情報処理学会論文誌　デジタルコンテンツ　Vol.8　No.2　1–15 (Aug. 2020)

推薦研究論文

ブロック型ビジュアルプログラミング機能を有する音声対話シナリオ編集システム

古市 瑞希[1,a)]　山本 大介[1,b)]　高橋 直久[1,c)]

受付日 2020年1月15日, 採録日 2020年5月25日

概要：名古屋工業大学で開発された音声インタラクションシステム構築ツールキット MMDAgent は，FST（Finite State Transducer）形式で記述された対話シナリオに基づいて音声対話の処理を実行し，ユーザは対話シナリオを編集することで対話の内容を自由に構成することができる．しかし，FST 形式は一般ユーザには馴染みがなく，対話シナリオを編集することは困難である．そこで，一般ユーザでも対話シナリオを編集できるようにするため，ブロック型ビジュアルプログラミング機能を有する音声対話シナリオ編集システムを提案する．提案システムでは，対話の内容を「ブロック」で表し，パズルのように組み合わせるだけで対話シナリオを作成できる．これにより，対話シナリオについて知識がない一般ユーザでも，対話シナリオを作成することができる．また，提案システムに基づきプロトタイプシステムを実装し，評価実験を行った．その結果，テキストで記述する方法に比べて，提案システムを用いることで短い時間で対話シナリオの作成や内容の把握ができることが分かった．

キーワード：音声対話システム，ビジュアルプログラミング，対話シナリオ編集システム

Voice Interaction Scenarios Editor with Block-based Visual Programming Facilities

Mizuki Furuichi[1,a)]　Daisuke Yamamoto[1,b)]　Naohisa Takahashi[1,c)]

Received: January 15, 2020, Accepted: May 25, 2020

Abstract: MMDAgent, a toolkit for builging voice interaction systems developed at Nagoya Institute of Technology, executes voice dialogue based on voice interaction scenarios in the form of Finite State Transducer, and allows the user to create a dialogue by editing the voice interaction scenario. However, the FST format is unfamiliar to general users, and it is difficult to edit voice interaction scenarios. Therefore, in order to enable general users to edit voice interaction scenarios, we propose a voice interaction scenarios editeor with Block-based visual programming facilities. In the proposed system, a voice interaction scenario can be created simply by expressing the contents of the dialog as “blocks” and combining them like a puzzle. As a result, even a general user who has no knowledge of the voice interaction scenario can create a voice interaction scenario. We also implemented a prototype system based on the proposed system and performed evaluation experiments. As a result, it was found that the proposed system can create voice interaction scenario and enable the user to grasp the contents in a short time compared to the method described in text.

Keywords: voice interaction systems, visual programming, voice interaction scenarios editor

1. はじめに

近年，人の代わりに接客や情報発信を行うことを目的と

[1] 名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, Aichi 466–8555, Japan
[a)] m.furuichi.857@stn.nitech.ac.jp
[b)] yamamoto.daisuke@nitech.ac.jp
[c)] naohisa@nitech.ac.jp

本論文の内容は 2019 年 7 月 3 日の DICOMO2019 シンポジウムにて報告され，DCC 研究会主査/プログラム委員長により DCON への掲載が推薦された論文である．

図 1　名古屋工業大学の正門前に設置されている MMDAgent を搭載したデジタルサイネージ

Fig. 1　Digital signage equipped with MMDAgent installed in front of the main gate of Nagoya Institute of Technology.

する，音声対話システムを搭載したデジタルサイネージの設置が増加してきている．名古屋工業大学にも，**図 1** のように，正門前に音声対話システム MMDAgent [1] を搭載したデジタルサイネージが設置されており，学内のイベント情報の掲示や，学内施設への経路案内などを行っている．

MMDAgent とは，名古屋工業大学国際音声言語技術研究所で開発された音声インタラクションシステム構築ツールキットであり，音声認識，音声合成，3D モデルの描画や制御を統合したシステムである．MMDAgent は，FST（Finite State Transducer）形式で記述された対話シナリオに基づいて音声対話の処理を実行する．ユーザは対話シナリオを編集することで，対話の内容を自由に構成することができる．しかし，一般ユーザは FST 形式に馴染みがなく，対話シナリオを編集するためには FST 形式の構文を学習する必要がある．つまり，一般ユーザが対話シナリオを編集することは困難であるといえる．

馬場ら [2] は，個人によるものづくり「パーソナルファブリケーション」が注目を浴びていることを指摘している．個人がそれぞれの動機を原動力として，自身の趣味や自由な時間を活用し，ものづくりを行うことで，結果として多様なユーザをターゲットとしたものづくりが推し進められる．このことは MMDAgent についても当てはまる．MMDAgent はあらかじめ登録された対話シナリオに沿った動作しかできないため，ユーザが取得したい情報に関する対話シナリオが登録されていない場合，ユーザはシステムから情報を得ることができない．そこで，一般ユーザが対話シナリオの編集をできるようにすることで，ユーザは個人の目的のために対話シナリオを作成でき，対話シナリオのバリエーションを増やすことができる．これにより，多様なユーザの要求に応えることができるシステムになると同時に，ユーザの音声対話システムの利用機会を増やすこともできる．

