

Benutzerschnittstellen für Sprache und Stimme

Raphael Wimmer

wimmer@informatik.uni-muenchen.de

Betreuer Matthias Kranz, matthias@hcilab.org

Universität München,

Amalienstrasse 17, 80333 Munich, Germany

Zusammenfassung Sprach-Benutzerschnittstellen ermöglichen es dem Menschen, in natürlicher Sprache mit dem Computer zu interagieren. Für mobile Endgeräte wie Mobiltelefone oder Persönliche Digitale Assistenten (PDAs) bietet Sprache eine kostengünstige und anwenderfreundliche Benutzerschnittstelle. Die grundlegenden Entwurfsprinzipien unterscheiden sich aber erheblich von denen für grafische Benutzerschnittstellen. Dieser Arbeit stellt vier interessante Forschungsarbeiten über auditive Benutzerschnittstellen vor. Diese behandeln Entwurfskriterien für Sprachdialogsysteme, Techniken für Sprach-Benutzerschnittstellen in mobilen Endgeräten und alternative Eingabemöglichkeiten durch nonverbale Laute. Eine Zusammenstellung von grundlegenden Entwurfsprinzipien für Sprach-Benutzerschnittstellen ergänzt die Fallstudien. Zusätzlich zu den akademischen Arbeiten liefert diese Arbeit eine Marktübersicht über kommerzielle und freie Sprachsoftware und eine kurze Vorstellung von VoiceXML, der Standard-Beschreibungssprache für Sprachdialogsysteme.

1 Einleitung

Noch vor der Erfindung der Schrift war Sprache die erste Möglichkeit des Menschen, mit Anderen zu kommunizieren. Bei der Interaktion mit dem Computer führt die gesprochene Sprache allerdings ein Schattendasein. Grafische Benutzerschnittstellen (Graphical User Interfaces, GUIs) sind dagegen allgegenwärtig. Sie haben kein Gegenstück in der realen Welt (außer andere Maschinen). Insofern war es für den Menschen nur natürlich, seine Arbeitsweise an die der Anwendung anzupassen. Bei Sprach-Benutzerschnittstellen (Speech User Interfaces, SUIs) stehen die Entwickler aber vor der Herausforderung, deren Struktur an die der menschlichen Sprache anzupassen. Denn genauso wie man von einem anderen Menschen erwartet, dass er einfache Befehle und Aussagen versteht, erwartet der Benutzer implizit, dass auch ein Sprachdialogsystem ihn versteht, wenn er sagt "Nein, nicht diese Email öffnen, die andere von, äh, von dem Vertreter, der gestern da war". Diese Erwartungshaltung stellt die Entwickler von Sprach-Benutzerschnittstellen vor die Herausforderung, ein System, das bei weitem nicht die geistige Leistungsfähigkeit eines Menschen hat, so zu gestalten, dass es als Dialogpartner agieren kann. Aus diesem Grund sind Sprach-Benutzerschnittstellen für die Usability (Benutzerfreundlichkeits)-Forschung ein fruchtbares Gebiet. Die vier hier vorgestellten Arbeiten zeigen ein breites Spektrum an Ideen, um natürlichsprachige Kommunikation mit dem Computer benutzerfreundlich zu gestalten. Vor allem für mobile Endgeräte bieten sich Sprach-Benutzerschnittstellen an, da sie keinen Platz für Bildschirm und

Tastatur benötigen und auch parallel zu anderen Tätigkeiten benutzt werden können.

Diese Arbeit geht vor allem auf die folgenden vier wissenschaftlichen Veröffentlichungen ein. Yankelovich, Levow und Marx berichten in *"Designing SpeechActs: Issues in Speech User Interfaces"* [3] von ihren Erfahrungen bei Design, Implementierung und Evaluation einer Sprach-Benutzerschnittstelle für E-Mail, Kalender, Börsenkurse und Wetterbericht. In *"Nomadic Radio: Speech & Audio Interaction for Contextual Messaging in Nomadic Environments"* [4] stellen Nitin Sawhney und Chris Schmandt vom MIT Media Lab ihre Designprinzipien für eine mobile sprachgesteuerte Kommunikationsplattform dar und gehen besonders auf kontextbasierte Benachrichtigungen und unterschwellige Informationsdarstellung ein. *Voice User Interface Principles for a Conversational Agent* [1] von Ross, Brownholtz und Armes stellt schließlich allgemeine Design-Regeln für Sprachdialogsysteme auf. Takeo Igarashi und John F. Hughes beschreiben in *"Voice as Sound: Using Non-verbal Voice Input for Interactive Control"* [2], wie man mit Lauten statt durch Sprache Geräte steuern kann.

Ein ausführlicher Überblicksartikel zum Thema Sprachdialogsysteme ist "Spoken Dialogue Technology: Enabling the Conversational User Interface" [6] von Michael F. McTear.

2 Sprach-Benutzerschnittstellen - Begriffserklärung und Eingrenzung

2.1 Sprach-Benutzerschnittstellen

Sprach-Benutzerschnittstellen (Voice User Interfaces / Speech User Interfaces, SUI) stellen dem Benutzer Methoden zur Verfügung, um mit dem Computer über gesprochene Sprache zu interagieren.

2.2 Sprachdialogsysteme

Sprachdialogsysteme (Spoken Dialogue Systems) sind Computersysteme, mit denen Menschen interagieren können und in denen natürliche Sprache eine wichtige Rolle für die Kommunikation spielt [6]. Sie verwenden eine Sprach-Benutzerschnittstelle, um den Zugriff auf eine Anwendung oder Datenquelle zu realisieren.

Ein- und Ausgabe können sich unterschiedlich stark an die natürliche Sprache anlehnen. Aus praktischen Gründen werden zum Beispiel oft nur "Ja", "Nein" und die Zahlen von Null bis Neun zur Eingabe verwendet. Die Ausgabe kann auch lediglich aus einer Aktion bestehen ("Computer, medizinisch-holographisches Notfallprogramm aktivieren").

Ein Sprachdialogsystem besteht üblicherweise aus sechs Komponenten (Abbildung 1).

Spracherkennung. Eine Spracherkennungssoftware (Speech Recognition Engine) wandelt gesprochene Sprache in Text um (z.B. "Überweise 500 Euro

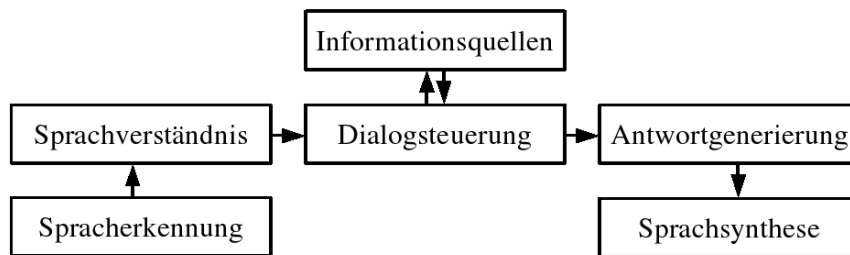


Abbildung 1. Üblicher Aufbau eines Sprachdialogsystems

an..."). Dafür werden meist Hidden Markov Models (HMM) [10] oder Dynamic Time Warping (DTW) [9] eingesetzt. Spracherkennung ist immer noch das Sorgenkind der Entwickler, da die durchschnittliche Erkennungsrate heutiger Implementierungen unter realistischen Bedingungen bei 80%-90% liegt. Sind laute Hintergrundgeräusche vorhanden - bei Mobiltelefongesprächen keine Seltenheit - sinkt die Erkennungsrate sogar bis auf 50%.

