Grundlagen des Programmierens in VoiceXML

Donnerstage 27. 05. und 03. 06. 2004

5. Grundlagen des Programmierens in VoiceXML

Richtziel:

Erlernen möglicher Verwendung von Markup-Programmiersprachen für Modellierung der natürlichsprachlichen Mensch-Computer Interaktion.

Schwerpunkte der Unterricht:

Grundlagen der Computerlinguistik

Modellierung und Simulation sprachlicher Prozesse auf dem Computer

Grundlagen der Mensch-Computer Interaktion

Natürlichsprachliche Kommunikation, Discourse und Dialog

Funktionale Eigenschaften und Blockstruktur des Dialogsystems

Einführung in die Problematik der Sprachein- und –ausgabe

Prinzipien der Dialogsteuerung, symbolische Darstellungen der Dialogführung

Rolle der Dialogmodelle und Dialogmodellierung

5. Grundlagen des Programmierens in VoiceXML

Eigenschaften von Markup Programmiersprachen

HTML, AIML, XML und VoiceXML Sprachen

Aufbau (Struktur) und Elemente der Sprache VoiceXML

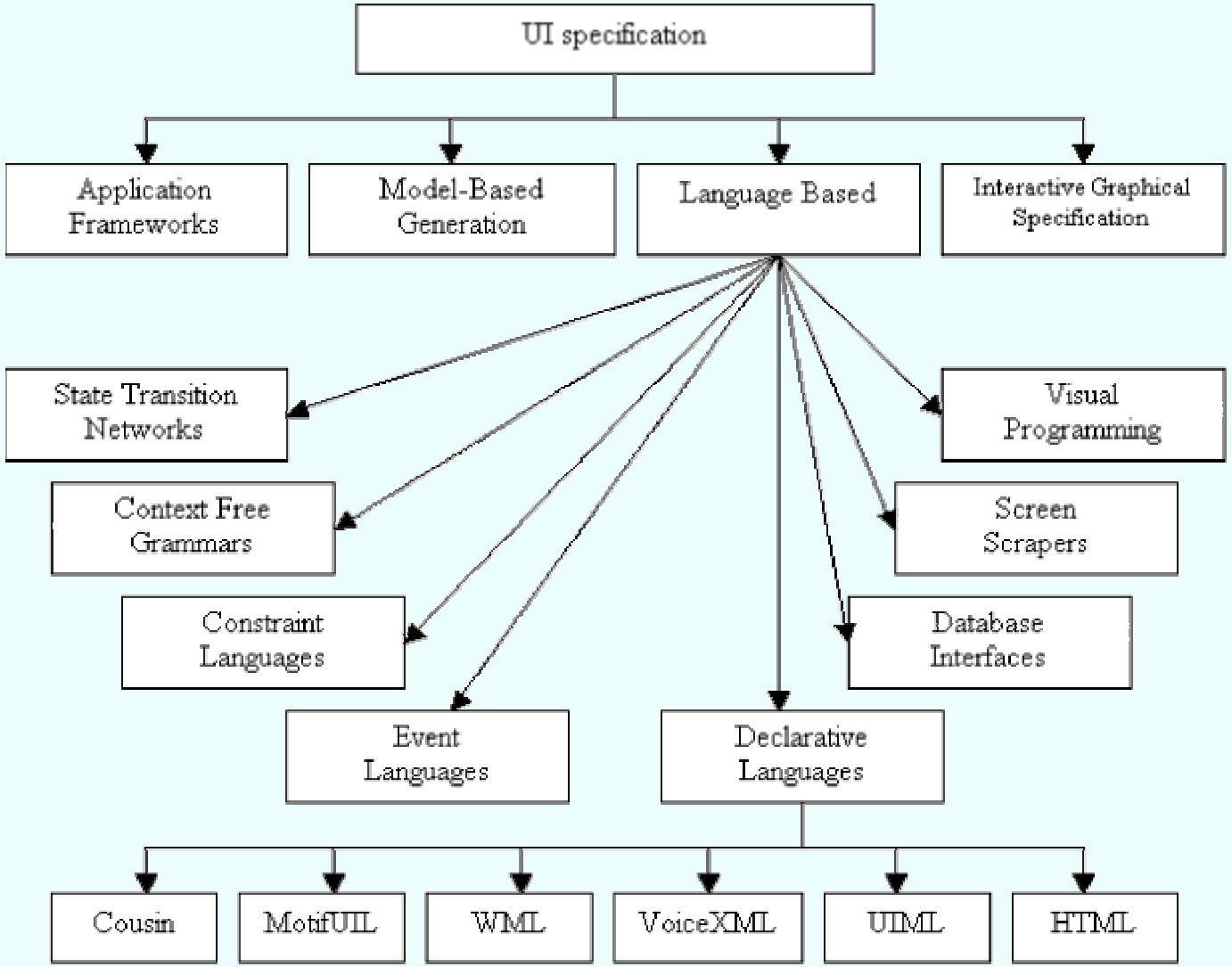
Merkmale von VoiceXML Programmen

Beispiele

Erstellung von Programmen in VoiceXML unterschiedlicher Komplexität

Formale Sprachen für Benutzungsschnittstellen

- deklarative (declarative user interfaces) – mark-up languages
- imperative (imperative user interfaces) – Java/JFC, C++/MFC, C/Motif, ...
- ereignisgesteuerte Sprachen, Übergangsnetzwerke, ...


UI specification
Application Frameworks
Model-Based Generation
Language Based
Interactive Graphical Specification
State Transition Networks
Context Free Grammars
Constraint Languages
Event Languages
Visual Programming
Screen Scrapers
Database Interfaces
Declarative Languages
Cousin
MotifUIL
WML
VoiceXML
UIML
HTML

Was ist XML-Sprache ?

– eXtensible Markup Language

– erweiterbare Auszeichnungssprache

Definition:

XML ist ein Satz von Regeln zur Definition von semantischen Tags, die ein Dokument in Teile zerlegen und die verschiedenen Teile des Dokuments beschreiben. Es ist eine Meta-Auszeichnungssprache, die eine Syntax definiert, mit der andere domänenspezifische, semantische, strukturierte Auszeichnungssprachen definiert werden können.

Charakteristische Merkmale der XML-Sprache

- die benötigten Tags können nach Bedarf definiert werden,
- die Tags müssen gewisse allgemeine Regeln erfüllen,
- die erstellten Tags können in einer Dokumententyp-Definition (DTD) dokumentiert werden,
- XML definiert eine Metasyntax,
- XML beschreibt Struktur und Semantik, keine Formatierung,
- jede XML-basierte Auszeichnungssprache wird als XML-Anwendung bezeichnet.

Prinzipien der XML-Programmierung –

schließen von Programmstrukturen zwischen wahlbare Marken (“tags”):

```
<student type=”PhD”>
        <age>27</age>
        <name>
                <first>Roland</first>
                <family>Müller</family>
        </name>
</student>
```

```
<?xml version="1.0"?>
<?xml encoding="ISO-8859-1"?>
<Firma>
  <Name_Firma> EuroSoftware GmbH </Name_Firma>
  <Abteilung>
    <Name_Abteilung> Geschäftsführung </Name_Abteilung>
      <Person>
        <Nachname> Müller</Nachname>
        <Vorname> Roland </Vorname>
        <Position> Geschäftsführer </Position>
        <Strasse> Schumannstr.12 </Strasse>
        <Ort> 93049 Regensburg </Ort>
        <Telefonnr> 0941/911933 </Telefonnr>
       </Person>        . . .
```

