複数の音声対話システムの統合制御機構とその評価

河口信夫 [1,2]　長森 誠 [3]　松原茂樹 [1,4]　稲垣康善 [3]

1) 名古屋大学統合音響情報研究拠点 (CIAIR)
2) 名古屋大学大型計算機センター　3) 名古屋大学大学院工学研究科
4) 名古屋大学言語文化部

あらまし　音声認識技術の高精度化によって，近年，様々な音声対話システムが構築されている．特定のドメインに関しては，すでに十分に実用的なシステムも開発されている．しかし，複数のドメインを頑健に扱うことが可能な音声対話システムはまだ少なく，加えて拡張性や開発容易性に関して考慮されているものは存在しない．本稿では，複数の音声対話システムを統一的に利用可能とし，拡張性や開発容易性を満たす統合制御手法を提案する．また，本手法に基づき複数の音声対話システムを統合したプロトタイプシステムを構築した．本稿では実装とその評価についても報告する．

キーワード　マルチドメイン音声対話システム, 音声認識, 統合制御機構

Design and Evaluation of A Unified Management Architecture for Multi-Domain Spoken Dialogue

Nobuo KAWAGUCHI[1,2]　Makoto NAGAMORI[3]
Shigeki MATSUBARA[1,4]　Yasuyoshi INAGAKI[3]

1)Center for Integrated Acoustic Information Research(CIAIR), Nagoya University
2)Computation Center, Nagoya University
3)Graduate School of Engineering, Nagoya University
4)Faculty of Language and Culture, Nagoya University

Abstract　Several spoken dialog systems for the specific task domain have been developed so far. But there are only a few spoken dialog system which can handle multi-domain task. Additionaly, there are no system which consider about extensibility and easy-development. This paper proposes a distributed architecture for the multi-domain spoken dialog systems which satisfies these requirements. We also report a evaluation of the proposed method. using our prototype multi-domain dialogue system.

keywords multi-domain spoken dialog system, speech recognition, unified management architecture

1　はじめに

音声認識技術の高精度化によって，近年，様々な音声対話システムが実用化され始めている．例えば，天気予報や，株価照会，航空機の空席案内といったサービスでは音声認識がすでに実運用されている[1]．しかし，これまでの音声対話システムは，多くが単一のドメインを対象としており，複数のタスクを扱うためには，多くの場合，対象ドメインをメニューから選択する必要があった．これによって，自然発話によってシステムが直観的に利用できるという音声認識の利点が失われてしまうことになる．例えば，自動車内においては，エアコン，カーオーディオ，カーナビに加え電子メールツールや音声ブラウザなどが音声で操作可能になることが考えられる．しかし，これらを使う度に，どのシステムを利用するかを明示的に指示する必要があるとすれば，せっかくの音声インタフェースが繁雑なものになってしまう．また，複数のドメインを対象として，単一のシステムを作り込

[1]2001 年 5 月現在のサービス例：天気予報：各携帯電話事業者，株価案内：野村証券，つばさ証券等，航空機の空席案内：日本航空 等.

んでしまうと，拡張性が問題になる．車には，後から様々な機器が付加される可能性があるが，単一のシステム構成では他ベンダーのシステムとの統合が困難になる．

我々は，複数の音声対話システムを統一的なデータ構造に基づく通信により結合し，統合的に制御する手法を提案している [8, 9]．本手法では，全体のシステムは**ワークモジュール**と呼ばれる単一ドメインの音声対話処理を行なうモジュールと，**マネージャ**と呼ばれる他の複数のモジュールを管理するモジュールから構成される．またシステム内では，フラグメントと呼ばれる音声入出力や付加情報を表現したデータを各モジュール間で分配・統合する．すべてのモジュールがフラグメントを送受信できるように設計することにより，それらを結合するだけでマルチドメイン音声対話システムが構成できる．この手法により，機能の追加・変更がドメイン毎に行え，拡張性が高く柔軟なシステムの実現が可能になる．本稿ではまた，既存の音声対話システムをワークモジュールに変更する実験を行い移行コストを評価した．マルチドメイン音声対話システムのプロトタイプに対し，操作性に関する評価実験を行ったところ，発話の 93.6% において正しいドメインを選択できることが確認され，ユーザビリティに対し好意的な評価を得た．

