Schupperstudiums-Projekt

„Mensch-Maschine-Kommunikation mit gesprochener Sprache

Einführung in VoiceXML

Cristina Vertan

vertan@informatik.uni-hamburg.de

Was lernen wir hier?

- Wie funktioniert VoiceXML?
- Grundelemente in VoiceXML
- Aufbau eines modulares Programmes (Subdialoge)

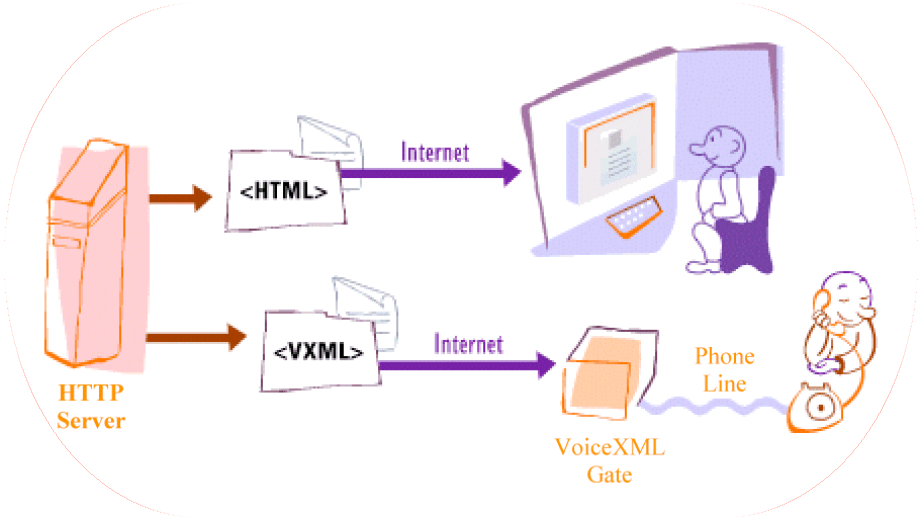
  z.B.: Ein Benutzer, der einen ausgebuchten Kinofilm
  sehen möchte, gelangt nicht mehr zur Buchung. Er wird aufgefordert,
  einen anderen Film zu buchen o.ä.

- Wie gebe ich dem Benutzer eine bestimmte Freiheit in seiner Aussage? (Grammatiken)

  z.B.: “Wohin möchten Sie fliegen?”
  Prag
  Ich möchte nach Prag.
  Ich möchte nach Prag fliegen.

Was ist Voice-XML?

- Voice-XML
  - Ist eine XML-basierte Dialog-Markup-Sprache
  - Sie verbindet WWW mit Telefon und Wireless-Geräten



  - Wird (normalerweise) von einem Voice-Browser statt einem GUI-Browser, z.B. Explorer, ausgeführt
- Voice-XML-Anwendungen ermöglichen:
  - Benutzereingabe durch Sprachsignale oder Telefontasten
  - Sprachausgabe durch Sprachsynthese oder Audiodateien

Voice-XML: ein einfaches Beispiel

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

B: Wechselkurse

C: Ich habe Sie leider nicht verstanden *(Default-Antwort)*

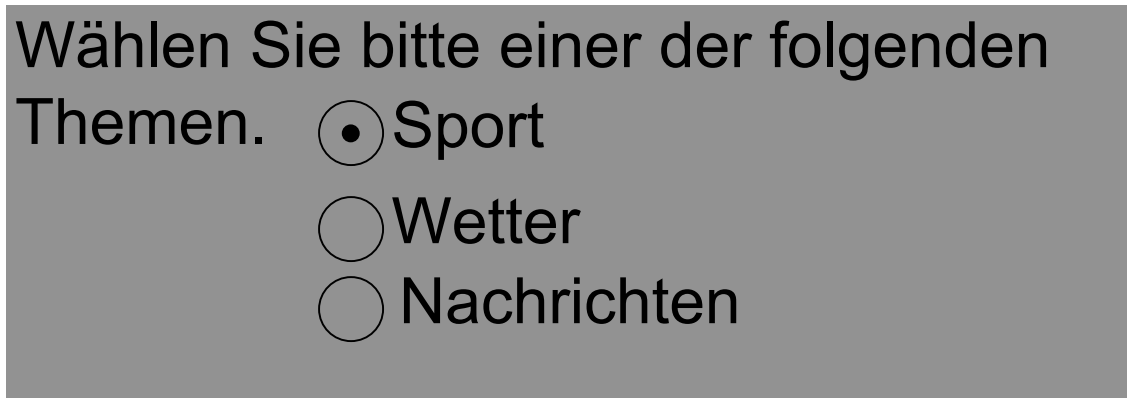
C: Wählen Sie bitte eines der folgenden Themen: Sport, wetter, Nachrichten

