対話システムにおけるモジュール統合とプロトタイピング

新田 恒雄　　下平 博　　西本 卓也
豊橋技術科学大学　北陸先端科学技術大学院大学　京都工芸繊維大学
大学院工学研究科　情報科学研究科　工芸学部電子情報工学科

1. 対話統合フレームワーク

対話システムに搭載する実用的対話エンジンモジュール（音声認識，音声合成，（擬人化エージェント）顔画像合成ほか）の研究が盛んになりつつある。図1にマルチモーダル対話(MMI)機能を持つシステムのアーキテクチャを示した[1]。対話システムでは，モジュールを統合制御し対話を円滑に進める「対話マネジャ」の機能が，システムの成否を握っている（図では，分散環境下のエンジンモジュールを想定し，エージェントマネジャを別に設けている）。一方，対話システムの構築は，多くの特殊な知識を必要とする。このことは個別モジュールの専門知識を持たない開発者に，大きな負担を強いる。プロトタイピングツールは，エンジンモジュールのパラメータ設定，対話シナリオの記述や制御にGUI環境を提供することにより，開発者の負担を大きく軽減する。

以下では最初に，音声対話システムにおけるモジュール統合制御機能と課題，およびプロトタイピングを中心に報告する。続いて二つの重要なテーマ，分散モジュールの統合制御（エージェントマネジャ），およびタスク記述と管理（対話マネジャ）について，技術の動向と課題を報告する。

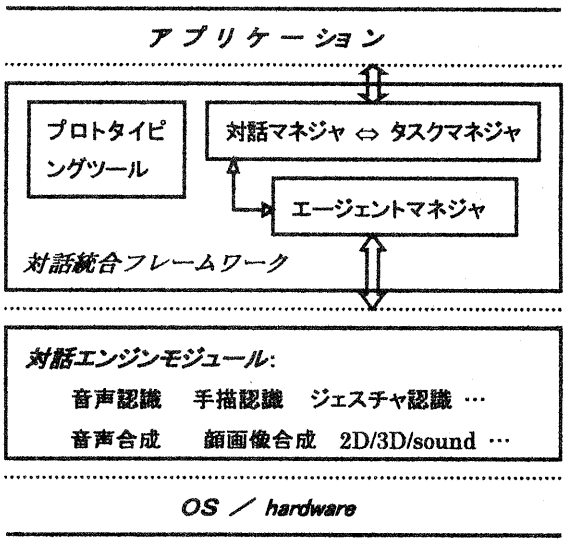

図1　対話統合フレームワーク

2. 対話システムにおける統合制御機能

対話システムに要請される統合制御機能を，モダリティ統合制御と対話フロー制御の二つに分けて説明する。

2.1 モダリティ統合制御

各モダリティは瞬時にかつ同期的に動作する機能を持つことが要請される。この前提が満たされない場合，様々な対話障害が引き起こされる。また入出力イベントへのタイムスタンプ付与も必須である。精緻なモダリティ制御はこれによって可能になる。

また，エンジンモジュール（エージェント）を分散配置した環境では，モジュール間通信を通して同期制御できなければならない。

次に，複数モダリティの利用に関しては，以下の二つの形態をサポートすることが要請される。

[A] モダリティを分離し（逐次もしくは同時に）選択利用する機能：例えば，システムがプロンプトをディスプレイ（音声）で出し，ユーザがタッチ（音声）で入力する）など。これによって対話を「直感的に分かり易くする」ことができる。

[B] モダリティを融合し連繋利用する機能：例えば，同じ（異なる）内容をテキスト／音声／絵／エージェントの動作で説明する。あるいは，音声とジェスチャで照応の利用を可能にするなど。これによって対話を「円滑にしたり，頑健にしたり，あるいは効率を高める」ことができる。

2.1 対話フロー制御

対話フロー制御には最低限：

- 対話モデル
- 対話モデルの指示に沿ってイベントが発生した際に対応する処理を実行するプロセッサ

の二つが必要である。また円滑な対話を目指すには，以下の二つの機能をサポートする必要がある。この他，複数ユーザを受け入れる場合には，話者を認識して対話の流れを制御する機能も必要になる。