以下，2 節では統合制御機構の必要性と関連研究について述べ，3 節ではシステムの構成，4 節では単独の対話システムをマルチドメインシステムに変更する実験について報告し，5 節では複数の音声対話システムを結合した音声対話システムのプロトタイプの評価実験について述べる．

2 音声対話システムのアーキテクチャ

これまでに構築されてきた音声対話システムの多くは，天気予報や航空機の空席案内といった単一のドメインを対象にしている [4]．また，複数のドメインを対象にしているシステムであっても，基本的には，システム内では整合性のある処理手法を用いているため，後から付加的に新しいドメインが追加可能な，拡張性や開発容易性を持った機構は考えられてこなかった．しかし，音声対話システムはこれまでの単純なシングルドメインのものから，より複雑なマルチドメインへと発展していくことが予想される．この場合，これまでの音声対話システムの構築手法では，拡張性や開発容易性を満たすことができない．

例えば，車内電子秘書システム [7] を考えてみる．システムを購入した当初は，車内にはカーナビしかなかったが，次第に電子メールシステムや音声ブラウザシステムが搭載されていくことが想像できる．また，これらのシステムは同じベンダからのシステムとは限らない．これを実現するには，オープンで拡張性のある音声対話システムのアーキテクチャが必要になる．本稿では，これらの要求を満たす，階層的な構成を持つシステムアーキテクチャを提案する．

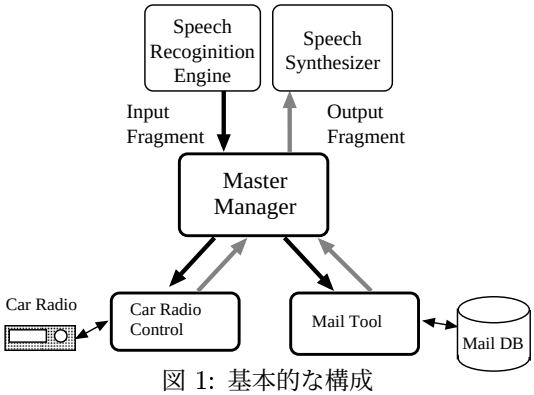

図 1: 基本的な構成

一方，米国では DARPA が主催する Communicator と呼ばれる音声対話システムの研究開発プロジェクトが進められている．このプロジェクトでは，音声対話システムの基本構造として MIT の GALAXY-II と呼ばれる HUB アーキテクチャ [5, 6] を採用している．このアーキテクチャでは，音声認識や言語理解，発話生成等のコンポーネントが各々サーバとなって 1 つの HUB に接続され，HUB 上のスクリプトが各コンポーネント間のデータの受け渡しを行う形式になっている．この構成は，1 つや 2 つのドメインのシステムに対しては有効であることが期待できるが，多数のドメインに対してそのまま有効ではありえない．というのも，HUB アーキテクチャは構成の変更が容易であると記述されているが，結局はコンポーネントを入れ換える度に HUB のスクリプトを書換えなければならない．また，サーバの変更が HUB に直接反映されるため，局所的な変更が全体に影響を及ぼすこともある．また，HUB アーキテクチャは，黒板モデル [2] に似た手法であり，スケーラビリティに乏しい．

本稿の手法では，システムは階層的に構成されるため，末端のシステムの変更は全体に大きな影響を与えず，さらにスケーラビリティも満たす．また，HUB のスクリプトではなく，マネージャを入れ換えたり，モジュールの構成を変更することによって振舞いを変更することができる．

また，契約ネットプロトコル [1] は，マルチエージェントが階層的に接続されており，本稿の手法に近い．しかし，契約ネットプロトコルでは，一般に必ず一つのエージェントが仕事を受け持つのに対し，本手法では，複数のモジュールに分配され，処理結果が統合された後に有効な処理を判断するという手法をとっている点で異なる．