B: Sport

C: *führt den Dialog, der in sport.vxml beschrieben ist, durch*

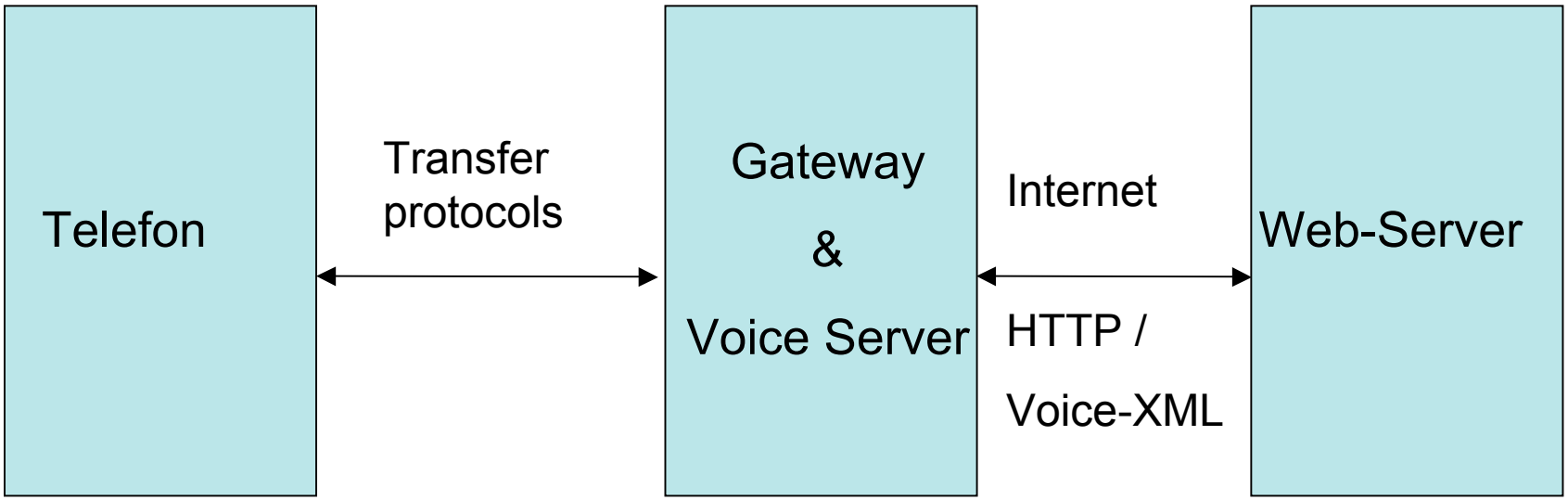
```
<?xml version=„1.0“?>
<!DOCTYPE ...>
<vxml version=„2.0“ xmnls=„...“>
<menu>
 <prompt> Wählen Sie bitte eines der
    folgenden
    Themen:<enumerate/></prompt>
<choice next=„http://...“>Sport</choice>
<choice
    next=„http://...“>Wetter</choice>
<choice
    next=„http://...“>Nachrichten</choic
    e>
<noinput> Wählen Sie bitte eines der
    folgenden Themen:<enumerate/>
</menu>
</vxml>
```

Analogie mit GUI



- Vorteile GUI: Die Eingabe des Benutzers wird eindeutig vom System erkannt
- Nachteile GUI:
- Flexiblere Eingaben, wie „Sagen Sie mir etwas über Sport“, sind nicht möglich
- Man ist abhängig von der Existenz eines Bildschirms

Wie arbeitet VoiceXML?



Telefon

Handy

Telefonschnittstelle

Voice-Browser

Spracherkenner

Sprachsytheser

Touchtone

Audioplay record

VoiceXML-Dokumente

Audio-Dateien

DB-Schnittstelle

Transaktionsbearbeitung

Was muss ich kennen um einen einfaches Dialogsystem aufzubauen?

