ボイスウェブの可能性 － VoiceXML 概説－

荒木 雅弘
京都工芸繊維大学工芸学部電子情報工学科
araki@dj.kit.ac.jp

本稿では，音声対話を記述するための言語である VoiceXML を紹介する．VoiceXML は自動電話応答システムや音声による Web アクセスのための標準化言語として設計されたが，単純なシステム主導対話だけでなく，一定の形式の混合主導対話も記述可能である．また，仕様を拡張し，統計的言語モデルとロバストな言語解析を用いることによって，ユーザ主導対話が記述可能な製品も発売されている．本稿では主導権という観点から対話を分類し，それぞれの対話が VoiceXML を用いてどのように記述されるかを紹介しながら，VoiceXML の主要な概念について説明する．

音声対話アプリケーションと VoiceXML

「目的地までの道順が分からない．Web のあのページに書いてあったのに ...」「電車の時刻表が今ここで調べられたら ...」というような状況に出会うことはよくあるだろう．Web が発達し，生活と仕事に必要な情報は今やすべて Web 上にあるといっても過言ではない．i モードや通信可能な携帯端末を使えば，いつでもどこでも Web 上の情報にアクセスすることは可能であるが，使い勝手の問題や所要時間を考慮すると，やはり音声での Web アクセスに軍配があがるだろう．

1990 年代後半の音声認識の高精度化・高速化を背景に，コールセンターの対応自動化や電話による Web アクセスなどの実現を目指して，音声認識・音声合成・対話処理を統合した音声対話システムの開発が，米国の AT&T，IBM，Lucent，Motorola などの各社で試みられるようになった．

音声対話システムの開発には，音声信号・自然言語・ユーザインタフェース・アプリケーションロジックなどの多岐にわたる知識が必要であることから，効率的に開発・保守するためには音声インタフェースをコントロールできる高レベルのサービス記述言語が必要であるという認識は各社とも共通しており，各社は独自のサービス記述言語の開発を行っていた．1990 年代後半の XML を中心とした技術標準の流れに乗って，1999 年 3 月に音声インタフェースのための標準マークアップ言語 VoiceXML（Voice Extensible Markup Language）を策定するための VoiceXML フォーラム[☆1] が上記の各社を中心に発足した．2000 年 3 月には，この VoiceXML フォーラムより VoiceXML の正式仕様であるバージョン 1.0 が発表された．

一方，WWW の標準化団体である W3C（World Wide Web Consortium）では 1999 年 3 月に音声ブラウザに関する複数の仕様の標準化活動を行う Voice Browser Working Group[☆2] が活動を開始した．そのサブグループである Dialog Markup Language Subgroup が仕様のベースとして VoiceXML 1.0 を採用したことによって，VoiceXML Forum と W3C における標準化活動が一本化された．このような経緯で VoiceXML 2.0 の仕様策定は W3C で行われており，2003 年 9 月現在，勧告候補が発表されている [1)]．

VoiceXML は XML をベースにしたボイス・マークアップ言語と呼ばれるもので，音声認識，DTMF（Dual Tone Multiple Frequency; 電話のトーンダイアル）キー入力，録音，音声合成，オーディオファイルの再生，電話転送機能などを組み合わせて，音声アプリケーションを開発するための言語である（**図 -1**）．

VoiceXML で記述されたコンテンツは電話からの音声入力によって探索でき，目的のコンテンツは，音声合成（text-to-speech）や録音音声で聞くことができる．VoiceXML は電話による自動音声応答システムや音声に