3 複数の音声対話システムの統合制御機構

本稿では，複数の音声対話システムを統一的に制御可能とするために，各コンポーネント間を流れるデータ

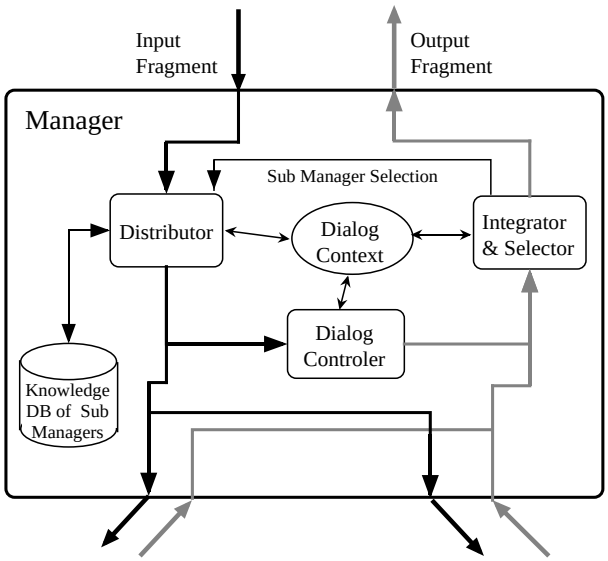

図 2: マネージャの内部構成

を**フラグメント**と呼び，各ドメインの対話システムを**ワークモジュール**，モジュールを統合するモジュールを**マネージャ**と呼ぶ．

3.1 フラグメント

フラグメントにはユーザの発話情報を含む **INPUT Fragment**，ワークモジュールの応答に関する **OUTPUT Fragment**，制御情報である **CONTROL Fragment** がある．フラグメントは，属性と値の対のリストであり，フレーム形式で表現される．基本的な属性を表1に示す．

表 1: フラグメントの属性

属性	種類[2]	説明
ID	I/O/C	フラグメントの ID
phrase	I/O	音声認識結果
relevance	I/O	関連度
moduleID	O	出力モジュール名
select	C	選択されたモジュール名

3.2 全体の構成

図1にアーキテクチャの基本的な構成を示す．この図では，１つのマネージャに２つのワークモジュールが接続されている様子を表している．また，音声認識器，音声合成器もマネージャに接続されている．各モジュール間は，フラグメントの流れで接続されている．

[2]I,O,C はそれぞれ INPUT,OUTPUT,CONTROL Fragment を表している

図 2は，マネージャの基本的な内部構成を示す．マネージャは受けとった入力フラグメントを下部のモジュールに分配し，各モジュールからの応答を統合して，選択した結果を上位モジュールに応答する．

3.3 処理の流れ

処理の流れを図 3に示す．なお，図中の番号は以下の項目に対応している．

1. ユーザ発話の認識
   音声認識器がユーザ発話を認識し，属性として ID, phrase を持つ INPUT Fragment を生成しマネージャの分配部に送信する

2. マネージャの処理（分配部）
   マネージャの分配部は受け取った INPUT Fragment を下部のワークモジュールに分配する．

3. ワークモジュールの処理（解析フェーズ）
   受け取った INPUT Fragment の属性 phrase（認識結果）を参照し，それがワークモジュールとして動作できるかを分析する．動作できる場合は relevance を 1.0 とし，処理を実行した場合の内部状態を保存して待機，できない場合は 0.0 として，ID， moduleID， relevance を持つ OUTPUT Fragment を生成しマネージャに送信する．

4. マネージャの処理（統合部）
   統合しているワークモジュールから受け取ったそれぞれの OUTPUT Fragment の relevance の値を確認して，動作させるワークモジュールを決定する．どちらも 1.0 であった場合， [9] の関連度計算を行いその大小で決定する．自分が最も上位のマネージャの場合，属性 select に選択したモジュール名を記録し， CONTROL Fragment をワークモジュールに送信する．中位のマネージャの場合は，上位に OUTPUT Fragment を送信し，自身が select されるのを待つ．

5. ワークモジュールの処理（実行フェーズ）
   CONTROL Fragment の属性 select を参照し，自分と同じシステム名であった場合は内部状態を保存していた状態に移す．異なる場合は保存していた状態を取消し，認識結果が入力される直前の状態に戻す．

