Vol.2015-ICS-179 No.2
2015/3/20

エージェントを用いたスマートフォン向け音声対話システム拡張機構の開発と評価

仲野 良佑[1]　打矢 隆弘[1]　西村 良太[1]　山本 大介[1]　内匠 逸[1]

概要：3D キャラクタとのリアルタイムでの音声対話が可能な音声対話システムとして MMDAgent が提案されている．この動作環境を拡大し，利便性を向上させるために MMDAgent を Android スマートフォン上へ移植したスマートメイちゃんが提案されている．これらのツールキットにおいて採用されているシナリオ定義手法では，複雑な対話シナリオの作成や，端末間通信による連携を必要とするシナリオの構築が困難である．本研究では，これらの問題点を解決するため，スマートメイちゃんをソフトウェアエージェントにより拡張する機構を提案，実装し，有効性を確認するための評価実験を行った．エージェントを用いてスマートメイちゃんの拡張を行うことで，複雑な応答文や条件分岐を持つシナリオの開発がより容易となり，また，ネットワークと連携する必要があるシナリオの開発も，エージェント間通信の延長線上と捉えることで格段に容易となる．評価実験の結果，提案機構によって問題点を解決できたことを確認したが，実用的な機構とするためにはいくつかの課題点も存在することが確認された．

キーワード：エージェント，音声対話システム

1. はじめに

近年，音声による操作が様々なデバイスにおいて急速に普及してきており，特に携帯型デバイスにおいて音声対話システムの採用例が非常に多くなっている．その筆頭となるスマートフォンにおいては，Apple 社の Siri[1] や NTT docomo のしゃべってコンシェル [2] など，多くの音声対話システムが利用できる．

また，リアルタイムで描画される 3DCG エージェントによって魅力ある対話が行える音声対話システムとして，MMDAgent[3] が提案されている．音声に合わせ 3DCG エージェントをアニメーションさせることにより，表情やしぐさなどの視覚的な情報を含んだ対話を行うことができ，ユーザにとってより親しみやすいシステムとなっている．MMDAgent は PC 上で動作する音声対話システムであるが，これを Android 上へ移植したものとしてスマートメイちゃん [4] が提案されている．

スマートメイちゃんは Android スマートフォン上で単体で動作し，MMDAgent と同様，リアルタイム 3DCG エージェントを表示しながら遅延の少ない自然な会話が行える．また，Siri やしゃべってコンシェルなどと異なり，スマートメイちゃんを用いて対話を行う場合は会話の解釈にネットワークを介する必要がない．そのため，スマートメイちゃんによって実現される会話は，Siri やしゃべってコンシェルと比べ圧倒的に応答が速い．

多くの利点があるスマートメイちゃんにおける対話であるが，スマートメイちゃん上で対話を行うためのシナリオの作成に着目すると，以下の問題点が存在する．

(P1) 内容が変換するシナリオの作成が困難

スマートメイちゃんの対話シナリオは，FST(Finite State Transducer) と呼ばれる有限状態遷移モデルを用いたスクリプト言語で記述されるが，条件分岐を含むような複雑なシナリオを記述する際は記述が非常に複雑になり，シナリオ作成が困難となる．シナリオ記述はプラグインを用いた拡張が可能であるが，この場合は 1 つのシナリオがプラグインと FST スクリプトに分断され，開発者の負担が増加する．

(P2) 通信を用いるシナリオの作成が困難

スマートメイちゃんが利用される場面が増加した場合，スマートフォン間でデータをやりとりするシナリオを記述する要求が発生することが考えられる．現在の機構では，このようなシナリオの作成は容易ではなく，記述が非常に困難となる．