☆1 http://www.voicexml.org/
☆2 http://www.w3.org/Voice/

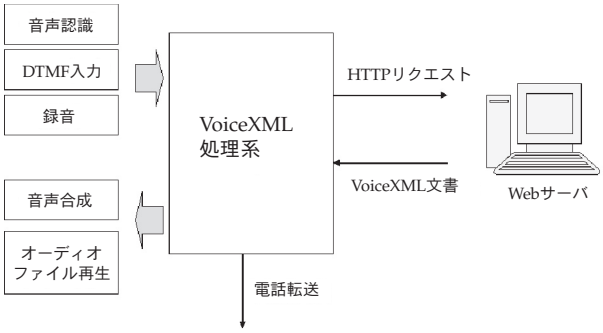

図 -1　VoiceXML の設計概念

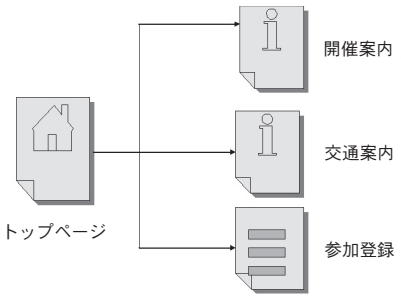

図 -2　学会案内・参加登録システムのサイトマップ

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <menu>
  <prompt>
   こちらは○○学会開催案内・参加登録システムです。
   ご希望のメニューをお選び下さい。<enumerate/>
  </prompt>
  <choice next="guide.vxml"> 開催案内 </choice>
  <choice next="access.vxml"> 交通案内 </choice>
  <choice next="register.vxml"> 参加登録 </choice>
 </menu>
</vxml>
```

図 -3　メニューを用いたトップページ（top.vxml）の例

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <form>
  <block>
   開催案内です。○○学会は、平成 15 年 7 月 28 日
   京都△△大学にて開催されます。
   <goto next="top.vxml"/>
  </block>
 </form>
</vxml>
```

図 -4　開催案内ページ（guide.vxml）の例

よる Web アクセスを主要な応用として設計されたが，柔軟な対話制御やスコープを持つ文法・変数を利用することによって，より広い範囲の音声インタフェース開発に応用可能である．

本稿では，さまざまな対話パターンを VoiceXML を用いて表現する方法を説明することを通じて，VoiceXML の基本概念を解説することを目的とする．対話を分類する方法はいくつか考えられるが，本稿では主導権に着目し，システム主導・混合主導・ユーザ主導の対話に関して，それぞれ適した対話タスクと対応付けながら説明する．なお，VoiceXML2.0 仕様書はまだ勧告ではないものの，準拠製品が複数発売されていることもあり，今後大きな仕様変更はないものと予想されるので，本稿でも VoiceXML2.0 の仕様書 [1] に準じた説明を行う．

システム主導対話の表現

まず，学会等の案内や参加登録を行う Web ページと同等の機能を持つ音声対話システムを例として，システムが主導するタイプの対話をどのようにして VoiceXML で記述するのかを解説する．ここでは**図 -2** に示すようなサイトマップを持つ Web サイトと同等の音声対話システムを想定する．

■メニューにより分岐する対話

サイトのトップページ（ユーザが電話をかけてきたときに，最初に処理系にロードされ，対話が始まるページ）は，このサイトの機能を説明し，ユーザを開催案内・交通案内・参加登録のいずれかに導く役目を持つ．このような機能は VoiceXML の menu 要素を用いて実現する．

1 行目は XML 宣言であり，すべての XML 文書に必須のものである．2 行目の <vxml> から 12 行目の </vxml> の中に VoiceXML の仕様で定義された要素を用いて対話を記述する．vxml 要素では，VoiceXML のバージョンを記述する version 属性と，VoiceXML の名前空間を表す xmlns 属性（本稿では省略）が必須属性である．3 行目の menu 要素がメニュー選択を行う対話を構成する親要素である．典型的には，システム発話によって選択肢を提示する prompt 要素と，選択肢（これがそのままユーザ入力発話の文法になる）およびその選択肢が選ばれたときの遷移先を定義する複数の choice 要素を子要素として持つ．また，menu 要素のうちの prompt 要素において，enumerate 要素を用いると，選択肢の集合を読み上げることができる．

■情報提示を行う対話

次に，単純な開催案内を行うページの例を**図 -4** に示す．ここで用いられている form 要素と，先に説明した menu 要素が VoiceXML において対話を構成する 2 つの

```
S (System): こちらは○○学会開催案内・参加登録システムです。
    ご希望のメニューをお選び下さい。
    開催案内、交通案内、参加登録。
U (User): 開催案内
S: 開催案内です。○○学会は、平成 15 年 7 月 28 日
    京都△△大学にて開催されます。
    こちらは○○学会開催案内・参加登録システムです。
    ご希望のメニューをお選び下さい。
    開催案内、交通案内、参加登録。
```

図 -5　学会案内・参加登録システムの対話例

重要な要素である．menu 要素は**図 -3** に示したように，ユーザに選択肢を示し，対話をガイドする役割を果たす．一方，form 要素はユーザに情報を提示し，ユーザから入力を受け取り，必要な値をサーバに渡すか，あるいは対話フローを制御する処理を行う．form 要素の主要な構成要素は，変数操作や条件分岐のような実行可能内容を記述することができる block 要素，変数を指定してユーザからその値を得る field 要素，変数値を得た後の処理を記述する filled 要素などである．図 -4 の form 要素はユーザへの情報提示部分と対話フローの制御部分のみを持つ．