そこで我々は，一般ユーザによる対話シナリオの編集を可能にする，ブロック型ビジュアルプログラミング機能を有する音声対話システムの対話シナリオ（以下，「音声対話シナリオ」）編集システムを開発している [3]．ビジュアルプログラミングとは，プログラムをテキストで記述するのではなく，視覚的なオブジェクトを用いて行うプログラミングであり，プログラミングの知識がない人でも簡単にプログラムを作ることができる．近年では，2020 年に小学校でプログラミング教育が必修化されることもあり，Scratch [4] や viscuit [5] などの子ども向けのビジュアルプログラミング言語が普及している．本論文では，ビジュアルプログラミングエディタを構築するための JavaScript ライブラリである Google Blockly [6] を用いて，対話の内容を「ブロック」で表し，パズルのように組み合わせることで対話シナリオを作成することができるシステムを提案する．これにより，FST 形式の構文についての知識がない人でも，対話シナリオを作成することができる．また，FST 形式の記述に慣れているユーザであっても，テキストで記述するのではなく，ブロックで対話シナリオを作成することで，記述ミスが減り，対話シナリオの把握が容易になる．

以下，2 章で関連研究について述べる．3 章で FST 形式の対話シナリオについて詳しく説明する．4 章で提案システムの要求仕様と詳細について述べ，5 章で提案システムの実現法について述べる．6 章で評価結果の報告と考察を行う．7 章で本論文をまとめる．

2.　関連研究

音声対話システムの編集を簡単にすることを目的とした研究は他にも存在している．MMDAgent の対話シナリオの編集方法としては，MMDAE [7], [8] や EFDE [9], [10] が提案されている．MMDAE は，Web ブラウザを用いたシナリオエディタで，対話シナリオを見やすく，編集しやすくし，より簡単に扱えるようにするという点では本研究と同様であるが，ユーザは FST 形式の知識がないと使用できないという点で本研究とは異なる．一方，EFDE は，タブレット端末のタッチ操作で状態遷移図を描くことにより MMDAgent の対話シナリオを作成できるシステムである．対話シナリオをテキストで記述するのではなく，新しい記述方法を提案している点は本研究と同様であるが，一般ユーザにとって状態遷移図はあまり馴染みがなく，一般ユーザ向けのシステムではないと考える．

MMDAgent 以外の音声対話システムを対象としたものとしては，pepper [11] の動作をカスタマイズできるプログラミングツールとして Choregraphe [12] がある．pepper を動かすための機能がつまった「ボックス」と呼ばれるモジュールがあらかじめ用意されており，ユーザはそれらを並べて，線でつなぎ合わせるだけで，自分だけの pepper を作ることができる．このプログラミング方法は，前述の EFDE に似ており，グラフ構造に慣れた人には分かりやすい方法であるといえる．しかし，一般ユーザにとっては，ボックスを選び，さらに線でつなぐという操作が多いこと

で，利用方法を間違えたり，利用を諦めてしまう可能性が高くなる．また，ボックスの配置はユーザが自由に決めることができるので，ユーザが思いついたままにボックスを配置し，線でつないだ場合，可読性の低いプログラムになってしまう恐れがある．これに対し，本研究では，ユーザの操作をできるだけ減らし，他の人が作った対話シナリオでも内容の把握がしやすいシステムを目指す．

また，バーチャルキャラクタと音声の対話を楽しむための対話プラットフォームであるコミュクラフト [13] がある．コミュクラフトでは，対話シナリオをビジュアル操作で簡単に作成することができる．しかし，用語が専門的で，ふだんの生活では聞きなれない言葉が多い．これに対し，本研究では，子どもでも分かる，日常生活で使われる言葉を使ったプログラミング環境の構築を目指す．

音声対話システムの編集以外にも，ビジュアルプログラミングを用いて操作を簡単にすることを目的とした研究は多く存在している．光永ら [14] は，マイコン用ビジュアルプログラミング環境を提案している．キーボード操作に不慣れな児童や生徒でもプログラミングを楽しむことができるよう，タブレット端末を用いてタッチパネルによる操作を提案している．これに対し，本研究では，対話シナリオ作成時に対話のなかでのユーザやエージェントの発話文の入力が必要なため，キーボード入力をなくすことはできない．しかし，プルダウンメニューを用いるなど，ユーザのキーボード操作をできるだけ減らす工夫を行っている．

他にも，杉村ら [15] は，ECHONET Lite [16] 対応家電が普及し，各家庭で動作のカスタマイズが行われることを想定して，一般ユーザでも簡単にカスタマイズできるようECHONET Lite 向けのビジュアルプログラミング環境を提案している．Google Blockly をベースに開発している点は本研究と同様であるが，杉村らの研究では家電操作を対象としており，FST 形式で表現できるより汎用的な音声対話シナリオを対象としている本研究とは異なる．また，Repl-AI [17] では，フローチャートを用いて，チャットボットを簡単に作ることができる．これは，ユーザの発話もしくはシステムの発話の制御に特化したシステムになっている．これに対し，本研究では，フローチャートではなく，プログラミング初心者に広く支持されている Scratch のベース技術である Google Blockly を採用している．

また，一般ユーザによるアプリケーションの作成を可能にする方法として，大村 [18] は，自然言語を用いて，ホームネットワークアプリケーションの作成を行う手法を提案している．自然言語を用いてプログラミングできるようにすることで，情報技術に馴染みがないユーザへのプログラミングの敷居の高さを低減している．大村 [18] のシステムでは，ルール発動の契機となるイベント（Event），その際に評価される条件（Condition），および，そのときの動作（Action）からなる ECA ルールで動作するデバイスの編集を対象としており，FST 形式の対話シナリオの編集を対象としている本研究とは異なる．

3. FST 形式の対話シナリオ

FST 形式の対話シナリオの例を**図 2** に示す．FST 形式の構文は，状態番号，次状態番号，遷移条件コマンド，出力コマンドを半角スペースを区切り文字として，一行に記述し（「文」という），複数の文を系列的に記述することで対話シナリオとする．無条件遷移や無出力の場合には<eps>と記述する．また，FST 形式の対話シナリオは**図 3** のように状態遷移図で表すことができる．