これらの問題を解決するため，本稿では，スマートメイ

[1] 名古屋工業大学
Nagoya Institute of Technology

図 1　スマートメイちゃん

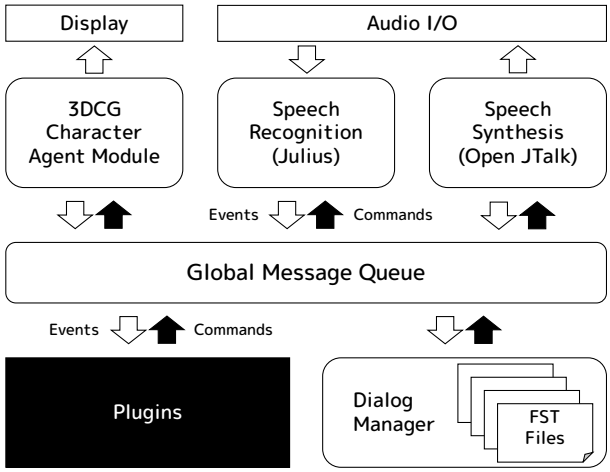

図 2　スマートメイちゃんの構成

ちゃんをソフトウェアエージェント (以下，エージェント) を用いて拡張することを提案する．

エージェントとは，自律性や協調性，即応性を備えるソフトウェアコンポーネントの総称である．特に本研究においては，自律的に動作し，自身で使用する知識を持ち，他のエージェントやインターネット上のサービスと通信することができるソフトウェアのことを指す．

エージェントを用いてスマートメイちゃんの拡張を行うことで，複雑な応答文や条件分岐を持つシナリオの開発や，ネットワークと連携する必要があるシナリオの開発が格段に容易となる．

本研究で提案する機構により，スマートメイちゃんの問題点を解決し，スマートメイちゃんのシナリオ作成の効率化・活発化を図る．

2. スマートメイちゃん

スマートメイちゃんは，MMDAgent を Android[5] スマートフォン向けに移植した音声対話システムである (**図 1**)．MMDAgent と同様のシステム構成となっており (**図 2**)，音声認識エンジン Julius[6] や音声合成エンジン Open JTalk[7] を利用して構成されている．MMDAgent と同様，リアルタイムに描画される 3DCG キャラクタと対話が可能である．

また，C++や Java 言語を用いたプラグインによる拡張が可能な設計となっている．プラグインからは Android API にアクセスすることが可能であり，プラグインと FST スクリプトを連携させることによって Android API を利用するシナリオも実現可能である．

3. 問題点

3.1 (P1) 内容が変化するシナリオの作成が困難

現在，MMDAgent やスマートメイちゃんにおいて，FST スクリプトを用いて実装されているシナリオは一問一答方式のものが多く，また，その応答文も定型であり，状態に応じて応答内容が変化したり，応答内容を動的に作成したりするシナリオの作成が想定されていない．

スマートメイちゃんの動作環境は Android スマートフォン・タブレットであることから，デジタルサイネージなどでの活用が主眼に置かれている MMDAgent と比較し，より個人的な用途に特化することが求められる．個人的な用途に特化するためには，利用者がどのような環境に置かれているか，また，過去にどのような対話をしたかを認識し，応答内容を変化させる必要がある．しかし，FST スクリプトでシナリオを記述する場合，条件分岐を行うことは容易ではなく，記述量が非常に増大する問題がある．

FST スクリプトで記述できる処理は，発話やモデルのアニメーションなど，予め定義されている内容に限られる．定義されていない処理を実行したい場合はプラグインを作成して補う必要があるが，シナリオごとにプラグインを開発することは開発者にとって負担が大きく，FST とプラグインによってシナリオが分断されるため，見通しも悪くなり，保守性も低下する．また，シナリオの実行には端末へプラグインとシナリオ両方の配置が必要となり，開発者の負担がさらに増加する．

このように，FST とプラグインによるシナリオ作成には限界があり，より表現力の高い別の方式を併用するか，または置き換える必要がある．

3.2 (P2) 通信を用いるシナリオの作成が困難

スマートメイちゃんが利用される場面が増加した場合，スマートメイちゃんがインストールされている端末間で対話情報の交換を行う必要があるシナリオや，複数のスマートフォン間で協調を図り，情報を共有する対話シナリオなど，端末間の通信を必要とするシナリオを作成する要求が発生することが考えられる．

また，MMDAgent がインストールされている PC やデジタルサイネージとの通信を行う要求が生じる可能性も考

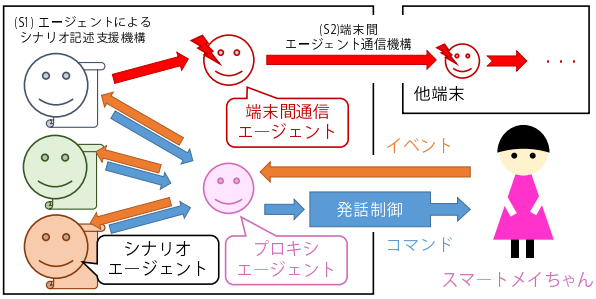

図 3 提案機構の全体図

えられる．

しかし現在の構造では，たとえ P1 の問題点が解決されたとしても，端末間通信を用いるシナリオはかなりの作成コストがかかることが予想される．

そのため，通信処理を補助する機構を作成し，シナリオから手軽に利用できる環境を整え，このようなニーズに対応する必要がある．

4. 提案機構

本稿では，ソフトウェアエージェントを用いてスマートメイちゃんを拡張する機構を提案する．エージェントを用いてスマートメイちゃんのシナリオを記述することによって，問題点の解決を図る．提案機構は，2 つの主要な内部機構を持っており，それぞれ前章で述べた問題点を解決するものである．各機構の概要を以下に示す．