図 -4 の例ではユーザ入力を必要としないので，form の内容は block 要素のみである．block 要素にテキストが直接記述されている場合は，合成音声によって出力される．この VoiceXML では，合成音声で開催案内を行った後，goto 要素で指定する top.vxml（トップページ）に対話を遷移させる．図 -3，図 -4 の VoiceXML によって可能な対話は**図 -5** に示すようなものになる．

■情報を取得する対話

次に，ユーザから情報を取得する対話の記述に移る．原則として，ユーザの入力によって値を得る変数 1 つに対して 1 つの field 要素を用意し，複数の関連する field 要素を form 要素によってまとめるという形態をとる．**図 -6** に学会参加登録を行う VoiceXML の例を示す．

form 要素は，その子要素（フォーム項目と呼ぶ）を実行する順序がフォーム解釈アルゴリズムによって定められている．すべてのフォーム項目は変数を持っており，項目の記述順に値の埋まっていない変数を探してその項目を実行するのがフォーム解釈アルゴリズムの概略である．block や field などのフォーム項目は明示的に変数を持つこともあれば，隠れ変数の存在が仮定されていることもある．たとえば，block 要素では隠れ変数が仮定されており，block 要素の内容を実行すると，隠れ変数に実行済みの値が代入される．field 要素の変数はユーザからの入力によって値が代入される．フォーム解釈アルゴリズムによって，form 内では順次，値の代入されていないフォーム項目が実行される．

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <form>
  <block> 参加登録ページです。</block>
  <field name="number">
   <prompt> 氏名を入力してください </prompt>
   <grammar src="name.grxml"
        type="application/srgs+xml"/>
  </field>
  <field name="number" type="digits">
    <prompt> 会員番号を入力してください </prompt>
  </field>
  <field name="party" type="boolean">
    <prompt> 懇親会に参加されますか </prompt>
  </field>
  <filled>
   <submit next="http://localhost:8080/register"/>
  </filled>
 </form>
</vxml>
```

図 -6　参加登録ページの例（register.vxml）

4 行目の block 要素でページの案内がされた後，5 行目の field 要素に制御が移る．field 要素では，値を埋めるべき変数（name 属性），値の入力を促すシステム発話（prompt 要素），ユーザ入力文法（grammar 要素，field 要素の type 属性，または option 要素）を指定する．値が入力された後の処理を filled 要素で記述することもできる．ここでは，5 行目の field 要素に処理が移ると，氏名入力を促すシステム発話が出力され，氏名入力用の文法（name.grxml）によって認識器を起動し，ユーザからの入力待ちになる．ユーザから文法に適合した発話がなされると，変数 name に値が入力され，次の値が埋まっていないフォーム項目（この場合は 10 行目の field 要素）に制御が移る．

10 行目の field 要素は grammar 要素で文法を指定する代わりに，type 属性で組み込み文法を指定している．組み込み文法には，真偽値（はい，いいえなど），日付，時間，数字列，数，通貨，電話番号がある．いくつかの処理系ではこれ以外にも住所や氏名などの文法を用意している場合がある．

すべての変数の値が埋まった後，処理は 16 行目の filled 要素に移る．ここでの filled 要素の内容は，17 行目の submit 要素のみである．submit 要素は next 属性によって値を送るサーバのプログラムを指定し，namelist 属性でサーバへ送る変数を指定する．namelist 属性がない場合はすべての変数とその値がサーバへ送られる．

サーバサイドの処理では，サーブレットなどで変数値を受け取り，データベースへの登録を行い，その処理結果（登録受理，満員によりキャンセル待ちなど）に応じて，応答となる VoiceXML を動的に作成する．図 -6 の

VoiceXML によって図 -7 に示すような対話ができる．

この例で見てきたように，システム主導対話はメニュー選択・情報提示・変数値の取得の対話を組み合わせて実現することができる．

■文法の記述

VoiceXML2.0 においては XML で表現された SRGS（Speech Recognition Grammar Specification）[2] 形式の文法をすべての処理系がサポートすることを義務付けている．たとえば，氏名を受け付ける文法は図 -8 のようになる．

rule 要素で文法規則を記述し，ruleref 要素でその規則を参照できる（文脈自由文法の非終端記号に相当）．規則の中身は one-of 要素で選択肢を示し，item 要素の repeat 属性でその出現回数を記述できる．これらを組み合わせると，文脈自由クラスの文法が記述可能である．

混合主導対話の表現

混合主導対話とは，ユーザとシステムそれぞれが必要な場合に主導権をとって進めてゆく対話である．ユーザの主導権取得は，システムがユーザに回答を要求しているような状態において，回答として想定している文法外の発話をユーザが行うことによって行う．すなわち，システムはある状態で複数の文法を用意してユーザ発話入力を待ち，主導権取得発話が入力されれば，それに適した処理を行う必要がある．VoiceXML ではこのようなメカニズムをスコープを使って実現している．

■ VoiceXML におけるスコープ

ここで混合主導対話を実現する手法を説明するために，VoiceXML におけるスコープの概念の導入を行う．VoiceXML の文法・変数・イベント（ヘルプを求める発話の入力や指定時間以上の無音など）の処理にはプログラミング言語と同様のスコープが定義されている（図 -9）．

最も広いスコープはセッション（session）で，ユーザが電話をかけてから切るまでに相当する．セッションスコープは接続情報に関するリードオンリーの変数のみを持ち，新たに変数を定義することはできない．

アプリケーション（application）はひとまとまりのタスクを指す概念で，1 つのルートドキュメント（大域的な文法や変数を記述する VoiceXML ファイル）と 1 つ以上のリーフドキュメント（具体的なやりとりを記述する VoiceXML ファイル）からなる．リーフドキュメントが次々に入れ替わっても，同一のルートドキュメントを親ドキュメントとして指定している限りは同一のアプ

