Vol.2016-SLP-112 No.14
2016/7/30

音声対話システムのオープンコンテンツ化実現のための モジュール仕様および管理手法

山西 元樹[1]　船谷内 泰斗[1]　李 晃伸[1]

概要：音声対話の内容に関わる言語モデルや対話エージェントのモデルや動きなどを音声対話コンテンツとし，それらをモジュール単位で組み替えることで，ユーザが任意のシステムを構築できるモジュールベースの音声対話システム構築環境について，その実現のための仕組みを検討する．本研究では，各モジュールを統一的に扱うためのアーキテクチャ，および各モジュールの依存関係をチェックしてシステムが正常に動作することを保証するための音声対話コンテンツのパッケージ管理システムを提案する．提案手法を MMDAgent へ実装し，それを用いて音声対話コンテンツのパッケージ化がユーザ生成に有効かどうかを検証した．

1. はじめに

近年，音声認識や音声合成などの技術の発展により，音声対話システムが実社会で運用される機会が増えつつあり，Apple の Siri[1] や NTT ドコモのしゃべってコンシェル [2], SoftBank の Pepper[3] のように商用化された音声対話システムも登場し，その普及に火が付き始めている．しかし，現状ではまだ様々な音声対話システムが広く普及しているとは言えない．その理由として，音声対話システムは音声認識器や音声合成器などのモジュールが複雑に結びついた巨大なシステムであるため，その構築に各モジュールに関する専門知識が必要となり高いコストがかかってしまうことや，ユーザが音声対話に求めるものが，簡単な一問一答の対話からエンターテイメント性の高いものまで幅広く，それに応じた対話シナリオや声質モデルなどが必要となるため，ありとあらゆる要望に単一のシステムでは対応できず，ユーザが望む対話を行うことが困難であることなどが挙げられる．

我々は，音声対話システムにおける単語辞書やモデル，対話シナリオ，表出（対話エージェント等）といったタスク依存要素を音声対話コンテンツと呼び，これをシステムから切り離してメディアとして扱うことに関する研究を行っている [4]．音声対話コンテンツをシステムの駆動部であるモジュールと分離することで，コンテンツの流通・再利用が容易となる．また，音声対話コンテンツが Wikipedia や YouTube のようなユーザ生成型コンテンツへと昇華することで，コンテンツが爆発的に生成されるようになり，それによりユーザが求めるシステムを自由に構築できるようになる環境の実現が期待される．

ユーザ生成型音声対話コンテンツを実現するためには，一般人からクリエイタまで様々な人間が自ら作りたいコンテンツを自由に作成できることが必要であり，そのためには音声対話システムに関する知識がなくてもシステムを構築できるような環境を実現することが必要である．そこで，本研究では，音声対話コンテンツをモジュールとして扱い，モジュールの組み合わせによる音声対話システムの構築を行うモジュールベースの音声対話システムの実現を目指す．コンテンツをモジュール化し一般化することで，ユーザのコンテンツ作成の敷居を下げ，コンテンツの再利用が容易となることで多種多様なシステムの構築が可能となることが期待される．

本稿では，モジュールベースの音声対話システム実現に向けた音声対話コンテンツのモジュール仕様と，モジュール間の重複や依存関係を解決する管理手法を提案する．

2. ユーザ生成型音声対話コンテンツ

この章では音声対話システムにおけるコンテンツおよびユーザ生成型音声対話コンテンツについて説明する．なお，本研究で扱う音声対話システムはソフトウェアエージェントを画面に表示するモジュールが追加されているものを想定する．

[1] 名古屋工業大学大学院　工学研究科
Graduate School of Engineering Nagoya Institute of Technology