[A] 対話主導権の制御に関して： 混在主導(mixed initiative; システム主導とユーザ主導の双方が混合して存在する)が可能であること。これには割り込み機能(barge-in) が必要である。

[B] 対話内容からの制御に関して： 以下への対応が要請される。

- タスク内の切換え（確認，誤り訂正，……）
- タスクの切換え（中断と復帰）
- 複数タスクの同時制御
- ユーザからの対話制御受付（barge-in 含む）／タスク切換え（ヘルプを含む）／打ち切り／…

3. 統合制御における課題

2. に概説した機能仕様を持つ対話システムの設計はまだ緒についたところである。そこで，これらの仕様を順次バランス良く確実にクリアしていくことが大切である。

3.1 確実に動作する対話システムの構築

扱われるモダリティは少なくても，確実に動作する「対話研究のための参照システム(reference system)」が欲しい。現実的な解は，現在進行中の XML ベースの対話システム構築[2],[3]に準拠することであろう。この場合，「ディクテーションソフトウエアが新聞コーパスから作られたｎグラム・レールの上を走るソフトを完成させた」のに対して，「想定した対話記述 (Data Type Definition で決まる) に沿って走るソフトを完成させる」ことになる。制約は大きいが確たる（客観的な）レールを持つため“安心して”設計できる長所がある。対話固有の現象も，ある程度は音声認識モジュールと協働して対応可能である。

3.2 統合制御機能の拡張

3.1 に述べたシステムが実現した後，**2.** に述べた様々な機能を順次搭載することになる。MMI 機能を持つ実用的対話システムでは，マルチドメイン・マルチタスクへの対応[4]，ユーザ端末へのスケーラブルな対応[5]ほかが重要である。またこの段階では，オープンな参照システムとその上の「XML ベース対話ライブラリ」の共同構築が研究上大きなドライビングフォースになると考えている。

4. プロトタイピングツール

対話システムの開発では，最初にシステムの機能仕様を UML 等のオブジェクトモデルで明らかにした後，対話モデルをスクリプト（例えば Voice XML）で記述し，パーザ，インタプリタを介してターゲットシステム上で動作させるのが一般的である[6]。しかし，text ベースの作業は非効率的なため，エンジンモジュールの諸設定，対話シナリオや制御を GUI 環境で行うツールが不可欠となる。これにより，設計ーテストー改良のサイクルが大幅に短縮できる[7]。プロトタイピングツールも，ドメイン特化に始まってマルチドメイン対応へと向かう。

参 考 文 献

[1] 新田：GUI からマルチモーダル UI(MMI)に向けて，情処会誌, vol.36, No.11, pp.1039-1046 (1995).

[2] http://www.voicexml.org/ (2000).

[3] http://www.w3.org/TR/2000/WD-reusable-dialog-reqs-20000426 (2000).

[4] S.Seneff, D.Goddeau, C.Pao, and J.Polifroni: Multi-modal discourse modeling in a multi-user multi-domain environment, Proc. ICSLP'96, pp.192-195 (1996).

[5] P.R.Cohen, A.Cheyer, A.Wang, and S.C.Baeg: An open agent architecture, Proc. AAAI'94-SA, pp.1-8 (1994).

[6] 新田，神尾，雨宮，松浦，内山，田村：マルチモーダル UI とラピッドプロトタイピング，情処 SLP 研資, 7-5, pp. 29- 34 (1995).

[7] 神尾，雨宮，松浦，新田：マルチモーダル UI におけるモダリティ制御統一のためのモデル化手法，情処論, 40, 4, pp.1472-1481 (1999).

