マルチモーダル対話システムのフレームワーク構築と web ベース対話システムの開発

代表研究者	桂　田　浩　一	豊橋技術科学大学国際交流センター講師
共同研究者	新　田　恒　雄	豊橋技術科学大学情報・知能工学系教授
共同研究者	入　部　百合絵	豊橋技術科学大学情報メディア基盤センター助教

1　マルチモーダル対話システムのフレームワーク構築

1-1　背景

研究代表者らは情報処理学会情報規格調査会「音声入出力インタフェース委員会」(WG4)において，MMI(MultiModal Interaction)システムのユースケース，要求仕様をまとめ，これらに基づいてマルチモーダル対話システムの標準的システムアーキテクチャを策定した．このアーキテクチャは，試行標準委員会の標準として情報処理学会の web サイトにおいて公開されている[1]．アーキテクチャの特徴は，既存の記述言語や開発フレームワークとの高い親和性を持っていることである．これによって実用システムの開発を効率化するとともに，各コンポーネントの役割を明確にして独立性を高めることで研究用プラットフォームとしても機能することを目指している[2]．図 1 に 6 階層モデルに基づく MMI システムアーキテクチャを示す．以下，アーキテクチャの詳細について述べる．

1-2　MMI 6 階層モデルの詳細

MMI 6 階層モデルは MMI システムのための階層型アーキテクチャであり，MMI システムの処理の単位ごとにコンポーネントが階層化されている．また，層間の通信には規定のイベントが用いられる．なお 6 階層モ

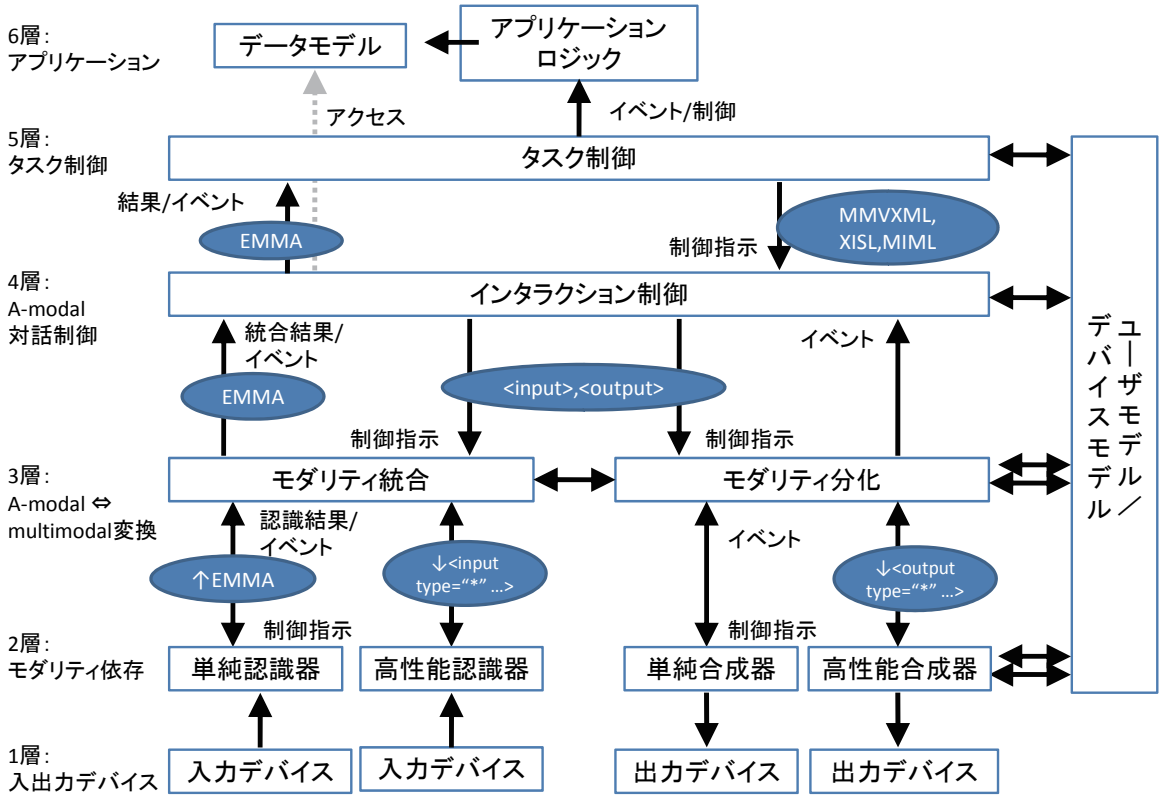

図 1：6 階層モデルに基づく MMI システムアーキテクチャ

デルでは，例えば 4 層と 5 層を統合して 1 つの層として実装するといったことが許されており，実装の際は必ずしも 6 つの階層にする必要はない旨が述べられている．MMI 6 階層モデルの各層の仕様は以下の通りである．

