音声対話記述言語 VoiceXML と MMI 記述言語 XISL の比較

桂田浩一*　中村有作*　山田真*　小林聡　山田博文***　新田恒雄***

*豊橋技術科学大学　大学院工学研究科　知識情報工学専攻
**豊橋技術科学大学　情報処理センター
***豊橋技術科学大学　マルチメディアセンター

〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘１－１
Email: katurada@tutkie.tut.ac.jp

あらまし：　本報告では音声対話記述言語 VoiceXML とマルチモーダル対話記述言語 XISL を比較し，それぞれの特徴を述べる．VoiceXML は電話による音声対話を対象とした対話記述言語であり, ユーザの発話文法や対話の制御など, 音声 Web サービスで必要な様々な事項を記述できる．一方, 利用可能なモダリティは音声と DTMF に限られており, モダリティを拡張するには言語仕様の変更が必要であることなど，マルチモーダルへの対応が難しい．我々はこれまでマルチモーダル対話記述言語 XISL を検討してきた．XISL は，１）利用モダリティの拡張が容易で，言語仕様の変更が必要ない，２）インタラクション記述のみを XML コンテンツから分離して記述するため，双方の再利用性が高い，といった特徴を持つ．VoiceXML と XISL を比較したところ上記の違いの他にも，音声中心の VoiceXML ではシステム主導の対話が書き易く，対話画面を標準的に想定した XISL ではユーザ主導の対話が書き易いといった特徴が見られた．

キーワード：　XML，XISL，マルチモーダル，対話記述言語

Comparison between VoiceXML and XISL

Kouichi KATSURADA*, Yusaku NAKAMURA*, Makoto YAMADA*,
Satoshi KOBAYASHI**, Hirobumi YAMADA*** and Tsuneo NITTA*

*Graduate School of Technology, Toyohashi Univ. of Technology
**Computer Center, Toyohashi Univ. of Technology
***Multimedia Center, Toyohashi Univ. of Technology

1-1 Hibarigaoka, Tempaku-cho, Toyohashi 441-8580, JAPAN
Email: katurada@tutkie.tut.ac.jp

Abstract : In this paper, we compare the difference between voice interaction description language VoiceXML and multimodal interaction description language XISL, and clarify their characteristics. VoiceXML enables us to describe speech grammar, dialogue control, and some other elements that are required for web services by telephone, however, its target modalities are limited only to speech and DTMF. Moreover, when system developers need additional modalities, they can hardly change the language specification. Thus, in VoiceXML, it is difficult to deal with multimodalities. From this background, we have developed a multimodal interaction description language XISL. XISL has such merits as: 1) it can expand modalities available without changing its specification, 2) it enables us to reuse both XML contents and XISL interactions independently.

Key words : XML, XISL, multimodal interaction, interaction description language

1. はじめに

マルチメディア技術の発展とともに,「キーボード, マウス, およびディスプレイ」以外の入出力手段を用いて Web アクセスを行う方法が検討され始めている. 特に音声によるアクセスは, 試験的サービスが実際に始まるなど[1], 今後急速な普及が予測される. こうしたインタラクティブな Web アクセスを提供するには, インタラクション(ユーザとシステムのやり取り)を明記する言語が必要となる. WWW コンソーシアム(W3C)では, 最初の試みとして, VoiceXML という音声中心のインタラクション記述言語が検討されてきた[2].

VoiceXML は, 電話を用いた音声対話を記述する言語で, ユーザに示すプロンプト, ユーザの発話文法, 対話シナリオの制御など, 対話に必要な様々な事項を記述できる. しかしながら VoiceXML は電話を対象としているため, 利用可能なモダリティが音声および DTMF(プッシュボタン)に限られている. 今後, 電話に限らず様々な端末を用いた Web アクセスが行われるようになると, マルチモーダル対話を記述する必要が生じるが, 現在の VoiceXML では限界がある. また VoiceXML では XML コンテンツと対話内容が混在して記述されるため, XML コンテンツおよび対話内容の再利用が難しい.

これに対して, 我々はマルチモーダル対話記述言語 XISL(eXtensible Interaction Sheet Language)を検討してきた[3][4]. XISL では端末の種類に依存するマルチモーダル入出力の詳細を言語仕様から切り離しているため, 入出力モダリティを任意に拡張することが可能である. また XISL にはインタラクションのみを記述し, XML コンテンツを分離しているため, XML コンテンツとインタラクション双方の再利用性が高く, これらの組み合わせによりシームレスな Web サービスを提供し易い利点を持つ.