モジュール統合におけるエージェントマネジャと通信方式

下平 博
北陸先端科学技術大学院大学 情報科学研究科
sim@jaist.ac.jp http://www-ks.jaist.ac.jp/

1 エージェントマネジャ

擬人化音声対話エージェントシステムを構成する、タスク制御、音声認識、音声合成、顔画像合成等のモジュールが連携して1つの対話システムとして円滑に動作すためには、分散環境におけるシステム制御と情報管理等の仕組が必要である。このような機能を担うモジュールは、マルチエージェントシステム (MAS)[1] の分野では、世話人 (facilitator)、調停者 (mediator)、ブローカ (broker)、 黒板 (blackboard) と呼ばれている。本稿ではこれをエージェントマネジャ(AM) と呼び、モジュール統合のための AM として必要な機能、および AM を介したモジュール間通信の方式について検討する。

MAS おけるシステムの構成法、通信・制御方式については、 FIPA[2]、OMG[3] 等の組織がそれぞれが標準化を試みている他、SRI による OAA[4] がある。これらの標準化方式は大規模システムまでのスケーラビリティや参加エージェント構成の動的な変化も考慮した汎用的な枠組となっている。このため、必ずしも小規模の擬人化対話エージェントシステムに適した仕様ではないが、システム設計の際の参考になる。

2 エージェントマネジャの機能

AM の基本的機能として、(1) 各モジュールの起動・停止・再起動、(2) モジュール間通信の管理 (例えば、経路制御) がある。また、各モジュールの状態の把握や、非同期通信を実現するための共通の情報交換テーブル、すなわち、(3) ブラックボード (黒板) 機能の提供がある。さらに、(4) それぞれのモジュールが発行するコマンドを解釈して、しかるべきモジュールにしかるべきコマンドを発行する機能が必要である. 例えば、タスク制御モジュールから「質問」タグの付いた発話テキストが与えられると、音声合成ならびに顔画像モジュールに対して質問時の音声に適した韻律や表情・動作を生成するためのコマンド列を発行する。

上記の機能に加えて、人間のような自然な振舞いを行う対話エージェントの構築を目指す場合には、以下に示すような (5) 同期処理、割り込み処理等の機能が必要である。

- 合成音声と合成画像 (唇、顎、頭等) の動きや表情が同期するためのタイミング制御を行う。
- Barge-in(システムの発話中にそれを遮るかたちでユーザが発話を開始すること) 等のユーザの割り込みに対して、音声認識モジュールが割り込みを検出し、必要であれば合成音声を止めるとともに適切な顔の表情を作る。
- ユーザの発話中にシステムが適切な反応を随時返す。例えば、聞いていることを示すための、合成音声による「はい」、「うん」等の相槌や、顔画像による頷きや視線の注視等が該当する (図 1)。また、ユーザの発話がうまく聞きとれなかった場合に、すかさず、「えっ?」等の割り込みを入れる。

上述した (1)〜(5) の機能を実現するためには、各モジュール間通信が必ず AM を経由して行われるような論理的にはスター型の接続が望ましい。

U: パソコンですけどね、Celeron を			
S: (音声)	はい	えっ?	何ですか?
(画像)	(うなずく)		(首を傾げる)

図 1: ユーザの発話に対するシステム応答の例

3 モジュール間通信

現状の技術で対話エージェントシステムを構築するには、複数の計算機による分散環境を利用することが現実的である。そのため、モジュール間の通信は必然的に複数の異なるアーキテクチャ間のネットワークを介した通信となる。このような分散環境におけるモジュール (オブジェクト) 間通信の、低レベルの通信方式としては主に UNIX 上で開発された socket や RPC (remote procedure calls) がある。一方、コマンドやデータ通信のための上位レイヤの通信方式としては、以下のような方式が提案されている。

1. CORBA (Common Object Request Broker Ar-

chitecture) [5]
2. COM/DCOM/ActiveX[6], OpenDoc, ···
3. Java/Java-RMI [7]
4. SOAP (Simple Object Access Protocol) [8]

また、エージェント環境を対象とした方式としては、SRIによるOAA[4]や、KTHによるBroker[9]が提案されている。通信用の言語としては通常のコマンド形式の他に、宣言的な記述を用いたKQML[10]がある。