1-2-1　第 1 層：入出力デバイス層

入出力デバイスの制御を行なうラッパープログラムである．この層で取り扱うデータは端末やモダリティの種類，OS，実装方法などに依存してシステムごとに大きく異なる．そのため，試行標準では第 1 層の仕様および第 2 層とのインタフェースを定めていない．

1-2-2　第 2 層：モダリティ依存層

第 2 層のコンポーネントは，第 3 層から試行標準規定の制約情報を受け取り，画面表示，音声入出力，擬人化エージェント制御など，個々のモダリティの制御を行なう．SVG[3]の rect 要素，VoiceXML[4]の prompt 要素と SSML[5]の listen 要素などを処理するモジュールに対応する．認識器は結果として EMMA[6]を，合成器はイベントを第 3 層へ返す．

単独モダリティ認識（合成）モジュールのラッパーとして働く単純認識（合成）器がこの層で用意される．また，比較的リアルタイム性が要求される状況において，複数のモダリティを制御しながら，第 3 層からは単純認識（合成）器として見えるように振る舞う高性能認識（合成）器もこの層に用意される．擬人化エージェントなどの，分解すると複数のモジュールから構成されるコンポーネントは，第 3 層からは単一のモジュールに見えるよう高機能合成器として第 2 層で定義される．

なお，擬人化エージェント出力のように複数のエンジンが密な連携を要求される処理や，音声認識結果を画面に逐次的に出力するなどといった入力状況のフィードバックといった処理は，以下の 3 つの基本機能の組み合わせで実現することができる．

1. ディストリビュータ
   1 つのデバイス／モジュールの出力を分流し，複数のモジュールに配信し，複数のモジュールからの入力を 1 つのモジュール／デバイスにまとめて送る機能
2. フィルタ
   入力の書式を変換する，または，入力の情報を部分的に抽出し，出力する機能
3. トリガ
   入力がある条件を満たすとき，特定の処理を行ない（呼び出し），特定の出力を行なう機能

1-2-3　第 3 層：A-modal⇔multimodal 変換層

入力統合，出力分化，入出力同期制御などを行なう層である．逐次入力や同時入力の解釈，逐次出力や同時出力の同期制御などがこの層のモジュールで実行される．

モダリティ統合モジュールは第 4 層からユーザが入力する情報の制約（データ型）を受け取り，支配下にある第 2 層のモジュールが解釈できる形式（例えば音声認識の場合は文法）に変換した指示を送る．第 4 層からの情報には，特定のモダリティに依存する内容が含まれる場合もある．第 2 層からは EMMA を受け取り，モダリティに依存した情報を除去して第4層へ送る.複数の第2層モジュールからの入力が想定される場合，そのタイミング制御と統合規則を内部記述言語により記述する．

モダリティ分化モジュールは第 4 層から出力したい情報内容を受け取り、支配下にある第 2 層のモジュールが解釈できる形式（例えば音声合成の場合は文字列）に変換した指示を送る．第 4 層からの情報に特定モダリティに依存する内容が含まれている場合もある．出力を複数の第 2 層に分化させる場合は，その分化規則，第 2 層の出力に関して時間同期が必要な場合はそのタイミング制御を内部記述言語により記述する．

内部記述言語としては SMIL[7]（出力），XISL[8][9]（出力，統合解釈）などが想定される．

1-2-4　第 4 層：A-modal 対話制御層

各タスク内の対話制御と応答内容の決定を行なう層である．フォーム処理の充足判定や，タスク内の対話遷移処理などがこの層のモジュールで行なわれる．

第 4 層は単独のモジュールであり，第 5 層からモダリティ独立（モダリティ依存の情報が含まれていてもよい）なインタラクションパターン記述言語で記述された制御情報を受け取り，その解釈を行なう．具体的には，フォーム処理における充足性判定や状況判断，フォーム充足のための次応答処理のモダリティ制御，バージインやシステム割り込みなどタスク内の対話遷移処理が想定される．解釈の途中で適宜第 3 層のモジ

ュールを呼び出すことによりインタラクションを実現する．インタラクションの結果は，記述言語の指示に従って第 5 層へ返される．VoiceXML の FIA (Form Interpretation Algorithm) や XHTML[10]の form 要素の処理などに対応する．

1-2-5　第 5 層：タスク制御層

対話タスクの全般的な制御を行なう層であり，アプリケーション層との入出力通信はこの層が行なう．

第 5 層は単独のモジュールであり，第 6 層にあるバックエンドアプリケーションと連携して第 4 層へ渡すインタラクションパターン記述を動的に生成する．この制御を行なう内部記述言語としては SCXML[11]などが想定される．