4 ワークモジュールへの移行とコストの評価

本節では，既存の音声対話システムを本手法のワークモジュールへ移行する実験を行った．移行コストの調査によって本アーキテクチャに基づくシステム構築の容易さが評価できる．

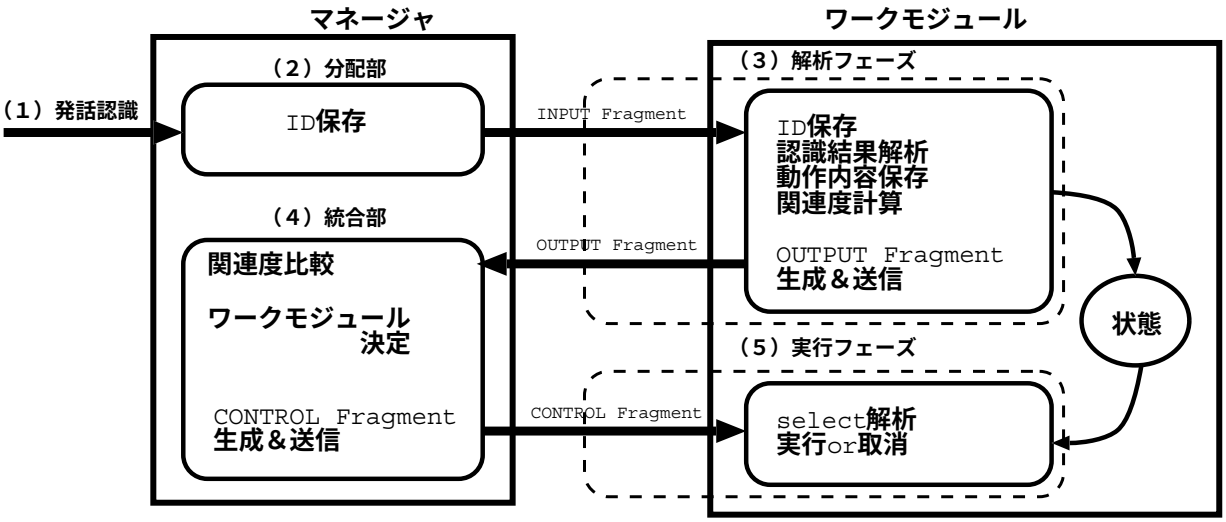

図 3: 処理の流れ

以下，変更対象とする既存の音声対話システムの詳細，評価のために収集するデータ，変更作業の手順について述べる．

4.1 移行する音声対話システム

今回ワークモジュールとして変更する既存の音声対話システムとして二つのシステムを用意した．以下にそれらのシステムの特徴について簡潔に述べる．

MP3 プレーヤ [11]

■ 実装言語: C++ 言語

■ 使用する音声認識エンジン:
スペース区切りの認識結果を出力するもの

■ システムの内容:
音声で動作制御を行なうことのできる MP3 プレーヤ．再生・停止などプレーヤとして基本的な機能以外にも時間を指定した早送りや巻戻し，曲の検索も行なうことが可能である

■ 処理方法
認識結果からキーワードを抽出し，そのキーワードから中間言語を生成する．中間言語を解釈することによりユーザに対し出力（再生・早送りなど）を行う．

レストラン検索システム [12]

■ 実装言語: C 言語

■ 使用する音声認識エンジン:
Julius を使用．システム内に組み込まれている．

■ システムの内容：
自動車運転中を想定したレストラン検索システム．システム主導であり質問に答えていくことで希望のレストランや喫茶店を検索するシステム．

■ 処理方法
認識結果からキーワードを抽出した後解析を行う．
それらからスロット埋めていきスロットの状態からシステムの内部状態を遷移させて動作する

4.2 収集データ

調査対象は「移行コスト」である．移行コストを考えた場合，作業時間と作業量の 2 つを考えることができる．また，作業者の能力によって，それらの時間は変化すると考えられる．そこで，(1) 作業者のプログラミング歴，(2) 変更に要した時間とその作業内容，(3) 変更前・変更後の構成ファイル数，(4) 追加・削除ファイル名及びソースコード行数，(5) 変更ファイル名及びソースコードの変更量（コード変更量），をデータとして収集する．