(S1) エージェントによるシナリオ記述支援機構

(P1) を解決．エージェントを用いた対話シナリオの記述を支援する．

(S2) 端末間エージェント通信機構

(P2) を解決．エージェントから手軽に利用できる端末間通信機構を提供する．

提案機構の全体図を**図 3** に，提案機構を含むスマートメイちゃんの全体図を**図 4** に示す．図 4 から分かるように，提案機構はスマートメイちゃんが持つ FST ベースの対話機構を置き換えるものではなく，FST による対話シナリオは提案機構を付加した後も引き続き利用可能である．

4.1 エージェントによるシナリオ記述支援機構

提案機構では，シナリオの実装をエージェントを用いて行う．シナリオを提供するエージェントをシナリオエージェントと呼ぶ．提案機構は，シナリオエージェントを記述するために必要な処理や，よく利用される処理をパッケージ化したものを提供し，シナリオ記述量の削減を支援する．

本機構の実装には，Jade LEAP for Android 4.3.0[8][9] を用いた．Jade LEAP は for Android は Android 上で動作するエージェントフレームワークであり，Java で記述さ

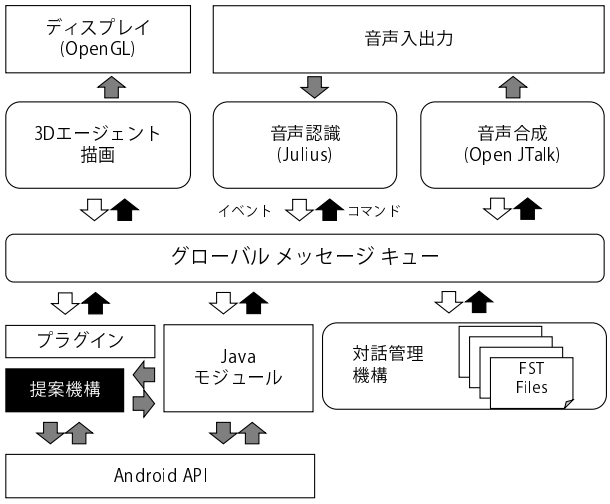

図 4 提案機構を含むスマートメイちゃんの全体図

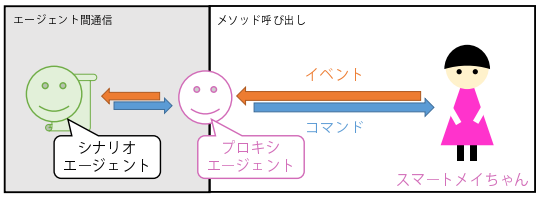

図 5 プロキシエージェント

れたエージェントに対して動作基盤を提供する．

4.1.1 プロキシエージェント

提案機構では，シナリオエージェントとスマートメイちゃん間の通信を仲介するものとして，プロキシエージェントを配置する (**図 5**)．プロキシエージェントは，提案機構内で 1 つだけ存在する．

プロキシエージェントは，スマートメイちゃんからのイベントの受信及びコマンドの送信を仲介する．スマートメイちゃんとの通信はスマートメイちゃんが備えるプラグイン機構を利用し，シナリオエージェントとの通信はエージェント間通信を用いる．それぞれの処理を以下に述べる．

イベントの受信

(1) スマートメイちゃんからイベントがプロキシエージェントに通知される．

(2) プロキシエージェントは通知されたイベントをエージェント間通信メッセージに変換する．

(3) プロキシエージェントは変換したメッセージを各シナリオエージェントにブロードキャストする．

(4) イベントメッセージを受け取ったシナリオエージェントは各自処理を行う．

コマンドの送信

(1) 各シナリオエージェントがプロキシエージェントへ実行したいコマンドを記述したエージェント間通信メッセージを送信する．

(2) プロキシエージェントは受信したエージェント間通信メッセージをコマンドへ変換する.

(3) 変換したコマンドをスマートメイちゃんへ通知する.

(4) スマートメイちゃんがコマンドを実行する.