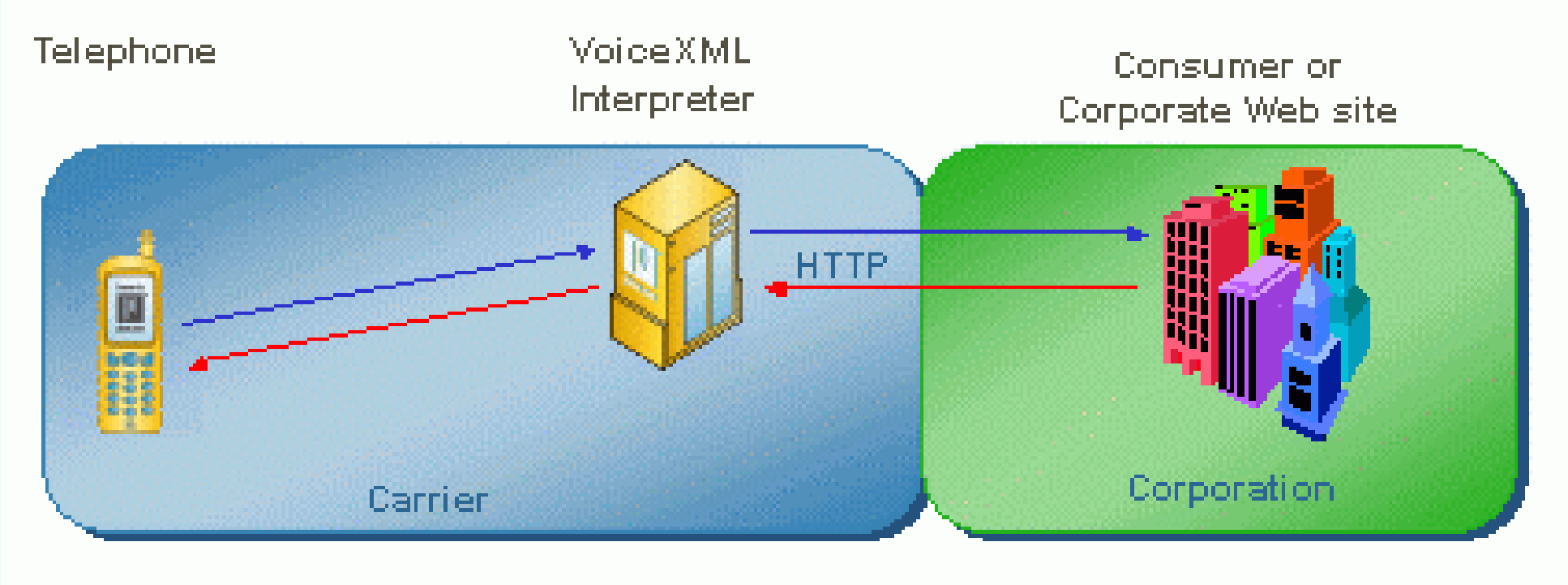
VoiceXML ist

- eine Erweiterung der XML Sprache, gerichtete auf die Entwicklung von Sprachschnittstellen und Sprachportalen,
- relativ einfache, leicht lernbare und verstehbare Programmiersprache,
- Aufbau der Sprache ermöglicht praktisch beliebeige Zerlegung des Dialogkonzepts auf die beliebige Zahl von Dialogbausteinen – Teildialogen, die einfach und separat model-liert und implementiert werden können

VoiceXML

- ist die XML-nahe Sprache, erweiterte für die Entwicklung von Sprachschnittstellen und Sprachportalen,
- bietet dem Entwickler volle Kontrolle über die Dialogsteuerung an,
- die in VoiceXML entwickelten Anwendungen können ohne Probleme auf dem Standard-web-server abgespeichert und von dort aufgerufen werden

Prinzipielle Anwendung der VoiceXML Sprache



In VoiceXML können einfach beschrieben werden:

- gesprochene “prompts” (synthetische Sprache)
- Ausgabe von audio files und streams
- Erkennung von gesprochenen Wörtern, Wortverbindungen, Phrasen und Sätzen
- Erkennung von “touch tone (DTMF) key press”
- Aufnahmen von Spracheingaben
- Dialogsteuerung
- Telephone control (call transfer and hangup)