4.3 データ収集の手順

正確なデータを収集するためには，変更するすべてのシステムに対し作業手順を統一する必要がある．そこで，以下の順序に従って音声対話システムの変更作業を行なうこととした．

1. 変更の打合せ
   あらかじめ作成した仕様書に基づき，作業者と変更の打合せを行なう．本手法のアーキテクチャ，フラグメント及び属性の役割，処理の流れを理解する．

2. ソースコード解読・システム内容の理解
   作業者が音声対話システムの作成者ではない場合，そのシステムの知識を持たずに変更を行なうことは困難であり，また無駄な時間の浪費につながる．そこで，変更する音声対話システムの使用や，ソースコードの解読を通じてシステムを理解する

3. 変更開始
   ワークモジュールに変更する作業を開始する．時間の測定はこの時点で開始する．変更者は 1 つの作業が終了する毎に作業報告として，できる限り詳細な作業内容と所要時間を記入する．

4. 接続テスト・デバッグ
   変更によってできたワークモジュールをテストモジュールに接続してテストを行なう．フラグメントを正しく送受信することができ，動作できればテストを完了する

表 2: MP3 プレーヤのソース変化量

変更前の総行数	2172
変更後の総行数	2439
diff による追加行数	249
diff による削除行数	0

表 3: レストラン検索システムのソース変化量

変更前の総行数	1930
変更後の総行数	2162
diff による追加行数	241
diff による削除行数	160

4.4 移行コストの評価

作業手順に従い，音声対話システムの変更を行った．作業報告書から変更に要した時間を測定し，コード量の変化を調査した．

調査結果は MP3 プレーヤに関するものを表 4，表 2，レストラン検索システムに関するものを表 5，表 3に示す．

MP3 プレーヤの変更について述べる．作業者はプログラミング歴は C 言語が 3 年， C++ が 1 年であり， MP3 プレーヤの作成者でもある．構成ファイル数の変化については変更前と変更後の構成ファイル数に変化はなく，ファイルの追加，削除は行われていなかった．コードの変更は MP3 プレーヤのメインとなる 1 つのファイルのみ行っており，変更前のソースコードの行数は 2172 行，変更後は 2439 行であった．また 2 つのテキストの差分を Unix の diff コマンドを用いて確認したところ， 249 行が変化しており，すべて変更の際に追加されたものであった．追加の内訳はフラグメント処理に関連するコードが 112 行，状態の保存を行なう機能に関するものが 126（ただし，このうちの 112 行は if 文の条件式である），それ以外のものが 11 行の追加となっていた．

作業時間に関しては変更作業には 5 時間 6 分．テスト・デバッグには 1 時間 20 分の時間を費やしており，フラグメント関連の項目が作業時間の 4 割を占めていた．

レストラン検索システムについては，変更者はレストラン検索システムの作成者ではなく，プログラミング歴は C 言語が 1 年， Java が 1 年半である．

構成ファイル数は，変更によって大幅に削除されていることがわかった．これはレストラン検索システムで用いる音声認識エンジン Julius の関連ファイルが，変更によってすべて不必要となったためである（125 個のファイルが削除された）．コードの変更はこちらもメインである 1 つのファイルのみであり，変更前のソースコードの行数は 1930 行，変更後は 2162 行となっている．また同じように diff で差分を確認したところ， 389 行が変化しており，追加が 241 行，削除が 160 行であった．追加の内訳は，フラグメント処理に関するコードがで 117 行，状態を保存する機能に関するものが 75，それ以外で 39 行の追加となっていた．

作業時間に関しては変更作業は 6 時間 22 分，テスト・デバッグには 15 分の時間を費やしており，やはりフラグメントに関連する作業の割合が他の作業に比べて高くなっている．

以上より， MP3 プレーヤ，レストラン検索システムともに，大きな変更を必要とすることなくワークモジュールに移行できることが確認できた．また，フラグメント入力関連に多くの作業時間がかかっているが，フラグメントの入出力に関してはどのようなシステムも共通な動作となることが考えられるため，フラグメント入出力力ライブラリを用意すれば作業時間が短縮されると考えられる．