1-2-6　第 6 層：アプリケーション層

データモデルとアプリケーションロジックを実装する層である．この層では 5 層に対する API を定義することが要求される．試行標準では仕様を定めていない．

1-2-7　第 7 層：ユーザモデル／デバイスモデル

外部オントロジーで定義されるユーザモデル／デバイスモデル変数を管理し，2～5 層のコンポーネントに対して API を提供する機能を持っている．

2　Web ベースマルチモーダル対話システム

2-1　背景

MMI の実現と普及に向けて，これまでに多くのシステムが開発・公開されてきた．我々の研究グループでも，Galatea Project において Galatea for Windows[12]という MMI システムを構築してきた．MMI システムの特徴は，ユーザの音声やジェスチャー，擬人化エージェントといった多様な入出力を用いる点にある．これらの入出力を複合的に扱えることから，MMI システムは，人間にとってより自然なやり取りをコンピュータ上で実現する対話システムであると言える．

これまで研究代表者らは Galatea for Windows 上で動作する幾つかのアプリケーションを開発してきた．しかしその過程で以下の二つの問題が明らかになった．

(1)　音声認識エンジンのインストールなど，システムの導入コストがかかる．

(2)　複数ユーザを対象に対話が行えない．

MMI システムでは多様なモダリティを利用するため，音声認識エンジンや各種ライブラリを導入する必要がある．単なるインタフェースとして MMI を利用したいユーザにとって，この作業は大きな負担であり，MMI システムの普及の妨げになると言える．また，Galatea for Windows は単一端末／単一ユーザとの対話を想定していたため，各ユーザが MMI システムをインストールする必要があった．ユーザプロファイルの管理や，動的環境情報[13]の利用の観点から，単一システムで複数ユーザに対応することは重要な課題と言える．

我々はこれらの問題を解決するため，ウェブブラウザを用いた MMI システムを開発することにした．ウェブブラウザはほとんどの PC にインストールされており，ユーザが特別なソフトウェアを導入する必要も無い．さらに，近年では Ajax 等の技術により，デスクトップアプリケーションと遜色ないアプリケーションが実現されている[14]～[17]．したがって MMI システムのフロントエンドとして最適であると判断した．

近年の Ajax 等を用いたアプリケーションではブラウザとウェブサーバで役割分担をしつつ負荷の高い処理を協調処理している．MMI システムにおいても音声認識などの処理は負荷が大きいためブラウザ単体では実現が困難と考えられる．そこで MMI システム全体をブラウザサイドと，サーバサイドに分割し，それらの連携によってシステム全体を構成することとした[18][19]．

2-2　Galatea for Windows の構成

Galatea for Windows は，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(Galatea Project)により公開されている MMI システムである．Galatea for Windows は，図 2 に示すように二つのモジュールから構成されている．

対話制御部は，MMI シナリオ記述言語 XISL に記述された対話を解釈・実行するモジュールである．XISL には対話内容や遷移，算術演算，条件分岐，CGI の実行などを記述できるため，一般的な MMI アプリケーシ

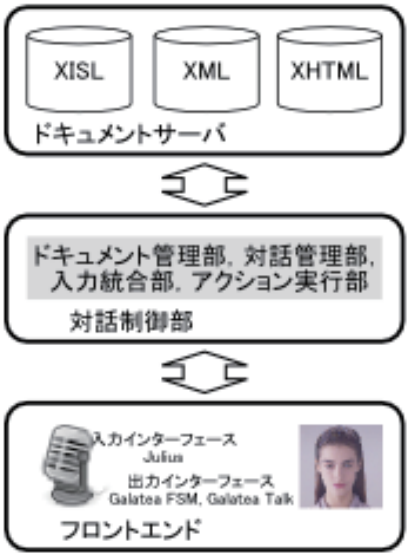

図 2：Galatea for Windows の構成

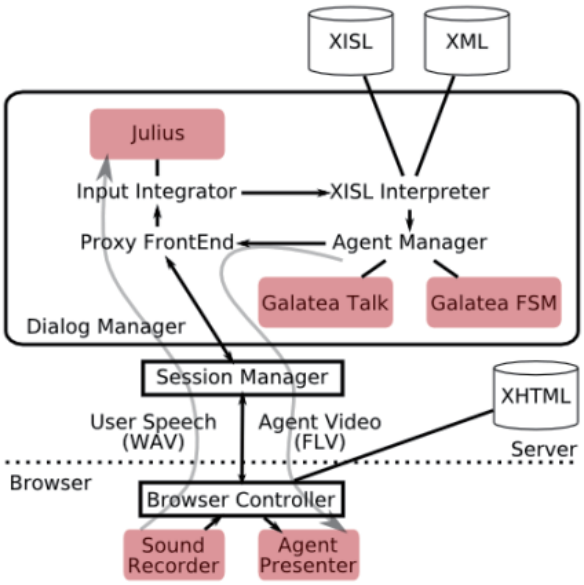

図 3：Web ベース MMI システムの構成

ョンであれば容易に作成することができる．対話内容を記述する XISL や，ユーザに対して表示される XHTML 文書などのコンテンツは，システムから独立したドキュメントサーバ上に配置されるため，再利用性が高く保たれる．