- Spracherkenner und Sprachsyntheser sind vorhanden. Kostenslose Versionen sind nicht extrem performant, aber reichen für unser Projekt aus.
- Der VoiceXML-Interpreter ist vorhanden
- DTD-Spezifikation und alles was, mit XML/VXML-Versionen zu tun hat, kann man aus der Muster-Datei kopieren.
- Minimale Kenntnisse über Markup-Syntax
- Grund-Elemente aus der VoiceXML-Syntax

Auf was muss ich bei der Syntax achten?

- Ein Tag wird duch die „<” bzw. „>“-Klammer spezifiziert.
- Die Tags kann man nicht beliebig erweitern. Auf der VoiceXML-Einführungs-Webseite findet ihr alle vorhandenen Tags
- Die meisten Tags sind paarweise vorhanden:
  - <tag> ein öffnender Tag
  - </tag> ein schliessender Tag
  - dazwischen steht die für diesen Tag wichtige Information
- Falls ein Tag keine Information enthält, schreibt man
  - <tag/>
- Ein Tag kann Attribute, also Beschreibungen, enthalten, dann sieht es so aus:
  - <tag Attribut1=„wert1“ Attribut2=„wert2“ ...>
  - Meistens sind Attribute optional, müssen also nicht zwingend angegeben sein
  - welche Attribute für welchen Tag möglich sind, ist festgelegt

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 Datei)
- <field> markiert ein Eingabefeld in einem <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 anderen Dokument

Beispiel

Versionsangaben zu VXML

Kommentar

block erwartet keine Aktion -
hier ist die Begrüßung damit realisiert

enumerate bewirkt, dass die
nachfolgenden choices aufgezählt werden

field wird über den Namen angesprochen

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<vxml xmlns="http://www.w3org/2001/vxml"
 version="2.0">
<!--Hello World Program -->
<form>
 <block>
  <prompt> Guten Tag!
  </prompt>
 </block>
 <field name=”abteilung”>
  <prompt>Mit welcher Abteilung darf ich Sie
          verbinden?<enumerate/>
  </prompt>
  <choice next=”abtA”>Abteilung A</choice>
  <choice next=”abtB” >Abteilung B</choice>
  <prompt>Ich verbinde Sie mit <value expr=”abteilung”/>.
          Auf Wiederhören.
  </prompt>
 </field>
</form>
</vxml>
```

Dialogentwurf-Prinzipien -1-

- Das Kurzzeitgedächtnis soll nicht überfordert 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 Bildschirm-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 Zahl von Auswahlmöglichkeiten geben
  - Für Dialogabweichungen müssen Event-Handler geschrieben werden (d.h. was passiert, wenn der Benutzer etwas sagt, was das System nicht bearbeiten kann)
- Außerdem muss man immer daran denken, dass der Benutzer häufig versucht, mit dem System zu reden wie mit einer Person.

VoiceXML-Dialoge -1-

- Es gibt 2 Typen von Dialogelementen:
  - form
  - menu
- Forms sammeln Werte (Sätze/Wörter/Tasteneingaben), die eine Variable (Feld) binden sollen (sie sind ähnlich wie Textfelder 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 spezifiziert sind

VoiceXML-Dialoge -2-

- Menüs geben dem Benutzer eine Auswahlmöglichkeit
- Die Dialog-Transition (Umschaltung in einen anderen Dialog) ist auf eine Auswahl beschränkt
- menu ist eine “Abkürzung” für eine Form mit nur einem Feld
- Menüs sind für Sprachanwendungen noch einfacher zu beherrschen, andererseits muss man bedenken, dass sie den Benutzer relativ stark einschränken.

Dialog-Transitionen

- Werden via URI (Uniform Ressource Identifier) spezifiziert,
- Ein URI definiert den nächsten Dialog im aktuellen oder einem anderen Dokument,
- Spezifiziert ein URI kein Dokument, gilt das aktuelle Dokument als “default”
- Spezifiziert ein URI keinen Dialog, gilt der erste Dialog als “default”
- Transitionen werden angegeben mit den Tags:
  - <choice next=URI>
  - <goto next URI>
  - <link next URI>

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>
```

<enumerate/> bewirkt eine Wiederholung der Frage und der Auswahl

Catch und Help Element

Das <help> -Element ist eine Abkürzung für

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

z.B.

