Einführung in Dialogsysteme

Christoph Draxler
draxler@phonetik.uni-muenchen.de

Tel. 2180 2807
Raum 220

29. Juni 2018

Seminar Dialogsysteme

Dialogsysteme sind Computerprogramme, die eine sprachgesteuerte Nutzung von Diensten ermöglichen. Zu diesen Diensten zählen Informationsdienste, aber auch Unterhaltungsangebote und der online-Verkauf.

Seminar Dialogsysteme

Dialogsysteme sind Computerprogramme, die eine sprachgesteuerte Nutzung von Diensten ermöglichen. Zu diesen Diensten zählen Informationsdienste, aber auch Unterhaltungsangebote und der online-Verkauf.

Das Seminar "Dialogsysteme"behandelt die technologischen Grundlagen von Dialogsystemen, d.h. Spracherkennung und -synthese, Techniken der Dialogführung sowie den Zugriff auf den gewünschten Dienst. Darüberhinaus führt es ein in die Gestaltung von sprachgesteuerten Benutzerschnittstellen und in die Evaluation von Dialogsystemen, bei der weniger die Technologie als die Nützlichkeit und Benutzerfreundlichkeit untersucht wird.

Seminar Dialogsysteme

Dialogsysteme sind Computerprogramme, die eine sprachgesteuerte Nutzung von Diensten ermöglichen. Zu diesen Diensten zählen Informationsdienste, aber auch Unterhaltungsangebote und der online-Verkauf.

Das Seminar "Dialogsysteme"behandelt die technologischen Grundlagen von Dialogsystemen, d.h. Spracherkennung und -synthese, Techniken der Dialogführung sowie den Zugriff auf den gewünschten Dienst. Darüberhinaus führt es ein in die Gestaltung von sprachgesteuerten Benutzerschnittstellen und in die Evaluation von Dialogsystemen, bei der weniger die Technologie als die Nützlichkeit und Benutzerfreundlichkeit untersucht wird.

Für das Seminar werden wir Dialoge aufnehmen, transkribieren und analysieren. Außerdem steht ein einfaches Dialogsystem auf der Basis von VoiceXML zur Verfügung. Wir werden den Aufbau dieses Systems kennenlernen und es in geeigneter Weise erweitern.

Dialogaufnahmen

Im Rahmen des Seminars führen wir im Tonstudio Dialogaufnahmen durch. Dabei werden Sie mit einem/r Dialogpartner/in ein vorgegebenes Alltagsthema besprechen – eine Terminabsprache treffen, eine Diskussion führen, eine Auskunft suchen.

- 30 Teilnehmer à 2 Personen pro Dialog $\rightarrow$ 15 Dialoge
- max. Dialogdauer ca. 5 Minuten $\rightarrow$ 75 Minuten
- Treffpunkt: Tonstudio im IPS
- bitte pünktlich sein!
- Wer möchte Aufnahmen durchführen?

Die Aufnahmen stelle ich dann für die weitere Verarbeitung (Chunking, Transkription, usw.) bereit.

Voraussetzungen

Für die erfolgreiche Teilnahme am Seminar sollten Sie folgende Voraussetzungen erfüllen

- Programmierkenntnisse (egal in welcher Sprache)
- elementares Verständnis von Audioverarbeitung am Computer
- orthographische Transkription eines vollständigen Dialogs
- Bereitschaft zur praktischen Programmierarbeit in VoiceXML und ggf. JavaScript

Das Seminar wird mit einer Klausur abgeschlossen. Diese findet in der Regel in der letzten Seminarstunde statt.

Literatur

Klabunde et al. (eds) Computerlinguistik und Sprachtechnologie, 2. Auflage, Spektrum Akademischer Verlag, Heidelberg 2004

Abbot, Kenneth Voice Enabling Web Applications, VoiceXML and Beyond, Apress, 2001

Berwein, Angelika Magisterarbeit 2011, Institut für Phonetik und Sprachverarbeitung

McTear, Michael Spoken Dialogue Technology. Toward the conversational user interface, Springer, New York, 2004

Dahm, Markus Grundlagen der Mensch-Computer-Interaktion, Pearson Studium, 2006

Nass, Clifford; Brave, Scott Wired for Speech - How Voice Activates and Advances the Human-Computer Relationship, MIT Press, Cambridge (MA), 2005

VoiceXML `www.w3.org/TR/voicexml21/` Spezifikation

weitere Literatur wird im Seminar bekanntgegeben.

Einführung in Dialoge und Dialogsysteme
- Dialog
- Dialog-Management

Spracherkennung und -synthese: Auffrischung
- Spracherkennung
- Sprachsynthese

Implementation von Dialogsystemen
- XML
- VoiceXML
- Beispielanwendung: New Personal Information Manager

Evaluation von Dialogsystemen
- Einführung Evaluation
- Frameworks

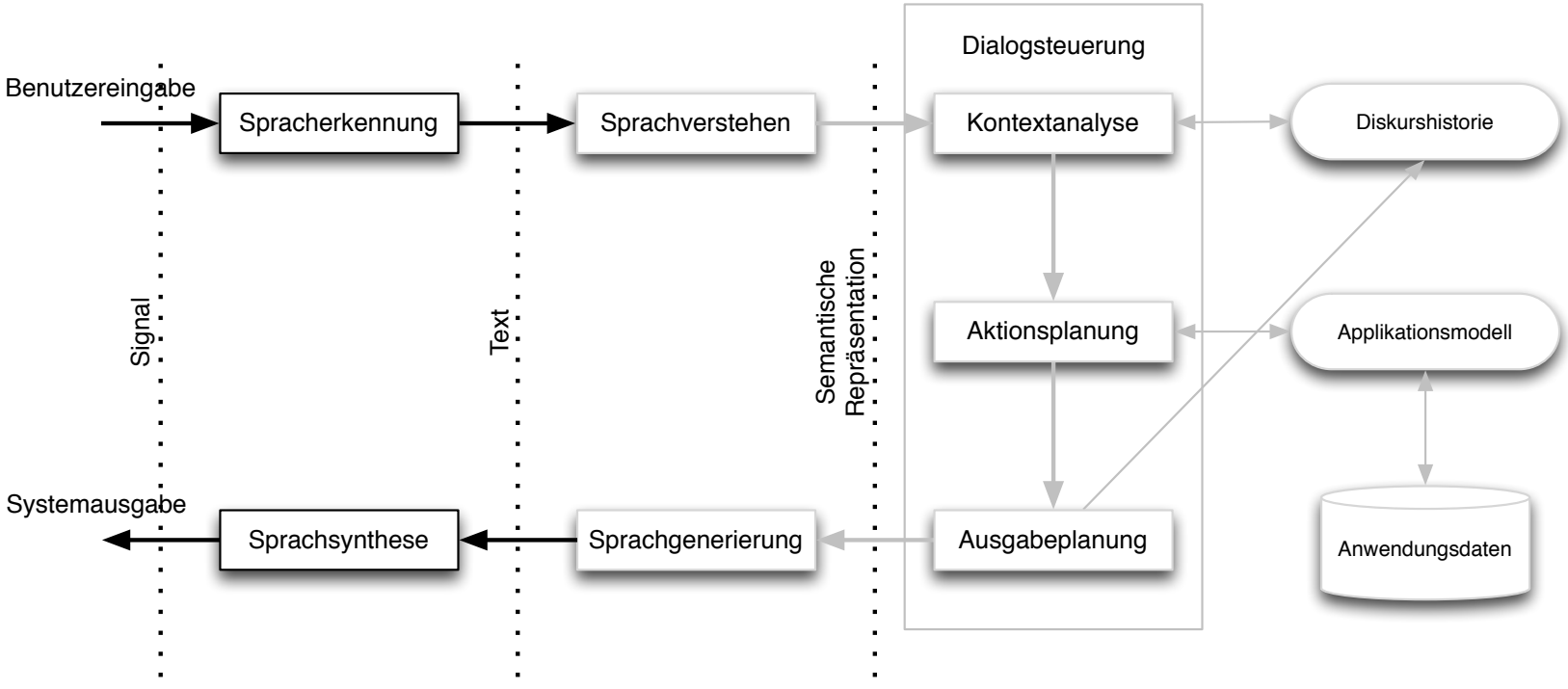
Anhang
- Voxeo-Server für eigene Projekte in VoiceXML
- Transkription mit Octra

Einführende Folien

siehe hierzu die Folien von Klaus Schulz

- Dialogbegriff: DialogseminarSitzung2.pdf
- Dialog-Management: DialogseminarSitzung3.pdf

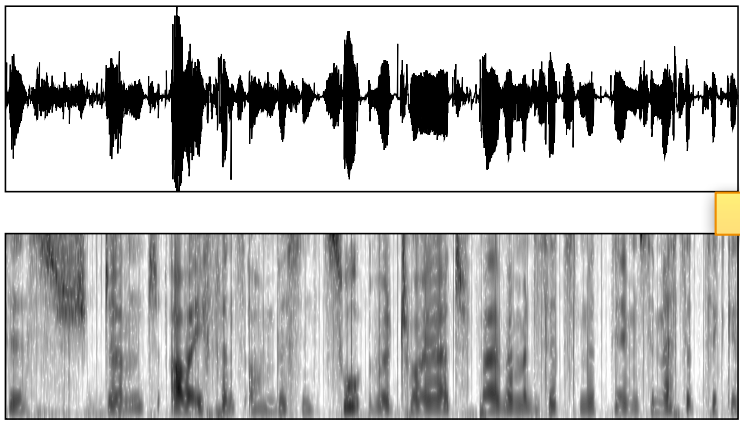
Kontext Dialogsysteme



aus: Klabunde et al. Computerlinguistik und Sprachtechnologie, 2. Auflage, 2004

Spracherkennung

Abbildung von Merkmalen akustischer Signale auf linguistische Einheiten (engl. *automatic speech recognition*, ASR).



Das stimmt. Aber ich finde es auch ein bisschen zu überteuert, weil wenn man das jetzt mit dem Semesterticket zum Beispiel von Regensburg vergleicht, oder Augsburg, da zahlst Du so ungefähr 100 € für das halbe Jahr, hast dabei aber auch die Gebühren für die Uni dabei **und** das Semesterticket. Also, es könnte noch billiger sein, finde ich. Und es ist ja schon wieder erhöht worden, also im Vergleich zum letzten Jahr. Da waren es 142 € und jetzt sind es ja ungefähr 4 € mehr. Ne, es ist schon höher geworden.

Die folgenden Folien basieren auf [?]

Spracherkennung: Mehrstufiger Prozess

- akustische Analyse des Sprachsignals und Extraktion von Merkmalen
- Erstellen von hypothetischen Sequenzen von Wörtern
- Reduzieren auf syntaktisch akzeptable Wortfolgen
- Semantische Analyse zur Erfassung der Bedeutung
- Einbeziehen des pragmatischen Kontexts

Akustische Analyse

Das akustische Sprachsignal enthält neben dem linguistischen Inhalt auch Informationen über den Sprecher, die Situation und die Umgebung.

- Geschlecht, Alter, regionale Herkunft, emotionaler Zustand, ...
- Dialog, Monolog, gelesene oder spontane Sprache, ...
- Studio, Büro, zuhause, unterwegs, ...

Was können Sie hier über Sprecher, Situation und Umgebung sagen? [Audiobeispiel]

Spracherkennung als Musterabgleich

Unbekanntes sprachliches Muster (= die zu erkennende Äußerung) wird mit Referenzmustern abgeglichen.

- Referenzmuster werden durch Training erzeugt
- Entscheidung, welches Muster passt, erfolgt mittels eines *Ähnlichkeitsmaßes*

Signalanalyse

Das akustische Sprachsignal wird digitalisiert. Die Signalanalyse generiert eine parametrische Repräsentation des Sprachsignals.

Vorverarbeitung Spektralanalyse berechnet Energie in einzelnen Frequenzbereichen

Merkmalsextraktion erkennt robuste Merkmale, reduziert die Datenmenge und ihre Dimensionalität erheblich

Beide Schritte werden ca. alle $10ms$ durchgeführt. Ergebnis ist eine Folge von *Merkmalsvektoren*.

Untereinheitenvergleich

Abbildung von Merkmalsvektoren auf linguistische Einheiten. Diese Einheiten können sein

- Phoneme
- Diphone
- Halbsilben
- Silben
- ganze Wörter

Der Untereinheitenvergleich liefert zu jedem Zeitpunkt eine Bewertung der in Frage kommenden Einheit.

Lexikalische Dekodierung

Die Folgen von Einheiten müssen sinnvoll zusammengefasst werden.

`/ S p R a: X ? E6 k E n U N /`

kann man auf mehrere Arten in kleinere Einheiten zerlegen:

1. sprach er kennung
2. sprach erkennung

Wenn einzelne Einheiten falsch erkannt werden wird die Zerlegung ebenfalls unzuverlässiger.

Lexikon

Ein *Lexikon* enthält die zulässigen Folgen von Einheiten und bildet sie auf höhere Einheiten ab.

Wort	Phoneme
Bonn	B O n
Dortmund	d O6 t . m U n t
fahren	f a: . R @ n
ich	I C
nach	n a x

Die erkannten Einheiten müssen im Lexikon gesucht und auf höhere Einheiten abgebildet werden, z.B. Wörter.

Syntaktische Einschränkung

Nicht alle Folgen von Wörtern sind sinnvoll.

```
public $ZEITRAUM =
   [für | von] ($WOCHENTAG [ab] [$ZEIT] bis $WOCHENTAG [$ZEIT]
   | $WOCHENTAG [ab] $ZEIT
   | $WOCHENTAG)
   | [von | ab] $ZEIT  [bis $ZEIT]
   | [am] $WOCHENTAG von $ZEIT bis $ZEIT;
```

Eine *Grammatik* beschreibt zulässige Folgen von höheren Einheiten.

Semantische und pragmatische Analyse

Prinzipiell lassen sich weitere semantische und pragmatische Einschränkungen der Suche nach Wortkandidaten formulieren. Dieser Schritt wird allerdings meist vom Dialogmanager ausgeführt und ist unabhängig vom Spracherkenner.

Evaluation von Spracherkennung

Ein *Fehlermaß* ist ein standardisiertes Messverfahren für Fehler

Wortfehlerrate (engl. *word error rate, WER)* $= (S + D + I)/N$ mit
$S =$ Anzahl Substitutionen, $D =$ Anzahl Löschungen,
$I =$ Anzahl Einfügungen, $N =$ Anzahl Wörter in der Referenztranskription

Wortgenauigkeit (engl. *word accuracy, WA)*
$= 1 - WER = (N - S - D - I)/N$

Weitere Maße sind *Sentence Error Rate*, *number of errors per sentence*, oder *word error per sentence*.

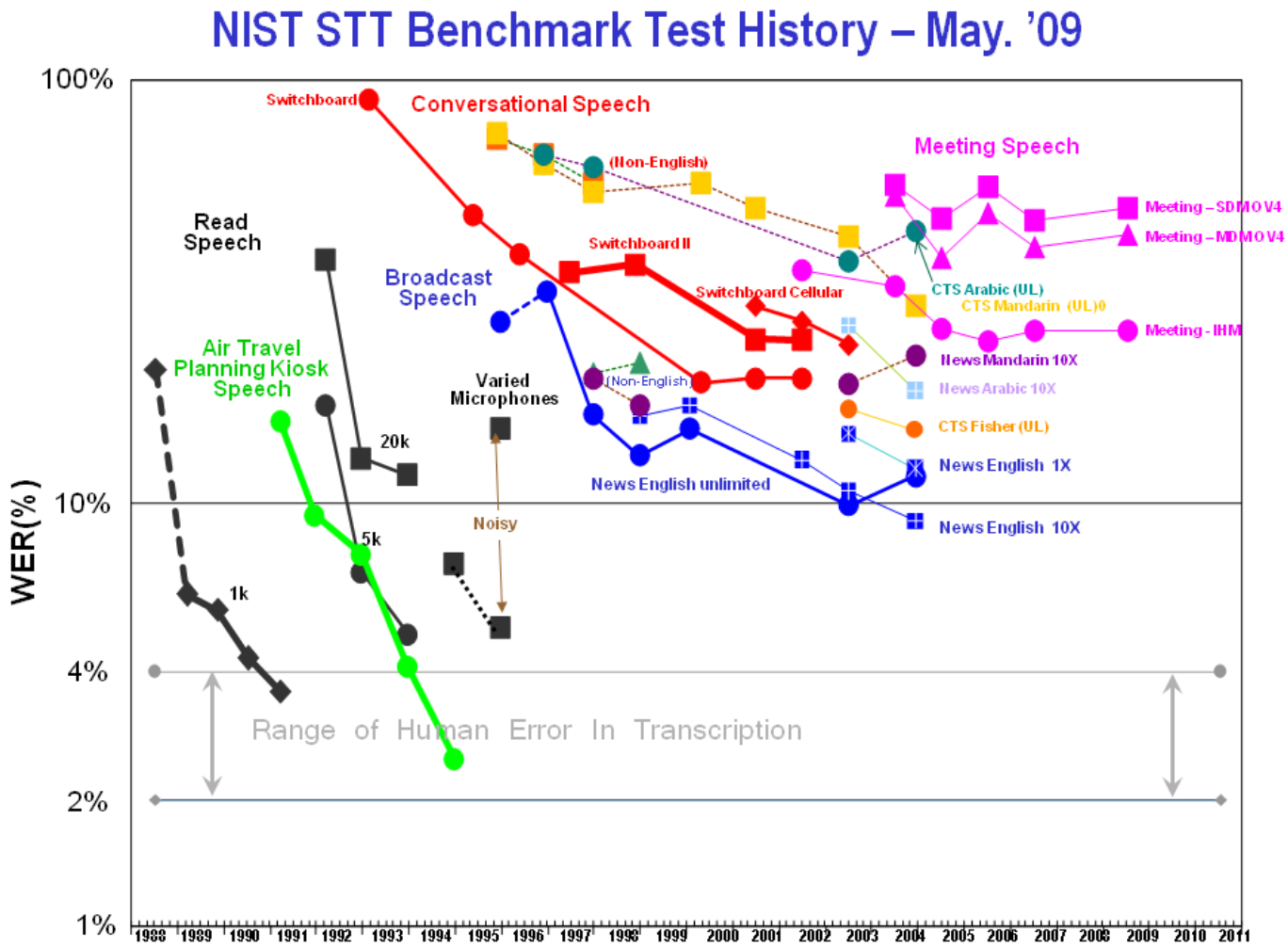
Beispiel WER

`REF: Heute ist schönes Frühlingswetter.`
`HYP: Heute keines Frühlingswetter.`

Ersetzung von '`keines`' zu '`schönes`', Löschung von '`ist`':
$(1 + 1 + 0)/4 = 0.5$