フロントエンドは，音声認識／顔画像合成／音声合成／ウェブページの表示を行うモジュールで，ユーザの直接的なインタフェースとなる．音声認識には Julius[20]，音声合成には Galatea Talk[21]，顔画像合成には Galatea FSM[22]が用いられている．これらの各ソフトウェアは単独で利用できるようパイプ／ソケットによる命令伝達の仕様が公開されており，様々なアプリケーションにおいて個別に利用できるよう整備されている．

2-3　ウェブブラウザへの対応

PC や携帯電話，PDA といったほとんどの情報端末にはウェブブラウザが導入されている．しかしブラウザの種類は多様であり，また端末の性能も高いものから低いものまで様々である．我々はできるだけ多くの端末／ブラウザで動作する MMI システムを実現するため，大部分のブラウザで利用可能な JavaScript と Java Applet／Adobe Flash のみを利用した低負荷の MMI システムを設計することにした．

しかし一般的な MMI システムでは音声認識，およびその他の様々なモダリティ(本システムでは顔画像合成，音声合成)を複合利用するため，全てのモジュールを低負荷に実装するのは困難である．そこで我々は高負荷の処理をウェブサーバ上のプログラム Dialog Manager によって行い，ブラウザ側では低負荷の処理のみを行うようシステムを設計した．図 3 に MMI システムの構成を示す．以下，図 3 の構成に従って各モジュールについて説明する．

2-3-1 音声認識とその他の入力

音声認識エンジンの内部処理は複雑であり，処理能力の低い端末で全体を実行するのは困難である． そこで本システムでは分散音声認識により音声認識を実現することにした．分散音声認識システムには，クライアントサイドで録音のみを行う場合と特徴量抽出までを行いサーバサイドで認識する手法がある．このうち録音のみを行う方法がクライアントの負荷がより軽いこと，および西村らによる w3voice[23]によって実用性が確かめられていることから，我々はこの方法を採用することにした．

本システムでは Java Applet を用いてブラウザ上で音声を録音(44.1[kHz]，16[bit])し，Base64 でエンコードしたデータをサーバに転送する．サーバ側では，ブラウザとのインタフェースを担う Session Manager がデータを受け取り，音声ファイルにデコードした上で Julius に送信する．Julius で生成された認識結果は Input Integrator に送られ，入力統合処理が行われる．

マウスによるポインティング，およびキーボードからの入力についても同様に，ブラウザで取得されたものが Session Manager を通して Input Integrator に送信され，統合処理が行われる．