Sprachverständnis. Nachdem die Äußerung des Benutzers als Text vorliegt, kann ein Syntaxanalysator (Parser) diese auf bedeutungsvolle Inhalte analysieren. Der Algorithmus muss feststellen, welche Daten bzw. Befehle der Text enthält (z.B. Aktion="Überweisung", Betrag="500,00", ...)

Dialogsteuerung. Die Dialogsteuerung erhält die aufbereiteten Daten vom Syntaxanalysator und verarbeitet diese. Dabei können gespeicherte Daten verändert werden (z.B. Kontodaten) und Ausgaben vorbereitet werden (z.B. eine Bestätigungsmeldung).

Informationsquellen. Die Dialogsteuerung hat Lese- und Schreibzugriff auf eine Datenbank. Diese kann Informationen liefern (z.B. Benutzername="Herr Meyer") und auch Datenänderungen entgegennehmen (z.B. Benutzer[4534].Konto[00].Betrag = "1242,00")

Antwortgenerierung. Nachdem die Dialogsteuerung eine passende Antwort auf die Anfrage ermittelt hat (z.B. Status="ÜberweisungOK"), wird ein Antworttext erzeugt (z.B. "Vielen Dank für Ihre Überweisung. Was kann ich sonst noch für Sie tun?").

Sprachsynthese. Der Antworttext wird von einer "Text-To-Speech(TTS)-Engine" in eine Sprachausgabe umgewandelt. Entweder werden dazu einzelne Phoneme (atomare Bestandteile gesprochener Sprache) zu Wörtern zusammengesetzt, oder vorgefertigte Sprachbausteine aneinandergehängt. Alternativ kann auch ein vorher aufgenommener Text ausgegeben werden.

2.3 Beschränkung auf Usability-Forschung

Sprachdialogsysteme sind ein komplexes Thema mit vielen sehr unterschiedlichen Problemstellungen. Während vor allem die Spracherkennung und Sprachsynthese in den letzten zehn Jahren schon eine erhebliche Entwicklung durchgemacht haben, erhält inzwischen auch ein anderer Aspekt von Sprachdialogsystemen verstärkte Beachtung: Die Benutzerfreundlichkeit (Usability) der Systeme. Sprach-Benutzerschnittstellen gehorchen gänzlich anderen Prinzipien als grafische Benutzerschnittstellen. Zum Beispiel sieht der Benutzer während einer Sitzung mit einem Sprachdialogsystem nicht, welche Felder eines Formulars er schon ausgefüllt hat. Er muss sich also ein mentales Modell des Formulars bilden. Diese und weitere Herausforderungen im Design von Sprach-Benutzerschnittstellen behandeln die Arbeiten, die im Folgenden vorgestellt werden.

3 SpeechActs: Eine Sprach-Benutzerschnittstelle für Messaging und Kalender

3.1 Was ist SpeechActs?

SpeechActs war 1995 ein Forschungsprototyp für Sprachdialogsysteme bei Sun Microsystems. Das System stellt dem Benutzer über ein Sprachdialogsystem den Zugriff auf seine E-Mails, seinen Kalender, Börsen- und Wechselkurse und den Wetterbericht zur Verfügung. Die Sprachausgabe arbeitet mit einer synthetisierten Computerstimme.

Beispielablauf:

SpeechActs: Sie haben 14 neue und 47 alte Nachrichten. Die erste neue Nachricht ist von Eric Baatz, Betreff "Bericht fertig".

Benutzer: Will ich hören.

SpeechActs: "Der erste Entwurf wartet auf deine Kommentare. Eric"

Benutzer: Antwort inklusive Originalnachricht.

SpeechActs: Bereite Nachricht an Eric Baatz vor. Bitte sprechen Sie nach dem Piepton. Wenn Sie fertig sind machen Sie eine Pause von mehreren Sekunden

Benutzer: Eric, ich schick's dir heute nachmittag.

SpeechActs: Sagen Sie Abbrechen, Senden oder Vorschau.

Benutzer: Senden.

SpeechActs: Nachricht gesendet. Was nun?

Benutzer: Nächste Nachricht.

SpeechActs: Nachricht zwei von Stuart Adams, Betreff: "Können wir uns um drei treffen?".

Benutzer: Wechsle zum Kalender.

...

3.2 Evaluation

Im Laufe der Entwicklung wurden iterative Usability-Tests durchgeführt. Als Versuchsaufbau diente ein Raum mit Telefon, das mit dem SpeechActs-System verbunden war. Den Teilnehmern wurde aufgetragen, innerhalb von 20 Minuten 22 Aufgaben über das SpeechActs-System zu erledigen. Diese umfassten das Lesen und Beantworten von Mail, das Aufrufen von Kalendereinträgen, Börsenkursen und Wetterberichten. Ein Problem, das dabei festgestellt wurde, war die nicht zufriedenstellende Qualität der Spracherkennung. Auch waren die Benutzer mit dem Mail-System von Sun vertraut, das seine Eingaben über die Wähltastatur entgegennahm. Oft wurde deshalb von den Teilnehmern angemerkt, dass eine kurze Tastensequenz viel schneller gewesen wäre als die Sprach-Kommandos.

3.3 Herausforderungen

Im Laufe der Entwicklung und Evaluation wurden einige Prinzipien klar, die beachtet werden müssen, um ein benutzerfreundliches Sprachdialogsystem zu bauen.

Konversation imitieren Um die Benutzung einfach zu gestalten, muss sich der Computer an das menschliche Konversationsverhalten anpassen, anstatt ihm die traditionelle Arbeitsweise einer grafischen Anwendung aufzuzwingen.

So sollte es ein Sprachdialogsystem vermeiden, den Benutzer explizit zu einer Eingabe aufzufordern, sondern ihm die freie Wahl über seine Aktionen lassen. Die führt allerdings auch zu einer hohen Zahl unterschiedlicher Situationen und Verzweigungen in der Gesprächsführung, die von der Software behandelt werden müssen.

Gesprächstempo Besonders wichtig für regelmäßige Benutzer ist die Geschwindigkeit, mit der er seine Aufgaben über eine Sprachschnittstelle erledigen kann. Ein Benutzer, der zum ersten Mal mit einem Dialogsystem interagiert, ist zweifellos dankbar für ausführliche Fragen und Antworten des Systems. Bei regelmäßiger und intensiver Nutzung eines Dialogsystems werden häufig vorkommende Ausgaben des Dialogsystems wie "Sie haben eine neue Nachricht erhalten" dem Benutzer schnell lästig. Aus diesem Grund muss ein Dialogsystem verschiedene Verfahren implementieren, um die sich in der Interaktionsgeschwindigkeit an den Benutzer anzupassen. Für SpeechActs entwickelten Yankelovich et al. diverse Methoden, den Gesprächsfluss zu beschleunigen.