Diskussion

- kein universeller Spracherkenner verfügbar
- aber Systeme für spezielle Anwendungen
  - beschränktes Vokabular
  - eingeschränkter Nutzerkreis
  - festgelegte Signalqualität
- unterschiedliche Anforderungen
  - Diktiersysteme
  - Gerätebedienung
  - Teledienste
- im Gegensatz zum Menschen kaum flexible Anpassung



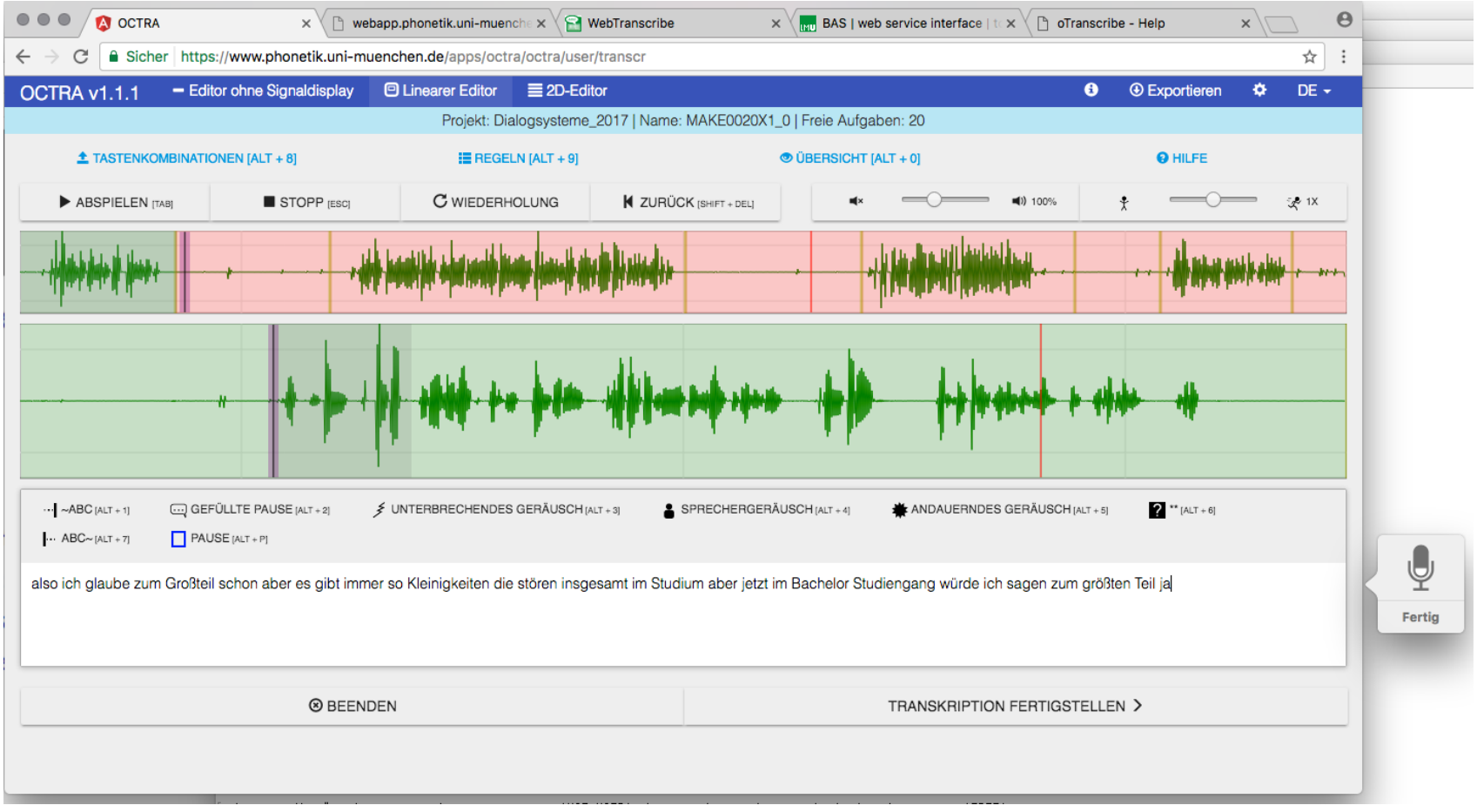
Die Leistung von Spracherkennung verbessert sich laufend, gleichzeitig werden die Aufgaben schwieriger.

`www.itl.nist.gov/iad/mig/publications/ASRhistory/`

Beispiele

Diktiersysteme	Nuance, Dragon Dictate
Suchsysteme	Siri, Alexa, Cortana, Android- und Google-Spracherkennung
Testsysteme	webASR in Sheffield: www.webasr.org
eingebaut	Mac OSX, Windows 10

Cool! Transkription durch Diktieren



Sprachsynthese

Generierung von Sprachsignalen aus linguistischen Einheiten

- Sprache als (Haupt-)Ausgabemodalität
- Auskunft, Feedback, Textwiedergabe

Häufig in Kombination mit anderen Modalitäten, z.B. Grafik.

Die gängige Abkürzung ist TTS *Text To Speech*.

Sprachsynthese als Prozess

Sprachsynthese als zweistufiger Prozess

- linguistische Analyse des Eingabetextes
- Umsetzung der linguistischen Repräsentation in ein Sprachsignal

Im Gegensatz zur Spracherkennung ist die Synthese ein weitgehend linearer Prozess ohne Rückkopplung. Fehler pflanzen sich daher bis ins Signal fort.

Textnormalisierung

Mit computerlinguistischen Verfahren müssen die Elemente eines Textes *normalisiert* und ggf. *expandiert* werden.

```
Bei der Wahl am 12.3.1988 gewann
Tony Blair ca. 52% der Wählerstimmen.
```

- *Wählerstimmen* zerlegen in *Wähler* und *Stimmen*,
- Erkennen von *12.3.1988* als Datum mit besonderer Sprechweise *neunzehnhundert*,
- *ca.* und *%* als 'circa' und 'Prozent' wiedergeben,
- Erkennen von *Tony Blair* als englischen Eigennamen...

Diese Normalisierung ist keineswegs trivial!

Disambiguierung

Ein *Text* kann mehrere Lesarten haben. Die Wortfolge *was willst du denn schon wieder* hat mindestens 4 Lesarten:

- Wiederholung einer Handlung
- Freiwilligkeit einer Handlung
- Ausführender einer Handlung
- Zeitliches Auftreten einer Handlung

Die passende Lesart muss durch *Kontextanalyse* ermittelt werden.

Disambiguierung in gesprochener Sprache

Anschließend muss die intendierte Lesart mit den Mitteln gesprochener Sprache, hier z.B. durch *Prosodie* ausgedrückt werden.

Wiederholung 'Was, willst Du schon *wieder*?'

Freiwilligkeit 'Was *willst* Du schon wieder?'

Ausführender 'Was willst *Du* schon wieder?'

Zeit 'Was willst Du *schon* wieder?

Um diese Lesarten mit Sprachsynthese auszudrücken muss die Wortfolge um explizite Anweisungen zur Prosodie ergänzt werden.

Notation für Sprachsynthese

Solche explizite Anweisungen sind mit Auszeichnungssprachen wie SSML (*speech synthesis markup language*) möglich. Hier wird eine junge männliche Stimme ausgewählt und in der Äußerung das Wort *one* betont.

```
<p>
  <s xml:lang="en-US">
    <voice name="David" gender="male" age="25">
      For English, press <emphasis>one</emphasis>.
    </voice>
  </s>
...
</p>
```

SSML ist ein vom W3C verabschiedeter Standard:
www.w3.org/TR/speech-synthesis/

Sprachausgabe

Es gibt im Wesentlichen drei Arten der Sprachausgabe technischer Systeme.

Pre-recorded Sämtliche Äußerungen eines Systems sind von einem (professionellen) Sprecher aufgenommen und werden abgespielt.

Formantsynthese parametrisches Quelle-Filter-Modell der Sprachproduktion implementiert, jede Äußerung kann in jeder Stimme generiert werden.

Unit-selection Äußerung wird aus Bausteinen von aufgenommener Sprache zusammengesetzt; je mehr und besser passende Bausteine, desto besser die Synthesequalität.

Die erste ist *keine* Synthese, die beiden anderen schon.

Frage

Was hat die folgende Zeitungsmeldung mit den Synthese-Architekturen zu tun? [Sprecher von The Simpsons hört auf]

Evaluation

Evaluation von Sprachsynthese basiert auf meist *subjektiven* Tests:

Verständlichkeit wie gut lassen sich Laut, Wörter, Äußerungen verstehen?

Natürlichkeit wie natürlich klingt eine synthetisierte Äußerung?

Angenehmheit wie angenehm ist die Stimme?

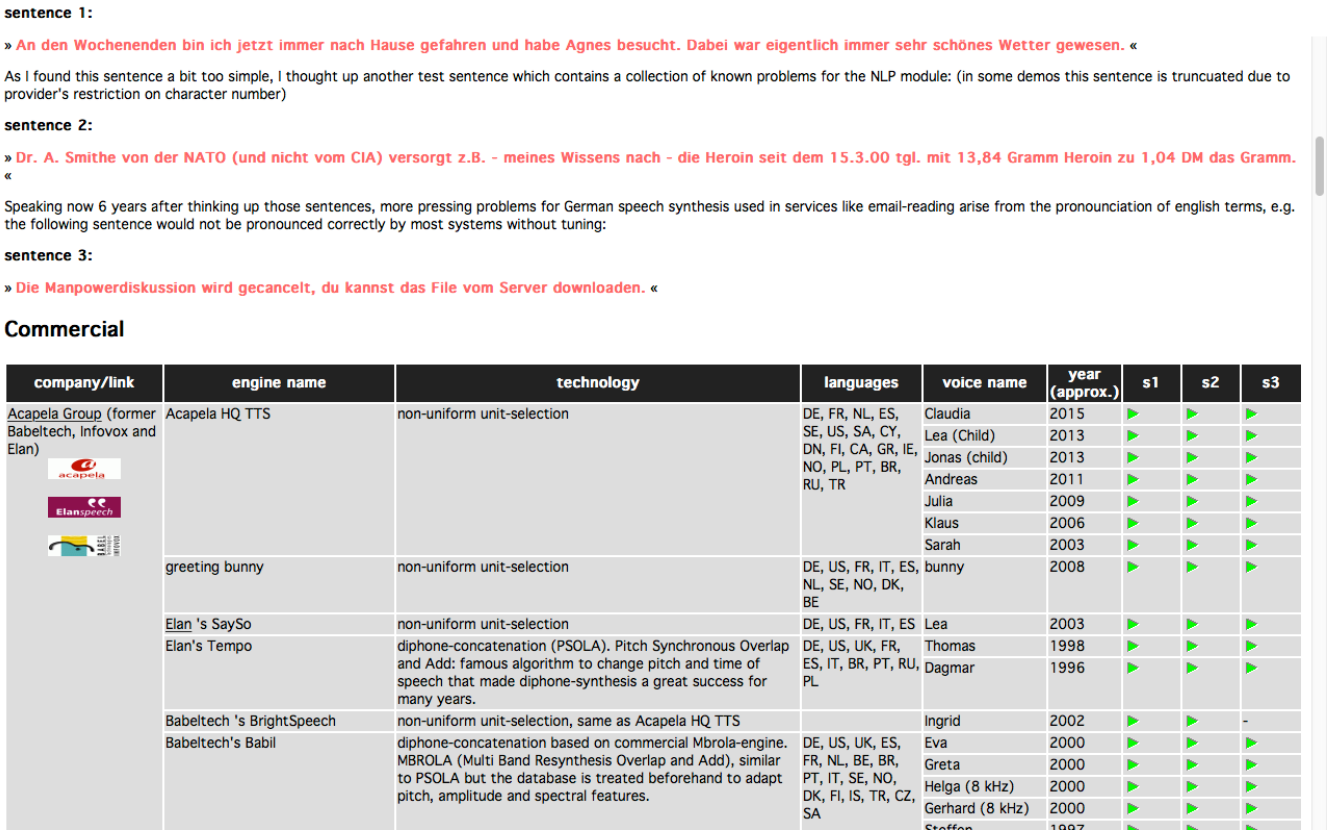
weitere Kriterien sind z.B. auch *Aufgaben-Verständlichkeit* und *globale Akzeptanz*.

Machen Sie den Test!

webapp.phonetik.uni-muenchen.de/WebExperiment/synthesetest.html

Synthesesysteme im Test

Felix Burkhardt hat eine Webseite mit vielen Beispielen zusammengestellt:



sentence 1:

» An den Wochenenden bin ich jetzt immer nach Hause gefahren und habe Agnes besucht. Dabei war eigentlich immer sehr schönes Wetter gewesen. «

As I found this sentence a bit too simple, I thought up another test sentence which contains a collection of known problems for the NLP module: (in some demos this sentence is truncuated due to provider's restriction on character number)

sentence 2:

» Dr. A. Smithe von der NATO (und nicht vom CIA) versorgt z.B. - meines Wissens nach - die Heroin seit dem 15.3.00 tgl. mit 13,84 Gramm Heroin zu 1,04 DM das Gramm. «

Speaking now 6 years after thinking up those sentences, more pressing problems for German speech synthesis used in services like email-reading arise from the pronounciation of english terms, e.g. the following sentence would not be pronounced correctly by most systems without tuning:

sentence 3:

» Die Manpowerdiskussion wird gecancelt, du kannst das File vom Server downloaden. «

Commercial

company/link	engine name	technology	languages	voice name	year (approx.)	s1	s2	s3
Acapela Group (former Babeltech, Infovox and Elan)	Acapela HQ TTS	non-uniform unit-selection	DE, FR, NL, ES, SE, US, SA, CY, DN, FI, CA, GR, IE, NO, PL, PT, BR, RU, TR	Claudia	2015	▶	▶	▶
				Lea (Child)	2013	▶	▶	▶
				Jonas (child)	2013	▶	▶	▶
				Andreas	2011	▶	▶	▶
				Julia	2009	▶	▶	▶
				Klaus	2006	▶	▶	▶
				Sarah	2003	▶	▶	▶
	greeting bunny	non-uniform unit-selection	DE, US, FR, IT, ES, NL, SE, NO, DK, BE	bunny	2008	▶	▶	▶
	Elan 's SaySo	non-uniform unit-selection	DE, US, FR, IT, ES	Lea	2003	▶	▶	▶
	Elan's Tempo	diphone-concatenation (PSOLA). Pitch Synchronous Overlap and Add: famous algorithm to change pitch and time of speech that made diphone-synthesis a great success for many years.	DE, US, UK, FR, ES, IT, BR, PT, RU, PL	Thomas	1998	▶	▶	▶
				Dagmar	1996	▶	▶	▶
	Babeltech 's BrightSpeech	non-uniform unit-selection, same as Acapela HQ TTS		Ingrid	2002	▶	▶	-
	Babeltech's Babil	diphone-concatenation based on commercial Mbrola-engine. MBROLA (Multi Band Resynthesis Overlap and Add), similar to PSOLA but the database is treated beforehand to adapt pitch, amplitude and spectral features.	DE, US, UK, ES, FR, NL, BE, BR, PT, IT, SE, NO, DK, FI, IS, TR, CZ, SA	Eva	2000	▶	▶	▶
				Greta	2000	▶	▶	▶
				Helga (8 kHz)	2000	▶	▶	▶
				Gerhard (8 kHz)	2000	▶	▶	▶
				Steffen	1997	▶	▶	▶

http://ttssamples.syntheticspeech.de

Gemeinsamkeiten und Unterschiede

Spracherkennung und -Synthese haben einige Gemeinsamkeiten

- Seltene Ereignisse problematisch
- Linguistische Textanalyse u.U. aufwändig bis unmöglich
- Lautdauermodellierung komplex (Laut, Nachbarn, Prosodie, usw.)
- Korpusdesign – maximal repräsentativ vs. minimal groß

und weitere...

Auszeichnungssprachen

Eine Auszeichnungssprache (engl. *Markup Language*) erlaubt das Setzen von Marken, sog. Auszeichnungen (engl. *Tags*), in einem Text. Beispiele sind

- LaTeX zum Setzen von Textdokumenten
- HTML (*Hypertext Markup Language*) für Webseiten
- SVG (*Scalable Vector Graphics*) für Vektorgrafiken
- ... und viele andere mehr

Diese Marken enthalten zusätzliche Informationen zum Text und sie können maschinell verarbeitet werden.