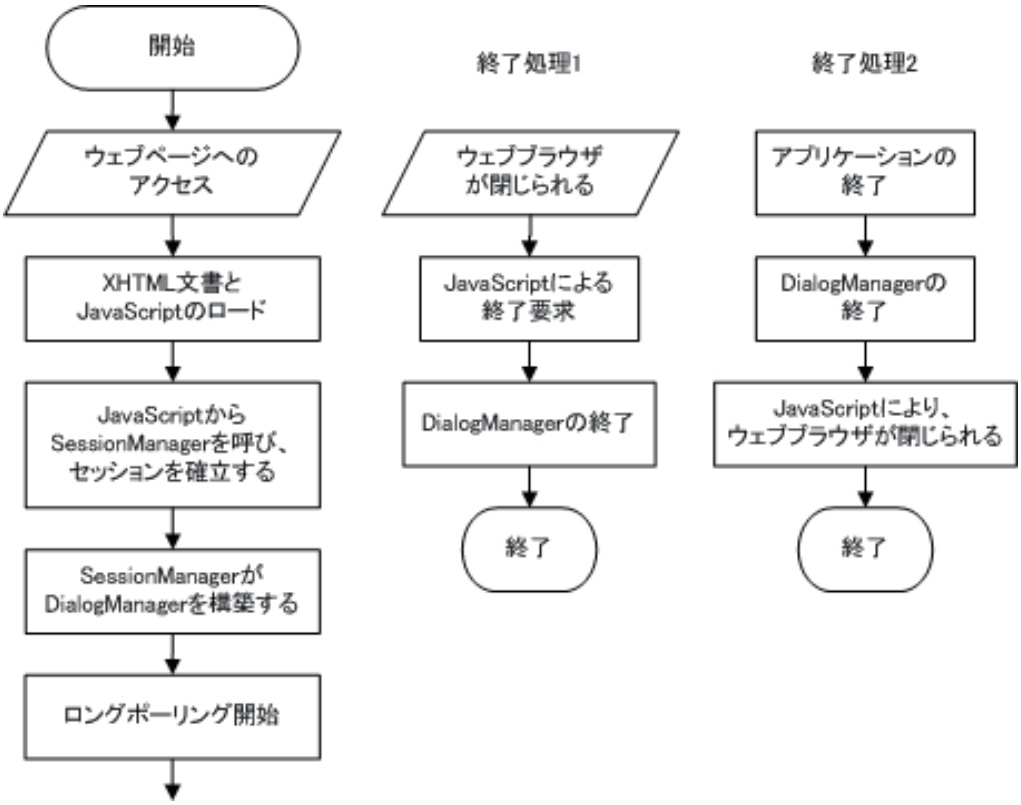

図 4：システムの起動と終了

2-3-2　XISL の実行

Input Integrator において統合処理(2 秒をタイムアウトとし，一単位の入力と捉える)が行われた後に，入力データは XISL Interpreter に送信される．XISL Interpreter は XISL 文書をダウンロードして実行するモジュールで，XISL の記述内容に従って対話進行を行う．XISL に記述された出力内容は Agent Manager に伝達される．

2-3-3 エージェントとその他の出力

顔画像合成，および音声合成による擬人化エージェント出力は非常に負荷の高い処理を行うため，ブラウザ上で動作させるのは困難である．そこで，これらのモジュールはサーバで動作させることにし，出力された擬人化エージェントを FLV 形式の動画像としてブラウザに配信し，ブラウザ上で Adobe Flash を用いて再生することにした．

Agent Manager では XISL に記述された出力を解析し，内容が擬人化エージェントに関するものであれば Galatea Talk と Galatea FSM を用いて動画像を生成する．その他の出力(Web ページの表示など)であればそのまま Session Manager を通してブラウザに送信し，ページ表示などを行わせる．

2-3-4 サーバとブラウザ間の通信

本システムでは，ウェブブラウザからサーバに，ユーザからの入力内容，出力完了通知などを送信する必要がある．これらは非同期通信 Ajax を利用してサーバに送信している．一方，MMI システムからユーザへの通知のようにサーバからブラウザへの通信が必要な場合には，ウェブブラウザからサーバにロングポーリングを行うことで，擬似的なサーバ通知(Comet)を実現している．

本システムのようにブラウザをクライアントとした場合には，複数のユーザが同時に MMI システムを利用することも考えられる．そこで複数のクライアントから入力があった場合にも対応できるよう，Session Manager が IP アドレスと User-Agent 名を保持し，ユーザ識別に用いることとした．

図 6：アプリケーションの実行例

```
<script type="text/javascript"
      src="script/galatea.js"></script>
<script type="text/javascript">
      galatea.initialize();
      galatea.startTask("path.to.xisl");
</script>
```

図 5：Web ベース MMI システム起動のための HTML の記述

```
<exchange>
 <exchange_var>
  <var name="item" />
 </exchange_var>
 <operation><!-- 商品購入 -->
  <input type="speech" event="recognize"
   match="grm/商品名" return="item" />
 </operation>
 <action>
  <output type="agent" event="speech">
   <param name="text">
    <value expr="item" />ですね？
   </param>
  </output>
 </action>
</exchange>
```

図 7：サンプルの XISL

2-4　MMI アプリケーションサンプル

図 4 に本システムの起動から終了までのフローを示す．図 4 に示すように，本システムはブラウザからのウェブページへのアクセスによって起動される．ウェブページの XHTML には，図 5 に示すような記述を含めるだけで，JavaScript で記述されたクライアントプログラムの実行とサーバプログラムへの接続・連携を行うことができる．

クライアントプログラムは，ウェブページの読み込みが完了すると，Session Manager を通じて Dialog Manager の初期化を行い(図 5 の initialize メソッド)，XISL を実行させる(図 5 の startTask メソッド)．

本システム上に構築したショッピングようの MMI アプリケーションの実行例と XISL の一部を，図 6 と図 7 にそれぞれ示す．なお，図 7 はユーザが「リンゴ」と発話すると「リンゴですね？」と擬人化エージェントが答える 1 ターンのやり取りを記述した XISL である，

まずブラウザでショッピングサイトの URL にアクセスすると，上述の手順により XISL の実行が開始される．図 6 のウェブページが表示されている状態で例えば「ブドウ」と発話すると，galatea.js から起動されたアプレットにより録音された音声がサーバに送信され, 音声認識, 入力統合処理を経て XISL Interpreter に渡される．XISL Interpreter では，この入力と図 7 中の input 要素とのマッチングを行う．入力が input 要素と適合しない場合入力は棄却され, 両者が適合した場合のみ Agent Manager に文字列「ブドウですね？」という出力内容が送られる. Agent Manager では, 生成された文字列からエージェントの動画像が合成され, Session Manager を通じてクライアントプログラムに動画像のパスが送信される．この動画像がブラウザ上で再生されることでユーザは擬人化エージェントによる「ブドウですね？」という出力を受け取ることができる．