Einmischen (Barge-In) - In den Testreihen versuchten Benutzer häufig erfolglos, bekannte Ansagen zu unterbrechen, um einen weiteren Befehl zu geben. Die Entwickler fügten daraufhin eine als "Barge-In" bezeichnete Funktionalität hinzu, die es dem Benutzer erlaubt, neue Befehle zu geben, während das System noch spricht. Die Sprachausgabe wird dann abgebrochen.

Freie Wahl der Wiedergabegeschwindigkeit - Eine weitere Methode um den Dialogfluss zu beschleunigen ist, die Geschwindigkeit der Sprachausgabe zu erhöhen. Damit können Benutzer bekannte Ansagen oder unwichtige Nachrichten schneller und wichtige Informationen langsamer abspielen.

Komprimierte Darstellung Um die Geschwindigkeit von Interaktionen zu erhöhen, sollten Ansagetexte möglichst kurz ausfallen. Dies kann unter anderem geschehen, indem eine Antwort mit dem nächsten Gesprächsschritt verbunden wird. Auch eine ausklingende Darstellung (tapered presentation) ist sinnvoll. Dabei werden sich wiederholende Satzteile weggelassen:

Momentan wird Sun Microsystems bei 25 gehandelt, einen halben Punkt mehr als gestern. Silicon Graphics liegt bei 23 1/2, um ein Halb gefallen. IBM bei 69, ein Achtel gestiegen.

Sprachkonventionen Beschleunigend auf den Gesprächsverlauf wirkt sich auch die Nutzung von Sprachkonventionen aus, um dem Benutzer mitzuteilen, was er als nächstes sagen könnte:

SpeechActs: "*Sie haben heute drei Termine*".
Benutzer: "Und was habe ich morgen?"

Tastaturkürzel (Shortcuts) - Besonders im Vergleich zu Systemen, die Eingaben über die Wähltastatur entgegennehmen wirken Sprachdialogsysteme oft langsam. Tastaturkürzel für häufig benutzte Aktionen beschleunigen hier die Arbeit und reduzieren gleichzeitig die Belastung für die Stimme.

Audio-Icons - Für fortgeschrittene Benutzer können zusätzlich "Audio-Icons" implementiert werden, die lange, gesprochene Ansagen ersetzen. Anstelle der Ansage "Lösche E-Mail" kann dann ein Soundclip mit einem "Papierzerknüllen"-Geräusch treten.

Geschwindigkeit des Systems optimieren - Bei der Implementierung von SpeechActs 1995 war die Hardware noch nicht leistungsfähig genug, um die Spracherkennung in Echtzeit durchzuführen. Dies resultierte in einem etwas langsameren Gesprächstempo, das den Benutzern als störend und unnatürlich auffiel. Dieses Problem ist mit heutiger Hardware nicht mehr gegeben.

Vokabular Bei grafischen Anwendungen muss der Benutzer sich - mal mehr, mal weniger - an die Eingabe- und Ausgabemöglichkeiten der grafischen Benutzerschnittstelle anpassen. Im Gegensatz dazu erwartet der Benutzer, dass ein Sprachdialogsystem sein Vokabular versteht und spricht.

Idealerweise lehnt sich das Vokabular des Sprachdialogsystems an das zwischenmenschlicher Gespräche an. Dabei reicht es aber nicht aus, alle Äußerungen des Benutzers zu verstehen. Vielmehr muss der Computer implizite, kontextbezogene Informationen auswerten können.

Ein grafischer Kalender hat kein Konzept für den Begriff "nächsten Montag". Der Benutzer übersetzt dies selbst in ein entsprechendes Datum, das er

dem Kalender mitteilt. Vom Dialogsystem hingegen wird erwartet, dass es diese Information selbständig umwandelt. Aber nicht nur Daten, auch Befehle sind kontextsensitiv und können sich auf vorhergehende Antworten beziehen:

Benutzer: "Welche Termine habe ich morgen?"
SpeechActs: "... "
Benutzer: "Und am Freitag?"

Trotz der oben beschriebenen Bemühungen, die Interaktion möglichst stromlinienförmig und kompakt zu gestalten, verwendet SpeechActs einige Standardfloskeln wie "Auf Wiedersehen" am Ende eines Dialogs. Dies weicht die knappe Sprache etwas auf und nähert die Dialoge an reale Konversation an.

Umwandlung von grafischen in Sprach-Benutzerschnittstellen Im Gegensatz zur Interaktion mit grafischen Benutzerschnittstellen fehlt es dem Benutzer von Sprachdialogsystemen an visuellem Feedback. Bei einer Grafikanwendung weiß er jederzeit, in welchem Kontext oder Modus er sich gerade befindet. Da Sprache allerdings eine sehr stark temporale Natur hat, vergisst der durchschnittliche Benutzer schnell, an welcher Position im Dialog er sich gerade befindet.

Neben der Herausforderung, die menschliche Konversation zu imitieren, ist es daher besonders wichtig, dass der Benutzer sich ein mentales Modell der Interaktionsmöglichkeiten bilden kann. Dazu gehört, dass er weiß, wo im Dialog er sich gerade befindet und was seine Optionen im Moment sind. Diese Designherausforderung manifestiert sich in vielen Details:

Kontinuität Besonders verwirrend ist, wenn sich der Dialogkontext vom Benutzer unbeabsichtigt ändert.

Im SpeechActs-System trat dieses Problem beispielsweise im Posteingang auf: Im grafischen E-Mail-Programm das bei Sun verwendet wird sind die Nachrichten im Posteingang durchnummeriert. Wird dort eine E-Mail als gelesen markiert, verschwindet sie aus dem Posteingang und die restlichen Nachrichten werden wieder von vorne durchnummeriert, so dass keine Lücken entstehen.

Auch im SpeechActs-System konnte der Benutzer auf Nachrichten im Posteingang über deren laufende Nummer zugreifen. Wenn eine E-Mail als gelesen markiert wurde und aus dem Posteingang verschwand, änderten sich auch hier für die anderen Nachrichten die zugeordneten Nummern. Da der Benutzer aber bei Sprachdialogsystemen die Betreffzeilen der Nachrichten schlechter scannen kann als in einem grafischen E-Mail-Client, ist er stark darauf angewiesen, auf eine Nachricht über ihre Nummer zugreifen zu können.

Yankelovich et al. lösten dieses Problem, indem sie gelesene Nachrichten nicht sofort aus dem Posteingang entfernen. Stattdessen bekommen diese ein "Gelesen"-Attribut gesetzt. Wenn der Benutzer später wieder auf die bereits gelesene Nachricht zugreift, informiert ihn SpeechActs lediglich darüber, dass er diese Nachricht bereits gelesen hat. Erst bei Beendigung einer Sitzung werden die gelesenen Nachrichten aus dem Posteingang entfernt.