4.1.2 発話制御機構

エージェントはそれぞれが非同期的に動作を行う特徴がある. このため, エージェントから発話を行う場合, 複数のエージェントから発話命令が同時に行われ, 発話命令が衝突することがある. 発話命令が衝突した場合, スマートメイちゃんの発話に問題が生じるため, 避ける必要がある.

提案機構では, プロキシエージェント内に発話制御機構を導入し, コマンドの送信時に発話命令の競合を検出し, 適切に処理することで問題の解決を図る.

発話制御機構は, プロキシエージェントに実装され, シナリオエージェントからコマンドを受信した際, またはスマートメイちゃんからイベントを受信した際に処理を行う.

コマンド受信時

発話制御機構は, すべてのリクエストをキューに追加する. また, キューされたコマンドが処理中でない場合はコマンド実行処理を行う.

イベント受信時

イベントが発話中フラグである場合, 内部の発話フラグの状態を更新する. 発話中フラグがオフになった場合やイベントが発話終了を示す場合には, キューされたコマンドが処理中でない場合はコマンド実行処理を行う.

コマンド実行処理

キューの先頭にコマンドが存在する場合は, そのコマンドが発話以外の場合は即座に実行する.

キューの先頭に存在するコマンドが発話の場合かつ, スマートメイちゃんが発話中の場合 (発話中を示すフラグが設定されている場合), 発話が完了するまで待機する. 発話中でない場合は, 発話中を示すフラグを設定し, 発話を行う.

4.2 端末間エージェント通信機構

エージェント間通信によってエージェントの相互作用が実現可能であるが, このエージェント間通信を端末間にまで拡張することで, 端末間における情報共有を簡単に実現可能とする. 端末間のシナリオ連携が, 端末内におけるシナリオ連携と同様の記述となるため, 開発のハードルが下がり, また, 開発コストも減少することが期待できる.

そこで, 提案機構には, 端末間通信を担うエージェントとして, 2 つの通信エージェントを配置する (**図 6**)

通信エージェントは, エージェント間通信で送信されるリクエストに応じて, 他の端末に存在するエージェントと通信を行う. また, それぞれの通信エージェントが保持する通信について, 通信開始や切断などが発生した場合に, 各エージェントへ通知を行う.

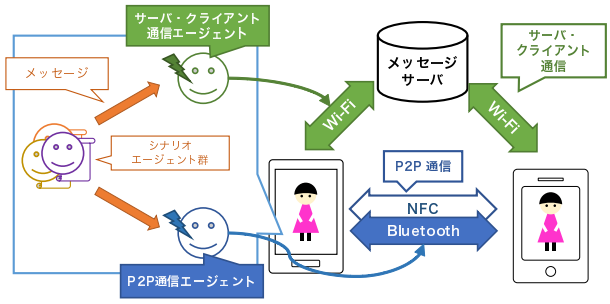

図 6 端末間通信機構の概要図

表 1 通信機構の比較

	1 対 1	1 対多
通信相手	1 名	複数名
通信規格	NFC, Bluetooth	Wi-Fi
通信距離	約 10m 以内	約 100m 以内*1
通信の開始	相手の端末と接触	エージェントによる操作

提案機構では, 通信エージェントを 1 対 1 と 1 対多向けのもので分割しており, 用いる通信規格が異なる. それぞれの比較を**表 1** に示す.

端末間の通信確立には, 特に 1 対 1 通信の確立にはなるべく簡易な手法が要求される. 本研究では, P2P 型通信においては NFC(Android Beam)[10] を用いて接続し, Bluetooth を用いて通信を行うよう設計している.

通信モデルの違いから, 2 つの通信モデルは全く別の実装となっているが, それぞれの利用インタフェースは極力揃えており, 開発状況によって使用する通信機構を入れ替えたり, または利用時に動的に変更したりすることが可能である.

4.2.1 1 対 1 通信エージェント

1 対 1 通信エージェントは, Android Beam を用いて通信を開始する. Android Beam は, Android 端末間において, NFC を用いて情報を伝送するための規格である. 適切なパラメータを指定することで, 相手端末で特定のアプリケーションを起動することができ, また, 起動時にパラメータを付与することができる.

Android Beam は NFC で通信を行うが, Android 4.1 (API レベル 16) 以降であれば Bluetooth も併用することができる. しかし, 相互に情報を送り合うことができないため, Android Beam は Bluetooth 接続に必要な情報の伝送のみに利用し, そこで得られた情報を用いて Bluetooth 接続を別途行うことによって通信を行う.