```
S: 参加登録ページです。氏名を入力してください
U: あいさきいちろう
S: 会員番号を入力してください
U: 12345
S: 懇親会に参加されますか
U: はい
（登録処理を行うサーブレット register に処理を移す）
```

図 -7　参加登録対話の例

```
<?xml version="1.0" encoding="Shift_JIS"?>
<grammar mode="voice" xml:lang="ja-JP"
       version="1.0" root="name">
 <rule id="name" scope="public">
  <ruleref uri="#last"/> <ruleref uri="#first"/>
 </rule>
 <rule id="last">
  <one-of>
   <item> あいさき </item>
   <item> あいざわ </item>
   ...
  </one-of>
 </rule>
 <rule id="first">
  <one-of>
   <item> あいこ </item>
   <item> いちろう </item>
   ...
  </one-of>
 </rule>
</grammar>
```

図 -8　氏名入力文法の例（name.grxml）

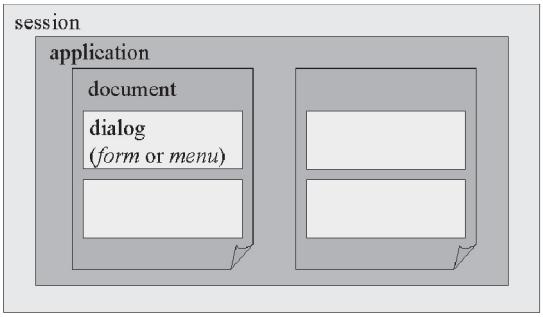

図 -9　VoiceXML におけるスコープ

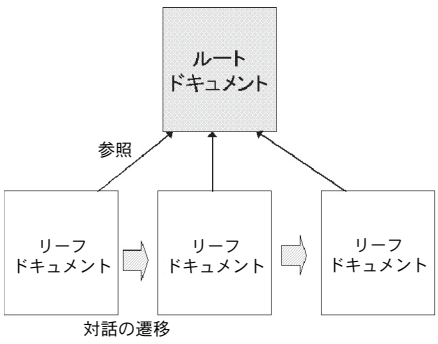

図 -10　アプリケーションの概念

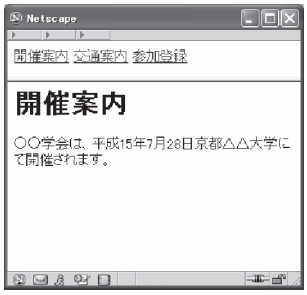

図 -11　フレームを用いた Web ページの例

リケーション内の処理とみなされ，ルートドキュメントに書かれた文法や変数は常に参照可能（プログラミング言語での大域変数に相当）である（**図 -10**）．

アプリケーションの内側にドキュメント（document）スコープがあり，さらにその内側に対話（dialog）スコープがある．ドキュメントは 1 つの VoiceXML ファイルに相当し，対話は form 要素または menu 要素である．アプリケーション－ドキュメント－対話はスコープ階層を構成し，たとえばアプリケーションレベルで書かれた文法は，そのアプリケーションを構成する VoiceXML ファイル内であればどの対話状態であっても有効である．

■スコープを用いたユーザ主導権取得発話の処理

VoiceXML ではスコープを用いることによって，対話がどの状態にあるかにかかわらずいつでも入力可能な文の集合を文法によって定義し，対象となる発話が入力されると，特定のドキュメントに遷移するなどの処理を行うことができる．これは Web ページに例えると，フレームによって常に遷移可能なページが示されており，ユーザは現在のページからいつでも別のページに遷移できるようなものである（**図 -11**）．

このような Web ページにおけるハイパーリンクに相当する機能は，VoiceXML の link 要素で実現できる．link 要素は子要素として文法（grammar 要素）を持ち，その文法にマッチした発話が行われると，next 属性で指定した場所へ遷移するか，event 属性で指定したイベントを投げる．投げられたイベントは対応する catch 要素で処理される．ルートドキュメントに link 要素を記述することによってユーザが主導権をとるタイプの発話を捕捉し，対応する処理を行うように記述すれば，混合主導対話が実現できる．図 -2 のトップページをルートドキュメントとし，ユーザ主導発話を捕捉できるようにするには，図 -3 の vxml 要素の直後に link 要素を追加する（**図 -12**）．

また，リーフドキュメントとなる VoiceXML ファイルでは，vxml 要素の application 属性でルートドキュメントを指定する（**図 -13**）．

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <link next="guide.vxml">
  <grammar> 開催案内 </grammar>
 </link>
 <link next="access.vxml">
  <grammar> 交通案内 </grammar>
 </link>
 <link next="register.vxml">
  <grammar> 参加登録 </grammar>
 </link>
 <menu>
  <prompt>
   こちらは○○学会開催案内・参加登録システムです。
   ご希望のメニューをお選び下さい。<enumerate/>
  </prompt>
 ( 以下省略 )
```