図 2 の状態番号が 0 のような状態を初期状態といい，MMDAgent が動作していないときの状態である．図 2 の 1 行目は，状態 0（初期状態）であるときに，音声認識によって「こんにちは」と認識すると，無出力で次状態 10 へ遷移するということを表す．図 2 の対話シナリオは，音声認識によって「こんにちは」と認識すると，MMDAgent のキャラクタ型エージェント「メイちゃん」がお辞儀の動作をし，「こんにちは」と発話するという流れである．

そして，MMDAgent のユーザは FST 形式の対話シナリオを編集することで，対話の内容を自由に構成することができる．しかし，FST 形式は状態番号，次状態番号，遷移条件コマンド，出力コマンドからなる文を系列的に記述する形式であり，自由度の高い表現形式である一方で，一般ユーザにとっては馴染みがない形式である．そのため，一般ユーザが対話シナリオを編集するためには FST 形式の対話シナリオについて学習する必要がある．また，FST 形式の対話シナリオは図 2 のように，テキスト形式で記述されるため，FST 形式の記述に慣れているユーザであっても，記述のミスが起きたり，対話内容の把握が困難であるといえる．

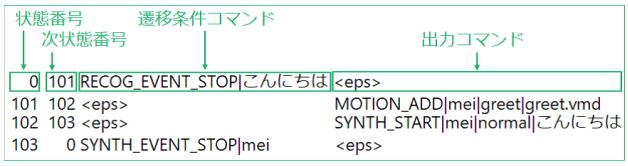

図 2　FST 形式の対話シナリオの例
Fig. 2　Example of FST-style interaction scenario.

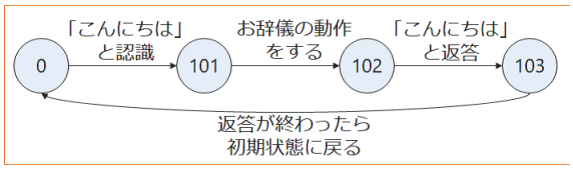

図 3　図 2 の対話シナリオを表す状態遷移図
Fig. 3　State transition diagram showing the interaction scenario in Fig. 2.

4. 提案システム

本研究では，3 章で述べた FST 形式の対話シナリオの問題点に対して，対話シナリオの編集を簡単にすることを目的とし，ブロック型ビジュアルプログラミング機能を有する音声対話シナリオ編集システムを提案する．ユーザは対話の内容が書かれたブロックをパズルのように組み合わせることで対話シナリオを作成することができる．

提案システムの要求仕様は以下の 3 つである．

要求仕様 1　FST 形式の構文について知識がない一般ユーザでも対話シナリオを作成することができる．

要求仕様 2　対話シナリオの記述のミスを減らす．

要求仕様 3　対話シナリオの把握を容易にする．

提案システムの実現には，ブロックを用いて対話シナリオを作成し，ブロックの対話シナリオから FST 形式の対話シナリオに変換する必要がある．そこで，提案システムでは，新たに「命令ブロック」と「構造ブロック」を定義し，「対話シナリオ変換機能」を実装する．提案システムの構成図は**図 4** である．提案システムのブロックと機能の詳細は，それぞれ 4.1 節，4.2 節，4.3 節で述べる．

これらのブロックや機能を用いた，提案システムの要求仕様に対する具体的なアプローチは以下である．

要求仕様 1　FST 形式の構文について知識がない一般ユーザでも対話シナリオを作成することができる．

アプローチ 1-1　対話の内容を表す「命令ブロック」を定義する．対話シナリオの作成において，一般に馴染みがない FST 形式の遷移条件コマンドや出力コマンドを用いるのではなく，対話の内容を自然言語で表した「命令ブロック」を用いることで，対話シナリオについて事前に学習したり，コマンドを覚えたりすることなく，対話シナリオを作成することができる．また，対話の内容を自然言語を用いて表すことで，大村 [18] の研究と同様に，情報技術に馴染みのないユーザに対してプログラミングの敷居の高さを低減することができると考える．

アプローチ 1-2　命令ブロックや構造ブロックを用いた対話シナリオの作成において，FST 形式のように遷移条件と出力を対にして記述するのではなく，音声対話システムのエージェントとユーザの「対話」に沿って記述できるようにする．対話を書き起こすように対話シナリオを作成できることで，一般ユーザでも簡単に対話シナリオが作成できる．

要求仕様 2　対話シナリオの記述のミスを減らす．

アプローチ 2-1　FST 形式の構造とブロック型ビジュアルプログラミング言語の構造を対応付ける「構造ブロック」を定義し，命令ブロックや構造ブロックにより，ブロックを組み合わせて対話シナリオを作成できるようにする．これにより，対話シナリオ作成時のテキスト入力が減るため，記述のミスも減る．

アプローチ 2-2　対話シナリオ変換機能で，状態番号を自動で付与する．これにより，状態番号の計算ミスや記述ミスがなくなり，エラーのない対話シナリオの作成を助ける．

要求仕様 3　対話シナリオの把握を容易にする．

アプローチ 3-1　命令ブロックをカテゴリごとに色分けする．対話の中で似たような内容を表すものを 1 つのカテゴリとし，色分けすることで，ユーザは使いたいブロックを探しやすくなり，また，対話シナリオの把握も容易になる．

アプローチ 3-2　繰り返し利用される定型的な対話シナリオを関数としてまとめて，再利用できるようにする構造ブロック（関数ブロック）を定義する．これにより，全体のブロックの数を減らすことができ，対話シナリオの可読性が向上する．

以下，提案システムのブロックと機能について説明する．

4.1 命令ブロック

命令ブロックとは，対話の内容を表すブロックである．

図 5 の遷移条件コマンドや**図 6** の出力コマンドに対応するブロックを**図 7**，**図 8** のように定義する．対話の内容は自然言語を用いて表す（アプローチ 1-1）．また，図 7 のようにユーザの発話に関するブロックは青色で，図 8 のよ