本実験によって，提案手法への移行が容易であることが確認でき，ライブラリの実現によって，さらに改良が可能であることがわかった．

5 マルチドメインシステムの評価実験

複数のドメインを統一的に扱う音声対話システムは，統合後も各々のドメインの対話システムと同等に扱えるべきである．本節では，本手法を用いて構成したマルチドメイン音声対話システムに対し，操作性に関して行った評価実験について報告する．

前章までに述べた仕様，マネージャの関連度計算法，変更したワークモジュールを用いてマルチドメイン音声対話システムを実装した．マネージャは Java(JDK1.2) を用いて実装を行なった．モジュール間の接続はソケット通信を用いており，接続は GUI で行うことができる．

評価基準は， (1) ユーザ発話に対するドメイン選択の正確さ, (2) システムの処理速度, (3) システムの快適さ，とした．

表 4: MP3 プレーヤの移行に要した時間

作業内容	所要時間	割合
ソケット生成（初期化）関数の生成	1 時間 16 分	19.2 %
Input Fragment 受理・解析モジュール作成	1 時間 45 分	25.5 %
Input Fragment 受理・解析モジュール組み込み	25 分	6.3 %
関連度計算・状態保存機能付加	1 時間 05 分	16.4 %
Output Fragment 生成・送信モジュール作成・接続	35 分	8.8 %
テスト・デバッグ	1 時間 20 分	22.7 %
合計	6 時間 26 分	100 %

表 5: レストラン検索システムの移行に要した時間

作業内容	所要時間	割合
ソケット生成（初期化）関数の生成	47 分	12.3 %
不要コード除去	15 分	3.9 %
Input Fragment 受理・解析モジュール作成	1 時間 06 分	17.3 %
文字コード変換モジュール作成	37 分	9.7 %
Input Fragment 受理・解析モジュール組み込み	29 分	7.6 %
Output Fragment 生成・送信モジュール作成・接続	14 分	3.7 %
関連度計算・状態保存機能付加	1 時間 26 分	22.5 %
認識結果前処理モジュール作成	1 時間 13 分	19.1 %
テスト・デバッグ	15 分	3.9 %
合計	6 時間 22 分	100 %

5.1 実験システム

実験で使用するシステムは**マルチドメイン音声対話システム**と**手動切替えシステム**である．手動切替えシステムとは使用したい音声対話システムを GUI を用いて切替えるシステムである．

また，モジュールの組合せとして**ドメインが類似しない場合**と**ドメインが類似する場合**の 2 通りを考える．これは，ドメインが類似しない場合のほうが，マルチドメインシステムが簡単に動作することが予想されるため，あえてドメインの類似したシステムでの動作を確認するためである．**ドメインが類似しない場合**のシステムは MP3 プレーヤとレストラン検索システムが接続されている (図 4)．**ドメインが類似する場合**のシステムは MP3 プレーヤと，音声操作ラジオシステムが接続されている (図 5)．

5.2 WOZ の利用

今回の実験ではユーザの発話内容を処理することが重要であるため，音声認識誤りによってシステムが誤作動を起こし実験結果に影響が出ることは望ましくない．そこで音声認識誤りを防止するため Wizard of OZ 方式を用いた． Wizard of OZ はあらかじめ被験者が発話すると予想される単語をパネルとして用意し，できる限り被験者の発話に近いものを選択できるようにした．

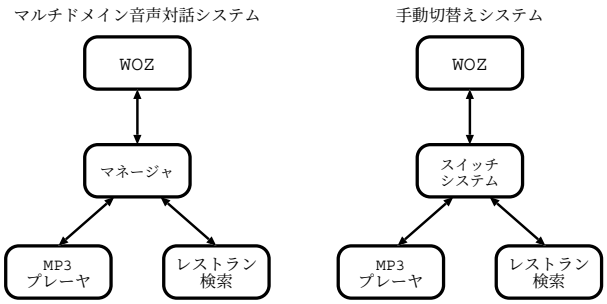

図 4: ドメインが類似しないシステム構成

5.3 収集データ

被験者 16 名に 4 つのシステムを使用してもらい，主観的評価に基づくアンケートを実施した．実験時間は 1 つのシステムにつき 10 分とし，合計 40 分である．