本稿ではまず VoiceXML, 続いて XISL についてそれぞれ特徴を述べた後, 双方の比較を試みる.

2. VoiceXML

VoiceXML は W3C で仕様策定が進んでいる音声対話記述言語であり, 現在はバージョン 1.0 が, また近いうちにバージョン 2.0 が勧告される予定となっている. 最近では VoiceXML を用いた音声ポータルの試用システム[1]も提供され始めており, 今後の普及が最も見込まれる音声対話記述言語といえる. 本節では VoiceXML の概略と特徴を述べる.

2.1. VoiceXML の概略

VoiceXML 文書の例を図 1 に示す. VoiceXML のルートタグは<vxml>(図 1(A))であり, 内部に<form>(図 1(C)), <menu>, <meta>(図 1(B))といった要素を持つ. <form>には1単位の対話シナリオが, <menu>にはユーザの選択を伴う対話が記述される. また

```
<?xml  version="1.0"  encoding="Shift-JIS"?>
<vxml  version="1.0"  application="app-root.vxml">  ---------- (A)
  <meta  name="burger-order">  ---------- (B)
  <form>  ---------- (C)
    <initial>  ---------- (D)
      <prompt>
        いらっしゃいませ，ご注文は何になさいますか
      </prompt>
    </initial>
    <field  name="burger">  ---------- (E)
      <prompt>  ---------- (F)
        バーガーを選んでください
      </prompt>
      <grammar>  ---------- (G)
        ハンバーガー｜チーズバーガー
      </grammar>
    </field>
    <field  name="number">
      <prompt>
        お幾つご注文ですか
      </prompt>
      <grammar>
        一つ｜二つ｜三つ
      </grammar>
    </field>
    <filled  mode="all">  ---------- (H)
      <value expr="burger">を<value expr="number">ですね  (I)
      <if  cond="number == ‘ひとつ'">  ---------- (J)
        <assign  name="Qty"  expr="1"/>  ---------- (K)
        <elseif  cond="number == ‘ふたつ'"/>
                          :
      </if>
      <if  cond="burger =='ハンバーガー'">
        <assign  name="HBG"  expr="HBG + Qty"/>
                          :
      </if>
      <goto next="#payment"/>  ---------- (L)
    </filled>
  </form>
</vxml>
```

図 1 VoiceXML 文書の記述例

<meta>には文書に関する情報が記述される.

<form>要素は幾つかの<field>要素（図1(E)），<initial>要素（図1(D)），<filled>要素（図1(H)）等から構成される．このうち<field>要素には１ターンのシステムとユーザのやり取りが記述される．<field>要素内の<prompt>（図1(F)）にはユーザの入力を促すためのプロンプトが記述され，ユーザは<grammar>（図1(G)）で指定される入力文法に従って発話する．<field>要素内に<filled>要素があれば，ユーザの入力を確定後にその内容が実行される．

一般的に，<form>要素の内部は上から順に実行される（対話が進行する）が，図1のように内部に<initial>要素を持つ場合には“混合主導対話”となり，内部の<field>が任意の順番で実行可能となる．このとき複数の<field>の同時入力も可能となる．例えば図1のVoiceXMLでは次のような対話進行が考えられる．

対話例（１）
S：いらっしゃいませ，ご注文は何になさいますか？
U：ハンバーガーをください
S：お幾つご注文ですか
U：一つください
S：ハンバーガーを一つですね
：

対話例（２）
S：いらっしゃいませ，ご注文は何になさいますか？
U：ハンバーガーを一つください．
S：ハンバーガーを一つですね
：

上の対話例（１）では二つの<field>が逐次満たされ，（２）では同時に満たされている．<field>が満たされた後，図1のVoiceXMLでは<filled>要素（図1(H)）が実行される．図1のように<filled>要素が<form>（図1(I)）の子要素として記述されるとき，<filled>の属性modeによって<filled>要素実行の契機となる<field>の数を規定できる．属性modeの値が”all”のとき，全ての<field>が満たされなければ<filled>は実行されない．値が”any”のとき，一つ以上の<field>が満たされれば<filled>が実行される．

<filled>要素内では，図1のVoiceXMLの場合，<if>要素（図1(J)）による入力内容に応じた条件分岐と，<assign>要素（図1(K)）による変数への代入，<goto>要素（図1(L)）による次対話への遷移等が実行される．