XML

XML (für *eXtensible Markup Language*) ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien (siehe Wikipedia-Artikel zu XML)

Tag entweder ein Paar `<tag>...</tag>` oder die Kurzform `<tag/>`

Element Auszeichnungselement mit Inhalt innerhalb von Tag-Paaren oder in einem Tag

Attribute benannte Werte innerhalb eines Tags, geschrieben `attribut="wert"`

Der Dokumentinhalt steht normalerweise innerhalb von Tag-Paaren.

XML Dokumenttyp

XML ist eine Metasprache. Dokumenttypdefinitionen (*DTD*) legen das Tag-Inventar für eine konkrete XML-Anwendung fest.

- ein XML-Dokument ist *wohlgeformt*, wenn es der XML-Syntax entspricht
- ein XML-Dokument ist *gültig*, wenn es wohlgeformt ist und einer Dokumenttypdefinition (DTD) entspricht

Dokumenttypdefinitionen liegen entweder im DTD-Format oder als XMLSchema vor.

Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">

  <var name="benutzername"/>

  <form id="hallo">
    <record name="benutzername" maxtime="10s" finalsilence="1000ms">
      <prompt>
        Hallo. Was ist dein Benutzername?
      </prompt>
    </record>
  </form>
</vxml>
```

<var> ist ein einfacher Tag, alle anderen Elemente bestehen aus Tag-Paaren. Die erste Zeile ist ein sog. Verarbeitungshinweis (engl. *processing instruction*) - er gibt z.B. an, um welche XML-Version es sich handelt.

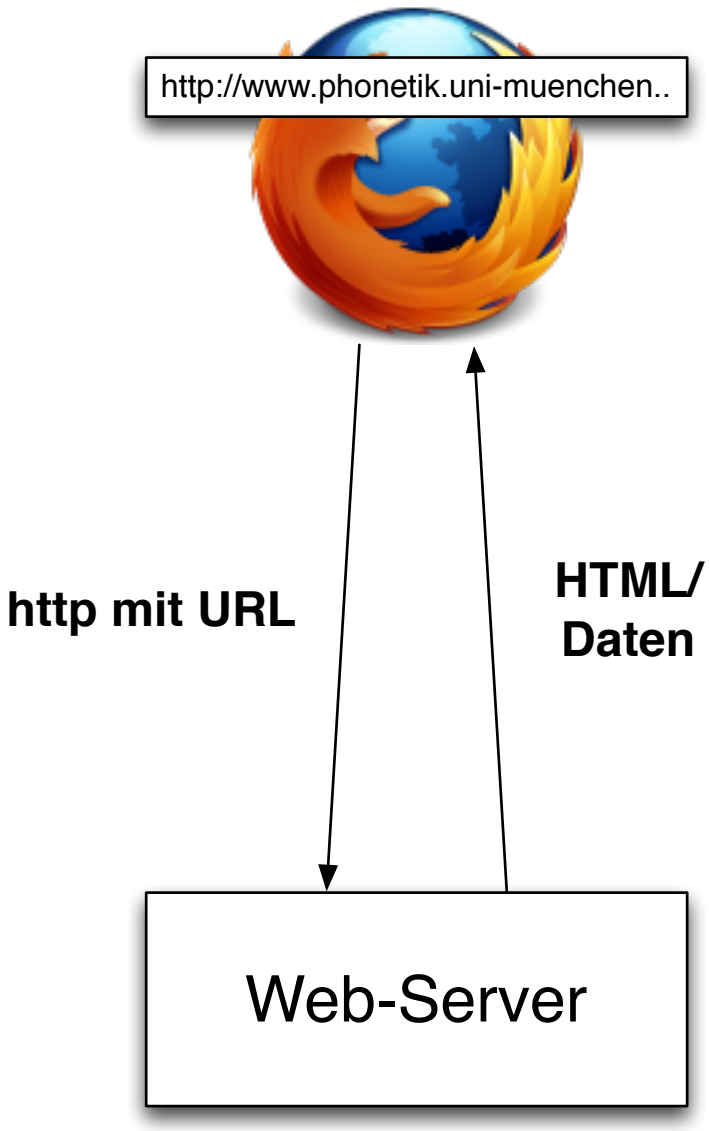
Einführung

Dialogsysteme

- erlauben die Kommunikation per Stimme und Telefon
- bieten Zugriff auf Informationen oder Dienste
  - Reise-, Wetter-, Nachrichtenauskunft
  - Bank-, Provider- und sonstige Transaktionen
  - Einkäufe
  - Unterhaltung
- greifen dazu auf Ressourcen im WWW zu

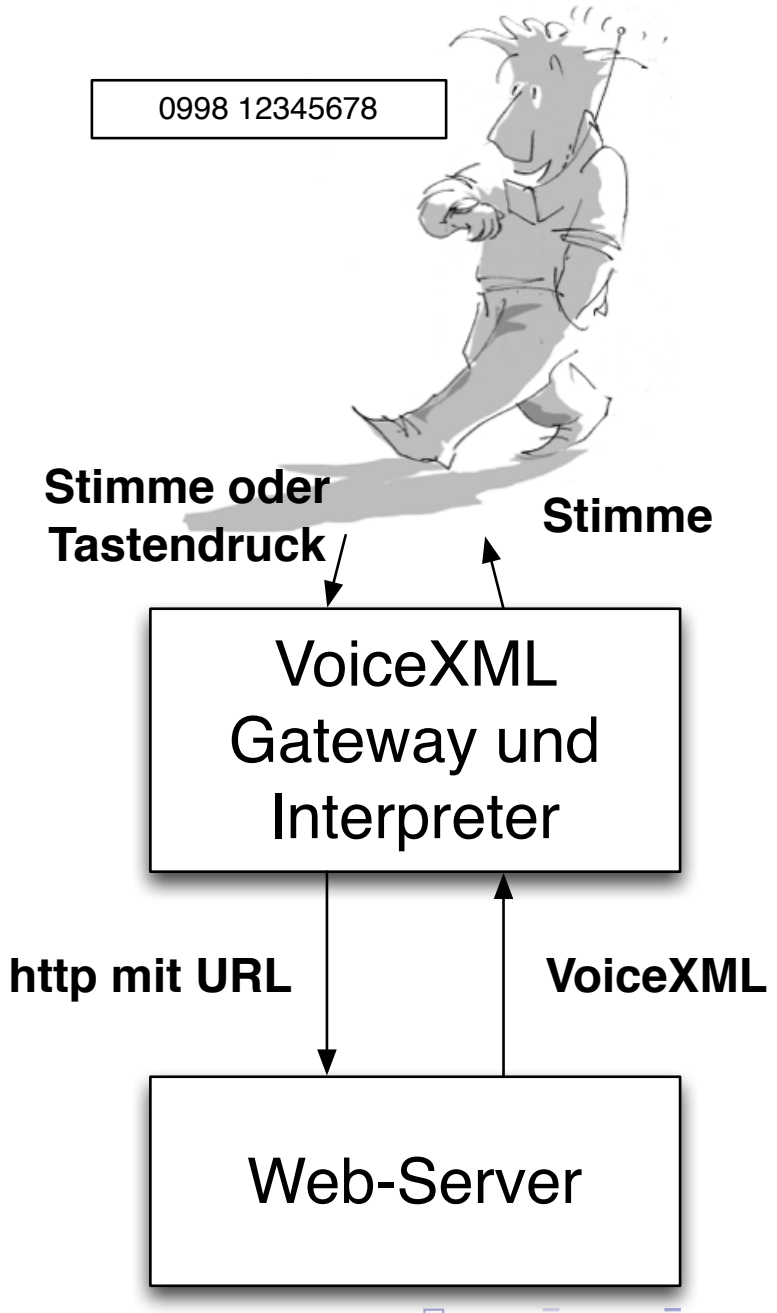
Browsen im Web

- im Browser URL öffnen
- Browser schickt `http`-Anfrage an Server
- Server schickt `HTML` oder Daten zurück an Browser
- Browser stellt `HTML` oder Daten dar
- weitere Navigation per Links

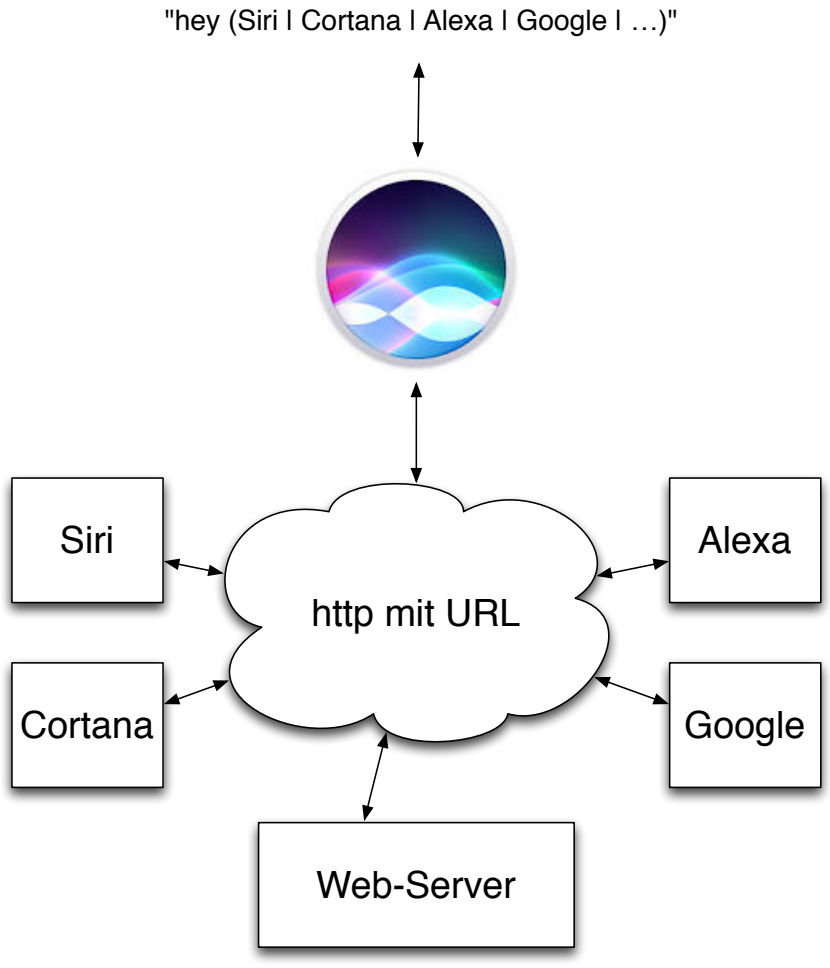


Browsen per Telefon

- Telefonnummer des Dienstes anrufen
- VoiceXML Gateway hebt ab und schickt `http`-Anfrage an Server
- Server schickt `VoiceXML` zurück an Interpreter
- Interpreter spricht Inhalt des `VoiceXML`-Dokuments
- weitere Navigation per Tastendruck oder Sprechen

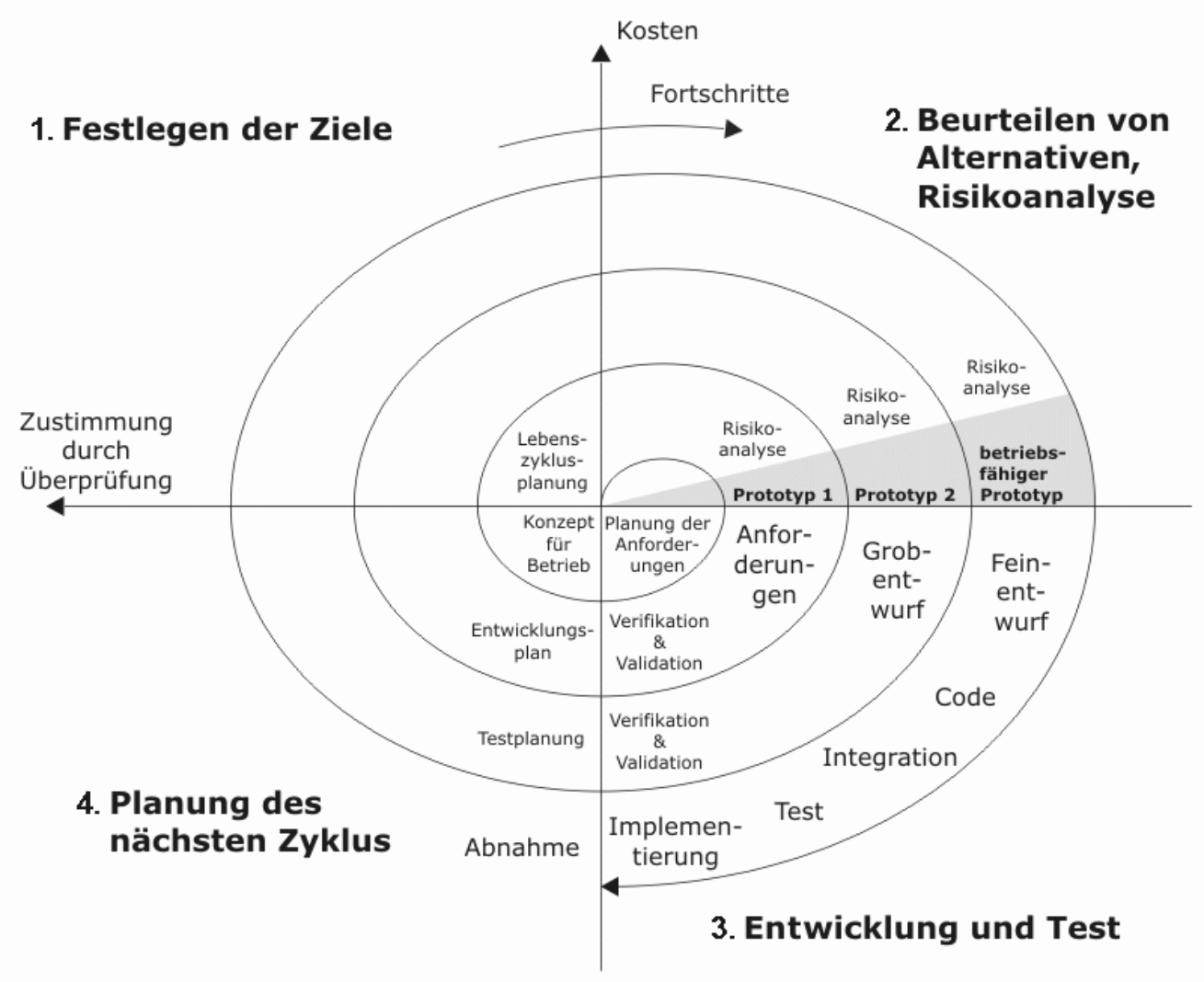


Exkurs: Browsen per Assistent



- Aktivierungscode sprechen
- Gerät öffnet Kommunikation mit Sprachdienst
- Sprachdienst führt Internetzugriff aus

Erinnerung: Entwicklung eines Dialogsystems



Dialogsystem = Softwaresystem

Prototyp 1

Mit einem ersten Prototypen *simuliert* man das geplante System.

Papierprototyp Ausdenken und Aufzeichnen möglicher Dialogabläufe

Feldstudien Beobachtungen im konkreten Anwendungsszenario

Wizard of Oz Rollenspiel mit verstecktem Experten

Ziel ist, einen guten Überblick der notwendigen Funktionalität des Systems zu bekommen, ohne bereits viel Implementationsarbeit leisten zu müssen. Hier sind *Anwendungsexperten* gefragt

Schrittweise Umsetzung

Die Auswertung des ersten Prototyps gibt einen ersten Eindruck vom Funktionsumfang und den Anforderungen.

- ist ein Dialogsystem überhaupt sinnvoll?
- ist die gewählte Dialogstrategie passend?
- kann man die Benutzer durch geeignete Wortwahl steuern?
- ist das System technisch realisierbar?
- ...

Jetzt kann man einen zweiten Prototypen implementieren.

VoiceXML-Anwendung

Die Interaktion mit dem Nutzer wird durch eine VoiceXML-Anwendung implementiert. Diese besteht aus

- *VoiceXML-Dokumenten*, die sprachliche Ein- und Ausgaben definieren, und dem
- *VoiceXML-Algorithmus* (sog. *Form Interpretation Algorithm*, FIA), der VoiceXML-Dokumente ausführt.

VoiceXML-Dokumente kann man anschaulich mit Formularen mit auszufüllenden Feldern vergleichen.

Aufbau einer VoiceXML-Anwendung

Eine VoiceXML-Anwendung besteht aus einem

Wurzeldokument bei dem die Anwendung startet und das immer verfügbar ist, sowie aus

Dialogdokumenten die einzelne Dialogschritte implementieren; diese werden nach Bedarf dynamisch geladen

Links verknüpfen die Dokumente miteinander

Ereignisse reagieren auf Systemzustände

Dialogdokumente sind automatisch mit dem Wurzeldokument verknüpft.

Links

Es gibt im wesentlichen zwei Arten von Links

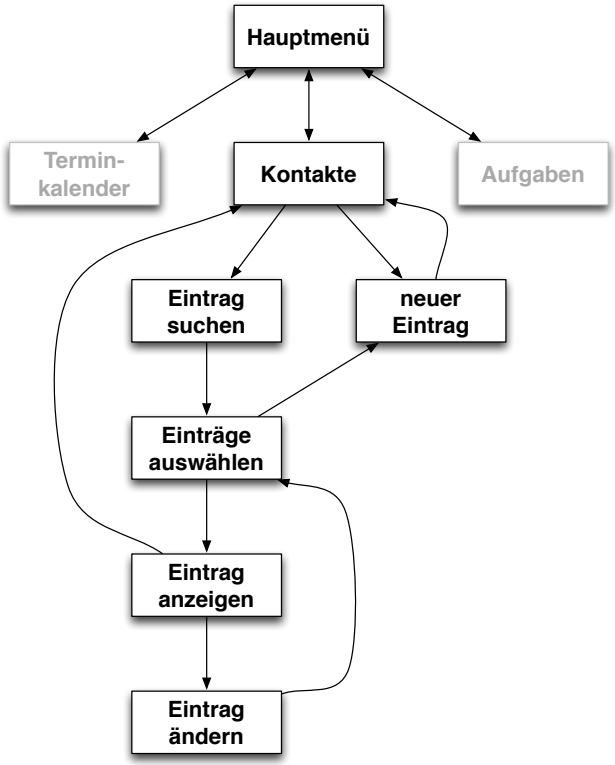
Sprung der Dialog wird an der angegebenen Stelle fortgesetzt

Aufruf der Dialog wird an der angegebenen Stelle fortgesetzt und kehrt nach deren Beendigung zurück

Das Ziel eines Links ist ein anderes Dialogdokument.

Beispiel: Kontakte, Termine, Aufgaben

Ein Dialogsystem zur Verwaltung von Kontakten, Terminen und Aufgaben implementiert den Zugriff aufs Adressbuch.



Überlegen Sie sich, ob die angegebenen Dialogschritte sinnvoll sind, welche Schritte miteinander verknüpft werden müssen, und wie der Dialog in den jeweiligen Schritten konkret aussehen kann.

Dialogschritt `Einträge auswählen`

System Ich habe N Einträge gefunden. Wählen Sie einen Eintrag per 'Ja' aus oder sagen Sie 'neuer Eintrag'.

1. Max Mustermann
2. Maxi Musterfrau
3. ...

Ende der Liste. 'Nochmal' vorlesen, 'neuer Eintrag' oder 'Ende'.

Nutzer Nochmal

System Ich habe N Einträge gefunden. Wählen Sie einen Eintrag ...

Unterbricht der Nutzer die Aufzählung mit 'Ja', dann geht der Dialog im Dialogschritt `Eintrag anzeigen` weiter. Welche Alternativen können Sie sich vorstellen? Ist klar, wohin 'Ende' führt?

Struktur und Ablauf

Die Dokumente und Links können als Netzwerk dargestellt werden, mit Dialogdokumenten als Knoten und Links als Verbindungen. Dieses Netzwerk ist die *statische* Struktur der VoiceXML-Anwendung.

Ein Dialog ist ein konkreter Durchlauf der statischen Struktur. Dabei können einzelne Dokumente wiederholt besucht werden, andere gar nicht - das ist der *dynamische* Ablauf der VoiceXML-Anwendung.

VoiceXML Standardisierung

aktueller Standard seit 2007 ist VoiceXML 2.1

- Standard des World Wide Web Consortium
- VoiceXML 2.1 Technical Report

sog. 'working draft' VoiceXML 3.0 von 2010

- 'vollständiges Redesign der Spezifikation'
- domänenspezifische Sprache
- VoiceXML 3.0 Technical Report

VoiceXML wird von vielen Herstellern unterstützt und weiterentwickelt.

VoiceXML: Ein-/Ausgabe

`<prompt>`	Sprachausgabe
`<audio>`	Wiedergabe einer Audiodatei, z.B. Ton, Earcon
`<grammar>`	Spracherkennung für eine Eingabe
`<record>`	Audioaufnahme ohne Erkennung
`<dtmf>`	Telefon-Wähltöne

VoiceXML: Attribute

Wie in XML können auch VoiceXML-Tags Attribute haben. Diese spezifizieren Eigenschaften des Tags.

- Schreibweise `attribut="Wert"`
- z.B. `<menu id="hauptmenue" dtmf="true" accept="approximate">`

Attribute stehen stets im öffnenden Tag und sollten immer in doppelte Anführungszeichen eingeschlossen sein.

VoiceXML: Navigation

`<menu>` Liste von Auswahlmöglichkeiten mit Sprung zu anderen Dialogdokumenten

`<link>` Sprung in ein anderes Dialogdokument

`<subdialog>` Aufruf eines Unterdialogs mit Rückkehr an die Stelle des Aufrufs

`<menu>` und `<link>` kehren nicht zurück, `<subdialog>` dagegen schon.

Formulare

Eine VoiceXML-Anwendung besteht meist aus einer Reihe von Formularen:

- System fragt nach benötigten Angaben
- Nutzer spricht diese Angaben
- System trägt den Inhalt in entsprechende Felder ein

Die Eingabe des Nutzers muss entweder zu einer Liste von Optionen passen, oder zu den Regeln einer Grammatik.

VoiceXML: Formulare

`<form>`	Formular
`<field>`	Formularfeld
`<filled>`	Ereignis, das ausgelöst wird, wenn bestimmte Felder gefüllt sind.
`<goto>`	Sprung zu einem Feld im Formular
`<submit>`	Absenden des Formulars an den Server

VoiceXML: Sprachausgabe

Ausgaben des Systems werden per Sprachsynthese generiert

- `<choice>` Auswahl von Optionen
- `<prompt>` Ausgabe von Text
- `<enumerate>` Ausgabe von Aufzählungen

oder durch Abspielen von Audiodateien:

- `<audio>` Abspielen einer Audiodatei

VoiceXML: Sprachsynthese

Die Sprachsynthese kann mit Tags recht fein gesteuert werden. Diese sind teilweise systemabhängig (hier die von Voxeos Prophecy TTS-System).

`<break>`	Pause einfügen
`<paragraph>`	Absatzgrenze markieren, z.B. durch Pause, Intonation
`<emphasis>`	Hervorhebung durch Betonung
`<prosody>`	Veränderung der Prosodie, z.B. F0, Sprechgeschwindigkeit, Dauer, Lautstärke, usw.
`<say-as>`	Formatierungsanweisung, z.B. Datums- und Zeitangaben, oder explizite Vorgabe der Aussprache
`<phoneme>`	Angabe des zu sprechenden Phonems in einem erlaubten Inventar

Beispiel Hello World

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xml:lang="de-DE">
  <form>
    <record name="username" maxtime="10s" finalsilence="1000ms">
      <prompt>Hallo. Was ist dein Benutzername?</prompt>
    </record>
    <block>
      <prompt>
        Willkommen <value expr="username"/>!
      </prompt>
     </block>
  </form>
</vxml>
```

