ヒューマンインタフェース 97−9
音声言語情報処理 40−9
（２００２．２．１）

多様な端末からのアクセスを可能にする MMI アーキテクチャ

桂田浩一*　大谷 佳彦*　中村有作*　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，端末の拡張性が高いマルチモーダル対話（MMI）システムの汎用アーキテクチャを提案する．本アーキテクチャでは，MMI の対話シナリオを格納するドキュメントサーバ，対話進行を管理する対話制御部，入出力端末を管理するフロントエンドの 3 つのモジュールを明確に分割してシステムを構築する．また，対話シナリオの記述に用いる言語として，入出力に関する記述を自由に規定できる XISL を用いる．これにより，端末を拡張する際には，端末依存のフロントエンドの実装と入出力記述の変更のみで対応できるようになるため，新たな対話記述言語やその処理系を導入することなく，容易に新規端末を導入できるようになる．本アーキテクチャに基づき，試作中の MMI オンラインショッピングシステムについて概略を説明する．

キーワード：　マルチモーダル対話，システムアーキテクチャ，XISL，対話制御

MMI System Architecture for Terminals with Various Types of Modalities

Kouichi KATSURADA*, Yoshihiko Ootani*, Yusaku NAKAMURA*,
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：This paper presents an MMI system architecture for terminals with various types of modalities. In the architecture, an MMI system is divided into three modules: a document-server module that holds dialogue scenarios, a dialogue-control module that controls dialogue flows, and a front-end module that manages user's inputs and system's outputs. This modularity enables us to easily expand input/output modalities of terminals by only adding the modality to the front-end module. Moreover, we use the language XISL to describe dialogue scenarios. Since XISL has the flexibility to modify input/output description, including the addition of new modalities, we can cope with the change of user interface without introducing a new set of description language and its processor. An online shopping system designed by this MMI system architecture is also described.

Key words：Multimodal Interaction, System Architecture, XISL, Dialogue Flow Control

1. はじめに

マルチモーダル対話（MMI）を用いて Web コンテンツにアクセスする, 様々な取組みが始まっている．WWW コンソーシアム（W3C）では, これまで MMI の要求仕様[1]を検討してきたが, 近日中にワーキンググループが結成され，本格的な議論が始まることになっている．また別の動きとして，MMI シナリオ記述言語 SALT を検討するフォーラム[2]が結成され，MMI を本格的に利用するための基盤が整備されつつある．さらに，音声対話記述言語 VoiceXML を拡張し，MMI の利用を可能にする試み[3]も始まっている．

MMI はモノモーダルインタラクションの拡張と捉えることができる．すなわち, 単一モダリティを利用するインタラクション（例えばキーボードのみを用いたインタラクション）に何らかの新規モダリティを追加したものと捉えられる．現状では, 追加するモダリティとして音声などが考えられるが, 今後の技術の発展によっては, 様々なモダリティが追加される可能性がある．（例えばジェスチャ認識の発展により, 手話がモダリティとして追加される可能性もある．）したがって，モダリティの拡張を予め考慮に入れた, 柔軟性の高いシステムを構築する枠組みが必要と考えられる．

我々はモダリティに対する拡張性の高い MMI システムアーキテクチャ[4]を提案している．このアーキテクチャでは，MMI システムをドキュメントサーバ（対話シナリオの保持），対話制御部（対話シナリオの進行），およびフロントエンド（入出力を管理する端末）の三つのモジュールに分割して実装する．新たな端末を導入する際には, フロントエンドのみを実装すればよいため, 端末の拡張が容易なモジュール構成になっている．また対話シナリオを記述する言語に, 入出力の記述に関して高い自由度を持つ XISL[5][6][7]を用いており，フロントエンドの拡張性を高めている．これらモジュール分割と対話記述言語により, 端末側の拡張性が高い MMI システムが構築可能になる．以下では，MMI システムのアーキテクチャについて述べたあと，対話シナリオ記述言語 XISL を概説し，最後にオンラインショッピングを対象としたアプリケーションの実装例を示す．