2.2. VoiceXMLの特徴

VoiceXMLでは大域変数の定義や，任意の時点で入力可能な対話を記述するために，「アプリケーションルート」という文書の概念を導入している．アプリケーションルートは図1の(A)のように<vxml>タグの属性applicationで指定される．ここで，図1のVoiceXML文書に対するアプリケーションルートを図2に示す．図2の(M)では<form>要素の属性scopeの値が”document”に指定されている．アプリケーションルートで属性scopeの値が”document”のとき，その<form>の<grammar>はアプリケーション中の任意の時点で有効となる．一方，アプリケーションルート以外の<form>で属性scopeが定義され，そ

app-root.vxml

```
<?xml  version="1.0"  encoding="Shift-JIS"?>
<vxml  version="1.0">
  <meta  name="burger-root">
  <form  scope="document">  ---------- (M)
    <grammar>
      ヘルプ
    </grammar>
    <initial>
      <prompt>
        何でしょうか？
      </prompt>
    </initial>
    <field  name="usage">
      <grammar>
        使い方が分かりません
      </grammar>
        :
    </field>
    <field  name="call_staff">
      <grammar>
        店員さんを呼んでください
      </grammar>
        :
    </field>
      :
  </form>
</vxml>
```

図 2　VoiceXMLのアプリケーションルート

の値が”document”のとき，<grammar>はその文書内で有効となる．また属性 scope の値が”dialog”のとき，<grammar>は<form>内で，有効となる．このように VoiceXML では<grammar>のスコープが様々なレベルで設定できるため，図 2 のような大域的な割り込み対話や，局所的な対話などが記述可能である．

またVoiceXMLは<prompt>タグを用いたシステム主導の対話が書き易く設計されている．システム主導とは，1）システムがユーザに入力の催促や質問を行い，2）それに対してユーザが答える，といった形式の対話をいう．VoiceXML ではまず<prompt>をユーザに示し，ユーザが<grammar>に従って入力し，それに対してシステムが<filled>内部を実行する，という対話形式を基本としている．したがってシステムがユーザの情報収集を行うような対話は書き易い．

3. XISL

我々はマルチモーダルを利用した Web アクセスを実現するために，マルチモーダル対話記述言語 XISL を検討してきた[3][4]．以下，XISL の概略と特徴を述べる．

3.1. XISL の概略

XISL 文書の例を図 3 に示す．ルートタグである<xisl>要素（図 3(a)）は，<head>要素（図 3(b)：当該文書に関する情報）と<body>要素（図 3(c)：文書の内容）から構成される．<body> 中 の <dialog> 要 素 （ 図 3(d) ） は VoiceXML の<form>，<menu>に相当するもので，1単位の対話シナリオを表す．<dialog>要素は，対話の導入処理を行う<begin>（図 3(e)），1ターンの対話を記述する<exchange>（図 3(f)），対話の終了処理を行う<end>の各要素から構成される．このうち<exchange>は VoiceXML の<field>要素に相当する．

<exchange>は，ユーザの1単位の入力を記述する<operation>（図 3(g)）とシステムの1単位の動作を記述する<action>（図 3(i)）からなる．<operation>要素内にはユーザの1入力を記述する<input>（図 3(h)）を，<action>要素内にはシステムの1出力を記述する<output>（図 3(k)）およびその他の動作に関するタグを記述する．