図 -12　リンクを用いたユーザ主導発話の捕捉

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml application="top.vxml" version="2.0">
 <form>
  <block>
   開催案内です。○○学会は、平成 15 年 7 月 28 日
   京都△△大学にて開催されます。
  </block>
 ( 以下省略 )
```

図 -13　リーフドキュメントの例

■ユーザからシステムへの自然な主導権の移動

ユーザに要求する発話の種類によっては，まずはユーザに主導権を渡したほうが効率的な場合がある．たとえば，列車の券売機における音声対話インタフェースでは，出発駅・目的駅・席の種類・枚数などを入力する必要があり，システム主導で 1 項目ずつ聞いていては，特にシステムの利用方法に精通したユーザにとっては効率が悪くなってしまう．しかし，すべてのユーザに全項目を 1 発話で入力させるのは，利用頻度が低いユーザには使いにくいものになる．このような場合は，まず明示的にユーザに主導権を渡し，切符購入に必要な情報が不足している場合のみシステムが主導権をとって対話を継続するような形式が望ましい．

VoiceXML の仕様書では，上記のような 1 発話で複数のフィールド変数の値を埋め，埋まっていない変数の値をシステム主導で取得するタイプの対話を混合主導（mixed initiative）と呼んでいる．このような混合主導対話は form 要素を用いて記述する．基本的には図 -6 に示したように，form 要素中に必要な数だけの field 要素を列挙した形式になるが，form レベルの文法を指定することと，form 要素の冒頭に initial 要素を置くことによって，混合主導対話を実現する．initial 要素は以後の

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <form>
  <grammar src="ticket.grxml"
        type="application/srgs+xml"/>
  <initial name="ticket">
   <prompt> ご希望の切符をどうぞ </prompt>
  </initial>
  <field name="from">
   <prompt> 出発駅を入力してください </prompt>
   <grammar src="station.grxml"
         type="application/srgs+xml"/>
  </field>
  <field name="to">
   <prompt> 到着駅を入力してください </prompt>
   <grammar src="station.grxml"
         type="application/srgs+xml"/>
  </field>
  <field name="number" type="number">
   <prompt> 枚数を入力してください </prompt>
  </field>
  <field name="seat" >
   <prompt> 指定席ですか、自由席ですか </prompt>
   <option> 指定席 </option>
   <option> 自由席 </option>
  </field>
  <filled>
   <submit next="http://localhost:8080/ticket"/>
  </filled>
 </form>
</vxml>
```