2. MMI システムアーキテクチャ

2.1. アーキテクチャの設計

本研究では, 柔軟性の高い MMI システムを実現するために, 次の条件を満たすようアーキテクチャを設計した．

- PC に限らず，携帯電話やデジタル TV など，多様な端末が導入できる．
- 端末を導入する際に, アーキテクチャを変更しなくてよい．
- 対話シナリオの変更によって, 様々なアプリケーションを実現しうる．

上記の条件を満たすには，1）端末に依存するモジュールと依存しないモジュールを独立させること，2）対話シナリオに依存する部分と依存しない部分を分離することが必要である．図 1 に今回提案する, 以上の条件を満たしたアーキテクチャを示す．

ドキュメントサーバは対話シナリオを格納するモジュールである．様々なアプリケーションに関する対話シナリオを準備することで, 多様なサービスを提供することができる．対話制御部は, 対話シナリオに従って対話を進行させるモジュールで, 端末と対話シナリオの双方から独立である．フロントエンドはユーザの入出力を受け付けるモジュールで, 端末の種類ごとに用意される．三つのモジュールを独立させ,

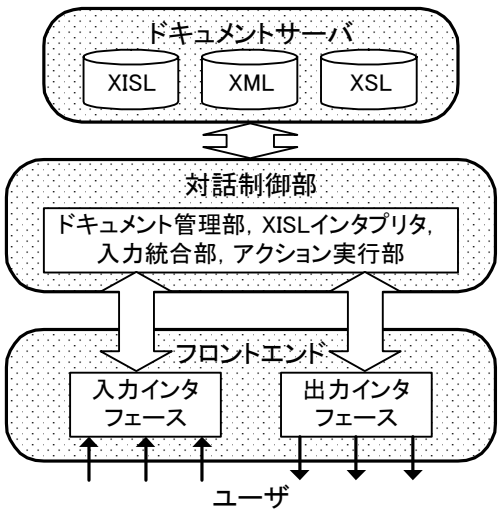

図 1：MMI システムアーキテクチャ

モジュール間通信を行なうことにより, 柔軟性の高いアーキテクチャを実現している．以下, XISL の概略を述べたあと，各モジュールの概要を述べる．

2.2. MMI 記述言語 XISL

XISL は MMI 記述言語であり，ユーザとシステムとの対話シナリオを記述する. 原則として, ユーザ入力とシステム動作の組みを対話の 1 ユニットとして記述し, そのシーケンスからシナリオを構成する．システム動作としては, ユーザへの応答, 簡単な演算, ユーザ入力の内容による条件分岐を記述できる. 言語の詳細は 3.2 節で説明する．

2.3. 対話制御部

対話制御部は，ドキュメントサーバから XISL を含む文書を受け取り, それらを解釈し, 対話の進行を制御する. 図 2 に示すように，ドキュメント管理部，XISL インタプリタ，入力統合部, アクション実行部の各サブモジュールから構成されている．

ドキュメント管理部は, フロントエンドやアクション実行部からのリクエストに基づき, ドキュメントサーバからファイルを取得し, リクエスト元に送信する．

インタプリタは, 起動命令, もしくは対話切替の命令を受け取ると, ドキュメント管理部に XISL ドキュメントをリクエストする．ドキュメントを受け取ると，XISL の書式をチェックするとともに, ユーザ入力に関する記述（入力記述），およびシステム動作に関する記述（動作記述）をそれぞれ抽出して，入力統合部，アクション実行部へ送信する．

入力統合部は, まずインタプリタから送信された入力記述に基づいて, 受付可能な入力シーケンスを抽出する. 続いてそのシーケンスを文法の一種と見なして, GLR テーブルを作成し, 同時に, 入力記述をフロントエンドに送信する. フロントエンドからはユーザ入力が順次送信されるが，GLR テーブルを用いて，その系列が文法に合致するかどうかをチェックする. 結果が一纏まりの入力として還元できる場合には，それをアクション実行部に送信する．

アクション実行部は, システム動作に関する記述をインタプリタから受け取り, 入力統合部から送信される入力に従って，応答を行なう．出力に無関係な動作（XML の編集や対話切替など）はアクション実行部で直接実行され, 必要であれば他のサブモジュールと情報を遣り取りする．