Was macht dieser VoiceXML-Code?

Beispiel Veränderung der Prosodie

Das Prophecy TTS-System von Voxeo erlaubt die Modifikation der Prosodie über Attribute im Tag `<prosody>`:

`duration`	Dauer, z.B. `duration="2000ms"`
`rate`	Sprechgeschwindigkeit, z.B. `rate="fast"`
`volume`	Lautstärke, z.B. `volume="soft"`
`pitch`	Grundfrequenz f0, z.B. `pitch="high"`

Laut Dokumentation von Voxeo funktioniert das `prosody`-Attribut aktuell (10.06.2016) noch nicht.

VoiceXML: Ereignisse

Ereignisse werden durch Systemzustände ausgelöst. Sie rufen Dialogdokumente auf.

`<help>`	Aufruf der Hilfefunktion
`<noinput>`	keine Eingabe erfolgt, z.B. nach X Sekunden
`<nomatch>`	Eingabe nicht erkannt
`<error>`	sonstiger Fehler ist aufgetreten
`<filled>`	Formularfelder sind gefüllt
`<catch>`	Aktion, die beim Auftreten anderer Ereignisse ausgeführt wird

Attribute sind z.B. `count`, `cond`, usw.

Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>

<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE"
    application="PIMroot.vxml">

  <noinput count="1">Ähem</noinput>
  <noinput count="2">Wie bitte?</noinput>
  <noinput count="3">Leider habe ich nichts gehört.</noinput>


  <form>
    <record name="username" maxtime="10s" finalsilence="1000ms">
      <prompt>Hallo. Was ist dein Benutzername?</prompt>
    </record>
    <block>
      <prompt>
        Willkommen <value expr="username"/>!
      </prompt>
     </block>
  </form>
</vxml>
```

VoiceXML Algorithmus

Der *Form Interpretation* Algorithmus führt VoiceXML Dokumente aus:

- wähle nächstes zu besuchendes Element, z.B. Dokument oder Formularfeld
- synthetisiere Sprache für die Ausgabe, z.B. Menüs, Ansagen, usw.
- nimm Sprache des Benutzers auf und vergleiche sie mit Optionenliste oder analysiere sie mit Grammatik
- tritt ein Ereignis ein, behandle es sofort, z.B. Hilfetext ausgeben bei Eingabe von 'Hilfe'
- verarbeite die Eingabe oder das Ereignis

Standardmäßig wird das nächste nicht gefüllte Element ausgewählt.

VoiceXML Anbieter

Geschäftsidee

- zuerst gratis Entwicklung
- dann kostenpflichtiger Dienst

Angebote

- mehrsprachige Ressourcen
- freie Entwickertools
- Tutorials, Dokumentation, Disussionsforen

Beispiel Voxeo

voxeo.com

- VoiceXML 2.1

Tools

- Application Manager verwaltet VoiceXML Anwendungen
- File Manager verwaltet Dateien und enthält VoiceXML-Editoren
- Debugger gibt Protokolle von Anrufen zur Fehlersuche aus
- ...

Voxeo: Schritte

1. Benutzerkonto anlegen, Bestätigungsmail abwarten
2. mit Login bei voxeo anmelden
3. Menü `Documentation > Quick Start Guide` anklicken
4. VoiceXML Dokument erstellen und hochladen
5. mit Skype o.ä. anrufen

HelloWorld.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version = "2.1" xml:lang="de-DE">

  <form>
    <block>
      <prompt>
        Hallo. Willkommen beim ersten VoiceXML-Programm von
        Christoph Draxler.
      </prompt>
    </block>
  </form>
</vxml>
```

Was macht diese VoiceXML-Anwendung?

Tipps

- Telefonnummer der Anwendung kommt per Mail
- Code aus dem Tutorial verwenden und schrittweise verfeinern
- immer den Debugger mitlaufen lassen
- nationale Rufnummern sind bei vielen Anbietern gratis, oder eine Skype-Nummer verwenden

Verwendete VoiceXML-Tags

`<block>`	gruppiert VoiceXML-Befehle
`<audio>`	Wiedergabe einer Audiodatei, z.B. Ton, Earcon
`<catch>`	Aktion, die beim Auftreten anderer Ereignisse ausgeführt wird
`<choice>`	Auswahl von Optionen
`<dtmf>`	Telefon-Wähltöne
`<enumerate>`	Ausgabe von Aufzählungen
`<error>`	Ereignis, sonstiger Fehler ist aufgetreten
`<field>`	Formularfeld
`<filled>`	Ereignis, das ausgelöst wird, wenn bestimmte Felder gefüllt sind.
`<form>`	Formular
`<goto>`	Sprung zu einem Feld im Formular
`<grammar>`	Spracherkennung für eine Eingabe
`<help>`	Ereignis, Aufruf der Hilfefunktion
`<link>`	Sprung in ein anderes Dialogdokument
`<menu>`	Liste von Auswahlmöglichkeiten mit Sprung zu anderen Dialogdokumenten
`<nomatch>`	Ereignis, Eingabe nicht erkannt
`<noinput>`	Ereignis, keine Eingabe erfolgt, z.B. nach X Sekunden
`<prompt>`	Sprachausgabe
`<record>`	Audioaufnahme ohne Erkennung
`<subdialog>`	Aufruf eines Unterdialogs mit Rückkehr an die Stelle des Aufrufs
`<submit>`	Absenden des Formulars an den Server

New Personal Information Manager (NPIM)

In ihrer Magisterarbeit hat Angelika Berwein ein Rohgerüst für eine VoiceXML-Anwendung erstellt. Diese entwickeln wir nun schrittweise unter dem Namen *NPIM* weiter.

1. Wurzeldokument
2. Eingabe des Benutzernamens
3. Begrüßung
4. Fehlerbehandlung, Hilfestellung
5. Auswahl aus Menü
6. Erkennung einer Eingabe per Grammatik

Mit `TODO:` sind die Stellen im Code markiert, die im Laufe der Entwicklung ersetzt oder ergänzt werden.

Wurzeldokument NPIMroot.vxml

Zu Beginn besteht das Wurzeldokument nur aus der Definition einer globalen Variablen sowie einer Willkommensnachricht.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">

  <!-- TODO: Fehlerbehandlung, Hilfestellung -->

  <!-- globale Variablen -->
  <var name="benutzername"/>

  <!-- TODO: Testausgabe durch echte Dialogaufrufe ersetzen -->
  <form id="root">
    <block>
      <prompt>
        Herzlich willkommen beim Personal Information Manager des
        Seminars Dialogsysteme.
      </prompt>
    </block>
  </form>
</vxml>
```

In späteren Schritten wird dieses Wurzeldokument erweitert.

NPIMusername.vxml

NPIM fragt nun nach dem Benutzernamen und speichert ihn in der lokalen Variable username

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml"
  xml:lang="de-DE" application="NPIMroot.vxml">

  <form id="login">
    <record name="username" maxtime="10s" finalsilence="1000ms">
      <prompt>Hallo. Was ist dein Benutzername?</prompt>
    </record>

    <block>
      <!-- gibt die Werte aller Variablen in namelist zurück -->
      <return namelist="username"/>
    </block>
  </form>
</vxml>
```

Code-Analyse

Zuerst wird der gesprochene Name aufgenommen; die Aufnahme wird als Audioaufnahme in der Variablen `username` gespeichert.

```
<record name="username" maxtime="10s" finalsilence="1000ms">
  <prompt>Hallo. Was ist dein Benutzername?</prompt>
</record>
```

Dann wird diese Variable als Rückgabeparameter an den aufrufenden Code zurückgegeben.

```
<block>
  <return namelist="username"/>
</block>
```

Die Eingabe des Benutzernamens muss jetzt noch in `NPIMroot.vxml` eingebaut werden.

Begrüßung

Das Wurzeldokument `NPIMroot.vxml` wird nun um den Aufruf des Dialogs für die Eingabe des Nutzernamens und die Ausgabe dieses Namens erweitert.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE">

  <!-- TODO: Fehlerbehandlung, Hilfestellung -->

  <!-- globale Variablen -->
  <var name="benutzername"/>

  <form id="root">
    <subdialog name="login" src="NPIMusername.vxml#login">

      <filled>
        <!-- Ergebnis des Output-Parameters "username" aus dem Formular "login" wird
             der globalen Variable "benutzername" zugewiesen -->
        <assign name="benutzername" expr="login.username"/>
      </filled>
    </subdialog>

    <block>
      <prompt>Willkommen <value expr="benutzername"/>!</prompt>

      <!-- TODO: Gehe zum Hauptmenü -->
    </block>
  </form>
</vxml>
```

Fehlerbehandlung

Das Wurzeldokument `NPIMroot.vxml` wird nun um Tags für das Nichterkennen bzw. das Fehlen von Eingaben sowie Hilfestellung erweitert.

```
<nomatch>
  Leider habe ich dich nicht verstanden.
  <throw event="help"/>
</nomatch>

<noinput count="1">
  Es wurde keine Eingabe erkannt. Bitte sprich lauter.<reprompt/>
</noinput>
<noinput count="2">
  Es wurde wieder keine Eingabe erkannt. Auf Wiederhören.<disconnect/>
</noinput>

<help><reprompt/></help>
```

Diese Zeilen stehen oben im Dokument, nach dem öffnenden `<vxml>` Tag.

Was macht der Code, wenn zwei Mal gar keine Eingabe erfolgt ist?

Aufruf des Hauptmenüs in NPIMRoot.vxml

Der Dialog mit dem Hauptmenü steht in der Datei NPIMMenu.vxml.

```
...
<form id="root">
  <subdialog name="login" src="NPIMusername.vxml#login">
    ...
  </subdialog>

  <block>
    <prompt>Willkommen <value expr="benutzername"/>!</prompt>

    <goto next="NPIMmenu.vxml#mainmenu"/>
  </block>
</form>
...
```

<goto next="NPIMmenu.vxml#mainmenu"/> ist der Sprungbefehl: VoiceXML springt in das Dokument NPIMMenu.vxml, dort direkt zu dem Tag mit dem Attribut id=mainmenu und setzt dort den Dialog fort.

Auswahl aus Menü

Die Datei `NPIMmenu.vxml` bietet drei Optionen zur Auswahl an.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE" application="NPIMroot.vxml">

  <help>Sage <enumerate />.</help>

  <menu id="mainmenu">
    <prompt>
      Du bist jetzt im Hauptmenü. Wähle eine der folgenden Optionen:
      <enumerate/>
    </prompt>

    <choice next="NPIMcalendar.vxml">Kalender</choice>
    <choice next="NPIMtasklist.vxml">Aufgabenliste</choice>
    <choice next="NPIMcontacts.vxml">Adressbuch</choice>
  </menu>
</vxml>
```

Was macht der Code? Was passiert, wenn keine Eingabe erfolgt?
Und was, wenn die Eingabe zum ersten Mal nicht erkannt wurde?