本稿で対象としている擬人化音声対話エージェントシステムの通信方式として、上述の既存技術を採用する選択肢の他に、UNIXシステムで使われている標準入出力を用いる簡便な方式も存在する。モジュールの標準入出力を用いる方式には次のような利点がある。

- 透過的インタフェース：モジュールは独立したプロセスとして動作し、その標準入出力をモジュール間通信のコマンドやデータの通信に用いる。この結果、個々のモジュールの単独デバッグが可能で、開発が容易となる。
- ラッピング技術：モジュール間の通信を実現するために、ラッパ(wraepper)を各モジュールにかぶせ、標準入出力をモジュール間通信の方式に変換する。例えばUNIXのrsh (remote shell)を利用するのが最も簡易である。さらに、ssh (secure shell) [11]を用いれば、通信のセキュリティが向上する。

この他、通信方式を策定する際には、同期処理・非同期処理への配慮も必要である。

3.1 通信コマンド形式

モジュール単位の開発とデバッグ処理の容易性から、ここでは以下のようなテキスト型の通信コマンドを用いる。

書式1 (1行コマンド)

発信ID 宛先IDs > コマンド [パラメータ]

例：AMGR-0001 ASR > set Beamwidth 1000
(ビーム幅を1000に設定)
AMGR-0002 ASR,TTS,FACE,TMGR > init
(各モジュールを初期化する)

書式2 (複数行コマンド)

発信ID 宛先IDs EOF コマンド [パラメータ]
データ
⋮
EOF

例：TMGR-0010 TTS EOF Text ("EOF"までが音声合成する文章)
いらっしゃいませ。.....入力文1
何をお探しでしょうか?.....入力文2
EOF .....データの終了

4 課題

分散環境におけるシステムの実現方法には本稿で説明したようないくつかの標準化の動きがあるので、今後はこれらの動向を念頭に入れたシステム設計が必要である。また、自然な音声対話ににおいては反射的な反応等のリアルタイム性が重要視されるので、通信速度(スループット、レイテンシー)の観点からのAMの評価が必要である。AMの負荷が高く、処理速度低下の原因になるようであれば、即応性を要する処理と、そうではない処理にAMを分割する等の対策が必要である。

参考文献

[1] Robert A. Flores-Mendez. Towards a Standardization of Multi-Agent System Frameworks, 1999. http://www.acm.org/crossroads/xrds5-4/multiagent.html.
[2] FIPA (The Foundation for Intelligent Physical Agents). http://www.fipa.org/.
[3] OMG (Object Management Group). http://www.omg.org/.
[4] OAA (The Open Agent Architecture). http://www.ai.sri.com/~oaa/.
[5] CORBA (The Common Object Request Broker Architecture). http://www.corba.org/.
[6] COM (Component Object Model / DCOM (Distibuted COM). http://www.microsoft.com/com/.
[7] Java Remote Method Invocation. http://www.javasoft.com/.
[8] SOAP (Simple Object Access Protocol). http://www.microsoft.com/japan/developer/-workshop/xml/general/soapspec.asp または、http://www.w3.org/TR/2000/NOTE-SOAP-20000508.
[9] The Broker Architectue. http://www.speech.kth.se/proj/broker/.
[10] KQML (Knowledeg Query and Manipulation Language). 例えば、http://agents.umbc.edu/kqml/.
[11] Secure shell (ssh). http://www.ssh.org/.

擬人化音声対話エージェントのためのタスク管理機能

西本 卓也
京都工芸繊維大学 工芸学部 電子情報工学科
nishi@dj.kit.ac.jp http://www-vox.dj.kit.ac.jp/nishi/

1. タスク管理機能の目標

人間性の豊かな擬人化音声対話エージェントを構築する場合には、自然で多様な表情が可能な顔表情出力、感情を表現できる音声合成、といった要素と、意味理解や推論などの自然言語処理能力、といった要素が重要になる。しかし、幅広い応用を前提とした汎用ツールキットにおいては、対話タスクにおける自然言語処理は必ずしも必要でなく、またアプリケーションにゆだねられる部分が大きい。