Subdialoge Grafische Benutzerschnittstellen verwenden oft Dialogfenster und Assistenten (Wizards), um komplexere, eigenständige Aufgaben auszuführen. Auch in SpeechActs kommen solche Subdialoge vor, zum Beispiel, wenn der Benutzer auf eine E-Mail antworten will. Diese sind allerdings nicht so deutlich von der restlichen Benutzerschnittstelle abgehoben, wie in grafischen Benutzerschnittstellen. Ziel war es nun, dem Benutzer mitzuteilen, wenn er einen Subdialog erfolgreich abgeschlossen hatte und wieder zur normalen Programmoberfläche zurückkehrte. Die Sun-Forscher versuchten es erst mit einem Signalton, wenn die Antwort verschickt wurde. Dies wurde dann zugunsten eines gesprochenen "Was nun?" fallengelassen.

Status-Signalisierung Wenn das System nach einem Befehl schweigt, weiß der Benutzer nicht, ob es gerade arbeitet, den Befehl nicht verstanden hat oder sogar abgestürzt ist. Um die Tätigkeit des Systems hörbar zu machen, kann - ähnlich der Sanduhr in grafischen Benutzeroberflächen - ein Warteklang oder eine Wartemelodie abgespielt werden.

Dialogsteuerung durch den Computer Während grafische Benutzeroberflächen dem Benutzer oft einen festen Weg vorgeben, um eine Aufgabe zu erfüllen, muss sich eine Sprach-Benutzerschnittstelle dem Benutzer unterordnen, um akzeptiert zu werden.

Ein häufig vorkommendes Verhalten bei Benutzern von Sprachdialogsystemen ist, nicht immer explizit auf Fragen zu antworten. Auf die Frage "Ihre Nachricht wird an Matt Marx verschickt. Okay?" antworteten im Test viele entweder gleich mit einem neuen Befehl ("Nächste Nachricht") oder hängten den nächsten Befehl an die Antwort an ("Ja, nächste Nachricht"). Dieses Verhalten wurde auch von anderen Forschern festgestellt [5].

SpeechActs behilft sich hier mit einer toleranten Strategie: Als Antwort auf eine Frage sind auch "Ja, ..." und "Nein, ..." erlaubt. Wenn keine explizite Bestätigung mit "Ja" oder "Nein" erfolgt, macht SpeechActs "das Richtige", das heißt, das Dialogsystem führt dann automatisch die Aktion aus, die am wenigsten destruktiv ist. Wenn wichtige Fragen explizit beantwortet werden sollen, greift SpeechActs auf anleitende Aufforderungen (Directive Prompts) zurück, die dem Benutzer die Antwort vorschreiben: "Sagen Sie Ja, Nein oder Abbrechen!"

Erkennungsfehler Das größte Problem von Sprachdialogsystemen sind die Mängel der Spracherkennungssoftware. Dadurch kommt es zu inkonsistentem Verhalten, weil der Computer einmal einen Befehl versteht, beim nächsten Mal aber nicht.

Denn wenn der Benutzer einen Befehl gibt und eine Aktion ausgeführt wird, stellt er einen Ursache-Wirkung-Zusammenhang her. Ein inkonsistentes Verhalten des Rechners verhindert somit, dass der Benutzer sich ein konzeptuelles Modell (conceptual model) baut, wie die Anwendung reagieren wird.

Generell werden drei Arten von Erkennungsfehlern unterschieden:

Zurückweisung (Rejection Error) Das Dialogsystem kann keine gültige Eingabe erkennen und verweigert den Dienst. Dieser Mauereffekt (brick wall effect) führt sehr schnell zu Frustration beim Benutzer. Abhilfe schafft hier eine schrittweise Hilfestellung (progressive assistance): Beim ersten Erkennungsfehler gibt das System eine kurze Fehlermeldung aus, wenn der Fehler wiederholt gemacht wird bietet das System ausführlichere Hilfestellung an.

SpeechActs: Was haben Sie gesagt?

SpeechActs: Wie bitte?

SpeechActs: Wie bitte? Formulieren Sie die Aussage bitte um..

SpeechActs: Ich verstehe Sie nicht. Sprechen Sie deutlich, ohne übermäßig zu betonen.

SpeechActs: Es klappt immer noch nicht. Sprechen Sie bitte nach dem Piepton.

Wenn Fehler wiederholt auftreten, ist das oft ein Zeichen für Äußerungen, die nicht im Vokabular des Systems enthalten sind. Deshalb werden die Benutzer aufgefordert, ihren Befehl umzuformulieren. Die progressive Hilfestellung sorgt damit auch dafür, dass der Benutzer die richtigen Befehle lernt.

Ersetzung (Substitution Error) Die Zurückweisung von Befehlen ist "nur" störend, wohingegen das falsche Verständnis eines Befehls großen Schaden anrichten kann. Zum Beispiel fragt der Benutzer vielleicht nach dem Wetter am Niederrhein, das System versteht aber "Auf Wiedersehen" und beendet die Sitzung. Deshalb ist es sinnvoll, den Benutzer bei kritischen Aktionen noch einmal um Bestätigung zu bitten. Um den Benutzer nicht übermäßig mit Rückfragen zu belästigen, kann man wieder zwei Fälle unterscheiden: Wird ein Befehl als Frage nach Auskunft verstanden ("Wie ist das Wetter in Athen?"), kann die richtige Erkennung implizit in der Antwort verifiziert werden. Dazu baut das Dialogsystem Teile der Frage in die Antwort ein ("Das Wetter in Athen ist sonnig bei 24 Grad."). Wird hingegen ein kritischer Befehl erkannt, der den Verlust von Daten zur Folge haben kann, fragt das System noch einmal explizit rück beziehungsweise entscheidet sich im Zweifelsfall für die nicht-destruktive Antwort.

Einfügen (Insertion Error) Hintergrundlärm oder vom Benutzer erzeugte Geräusche können von der Spracherkennung als Befehle aufgefasst werden. Dieser Gefahr kann mit den oben beschriebenen Verfahren begegnet werden. Weiterhin kann ein "Stummschalten"-Knopf eingeführt werden, der die Spracherkennung abschaltet.

3.4 Schlüsse

Der SpeechActs-Prototyp hat ein grundlegendes Prinzip für das Design von Sprach-Benutzerschnittstellen etabliert: Es ist immens wichtig, die Kommunikation zwischen Mensch und Computer an die Grundlagen der zwischenmenschlichen Konversation anzugleichen. Wichtig ist auch, dass Benutzerschnittstellen für Sprache nicht einfach aus grafischen Benutzerschnittstellen abgeleitet werden können, sondern fast gänzlich eigenen Regeln unterliegen.

Bei der Spracheingabe tritt als limitierendes Element die unausgereifte Spracherkennung auf. Bei einer Erkennungsrate von 90% wird immer noch jeder zehnte Befehl falsch erkannt. Insofern sind Rückmeldungen und Verifizierung von Eingaben sehr wichtig. Um den Benutzer nicht zusehr zu belästigen, sollte die Verifizierung allerdings meistens implizit erfolgen ("Wie ist das Wetter in Athen?" - "Das Wetter in Athen ist sonnig bei 24 Grad."). Nur bei kritischen Aktionen wie dem Löschen einer Nachricht sollte das System explizit nachfragen.