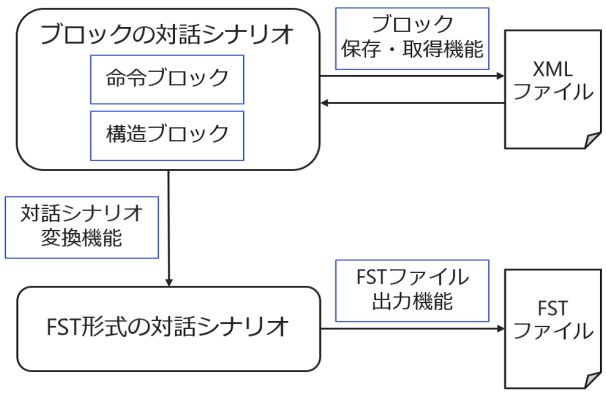

図 4　提案システムの構成図

Fig. 4　The structure of the proposed system.

```
RECOG_EVENT_STOP|こんにちは <eps>
```

図 5　「こんにちは」と音声認識する遷移条件コマンド

Fig. 5　The condition command to accept the voice recognition as “Hello”.

```
<eps> SYNTH_START|mei|mei_voice_normal|こんにちは
```

図 6　「こんにちは」と発話する出力コマンド

Fig. 6　The output command to output the speech as “Hello”.

図 7　図 5 のコマンドに対応する命令ブロック
Fig. 7　The instruction block corresponding to the command in Fig. 5.

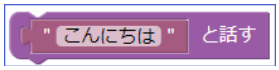

図 8　図 6 のコマンドに対応する命令ブロック
Fig. 8　The instruction block corresponding to the command in Fig. 6.

図 9　UNIT ブロック
Fig. 9　The UNIT Block.

うにエージェントの発話に関するブロックはピンク色で定義する（アプローチ 3-1）．

4.2　構造ブロック

構造ブロックとは，ブロックを組み合わせて対話シナリオを作成するために必要となるブロックである．

Google Blockly は，ブロックから JavaScript や Python など，分岐やサブルーチンなどの制御構造を持つプログラミング言語への変換を対象としており，FST 形式はそれらの言語の構造とは異なる．そのため，Google Blockly の既存のブロックや命令ブロックを組み合わせるだけでは，対話シナリオを作成することができない．そこで，FST 形式の構造に対応するために，4 つの構造ブロックを提案する．そして，この 4 つの構造ブロックと 4.1 節の命令ブロックにより，ブロックを用いて対話シナリオを作成するシステムを提案する（アプローチ 2-1）．またこのとき，音声対話システムの「対話」に沿って記述できるようにする（アプローチ 1-2）．

4.2.1　UNIT ブロック

UNIT ブロックを**図 9** に示す．

UNIT ブロックとは，一連の対話シナリオを，1 つのブロック群「UNIT」としてまとめるブロックである．UNIT の例を**図 10** と**図 11** に示す．ここで，「メイちゃん」とはエージェントを表す．提案システムでは，UNIT ブロックの「開始」の横に入力された数値より，初期状態から遷移する次状態番号を求める．よって，複数の UNIT ブロックを用いて，異なる数値を入力した UNIT を作成することで，初期状態から遷移可能な複数の対話シナリオを作成することができる．つまり，図 10 と図 11 の UNIT を作成した場合，ユーザが「こんにちは」と話しかけたら，エージェントが「こんにちは」と応答する対話シナリオと，ユーザが「こんばんは」と話しかけたら，エージェントが「こんばんは」と応答する対話シナリオを作成することができる．なお，同じ数値の UNIT があった場合には，提案システムでは初期状態から同じ次状態番号へ遷移する FST 形式の文が複数出力され，MMDAgent が動作するときに遷移条件に応じて状態遷移を行う．

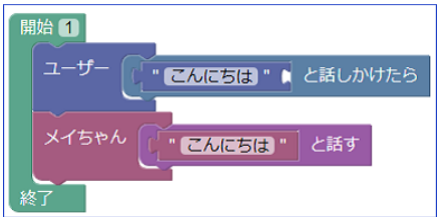

図 10　UNIT の例 1
Fig. 10　Example 1 of UNIT Block.

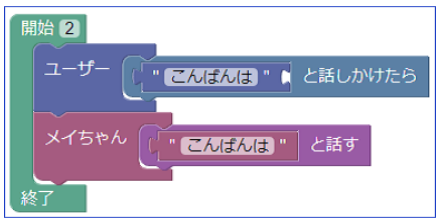

図 11　UNIT の例 2
Fig. 11　Example 2 of UNIT Block.

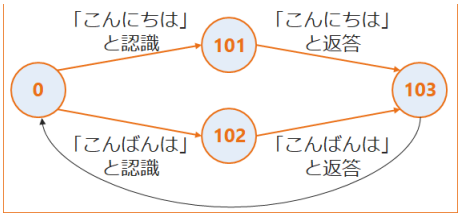

図 12　分岐の状態遷移図の例
Fig. 12　Example of branch state transition diagram.

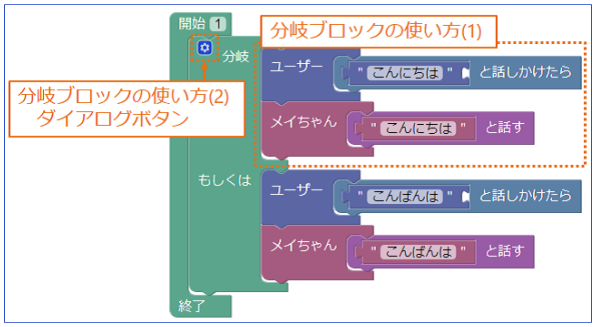

図 13　図 12 の対話シナリオを表したブロック
Fig. 13　Block showing the interaction scenario in Fig. 12.