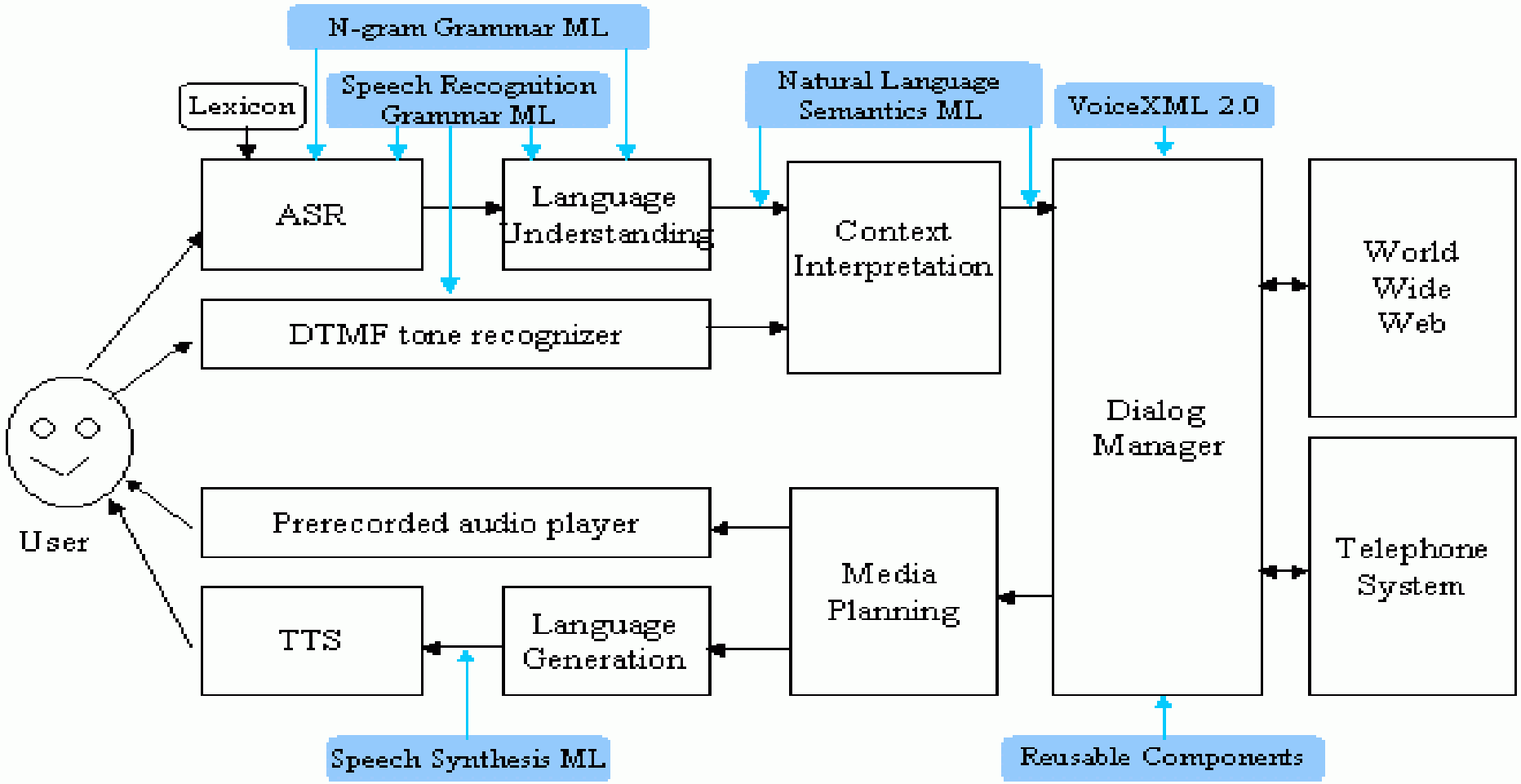


Figure 1. W3C Speech Interface Framework

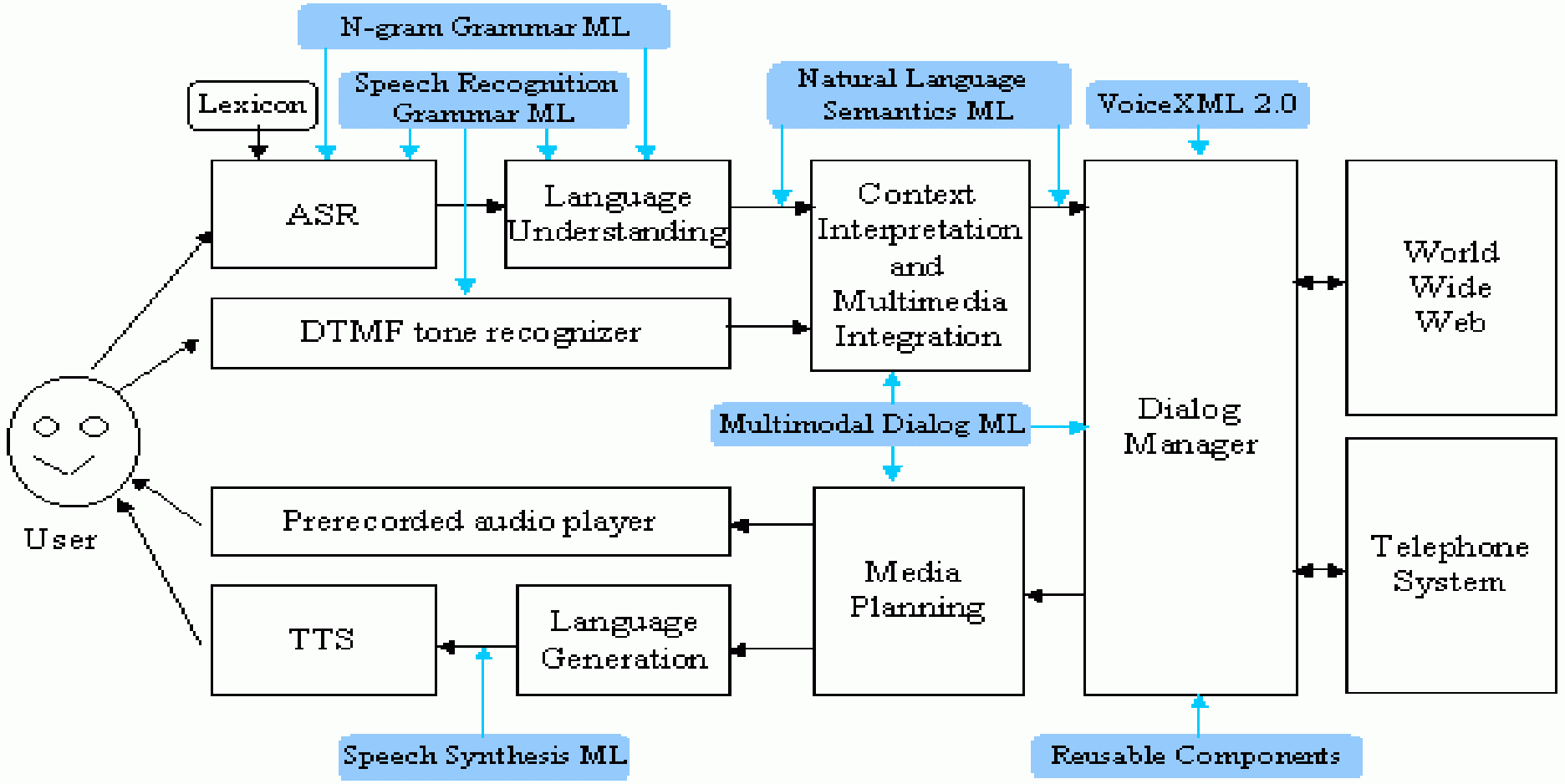


Figure 2. W3C Speech Interface Framework extended to support multimodal input and output

Wie VoiceXML System funktioniert ?

- Der Benutzer ruft z.B. das Sprachportal an.
- Das VoiceXML Steuerungssystem sendet die Anforderung auf die angegebene URL Adresse und ruft die entsprechende VoiceXML Applikation auf.
- Dialogstart.
- Das Steuerungssystem interpretiert die VoiceXML Seite sowie auch die vom Benutzer angegebene und vom Spracherkennungssystem erkannte Kommandos.
- Die vom System erhaltene Ergebnisse werden dem Benutzer durch das TTS, audio-files oder streaming media präsentiert.

Das einfachste VoiceXML Programm:

```
<?xml version="1.0">
<vxml version="1.0">
<!-- hello world example -->

<form>
  <block> Hello World! </block>
</form>
</vxml>
```

VoiceXML Elements

Element	Purpose
<assign>	Wertzuweisung an eine Variable
<audio>	Audio clip zwischen “prompt” tags ausgeben (überspielen)
<block>	Definiert den Block des (non-interactive) ausführbaren Code
<break>	JSML Element zur Einbettung einer Pause in Ausgabe
<catch>	Definition und Behandlung (catch) eines Ereignisses
<choice>	Definition des Menü-Items
<clear>	Löscht eine oder mehrere Variablen des Form-Elements
<disconnect>	Schliesst (beendet) die Session
	JSML Element zur Klassifizierung des Textbereiches als Teiltyp
<dtmf>	Spezifiziert eine “touch-tone key” Grammatik
<else>	“Unerfolgreiche” Alternative des <if> Elements

<elseif>	“Unerfolgreiche” Alternative des <if> Elements+neue Verzweigung
<emp>	JSML Element zur Einstellung der Betonung der Sprachausgabe
<enumerate>	Numerierung und Ausgabe der Menü-Alternativen
<error>	Definition und Behandlung eines Fehlerereignisses
<exit>	Verlassen der Session
<field>	Definiert ein Eingabefeld im Formular (form)
<filled>	Ausführung einer Aktion, falls Eingabefeld gefüllt wurde
<form>	Definiert ein Formular für einen Dialog zur Dateneingabe (Datensammlung) und Informationsausgabe
<goto>	Übergang zu einem anderen Dialog des gleichen oder anderen Dokuments
<grammar>	Spezifiziert die Grammatik für die Spracheingabe (natürliche Spracherkennung
<help>	Definition und Behandlung eines Hilfe-Ereignisses

<if>	Einfache Verzweigung
<initial>	Definiert die Voranmeldung beim Öffnen des Dialogs
<link>	Definiert die Verbindungen zu anderen Dokumenten, Dialogen, Grammatiken
<menu>	Dialog, der die Auswahl zwischen mehreren Alternativen erlaubt
<meta>	Definiert das Meta-data Item vom Typ ‘name/value’
<noinput>	Definition und Behandlung keiner Benutzerreaktion
<nomatch>	Definition und Behandlung einer falschen Benutzerreaktion
<object>	Definiert die Interaktion mit benutzerdefiniertem Objekt
<option>	Spezifiziert Alternatives der Eingabe in <field>
<param>	Definiert Parameter in <object> oder <subdialog>
<prompt>	Ausgabe über TTS oder direkte (aufgezeichnete) Audioausgabe
<property>	Einstellung von Implementierungs-Plattformparametern (settings)
<pros>	JSML Element zur Einstellung des Prosodie der Sprachausgabe

<record>	Aufzeichnung der direkten Audioeingabe
<reprompt>	Ausgabe von alternativen Prompts nach einem Entstehen eines Ereignisses
<return>	Rückkehr aus dem Subdialog-Modul
<sayas>	JSML Element zur Einstellung, im welchen Stil die Sprachausgabe erfolgen soll
<script>	Spezifiziert benutzerorientierte Verwendung eines ECMAScripts
<subdialog>	Ruft einen anderen Dialog als bereits laufenden Subdialog auf
<submit>	Übertragung einer Variablenliste an Dokumentenserver
<throw>	Ruft ein Ereignis hervor
<transfer>	Überweisung des Anrufenden auf anderen Zielort
<value>	Fügt den Wert einer Ausdrucksauswertung in ein Prompt ein
<var>	Definiert eine Variable
<vxml>	Anfangselement eines jeden VoiceXML-Dokuments

Das “echte” VoiceXML Programm:

```
<?xml version="1.0">
<vxml version="1.0">
<!-- hello world example -->

<var name="hi" expr="'Hello World!'"/>

<form id="say_hi">
  <block>
    <prompt><value expr="hi"/></prompt>
    <goto next="#say_goodbye"/>
  </block>
</form>
```

```
<form id="say_goodbye">
  <block>
    Goodbye!
  </block>
</form>
</vxml>
```