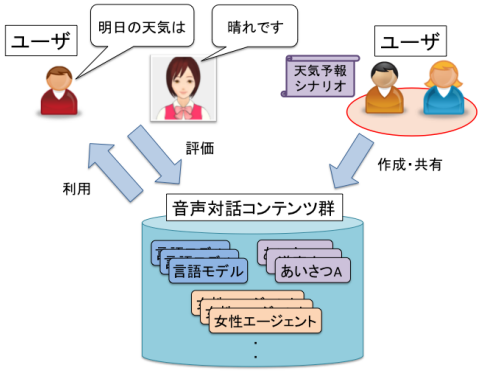

図 1 ユーザ生成型音声対話コンテンツの概念図

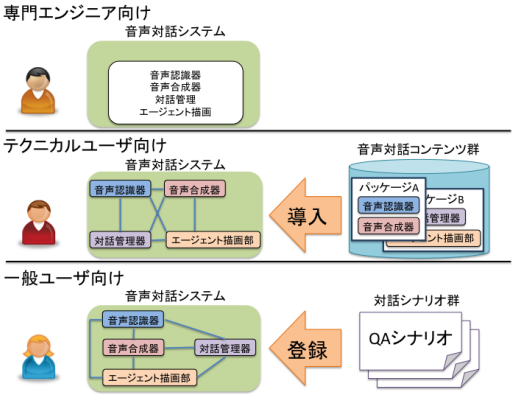

図 2 ユーザ層に適合した音声対話システム構築環境

2.1 音声対話コンテンツとは

音声対話システムは，声や画面に表示されるモデルやその動き，対話シナリオといった対話タスク依存の部分と，音声認識エンジンや音声合成の方式といった対話タスク非依存の部分に分けることができる．本研究では，対話タスク依存部分を音声対話コンテンツ，対話タスク非依存部分を処理モジュールとする．音声対話コンテンツとして，音響モデルや言語モデル，音響モデルといった各モデルや，対話シナリオ，画面に描画するエージェントなどのモデルやそのモーションがある．

2.2 ユーザ生成型音声対話コンテンツとは

ユーザ生成型音声対話コンテンツとは，音声対話コンテンツにユーザ生成型コンテンツの仕組みを取り入れたものである [5][6][7][8]．ユーザ生成型音声対話コンテンツの概念図を図 1 に示す．ユーザ生成型コンテンツでは，ユーザがコンテンツを作成・共有したり，他のユーザの作成したコンテンツを利用・評価・改良することができる．このように音声対話コンテンツがユーザ生成型コンテンツ化することで，ユーザの求めるものをそのまますぐにコンテンツに反映でき，多種多様なコンテンツを生成することができることが挙げられる．

3. 音声対話コンテンツのモジュール化

高度で複雑な音声対話システムから簡単で作りやすい音声対話システムまで様々なシステムの作成を行える環境の構築にあたっては，技能の異なるユーザ層ごとに適した構築のスタイルを提供する必要があると考えられる．その概念図を図 2 に示す．上段は専門エンジニア向けの構築環境である．現在の音声対話システムはこれに当てはまる．この環境では，研究科や企業の専門エンジニアが全てのモジュールを作成し，それらをひとまとまりにして構築する．

図 2 の下段は一般ユーザ向けの構築環境である．これまでのユーザ生成型音声対話コンテンツの研究で構築されたシステムの構築環境はこれに当てはまる．この環境では，

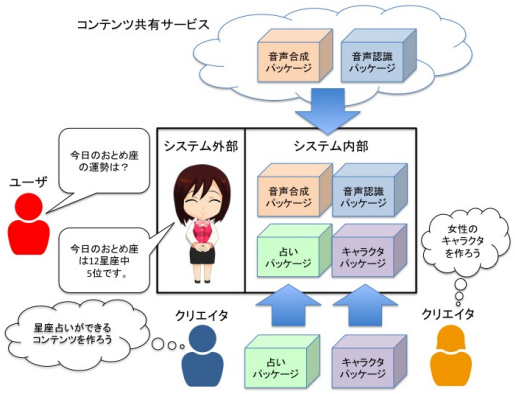

図 3 モジュールベースの音声対話システム概要図

一般ユーザは一問一答の対話シナリオをシステムに登録して使用する．作成するコンテンツを一問一答に限定することでコンテンツ登録の敷居を下げることができるが，複雑な対話を行うシナリオや対話シナリオ以外の音声対話コンテンツには対応していない．