<action>要素，<begin>要素，および<end>要素には，XML 文書から要素（属性値）を取り出す<get_value>（図 3(j)）（<get_attr>）や，XML 文書に要素（属性値）を書き込む<set_value>（図 3(n)）（<set_attr>），演算や代入を行う<assign>（図 3(m)），条件分岐や繰

```
<?xml  version="1.0"  encoding="Shift-JIS"?>
<!DOCTYPE  xisl  SYSTEM  "xisl.dtd">
<xisl  application="app-root.xisl>  ---------- (a)
  <head> …… </head>  ---------- (b)
  <body>  ---------- (c)
    <dialog  id="order">  ---------- (d)
      <begin>  ---------- (e)
        <output  type="window"  event="navigate">
          <![CDATA[  <param  name="path">
            hamburger.xml
          </param>  ]]>
        </output>
        <output  type="agent"  event="speech">
          <![CDATA[  <param  name="speech_text">
          いらっしゃいませ，ご注文は何になさいますか
          </param>  ]]>
        </output>
      </begin>
      <exchange>  ---------- (f)
        <operation  comb="par">  ---------- (g)
          <input  type="touch"  event="click"  ---------- (h)
                  match="item[burger:=`id]/fig"/>
          <input  type="speech"
                  event="(一つ|二つ|三つ)"
                  match="hamburger"  namelist="number"/>
        </operation>
        <action  comb="par">  ---------- (i)
          <get_value  target"ham.xml"  var="burg_name"
                      match="item[@id=$burger]/name">  -- (j)
          <output  type="agent"  event="speech">  ---------- (k)
                  <![CDATA[  <param  name="speech_text">
                    $burg_name を$number ですね
                  </param>  ]]>
          </output>
          <switch  var="$number">  ---------- (l)
            <case  value="一つ">
              <assign  var="Qty"  expr="1"/>  ---------- (m)
            </case>
              :
          </switch>
          <switch  var="$burger">
            <case  value="ham">
              <assign  var="HBG"  expr="$HBG+$Qty"/>
              <set_value  target="hamburger.xml" var="$HBG"
                      match="item[@id='ham']/quantity"/>  - (n)
            </case>
              :
          </switch>
            :
        <goto  next="payment">  ---------- (o)
        </action>
      </exchange>
        :
    </dialog>
  </body>
</xisl>
```

図 3　XISL 文書の記述例

り返し命令を行う<if>，<switch>（図 3(l)），<while>等が記述できる．

XISL の<exchange>，<input>，および<output>は，それぞれ同時並行的，逐次的，択一的（<exchange>，<input>のみ）な実行（もしくは入力受付）が可能である．この制御のために<dialog>，<operation>，<action>では属性 comb（属性値は par（同時並行），seq（逐次），alt（択一））を用意し，要素内部の対話進行を制御している．こうした制御は<seq_input>，<alt_exchange>（図 4(p)）といった制御タグによって明示することも可能である．さらに XISL では<call>，<goto>（図 3(o)）タグを利用することにより，ある<dialog>から次の<dialog>に対話を遷移することができる．<call>の場合，遷移先の<dialog>中の<return>タグによって呼び出し元に戻るのに対し，<goto>で遷移した対話は元に戻らない．

XISL 文書では VoiceXML と同様に，アプリケーションルートを指定できる．アプリケーションルートは VoiceXML と同じく文書の冒頭（図 3(a)）で指定する．図 3 の XISL に対するアプリケーションルートを図 4 に示す．

app-root.xisl

```
<?xml  version="1.0"  encoding="Shift-JIS"?>
<!DOCTYPE  xisl  SYSTEM  "xisl.dtd">
<xisl>
  <head> …… </head>
  <body>
    <dialog  id="help"  scope="document">
      <exchange>
        <operation>
          <input  type="speech"
                  event="へるぷ"  match="/"/>
        </operation>
        <action>
          <output  type="agent"  event="speech">
            <![CDATA[  <param  name="speech_text">
            何でしょうか？
            </param>  ]]>
          </output>
        </action>
      </exchange>
      <alt_exchange>  ---------- (p)
        <exchange>
          <operation>
            <input  type="speech"
                    event="使い方が分かりません"
                    match="/"/>
          </operation>
          <action>
              :
          </action>
        </exchange>
        <exchange>
          <operation>
            <input  type="speech"
                    event="店員さんを呼んでください"
                    match="/"/>
          </operation>
          <action>
              :
          </action>
        </exchange>
            :
      </alt_exchange>
    </dialog>
  </body>
</xisl>
```

図 4　XISL のアプリケーションルート

3.2.　XISL の特徴

XISL はインタラクションのみを記述する言語であり，XML コンテンツとは分離して記述する．インタラクションは XML コンテンツ中のオブジェクトに対するものとして記述する．例えば図 3(h)の<input>タグの属性 match には，対応する図 5 の XML コンテンツの(q)を指すパスが記述されている．つまり(q)のボタンが押され（ボタンは XSL によって画面に表示されているとする），同時に「一つ」といった発話があった場合に，アクション(i)が実行される．

```
<?xml  version="1.0"  encoding="Shift-JIS"?>
<?xml-stylesheet  type="text/xsl"  href="ham.xsl"?>
<!DOCTYPE  hamburger  SYSTEM  "ham.dtd">
<hamburger>
  <title> Select items. </title>
  <p> May I help you? </p>
    <item id="ham">
      <name>  ハンバーガー  </name>
      <fig> hamburger.gif </fig>  ---------- (q)
      <price> 65 </price>
      <quantity> 0 </quantity>
    </item>
    <item id="cheese">
      <name>  チーズバーガー  </name>
      <fig> cheeseburger.gif </fig>
      <price> 80 </price>
      <quantity> 0 </quantity>
    </item>
            :
    <item>
      <name>  トータル  </name>
      <price id="total"> 0 </price>
      <quantity id="total"> </quantity>
    </item>
  <p> O.K.? </p>
  <button> [yes] </button>
</hamburger>
```

