Entwicklung von Dialog- und Multimodalen Anwendungen mit Voice-XML und V+X

Cristina Vertan

Inhalt

- Dialoganwendungen 
- Voice - XML
- Dialoggrammatikentwurf
- V+X Erweiterungen
- Aufgaben für das Praktikum

Übersetzungsprobleme

- Engl.: Spoken Language Dialog System (SLDS)

  übersetzt man meistens (inhaltlich nicht ganz korrekt) mit Dialogsystem. Dummerweise ist dieses Wort sogar in der Informatik anderweitig belegt.

- Speech application

  übersetzt man oft mit Sprachsignal-Anwendung, in unserem Kontext aber meistens als Dialoganwendung. Leider kann man im Deutschen nicht einmal zwischen Speech und Language unterscheiden.

Was sind Dialog-Anwendungen ?

- Ein SLDS ist eine Anwendung die eine Mensch-Maschine-Interaktion mit beschränktem Zweck ermöglicht.
- Es gibt 2 Typen von Dialoganwendungen:
  - Transaktions-basierte : ermöglichen die Durchführung einer Transaktion (z.B.einkaufen, verkaufen, oder eine Flug/Zug-Platzreservierung)
  - Informations-Versorgung: Das System gibt eine Information als Antwort auf eine Anfrage (z.B.Zeitplan, Wetterdienste, Fernwartung)

Komponenten einer Dialoganwendung

- Dialogentwurf (script writing / call-flow layout): Man beschreibt, wie die Mensch-Maschine Interaktion schrittweise abläuft)
- Grammatikbeschreibung - spezifiziert, was der Benutzer in jedem Dialogschritt sagen darf (d.h. Was das System erkennen kann)
- Promptentwurf - spezifiziert, was das System sagt, um den Benutzer dazu zu bringen, eine der möglichen Antworten (nach der Dialoggrammatik) zu geben.

Fehlermeldungen in Dialogsystemen

- Da alle Spracherkenner manchmal Fehler produzieren, muß man eine Lösung für solche Eingaben finden, die nicht erkannt werden.

Für unser System zusätzlich:

- Sätze die nicht genau der Dialoggrammatik entsprechen, sollen von einem NL-Modul weiter analysiert werden.

Typen von Dialoganwendungen

Typ	Lösung
• „Touch-tone-only “ - Anwendungen (als Antwort auf Prompts kann der Benutzer nur Tasten drücken)	• Voice-XML
• „Speech.only“ Anwendungen	• Voice XML
• Speech-and Touch-tone-Anwendungen	• Voice XML
• Multimodale Anwendungen	• V+X, SALT

Inhalt

- Dialoganwendungen
- Voice - XML 
- Dialoggrammatikentwurf
- V+X Erweiterungen
- Aufgaben für das Praktikum

Was ist Voice-XML ?

- Voice-XML
  - ist eine XML-basierte Dialog-Markup-Sprache,
  - Sie verbindet WWW mit Telefon und Wirelessgeräten
  - wird (normalerweise) von einem Voice-Browser statt einem GUI-Browser ausgeführt
- Voice-XML-Anwendungen ermöglichen:
  - Als Benutzereingabe Sprachsignale oder Telefontasten
  - Sprachausgabe via Sprachsynthese oder Audiodateien

Voice-XML: Ein einfaches Beispiel

C: Nennen Sie bitte eines der folgenden Themen: Sport, Wetter, Nachrichten

B: Wechselkurse

C: Ich habe leider nicht verstanden (default Antwort)