4 Nomadic Radio: Sprache als Benutzerschnittstelle für mobile Endgeräte

Nitin Sawhney und Chris Schmandt stellen mit dem Nomadic Radio ein mobiles Endgerät vor, das den Benutzer auf Schritt und Tritt begleitet und ihn sowohl interaktiv als auch selbständig mit gesprochenen Informationen versorgt. Diese umfassen E-Mails, Nachrichten auf dem Anrufbeantworter oder Radionachrichten. Das Ziel des Nomadic Radio ist es, durch intelligente Benachrichtigungsstrategien den Benutzer möglichst wenig bei seiner eigentlichen Tätigkeit zu stören. Dafür setzt es neben der Sprache weitere auditive Elemente ein.

4.1 Grundlagen und Probleme der mobilen Kommunikation

Grundlegendes Problem beim Design eines mobilen Endgerätes ist die notwendige Vielseitigkeit der Benutzerschnittstelle. Benutzer von mobilen Kommunikationslösungen wie dem Nomadic Radio wollen unterwegs sowohl schnelle, kurze als auch ausgedehnte Transaktionen durchführen.

Sawhney und Schmandt unterscheiden außerdem zwei primäre Grundfunktionalitäten eines mobilen Endgerätes, Navigation und Notifikation. Zum einen will der Benutzer aktiv mit dem Gerät interagieren, zum anderen soll das System auch selbständig Benachrichtigungen ausgeben können.

Eine Lösung für diese Anwendungsszenarien sind auditive Schnittstellen.

Vorteile von Audio in mobilen Endgeräten Im Vergleich zu grafischen Benutzerschnittstellen haben auditive Benutzerschnittstellen gewisse Vorteile in mobilen Endgeräten.

- Mobile Endgeräte werden zunehmend kleiner. Bei diesen Größen stoßen grafische Benutzerschnittstellen an die Grenzen der Bedienbarkeit. Bildschirm und Tastatur verlieren ihren Nutzen, da sie nicht beliebig verkleinert werden können.
- Sprachein- und -ausgabe ermöglichen eine unauffällige Bedienung des Systems. Hände und Augen sind frei für andere Dinge.
- Durch einfache, zuverlässige Sprachkommandos können Aktionen schneller ausgeführt werden, als mit langwierigem Tippen von Befehlen.
- Gesprochene Sprache ist ausdrucksstärker und effizienter als Schrift und beansprucht die Aufmerksamkeit des Benutzers weniger.
- Der Mensch kann gleichzeitig mehrere Audioquellen parallel und unterbewusst wahrnehmen und seine Wahrnehmung gezielt auf einzelne richten.

Problemstellungen

- Sprache kann ermüden und langsam sein. Deshalb sollte der Benutzer Sprachausgaben beschleunigen und überspringen können.
- Im Gegensatz zu grafischen Benutzerschnittstelle muss der Benutzer die möglichen Befehle vorher auswendig lernen.
- Ungünstige Umgebungseinflüsse können die Spracherkennung stören. Deshalb empfiehlt es sich, zusätzlich auch die Eingabe über Tasten zu ermöglichen.
- In unserer Gesellschaft gibt es soziale Konventionen. Es wirkt unnatürlich, wenn jemand scheinbar mit sich selbst spricht.
- Sprache ist öffentlich. Privatsphäre und Vertraulichkeit von Daten sind schwer zu wahren.

4.2 Das Nomadic Radio

Das Nomadic Radio ist ein Prototyp eines mobilen Kommunikations- und Informationssystems. Als Ein- und Ausgabe-Medium kommt ein VoiceBeam Neckset (Abbildung 4.2) zum Einsatz. Es ist bequem zu tragen und verfügt über ein Mikrofon und Stereo-Lautsprecher. Die Java-basierte mobile Anwendung läuft auf tragbaren Rechnern (wearable computers). In den Programmiersprachen C und Perl geschriebene Server zeichnen Nachrichten aus dem Radio auf und bereiten E-Mails und Sprachnachrichten auf. Server und Client kommunizieren über WLAN. Die Sprachanalyse und Sprachsynthese übernimmt die AT&T Watson Speech API. Die Erkennungsrate beträgt über 90%, sinkt aber in lauten Umgebungen auf 50%. Der Benutzer kann während er einen Push-To-Talk(PTT)-Knopf drückt dem Nomadic Radio Befehle geben. Alternativ kann das System auch kontinuierlich lauschen. Eine Befehlssitzung wird dann mit "Nomadic Wake Up" eingeleitet. Wenn der Benutzer seine Sitzung beenden will, versetzt er das System mit "Nomadic Sleep" wieder in den Lauschzustand.



Abbildung 2. Voicebeam Neckset als Ein- und Ausgabe-Medium für das Nomadic Radio

4.3 Verwendete Techniken

Das Nomadic Radio implementiert eine Reihe raffinierter Techniken um die Aufmerksamkeit des Benutzers nicht zu sehr zu belasten.

Räumliches und paralleles Hören Die räumliche Anordnung von Geräuschquellen vermittelt dem Benutzer Informationen über die Art der Nachricht und erlaubt es ihm, selektiv einzelne Audiostreams herauszuhören. Das VoiceBeam Neckset hat Stereo-Lautsprecher, über die ein simulierter 3D-Surround-Sound ausgegeben wird. Empfangene Nachrichten werden nach Ankunftszeit um den Kopf herum angeordnet. Eine Mittags erhaltene Nachricht erklingt direkt von vorne, während eine Nachricht, die um 16 Uhr ankommt von rechts hinten zu

hören ist. Der Benutzer kann sich auf eine Quelle konzentrieren und die anderen Nachrichten unterbewusst im Hintergrund wahrnehmen. Ein weiteres Feature, das die Fähigkeit des Menschen zur räumlichen Audiowahrnehmung ausnutzt, ist die Nachrichtenvorschau (Message Scanning). Das System liest den Anfang einer neuen Nachricht vor, wartet kurz, ob der Benutzer auf diese zugreifen will und spielt ansonsten den Anfang der nächsten Nachricht ab. Die Audioquellen wandern dabei lauter werdend von rechts zur Mitte. Anschließend klingen sie nach links aus, während die nächste Nachricht schon rechts erklingt. Weil der Mensch sich auf eine räumlich angeordnete Quelle konzentrieren kann, kann der zeitliche Abstand zwischen zwei Nachrichten reduziert werden, was die Vorschau erheblich beschleunigt.

Skalierbare Audiodarstellung Heutige mobile Endgeräte wie Mobiltelefone oder PDAs benachrichtigen den Benutzer auch zu unpassenden Gelegenheiten über Anrufe und Termine. Dies kann während einer Besprechung passieren, aber auch während der Benutzer sich stark konzentrieren muss. In den meisten Fällen rechtfertigt es die Priorität einer Nachricht nicht, den Benutzer dabei zu stören. Unterbrechungen kosten Zeit, laut einer Studie von O’Conaill und Frohlich 1995 [7] bei Büroarbeitern zehn Minuten pro Stunde.

Probleme von heutigen Systemen

- Undifferenzierte Signaltöne vermitteln nur binäre Informationen (Anruf / kein Anruf)
- Mangels Kenntnis der Umgebung des Benutzers wird dieser oft zu unpassenden Zeiten unterbrochen.
- Die Geräte lernen nicht aus dem Verhalten des Benutzers und belästigen ihn immer wieder mit den gleichen Problemen.
- Das System hat keinen Überblick über die empfangenen Nachrichten. Dies führt zu unkoordinierten Benachrichtigungen.