本研究では，図 2 の中段で示すテクニカルユーザ向けの構築環境を研究対象とする．この環境では，音声対話コンテンツを一般化されたモジュールとして扱い，モジュールの組み合わせによってシステムの構築を行う．本研究では，これをモジュールベースの音声対話システムと呼ぶ．

モジュールベースの音声対話システムの概要図を図 3 に示す．ユーザはコンテンツ共有サービスやクリエイタが作成したコンテンツパッケージをスマートフォンにアプリケーションをインストールするように導入し、組み合わせることで自分が作成したいシステムを構築することが可能となる。また，コンテンツ作成がモジュール単位となることで，コンテンツ作成に必要な知識が作成するモジュールに関することに絞られるため，クリエイタよるコンテンツ作成の敷居を下げることも期待される．

音声対話システムのモジュール化を実現するためのアーキテクチャは，モジュールの高度な部品化と容易な組み換えを実現する必要がある．また，モジュールを組み合わせて構築したシステムが動作するためには，全てのモジュー

ルを管理し正常に機能することを保証する仕組みが必要である．そこで本研究では，モジュール化を実現するためのアーキテクチャとして分散型音声対話アーキテクチャを，モジュールを管理する仕組みとして音声対話コンテンツのパッケージ管理システムを提案する．

4. 分散型音声対話アーキテクチャ

モジュールの部品化を実現するための仕組みとして，分散型音声対話アーキテクチャを提案する．なお，本研究では音声インタラクションシステム構築ツールキットMMDAgent[9] への実装を想定している．MMDAgent は，各モジュールがそれぞれ独立したスレッドで非同期に動作しており，それらはテキスト形式のメッセージのやりとりで連携している．全てのモジュールはグローバルメッセージキューを用いてメッセージの同期を行う．メッセージの例を図 4 に示す．この例は，エージェントモデルをディスプレイに表示させ，「こんにちは」という発話に対して「いい天気ですね」という文を発話する対話シナリオを表している．”MODEL_ADD|mei|MEI.pmd”は”MEI.pmd”というエージェントモデルをディスプレイに表示する命令のメッセージで，”RECOG_EVENT_STOP| こんにちは”は「こんにちは」という発話が入力されてきたことを伝えるメッセージ，”SYNTH_START|mei|normal| いい天気ですね。”は「いい天気ですね」という文章を合成して発話する命令のメッセージ，”SYNTH_EVENT_STOP|mei”は合成音声の発話が終了したことを伝えるメッセージである．

分散型音声対話アーキテクチャの概要図を図 5 に示す．本アーキテクチャでは，全ての音声対話モジュールを有限状態トランスデューサ (FST) とみなし動作させる．例えば，音声対話器なら音声を入力として認識結果を出力とする FST と考えることができ，エージェントのモデルやモーションであればそれらを動作させるイベントを入力として，動作している状態を出力とする FST とみなすことができる．このように，全てのモジュールを FST と考えることで，モジュールの機能に依らずに統一的に扱うことが可能となる．

実際には，MMDAgent の構成要素のうち音声認識や音声合成，コンテンツを拡張するためのプラグイン (.dll) と対話シナリオ (.fst)，エージェントの 3D モデル定義ファイル (.pmd)，モーション定義ファイル (.vmd) の 4 つをモジュールとみなす．この中でプラグインは各プラグイン毎に内部でメッセージの入出力が行えるように変更を行い，3D モデル定義ファイルとモーション定義ファイルは MMDAgent 内部でモデルとモーションそれぞれをコントローラで監視し，そこでメッセージの入出力を行う．

5. 音声対話コンテンツのパッケージ管理

モジュールベースの音声対話システムの構築においては，モジュール間の依存関係や重複をシステムが管理し，自動で解決する仕組みが重要である．

```
# ADD_MODEL
00 01 <eps>                          MODEL_ADD|mei|MEI.pmd|
                                     0.0,0.0,-10.0
# Hello
01 11 RECOG_EVENT_STOP|こんにちは SYNTH_START|mei|normal|いい天気ですね。
11 01 SYNTH_EVENT_STOP|mei         <eps>
```