Auswahl (Alternative)

Eine Variante des Hauptmenüs – was ist hier anders?

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xml:lang="de-DE" application="NPIMroot.vxml">

  <help>Sage <enumerate/></help>

  <menu id="hauptmenue">
    <prompt>
      Du kannst nun den Kalender, Deine Aufgaben oder Termine auswählen.
    </prompt>

    <choice next="NPIMcalendar.vxml">Kalender</choice>
    <choice next="NPIMtasklist.vxml">Aufgabenliste</choice>
    <choice next="NPIMcontacts.vxml">Adressbuch</choice>
  </menu>
</vxml>
```

Was macht der Code?

Diskussion

Ein Auswahlmenü ist sehr beschränkt in seinen Möglichkeiten:

- es erkennt nur genau die vorgegebenen Wörter
- es kann nur eine Angabe pro Eingabe geliefert werden

Dazu kommt:

- Menschen verwenden oft Floskeln, Füllwörter und ähnliches
- diese Floskeln sind meist kontextabhängig
- formatierte Eingaben, z.B. Datum, Uhrzeit, Telefonnummer können sehr verschieden gesprochen werden

Grammatiken erlauben eine flexiblere Erkennung von Nutzereingaben

Grammatiken: Motivation

Hier ein Auszug der Antworten auf die Frage 'Wie spät ist es jetzt?' in der Sprachdatenbank Ph@ttSessionz:

```
vierzehn Uhr sieben
es ist jetzt zehn Uhr einundvierzig
jetzt haben wir es genau acht Uhr und neunundfünfzig
jetzt ist es elf Uhr neununddreißig
kurz nach acht
zehn vor neun
es ist jetzt viertel nach zwei
es ist zehn Uhr dreiundvierzig
fünf Minuten nach zwölf
...
```

Einerseits sind die Äußerungen ziemlich verschieden, andererseits aber doch auch ähnlich.

Grammatiken (Forts.)

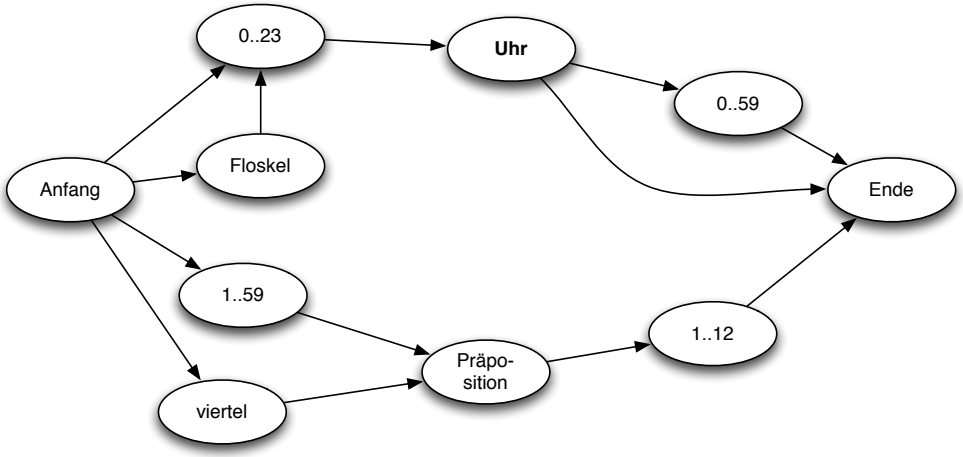
So unterscheidet sich z.B. beiden Äußerungen in

```
vierzehn Uhr sieben
es ist jetzt zehn Uhr einundvierzig
```

prinzipiell nur durch die vorangestellten Füllwörter *es ist jetzt* – dann folgen die Stunden, gefolgt von 'Uhr' und den Minuten.

Entwurf einer Grammatik

Ein einfaches Diagramm erlaubt den Entwurf einer Grammatik. Es hat einen Anfangs- und einen Endzustand, dazwischen beschriftete Knoten. Diese Beschriftungen entsprechen den an dieser Stelle erlaubten Ausdrücken. Ein Uhrzeitausdruck ist dann gültig, wenn er einen Pfad durch das Diagramm beschreibt.



Dieses Diagramm kann man einfach erweitern, und es erlaubt auf intuitive Weise zu testen, ob ein bestimmter Uhrzeitausdruck erkannt wird oder nicht.

Grammatiken (Forts.)

Etwas formaler könnte man die beiden Äußerungen auch mit Regeln wie diesen beschreiben:

```
$UHRZEIT = $FLOSKEL $STUNDEN Uhr $MINUTEN
$STUNDEN = 0 | 1 | 2 | 3 | ... | 23
$MINUTEN = 1 | 2 | 3 | ... | 59
$FLOSKEL = [es] [ist] [jetzt]
```

Elemente in [] sind optional, der senkrechte Strich | trennt Alternativen, $ bezeichnet Symbole, die durch weitere Symbole oder Nutzereingaben ersetzt werden.

Grammatiken (Forts.)

Zu einer Grammatik fehlen noch zwei Dinge:

1. Wo startet die Grammatik?
2. Was soll die Grammatik zurückgeben?

In VoiceXML gibt man die Startregel einer Grammatik explizit an, z.B. `root $UHRZEIT`.[1]

[1] die genaue Syntax hängt vom verwendeten Grammatikformat ab

Grammatiken (Forts.)

Die *Semantik* einer Regel ist das, was die Grammatik erkennen soll – das ist nicht unbedingt genau das, was der Nutzer gesagt hat! So haben die Wörter 'ja', 'jawohl', 'klar' und 'okay' alle die Semantik 'ja'.

In VoiceXML schreibt man die Semantik einer Grammatikregel in geschweifte Klammern:

```
$UHRZEIT = $FLOSKEL $STUNDEN Uhr $MINUTEN
  {out.hour = rules.STUNDEN; out.minute = rules.MINUTEN}
```

gibt ein Objekt mit den Feldern `hour` und `minute` zurück; die Felder enthalten die entsprechenden Nutzereingaben der Regeln `$STUNDEN` bzw. `$MINUTEN`.

Grammatiken: Aufgaben

Schreiben Sie eine Grammatik für

- deutsche Mobiltelefonnummern
- eine kurze informelle Wegbeschreibung
- eine Pizzabestellung mit Größenangabe und Auflagen

Überlegen Sie sich ein paar typische Äußerungen und erstellen Sie dann entsprechende Regeln.

Grammatiken einbinden

Grammatiken können entweder

inline als Inhalt von `<grammar>`-Tags oder

extern im `src`-Attribut eines `<grammar>`-Tags als Referenz auf eine externe Datei stehen.

Der Gültigkeitsbereich hängt davon ab, *wo* sie eingebunden werden: Grammatiken, die im Wurzeldokument eingebunden werden, gelten in der gesamten Anwendung, alle anderen nur innerhalb des Dokument bzw. des Tags, wo sie eingebunden werden.

Beispiel für eine globale Grammatik

Die folgende inline-Grammatik beendet den Dialog, sobald in irgendeinem Dialogschritt *Ende, tschüss*, o.ä. gesagt wird.

```
<link next="#ende">
  <grammar mode="voice" type="application/srgs">
    #ABNF 1.0;
    language de-DE;

    public $ENDE = ende | auf Wiederhören | tschüss | quit;
  </grammar>
</link>
```

Dazu muss a) diese Grammatik im Wurzeldokument stehen, und b) dort ein Element mit dem Attribut `id=ende` stehen.

Grammatikformate

Es gibt unterschiedliche Formate:

XML die Grammatik ist ein XML-Dokument

ABNF Augmented Backus-Naur Format

Die hier verwendeten Grammatiken sind in ABNF, denn das Format ist auch für Menschen gut zu lesen und recht kompakt.

Grammatiken in XML-Format haben den Vorteil, dass derselbe Parser zum Prüfen der Grammatik verwendet werden kann wie für die eigentliche XML-Anwendung.

Zwischenstand

Die VoiceXML-Anwendung NPIM besteht nun aus den folgenden Dateien

NPIMroot mit globaler Fehlerbehandlung und Ende-Grammatik

NPIMusername zur Eingabe des Benutzernamens

NPIMmenu für das Hauptmenü

Es gibt eine globale Grammatik für das Beenden des Anrufs, und eine einfache Fehlerbehandlung.

Grammatik Mobilnummern

Die folgende Grammatik ist ein Vorschlag zum Erkennen deutscher Mobilnummern.

```
#ABNF 1.0 UTF-8;
language de-DE;
tag-format <semantics/1.0>;
root $MOBILNR;

$MOBILNR = $PROVIDER $RUFNUMMER

$PROVIDER = 0 1 $ZIFFER <2>

$RUFNUMMER = $ZIFFEROHNENULL $ZIFFER <7-8>

$ZIFFEROHNENULL = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

$ZIFFER = 0 | $ZIFFEROHNENULL
```

Prüfen Sie, ob die Grammatik Ihre Mobilnummer erkennen könnte, so wie Sie sie meistens sprechen.

Grammatik Mobilnummern

```
<grammar xml:lang="de-DE" root="MOBILNUMMER">
  <rule id="MOBILNUMMER">
    <item>
      <ruleref uri="#PROVIDER"/>
    </item>
    <item>
      <ruleref uri="#RUFNUMMER"/>
    </item>
  </rule>

  <rule id="RUFNUMMER">
    <item>
      <ruleref uri="#ZIFFEROHNENULL"/>
    </item>
    <item repeat="6-7">
      <ruleref uri="#ZIFFER"/>
    </item>
  </rule>

  <rule id="PROVIDER">
    <item>0</item>
    <item>1</item>
    <item repeat="2">
      <ruleref uri="#ZIFFEROHNENULL"/>
    </item>
  </rule>
```

Fortsetzung folgt!

Mobilnummern Grammatik (Forts.)

```
  <rule id="ZIFFER">
    <one-of>
      <item>0</item>
      <item>
        <ruleref uri="#ZIFFEROHNENULL"/>
      </item>
    </one-of>
  </rule>