Ein mobiles System sollte eine Benachrichtigungsstrategie verfolgen, die den Kontext des Benutzers miteinbezieht und zum Beispiel unwichtige Nachrichten erst später abspielen, wenn der Benutzer gerade Zeit hat. Deshalb muss es ein Bild der Umgebung haben (zum Beispiel durch passives Mithören der Umgebungsgeräusche) und auch die Wichtigkeit einer Nachricht und die Aktivität des Benutzers miteinbeziehen.

Das Nomadic Radio setzt die Forderung nach einer kontextabhängigen Benachrichtigung mit einer Skalierbaren Audiodarstellung (Scaleable Auditory Presentation) um.

Je nach Situation und Priorität der Nachricht stellt das System diese durch unterschiedliche Signale dar. Ziel ist, die Informationen dem Benutzer möglichst unobtrusiv darzustellen.

Stille. Wenn der Benutzer absolut nicht gestört werden darf, bleibt das System stumm. Nur Nachrichten mit höchster Priorität werden ausgegeben.

Hintergrundgeräusche (Ambient Audio) wie der Klang von fließendem Wasser dienen als unaufdringliche Signalisierung von Netzaktivität. Wenn Daten

übertragen werden fließt das Wasser schneller, ankommende Nachrichten werden als leise Platscher wiedergegeben.

Benachrichtigungs-Klänge teilen die Dringlichkeit einer Nachricht mit. Der Benutzer kann bestimmten Gruppen eigene Klänge zuordnen.

VoiceCues sind 1-2 Sekunden kurze charakteristische Stimmproben. Wenn eine neue E-Mail eintrifft, ertönt die Stimme des Absenders. Dies hat sich als besonders effektive Informationsquelle erwiesen, da die menschliche Stimme viel Aufschluss über die Dringlichkeit einer Nachricht geben kann, leicht einem Absender zuzuordnen ist und .

4.4 Evaluation und Schlüsse

Zwei Testpersonen verwendeten ein Nomadic Radio einige Tage für ihre üblichen Messaging-Aktivitäten. Sie hatten Probleme, das System zu benutzen, wenn Sie mit anderen Menschen zusammen waren, und bevorzugten auch die "Hintergrundgeräusche" gegenüber gesprochenen Benachrichtigungen. Des weiteren wünschten sich beide eine Zweitastenbedienung (OK, Abbrechen) zusätzlich zur Sprach-Benutzerschnittstelle, um reine Bestätigungen schneller ausführen zu können.

Auf mobilen Endgeräte ermöglichen auditive Benutzerschnittstellen eine kompakte und benutzerfreundliche Kommunikationsmethode. Wichtig ist allerdings, den Benutzer nicht mit Informationen zu überfluten. Unterschwellige Benachrichtigungen durch Hintergrundgeräusche bieten sich deshalb an und werden auch gerne angenommen.

5 Grundregeln für Sprach-Benutzerschnittstellen

Auch für das Design von Sprach-Benutzerschnittstellen gibt es allgemeine Empfehlungen. Steven Ross, Elizabeth Brownholtz und Robert Armes fassen ihre Erfahrungen im Design von Benutzerschnittstellen in *Voice User Interface Principles for a Conversational Agent* zusammen.

5.1 Anpassung an menschliche Sprachgewohnheiten

- Das System muss natürliche Sprache verstehen und bei Erkennungsproblemen nachfragen
- Das System muss Höflichkeitsformen wie "Guten Morgen" oder "Danke" verstehen.
- Das System darf Benutzer nur bei wichtigen Benachrichtigungen unterbrechen. Der Benutzer darf das System immer unterbrechen.
- Das System soll nachfragen, ob es sprechen darf.
- Der Benutzer darf nicht in einem Modus feststecken und soll keine gestellten Fragen beantworten müssen, um einen anderen Befehl geben zu können.

5.2 Verlässlichkeit und vertrauensbildende Maßnahmen

- Bei Antworten soll das System die ursprüngliche Frage mit einbauen ("Das Wetter in Athen ist sonnig bei 24 Grad Celsius" statt "Sonnig bei 24 Grad Celsius")
- Wenn der Erfolg einer Aktion nicht implizit wahrnehmbar ist, muss das System eine Bestätigung ausgeben.
- Destruktive oder nicht reversible Aktionen muss der Benutzer explizit bestätigen
- Bei länger andauernden Operationen muss das System deutlich machen, ob und was es macht.
- Der Benutzer sollte Operationen unterbrechen können.
- Wenn der Benutzer keine Antwort auf eine Frage gibt, sollte das System die Frage wiederholen.

5.3 Konsistenz und Transparenz des mentalen Modells

- Das System sollte nach Möglichkeit die Sprache verwenden, die es vom Benutzer erwartet. Es sollte keine Worte benutzen, die es nicht selbst versteht.
- Das System sollte keine Vermutungen über die Absichten des Benutzers anstellen, sondern nur das tun, was ihm explizit befohlen wird.
- Die Sprachbenutzerschnittstelle sollte konsistent sein, gleiche Aktionen sollten auf die gleiche Weise ausgeführt werden. Ähnliche Formulierungen sollten zusätzlich zur "konsistenten" Form erlaubt sein.
- Das System sollte den Benutzer nicht glauben machen, es sei intelligenter als es ist, da sonst der Benutzer Befehle gibt, die das System überfordern.

6 Sprache als Geräusch: Nonverbale Eingabemöglichkeiten

Neben Sprache kann man auch andere Fähigkeiten der menschlichen Stimme nutzen, um mit dem Computer zu kommunizieren. Takeo Igarashi und John F. Hughes von der Brown University haben drei Interaktionstechniken untersucht, die sie als "Control by Continuous Voice", "Rate-based Parameter Control by Pitch" und "Discrete Control by Tonguing" bezeichnen.

"*Control by Continuous Voice*" ermöglicht es, Parameter durch die Länge eines anhaltenden Tons zu beeinflussen. Der Benutzer sagt dann zum Beispiel seinem Fernseher "Lauter, aaaaaaaaaaaaaah". Solange der Benutzer einen bestimmten Ton von sich gibt, wird die Lautstärke kontinuierlich erhöht. Ein ähnliches System existiert mit SUITEkeys [8]. Dieses erfordert aber keinen kontinuierlichen Ton sondern einen Stopp-Befehl ("Maus nach unten stopp").

"*Rate-based Parameter Control by Pitch*" erweitert das oben beschriebene Verfahren um die Möglichkeit, einen zweiten Parameter durch die Tonhöhe zu steuern. Im obigen Szenario könnte der Benutzer die Tonhöhe variieren, um die Geschwindigkeit der Parameteränderung zu beeinflussen. Je höher der Ton, desto schneller wird die Lautstärke erhöht.

"*Discrete Control by Tonguing*" ermöglicht die Eingabe von diskreten Werten. Dazu macht der Benutzer für jeden Schritt ein Geräusch: Mit "Kanal hoch, ta, ta, ta, ta" kann er den Fernsehsender wechseln.