スマートメイちゃんの起動と同時に提案機構が起動し, 1 対 1 通信エージェントも起動するが, 接続処理は端末同士を接触させ, Android Beam を用いて通信開始をするまで行われない. 端末同士を接触させると, 1 対 1 通信エー

*1 Wi-Fi アクセスポイントまでの距離.

ジェントが Bluetooth サーバを起動させ，接続を受け付ける準備を整える．

クライアント側の端末では，スマートメイちゃんが起動している必要はない．クライアント側でスマートメイちゃんがすでに起動している場合は，接続時に再起動される．

4.2.2 1 対多通信エージェント

1 対多通信エージェントは，メッセージを中継するメッセージサーバを介して通信を行う．メッセージサーバとの接続が確立すれば，同じメッセージサーバに接続しているすべてのクライアントと接続が可能である．

スマートメイちゃんの起動と同時に提案機構が起動し，1 対多通信エージェントも起動される．1 対多通信を中継する通信エージェントは，自身が起動されると自身の内部に記録されている IP アドレスへ通信を試みる．通信が確立しなかった場合，それ以降の処理を行わない．

通信が確立した場合，その通信を維持し続け，メッセージサーバからのプッシュ通知に備える．メッセージサーバとの通信は JSON を用いる．

5. 実験と評価

提案機構の有用性を調査するため，評価実験を行った．

5.1 複雑なシナリオの記述量に関する実験

3.1 節で示した問題点 (P1) 内容が変化するシナリオが困難，またその解決策である (S1) エージェントによるシナリオ記述支援機構 について，複雑なシナリオの構成が容易になったことを確認した．

5.1.1 実験条件

以下の返答を実装する星占いシナリオについて，FST スクリプトを用いて記述する従来手法と Java を用いて記述する提案手法の両者でそれぞれ実装を行った．

「星占い」と話しかけられた時 それでは占います．あなたの星座を教えて下さい，と返答する．

星座を話しかけられた時 今日の (星座) の運勢は (運勢)．ラッキーカラーは (色) です．と返答する．運勢に関する返答は，以下からランダムに選択される．

- 素晴らしいです．おめでとうございます．
- グッドです．
- 普通です．
- ちょっと悪いです．
- 残念．とても悪いです．

ラッキーカラーは 12 色から 1 色がランダムに選ばれる．

占いの結果は，毎日ランダムに変更され，また，その日 1 日は同じ結果を維持する．

このシナリオを，従来手法と提案機構で実装し，ステップ数及び循環的複雑度 [11] で評価を行う．

表 2 シナリオの複雑性に関する実験結果

	従来手法	提案手法
ステップ数	234	132
循環的複雑度	67	21

循環的複雑度は，コード中の分岐数を示すものである．分岐網羅テストを行う際のテストケース数の上界と等しい．一般的に，循環的複雑度が大きい場合，そのコードのデバッグやテストが困難となる．

5.1.2 実験結果と考察

実験結果を**表 2** に示す．実験結果より，提案手法は従来手法に比べ，ステップ数については約 43%削減され，循環的複雑度は約 68%削減された．

同じシナリオを実装したにもかかわらず，提案機構ではステップ数や循環的複雑度が減少している．これは以下の要因によるものだと考えられる．

条件分岐に関する記述の簡素化

シナリオ上で条件分岐を使用する場合，従来手法では複雑な記述が必要であり，また，分岐先の内容も冗長になっていた．提案手法ではより柔軟な記述が可能となったため，条件分岐に関する記述が簡素化されたほか，処理の共通化も行いやすくなった．これにより，ステップ数が減少した．

発話内容の動的組み立て

従来手法ではあらかじめすべてのパターンの対話文を網羅しておく必要があったのに対し，提案手法では発話する内容を動的に組み立てることが可能であり，条件分岐数が削減された．これにより，循環的複雑度が減少した．

この結果から，提案機構は従来機構に比べ複雑なシナリオの構築が容易になっていると言え，問題点 (P1) は解決されたと考えられる．

5.2 シナリオ記述支援機構に対する開発者視点での評価に関する実験

3.1 節で示した問題点 (P1) 内容が変化するシナリオが困難，またその解決策である (S1) エージェントによるシナリオ記述支援機構 について，提案機構が実装するシナリオ記述支援機構について開発者がどのような印象を持つか，アンケート調査を行った．

5.2.1 実験条件

提案機構と既存手法を用いて，「こんにちは」に対していくつかの応答を返すことができるシナリオを作成した．このシナリオを，MMDAgent を用いて対話したことがある被験者 22 人にそれぞれ見せ，シナリオ記述についてのアンケートを依頼した．

表 3　アンケート結果

	Q5	Q6	Q7	Q8	Q9	Q10
全体	4	4	4	4	2	5
FST+	3	4	3.5	2	1	5
FST-	4	4	4	4	3	5
Java+	4	4	4	4	2	5
Java-	4	4	4	4	2.5	5

アンケートの設問を以下に示す.