  <rule id="ZIFFEROHNENULL">
    <one-of>
      <item>1</item><item>2</item><item>3</item><item>4</item>
      <item>5</item><item>6</item><item>7</item><item>8</item><item>9</item>
    </one-of>
  </rule>
</grammar>
```

Länger als die Version in ABNF, aber gleich ausdrucksstark.

Barge-in

Mit *barge-in* bezeichnet man das Hineinreden in die Ansage des Systems durch den Benutzer. Das ist in vielen Fällen sinnvoll, z.B.

- zur schnellen Auswahl eines Menüitems, um das Anhören aller Optionen zu vermeiden
- um eine Hilfe anzufordern, z.B. Wiederholen, 'Hilfe', usw.
- um den Dialog zu beenden

In der NPIM Anwendung kann der Benutzer zu jeder Zeit[2] den Dialog durch das Äußern eines der Ende-Wörter beenden.

[2] Wirklich zu jeder Zeit? Wann evtl. nicht?

Dialogsystem-Architekturen: Erinnerung

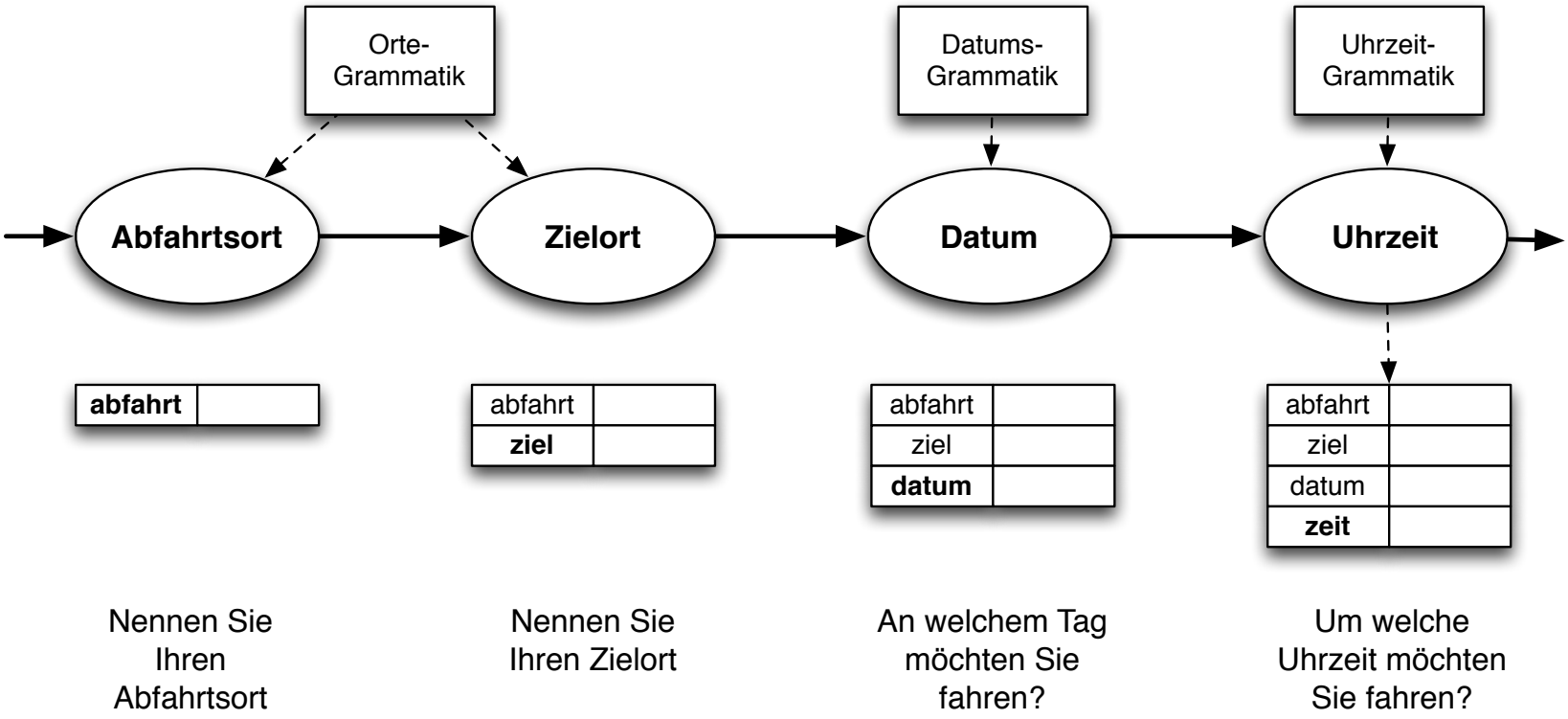
Im Seminar haben wir zwei Dialogsystem-Architekturen besprochen

finite state oder *system initiative*, ein Item pro Eingabe

framebasiert oder *mixed initiative*, mehrere Items pro Eingabe

Beide Architekturen bzw. Arten der Initiative können in VoiceXML implementiert werden.

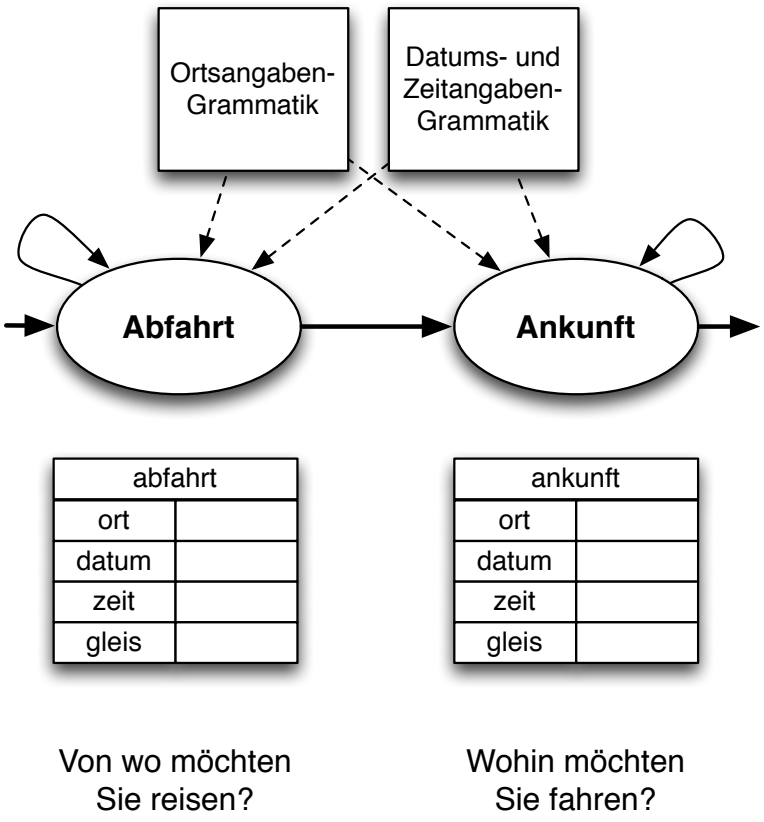
Beispiel: finite state Architektur



In jedem Dialogschritt wird eine neue Information erfragt und gespeichert. Macht der Benutzer weitere Angaben, dann werden diese ignoriert bzw. sie führen eventuell sogar dazu, dass die Eingabe nicht erkannt wird[3].

[3] Wieso?

Beispiel: framebasierte Architektur



Der Dialog geht vom Zustand *Abfahrt* in den Zustand *Ankunft* wenn ausreichend viele Felder im Frame `Abfahrt` gefüllt sind. Wenn der Benutzer weitere Angaben gemacht hat, z.B. Datum und Uhrzeit, dann werden die im aktuellen Frame ebenfalls abgelegt. Damit müssen sie nicht noch einmal erfragt bzw. können sie in späteren Dialogschritten verwendet werden.

VoiceXML Implementierung: finite state

Standardmäßig implementiert VoiceXML system-initiative Dialoge und damit eine finite state Architektur.

Der VoiceXML Form Interpretation Algorithmus

- geht ein Formular von oben nach unten durch
- versucht, die Felder zu füllen. Sobald ein Feld durch Erkennen der Benutzereingabe gefüllt werden kann, wird für dieses Feld ein `filled`-Ereignis ausgelöst
- reagiert auf dieses Ereignis und überträgt den erkannten Wert in die dem Feld zugeordnete Variable.

Die Abfolge der Dialogschritte ist durch den Algorithmus festgelegt, jede Eingabe geht einen Dialogschritt weiter.

VoiceXML Implementierung: framebasiert

Mit den folgenden Attributen und Tags kann ein gemischt-initiativer und damit framebasierter Dialog implementiert werden.

slot	das Attribut gibt an, durch welche Grammatikregel es gefüllt werden kann
<initial>	der Inhalt des Tags wird nur einmal zu Beginn des aktuellen Dialogschritts ausgegeben
<filled>	enthält Code, der ausgeführt wird, wenn eines, alle oder bestimmte Felder gefüllt wurde(n)

Damit kann eine komplexe Benutzereingabe mit der Grammatik analysiert und können die erkannten Items den entsprechenden Formularfeldern zugeordnet werden.

Validierung vs. Evaluation

Validierung prüfen, ob ein System *objektiv* messbaren Kriterien genügt

- Vollständigkeit des Vokabulars
- Wortfehlerrate
- garantierte Reaktionszeit
- ...

Evaluation prüfen, ob ein System *heuristischen* Kriterien genügt.

- Natürlichkeit
- Nützlichkeit
- Benutzerakzeptanz
- ...

Validierung und Evaluation (cum grano salis[4])

Gegeben sei der Text

> *dise masname eliminirt schon di gröste felerursache in der grundschule, den sin oder unsin unserer konsonantenverdopelung hat onehin nimand kapirt.*

Was bedeuten hier *Validierung* und *Evaluation*?

Validierung nur 7 von 20 Wörtern ($= 35\%$) orthographisch korrekt

Evaluation der Inhalt ist aber gut verständlich

Warum können wir den Text verstehen? Unter welchen Randbedingungen?

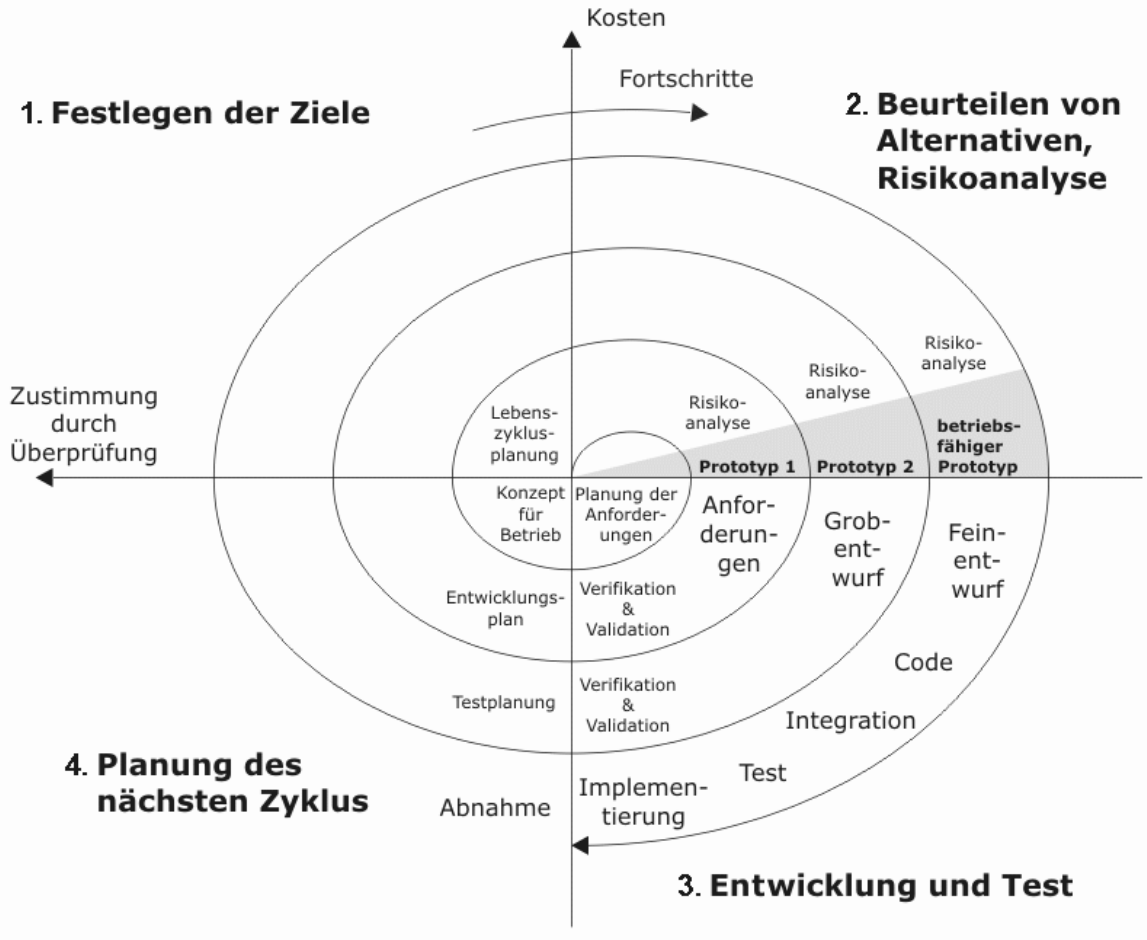
[4] nicht ganz ernst zu nehmen

Validierung

Prüfen, ob ein System die spezifizierten objektiven Anforderungen erfüllt

- Aufdecken von Fehlern und Fehlfunktionen
- in unterschiedlichen Phasen des Entwicklungs- und Lebenszyklus des Systems

Beispiel Software-Entwicklung: Spiralmodell



In jedem Zyklus findet eine Validierung statt.

https://commons.wikimedia.org/wiki/File:Spiralmodel_nach_Boehm.png

Validieren von Dialogsystemen

Dialogsysteme sind komplexe Softwaresysteme

- Testverfahren aus dem Software Engineering
- strukturelle bzw. Funktions- und Performanztests

Meist sind verschiedene Tests notwendig

White Box Test

Test der einzelnen Komponenten

- bereits während der Implementationsphase
- Systemimplementation ist bekannt
  - Source Code liegt vor
  - Datenfluss ist dokumentiert
- umfasst Einheiten- und Integrationstests

White Box Test

Einheiten vs. Integrationstest

- einzelne Module vs.
- Zusammenspiel von Modulen

Vordefinierte Test-Fälle

- Eingabe X $\rightarrow$ Ergebnis Y
- erwartetes = beobachtetes Ergebnis?

White Box Tests sind teilweise automatisierbar (sog. *test suite*)

Probleme

White Box Tests sind nicht ausreichend, denn

- die Leistung eines komplexen Systems ist *nicht* aus der Leistung der einzelnen Komponenten ableitbar
- auch wenn es evtl. Korrelationen gibt

Daher muss eine eigene Evaluation des Gesamtsystems *in der Anwendung* erfolgen.

Black Box Test

Test eines Systems, ohne Kenntnis des inneren Aufbaus

- formale und informale Spezifikationen
- Testfälle
- Anwendung auf das Gesamtsystem

Black Box Tests sind in der Regel aufwendiger als White Box Tests

- benötigen eigenes Personal
- informale Spezifikationen schlecht zu prüfen
- Gesamtsystem evtl. komplex
- ...

Black Box Test

in der Test- und vor der Einführungsphase

- nach dem White Box Test

Prüfen der Spezifikation

- funktionale vs.
- nicht-funktionale Anforderungen

Performanz- und Zuverlässigkeitstests

funktionale Anforderungen

Interaktion von System und Umgebung

- unabhängig von der Implementierung

Umgebung

- Anwender, externe Systeme

nicht-funktionale Anforderungen

stehen nicht unmittelbar in Bezug zur Funktionalität [?].

- Bedienbarkeit
- Zuverlässigkeit
- Performanz
- Wartbarkeit
- ...

Beispiel PIM

funktionale Anforderungen

- PIM verwaltet Kontakte, Aufgaben, Termine eines Nutzers
- Sprachein- und -ausgabe über das Telefon

nicht-funktionale Anforderungen

- das System darf höchstens einmal pro Woche ausfallen
- Dialoge innerhalb von 90sec abschließen

Evaluation

Untersuchung des Systemverhaltens in der Anwendung

- in der Einführungsphase und
- während der Nutzung

mehrstufige Evaluationen möglich

- Pilot- und Feldevaluationen
- Nutzer- und Akzeptanzuntersuchungen

Teil- und Komplett-Evaluationen

Systemevaluation

Quantitative Beschreibung der Leistung. Typische Maße sind

- Wortfehler-Rate der Spracherkennung
- Transaktionserfolg
- Anzahl Turns/Dialogdauer
- Korrektur-Rate
- Angemessenheit der Systemantworten

Nutzerevaluation

Qualitative Beschreibung von Nutzer-Beurteilungen