図 -14　混合主導対話を行う VoiceXML の例

field 要素が混合主導的に埋められることを宣言する役割を持ち，実質的な内容はユーザへのプロンプトを指定する程度である．**図 -14** に混合主導対話による切符購入のための VoiceXML ファイルを示す．

この VoiceXML ファイルに対話が遷移すると，まず initial 要素内のプロンプト（「ご希望の切符をどうぞ」）が出力される．ここでユーザはすべての情報を 1 発話で伝えてもよいし（たとえば「東京から京都まで自由席 2 枚」），部分的に情報を与えてもよい（たとえば，「京都まで 2 枚」）．部分的な情報が与えられた場合は，切符を発券するのに必要な情報を得るためにシステムが主導権をとり，値が埋まっていない field 要素に順次遷移する．「京都まで 2 枚」という発話では変数 to と number の値が埋められたので，出現順でまだ値の埋まっていない変数 from の値を取得するために，9 行目の field 要素に遷移する．この様子を**図 -15** に示す．

このように，VoiceXML では非常に限られた形式ではあるが，混合主導的に対話を進める仕組みを用意している．

ユーザ主導対話の表現

ユーザが対話の主導権を持った方がよいタイプの対話

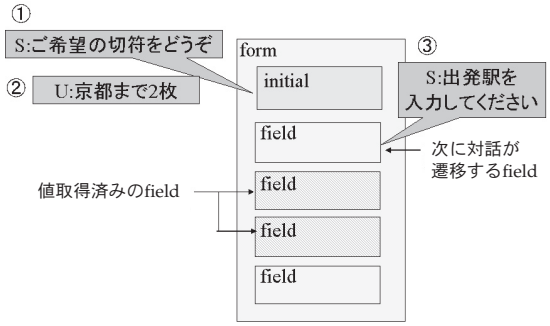

図 -15　混合主導対話の処理例

として，コマンドによる機器制御と質問応答システムが挙げられる．これらの対話の特徴は 1 回または少数回のやりとりで対話が終了することである．ユーザが主導権をとったまま対話を継続するためには，対話文脈の処理が必要になるが，VoiceXML には文脈処理の機構は用意されていない．サーバサイドの処理で実現は可能ではあるが，対話処理は理論と実装のギャップが大きい分野であるので，現状の技術を考えるとユーザ主導対話は 1，2 回のやりとりで終了するようなものになってしまう．

■コマンドを受け付ける対話の記述

コマンドによる機器制御では，エアコンを音声によって操作するような場面を想定するとよい．このタイプの対話はユーザが簡単な発話を行い（たとえば，「温度を下げて」），システムも簡潔な応答（たとえば，「28 度に設定しました」）を返せばよい．このタイプの対話は，家電製品の制御のように，ユーザが何を発話するべきかが比較的分かりやすい場面に適用できる．

VoiceXML でこのようなコマンド制御対話を記述する場合，コマンドの種類が少数に限られている場合には，menu 要素を用い，各々のコマンドに対応する choice 要素に文法を記述する方法が考えられる．**図 -16** のような VoiceXML を用いると，「温度を下げて」，「下げて」，「冷やして」，「暑すぎる」などの発話をすべて温度を下げるというコマンドに対応させることができる．

■質問応答対話の記述

ユーザからの自由な質問を受け付ける質問応答型の対話もユーザ主導で実現する方法が考えられる．観光案内やソフトウェアのヘルプなど，ある程度ドメインを限定しておき，質問タイプとカテゴリー分けされたキーワードをそれぞれフィールド変数としてユーザの質問文を解釈することができる．