Q1　FST ファイルの文法についてどの程度知っていますか?

Q2　FST ファイルを使って MMDAgent の対話を記述した経験はどの程度ありますか?

Q3　Java プログラム言語の文法についてどの程度知っていますか?

Q4　Java を使ってプログラムを記述した経験はどの程度ありますか?

Q5　シナリオの内容が理解しやすかったのはどちらですか?

Q6　ソースコードの分量が少ないと感じるのはどちらですか?

Q7　シナリオにバグがあるとした場合，原因の特定が容易だと感じるのはどちらですか?

Q8　どちらかのシナリオに返答を追加するとき，どちらを使用したいですか?

Q9　新しく「簡単な」シナリオを書き起こす場合，どちらの言語を使用したいですか?

Q10　新しく「複雑な」シナリオを書き起こす場合，どちらの言語を使用したいですか?

Q1 から Q4 に関しては，ほとんどない〜かなりあるの 5 段階評価，Q5〜Q10 は FST〜Java の 5 段階評価を行った．また，各設問には適宜自由記述欄を設けた．

アンケート後，回答者全体から，Q1, Q2 のスコアが共に 4 以上の回答者を「FST に習熟しているグループ (FST+)」，それ以外の回答者を「FST に習熟していないグループ (FST-)」に分割した．また，同様に回答者全体から Q3, Q4 のスコアが共に 4 以上の回答者を「Java に習熟しているグループ (Java+)」，それ以外の回答者を「Java に習熟していないグループ (Java-)」に分割した．

5.2.2　実験結果と考察

提案機構に関する設問は Q5〜Q10 である．この部分について，それぞれのグループの中央値について**表 3**，**図 7** に示す.

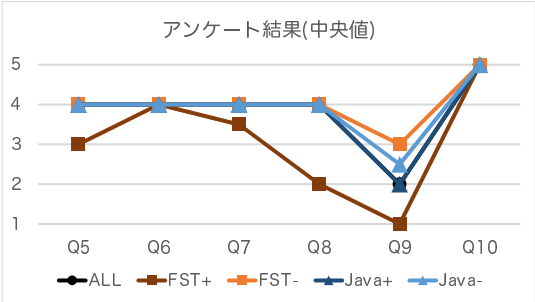

図 7　アンケート結果グラフ

概ね想定通りの結果となった．各設問は，中央値が 5 に近いほど提案機構に高評価となるが，ほぼすべての設問で FST に習熟しているか否かで中央値に差が生じているのに対し，Java への習熟度はそれほど影響を与えなかった．

Q5. シナリオの内容の理解しやすさについて

シナリオの内容の理解しやすさに関しては，FST に習熟しているユーザは評価が低いものの，それ以外のユーザは提案機構側が高評価であった．

Q6. ソースコードの分量について

提案機構側のソースコード量が若干少ないという評価となった．今回評価を依頼したシナリオはファイルサイズにそれほど差が生じていないものであったが，状態番号などの煩雑な記述がなく，すっきりした見た目をしている点が評価されたと考えられる．

Q7. シナリオのデバッグについて

提案機構側がより容易であるという評価が多い．Java では，開発環境やデバッガが利用可能であり，この点がデバッグに役立つとするコメントが多く，開発環境の差が評価されたと考えられる．

Q8. シナリオへの返答の追加について

FST に習熟しているか否かで大きく評価が分かれた．今回はシナリオの難易度が簡単すぎず，かつ，難しすぎないものであったため，どちらを選んでも記述難易度はほぼ中立的であった点が，評価が大きく分かれる結果に繋がったと考えられる．

Q9. 簡単なシナリオの作成について

提案機構が対象としない，簡単なシナリオの記述に関する設問である．想定通り，従来手法が優位となる結果となったが，FST に習熟していないグループは提案機構と従来手法で分かれる結果となった．コメントにおいて，Java を用いる場合は開発環境の準備やコンパイルが必要など，シナリオコストが大きいことが指摘されており，この点が Java に習熟しているグループからの評価が低い原因であると考えられる．