Das andere einfache Programm:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="1.0">
<!-- drink example -->
  <form>
    <field name="drink">
      <prompt>Would you like coffee, tea, milk, or
              nothing ?</prompt>
      <grammar src="drink.gram" type="application/x-jsgf"/>
    </field>
    <block>
      <submit next="http://www.drink.example/drink2.asp"/>
    </block>
  </form>
</vxml>
```

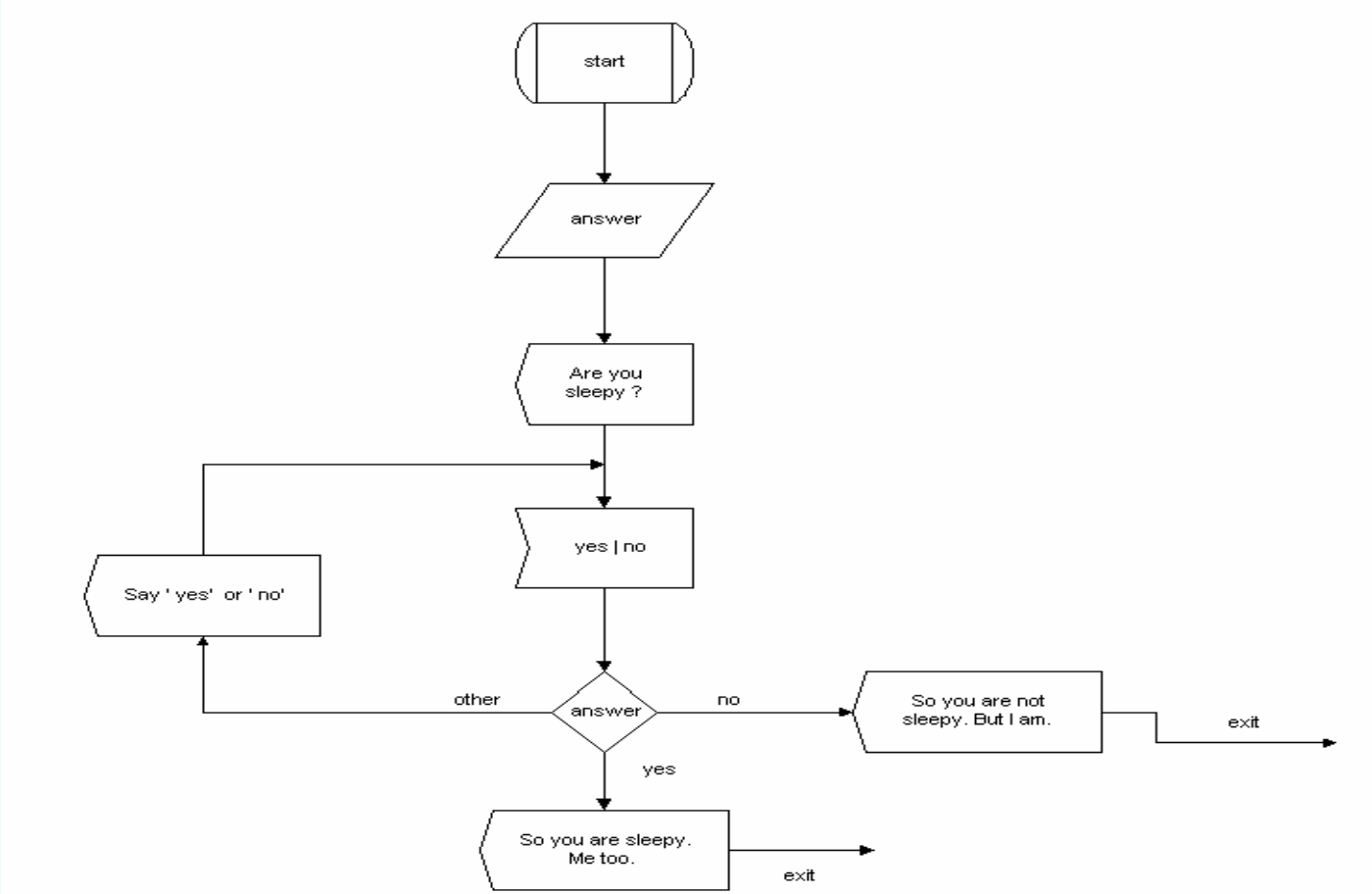
Verwendung von Variablen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0">
<!-- Variables Example -->

<var name="a" expr="1"/>
<form id="start">
  <var name="b" expr="'hello'"/>
  <block>
    <prompt> Variable a has value: <value expr="a"/> </prompt>
    <prompt> Let's try to evaluate an easy expression: a+1 = <value expr="a+1"/>
      </prompt>
    <prompt> Variable b has value: <value expr="b"/> </prompt>
    <prompt> Let's try string concatenation: b+' world' = <value expr="b+' world'"/>
      </prompt>
  </block>
</form>
</vxml>
```

```
Variable a has value: 1
Let's try to evaluate an easy expression: a+1 = 2
Variable b has value: hello
Let's try string concatenation: b+' world' = hello world
```

Zusammenhang der symbolischen Darstellung des Dialogs mit dem VoiceXML–Programm:



Programm:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="1.0">
<!-- simple grammar and event handling example -->

<form id="start">
  <field name="answer">
    <catch event="noinput">
      <prompt> Hey, don't sleep! </prompt>
    </catch>
    <catch event="nomatch">
      <prompt> say 'yes' or 'no' </prompt>
    </catch>
    <grammar> yes{yes} | y{yes} | no{no} | n{no} </grammar>
    <prompt> Are you sleepy? </prompt>
    <filled>
      <if cond="answer=='yes'">
        <prompt> So you are sleepy. Me too. </prompt>
      <else/>
        <prompt> So you are not sleepy. But I am. </prompt>
      </if>
    </filled>
  </field>
</form>
</vxml>
```

Mögliche Dialogführungen:

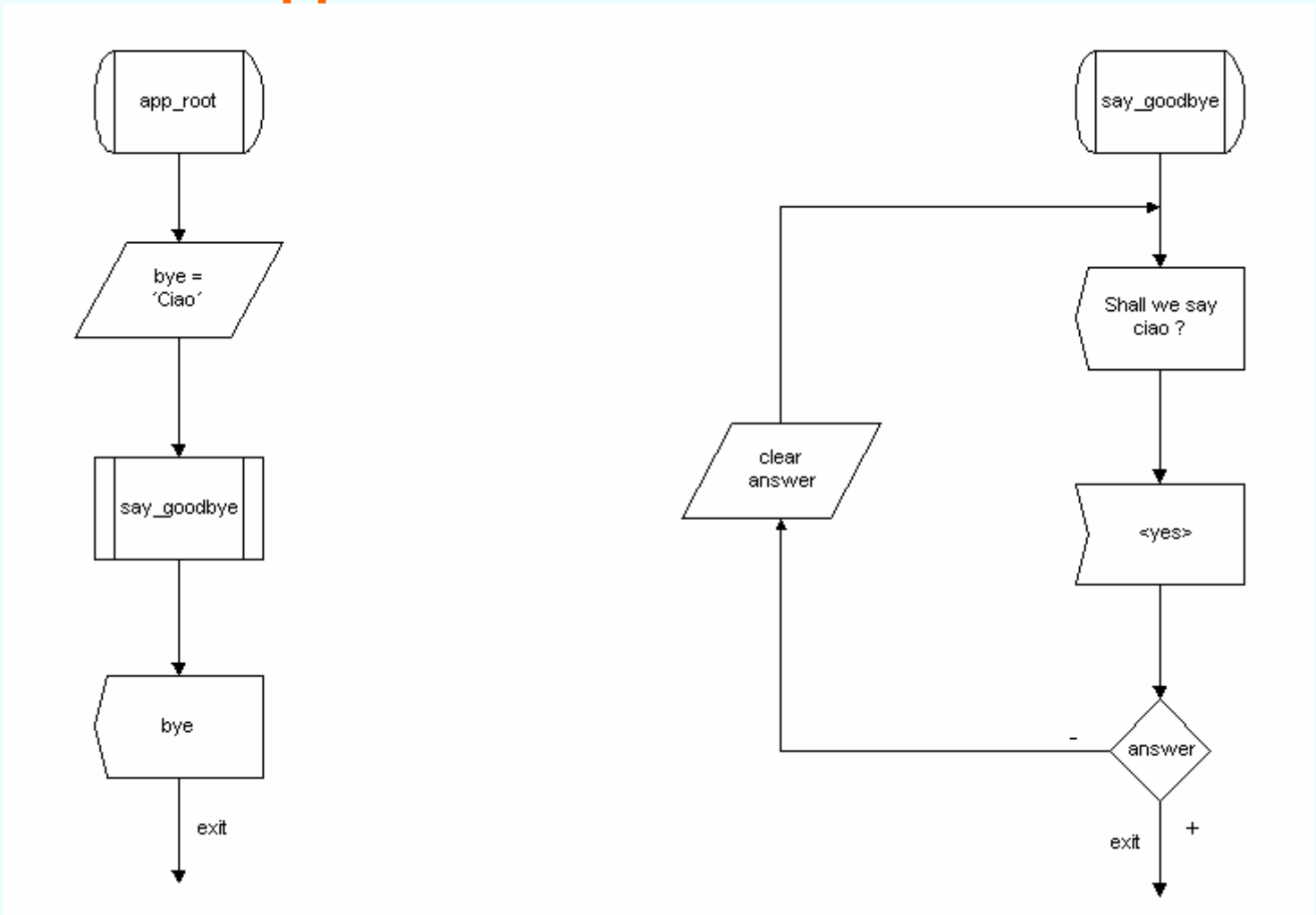
```
1)  Are you sleepy?
    yes
    So you are sleepy. Me too.