Diese Verfahren bieten eine sprachunabhängige, einfache und sehr direkte Steuerung von einfachen Parametern. Im Unterschied zu Sprachbefehlen ermöglichen Sie nuancierte und gleichzeitig kurze Befehle. Für den menschlichen Stimmapparat ist es aber relativ anstrengend, längere Zeit solche Geräusche hervorzubringen. Dies schränkt die Anwendung auf kurze, nicht zu häufige Befehlssequenzen ein.

7 Diskussion

Obwohl alle vier Arbeiten unterschiedliche Schwerpunkte haben, kristallisieren sich folgende drei grundlegenden Eigenschaften einer guten Sprach-Benutzerschnittstelle heraus:

Stringenz Sprechen ist naturgemäß langsamer und ermüdender als das Tippen auf einer Tastatur. Aufgrund der temporalen Natur von Sprache muss der Benutzer Sprachausgaben des Systems in voller Länge abhören und kann sie nicht wie bei grafischen Benutzerschnittstellen einfach ignorieren oder überfliegen. Deshalb ist es wichtig, den Benutzer zum einen von wiederkehrenden Sprachbefehlen zu entlasten und diese durch Befehltasten zu ersetzen. Außerdem sollte ein Sprachdialogsystem die Möglichkeit bieten, die Audioausgabe zu beschleunigen, zu verlangsamen und zu überspringen.

Konsistenz Im Gegensatz zur Benutzung einer grafischen Benutzerschnittstelle weiß der Anwender bei einer Sprach-Benutzerschnittstelle nicht jederzeit, in welchem Modus er sich befindet und welche Befehle ihm zur Verfügung stehen. In den meisten Anwendungsfällen hat der Benutzer nicht die Möglichkeit, ein Handbuch zu konsultieren. Aus diesem Grund sollte das Vokabular der Applikation einfach sein und konsistent bleiben. Das System muss dem Benutzer bei Bedarf Hilfestellung bieten. Es kann sinnvoll sein, je nach Aktion oder Modus wechselnde Hintergrundgeräusche zu verwenden, die dem Benutzer eine Orientierungshilfe geben.

Flexibilität Benutzer verwenden Synonyme für Befehle oder versuchen neue Befehle. Dies sollte das System berücksichtigen und Hilfestellung anbieten, wenn es eine Eingabe nicht verstehen kann. Auch Höflichkeitsformen, Umschreibungen und Äußerungen wie "Äh" und "Moment..." dürfen das System nicht durcheinanderbringen.

Besonders bei der Verwendung in in mobilen Geräten ist eine kontextsensitive Benachrichtigungsstrategie notwendig, die Störungen minimiert. Räumliche Audiodarstellung bietet die Möglichkeit, neben der eigentlichen Nachricht auch noch Metadaten übertragen zu können. Das Nomadic Radio und die Laut-Benutzerschnittstelle von Takeo Igarashi und John Hughes zeigen, dass non-verbale Ausdrucksmöglichkeiten wie unterschwellige Hintergrundgeräusche oder Stimmlaute Sprach-Benutzerschnittstellen deutlich bereichern können.

8 Stand der Technik: Praktische Anwendungen von Sprach-Benutzerschnittstellen

Sprachdialogsysteme sind heute nicht mehr wegzudenken. Vor allem im für telefonische Auskunftsdienste und Kundenbetreuung verwenden viele Unternehmen die künstlichen Kommunikationspartner. Da die Entwicklung von Software für Spracherkennung, -analyse, -generierung und -ausgabe aber hohen Forschungs- und Programmieraufwand erfordert, ist der Markt für Sprachtechnologie überschaubar. Im Folgenden werden einige kommerzielle und Open-Source-Applikationen für Sprachanwendungen vorgestellt. Ein weiterer Abschnitt beschäftigt sich mit VoiceXML, einer standardisierten Beschreibungssprache für Sprach-Benutzerschnittstellen.

8.1 Enterprise-Software

Viele Anbieter für Sprachapplikationen im Unternehmensbereich bieten ein umfassendes Portfolio aus Software für Spracherkennung und Sprachausgabe sowie kompletten Sprachdialogsystemen. Marktführer sind die beiden US-amerikanischen Unternehmen Scansoft und Nuance. In Deutschland existieren mit Crealog, Sympalog und Clarity ebenfalls einige Firmen mit eigenentwickelten Sprach-Portalen. Das US-Unternehmen Wizzard vertreibt die Text-To-Speech-Engine *AT&T Natural Voices*, die hochwertige Stimmen in vielen verschiedenen Sprachen bietet.

8.2 Anwender-Software

Für Büroanwender und Privatpersonen gibt es seit einiger Zeit schon Software zur Spracherkennung. Am weitesten verbreitet sind *Dragon Naturally Speaking* und IBM *ViaVoice*, dessen Engine auch in Linguatecs *Voice Pro* integriert ist. Beide Programme bieten spezielle Versionen für Mediziner und Juristen, die sich durch Wörterbücher mit Fachbegriffen auszeichnen. Der Webbrowser Opera verwendet IBM-Technologie um Sprachkommandos zu empfangen und Webseiten vorzulesen.

Schon seit längerem existieren rudimentäre Spracherkennung und Sprachausgabe auch in den Betriebssystemen Windows und MacOS. Besonders die Windows-Engines, die optional mit dem Microsoft Agent installiert werden können, sind aber nur eingeschränkt verwendbar. Sowohl Microsoft als auch Apple fügen ihren nächsten Betriebssystemen aber neue Sprach-Funktionalität hinzu. Apple hat mit *VoiceOver* eine Text-to-Speech-Engine und eine Spracherkennung in Mac OS 10.4 (Tiger) integriert. Microsoft plant, im Windows-XP-Nachfolger Longhorn eine Sprach-API zu integrieren.

8.3 Open-Source-Software

Es existieren einige Open-Source-Anwendungen für Sprach-Benutzerschnittstellen. Diese stammen vorwiegend aus dem akademischen Bereich und laufen meistens nur unter Linux. Einige werden schon längere Zeit

nicht mehr weiterentwickelt. CMS Sphinx [12] ist eine Software zur Spracherkennung, die in verschiedenen Versionen als C- und Java-Implementationen frei zur Verfügung steht. Eine weitere freie Spracherkennungssoftware ist ASR [17]. Zur Analyse der erkannten Sprache dient der Text-Parser Phoenix [11]. Zur Sprachausgabe verwenden viele Open-Source-Programme Festival [13]. Die TTS-Software existiert auch in einer abgespeckten Version für Embedded-Computer. Eine Alternative zu Festival stellt das tschechische Epos [15] dar. Als Plattform für Sprachdialogsysteme hat sich der Galaxy Communicator [14] der MIT Spoken Language Group etabliert. Auch mit dem VoiceXML-Browser publicVoiceXML [16] kann man einfache Dialogsysteme entwerfen.

8.4 VoiceXML

Was ist VoiceXML? VoiceXML ist eine Beschreibungssprache für Sprach-Benutzerschnittstellen. Der offene, auf XML basierende Standard wird von vielen Systemen unterstützt und findet seinen Weg auch in Anwenderprogramme wie den Webbrowser Opera. VoiceXML ist nicht auf freie Konversation mit dem Computer ausgelegt, sondern modelliert hauptsächlich Formulare, die durch Sprachbefehle ausgefüllt werden können.