3　MMI 6 階層モデルに準拠した Web ベースマルチモーダル対話システム

3-1　背景

2 節で開発したシステムは，Galatea for Windows をベースに構築している．実装には JavaScript などの標準技術のみを用いているため，ユーザは特別なソフトウェアのインストールや高性能端末の装備を必要とすることなく MMI を利用できる．しかし，このシステムはモジュール間の結びつきが強いため，使用するモダリティの変更などのシステム改変が容易でないという問題があった．

そこで研究代表者らは MMI 6 階層モデルに準拠するようシステムを再構築することにした[24][25]. MMI 6 階層モデルでは，入出力および対話制御の処理が粒度（モダリティ制御，タスク遷移...）毎に階層化され

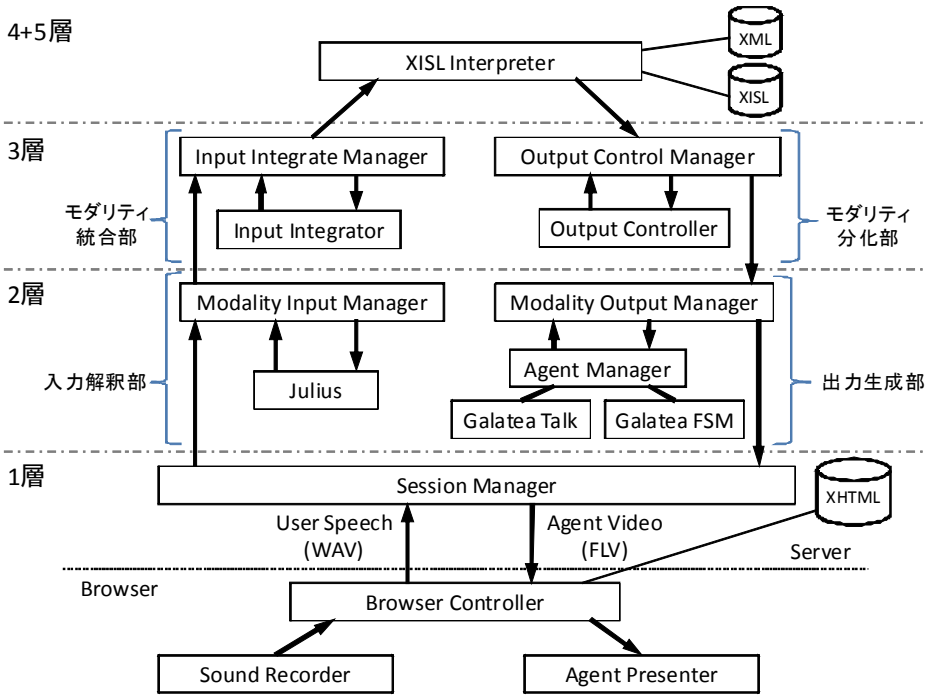

図 8：6 階層モデルに準拠した MMI システムのモジュール構成

ており，階層間で厳密にモジュール分割されている．このモデルに従うことで，モダリティの変更や対話記述言語の切り替えが容易になり，システムの拡張性を向上させることができる．

3-2　システムの構成

2 節で述べた Web ベース MMI システム（以後，従来システムと呼ぶ）を図 8 に示すように MMI 6 階層モデルに準拠する形で再構築した．従来システムと本システムの処理を比較すると，Web ブラウザのモジュール構成は同様であるが，サーバ上の Session Manager 以降（1 層以上）のモジュール構成が大きく異なっている．MMI 6 階層モデルに準拠したことにより，モジュール間の結合度が低くなり，保守性・拡張性が高まった．

なお，本システムでは MMI 6 階層モデルにおける 4 層と 5 層を 1 つの層に統合して実装している．4 層と 5 層を統合した理由は，今回用いる記述言語（XISL，SCXML）がタスク間遷移とタスク内遷移の両方を記述可能であるため，統合した方が実装が容易であったためである．以下，サーバ上のモジュール構成について述べる．

3-2-1　2 層の構成

従来システムでは，Julius，Galatea Talk，Galatea FSM といったエンジンがサーバ内に散在していたが，本システムではこれらのモジュールが 2 層に纏めて設置されている．特に本システムでは各エンジンに対するラッパーの置換，追加を容易にするために，ラッパーを Modality Input Manager，Modality Output Manager の配下に置くように設計した．また，層間の通信はこれらのマネージャが行なうように設計した．これにより，ラッパーおよびエンジンをマネージャの配下に配置するだけで容易に新規のモダリティを追加することができるようになった．

(1) 入力解釈部

従来システムでは，モダリティ統合と入力解釈が同一のモジュールで行なわれていたため，モジュールが肥大化していた．これに対して本システムでは，6 層モデルでの処理の観点から，入力解釈とモダリティ統合の処理をそれぞれ 2 層と 3 層に分離した．これにより，各モジュールの行なっている処理が明確になり，保守性が高められた．