対話システムである以上「対話ができること」が求められる。しかし「対話とは何か？」という定義は一般には難しい。そこで、システム構築においては、特定の対話記述言語で記述できるものを対話と定義せざるを得ない。

本稿では、このような対話記述言語として VoiceXML[1]を取り上げて検討する。特に擬人化音声対話エージェントへの応用における問題点を指摘する。

2. VoiceXML の概要

VoiceXML は主に電話を利用した音声応答サービスの提供を支援するマークアップ言語であり、XML をベースとした言語仕様になっている。AT&T, IBM, Lucent, Motorola によって設立された VoiceXML Forum が仕様を作成しており、Version 1.0 が 2000 年 3 月に公開された。

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form>
    <field name="drink">
      <prompt>
       コーヒー、紅茶、ミルクのうちどれかを選択してください。
      </prompt>
      <grammar type="application/x-jsgf">
      コーヒー{coffee} | 紅茶 {tea} | （ミルク | 牛乳）{milk}
      </grammar>
    </field>
    <block>
      <submit next="./drink2.asp"/>
    </block>
  </form>
</vxml>
```

図 1 単純な VoiceXML の記述例

```
C: コーヒー、紅茶、ミルクのうちどれかを選択してください。
H: オレンジジュース
C: 理解できません。(標準エラーメッセージ)
C: コーヒー、紅茶、ミルクのうちどれかを選択してください。
H: 紅茶
C: （drink2.asp に続く）
```

図 2 図 1 の対話例（C:機械, H:人間）

単純な VoiceXML の例を図 1 に示す。これにより図 2 のような対話をシステムと行うことができる。

VoiceXML では対話の種類として、変数の値を得るためのやり取りを定義する<form>と、ユーザに選択肢を提示し、その選択肢にしたがって別の対話に遷移する<menu>の 2 種類が用意され、図 1 では<form>を用いている。<field>は変数の値を得るための要素で、例では drink という変数に coffee,tea,milk のいずれかの値を与えることになる。<prompt>はユーザに入力を促すための音声出力を示す。<grammar>は入力時にユーザに許す言語の文法である。<block>は子要素として実行可能なコードを持ち、例ではドキュメントサーバに値を送信する<submit>を持っている。

VoiceXML のアーキテクチャでは、データベース検索や推論などの処理は「VoiceXML インタプリタ」の外部にある「ドキュメントサーバ」に依頼する。これは Web で動的なページを生成する処理系に似ており、VoiceXML 処理系は Web と同じ HTTP プロトコルでドキュメントサーバと通信する（図 3）。このような構成を取ることにより、VoiceXML インタプリタはユーザ・インタフェースに関する処理に専念し、アプリケーション固有の処理には Web サーバと共通の手法（CGI, PHP, Java Server Page など）を利用できるようになっている。

3. VoiceXML による対話タスクの記述

VoiceXML では入力として音声認識と DTMF（数字入力）を、出力には音声ファイルとテキスト合成音声をサポートしている。

VoiceXML インタプリタは、アプリケーションルートドキュメントに書かれたグローバルな文法やイベント処理を参照しながら、入力に応じてファイルを切り替えて処理を行う（図 4）。

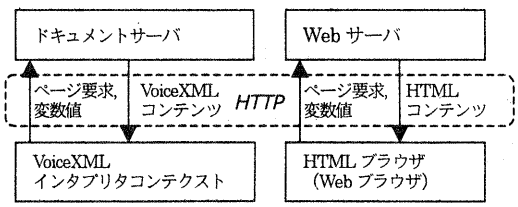

図 3 VoiceXML と Web のアーキテクチャの比較

また、音声認識の文法は JSGF (Java Speech Grammar Format) で記述する。擬人化エージェントを想定した VoiceXML 記述例を図5に、対話例を図6に示す。

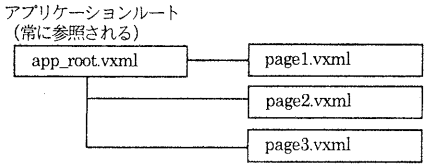

図 4　アプリケーションルートドキュメント