4.2.2　分岐ブロック

分岐ブロックとは，**図 12** の状態遷移図で示すように，ある状態から 2 つ以上の異なる状態へ分かれる状態遷移の構造に対応するブロックである．図 12 の状態遷移をブロックを用いて表すと，**図 13** のようになる．

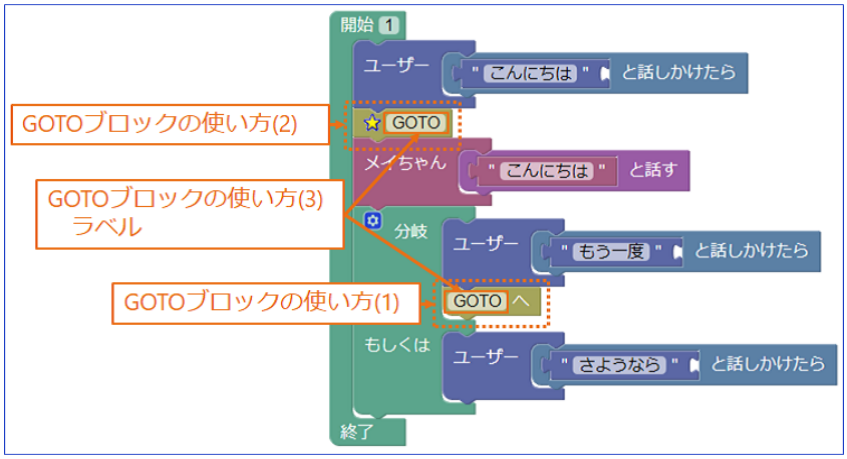

図 15　図 14 の対話シナリオを表したブロック
Fig. 15　Block showing the interaction scenario in Fig. 14.

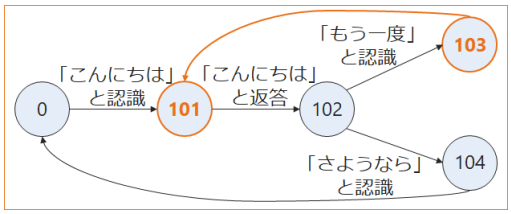

図 14　GOTO の状態遷移図の例
Fig. 14　Example of GOTO state transition diagram.

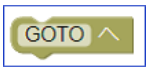

図 16　GOTO ブロック
Fig. 16　The GOTO Block.

図 17　GOTO_STAR ブロック
Fig. 17　The GOTO_STAR Block.

分岐ブロックの使い方は以下である．

(1) 分岐先の対話シナリオを分岐ブロックの間に挿入する．
(2) 分岐ブロックの初期形状は図 13 のように二股であるため，3 つ以上の対話シナリオに分岐する場合には，分岐ブロックの左上の青い「ダイアログボタン」を押して，分岐の数を変更する．

4.2.3　GOTO ブロック

GOTO ブロックとは，**図 14** の状態遷移図で示すように，任意の状態へ無条件で遷移する構造に対応するブロックである．図 14 の状態遷移をブロックを用いて表すと，**図 15** のようになる．

GOTO ブロックと GOTO_STAR ブロックに入力された文字列をラベルといい，GOTO ブロックと GOTO_STAR ブロックはラベルで対応付けをする．

GOTO ブロックの使い方は以下である．

(1) GOTO 開始位置に**図 16** の「GOTO ブロック」を挿入する．
(2) GOTO 先に**図 17** の「GOTO_STAR ブロック」を挿入する．
(3) GOTO ブロックと GOTO_STAR ブロックに同じラベルを付ける．図 15 では，「GOTO」というラベルを付けている．

ここで，GOTO ブロックについては，適切に接続しないと無限ループになる恐れもある．そのため，提案システムを一般ユーザが使用する場合には，GOTO ブロックの正しい使い方や無限ループの説明が必要であると考える．また，提案システムではユーザに応じて使用するブロックを選択することができるので，使用方法などの説明が難しい場合には，GOTO ブロックを使用しない対話シナリオ編集システムを用いる．

4.2.4　関数ブロック

関数ブロックとは，**図 18** の状態遷移図で示すように，繰り返し利用される定型的な対話シナリオを関数としてまとめて，再利用できるようにするブロックである（アプローチ 3-2）．図 18 の状態遷移図をブロックを用いて表すと，**図 19** のようになる．

FUNCTION ブロックと FUNCTION_CALL ブロック，FUNCTION_CALL ブロックと ARGMENT ブロックは，GOTO ブロックと同様，ラベルで対応付けをする．

関数ブロックの使い方は以下である．

(1) 関数にする対話シナリオを**図 20** の「FUNCTION ブロック」の間に挿入し，関数にラベルを付ける．
(2) 関数を呼び出す場所に**図 21** の「FUNCTION_CALL ブロック」を挿入する．
(3) 関数が引数を持つ場合には，FUNCTION ブロックのダイアログボタンを押して，引数の数を変更し，引数にラベルを付ける．引数の数を 1 つにした場合には，FUNCTION_CALL ブロックの形が**図 22** に変化する．

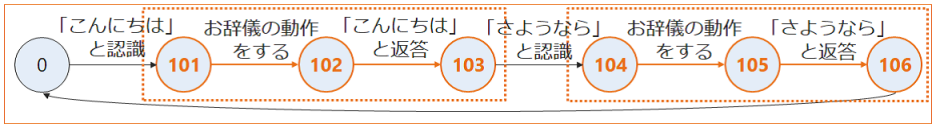

図 18　関数の状態遷移図の例
Fig. 18　Example of function state transition diagram.