2)  Are you sleepy?
    no
    So you are not sleepy. But I am.

3)  Are you sleepy?
    y
    So you are sleepy. Me too.

4)  Are you sleepy?
    si
    say 'yes' or 'no'
    ciao
    say 'yes' or 'no'
    no
    So you are not sleepy. But I am.
```

Multi-Document Application:

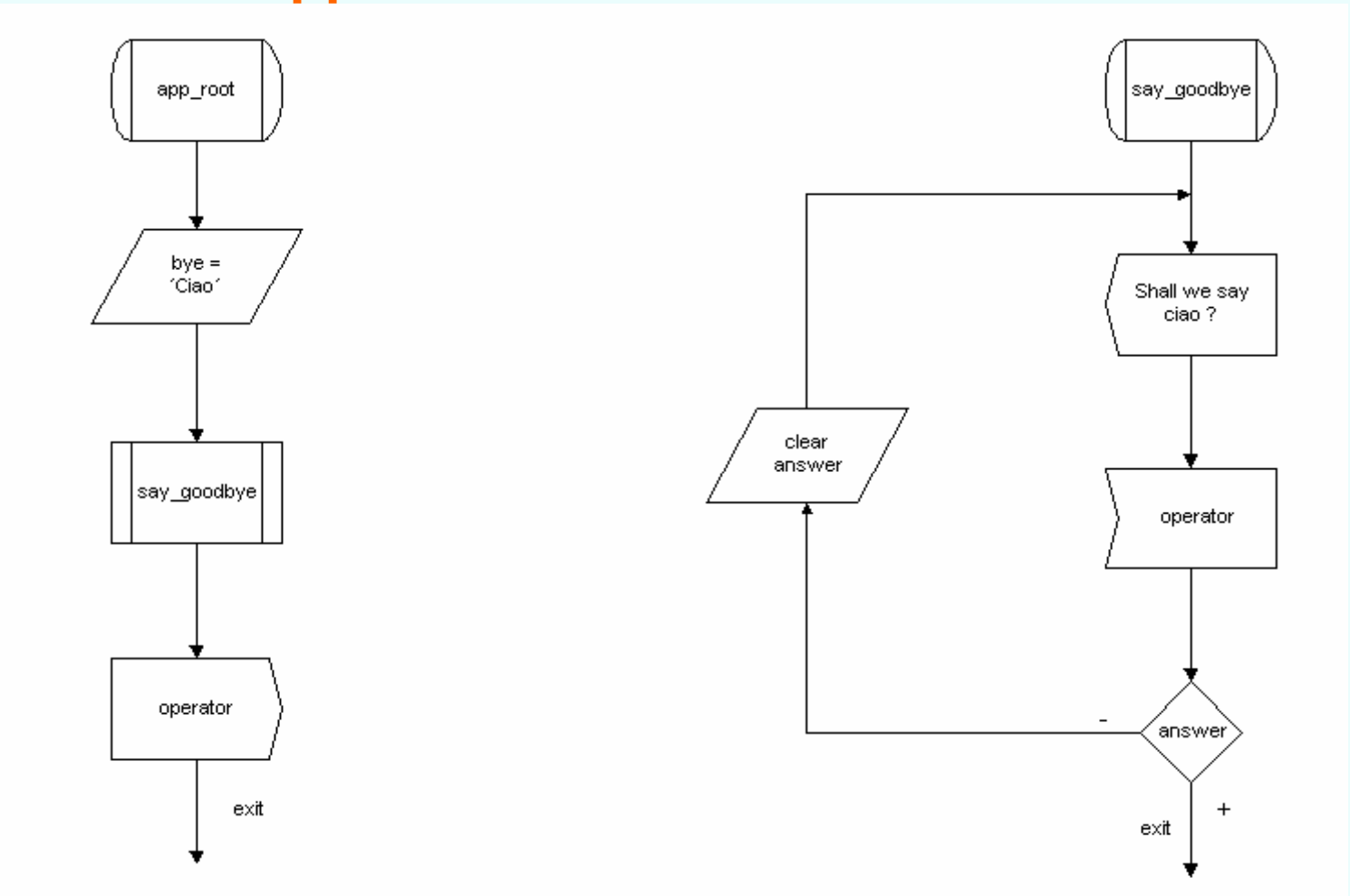


Programm:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0">
<!-- Application root document -->
  <var name="bye" expr="'Ciao'"/>
    <grammar> <rule id="root" scope="public">
                yes{yes} | y{yes} | no{no} | n{no} </rule> </grammar>
    <goto next="#say_goodbye"/>
</vxml>

<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0">
<!-- Application leaf document -->
  <form id="say_goodbye">
   <field name="answer" type="boolean">
    <prompt> Shall we say <value expr="application.bye"/> ? </prompt>
    <filled>
      <if cond="answer">
        <exit/>
      </if>
      <clear namelist="answer"/>
    </filled>
   </field>
  </form>
</vxml>
```

Multi-Document Application No. 2:

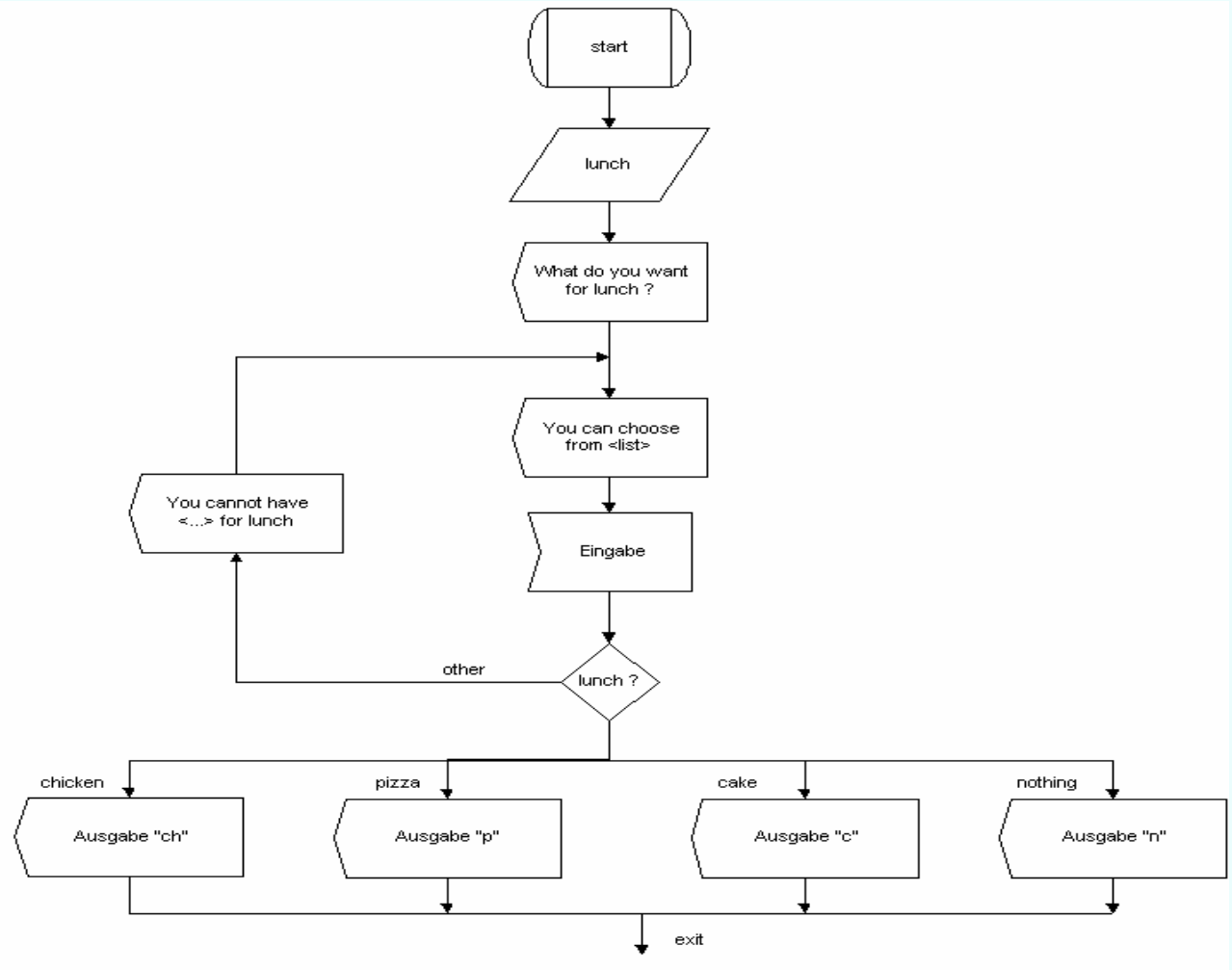


Programm 2:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0">
<!-- Application root document -->
  <var name="bye" expr="'Ciao'"/>
  <link next="operator_xfer.vxml">
    <grammar> <rule id="root" scope="public"> operator </rule> </grammar>
  </link>  <goto next="#say_goodbye"/>
</vxml>

<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0">
<!-- Application leaf document -->
  <form id="say_goodbye">
   <field name="answer" type="boolean">
    <prompt> Shall we say <value expr="application.bye"/> ? </prompt>
    <filled>
      <if cond="answer">
        <exit/>
      </if>
      <clear namelist="answer"/>
    </filled>
   </field>
  </form>
</vxml>
```

Auswahl aus mehreren Alternativen:



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="1.0" >
<!-- Options, link and shadow variables example -->

<catch event="noinput">
  <prompt> Hey, don't sleep! </prompt>
</catch>
<catch event="exit">
  <prompt> Exit! </prompt>
  <exit/>
</catch>
<link event="help">
  <grammar> [please] help [me] [please] </grammar>
</link>
<link event="exit">
  <grammar> exit | quit | q </grammar>
</link>

<form id="start">
  <field name="lunch">
    <catch event="nomatch">
      <prompt> You cannot have <value expr="lunch$.utterance+' '"/> for lunch. You can
choose from <enumerate/> </prompt>
      <!-- shadow variable used here to get user's utterance -->
    </catch>
```

```
    <catch event="help">
      <prompt> Just say what you want for lunch. </prompt>  </catch>
    <option dtmf="1" value="ch"> chicken </option>
    <option dtmf="2" value="p"> pizza </option>
    <option dtmf="3" value="c"> cake </option>
    <option dtmf="4" value="n"> nothing </option>
    <prompt> What do you want for lunch? You can choose from <enumerate/> </prompt>
    <filled>
      <if cond="lunch=='ch'">
        <prompt> You will have chicken for lunch </prompt>
        <prompt> I like chicken too </prompt>
      <elseif cond="lunch=='p'"/>
        <prompt> You will have pizza for lunch </prompt>
        <prompt> Italia fan?  </prompt>
      <elseif cond="lunch=='c'"/>
        <prompt> You will have cake for lunch </prompt>
        <prompt> You will be fat! </prompt>
      <else cond="lunch=='n'"/>
        <prompt> You will have nothing for lunch </prompt>
        <prompt> Diet? </prompt>
      </if>
    </filled>
  </field>
</form>
</vxml>
```

Mögliche Dialogführungen:

```
1) What do you want for lunch? You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   nichts

   You cannot have  nichts  for lunch. You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   chicken

   You will have chicken for lunch
   I like chicken too
```

```
2) What do you want for lunch? You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   ja
   You cannot have  ja  for lunch. You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   was?
   You cannot have  was?  for lunch. You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   pizza
   You will have pizza for lunch
   Italia fan?
```

```
3) What do you want for lunch? You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   nichts

   You cannot have  nichts  for lunch. You can choose from
   1.  chicken
   2.  pizza
   3.  cake
   4.  nothing

   nothing

   You will have nothing for lunch
   Diet?
```

Mehr komplizierte Applikation
(besteht aus mehreren root – leaf Dokumenten)

example07_1.vxml

```
<?xml version="1.0"? encoding="ISO-8859-1"?>
<vxml version="1.0" application="example07_root1.vxml">
<!-- root document example  - start to interpret this file -->

<form id="start">
  <block>
    <prompt> Do typos, do nothing and ask for help and observe the system behaviour
      </prompt>
    <prompt> If you observed enough answer yes or no. </prompt>
  </block>

  <field name="answer">
    <grammar> yes{yes} | y{yes} | no{no} | n{no} </grammar>
    <prompt> Are you sleepy? </prompt>
    <filled>
      <if cond="answer=='yes'">
        <prompt> So you are sleepy. Me too. </prompt>
      <else/>
```

```
        <prompt> So you are not sleepy. But I am. </prompt>
      </if>
    </filled>
  </field>

  <block>
    <goto next="example07_2.vxml"/>
  </block>
</form>
</vxml>
```

example07_2.vxml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<vxml version="1.0" application="example07_root1.vxml">

<!-- root document example  - start to interpret example07_1.vxml -->

<form id="start">
  <block>
    <prompt> Try it once more </prompt>
  </block>
```

```
  <field name="answer">
    <grammar> yes{yes} | y{yes} | no{no} | n{no} </grammar>
    <prompt> Are you sleepy? </prompt>
    <filled>
      <if cond="answer=='yes'">
        <prompt> So you are sleepy. Me too. </prompt>
      <else/>
        <prompt> So you are not sleepy. But I am. </prompt>
      </if>
    </filled>
  </field>

  <block>
    <goto next="example07_3.vxml"/>
  </block>
</form>
</vxml>
```

example07_3.vxml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="1.0" application="example07_root2.vxml">

<!-- root document example  - start to interpret example07_1.vxml -->
```

```
<form id="start">
  <block>
    <prompt> Try it once more with another root document </prompt>
  </block>

  <field name="answer">
    <grammar> yes{yes} | y{yes} | no{no} | n{no} </grammar>
    <prompt> Are you sleepy? </prompt>
    <filled>
      <if cond="answer=='yes'">
        <prompt> So you are sleepy. Me too. </prompt>
      <else/>
        <prompt> So you are not sleepy. But I am. </prompt>
      </if>
    </filled>
  </field>

  <block>
    <prompt> That's all. </prompt>
  </block>
</form>
</vxml>
```

example07_root1.vxml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- root document example - start to interpret example07_1.vxml-->
<vxml version="1.0">
  <catch event="noinput">
    <prompt> root1 - Hey, don't sleep! </prompt>
  </catch>
  <catch event="nomatch">
    <prompt> root1 - I don't understand you - say 'yes' or 'no' </prompt>
  </catch>
  <catch event="help">
    <prompt> root1 - You asked for help - say 'yes' or 'no' </prompt>
  </catch>
  <catch event="exit">
    <prompt> root1 - Exit! </prompt>   <exit/>   </catch>
  <link event="help">
    <grammar> [please] help [me] [please] </grammar>
  </link>
  <link event="exit">
    <grammar> exit | quit | q </grammar>
  </link>
</vxml>
```

example07_root2.vxml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- root document example - start to interpret example07_1.vxml-->
<vxml version="1.0">
  <catch event="noinput">
    <prompt> root2 - Hey, don't sleep! </prompt>
  </catch>
  <catch event="nomatch">
    <prompt> root2 - I don't understand you - say 'yes' or 'no' </prompt>
  </catch>
  <catch event="help">
    <prompt> root2 - You asked for help - say 'yes' or 'no' </prompt>
  </catch>
  <catch event="exit">
    <prompt> root2 - Exit! </prompt>  <exit/>   </catch>
  <link event="help">
    <grammar> [please] help [me] [please] </grammar>
  </link>
  <link event="exit">
    <grammar> exit | quit | q </grammar>
  </link>
</vxml>
```

