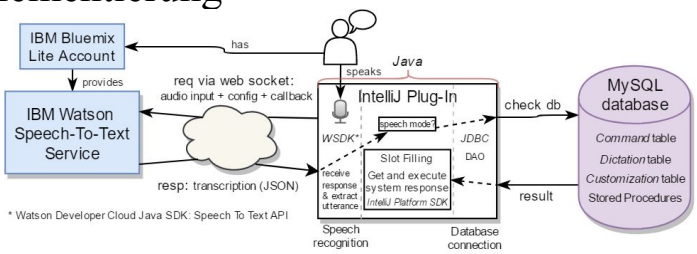

Abbildung 1 – Software-Architektur des Prototyps

Das Konzept wurde als Plug-In für IntelliJ IDEA implementiert. Als Spracherkennung wurde der IBM Watson Speech-To-Text Service der IBM Bluemix Plattform und dessenJava-SDK im Plug-In verwendet. Die Transkription wird über JDBC in einem DAO anhand des als MySQL-Datenbank umgesetzten Datenmodells überprüft. Dabei werden für beide Modi speziell erstellte Tabellen verwendet. Aus denResultaten werden dann die passenden Systemreaktionen ermittelt.

4 Heuristische Evaluierung

Der Prototyp wurde mithilfe von Heuristiken durch Domain-Experten evaluiert. Diese erhielten ein Aufgabenszenario und beliebig Zeit für ein vorevaluatives Training. Während und nach der Bearbeitung wurden die erfassten Usability-Probleme notiert und kategorisiert. Die Auswertung zeigt, dass die Verwendung von Schlagworten im Diktiermodus einen Vorteil gegenüber klassischen Eingabemethoden darstellt, da beliebige Konstrukte innerhalb von nur zwei Sekunden Antwortzeit eingefügt werden. Dies erweist sich in vielen Fällen als Zeitersparnis. Die Verwendung semantischer Begriffe wurde als der entscheidende Vorteil der Sprachsteuerung bezeichnet, da man so abstrahiert von der Programmiersprache Code einfügen kann. Als Wunsch wurde die Möglichkeit der Erstellung eigener Schlagworte geäußert. Als Problem wurde bewertet, dass Schlagworte ebenfalls umgewandelt werden, wenn man sie als Bezeichner verwendet. Kritisiert wurde die teils suboptimale Qualität der Transkriptionen. Dadurch ist oft ein Undo nötig. Hier wird das letzte eingefügte Wort gelöscht, jedoch sollte nach Meinung der Experten die letzte Transkription gelöscht werden, um eine schnellere Korrektur zu gewährleisten. Die Dateierstellungwurde als schneller empfunden, da eine automatische, einheitliche Namensgebung garantiert wird. Beim Löschen oder Öffnen muss noch bei inkorrekter Transkription der Befehl und das Slot-Filling neu gestartet werden, was die Effizienz und Zufriedenheit des Nutzers verringert. Eine einfache Wiederholung des Dateinamens wäre weitaus günstiger. Die kontextbezogene Hilfe wurde als gut gruppiert empfunden und erleichtert die Einprägung der Interaktionsmöglichkeiten. Sie bietet somit eine gute Lernförderlichkeit. Auf Dauer wurde sie teilweise gar nicht mehr benötigt.

5 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 einer 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. Außerdem sollte eine sprecherabhängige Spracherkennung verwendet werden, da durch eine höhere Erkennungsgenauigkeit die Usability gesteigert werden kann.

Literaturverzeichnis

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

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