入力解釈の処理の流れは次の通りである．まず 1 層から入力データが送られてくると，Modality Input Manager が入力データに応じてそれを解釈できるエンジンに渡す．エンジンから Modality Input Manager に

処理結果が返ると，6 層モデルの仕様に従って結果を整形した後，3 層に内容を送信する．

(2) 出力生成部

複数のエンジン（音声合成，顔画像合成）が連携して生成されるエージェントは，3 層においては 1 つのモダリティとして扱える方が望ましい．

そこで我々は Galatea Talk による合成音声を出力するラッパーと Galatea FSM による顔画像合成を出力する 2 つのラッパーを連携させ，エージェント出力を行なう Agent Manager を 2 層に構築した．

出力生成の処理の流れは次の通りである．まず 3 層から出力命令が 2 層の Modality Output Manager に送られると，Modality Output Manager は出力命令を実行できるモジュールに出力コンテンツを生成させる．生成されたコンテンツは，Modality Output Manager を通して 1 層に送信される．

3-2-2 3 層の構成

入力統合器（Input Integrator）と出力制御器（Output Controller）を容易に置換・追加をできるようにするために，これらを Input Integrate Manager，Output Control Manager の配下に設置した．これらのマネージャは 2 層と同様に層間の通信も管理している．

なお，入力統合器と出力制御器は，XISL を処理するよう設計した．

(1) モダリティ統合部

先にも述べたように，本システムでは従来システムの Input Integrator の機能を 2 層と 3 層に分割したため，システムの見通しがよくなっている．3 層では Input Integrator のモダリティ統合の処理のみを行なう．

モダリティ統合部では 2 層から解釈結果を受け取ると，Input Integrate Manager 配下の Input Integrator に入力統合を行なわせ，その統合結果を 4 層に送っている．

(2) モダリティ分化部

従来システムでは XISL Interpreter が出力のタイミングを管理するように設計されていた．

これに対して本システムでは，よりきめ細やかな出力タイミングの制御を可能にするために，出力タイミングの管理を XISL Interpreter から切り離し，Output Control Manager が行なうように設計した．

モダリティ分化部では 4 層から出力命令を受け取ると，Output Control Manager 配下の Output Controller に出力分化作業を行なわせ，Output Controller の発行した出力命令を Output Control Manager が 2 層に伝えている．

3-2-3 4+5 層の構成

この層は MMI 6 階層モデルにおける 4 層と 5 層を 1 つに統合した層で，対話の状態管理・制御を行なう．ここでは対話記述言語として従来システムと同様に XISL を利用することを想定している．

XISL Interpreter が本来行なうべきでなかった出力タイミングの制御を分離するために，出力管理を 3 層に切り離すよう設計した．これにより，XISL Interpreter は XISL の解釈・実行と対話の状態管理・制御のみを行なうだけでよくなり，システムの見通しが良くなった．

3-3 構築したシステムの機能拡張

今回構築したシステムは MMI 6 階層モデルに準拠しているため，モダリティ変更などの拡張性が向上している．これを確認するため，2 層の出力エンジンの変更，4+5 層の対話制御および記述言語の変更を試みた．以下，各層の切り替えの実例を示す．

3-3-1 出力エンジンの変更

出力エンジンの変更が可能になると，端末環境に応じて高速なエンジン，高品質なエンジンを使い分けることが可能になる．本稿では出力エンジンの切り替え例として，エージェント出力の切り替え例を示す．

(1) 音声合成エンジンの変更

音声合成エンジンを Galatea Talk から，アクエスト社が組み込み用途向けに開発した規則音声合成ライブラリ Aques Talk[26]を利用したエンジンに切り替えた．Aques Talk は Galatea Talk と比較すると処理が軽く，低機能の端末に組み込みやすいというメリットがある．

システムの変更内容は，新規に Aques Talk を利用したエンジンのラッパーを構築し，Galatea Talk のラッパーと置き換えたことと Aques Talk を用いたエンジンを設置したことのみである．

(2) 顔画像合成エンジンの変更

数枚の BMP から音声の長さだけ連番 BMP を生成するエンジン BMP Creator を自作し，顔画像合成エンジン

の Galatea FSM から切り替えを行なった．BMP Creator は音声の長さ分だけ BMP を複製する処理のみを行なっているため，Galatea FSM と比較するとシステムへの負荷が非常に軽く，高速であるという点がメリットである．