図 4 MMDAgent のメッセージ例

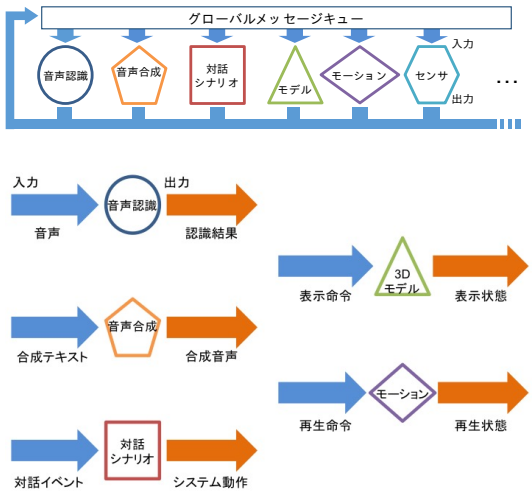

図 5 分散型音声対話アーキテクチャ概念図

音声対話システムは複数のモジュールが連携している．専門知識のあるユーザであれば，モジュール間の連携を全て満たしたシステムの構築が可能である．しかし専門知識のないユーザの場合，一部のモジュールを導入しなかったために，モジュール間の連携が不完全なシステムを構築してしまうことが考えられる．また，同じ機能のモジュールを誤って複数導入してしまい，その結果システムの動作に影響を及ぼすことも考えられる．

ところで従来より OS やソフトウェアでは，コンテンツの依存関係や重複を解決するための仕組みとして，パッケージと呼ばれる形式で管理・配布する仕組みが大いに用いられている．例えば Linux では，Advanced Package Tool (APT) と呼ばれるパッケージ管理システムが用いられている [10]．APT ではパッケージ毎に含まれているファイルや依存関係，バージョンといった情報を管理しており，その情報をもとにパッケージのインストール時に，連携している他のパッケージの自動インストールや重複しているパッケージの自動検出を行う．

そこで本研究では音声対話コンテンツをタスク毎にパッケージにまとめ，それらを管理する仕組みを提案する．本研究で扱うパッケージ管理の要件を以下に示す．

- モジュールが外部から与えられた名前で管理されること
- モジュール間の連携を外部から指定できること
- 使用するモジュールを一括で管理できること
- モジュール間の依存関係・重複を自動でチェックでき

ること

5.1 内部変数の導入

各モジュールは内部変数を持ち，その値をモジュールを組み合わせたときに外部から指定可能にする．モジュールは内部変数から必要なパラメータを参照できるものとし，その値は後述するパッケージ管理ファイルから統一のフォーマットで与えられるものとする．内部変数の実装は，プラグインではプログラム内部で持つものとしプラグイン内でその値を適用する．対話シナリオでは MMDAgent に読み込まれる際に，MMDAgent 内の対話管理部でその値を適用する．3D モデル定義ファイル，モーション定義ファイルも MMDAgent に読み込まれる際に，MMDAgent 内のコントローラでその値を適用する．

5.2 ファイルを用いたパッケージ管理

パッケージ毎にパッケージ管理ファイルを作成し，そこにパッケージ内のモジュールの名称と内部変数の値を全て記述する．このとき，他のパッケージをネストすることも可能とする．ここでモジュール間連携の指定は連携するモジュール名を内部変数として指定する．

5.3 モジュール間依存関係の管理

モジュールの入出力を用いてモジュール間の依存関係・重複の管理を行う．各モジュールについて，その入出力メッセージを以下の 4 つに分類する．

- 入力される可能性のあるメッセージ
- 出力される可能性のあるメッセージ
- 機能するために他のモジュールの入力に期待するメッセージ
- 機能するために他のモジュールの出力に期待するメッセージ

あるモジュールが正常に動作するためには，機能するために他のモジュールの入出力に期待するメッセージを入出力する可能性のあるメッセージがパッケージ内に含まれている必要がある．モジュールの依存関係が成立している例を図 6 に示す．このパッケージ内の対話シナリオについて，他のモジュールの入力に期待するメッセージを音声認識器が出力し，出力に期待するメッセージを音声合成器が入力としているため依存関係が成立している．このように各モジュールのメッセージを比較することでモジュール間依存関係を管理する．また，これら 4 種類のメッセージが全て同一であるモジュール同士を重複しているものとする．