<help>

Please say Student Accommodation, Courses or Sport

</help>

Implizite Verifizierung den Benutzereingaben

```
<form id=„courses_details“>
<block>Welcome to Courses Details</block>
<field name=„course_type“>
    <prompt> Please specify the type of course </prompt>
     <grammar> Proseminar | Seminar | Project | Praktika </grammar>
</field>
<field name=„confirm“ type ,“boolean“>
<prompt> List courses of type <value expr=„course_type“/> </prompt>
<filled>
<if cond=“confirm“>
<prompt> You can register to following <value expr=„course_type“/> </prompt>
<else /> try once more
<clear namelist=„course_type confirm“/>
</if> </filled></field></form>
```

Implizite Antwort „yes/no“ and variants

Löscht die Werte von Feldern *course_type* and *confirm*

Variables set to undefined

Subdialog

- Ein Subdialog ist eine neue Interaktion, die innerhalb eines Dialoges erzeugt wurde
- Alle Werte der Variablen und der aktivierten Grammatiken werden gespeichert. Für den neuen Subdialog ist damit eine neue Umgebung initialisiert.
- Wenn der Subdialog beendet ist, wird der Hauptdialog dort fortgesetzt, wo er unterbrochen wurde. Die Variablen und Grammatiken des Hauptdialoges sind wieder aktiv.
- Ein Subdialog wird durch `<subdialog>` initialisiert.
- Die Rückkehr zum Hauptdialog ist mit `<return>` markiert

VoiceXML -Beispiel für Subdialoge

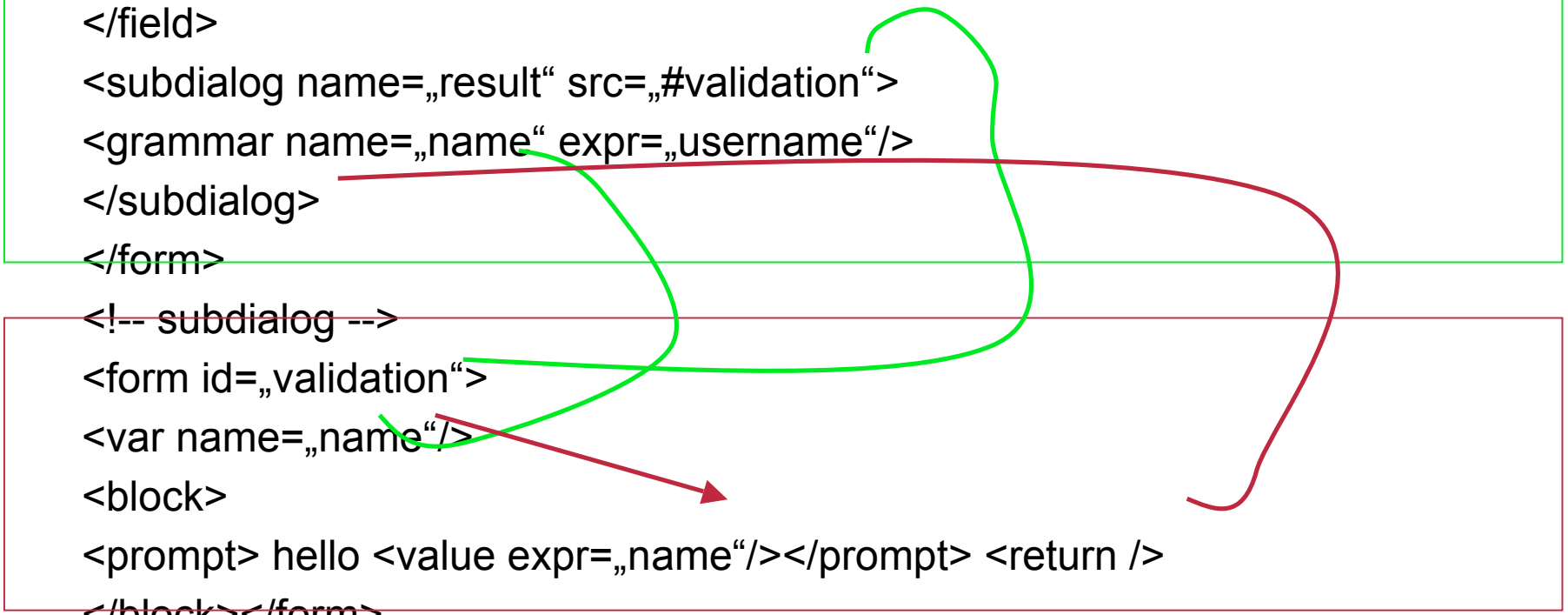
```
<?xml version=„1.0“>
<vxml version=„2.0“>
<menu id=„mainmenu“>
  <prompt> Welcome to the Student System Main Menu. The
    system provides you with information about student
    accommodation, courses and administrative issues. To begin
    say one of the following:
  </prompt>
  <enumerate/>
  <choice next=„accommodation.vxml“> accommodation </choice>
  <choice next=„courses.vxml“> courses </choice>
  <choice next=„administration.vxml“> administration </choice>
  </menu>
</vxml>
```