図 5　XML コンテンツ

また前節で説明したように，XISL では入出力に関する記述を<input>および <output>要素で記述する. モダリティはこれらの要素中の属性 type で指定し，入出力内容は属性 event を用いて記述するが，これら属性の値域は XISL では規定していない．したがって端末の種類に応じて入出力の仕様を容易に変更でき, XISL の言語仕様を変更することなくモダリティを拡張することができる.

4. VoiceXML と XISL の比較

VoiceXML と XISL は共に対話記述言語という点で共通点を持つ. また XISL のアプリケーションルートやその他多くのタグは VoiceXML との整合性を考慮して導入したため, 機能的に同一なものが多い. しかしながら幾つかの点で相違があるので, 本節ではこれを明らかにする.

まず最も大きな相違点として, VoiceXML は音声と DTMF という二つのモダリティのみを対象としているのに対し，XISL では任意のモダリティを対象としている. またモダリティの追加も XISL では<input>, <output>タグ内部の仕様を変更するだけでよく, 言語仕様の変更は必要ないという点が挙げられる．XISL は VoiceXML に比べてマルチモーダルの取り扱いという点に関して柔軟性があると言え, 様々なモダリティを利用した Web アクセスが実現し易い.

次に, VoiceXML では文書中にコンテンツとインタラクション記述が混在しているため, 同じコンテンツに対して複数種類の端末からアクセスすることは難しい．これに対して XISL はインタラクションのみを記述する文書であり，XML コンテンツは最初から分離されている．これにより, 端末毎に XISL 文書を準備すれば多種多様な端末による同一コンテンツへのアクセスが可能となり，XML コンテンツやインタラクションの再利用性を高めることができる.

また XISL では対話シナリオ制御を par, alt, seq といった属性や, 対応するタグで統一的に取り扱うのに対し, VoiceXML では seq はシステム主導の<form>, par は混合主導の<form>, alt は<menu>で実現する必要があり，制御の複雑な入れ子を記述するのは困難である．XISL のように統一的に取り扱うことで，マルチモーダル入出力の対話制御が容易に記述できる.

最後に対話のスタイルに関しては，2.2 節で述べたように VoiceXML はシステム主導の対話が記述しやすいという特徴を持つ．一方 XISL は基本的に <exchange> の内部が <operation>，<action>の順になっており，ユーザの入力に対するシステムの動作を記述するという形式になっている. したがってユーザの自発的な入力, すなわちユーザ主導が書きやすいという特徴を持つ. システム主導・ユーザ主導の使い易さはシステムの用途や状況によって異なるため, どちらの言語が適しているかは状況によって異なる.

5. まとめ

本稿では音声対話記述言語 VoiceXML とマルチモーダル対話記述言語 XISL を比較した．XISL はマルチモーダルを柔軟に取り扱え，XML コンテンツとの分離を実現しており，さらに対話進行の制御が容易に記述できるという特徴を持つ．このように，XISL はマルチモーダル対話を記述する上で必要な様々な事項を考慮に設計された言語であるといえる.

今後の課題としては，VoiceXML からの migration を考慮した XISL の改良，W3C のマルチモーダル記述 WG への提案，および XISL を用いたアプリケーションの開発などがある.

参考文献

[1] http://www.voizi.net/

[2] http://www.w3.org/TR/voicexml/, http://www.voicexml.org/

[3] 小林聡，中村有作，桂田浩一，山田博文，新田恒雄:“ マルチモーダル対話記述言語 XISL の提案 ”, 情報処理学会研究報告 2001-SLP-37, pp.43-48，2001.

[4] T. Nitta, K. Katsurada, H. Yamada, Y. Nakamura, S. Kobayashi: “XISL: An Attempt to Separate Multimodal Interaction from XML Contents”, Proc. of Eurospeech 2001, pp.1197-1200, 2001.