6. 具体的な実装

6.1 パッケージの構成

パッケージの構成要素を以下に示す．

- パッケージ管理ファイル (.xdf ファイル)
- 音声対話モジュール，パッケージ
- メッセージ仕様記述ファイル (.h ファイル)

パッケージ内に.xdf ファイルは一つ，音声対話モジュール，パッケージは複数個含むことができ，各モジュールについて.h ファイルを用意する．

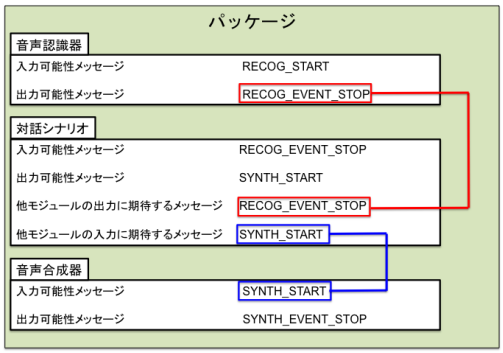

図 6 依存関係が成立しているモジュール例

```
julius ./RECOG/Plugin_Julius.dll
ojt    ./SYNTH/Plugin_Open_JTalk.dll
sample ./DIALOG/sample.fst -D recog=julius -D synth=ojt
Mei    ./Content/Mei/Mei.xdf
```

図 7 .xdf ファイルの記述例

6.2 パッケージ管理ファイル (.xdf ファイル)

パッケージに含まれているモジュールをまとめる.xdf ファイルを定義する．.xdf ファイルの記述例を図 7 に示す．.xdf ファイルは「モジュール名 コンテンツ実態へのパス オプション」の順に記述する．オプションでは変数の値を定義することができ，「-D 変数名=値」のように記述する．また，他のパッケージをネストする場合にはそのパッケージの.xdf ファイルを記述する．

6.3 メッセージ仕様記述ファイル (.h ファイル)

モジュール毎にそれぞれのメッセージの入出力仕様を記述する.h ファイルを定義する．.h ファイルの名前はモジュール名と同名とする．.h ファイルの記述例を図 8 に示す．`in` には入力される可能性のあるメッセージの仕様，`out` には出力される可能性のあるメッセージの仕様，`require-in` にはモジュールが機能するために他のモジュールの出力に要求するメッセージの仕様，`require-out` にはモジュールが機能するために他のモジュールに入力に要求するメッセージの仕様をそれぞれ記述する．ただし，合成音声の入出力のメッセージには認識結果や応答文が含まれており，これらを全て記述することは困難であるため，.h ファイルに記述するメッセージは”|” で区切られた最初の要素のみとする．

6.4 動作時の流れ

パッケージを用いた MMDAgent へのコンテンツ導入の

```
in RECOG_EVENT_STOP, SYNTH_EVENT_STOP
out MODEL_ADD, SYNTH_START, MOTION_ADD
require-in RECOG_EVENT_STOP, SYNTH_EVENT_STOP
require-out MODEL_ADD, SYNTH_START, MOTION_ADD
```

図 8 .h ファイルの記述例

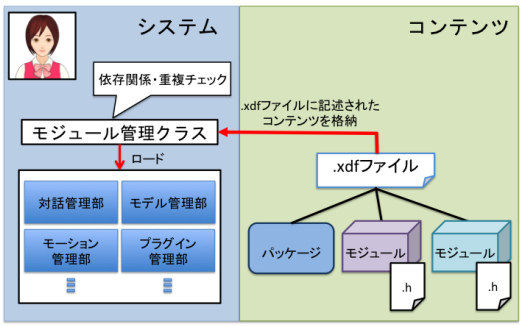

図 9 パッケージを用いたコンテンツ導入の流れ

流れを図 9 に示す．システムは起動時に.xdf ファイルを読み取り，そこに記述されている全てのモジュールをシステム内のモジュール管理クラスに格納する．このとき，格納されるモジュールの.h ファイルも読み取り，各メッセージの情報も全て格納される．その後モジュールの重複チェックと依存関係チェックを行う．このとき，重複するモジュールや依存関係が成立しなかったモジュールが一つでもある場合，システムはそれを検出することができ，システムを起動させなかったり，必要なモジュールを探すなどを行うことができる．モジュール管理クラスに格納された全てのモジュールの重複と依存関係がチェックできた場合，システムに必要なモジュールをロードする．