C. Nennen Sie bitte eines der folgenden Themen: Sport, Wetter, Nachrichten

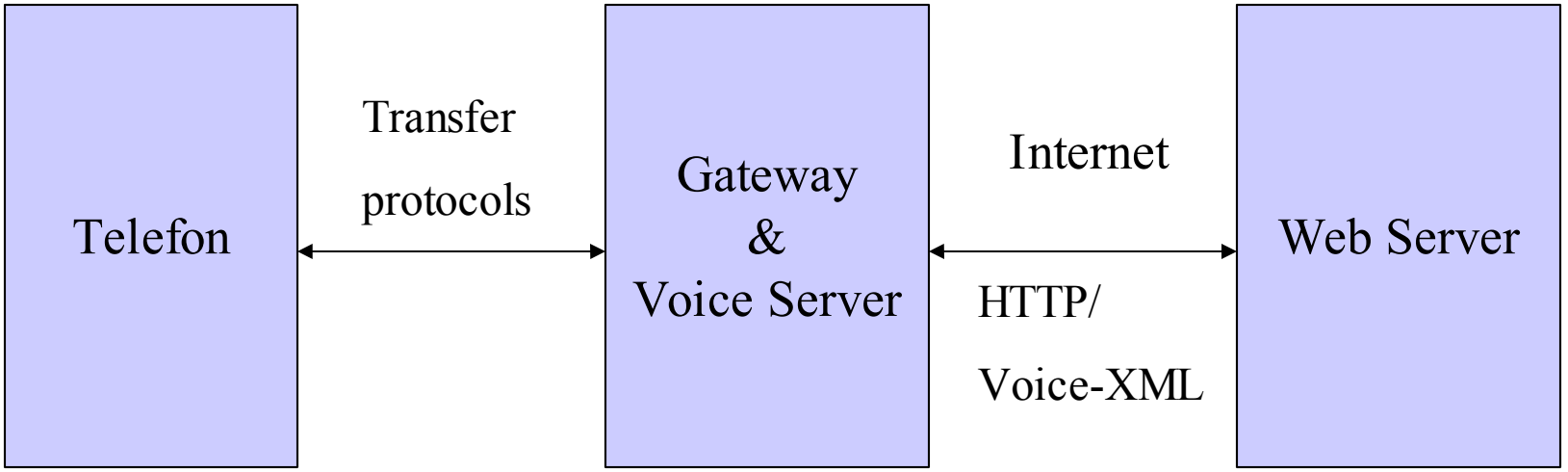
B: Sport

C: geht weiter zu http://www.sport.example/start.vxml

```
<?xml version=1.0”?>
<vxm version”2.0”>
<menu>
<prompt> Nennen Sie bitte eines der
  folgenden Themen: <enumerate/></prompt>
 <choice next=http://www.sport.example/start.vxml>
  Sport </choice>
<choice next=http://www.wetter.example/start.vxml>
  Wetter </choice>
<choice next=http://www.news.example/start.vxml>
  Nachrichten </choice>
 <noinput> Nennen Sie bitte eines der folgenden
  Themen </enumerate>
</noinput>
</menu>
</vxml>
```

Nach R Dale 2003

Voice-XML Architektur



Telefon

Handy

Telefonschnittstelle

Voicebrowser

ASR (Spracherkenner)

TTS (Sprachsynthese)

Touchtone

Audio play/record

VoiceXML Dokumente

Audio-Dateien

DB Schnittstelle

Transaktionsbearbeitung

Nach R Dale 2003

Voice XML Szenario

Ein Benutzer ruft via Telefon oder Handy an,

Die Telefonnummer wird an das voice gateway geleitet,

Das Gateway

- übersetzt die Nummer in ein URL,
- schickt eine HTTP-Anfrage an die spezifizierte URL,

Der Web Server antwortet mit einem Voice-XML-Dokument,

Das Voice-XML-Dokument enthält ein Dialogscript.

VoiceXML Elemente / tags

- <vxml> erstes Element in jedem VXML-Dokument
- <form> leitet einen Dialog (zur Informationsdarstellung oder Datensammlung) ein
- <block> enthält nicht-interaktiven (=executable) Code
- <prompt> markiert Audio-Ausgaben (synthetisiert oder aus file)
- <field> Markiert ein Eingabefeld in einer <form>
- <filled> nennt in einem Feld die Aktion die nach Füllen des einbettenden Feldes startet
- <menu> ein in <form> eingebetteter Dialog zur Auswahl aus Alternativen
- <choice> definiert je ein Menu item
- <grammar> spezifiziert die gültige Grammatik
- <goto> Verbindung zu einem anderen Dialog in demselben oder einem anderem Dokument

Dialogentwurf - Prinzipien -1-

- Das kurzfristige Benutzergedächtnis soll nicht überfordet werden. Deswegen:
  - darf man nicht mehr als 7 Auswahlmöglichkeiten benutzen (optimal 3 oder 4)
  - für jeden Schritt muß eine entsprechende Anweisung definiert werden
  - Die Benutzer sollen den nächsten Schritt überschauen können
  - Globale Information soll nicht benutzt werden, weil sie nicht erinnert werden kann
- Für multimodale Systeme sind diese Prinzipien nicht so streng. Trotzdem muß man immer daran denken dass die Kommunikationszeit so kurz wie möglich sein soll, und dass die Display-Möglichkeiten begrenzt sind.