Variablentransfer von einem Subdialog in einen Hauptdialog

```
<block> Welcome to the Student Information System </block>
<subdialog name=„result“ src=„#validation“> </subdialog>
<block>
<prompt>
Hello <value expr=result.username“ />
</prompt>
</block></form>
<!-- subdialog -->
<form id=„validation“>
<field name=„username“>
<grammar> Schröder | Schmidt | Schulz | Guest </grammar>
<prompt> Please say your username </prompt>
</field>
<filled><return namelist=„username“ /> </filled>
</form>
```

Variablentransfer von einem Hauptdialog zu einem Subdialog

```
<form id=„main_menu“>
<block> Welcome to the Student Information System </block>
<field name=„username“>
<grammar> Schröder | Schmidt | Schulz | guest </grammar>
<prompt> Please say your user name </prompt>
</field>
<subdialog name=„result“ src=„#validation“>
<grammar name=„name“ expr=„username“/>
</subdialog>
</form>
<!-- subdialog -->
<form id=„validation“>
<var name=„name“/>
<block>
<prompt> hello <value expr=„name“/></prompt> <return />
</block></form>
```



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 oder B oder C oder ... oder Z
- (A B C.....Z) bedeutet A und B und C und... und Z
- ?A bedeutet A ist optional
- +A bedeutet eine oder mehrere Wiederholungen von A
- *A bedeutet keine oder mehrere Wiederholungen von A

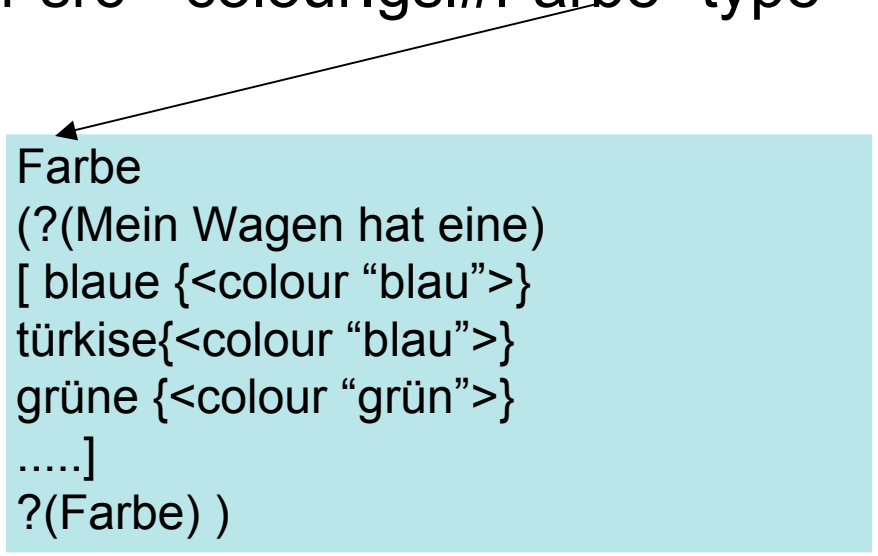
GSL Grammatik: Beispiel für Synonyme

```
<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> Also: Ihr Auto ist <value expr=“colour”/></prompt>
</filled>
```

Trennung von Grammatik und VoiceXML-Datei

- Man schreibt die Grammatik in eine Datei (.gsl)
- Entsprechend des gewählten Formalismus,
- Im Dialogskript steht dann:

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



Viel Spaß bei der Anwendung !