Dialogführungen:

a)

```
Do typos, do nothing and ask for help and observe the system
behaviour
If you observed enough answer yes or no.
Are you sleepy?
yes
So you are sleepy. Me too.
Try it once more
Are you sleepy?
no
So you are not sleepy. But I am.
Try it once more with another root document
Are you sleepy?
yes
So you are sleepy. Me too.
That's all.
```

b)

```
Do typos, do nothing and ask for help and observe the system
behaviour
If you observed enough answer yes or no.
Are you sleepy?
yes
So you are sleepy. Me too.
Try it once more
Are you sleepy?
no
So you are not sleepy. But I am.
Try it once more with another root document
Are you sleepy?
help
root2 - You asked for help - say 'yes' or 'no'
yes
So you are sleepy. Me too.
That's all.
```

Dialog mit einem Reservationssystem:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<vxml version="1.0">

<!-- mixed initiative dialog, filled and counters example -->

<link event="again">
  <grammar>[please] [all] again [please] | [please] once more
[please]</grammar>
</link>

<nomatch>
  Sorry, I did not understand you.
  <reprompt/>
</nomatch>

<form id="get_from_and_to_cities">
  <grammar src="cities.jsg"/>
```

```
  <catch event="again">
    <prompt> OK, once more!</prompt>
    <clear namelist="bypass_init from_city to_city"/>
    <reprompt/>
  </catch>

  <nomatch count="2">
    I am sorry, I still do not understand you. Valid cities are Paris,
Prague and Oslo
  </nomatch>
  <nomatch count="3">
    I did not understand again. I am giving up. Bye.
    <exit/>
  </nomatch>

  <block> Welcome to the airplain ticket reservation system. </block>

  <initial name="bypass_init">
    Where do you want to fly from and to?
    <nomatch count="1">
      Sorry, I did not understand you. Say something like "from Oslo to
Praha".
```

```
    </nomatch>
    <nomatch count="2">
      I am sorry, I still don't understand.
      I will ask you for information one piece at a time.
      <assign name="bypass_init" expr="true"/>
      <reprompt/>
    </nomatch>
  </initial>

  <field name="from_city" slot="from">
    <grammar src="city.jsg"/>
    From which city are you leaving?
    <prompt count="2"> Tell me from which city you are leaving, please
</prompt>
  </field>

  <field name="to_city" slot="to">
    <grammar src="city.jsg"/>
    To which city do you want to fly?
    <prompt count="2"> Tell me to which city you want to fly, please
</prompt>
  </field>
```

```
  <block>
    <prompt>
      A ticket from <value expr="from_city"/> to <value expr="to_city"/>
      is reserved for you.
    </prompt>
  </block>

  <filled namelist="from_city">
    <prompt>Your departure city is <value expr="from_city"/>.</prompt>
  </filled>

  <filled namelist="to_city">
    <prompt>Your arrival city is <value expr="to_city"/>.</prompt>
  </filled>

  <filled>
    <if cond="from_city==to_city">
      <prompt>
        Sorry, you can't fly from <value expr="from_city"/> to <value
expr="to_city"/>!
      </prompt>
```

```
      <prompt>
        Please repeat your order.
      </prompt>
      <clear namelist="bypass_init from_city to_city"/>
    </if>
  </filled>

</form>

</vxml>
```

Dialog 1:

Welcome to the airplain ticket reservation system.
Where do you want to fly from and to?
from Prague to Oslo
Your departure city is Prague.
Your arrival city is Oslo.
A ticket from Prague to Oslo is reserved for you.

Dialog 2:

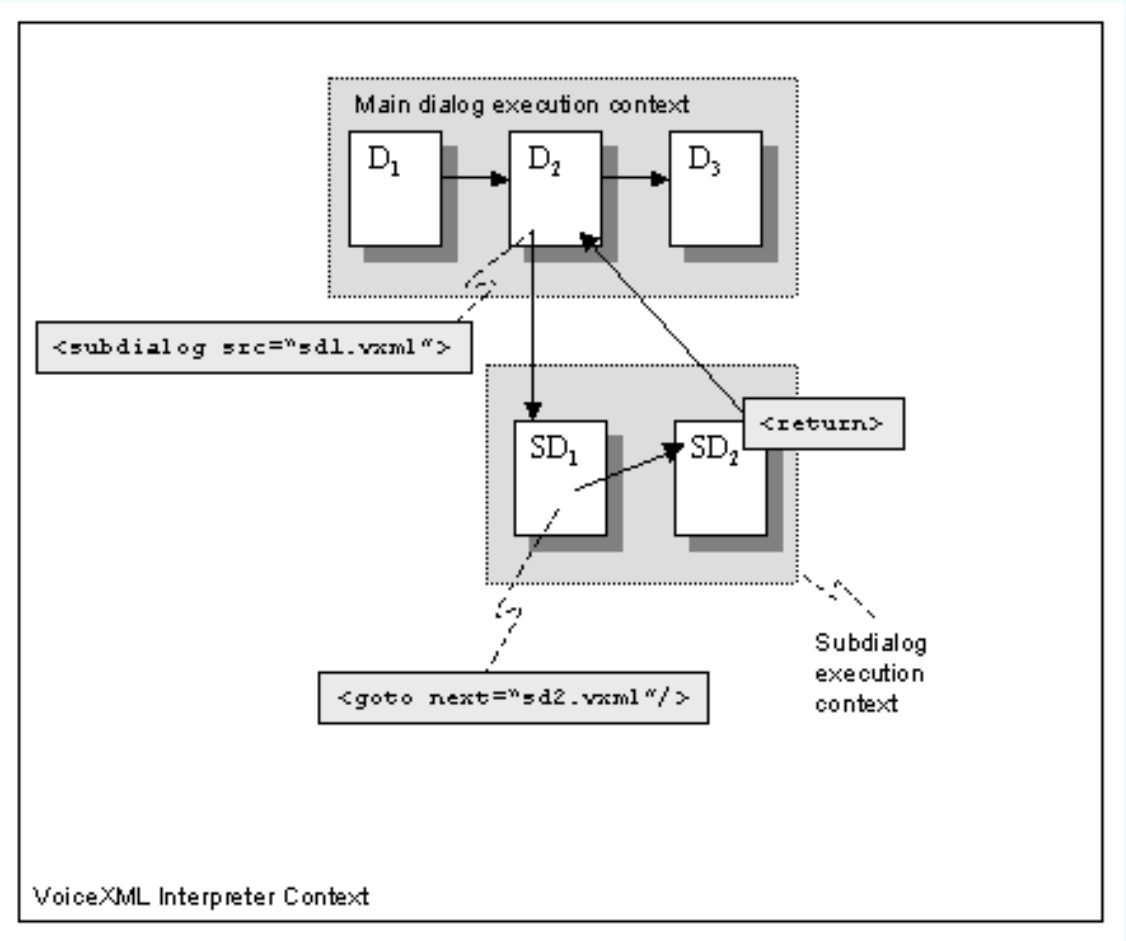
Welcome to the airplain ticket reservation system.
Where do you want to fly from and to?
Prague
Your departure city is Prague.
To which city do you want to fly?
Paris
Your arrival city is Paris.
A ticket from Prague to Paris is reserved for you.

Dialog 3:

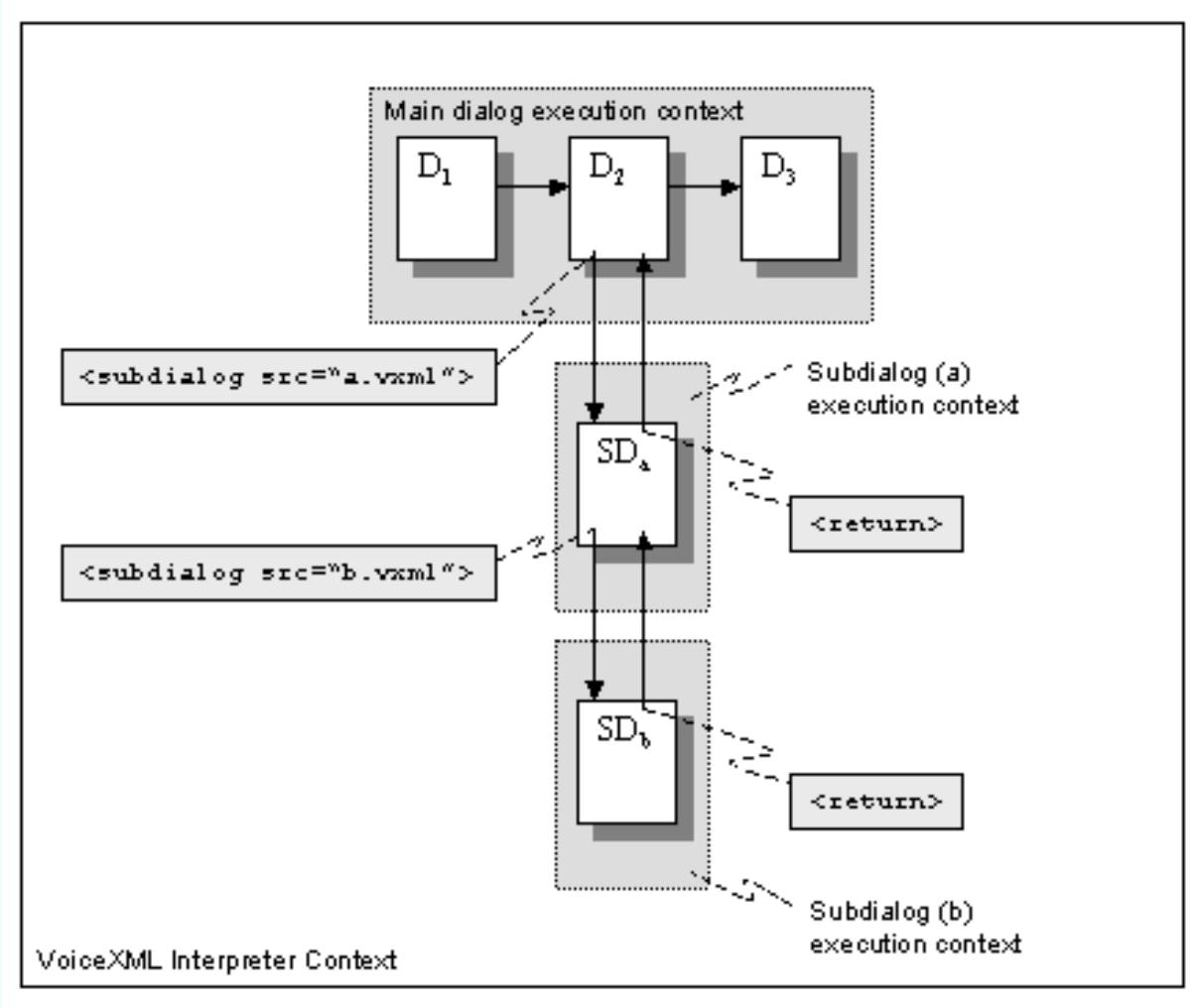
Welcome to the airplain ticket reservation system.
Where do you want to fly from and to?
Praha
Your departure city is Prague.
To which city do you want to fly?
Moscow
Sorry, I did not understand you.
Tell me to which city you want to fly, please
Moscow
I am sorry, I still do not understand you. Valid cities are Paris, Prague and Oslo
Paris
Your arrival city is Paris.
A ticket from Prague to Paris is reserved for you.

Verwendung von Teildialogen

a) Serielle Schaltung:



b) Genistete Schaltung:



Beispiel: Intercity Auskunft (Aufgabe zum Ergänzen)