タスクは，こちらで用意した問題をシステムを使用して解いてもらうという形式をとり，問題は異なるドメインに関するものを交互に配置した．また，システムやタスクセットの使用順序が結果に影響を及ぼさないようにそれらの順序を入れ換えて実験を行なった．

アンケート項目は (1) 切替の正確さに対する満足度，(2) システムの反応性，(3) システムの快適さ，の 3 項目であり 4 つのシステムに対して 5 段階評価ですべて記

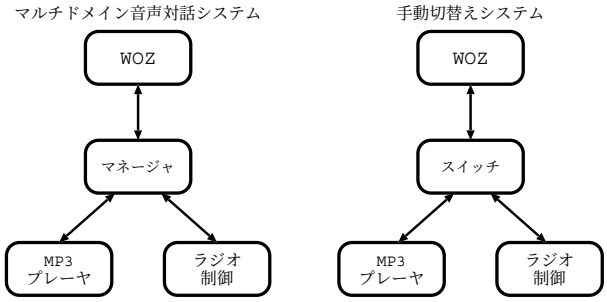

図 5: ドメインが類似するシステムの構成

入してもらった．また，それぞれのシステムに対して満足であった点，不満に感じた点なども自由に記入してもらった．

被験者の発話は DAT を用いて録音を行ない，実験後書き起こしを行なった．

5.4 実験結果と考察

タスク達成数とアンケートの集計結果を表 6, 表 7に示す．以下，ユーザ発話のドメイン選択に関する正答率，システムの処理速度，操作性に関する調査および評価について述べる．

表 6: ドメインが類似しないシステムの評価結果

システム名	正確さ	反応性	快適さ	タスク達成数
マルチドメイン	4.8	2.5	3.6	6.9
手動切替え	5.0	3.9	3.3	7.9

表 7: ドメインが類似するシステムの評価結果

システム名	正確さ	反応性	快適さ	タスク達成数
マルチドメイン	4.6	2.3	3.3	6.8
手動切替え	4.8	3.4	2.9	8.4

5.5 正答率の調査

アンケート結果における切替の正確さに対する満足度の評価はドメインが類似しないシステムで平均 4.8, ドメインが類似するシステムで平均 4.6 であり非常に高い評価であった．また，類似するシステムのほうが，若干満足度が低くなっているのは，ある程度の誤りが存在したためと考えられる．

そこで，実験で得られた書き起し文を用いて，マルチドメイン音声対話システムの正答率の調査を行った．正答率は実験で得られた被験者の音声データに対する書き起こし文を，類似しない場合とする場合のマルチドメインシステムに対しそれぞれ実際に入力し，実行結果を以下のように分類した結果から求めた．

A1 文脈情報を必要とせずに判断できる発話を意図通り正しく処理を行なった

A2 文脈情報がなければ判断できない発話を意図通り正しく処理を行なった

B ワークモジュールの選択は正しいが意図通りの処理ができなかった

C 動作しなかった（認識はしているが反応がない）

D 全くことなるワークモジュールが処理を行なった

今回の実験で得られた書き起こし文の総数は類似するシステムが 557 文，類似しないシステムが 830 文であった．調査を行なう前にすべての書き起こし文に対して，どちらのワークモジュールを使用する発話であるかという情報を手作業で付加した．

表 8: 正答率

	ドメイン類似		ドメイン非類似	
	発話数	割合	発話数	割合
総数	557	100%	830	100%
A1	398	71.5%	790	95.2%
A2	123	22.1%	0	0%
B	0	0%	3	0.4%
C	33	5.9%	37	4.5%
D	3	0.5%	0	0%

調査結果を表 8に示す．正答率はドメインが類似しないシステムで 95.2%，ドメインが類似するシステムで 93.6% であり，類似したドメイン間でのユーザ発話の判断でさえも，類似していない場合と同じように非常に高い正答率で判断ができている．すなわち，本実験によってワークモジュールの解析結果と関連度計算を組み合わせた単純な方法であるにも関わらず，ドメインに影響されることなく高い正答率でドメインを選択できることが確認された．これによって，複数の音声対話システムを統合してもユーザビリティが悪くならないことが示された．