- Likert-Skalen vs. freie Antworten
- Mean opinion scores

In der Regel Erhebungen im Interview oder per Fragebogen

Typische Fragen

- Es war einfach, die Aufgabe zu erledigen
- Es war einfach, sich im System zurechtzufinden
- Das System hat mich verstanden
- Die Sprachausgabe war leicht zu verstehen
- Die Antwortzeit des Systems war kurz
- Das System hat stets so reagiert, wie ich es erwartet habe
- ...

Performanzmessung

Maß für die Leistung eines Systems in Bezug auf eine Referenz

- für Einzelkomponenten einfach
- für komplexe Systeme problematisch

Referenz: Beispiele

Spracherkennung erkannter Text vs. Experten-Transkription

Textverstehen ausgefüllte Slots in der internen Wissensrepräsentation

Sprachverstehen generierte vs. Expertenabfrage an die Wissensquelle

Sprachsynthese Eingabestring vs. Transkription für Sprachsynthese

Referenz: Probleme

- Automatisierung nur über aufwendiges Protokoll
- Interaktion über mehrere Schritte erlaubt Varianten und Ambiguitäten, die eine Zuordnung zu Systemkomponenten schwierig machen
- Vergleich mit menschlichen Experten ist unfair

Dialogsystemevaluation

im Labor

- automatisch erstellte Protokolldaten
- Expertenbeurteilung des Dialogs
  - Transkriptionen der Dialogverläufe
  - Fragebogen, freie Interviews
- beschränkte Anzahl Versuchspersonen

Dialogsystemevaluation

im produktiven Systemeinsatz

- nur Protokolldaten
- nur stichprobenartige Transkripte
- kaum Feedback
- Massendaten erfodern Automatisierung und statistische Modellierung

System muss voll funktionsfähig sein

Beispiel: Bahnauskunftsdialoge

Globale Maße

- Dauer, Anzahl Turns, mittlere Wartezeit

Spracherkennung

- falsch bzw. nicht erkannte Eingaben

Sprachsynthese

- Verständlichkeit, Natürlichkeit
- Artikulation, Aussprache

Beispiel: Bahnauskunftsdialoge

Dialogstrategie

- Dialoginitiative
- Zielerreichung, Erfolg
- Informationsstückelung
- Nachfragen, Korrekturen

Qualität des Systems

- Zufriedenheit
- Wert

Evaluationsframeworks

Berechnen ein Maß aus verschiedenen Einzelmessungen

PARADISE PARAdigm for DIalog System Evaluation [?]

SERVQUAL Evaluation von Dienstequalität anhand subjektiver Kriterien [?]

QoS Quality of Service [?]

und weitere...

Quality of Service

Analyse von Faktoren [?]

- agent factors
- task factors
- user factors
- environmental factors
- contextual factors

QoS: Agent factors

- Spracherkennung
- Sprechererkennung
- Sprachverstehen
- Dialogsteuerung
- Sprachsynthese

d.h. in der Regel Systemkomponenten

QoS: Task factors

- Aufgabentyp
- Aufgabendomäne
- Aufgabenkomplexität
- Aufgabenhäufigkeit
- Aufgaben-Auswirkungen
- Aufgaben-Übertragbarkeit

QoS: User factors

- Anzahl der Nutzer
- Alter und Geschlecht
- Aufgaben-Erfahrung
- System-Erfahrung
- Motivation
- physische Faktoren
- Sprache, Akzent, Dialekt

QoS: Environmental factors

'physische' Faktoren

- Mikrofontyp und -position
- Übertragungskanal
- Raum- und Umgebungsakustik
- Technik

QoS: Contextual factors

'nicht-physische' Faktoren

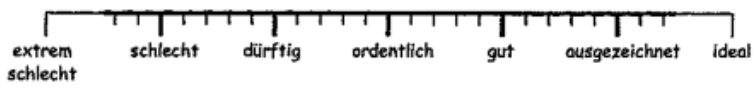
- Zugang zum System
- Verfügbarkeit des Systems
- Kosten der Interaktion
- Ähnlichkeit mit anderen Systemen

Beispiel: BoRIS

Bochumer Restaurant Informationssystem [?]

1. Ihr Qualitätsurteil über BoRIS:

1.1 Wie würden Sie Ihren Gesamteindruck von BoRIS einschätzen?



1.2 Wie würden Sie BoRIS einem Freund beschreiben?

-
-
-
-
-

1.3 Was würden Sie bei BoRIS verbessern?

-
-
-
-

Beispiel: BoRIS (2)

2. Bitte beurteilen Sie die folgenden Aussagen über Ihre Erfahrung mit BoRIS:

2.1 BoRIS hat genau so funktioniert, wie ich es mir vorgestellt habe.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.2 BoRIS bietet umfangreiche Informationen über Bochumer Restaurants.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.3 Ich bekam immer die gewünschte Antwort auf meine Eingaben.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.4 Ich spreche lieber mit einem Menschen als mit BoRIS .

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.5 Ich konnte mich äußern wie in einem normalen Gespräch mit einem Menschen.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.6 Ich bin davon überzeugt, dass BoRIS mir richtige Informationen geliefert hat.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.7 Das Gespräch mit BoRIS hat schnell zur gewünschten Information geführt.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.8 Der Anruf bei BORIS hat sich gelohnt.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.9 Jeder kann mit BoRIS ohne Probleme umgehen.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

2.10 Durch das Gespräch mit BoRIS fühlte ich mich :

wohl	☐	☐	☐	☐	☐	unwohl
sicher	☐	☐	☐	☐	☐	unsicher
ruhig	☐	☐	☐	☐	☐	aufgeregt
zufrieden	☐	☐	☐	☐	☐	unzufrieden

2.11 Ich habe die Fragen von BoRIS beantwortet:

stichwortartig ☐ in ganzen Sätzen ☐ teils-teils ☐

Beispiel: BoRIS (3)

3. Was Sie bei BoRIS sehr gestört hat:

3.1 BoRIS hat plötzlich das Gespräch unterbrochen und ist spontan zu einem vorherigen Menupunkt zurückgekehrt.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.2 BoRIS hat das Gespräch oft mittendrin abgebrochen.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.3 BoRIS konnte die Restaurantadressen nicht wiederholen.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.4 BoRIS bietet keine detaillierteren Informationen über die gewünschten Restaurants.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.5 Ich musste oft meine Eingaben wiederholen.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.6 Ich konnte mir die Auswahlkriterien am Anfang des Gesprächs nicht merken.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.7 Die Pausen zwischen meinen Eingaben und den Antworten von BoRIS waren zu lang.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.8 Die Ansage der Restaurantnamen und der Adressen war nicht deutlich.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.9 Der Weg bis zu der gewünschten Information war umständlich.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

3.10 Es war mir nicht klar, wie ich meine Eingaben modifizieren soll, um schnell die gewünschte Information zu bekommen.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

Beispiel: BoRIS (4)

4. Für einen guten Gesprächsverlauf mit BoRIS ist es sehr wichtig,