Dialogentwurf-Prinzipien -2-

- Dialogverhalten ist nicht immer fehlerlos. Deswegen
  - muss der Benutzer wissen, was er sagen darf,
  - sollen die Prompts eine vernünftige Anzahl an Auswahl geben
  - Für Dialogabweichungen müssen Event-Handler geschrieben werden
- Außerdem muss man immer daran denken, dass der Benutzer häufig versucht mit dem System wie mit einer Person zu reden.

Voice XML-Dokumente

- Ein Voice XML Dokument ist einem endlichen Automat (finite state machine) vergleichbar
- Der Benutzer ist immer in einem bestimmten Dialog-Zustand
- Jeder Zustand ist Endzustand oder enthält mindestens eine Transition zu einem weiteren Zustand
- Transitionen sind mit den URI (Addressen) des nächsten Zustands spezifiziert
- Die Ausführung ist beendet, wenn:
  - Ein Dialog keinen Nachfolger hat, oder
  - Es ein tag gibt, das explizit den Dialog beendet

Voice-XML- Dialoge -1-

- Es gibt 2 Typen von Dialogelementen:
  - Forms
  - Menus
- Forms sammeln Werte (Sätze/Wörter/Tasteneingabe), die eine Variable (Feld) binden sollen (sie sind ähnlich mit Textfeldern in GUIs)
- Event-Handler müssen für solche Werte (Sätze/Wörter/ Tasten) geschrieben werden, die im Sinne der Grammatik nicht gültig sind,
- Aktionen werden durchgeführt wenn die nötigen Variablen (Felder) nach der Filled-Bedingung gebunden sind

Form - Template

```
<form id = Identifier>
  <block> Message </block>
  <field name =VariableName>
      <prompt>Question</prompt>
      <grammar src=URI type=MediaType/>
      <catch event=eventType>HandlerMessage </catch>
     <filled>Actions </filled>
  </field>
</form>
```

Voice-XML- Dialoge -2-

- Menus geben dem Benutzer eine Auswahlmöglichkeit
- Die Dialogtransition ist auf eine Auswahl beschränkt
- Menu ist eigentlich ein “Abkürzung” für eine Form mit nur einem Feld
- Menus sind für Sprachanwendungen noch einfacher zu beherrschen, andererseits muss man bedenken, dass sie den Benutzer relativ stark einschränken.

Menu - Template

```
<menu id=Identifier>
    <prompt> Question <enumerate/> <prompt>
     <choice next=URI-1> Phrase-1 </choice>
     <choice next=URI-2> Phrase-2 </choice>
     <choice next=URI-3> Phrase-3 </choice>
<noinput> Message <enumerate/></noinput>
</menu>
```

Dialog-Transitionen

- Werden via URI spezifiziert
- Ein URI definiert den nächsten Dialog im aktuellen oder einem anderen Dokument
- spezifiziert ein URI kein Dokument, gilt als "default" das aktuelle Dokument
- spezifiziert ein URI keinen Dialog, gilt als "default" der erste Dialog
- Transitionen werden angefordert mit den Tags:
  - <choice next=URI>
  - <goto next URI>
  - <link next URI>

Catch und Help Element

Das <help> -Element ist eine Abkürzung für

<catch event="help"> ....</catch>

z.B.

<help>

Bitte sagen Sie Motor, Räder oder Vergaser

</help>

Erweiterungen zu V+X

- Einfache Erweiterung:

  <prompt> , <grammar> and <form> Elemente sind erweitert durch

  – <sequence-of> = die Aktionen sollen sequentiell duchgeführt werden
  – <one-of> = nur eine Aktion darf durchgeführt werden
  – <all-of> = alle Aktionen müssen duchgeführt werden.

Erweiterungen zu V+X -Beispiel -1-

```
<prompt>
  <all-of>
    <media type="voice">
          Willkomen beim System ...
    </media>
    <media type ="display-text">
          Willkomen beim System ...
    </media>
  </all-of>
</prompt>
```

Erweiterungen zu V+X -Beispiel -2-

```
<choice next =“www.train.xml”>
  <one-of>
    <mode=“voice” src=“train-voice.grammar”/>
    <mode=“keyboard” src=“train-text.grammar”/>
</one-of>
</choice>
```

Inhalt

- Dialoganwendungen
- Voice - XML
- V+X Erweiterungen
- Dialoggrammatikentwurf 
- Aufgaben für das Praktikum

Was ist eine Dialoggrammatik?