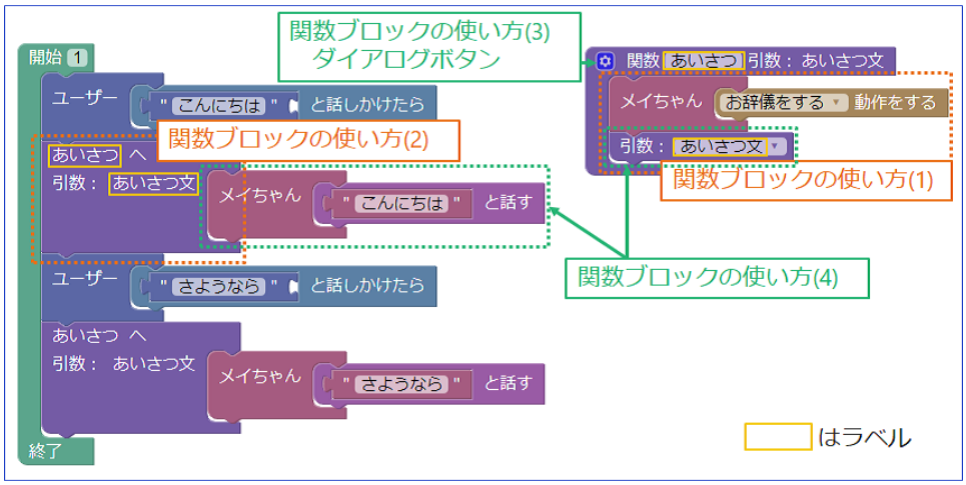

図 19　図 18 の対話シナリオを表したブロック
Fig. 19　Block showing the interaction scenario in Fig. 18.

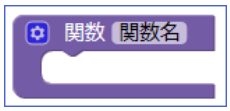

図 20　FUNCTION ブロック
Fig. 20　The FUNCTION Block.

図 21　FUNCTION_CALL ブロック
Fig. 21　The FUNCTION_CALL Block.

図 22　FUNCTION_CALL ブロック（引数を持つ場合）
Fig. 22　The FUNCTION_CALL Block (With arguments).

図 23　ARGMENT ブロック
Fig. 23　The ARGMENT Block.

(4) 引数にするブロックを図 22 の「FUNCTION_CALL ブロック」の間に挿入し，関数のなかで引数を呼び出す場所に**図 23** の「ARGMENT ブロック」を挿入する．

提案システムでは，引数として値ではなく，ブロックが渡される．このようにすることで，複数個のブロックを引数として渡すことができる．

4.3　対話シナリオ変換機能

対話シナリオ変換機能とは，ブロックで作成した対話シナリオを FST 形式の対話シナリオに変換する機能である．ブロックの組合せに応じて，状態番号と次状態番号を計算し，FST 形式の文を出力する（アプローチ 2-2）．

4.4　FST ファイル出力機能

FST ファイル出力機能とは，対話シナリオ変換機能で出力された FST 形式の対話シナリオを，MMDAgent が受理可能な形式である FST ファイルに出力する機能である．

4.5　ブロック保存・取得機能

ブロック保存機能とは，ブロックの対話シナリオを xml 形式に変換し，ファイルに出力する機能である．また，ブロック取得機能とは，指定した XML ファイルから，xml 形式で記述されたブロックの情報を取得し，ブロックを復元する機能である．これらの機能により，ユーザは一度作ったブロックの対話シナリオを再利用することができる．

5.　提案システムの実現法

本章では，提案システムの実現法について述べる．

5.1　命令ブロック

Blockly Developer Tools [19] を使用し，命令ブロックを作成する．Blockly Developer Tools とは，Google が提供する Blockly で使用するブロックを簡単に作成できるツールである．ブロックの名前や形，色などを決めると，ブ

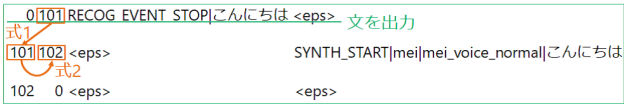

図 24　図 10 のブロックから変換され，出力される FST 形式の対話シナリオ（命令ブロックの計算式）

Fig. 24　The interaction scenario in FST format converted and output from the block in Fig. 10 (Formula for instruction Block).

ロックのプレビューと JavaScript 形式のブロックの定義が出力される．

5.2　構造ブロック

命令ブロックと同様に，Blockly Developer Tools を使用し，構造ブロックを作成する．

UNIT ブロックには状態番号の計算で用いる数値を入力するための数値入力欄を，GOTO ブロックと関数ブロックにはラベルを入力するための文字列入力欄を設ける．

また，分岐ブロックと関数ブロックでは，Google Blockly の Mutators 機能 [20] を利用する．Mutators 機能とは，ユーザがブロックの形状を自由に変更できる機能である．提案システムでは，分岐ブロックの分岐の数と，関数ブロックの引数の数をユーザが自由に変更することができるようにする．

5.3　対話シナリオ変換機能

ブロックの接続に応じて状態番号を計算し，ブロックから FST 形式の対話シナリオに変換する．変換方法は，上から順番にブロックを認識し，そのブロックに対応した FST 形式の文の出力や，状態番号の計算を行う．これを繰り返すことで，FST 形式の対話シナリオを生成する．このとき，システムが管理するデータは，変換しているブロックの状態番号と次状態番号，GOTO ブロックや関数ブロックで使用するラベルのシンボルと状態番号の対応表，関数や引数の呼び出し回数である．

以下のように，変換方法の詳細はブロックの種類によって異なる．

5.3.1　命令ブロック

図 10 のブロックから変換され，出力される FST 形式の対話シナリオを**図 24** に示す．

命令ブロックでは，そのブロックに対応した FST 形式の文を出力した後，次のブロックの状態番号を式 (1)，次状態番号を式 (2) で求める．

$$\text{次のブロックの状態番号} = \text{次状態番号} \tag{1}$$

$$\text{次のブロックの次状態番号} = \text{次のブロックの状態番号} + 1 \tag{2}$$

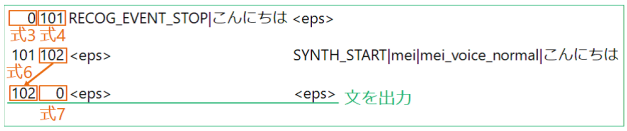

図 25　図 10 のブロックから変換され，出力される FST 形式の対話シナリオ（UNIT ブロックの計算式）

Fig. 25　The interaction scenario in FST format converted and output from the block in Fig. 10 (Formula for UNIT Block).