7. 検証実験

モジュールベースのユーザ生成型音声対話システム実現のためには，まず音声対話コンテンツのパッケージ化がユーザ生成に有効かどうかを検証する必要がある．そこで本研究では，FST の記述によるコンテンツ作成とパッケージの組み合わせによるコンテンツ作成のどちらがどの程度簡単にコンテンツ作成とシステム構築が行えるか比較する実験を行った．

7.1 実験条件

本実験では，従来の MMDAgent を使用したことがある被検者に提案手法の MMDAgent を使用してもらい，モジュールベースのユーザ生成型音声対話システムによるシステム構築が可能かどうか検証した．実験では，従来の MMDAgent と提案手法の MMDAgent それぞれに対して (1) 用意した 6 種類のタスクから 1 つを選択し，それを行うコンテンツを作成する． (2) ベースとなるシステムに作成したコンテンツを追加し，使用するモデルを用意された 6 種類から 1 つ選択して変更してもらう．の 2 種類の実験を行った．以下，実験条件を示す．

- ベースとなるコンテンツは MMDAgent の公開版サンプル [11] をそれぞれのシステムの仕様に合わせて構成したものを用いる．また，提案システムではパッケージの雛形はあらかじめ用意してある．
- それぞれのシステムの使用方法や FST の記述方法，コンテンツの作成・追加方法は事前に説明を行う．
- コンテンツの作成とシステム構築は 30 分で行う．また，作成するコンテンツは最低でも 3 種類の一問一答を作成してもらう．
- システム構築では，構築したシステムの動作確認まで行ってもらう．

また，実験終了後にアンケートをおこなった．アンケートでは FST に関する知識がどの程度あるのかということと，従来システムと提案システムを比較して，「わかりやすさ」「自由度」「親切さ」「不快さ」「思い通りに作成できる」「今後も使いたい」「今後も期待できる」「総合評価」の 8 項目について 5 段階の一対比較法で回答してもらった．また，それぞれのシステムについて自由記述欄を設けた．

表 1 アンケート結果 (1:従来システム，5:提案システム)

項目	評価点
わかりやすさ	2.7
自由度	2.9
親切さ	3.8
不快さ	2.5
思い通りに作れた	3.0
今後も使いたい	3.7
今後も期待できる	4.3
総合評価	3.4

7.2 実験結果

今回は 13 名に対して実験を行った．アンケートの結果を図 1 に示す．ここでは 1 に近い値であるほど従来システムのほうが高い評価であり，5 に近い値であるほど提案システムのほうが高い評価である．また，被検者を FST に関する知識で分類しそれぞれの平均をまとめた．結果を表 2 に示す．

従来システムに関する自由記述欄では，「モデル名の変更が面倒」「状態遷移番号の記述が面倒」「一行一行の意味がわかりにくい」といった FST 記述に関する意見があった．

提案システムに関する自由記述欄では，「.xdf ファイルの記述が面倒」「コンテンツの alias 名とパッケージの alias 名の違いがややこしい」「編集時に複数の.xdf ファイルを開く必要があり大変」といった.xdf ファイル記述に関する意見がみられた．

7.3 考察

表 1 を見ると，「親切さ」「今後も使いたい」「今後も期待できる」の項目で提案システムのほうが評価が高く，音声対話コンテンツにおいてもモジュール化しパッケージング

表 2 FST に関する知識別アンケート結果 (1:従来システム，5:提案システム)