6 まとめ

本稿では，複数のタスクドメインに対して動作する音声対話システムを簡便に構成するための手法を提案した．さらに，本手法を適用するためのコストや，実際に本手法を利用した場合の評価を行った．その結果，本手

法が，拡張性や開発容易性を満たし，さらにユーザビリティを悪くしないことが確認された．

本手法は，システムの構成だけでなく，システムを評価するための基本アーキテクチャとして用いることができる [10]．各モジュールが，音声認識や合成機能から切り離され，独立に変更や拡張が可能になるため，システムの開発サイクルが加速することが期待できる．また，例えば，音声認識器からの INPUT fragment を実際の音声認識結果でなく，書き起しテキストを用いて動作させれば， Offline での対話システムの評価に利用することができる．

また，本稿では，階層化は 1 階層のみであったが，ある程度タスクドメインに特化したマネージャを実現することによって，複数の階層を持つシステムの実現が期待できる．

今後の課題としては，より大規模なシステムのプロトタイプの実現と，モジュール間を接続した時に必要なパラメータを自動的にやりとりする手法の実現である．これによって，新しいモジュールの導入が Plug&Play で実現できるようになる．

謝辞

本研究は文部科学省科学研究費補助金 COE 形成基礎研究費 (課題番号 11CE2005) の補助を受けて行われた．

参考文献

[1] Smith., R.G.,“The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver”,IEEE Transaction on Computers, Vol.29, No.12, pp.1104–1113(1980).

[2] Cohen,P., Heyer,A.C., Wang,M. , and Baeg, S.C.: “An Open Agent Architecutre”, Proc. AAAI Spring Symposium, pp.1–8(1994).

[3] Inagaki, Y. and Matsubara, S.: “Models for Incremental Interpretation of Natural Language”, Proc. of the 2nd Symposium on Natural Language Processing , pp. 51–60(1995).

[4] Matsubara, S., Yamamoto, H., Kawaguchi, N., Inagaki, Y., Toyama, K.: An Interactive Multimodal Drawing System based on Incremental Interpretation, IJCAI97 Workshop: Intelligent Multimodal Systems,pp.55-62(1997).

[5] Zue,V.,“Conversational Interfaces: Advances and Challenges”, Proc. EUROSPEECH’97,pp.KN9–KN18(1997).

[6] Seneff, S. , Hurley, E. , Lau,R. , Pao,C., Schmid,P., and Zue,V. , “GALAXY-II:A Reference architecture for conversational system development”, Proc. ICSLP’98 (1998).

[7] 松原茂樹，河口信夫，外山勝彦，稲垣康善：発話の同時理解・同時生成に基づく車内音声対話秘書システムの提案，人工知能学会研究会，SIG-SLUD-9902, pp.1–6(1999).

[8] Nobuo Kawaguchi, Shigeki Matsubara, Katsuhiko Toyama,Yasuyoshi Inagaki: An Architecture for Multi-Domain Spoken Dialog Systems, *Proc. of the 5th Natural Language Processing Pacific Rim Symposium(NLPRS’99)*, pp.463–466(1999).

[9] 長森誠，河口信夫，松原茂樹，外山勝彦，稲垣康善：マルチドメイン音声対話システムの構築手法，情処研報， 2000-NL-137, pp.83–89(2000).

[10] Pilifroni, J., Seneff, S., “Galaxy-II as an Architecture for Spoken Dialogue Evaluation” , Proc. of 2nd Int. Conf. on Language Resource and Evaluation (LREC),pp.725–730(2000).

[11] 木村 晋一，松原 茂樹，河口 信夫，稲垣 康善，“音声対話 MP3 プレイヤーの実現とその評価”， 電気関係学会東海支部連合大会講演論文集， p.282 (2000).

[12] 早川昭二，磯部俊洋，河口信夫，武田一哉，板倉文忠， “音声対話システムを用いた車内対話の収集”，音響学会講演論文集 (2000).