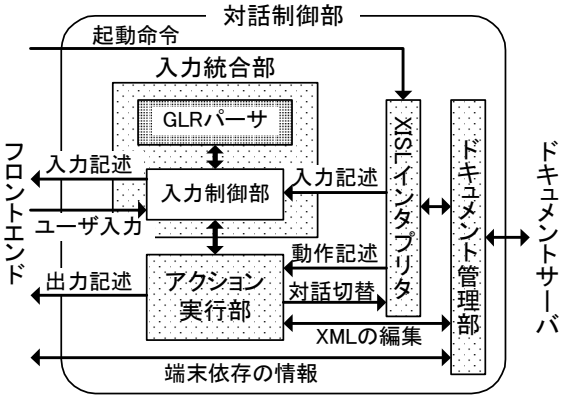

図 2：対話制御部

ここで注意されたいのは, 入力統合部とアクション実行部では, 入出力に関する個々の記述を解釈していない点である．入力に関しては, フロントエンドから送信される入力とインタプリタから送信される入力記述とのマッチングのみを行ない, 出力に関しては内容をそのままフロントエンドに送信している. これによって, 端末依存の処理をフロントエンドに集約でき, 対話制御部を端末非依存に実装することが可能になる．

2.4. フロントエンド

フロントエンドでは, 起動命令を送信した後に, 受付可能な入力を対話制御部から受け取り, その内容を解釈する．この際, 追加的に必要な文書（例えば, 音声認識における文法ファイルなど）があれば, 対話制御部にリクエストする. ユーザからの入力がある都度それを解釈し, 入力内容を対話制御部に送信する. 出力に関しては，その内容を解釈した後，必要な文書（表示のためのファイルやオーディオファイルなど）があれば，対話制御部にリクエストし，データが揃った時点で応答を行なう．

2.5. ドキュメントサーバ

ドキュメントサーバは，XISL，XML をはじめとする文書を格納するモジュールであり, 一般的な WWW サーバを想定している．ドキュ

メントサーバは, 対話制御部からのリクエストに応じて文書を送信する. またリクエストに応じてサーバサイドスクリプトを実行し, その結果を返信する.

3. XISL と入出力記述

3.1. XISL と入出力記述の仕様

前節でも述べたように, XISL にはユーザとシステムとの対話シナリオを記述する. 対話シナリオには, 入出力と対話制御の記述が含まれるが, このうち入出力に関してはフロントエンドの種類によってモダリティが異なる場合(例えば PC ではマウスとキーボード, 携帯電話では音声) があり, また将来的な端末の拡張や変更のために柔軟性を持たせるべきである. そこで入出力に関する仕様は XISL で詳細を決定せず, フロントエンドの開発者が自由に仕様を策定できるようにした. 次節では, XISL の概要を述べ, 続いてフロントエンドとして PC を利用した場合を例に, 入出力記述の仕様を説明する.

```
<?xml  version="1.0"  encoding="Shift-JIS"?>
<!DOCTYPE  xisl  SYSTEM  "xisl.dtd">
<xisl  version="1.0">  ---------- (a)
  <head> ····· </head>  ---------- (b)
  <body>  ---------- (c)
    <dialog  id="order">  ---------- (d)
      <begin>  ---------- (e)
        <output  type="mmi-browser"  event="open">
          <![CDATA[
            <param  name="window-name"> top-page </param>
            <param  name="uri"> ./hamburger.xml </param>
          ]]>
        </output>
        <output  type="speech"  event="tts-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="recognize"
                target="count.grxml"
                match="/grammar/rule[@id=count]"
                return="number"/>
        </operation>
        <action  comb="par">  ---------- (i)
          <get_value  target="ham.xml"  var="burg_name"
                match="item[@id=$burger]/name">  -- (j)
          <output  type="speech"  event="tts-speech">  ---- (k)
                <![CDATA[  <param  name="speech_text">
                  <value expr="burg_name">を  ----------(l)
                  <value expr="number">こですね
                </param>  ]]>
          </output>
          <switch  var="number">  ---------- (m)
            <case  value="一つ">
              <assign  var="Qty"  expr="1"/>  ---------- (n)
            </case>
              :
          </switch>
            :
          <goto  next="payment">  ---------- (o)
          </action>
        </exchange>
          :
      </dialog>
  </body>
</xisl>
```