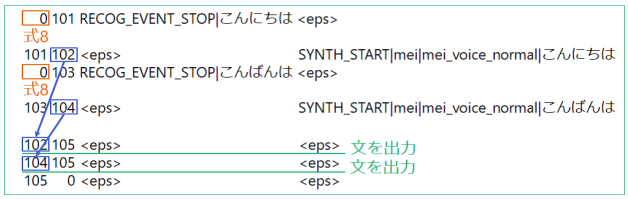

図 26　図 13 のブロックから変換され，出力される FST 形式の対話シナリオ（分岐ブロックの計算式）

Fig. 26　The interaction scenario in FST format converted and output from the block in Fig. 13 (Formula for branch Block).

5.3.2　UNIT ブロック

図 10 のブロックから変換され，出力される FST 形式の対話シナリオを**図 25** に示す．

UNIT の最初のブロックの状態番号を式 (3)，次状態番号を式 (4) で定義する．ここで UNIT 番号とは，1 つの UNIT で想定されるブロックの最大数を Block_Max としたとき，UNIT ブロックの「開始」の横に入力された数値から，式 (5) で定義する．なお，今回の提案システムでは，Block_Max = 100 とした．

$$\text{状態番号} = 0 \tag{3}$$

$$\text{次状態番号} = \text{UNIT 番号} + 1 \tag{4}$$

$$\text{UNIT 番号} = \text{入力された数値} \times \text{Block\_Max} \tag{5}$$

また，UNIT の最後では，状態番号を式 (6)，次状態番号を式 (7) で定義し，無条件・無出力遷移の文を出力する．

$$\text{状態番号} = \text{前のブロックの次状態番号} \tag{6}$$

$$\text{次状態番号} = 0 \tag{7}$$

5.3.3　分岐ブロック

図 13 のブロックから変換され，出力される FST 形式の対話シナリオを**図 26** に示す．

分岐ブロックを認識すると，すべての分岐先の最初のブロックの状態番号を式 (8) で定義し，分岐先のブロックの変換を行う．

$$\text{状態番号} = \text{分岐ブロックの前のブロックの次状態番号} \tag{8}$$

そして，分岐の最後では，すべての分岐先の最後のブロッ

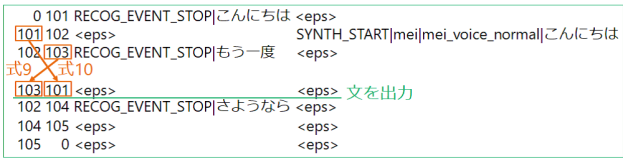

図 27　図 15 のブロックから変換され，出力される FST 形式の対話シナリオ（GOTO ブロックの計算式）

Fig. 27　The interaction scenario in FST format converted and output from the block in Fig. 15 (Formula for GOTO Block).

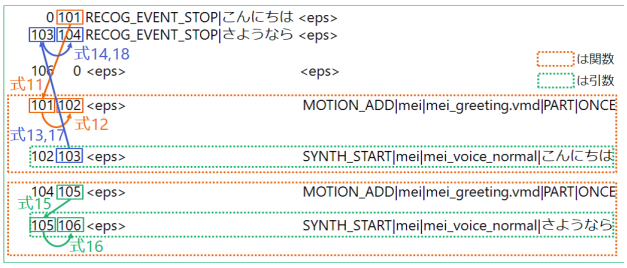

図 28　図 19 のブロックから変換され，出力される FST 形式の対話シナリオ（関数ブロックの計算式）

Fig. 28　The interaction scenario in FST format converted and output from the block in Fig. 19 (Formula for function Block).

クの次状態番号を状態番号とし，ある次状態番号へ無条件・無出力遷移する文を出力する．その後，命令ブロックの文出力後と同様に，式 (1)，式 (2) で状態番号を求める．

5.3.4　GOTO ブロック

図 15 のブロックから変換され，出力される FST 形式の対話シナリオを**図 27** に示す．

GOTO ブロックを認識すると，状態番号を式 (9)（前のブロックで計算済み），次状態番号を式 (10) の無条件・無出力遷移の文を出力する．なお，GOTO ブロックに対して，ラベルが完全一致する GOTO_STAR ブロックがある場合にのみ，状態番号が正しく付与される．

状態番号 = 前のブロックの次状態番号　(9)

次状態番号 = 同じラベルの GOTO_STAR ブロックの状態番号　(10)

5.3.5　関数ブロック

図 19 のブロックから変換され，出力される FST 形式の対話シナリオを**図 28** に示す．

FUNCTION ブロックを認識すると，FUNCTION ブロックに挟まれた最初のブロックの状態番号を式 (11)，次状態番号を式 (12) で求め，関数のブロックの変換を行う．

状態番号 = 同じラベルの FUNCTION_CALL ブロックの状態番号　(11)

次状態番号 = 状態番号 + 1　(12)