- Eine Dialoggrammatik enthält eine Regelmenge, die eine Sammlung von Wörtern und Sätzen spezifiziert, die von dem Spracherkenner erkennbar sind
- Es gibt verschiedene Formalismen, um eine Dialoggrammatik zu spezifizieren, wie:
  - Nuance Grammar Specification Language (GSL)
  - Speech Recognition Grammar Specification (SRGS)
  - usw.

Einbettung einer Grammatik -inline-

```
<?xml version="1.0"?>
<vxml version="2.0">
<form id="start">
    <field name ="Schaden">
    <prompt> Wo haben sie ein Problem?
    </prompt>
        <grammar type="application/x-gsl">
        ?(Mein Problem ist am) [Motor Fahrwerk Vergaser]
        </grammar>
    <filled>
        <prompt> Ich habe verstanden : sie haben ein Problem
    am <value expr="Schaden"/>
        </prompt>
    </filled>
    </field>
</form>
</vxml>
```

GSL Syntax

(A B C .....Z) bedeutet A $\vee$ B $\vee$ C $\vee$... $\vee$ Z

[A B C.....Z] bedeutet A $\wedge$ B $\wedge$ C $\wedge$... $\wedge$ Z

?A bedeutet A ist optional

+A bedeutet eine oder mehrere Wiederholungen von A

*A bedeute Zero oder mehrere Wiederholungen von A

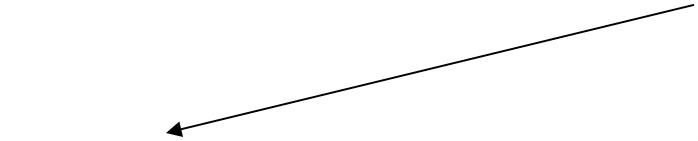
GSL Grammatik
Beispiel für Synonyme

```
<prompt> Welche Farbe hat ihr Wagen ? </prompt>
  <grammar type=application/x-gsl”>
    <![CDATA[
      ?(Meine Wagen hat eine)
        [ blaue {<colour “blau”>}
          türkise {<colour “blau”>}
          grüne {<colour “grün”>}
         .....]
    ?(Farbe) ]]>
</grammar>
<filled>
   <prompt> Ich verstehe: Ihr Auto ist <value expr=“colour”/></prompt>
</filled>
```

Trennung von Grammatik und Dialogskript

- Man schreibt die Grammatik in eine Datei (.gram) entsprechend dem gewählten Formalismus
- im Dialogskript steht dann:

<grammar src="colour.gram#Farbe" type="application/x-gsl"/>



```
Farbe

(?(Mein Wagen hat eine)
        [ blaue {<colour "blau">}
          türkise{<colour "blau">}
          grüne {<colour "grün">}
         .....]
    ?(Farbe) )
```

Komplexes Beispiel

```
FarbWahl
   (?Mein Wagen hat Farbe:x) {<colour $x>}
Haben
   [(Mein Wagen ist) (Die Farbe meines Wagens ist)]
Farbe [ rot {return("rot")}
        blau {return ("blau"}
        grün {return("grün")}
      ]
```

Inhalt

- Dialoganwendungen
- Voice - XML
- V+X Erweiterungen
- Dialoggrammatikentwurf
- Aufgaben für das Praktikum

Aufgaben für das Praktikum -1-

- Erstellen eines Korpus für Fragen und mögliche System-Antworten
- Erstellen eines Dialogentwurfs für das Korpus
- Trennung des Korpus in:
  - Sätze die mit einer voiceXML-Grammatik modelliert werden können
  - Komplexere Fälle die zum NL -Modul geschickt werden müssen

Bis hierhin anstelle der ausgefallenen Sitzung

- Auswahl der nötigen VoiceXML- Tags (das komplette Handbuch steht unter http://www.w3.org/TR/2001/WD-voicexml20-20011023")

Weitere Aufgaben für das Praktikum -2-

- Erweiterung des Tag-sets mit anderen nötigen Elementen: Einbettung von Bildern oder Java-Modulen
- Implementation eines Parsers für die gegebenen Tags
- Modellierung des Dialogs mit den ausgewählten Tags

Dies alles zunächst nur für eine Menge von 10 Sätzen