図 3　XISL 文書の記述例

3.2. XISL の概要

XISL は XML ベースの言語である. 最上位要素である<xisl> (図 3(a)) は, 内部に<head>（図 3(b)：文書に関するメタ情報), <body>（図 3(c)：対話シナリオの内容）を含む. <body>は一つ以上の<dialog>（図 3(d)：1 セットの対話) から構成される. <dialog>内部にはまず<begin>（図 3(e)：対話の初期処理), 続いて 0 個以上の<exchange>（図 3(f)：対話の 1 ユニット), 最後に<end>（対話の終了処理）が含まれる.

<exchange>には, 原則として, <operation>（図 3(g)：ユーザ入力）を最初に記述し, 続いて<action>（図 3(i)：システム動作）を記述する. これによりユーザ主導の対話が記述できる. ただし, システム主導の対話を記述するために, プロンプトを示すこともできる. この場合, ユーザへのプロンプト<prompt>をまず記述し, 続いてユーザの入力<operation>を, 最後にシステムの動作<action>を記述する.

<operation>にはユーザの 1 入力を表す<input>（図 3(h)）が一つ以上含まれ, <action>にはシステムの 1 出力を表す<output>（図 3(k)）をはじめとして, システムのアクションに関する様々な要素が含まれる. <prompt>には<output>が一つ以上含まれる. XISL では, 先に述べたように, <input>および<output>の記述方法に自由度を持たせている. これらのタグについては, 属性の種類のみを規定し, 属性値の記法や, 要素の内容については XISL の仕様外としている.

<action>, <begin>, および<end>には, 演

表１　PC 端末で利用可能な入力モダリティ

入力モダリティ	type 属性	event 属性	target 属性	match 属性	return 属性
マウスによる左クリック	“touch”	“click”	入力対象となる XML ファイル名	入力対象となる XML オブジェクトのパス	1:XML オブジェクトのパス 2:イベントが発生した x 座標 3:イベントが発生した y 座標
音声入力	“speech”	“recognize”	文法ファイルを記述する XML ファイル名	文法規則のパス	1,2,…:文法規則と同名の変数の集合
:	:	:		:	:

表２　PC 端末で利用可能な出力モダリティ

出力モダリティ	type 属性	event 属性	内容：CDATA セクションで，内部は一つ以上の<param>要素	
			<param>の name 属性	<param>の内容
ブラウザを開く	“mmi-browser”	“open”	“window-name”	ウィンドウに付ける名前
			“uri”	表示ページの URI
			:	:
合成音声	“speech”	“tts-speech”	“speech-text”	読み上げテキスト
			:	:
擬人化エージェント	“agent”	“agent-speech”	“speech-text”	読み上げテキスト
			:	:

算や条件分岐，繰り返し，XML 文書の読み書きを行なうための様々な命令（図 3(j)(m)(n)）や，<output>内部に変数の値を埋め込むための<value>（図 3(l)）等が用意されている．また<exchange>, <input>, <output>を並列的, 逐次的, 択一的（これは<exchange>と<input>のみ）に実行（もしくは入力受付）するための属性 comb や，comb と同様の機能を持つ制御タグを準備している．さらに<call>や<goto>（図 3(o)）によって，ある<dialog>から別の<dialog>に対話を遷移することができる. これらのタグの詳細は文献[7]を参照されたい.

3.3. フロントエンドの概要

フロントエンドとして PC を対象に，<input>および<output>の仕様を策定した. 実行環境は次の通りである.