Q10. 複雑なシナリオの作成について

提案機構が対象とする，複雑なシナリオの記述に関する設問である．これも想定通り，提案機構が優位となる結果となった．Q9 では問題視された記述コストに関しても，そのデメリットを上回るメリットが得られるとして，提案機構を選択する回答者が多かった．

5.2.3 総括

提案機構は，条件分岐を多く含むような複雑なシナリオを作成する場合，既存手法より記述量の削減や記述性の面で有効であると示された．また，すでに作成されたシナリオの保守や再利用性に関しても，シナリオの可読性やデバッグの容易さの面で既存手法よりも優位性がある．

提案機構を用いてシナリオを作成する場合，インポート宣言やメソッド宣言などの記述が必要である点，シナリオ記述後にコンパイルが必要である点，デバッグ環境を整える必要がある点など，既存手法よりも開発にかかるコストが増加している点がある．提案機構の利用にかかるコストと提案機構によって生じるメリットのバランスが，提案機構の利用意欲に大きな影響を与えていると思われるため，利用を促進するためには開発にかかるコストをいかに低減するかが重要になると思われる．

6. 端末間通信シナリオにおけるコード量に関する実験

3.2 節で示した問題点 (P2) 通信を用いるシナリオの構成が困難，またその解決策である (S2) 端末間エージェント通信機構 について，通信を用いるシナリオの構成が容易になったことを確認した．問題点 (P2) 及び提案機構上で端末間通信を行うシナリオを構築する場合，端末間通信機構を用いることで，用いない場合と比べてどの程度記述量が削減されるかの調査を行った．

6.1 実験条件

提案機構を用いて情報交換を行うシナリオを，提案機構が提供する通信機構を用いる場合と，通信機構を用いず，シナリオエージェントに端末間通信処理を実装した場合それぞれで作成し，そのソースコード量について調査を行った．

作成したシナリオは，接続されている端末のうち 1 つの端末に「こんにちは」と話しかけた場合，接続されている他の端末において「こんにちは」と発話するシナリオである (**図 8**)．シナリオの実現には，接続されるすべての端末にこのシナリオエージェントを導入する必要がある．

端末間通信機構を利用しないシナリオでは，構造上，通信処理をいくつかの子クラスに分散して実装を行った．

接続方法は 1 対 1 型とした．

6.2 結果と考察

結果を**表 4**, **表 5** に示す．表 4 はシナリオを構成するソースコード全体 (提案機構や標準ライブラリで提供される部分を除く)，表 5 はエージェントとなるクラス及びそのクラスに含まれる無名クラスのソースコードのみの比較である．クラス数には，インナークラス及び無名クラス，インタフェース宣言を含む．文字数には空白文字を含まない．

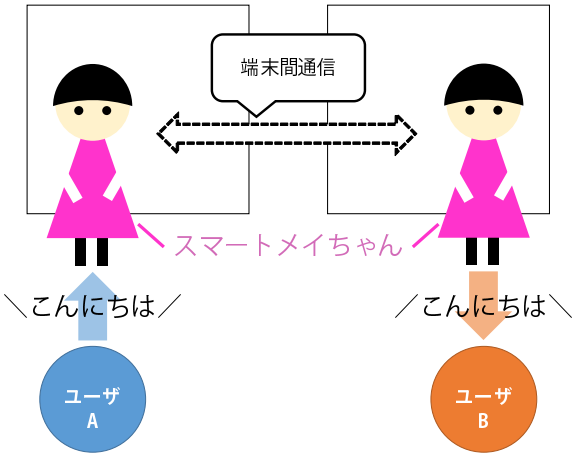

図 8 作成した端末間通信シナリオの概要

表 4 端末間通信シナリオ全体の比較

	通信機構あり	通信機構なし	削減率
文字数	1851	13392	86.2%
ステップ数	58	494	88.3%
クラス数	2	18	88.9%
ファイル数	1	5	80.0%

表 5 端末間通信シナリオエージェントの比較

	通信機構あり	通信機構なし	削減率
文字数	1851	3431	46.1%
ステップ数	58	110	47.3%
クラス数	2	4	50.0%

ステップ数やクラス数，ファイル数ともに 8 割以上のソースコードを削減することが出来ており，端末間通信機構がシナリオエージェントの記述量の削減に貢献していることが確認された．また，エージェントクラスと内包する無名クラスのみでの比較を行っても，半分近いコードが削減された．