システムの変更内容は，音声合成エンジンの変更と同様に，ラッパーの構築と置き換え，BMP Creator の設置のみである．

3-3-2 インタプリタの変更

対話記述言語を自由に選択できるようになると，開発者は自分の使いやすい対話記述言語を選択できる．

ここでは，インタプリタを XISL から SCXML へ切り替えた例を示す．

SCXML と XISL は双方とも XML 形式の言語であるが，それぞれ異なった特徴を持っている．例えば，XISL が手続き型に処理を記述するのに対し，SCXML ではイベントドリブン型に処理を記述する．XISL はマルチモーダル入出力を扱うことを考えて設計された言語であるが，SCXML は状態遷移を扱うことを考えて設計された言語であるため，SCXML 単体ではマルチモーダル入出力を取り扱うことができない．

SCXML を用いて状態管理を行なう処理系として，Apache による Commons SCXML[27]が知られている．研究代表者らは状態管理（4+5 層）に Commons SCXML を用い，マルチモーダル入出力命令（3 層）に XISL を用いる形でインタプリタを構築した．

【参考文献】

[1] http://www.itscj.ipsj.or.jp/ipsj-ts/ts0012/toc.htm:「マルチモーダル対話のための記述言語 Part1 要求仕様」, 情報処理学会試行標準 IPSJ-TS 0012 (2010-2).

[2] 荒木 雅弘, 西本 卓也, 桂田 浩一, 新田 恒雄: “階層的 MMI アーキテクチャに基づくプラットフォーム実装方法の検討”, 情報処理学会研究報告 2010-SLP-78, No.5 (2009-10).

[3] SVG：http://www.w3.org/TR/SVG11/

[4] VoiceXML：http://www.w3.org/TR/voicexml21/

[5] SSML：http://www.w3.org/TR/speech-synthesis11/

[6] EMMA：http://www.w3.org/TR/emma/

[7] SMIL：http://www.w3.org/TR/SMIL/

[8] 桂田 浩一, 中村 有作, 山田 真, 山田 博文, 小林 聡, 新田 恒雄: “MMI 記述言語 XISL の提案”, 情報処理学会論文誌 Vol.44, No.11, pp.2681-2689 (2003-11).

[9] XISL：http://www.vox.tutkie.tut.ac.jp/XISL/

[10] XHTML：http://www.w3.org/TR/xhtml1/

[11] SCXML：http://www.w3.org/TR/scxml/

[12] Kouichi Katsurada, Akinobu Lee, Tatsuya Kawahara, Tatsuo Yotsukura, Shigeo Morishima, Takuya Nishimoto, Yoichi Yamashita and Tsuneo Nitta: "Development of a Toolkit for Spoken Dialog Systems with an Anthropomorphic Agent: Galatea", Proc. of APSIPA09, pp.148-153 (2009-10).

[13] K.Katsurada, et al., "Management of Static/Dynamic Properties in a Multimodal Interaction System", Proc. Of InterSpeech'07, pp.2525-2528 (2007).

[14] Gmail: Google メール, http://mail.google.com/

[15] Google マップ - 地図検索, http://maps.google.co.jp/

[16] Google カレンダー, http://www.google.com/calendar/

[17] Google Docs, http://docs.google.com/

[18] 桐畑 輝樹, 工藤 正志, 高田 淳貴, 桂田 浩一, 新田 恒雄: "ウェブブラウザ上で動作可能なマルチモーダル対話システム", 情報処理学会研究報告 2008-SLP-73, pp.35-40 (2008-10).

[19] Kouichi Katsurada, Teruki Kirihata, Masashi Kudo, Junki Takada and Tsuneo Nitta: "A Browser-based Multimodal Interaction System", Proc. of ICMI'08, pp.195-196 (2008-10).

[20] 李晃伸, "大語彙連続音声認識エンジン Julius Ver.4", 信学技報, SP2007-54, pp.307-312 (2007).

[21] T. Yoshimura, et al., "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis", Proc. EUROSPEECH'99, pp.2347-2350 (1999).

[22] T. Yotsukura, et al., "An open source development tool for anthropomorphic dialog agent -face image synthesis and lip synchronization-", Proc. IEEE Fifth Workshop on Multimedia Signal Processing, 03_01_05.pdf (2002).
[23] 西村竜一, 他, "音声入力・認識機能を有する Web システム w3voice の開発と運用", 情報処理学会研究報告 2007-SLP-68-3, pp.13-18 (2007).
[24] 工藤 正志, 桂田 浩一, 入部 百合絵, 新田 恒雄: "MMI6 階層モデルに準拠した Web ベース MMI システムの開発", FIT2009 情報科学技術フォーラム, E-039 (2009-9).
[25] 工藤正志, 桂田 浩一, 入部 百合絵, 新田 恒雄: “階層型アーキテクチャに基づいた Web ベース MMI システムの開発”, 電子情報通信学会技術研究報告, SP2009-146, pp.351-356 (2010-1).
[26] Aques Talk：http://www.a-quest.com/aquestalk/
[27] Apache Commons: http://commons.apache.org/scxml

〈発 表 資 料〉

題　　名	掲載誌・学会名等	発表年月