Geschichte Der Ursprung von VoiceXML liegt in der Phone Markup Language (PML), die 1995 im Rahmen eines AT&T-Projekts entstand, das den Designprozess von Spracherkennungssoftware vereinfachen sollte. Auch Motorola, Lucent und IBM arbeiteten an eigenen PML-Versionen. 1998 gründeten AT&T, IBM, Motorola und Lucent das VoiceXML Forum, um an der Standardisierung der Sprache zu arbeiten. 2000 veröffentlichten sie VoiceXML 1.0, das sie kurz darauf beim W3C als Basis für einen internationalen Standard vorschlugen. 2004 wurde VoiceXML 2.0 als W3C Recommendation veröffentlicht.

Syntax VoiceXML verwendet zwei Arten von Datendateien. Die eigentlichen VoiceXML-Dateien modellieren die Benutzerschnittstelle, während die grXML-Dateien die Grammatiken für Elemente festlegen. Eine Grammatik legt dabei fest, welche Eingaben in ein bestimmtes Feld zulässig sind.

```
<vxml version="2.0">
<form>
<field name="drink">
<prompt>Would you like coffee, tea, milk, or nothing?</prompt>
<grammar src="drink.grxml" type="application/srgs+xml"/>
</field>
<block>
<submit next="http://www.drink.example.com/drink2.asp"/>
</block>
</form>
</vxml>
```

VoiceXML-Datei, die nach dem gewünschten Getränk fragt und das Ergebnis an ein ASP-Skript schickt.

```

<grammar mode="voice" xml:lang="en-US" version="1.0"
  root="command">
  <rule id="command" scope="public">
  <one-of>
  <item>coffee</item>
  <item>tea</item>
  <item>milk</item>
  <item>nothing</item>
  </one-of>
  </rule>
</grammar>

```

grXML-Datei, die festlegt, welche Antworten auf eine Frage gegeben werden können.

Wird keine zulässige Eingabe getätigt, wiederholt die Applikation die Frage. Neben Grammatik-Dateien für Spracheingabe gibt es auch eine Version für DTMF-Tastentöne.

Elemente von VoiceXML VoiceXML-Applikationen können zwei Basiselemente verwenden: *Formulare*, um Daten einzugeben und *Menüs*, um Aktionen auszuführen. Der Entwickler kann in VoiceXML auch *Sessions* und *Subdialoge* implementieren. Wenn Benutzereingaben nicht erkannt werden können und auch in anderen Sonderfällen werden *Events* ausgelöst, die von speziellen Methoden behandelt werden können.

Verbreitung von VoiceXML VoiceXML hat sich klar als Industriestandard durchgesetzt. Vor allem die großen Anbieter von Sprachdialogsystemen setzen auf VoiceXML zur Modellierung von Dialogen. Aber auch Anwendersoftware wie der Opera Browser und einige Open-Source-Applikationen [16] unterstützen VoiceXML.

9 Ausblick

Sprach-Benutzerschnittstellen bieten viele neue Möglichkeiten der Mensch-Maschine-Interaktion. Sie sind relativ billig in bestehende mobile Endgeräte zu integrieren und nutzen eine seit Jahrtausenden erprobte Interaktionsform. In unserer mobilen Multitaskinggesellschaft erlauben Sprachtechnologien die ständige Verfügbarkeit von Informationen. Gleichzeitig können intelligente Benachrichtigungsstrategien uns vor dem Informationsinfarkt retten. Ob es aber Sinn des Fortschritts ist, dass wir immer mehr Sachen gleichzeitig tun können, sei dahingestellt. Humanistischer ist da schon die Vorstellung, dass die Sprache den Computer menschlicher und persönlicher macht. Statt, dass der Mensch sich an den Computer anpasst passt sich nun der Computer an den Menschen an. Aber auch körperlich behinderten Menschen und Kindern bietet die Sprachein- und -ausgabe eine hervorragende Alternative zu Tastatur und Maus.

Sprach-Benutzerschnittstellen werden weder grafische Benutzerschnittstellen noch Kommandozeilen ersetzen. Erstere haben ihnen gegenüber den Vorteil der

besseren Übersicht und der steileren Lernkurve. Letztere sind der Spracheingabe in der Geschwindigkeit überlegen und bieten eine überragende Funktionalität. Statt vorhandene Benutzerschnittstellen zu ersetzen, erweitern Sprach-Benutzerschnittstellen diese vielmehr um einen weiteren Kanal, der besonders für kurze und räumlich variable Sitzungen geeignet ist. Spracherkennung erlaubt dann zum Beispiel in der Heimautomation, das Licht über Sprachkommandos zu steuern. Daneben werden vor allem mobile Endgeräte wie Mobiltelefone, PDAs und Spielkonsolen in Zukunft vermehrt auf Sprach-Benutzerschnittstellen setzen. Mit VoiceXML liegt auch ein anerkannter Standard vor, der die Entwicklung vorantreiben dürfte.

10 Literatur

1. Ross, S., Brownholtz, E., Armes, R.: Voice User Interface Principles for a Conversational Agent. Technical report, IBM Research (2004)
2. Igarashi, T., Hughes, J.F.: Voice as Sound: Using Non-verbal Voice Input for Interactive Control. In: UIST. (2001) 155–156
3. Yankelovich, N., Levow, G.A., Marx, M.: Designing SpeechActs: Issues in Speech User Interfaces. In: CHI. (1995) 369–376
4. Sawhney, N., Schmandt, C.: Nomadic radio: speech and audio interaction for contextual messaging in nomadic environments. *ACM Transactions on Computer-Human Interaction* **7** (2000) 353–383
5. Kitai, M., Imamura, A., Suzuki, Y.: Voice Activated Interaction System Based on HMM-based Speaker-Independent Word Spotting. In: Proceedings of the Voice I/O Systems Applications Conference. (1991)
6. McTear, M.F.: Spoken Dialogue Technology: Enabling the Conversational User Interface. *ACM Computing Surveys* **34** (2002) 90–169
7. O’Conaill, B., Frohlich, D.: Timespace in the Workplace: Dealing with Interruptions. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI’95), New York, ACM (1995) 262–263

11 Links

8. SUITEKeys for Windows: (<http://www.suitekeys.org>)
9. Übersicht Dynamic Time Warping: (<http://www.dcs.shef.ac.uk/stu/com326/>)
10. Wikipedia: Hidden Markov Models: (http://en.wikipedia.org/wiki/Hidden_Markov_Models)
11. Phoenix - Semantic Parser: (<http://cslr.colorado.edu/~whw/>)
12. CMU Sphinx - Speech Recognition Engine: (<http://cmusphinx.sourceforge.net>)
13. Festival - Speech Synthesis System: (<http://www.cstr.ed.ac.uk/projects/festival/>)
14. Galaxy Communicator - Dialogue System Toolkit: (<http://communicator.sourceforge.net>)
15. Epos - Speech Synthesis System: (<http://epos.ure.cas.cz>)
16. Public VoiceXML - VoiceXML Browser: (<http://www.publicvoicexml.org>)
17. ISIP ASR - Speech Recognition Engine: (<http://www.isip.msstate.edu/projects/speech/>)