4.1 dass BoRIS seine Fragen wiederholt, wenn ich nicht mehr weiter weiß.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.2 dass BoRIS meine Eingaben immer gut versteht.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.3 dass BoRIS klare Fragen stellt.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.4 dass BoRIS schnell auf meine Eingaben reagiert.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.5 dass BoRIS einfach zu bedienen ist.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.6 dass BoRIS passende Informationen auf meine Anfragen gibt.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.7 dass ich BoRIS Fragen stellen kann.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.8 dass ich - nachdem ich die gewünschte Information bekommen habe - das Gespräch weiterführen kann, solange ich noch zusätzliche Fragen habe.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.9 dass ich von Anfang an weiß, wie ich meine Eingaben formulieren soll.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.10 dass ich von Anfang an weiß, welche Art von Informationen mir BoRIS liefern kann und welche nicht.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.11 dass ich von BoRIS eine Bestätigung meiner Eingaben bekomme.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.12 dass die Pausen nach meinen Eingaben von einen "Warten Sie einen Moment" ausgefüllt werden.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.13 dass es einfach ist, die Eingaben zu modifizieren.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.14 dass das Gespräch ohne Unterbrechungen verläuft.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

4.15 dass BoRIS mit einer Hilfefunktion ausgestattet ist.

trifft zu ☐ ☐ ☐ ☐ ☐ trifft nicht zu

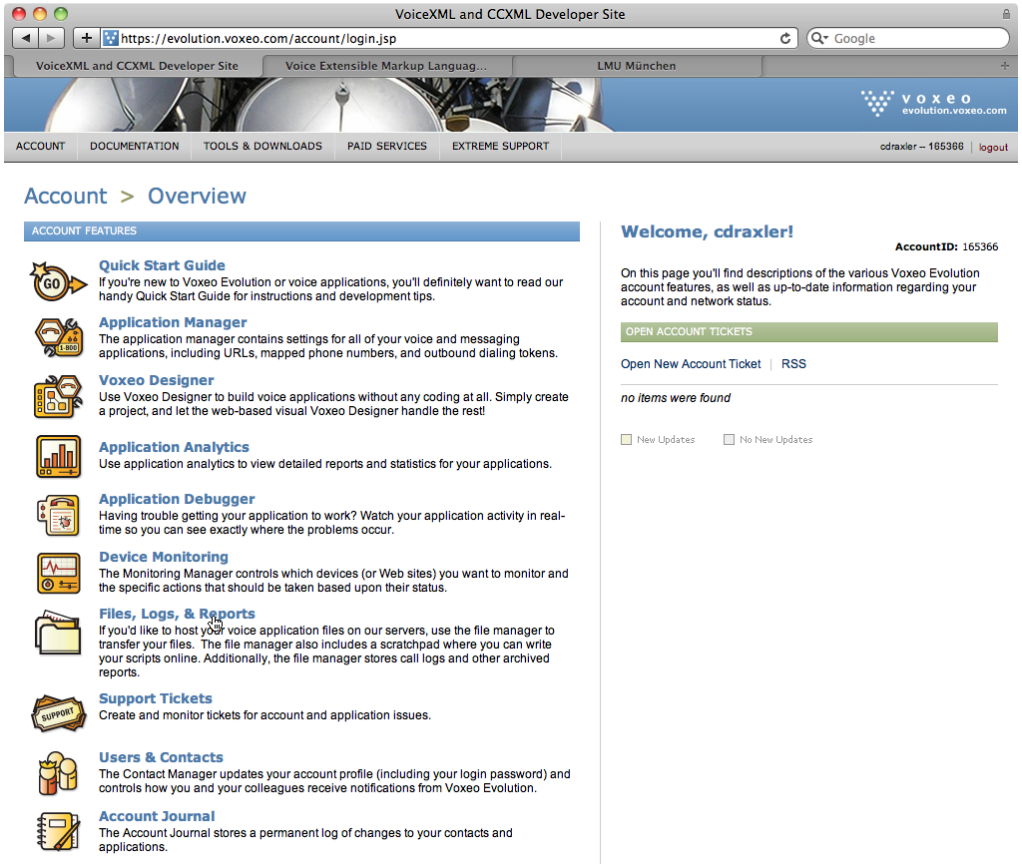
Erste VoiceXML-Applikation

Voraussetzung ist ein Account bei Voxeo. Diesen kann man gratis einrichten.

Vorgehen

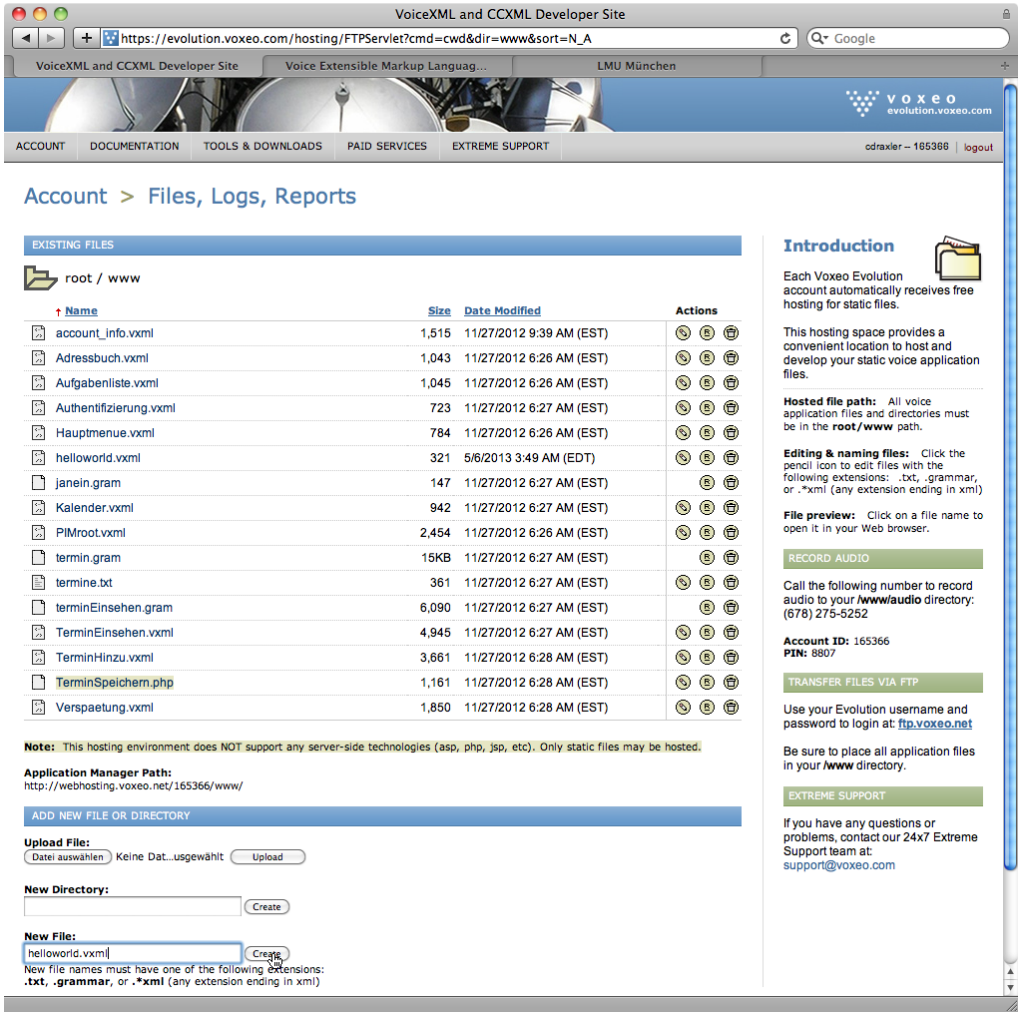
- im Dateieditor VoiceXML-Datei anlegen
- im Application Manager eine neue Dialoganwendung anlegen
- die passenden VoiceXML-Dateien der neuen Dialoganwendung zuordnen
- Dialoganwendung sichern und via Skype testen

VoiceXML-Datei(en) erstellen



Die Option `Files, Logs, & Reports` auswählen

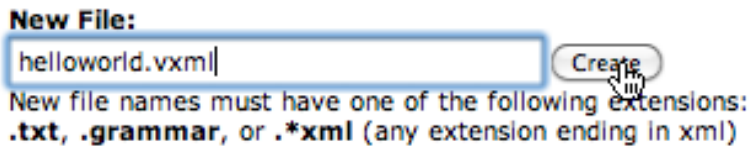
VoiceXML-Datei(en) erstellen



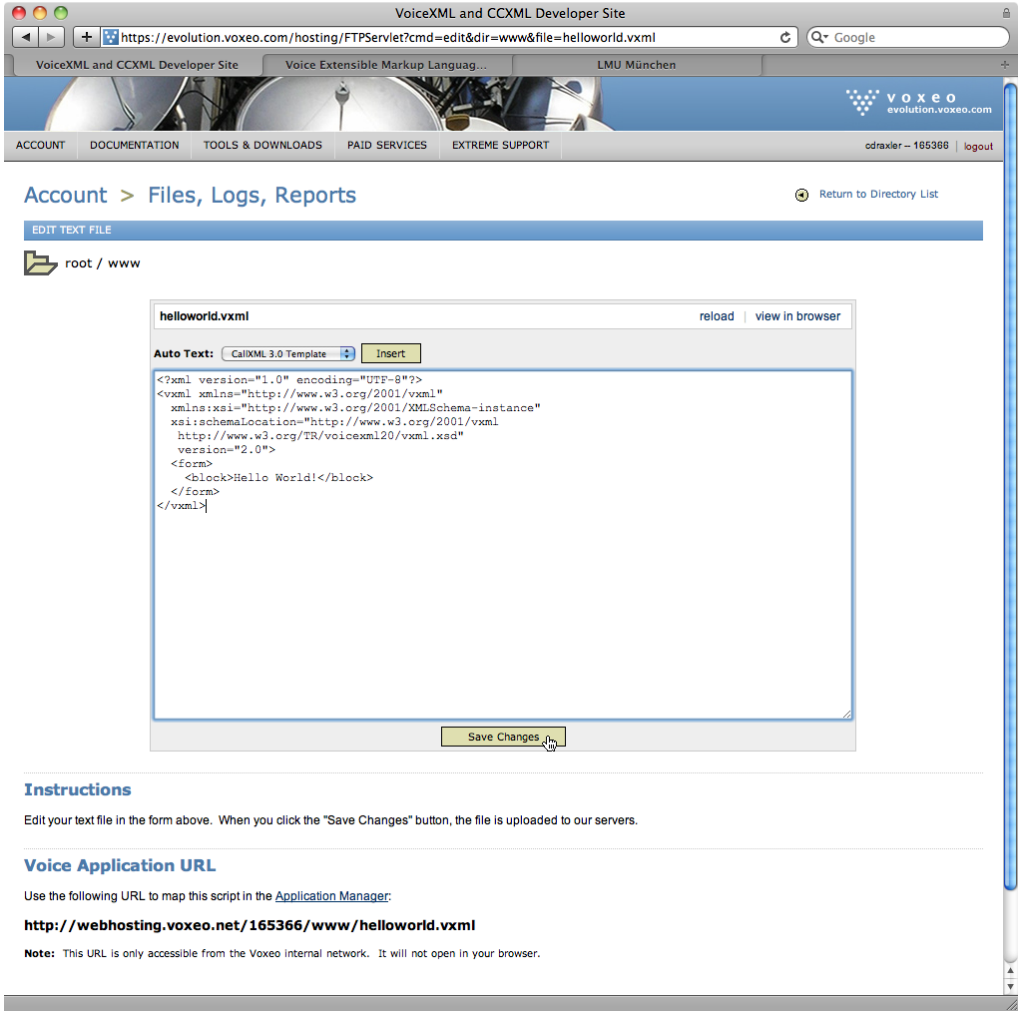
im Verzeichnis `www` befinden sich die bereits angelegten VoiceXML Dateien

VoiceXML-Datei(en) erstellen

in das Textfeld den Dateinamen (z.B. `helloworld.vxml`) eingeben und `Create` anklicken.

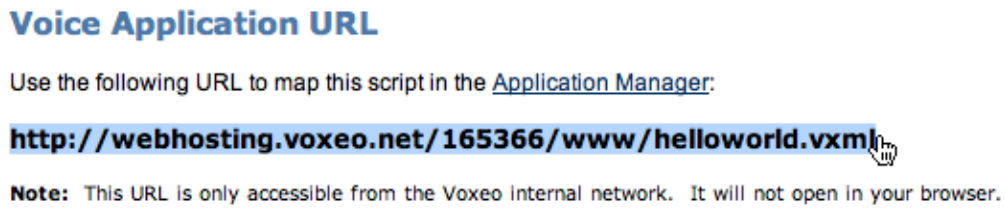


VoiceXML Datei(en) erstellen



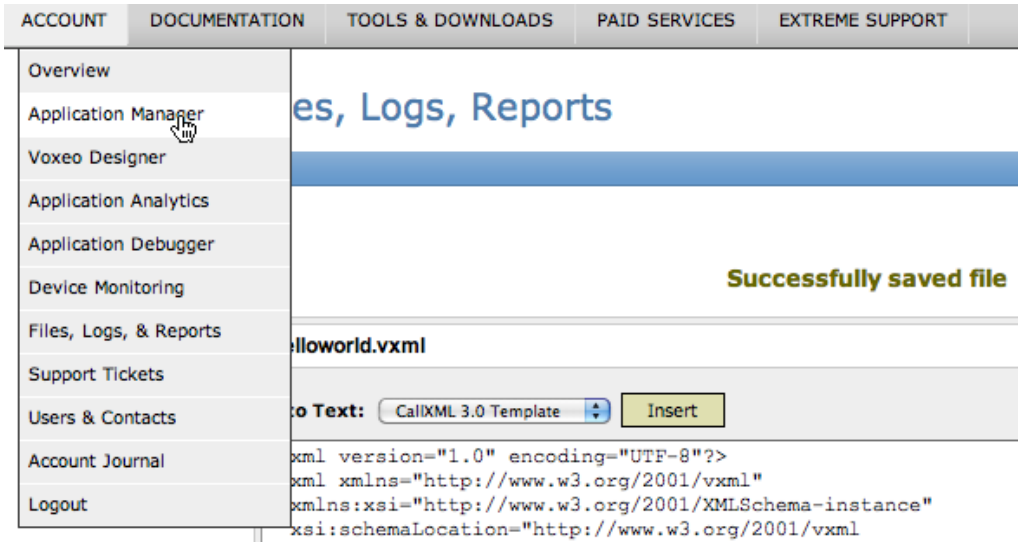
Datei im Editor erstellen und dann sichern. Man kann auch Dateien hochladen bzw. Inhalt per Copy & Paste einfügen.

VoiceXML-Datei(en) erstellen



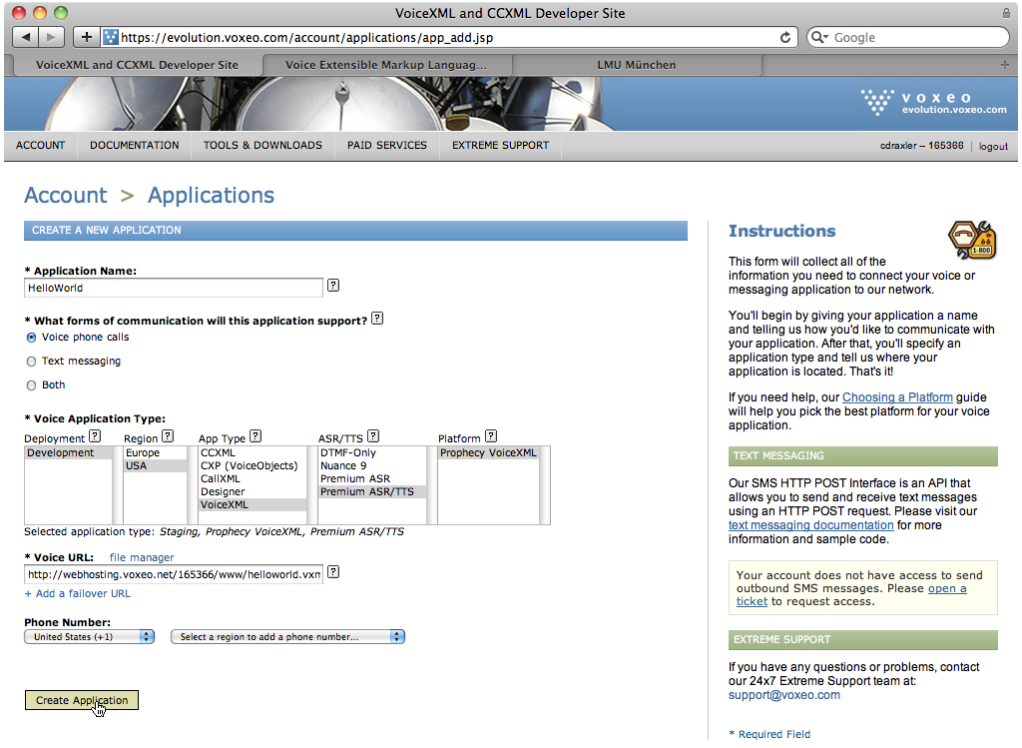
URL der VoiceXML-Datei kopieren oder merken

VoiceXML-Datei(en) erstellen



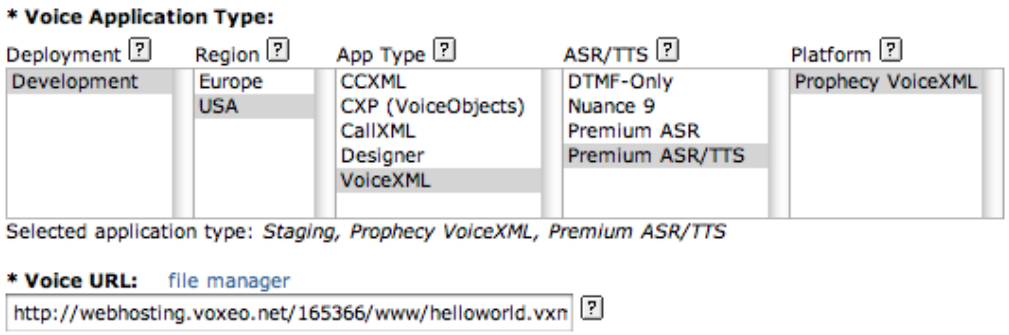
zum `Application Manager` wechseln

Dialoganwendung erstellen



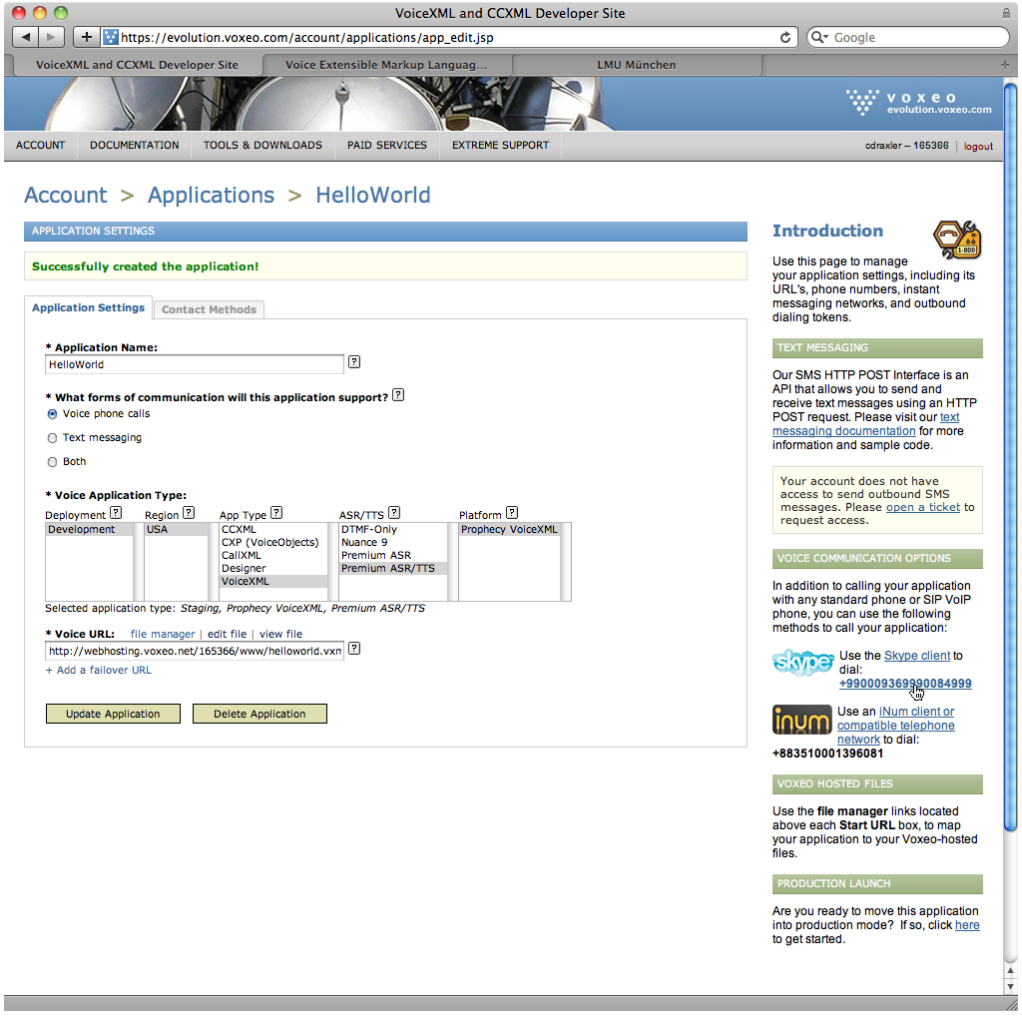
Namen der Anwendung eingeben, und dass es um ein sprachgesteuertes Dialogsystem (*'Voice phone calls'*) geht.

Dialoganwendung erstellen



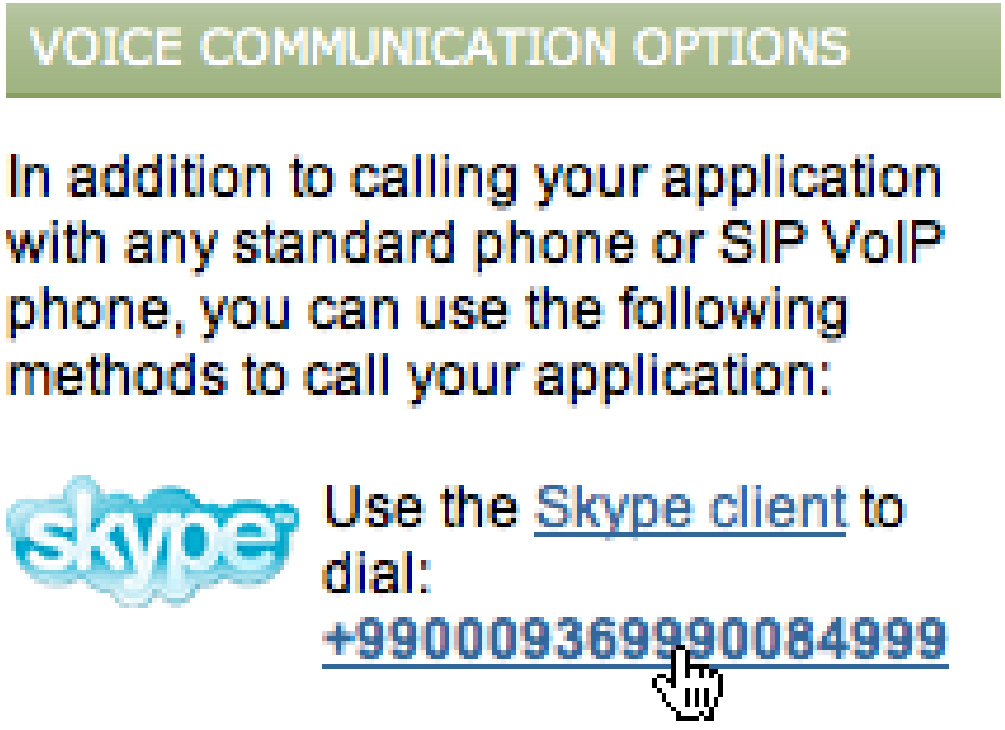
- Art der Bereitstellung (*Development*),
- Region (für Skype unbedingt USA),
- Art der Dialoganwendung (*VoiceXML*),
- Art der Spracherkennung bzw. Synthese (*Premium ASR/TTS*) sowie die
- Platform (*Prophecy VoiceXML*) sowie die URL der VoiceXML-Datei eingeben (alternativ auf "file manager" und dann die entsprechende VoiceXML-Datei klicken),
- mit *Create Application* abschließen.

Dialoganwendung erstellen



Nach dem Speichern erscheinen rechts die Möglichkeiten, die Anwendung anzurufen.

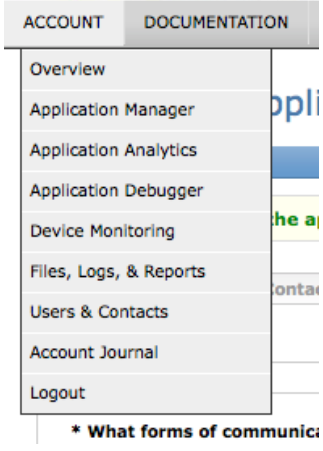
Dialoganwendung erstellen

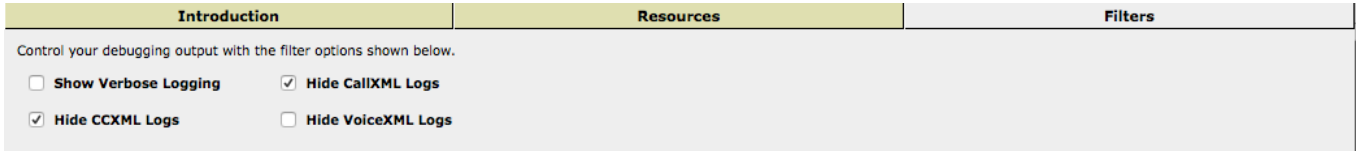


rechts im Fenster erscheint eine Skype-Nummer. Anrufen!

Debugger

Ein Debugger erlaubt das Nachvollziehen von Schritten in VoiceXML-Anwendungen. Im Menü Account kann man einen Debugger des Voxeo-Servers öffnen.





Wenn man sich nur die VoiceXML-Befehle anschaut bleibt der Debugger etwas übersichtlicher.

Transkription der Dialogaufnahmen

die Transkription macht den Inhalt der Dialoge zugänglich

- Wortfrequenz-Liste erstellen
- Verweisstruktur im Dialog untersuchen
- Häsitationen klassifizieren
- Diskursstruktur beschreiben
- ...

Basis dafür ist ein orthographisches Rohtranskript

OCTRA Transkriptionseditor

Zum Transkribieren steht der Editor OCTRA zur Verfügung.

- angepasst an die Erstellung eines Rohtranskripts
- Audiosteuerung per Tastatur

Tab	Abspielen starten bzw. stoppen
S	Setzen von Segmentgrenzen im Signal
RETURN	Öffnen des Segmenteditors im Signal
Umschalt $\rightarrow$	Sichern des Segments und weiter zum nächsten

- Schaltflächen für Marker `<usb>`, `<nib>`, `<p:>`
- Kommentarfeld

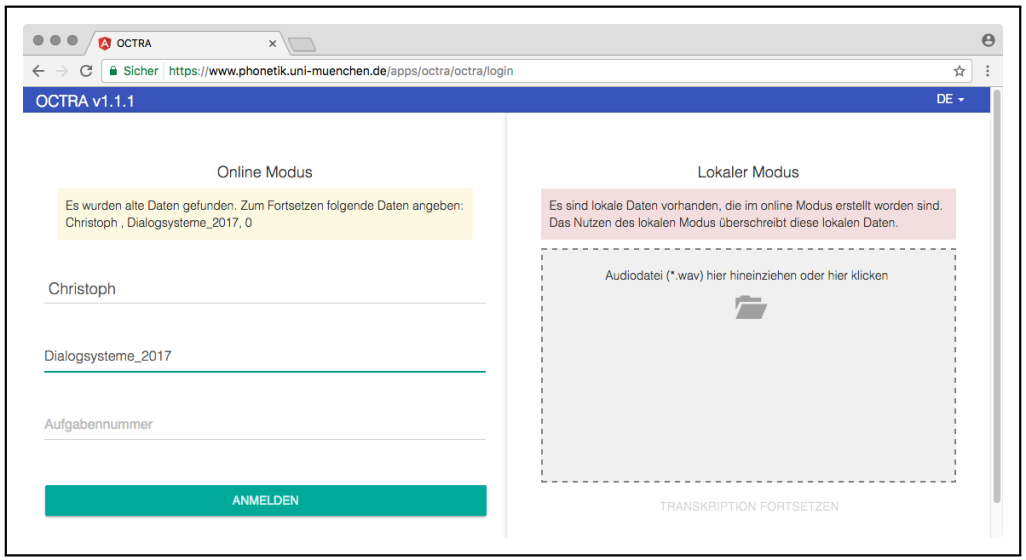
Der Editor ist in Entwicklung. Anregungen und Verbesserungsvorschläge bitte an mich!

URL

Bitte unbedingt in Google öffnen!

`https://www.phonetik.uni-muenchen.de/apps/octra/octra/login`

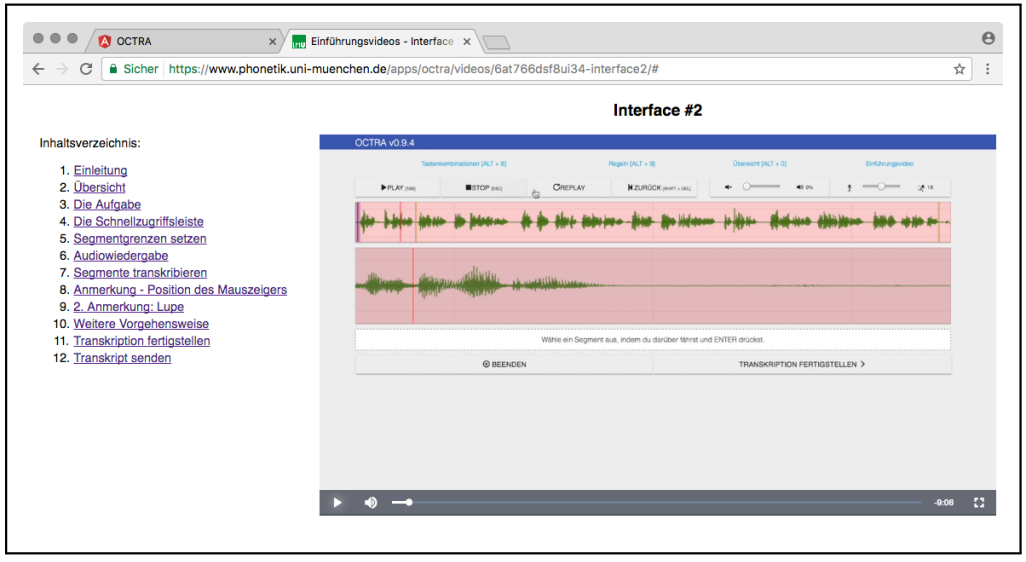
OCTRA Login



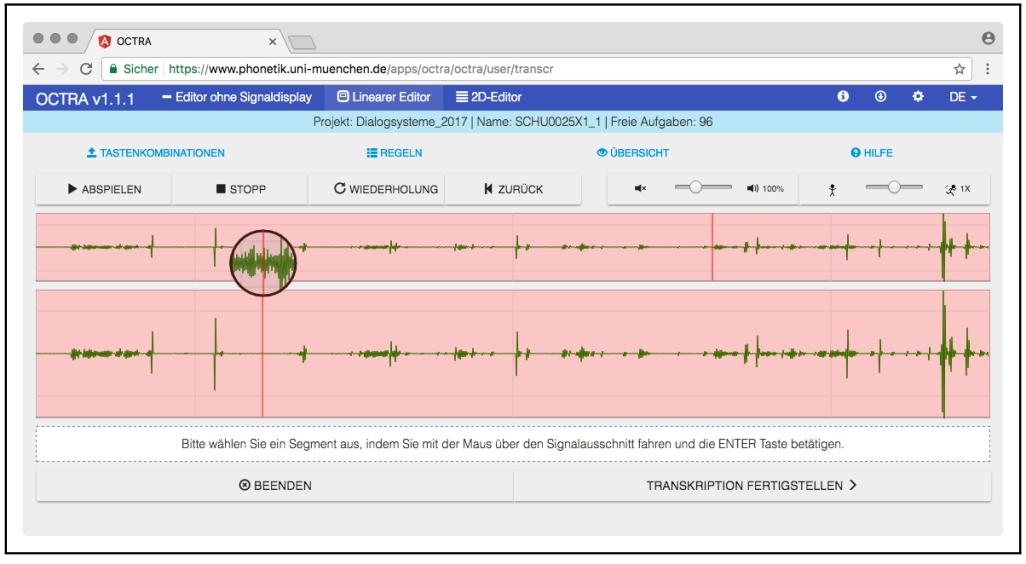
- Benutzernamen eingeben (ohne Leerzeichen!)
- Projekt *Dialogsysteme* auswählen
- OCTRA sucht nach der nächsten zu bearbeitenden Datei

OCTRA Hilfestellung

- online Videos unter 'Hilfe'
- 3 verschiedene Transkriptionseditoren
- Schaltflächen zum Steuern des Players
- Transkriptionsregeln
- Editorfeld mit Schaltflächen



OCTRA Transkription



Hier gezeigt: linearer Editor – es gibt noch einen Editor ohne Signaldisplay und den 2D Editor

- Tastatursteuerung des Players
- Abspielvorgänge werden geloggt, um den Transkriptionsablauf zu analysieren
- Nach dem Sichern erscheint sofort die nächste Audiodatei

Transkriptionskonventionen: Text

1. als Referenz dient der Duden – nur Wörter, die im Duden vorkommen (und ihre flektierten Formen) dürfen verwendet werden
2. verkürzte gesprochene Formen in der langen Form schreiben, z.B. `gibt es` für *gibt's*, `haben wir` für *hammwa*
3. Ziffern und Zahlen auschreiben, z.B. "zwei", "zwo", usw.
4. Wörter durch Leerzeichen trennen
5. Wörter nach ihrer Wortart und *unabhängig* von ihrer Position groß- und kleinschreiben
6. keine Satzzeichen verwenden

Transkriptionskonventionen: Marker

Marker nur setzen, wenn Sie den Sprachfluss stark beinträchtigen

1. nonverbale Sprechergeräusche werden mit <usb> und Leerzeichen vor dem betroffenen Wort markiert, z.B. Lachen, Husten, Schmatzen, lautes Atmen
2. Geräusche, die nicht vom Sprecher stammen, werden mit <nib> und Leerzeichen vor dem betroffenen Wort markiert, z.B. Türgeräusche, Klopfen
3. Sprache eines dritten Sprechers wird mit ** markiert (mit Leerzeichen vom nächsten Wort getrennt), ebenso unverständliche Passagen
4. Versprecher werden soweit möglich transkribiert und mit * am Wortanfang markiert, z.B. am **drit nein vierten April*
5. Markieren Sie an geeigneter Stelle eine Phrasengrenze mit dem senkrechten Strich |

Transkriptionskonventionen: Häsitationen

Häsitationsphänomene wie 'äh', 'öh', 'ähm' und 'mhm' werden unabhängig vom produzierten Vokal mit standardisierten Wörtern transkribiert:

`äh`	wenn kein Nasal zu hören ist
`ähm`	wenn ein Nasal zu hören ist
`mhm`	wenn zwei Nasale zu hören sind
`ne, nee`	bei initialem Nasal (kurzer Vokal bzw. langer Vokal)

Auf diese Weise kann man die Häsitationen später schnell finden, hält aber gleichzeitig das Lexikon kompakt.