提案機構では，通信をエージェント間通信に適したインタフェースへのマッピングまで行うため，利用するエージェントからはエージェント間メッセージの発行のみで完結させることができる．このため，提案機構側のシナリオエージェントではメッセージ処理に集中することができ，ソースコード量削減に繋がったと考えられる．

このように，通信を用いるシナリオの構成が用意になったことを確認でき，問題点 (P2) は解決したと考えられる．

7. まとめ

本稿では，本研究で用いるスマートフォン向け音声対話システムであるスマートメイちゃん及びその問題点につい

て説明し，これらの問題をエージェントを用いた拡張機構によって解決することを提案した．

提案機構であるエージェントを用いた拡張機構は，エージェントを用いたシナリオ記述機構と端末間エージェント通信機構を備え，エージェントによってシナリオ提供を行う環境を整備したものである．

構築したプロトタイプシステムを用いて複数の評価実験を行い，提案機構が問題点を解決することを確認した．

8. 今後の課題

本研究における今後の課題を以下に述べる．

8.1 開発環境の整備

提案したプロトタイプを用いてシナリオを開発する場合，Android アプリケーションを開発するための開発環境やライブラリ群のほか，エージェントを開発するためのライブラリも必要となる．また，シナリオエージェントのデバッグも Android 端末上で行う必要があるため，バグの特定や再現が困難な場合がある．

より実用的で開発者が利用しやすい機構とするためには，提案機構を利用しやすくする改良だけでなく，機構上でシナリオを構築・デバッグする環境の整備が必要である．

8.2 通信機構の改良

本研究において構築したプロトタイプシステムに含まれる通信機構は，1 対 1 通信機構は Android Beam を用いて相手と接続し，また，1 対多通信機構は予め決められたメッセージサーバへの接続を試みる設計となっている．

しかし，1 対 1 通信で利用している Android Beam は，通信を受け取る側のスマートメイちゃんが再起動されるほか，切断処理が構造上実装できない．また，1 対多通信において，メッセージサーバの接続先が固定されていることは拡張性を欠き，応用範囲が狭まるため望ましくない．よって，これらの通信機構の処理を見直し，より実用的なシステムとする必要がある．さらに，特に 1 対多通信の場合，ユーザが知らない間に通信を開始してしまわないようオプトアウトするインタフェースが必要になると思われる．

参考文献

[1] Apple Inc: Apple - iOS 8 - Siri(online)，入手先 ⟨http://www.apple.com/jp/ios/siri/⟩, (2015.02.17).
[2] 株式会社 NTT ドコモ: しゃべってコンシェル (online)，入手先 ⟨https://www.nttdocomo.co.jp/service/information/shabette_concier/⟩, (2015.02.17).
[3] 李晃伸，大浦圭一郎，徳田恵一: 魅力ある音声インタラクションシステムを構築するためのオープンソースツールキット MMDAgent，電子情報通信学会技術研究報告，pp.159-164 (2011).
[4] 山本大介，大浦圭一郎，西村良太，打矢隆弘，内匠逸，李晃伸，徳田恵一: スマートフォン単体で動作する音声対話 3D エージェント「スマートメイちゃん」の開発，電子情報通信学会技術研究報告，pp.159-164 (2011).
[5] Google Inc: Android Developers(online)，入手先 ⟨http://developer.android.com/⟩, (2015.02.17).
[6] Akinobu Lee and Tatsuya Kawahara: Recent development of open-source speech recognition engine julius, APSIPA, pp.131-137 (2009).
[7] 大浦 圭一郎, 酒向 慎司, 徳田 恵一: 日本語テキスト音声合成システム Open JTalk, 日本音響学会春季講論集, Vol.1, No.2-7-6, pp. 343-344 (2010).
[8] Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood: developing multi-agent systems with JADE, John Wiley & Sons, Ltd (2007).
[9] Giovanni Caire and Federico Pieri: LEAP USER GUIDE(online), 入手先 ⟨http://jade.tilab.com/doc/tutorials/LEAPUserGuide.pdf⟩, (2015.02.17).
[10] Google Inc: Beaming NDEF Messages to Other Devices — NFC Basics — Android Developers，入手先 ⟨http://developer.android.com/guide/topics/connectivity/nfc/nfc.html⟩, (2015.02.17).
[11] THOMAS J. McCABE: A Complexity Measure, IEEE TSE, vol. SE-2, pp.308-320 (1976).