```
<?xml version="1.0"?>
<vxml version="2.0">
  <var name="source_city"/>
  <var name="goal_city"/>
  <var name="source_time"/>
  <var name="goal_time"/>
  <var name="via_city"/>
  <var name="best_erg"/>
  <catch event="noinput">
    <prompt> Schlafe nicht, stehe auf ! </prompt>
  </catch>

  <form id="IC_demo">
    <block>
      <prompt> Intercity Auskunft </prompt>
      <prompt> Guten Tag. Hier ist die automatische Intercityauskunft.
               Welche Information benötigen Sie ? </prompt>
      <goto next="#eingabe_1"/>
    </block>
  </form>
```

```
<form id="eingabe_1">
  <field name="ein_1">
    <grammar src="IC_Auskunft.gram" type="application/x-jsgf"/>
     <filled>
       <prompt> Sie wollen am Freitag den 26. Juni zwischen 6 und 12 Uhr
                nach Berlin fahren ?   </prompt>
       <subdialog name="bestaet" src="IC2.vxml#basic">
        <filled>
          <assign name="best_erg" expr="bestaet.best"/>
        </filled>
       </subdialog>
       <if cond="best_erg=='nein'">
         <clear name="best_res"/>
         <prompt> Ich verstehe Ihnen nicht. Wiederholen Sie, bitte, die
                  Angaben. </prompt>
         <prompt> Welche Information benötigen Sie ? </prompt>
         <goto next="#eingabe_1"/>
       </if>
       <!--  Call "Auswahl_Zielort"  -->
       <assign name="goal_city" expr="ein_1$.utterance"/>
       <goto next="#eingabe_2"/>
     </filled>
```

```
    </field>
  </form>

  <form id="eingabe_2">
    <field name="ein_2">
      <grammar src="IC_Auskunft.gram" type="application/x-jsgf"/>
      <prompt> Wo wollen Sie abfahren ? </prompt>
       <filled>
         <prompt> Sie wollen von Regensburg abfahren ?
         </prompt>
         <subdialog name="bestaet" src="IC2.vxml#basic">
          <filled>
            <assign name="best_erg" expr="bestaet.best"/>
          </filled>
         </subdialog>
         <if cond="best_erg=='nein'">
           <clear name="best_res"/>
           <goto next="#eingabe_2"/>
         </if>
         <!--  Call "Auswahl_Regensburg"
         <assign name="source_city" expr="eingabe_1$.utterance"/>  -->
         <prompt> source_city = Regensburg </prompt>
```

```
        <goto next="#antwort_1"/>
      </filled>
    </field>
  </form>

  <form id="antwort_1">
    <block>
      <prompt> Nach Berlin Hbf fahren Sie ab Regensburg um ...
      </prompt>
      <goto next="#eingabe_3"/>
    </block>
  </form>

  <form id="eingabe_3">
    <block name="ein_3">
      <prompt> Soll ich die Verbindung wiederholen ? </prompt>
        <subdialog name="bestaet" src="IC2.vxml#basic">
        <filled>
          <assign name="best_erg" expr="bestaet.best"/>
        </filled>
        </subdialog>
        <if cond="best_erg=='ja'">
```

```
        <goto next="#antwort_1"/>
      </if>
    <prompt> Benötigen Sie eine weitere Information ?  </prompt>
      <subdialog name="bestaet" src="IC2.vxml#basic">
      <filled>
        <assign name="best_erg" expr="bestaet.best"/>
      </filled>
      </subdialog>
      <if cond="best_erg=='ja'">
        <goto next="#eingabe_4"/>
      </if>
    <goto next="#abschied"/>
  </block>
</form>

<form id="eingabe_4">
  <block>
    <prompt> Welche weitere Information benötigen Sie ? </prompt>
    <goto next="#eingabe_1"/>
  </block>
</form>
```

```
  <form id="abschied">
    <block>
      <prompt> Auf Wiederhören ! </prompt>
    </block>
  </form>
</vxml>
```

```
<?xml version="1.0"?>
<vxml version="2.0">
  <form id="basic">
    <field name="best">
      <grammar> ja{ja} | j{ja} | nein{nein} | n{nein} </grammar>
      <filled>
        <return namelist="best"/>
      </filled>
    </field>
  </form>
</vxml>
```

```
grammar IC_Auskunft;                    // name

public <main> = <frage1> | <frage2> | <abschied>;
<frage1> = <bitte> [<tag>] [<zeit>] <prep> <ort>;
<frage2> = <prep> <ort>;
<bitte>  = ich moechte;
<tag>    = heute | morgen;
<zeit>   = frueh | vormittags | nachmittags | abends;
<ort>    = Berlin | Muenchen | Regensburg;
<prep>   = von | nach;
<abschied> = [danke] auf wiederhoeren;
```

5. Grundlagen des Programmierens in VoiceXML