市販の製品には，ユーザ発話の多様性に対処するために，統計的言語モデルを用いることができるものもある．

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <menu>
  <prompt timeout="5000s"/>
  <choice next="cooler">
   <grammar type="application/grammar+xml"
         root="cooler">
    <rule id="cooler" scope="public">
     <one-of>
      <item> 温度を下げて </item>
      <item> 下げて </item>
      <item> 冷やして </item>
      <item> 暑すぎる </item>
     </one-of>
    </rule>
   </grammar>
  </choice>
  <choice next="warmer">
 （以下省略）
```

図 -16　コマンド制御対話の例

VoiceXML の仕様には統計的言語モデルの使用は明示されていないが，Nuance 社の Voice Web Server では，コーパスから統計的言語モデルを作成し，認識された文の解釈規則を用いて VoiceXML のフィールド変数に値を割り当てることができる．

通常の文法を用いる方式でも，ある程度の質問解釈は可能である．質問文のタイプおよび質問文に現れる概念を field 変数として混合主導の form を作成する．通常，混合主導対話はすべての field 変数を埋める対話を行うが，各 field 要素の cond 属性を false に指定しておくことで，フォーム解釈アルゴリズムは個々の field 要素を処理することはなく，質問文で言及されたキーワードのみ値が埋まることになる．

図 -17 に質問応答システムの VoiceXML の例を示す．質問入力後の filled 要素において，質問文のタイプが得られているかどうかを if 要素を用いてチェックしている．if 要素は cond 属性に条件文を書き，成立すればその内容を実行する．通常のプログラミング言語と同様の考え方で，else 要素，elseif 要素と組み合わせて使うこともできる．ここでは，質問タイプが得られていない場合は，その他の変数の値を clear 要素で消去する．これによって，initial 要素の変数もクリアされるので，対話は最初の状態に戻り，ユーザに質問再入力を促す．一方，質問タイプが特定できた場合には，そのタイプとその他の変数をサーバへ渡す．ユーザ質問文で入力された値の集合から質問文タイプを決定し，応答を生成するのはサーバサイドの処理になる．

対話システムを完成させるために

ここまでの説明では，部分的な対話を取り上げ，それ