```
[app_root.vxml]　(アプリケーションルート)
<vxml version="1.0">
  <!-- 変数の定義 -->
  <var name="product"/>
  <var name="name"/>
  <var name="address"/>
   (略)
  <!-- グローバルなイベントとその処理 -->
  <catch event="user.goaway">
    <!-- ユーザが立ち去ってしまった -->
    <prompt>またのご利用をお待ちしております。</prompt>
    <goto next="start.vxml"/>
  </catch>
   (略)
  <!-- グローバル文法の定義 -->
   (略)
</vxml>

[start.vxml]　(対話開始ファイル)
<vxml version="1.0" application="app_root.vxml">
  <menu>
    <prompt bargein="true">いらっしゃいませ</prompt>
    <grammar>
      ノートパソコン [(ください | ありますか)]
      {ノートパソコン}
      |デスクトップパソコン [(ください | ありますか)]
      {デスクトップパソコン}
    </grammar>
    <choice next="note_pc.vxml">
      ノートパソコン
    </choice>
    <choice next="desktop_pc.vxml">
      デスクトップパソコン
    </choice>
  </menu>
</vxml>

[note_pc.vxml]
<vxml version="1.0" application="app_root.vxml">
  <menu>
    <prompt bargein="true">
      ノートパソコンですね。どれでしょうか？
    </prompt>
    <choice next="note_pc_vender_a.vxml">A社</choice>
    <choice next="note_pc_vender_b.vxml">B社</choice>
    <choice next="note_pc_vender_c.vxml">C社</choice>
  </menu>
</vxml>
```

図 5　エージェントを想定した VoiceXML 記述例

```
S: いらっしゃいませ
U: えーっと、ノートパソコンありますか。
S: ノートパソコンですね。どれでしょうか？
U: A社のです。
 (以下略)
```

図 6　図5の対話例 (S:システム, H:ユーザ)

4. タスク管理機能の課題

現在 VoiceXML を用いて擬人化エージェントのためのシナリオを実際に記述しながら、実装すべき処理系の仕様を検討している。最終的にはオリジナルの仕様を部分的に簡略化し、部分的に拡張しなくてはならないが、互換性および拡張性に優れている、という XML の利点を生かした機能拡張を目指す。

現在検討中の項目を以下に挙げる。

- 電話系に依存しているイベント (電話を切る操作など) を削除する。また、擬人化エージェント環境に固有のイベントを定義する。例えばユーザがシステムの利用を開始したことを示すイベントが必要である。
- 音声出力に限定されている仕様を、画面による文字や画像の提示を併用できるように拡張する。また入力に関しても、キーボードやタッチパネルの操作に対応するイベントなどを定義する。
- 音声出力をあらわす<prompt>などのタグと関連付けて、エージェントの顔表情制御や合成音声の感情属性などの記述方法を定義する。アプリケーション開発者の便宜を考えると、「承諾」「質問」「挨拶」などの基本的な情報のみによってエージェントが適切に振舞うことが望ましい。また、より詳細な制御のための記述が容易に追加可能であることが望ましい。
- 音声認識および音声合成においては、日本語の表記に加えて読み情報を与える手段を定義する。音声認識においては、あらかじめ文法中に読み情報を付加しておく。音声合成においては、VoiceXML に加えて JEIDA 規格[2]なども取り入れることで、仮名レベルなどでの記述を可能にする。
- 将来的にはユーザ発話の韻律情報やジェスチャ認識への対応など、汎用ツールキットを幅広く応用していくための機能拡張が容易であることが望ましい。

一方で、他の処理系とのコンテンツの互換性に十分配慮すること、魅力的なアプリケーションを開発することも、今後 VoiceXML が普及するためには重要であろう。

参考文献

[1] VoiceXML Version 1.00: VoiceXML Forum, http://www.voicexml.org/ (2000).

[2] 蓑輪 利光, 赤羽 誠, 板橋 秀一: JEIDA 日本語テキスト音声合成用記号, 音講論, 2-1-4, pp.181-182 (2000.9).