- マシン：PC
- OS：Windows®2000
- 入力デバイス：マウス，キーボード，マイク
- 出力デバイス：ディスプレイ，スピーカ
- アプリケーション：
  Microsoft® Agent2.0
  東芝 LaLaVoice™2001
  IBM® ViaVoice™8 Pro
  Microsoft® Mediaplayer
  RealNetworks™ RealPlayer™
  Microsoft® MSXML3.0
  Microsoft® InternetExplorer6

<input>および<output>の仕様の一部を表１，および表２に示す．<input>は XISL の仕様として，type（入力モダリティ），event（入力イベント），target（入力対象の XML ファイル名），match（入力対象の XML 要素），return（入力内容）の計 5 種類の属性を持つ. 上記の実行環境では, 表１に示すような属性値が記述可能である．また，<output>は XISL の仕様として，type（出力モダリティ），event（出力イベント）の計 2 種類の属性を持ち, 内容は CDATA セクションとして記述する. 上記の実行環境では, 表２に示すような属性値が記述可能であり，CDATA セクションには

<param>要素を記述する．

上記の実行環境では，表１，表２に示した以外にも，様々なマウスの操作（ダブルクリックやドラッグ），ウィンドウ操作（表示，移動），サウンドや映像の表示などが仕様として用意されている．

4. アプリケーションの作成

筆者らの研究グループでは，MMI システムのアプリケーションとして，オンラインショッピング（OLS）のアプリケーションを開発している．本節ではその概要を示す．

OLS のアプリケーションは，商品購入のための XISL 対話シナリオ，画面への表示内容と商品データとを格納するための XML コンテンツ，表示スタイルを決定するための XSL スタイルシートからなる．

システムを起動すると，まずユーザの認証が行なわれ，その後，図 4 のような商品閲覧ページが表示される．商品閲覧ページには，ショッピングカート（図 4 の①），商品一覧（図 4 の②），商品の分類項目（図 4 の③）などが表示される．ユーザはショッピングカートの内容変更，商品一覧の変更，購入個数決定ページへの遷移や，商品の詳細説明ページへの遷移を実行できる．これらの操作を含め，OLS での全ての操作は，マウスによるクリック，音声入力，もしくはその両方によって行なわれる．

購入個数決定ページでは，商品の詳細説明ページへの遷移，および購入する商品の個数の決定を実行できる．個数を決定すると，個人情報記入ページか，もしくは商品閲覧ページに遷移する．商品の詳細説明ページでは，詳細を擬人化エージェントに説明させた後，遷移元のページに戻る．個人情報記入ページでは，購入情報を記入した後，ID を持っていないユーザはユーザ ID 登録ページで登録をし，システムは終了する．ヘルプページは，システム起動中の任意の時点で呼び出すことができる．

5. まとめ

多様な端末からの利用を可能にする MMI システムアーキテクチャを提案した．本稿で述べたアーキテクチャは端末に大きな拡張性を持たせており，フロントエンドの実装次第では，

図 4　商品閲覧ページ

ユーザとシステムが行なうより広い範囲の対話を記述することも可能である．今後は PC 以外のフロントエンド仕様策定とその実装，および XISL の記述の詳細化，アプリケーションの更なる開発を進めたい．

参考文献

[1] http://www.w3.org/TR/multimodal-reqs
[2] http://www.saltforum.org/
[3] 植田喜代志，秋田祥史，荒木雅弘，西本卓也，新美康永：“ VoiceXML のマルチモーダル化の検討 ”，情報処理学会研究報告 2001-SLP-38，pp.43-48 (2001).
[4] 大谷佳彦，桂田浩一，山田博文，小林聡，新田恒雄：“ 移植性に優れた MMI システムアーキテクチャの検討 ”，情報処理学会第 63 回全国大会講演論文集（分冊 2），pp.177-178 (2001).
[5] 中村有作，小林聡，桂田浩一，新田恒雄：“ XISL：コンテンツ記述とインタラクション記述分離の試み ”，情報処理学会第 62 回全国大会講演論文集（分冊 4），pp.71-72 (2001).
[6] 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).
[7] 桂田浩一，中村有作，山田真，小林聡，山田博文，新田恒雄：“ 音声対話記述言語 VoiceXML と MMI 記述言語 XISL の比較 ”，情報処理学会研究報告 2001-SLP-38，pp.49-54 (2001).