```
<?xml version="1.0" encoding="Shift_JIS"?>
<vxml version="2.0">
 <form>
  <block> こちらは京都観光案内システムです </block>
  <grammar src="kyoto.grxml"
        type="application/srgs+xml"/>
  <initial name="start">
   <prompt> ご質問をどうぞ </prompt>
  </initial>
  <field name="q-type" cond="false"/>
  <field name="place" cond="false"/>
  <field name="date" cond="false"/>
  <field name="transport" cond="false"/>
  <block>
   <if cond="qtype==undefined">
    <prompt>
     質問が理解できませんでした
    </prompt>
    <clear/>
   <else/>
    <submit next="http://localhost:8080/qa" />
   </if>
  </block>
 </form>
<vxml>
```

図 -17　質問入力を行う VoiceXML の例

らを VoiceXML で記述する方法を説明してきた．しかし，メニューと情報提供を組み合わせた単純な対話システムを除いては，データベース検索や更新などのサーバサイドでの処理が必要になり，サーバサイドプログラムでの対話管理も記述する必要がある．たとえば図 -6 の参加登録を行うページでは，定員オーバの場合の処理や，会員名簿との不一致があった場合の処理などはすべてサーバサイドプログラムに記述される．これらの処理がサーバサイドプログラムに埋め込まれてしまうと，対話の流れが見えにくくなり，保守・更新の労力が大きくなる．

このような問題に対しては，Web アプリケーション開発のフレームワークを用いて対処する方法が考えられる．Jakarta プロジェクトからリリースされているフレームワーク Struts[☆3] は，モデル・ビュー（音声対話の場合は VoiceXML）・コントローラを分離した Web アプリケーション開発を可能にするものである．コントローラはアプリケーションロジックプログラムから戻される処理結果に応じて，クライアントに返す VoiceXML を変更でき，その対応関係はコンフィグレーションファイルに XML 形式で記述する．すなわち，バックエンドアプリケーションを直接操作するプログラムとその処理結果によってどのような対話状態に分岐すべきかという情報が分離されている．

コントローラに返す処理結果は「登録成功」「登録失敗」

☆3 http://jakarta.apache.org/struts/

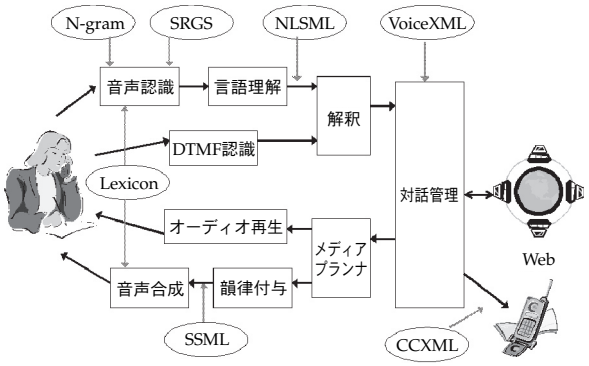

図 -18　W3C の音声インタフェースフレームワーク
（文献 3）より引用，翻訳および一部改変）

などのステータスであるので，コンフィグレーションファイルによる対話管理は再利用性が高くなる．すなわち，データベースが変更されても，おおまかなアプリケーションの流れが共通していれば，アプリケーションロジック部分と VoiceXML に関して若干の変更を行うだけで他のタスク・ドメインに応用できることが期待できる．

VoiceXML の現在と将来

現在，日本語が利用可能な VoiceXML 処理系はオムロン（Nuance 社の Voice Web Server の日本語版）や日本 IBM などから市販されている．また，Linux で稼働するフリーの処理系は IPA の擬人化音声対話エージェント基本ソフトウェアプロジェクトの成果物として公開されている☆4．

VoiceXML に関連する仕様は，W3C の Voice Browser Working Group で作業が継続しており，完成度があまり高くない仕様や，しばらく更新のない仕様も存在する．**図 -18** に W3C の音声インタフェースフレームワークを示す．

四角で示したものが音声対話システムの構成要素であり，楕円形で示したものが各構成要素が利用する知識表現形式またはその入出力形式を定めた仕様である．以下，W3C で策定が進められている VoiceXML 以外の音声インタフェースに関連する各種仕様について説明する．

・SRGS（Speech Recognition Grammar Specification）[2)]

音声認識に用いる文法記述方式を定めたもので，XML 形式が標準であるが，それと等価な ABNF（拡張 BNF）形式も同時に定められている．本稿では図 -8 にその記述例を示した．

・N-gram（Stochastic Language Models（N-Gram）Specification）☆5

統計的言語モデルのための各種知識を記述するためのものであるが，2001 年 1 月に出された Working Draft で更新が止まっている．

・NLSML（Natural Language Semantics Markup Language）☆6

言語理解結果の表現の仕様である．確信度や認識結果の表記法など意味の周辺情報は定めているが，意味に関する部分は XForms☆7 の仕様を参照するにとどまっている．こちらも 2000 年 11 月に出された Working Draft で更新がとどまっている．なお，SISR（Semantic Interpretation for Speech Recognition）☆8 は，SRGS に埋め込んだタグを利用して ECMAScript☆9（JavaScript の標準化版）のオブジェクトを生成する方法を規定しているものであるが，こちらは最近も更新されており（2003 年 4 月 Working Draft），VoiceXML2.0 の仕様書から判断しても言語処理の出力は SISR に移行していると考えてもよさそうである．

・CCXML（Call Control eXtensible Markup Language）☆10

基本的に転送しかできない VoiceXML の電話機能を拡張するもので，多人数電話会議などをサポートするための記述言語．

・SSML（Speech Synthesis Markup Language）☆11

音声合成のために，テキストに読み方や韻律情報を付与するためのマークアップ言語．強調を指定する emphasis 要素や高さ・速さ・大きさを調整できる prosody 要素などがある．

VoiceXML に関しては，次期バージョン 2.1 の Working Draft が 2003 年秋に公開される予定である．2004 年から始まるメジャーバージョンアップでは他の言語と統合しやすいようにモジュール化されるなど，W3C の Multimodal Interaction Activity☆12 の活動の影響を受けて改定される予定である．

☆4 http://hil.t.u-tokyo.ac.jp/~galatea/
☆5 http://www.w3.org/TR/ngram-spec/
☆6 http://www.w3.org/TR/nl-spec/
☆7 http://www.w3.org/TR/xforms-datamodel/
☆8 http://www.w3.org/TR/semantic-interpretation/
☆9 http://www.ecma-international.org/publications/standards/ECMA-262.HTM
☆10 http://www.w3.org/TR/ccxml/
☆11 http://www.w3.org/TR/speech-synthesis/
☆12 http://www.w3.org/2002/mmi/

参考文献
1）McGlashan, S. et al. (Eds.): Voice Extensible Markup Language（VoiceXML）Version 2.0（2003）. http://www.w3.org/TR/voicexml20/
2）Hunt, A. and McGlashan, S.（Eds.）: Speech Recognition Grammar Specification Version 1.0（2003）. http://www.w3.org/TR/speech-grammar/
3）Larson, J. A.: VOICEXML Introduction to Developing Speech Applications, Pearson Education（2003）.

（平成 15 年 8 月 28 日受付）