項目	全く知らない	あまり知らない	どちらでもない	少し知っている	よく知っている
わかりやすさ	4.0	4.0	3.0	2.5	2.3
自由度	3.0	4.0	3.0	2.8	2.8
親切さ	4.0	4.0	4.0	3.8	3.7
不快さ	3.0	2.0	3.0	2.5	2.5
思い通りに作れた	2.0	4.0	2.0	3.0	3.2
今後も使いたい	4.0	5.0	3.0	3.8	3.5
今後も期待できる	4.0	4.0	4.0	4.3	4.5
総合評価	4.0	4.0	3.0	3.3	3.3

することが必要であると感じる被検者が多かった．

一方で，「わかりやすさ」「自由度」「思い通りに作れた」の項目は，従来システムと提案システムで変わらないという評価になった．これは今回の実験では，.xdf ファイルをエディタで直接編集してもらったため，それがユーザがシステムを構築する際の負担となってしまったためと考えられる．

また，表 2 を見ると，「わかりやすさ」「自由度」の項目について，FST の知識が多い被験者は従来システムのほうが高い結果となり，慣れの影響も大きいとみられる．

8. むすび

本研究では，モジュールベースの音声対話システム実現を目指し，音声対話コンテンツをモジュール化するための分散型音声対話アーキテクチャと音声対話コンテンツにおけるパッケージ管理システムを提案した．また，MMDAgent 上で音声対話コンテンツとして扱われるファイルを分散型音声対話アーキテクチャで再定義し，パッケージ管理を行うためのファイル (.xdf ファイル) や入出力メッセージを比較することでパッケージ間依存関係を検出できる仕組みを実装した．

さらに提案システムを音声対話システムに関する知識のあるユーザに使用してもらい，コンテンツ作成とシステム構築のしやすさを従来システムと比較してもらった結果，モジュール化が向上に繋がる可能性が確認できた．

音声対話システムのモジュール化によって，専門知識のないユーザであっても音声対話システムを自由にカスタマイズ可能となり，ユーザのニーズに合った多種多様な音声対話システムの構築ができるようになうることが期待される．

今後の課題として，現在ユーザが直接エディタで記述しているパッケージ管理を自動で行う仕組みの構築や，より音声対話コンテンツに適したパッケージ管理手法の検討が挙げられる．

参考文献

[1] iOS-Siri-Apple
http://www.apple.com/jp/ios/siri/
[2] NTT ドコモ-しゃべってコンシェル
https://www.nttdocomo.co.jp/service/shabette_concier/
[3] Pepper とは？
https://www.aldebaran.com/ja/peppertoha
[4] 徳田恵一，”コンテンツ生成の循環系を軸とした音声技術基盤の構築を目指して”，音声言語シンポジウム，2011-12
[5] 大浦圭一郎，山本大介，内匠逸，李晃伸，徳田恵一，“ キャンパスの公共空間におけるユーザ参加型双方向音声案内デジタルサイネージシステム ”，人工知能学会誌，Vol.28, No.1, pp.60—67, 2013-01.
[6] 石川博規，山本大介，高橋直久，”音声対話コンテンツのパッケージ化とその配信システム”，マルチメディア、分散、強調とモバイル (DICOMO2014) シンポジウム，pp.789-795，July.2014
[7] 飯塚遼，李晃伸，”ユーザ生成型音声対話システムにおけるクリエイタとユーザの相互刺激によるインセンティブ向上の検討”，研究報告音声言語処理，2014-SLP-101，6，pp.1-6，2014
[8] 宮木京介，飯塚遼，李晃伸，”利用者による履歴付き対話の共同構築・拡張が可能なユーザ生成型音声対話システム”，日本音響学会 2015 年秋季研究発表会講演論文集，pp.197-198，2015-09
[9] Akinobu Lee, Keiichiro Oura and Keiichi Tokuda, “ MMDAGENT—A FULLY OPEN-SOURCE TOOLKIT FOR VOICE INTERACTION SYSTEMS ”, IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP) p.8382 -8385，2013.
[10] APT HOWTO
https://www.debian.org/doc/manuals/apt-howto/index.ja.html
[11] MMDAgent サンプルスクリプト
http://www.mmdagent.jp/