また，FUNCTION_CALL ブロックを認識すると，FUNCTION_CALL ブロックの次のブロックの状態番号を式 (13)，次状態番号を式 (14) で求める．

状態番号 = 同じラベルの FUNCTION ブロックの最後のブロックの次状態番号　(13)

次状態番号 = 状態番号 + 1　(14)

関数が引数を持つ場合には，ARGMENT ブロックを認識すると，引数となるブロックの最初のブロックの状態番号を式 (15)，次状態番号を式 (16) で求め，引数となるブロックの変換を行う．

状態番号 = 同じラベルの ARGMENT ブロックの前のブロックの次状態番号　(15)

次状態番号 = 状態番号 + 1　(16)

引数のブロックの変換が終わると，ARGMENT ブロックの次のブロックの状態番号を式 (17)，次状態番号を式 (18) で求める．

状態番号 = 同じラベルの引数となるブロックの最後のブロックの次状態番号　(17)

次状態番号 = 状態番号 + 1　(18)

関数や引数の呼び出し回数はシステムがカウントしており，図 19 のように，関数が 2 回呼び出された場合には，関数となるブロックを 2 回 FST 形式の対話シナリオに変換し，出力する．

なお，FST 形式の対話シナリオの文の順番は音声対話シナリオの動作への影響はなく，提案システムでは関数となる対話シナリオをまとめて出力する．

5.4　FST ファイル出力機能

FST ファイル出力機能では，対話シナリオ変換機能により出力された FST 形式の対話シナリオを FST ファイルに出力する．

5.5　ブロック保存・取得機能

ブロックの保存機能では，次節で説明するワークスペースにあるブロックの情報を xml 形式でファイルに出力する．ブロックの取得機能では，xml 形式で書かれたブロックの情報を読み込み，ブロックを復元する．

5.6　プロトタイプシステム

これまでに述べた提案に基づき作成したプロトタイプシステムを**図 29** に示す．

対話シナリオの作成には，以下の 3 つのエリアを使用する．

ツールボックス　ブロックの一覧を表示するエリアである．基本画面では，ブロックのカテゴリ名だけが表示

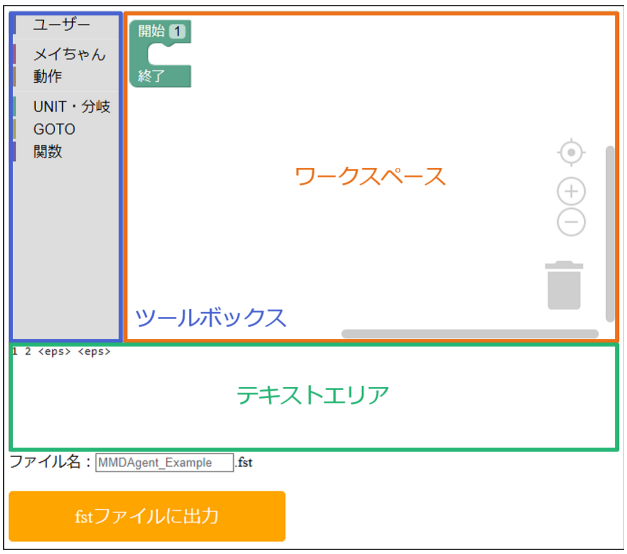

図 29　プロトタイプシステム
Fig. 29　The prototype system.

され，カテゴリ名をクリックすると，そのカテゴリに属したブロックの一覧が表示される．

ワークスペース　ユーザがブロックを組み合わせて，対話シナリオを作成するエリアである．

テキストエリア　対話シナリオ変換機能により，ワークスペースにあるブロックから変換された FST 形式の対話シナリオを表示するエリアである．テキストエリアは非表示にすることもできる．

また，FST ファイル出力機能やブロック保存・取得機能は，テキストエリアの下部に実装した．

なお，プロトタイプシステムは Web ブラウザ上で利用可能で，Google Chrome 79.0.3945.88 と Microsoft Edge 44.17763.831.0 で動作を確認した．また，プロトタイプシステムを用いて作成した対話シナリオは，MMDAgent version1.8 で動作を確認した．

6.　評価実験

5.6 節で説明したプロトタイプシステムを使用し，以下の 3 つの評価実験を行った．

評価実験 1　対話シナリオの編集に関する評価実験
要求仕様 2，3 に対する評価を行う．

評価実験 2　対話シナリオの把握に関する評価実験
要求仕様 3 に対する評価を行う．

評価実験 3　ユーザビリティに関する評価実験
要求仕様 1 に対する評価を行う．

6.1　対話シナリオの編集に関する評価

6.1.1　実験方法

指定した内容を再現する対話シナリオを作成し，その正解者数と所要時間，実験後のアンケートから評価を行う．ここで，正解者数とは，指定した内容どおりに MMDAgent が動作する対話シナリオを作成した人数のことである．この評価実験の目的と被験者，実験の手順，アンケート項目を以下に示す．

(1) 実験の目的
以下の 2 つの方法の比較から，対話シナリオの編集における提案システムの有効性を評価すること．
従来方法　テキストエディタにより，対話シナリオをテキストで編集する．
提案方法　提案システムにより，対話シナリオを編集する．

(2) 被験者
従来方法の経験が若干あり，提案方法の経験はない大学生 6 人

(3) 実験の手順
被験者を 2 つのグループに分け，第 1 のグループは従来方法，提案方法の順で，第 2 グループは提案方法，従来方法の順で以下の 3 つ課題の対話シナリオを作成する．課題の解答時間には制限を設けていない．全課題解答後にアンケートを行う．

- 課題
  (a) 順次構造の対話シナリオの作成
  (b) 分岐構造を持つ対話シナリオの作成
  (c) 変数を用いてランダムに遷移する対話シナリオの作成

(4) アンケート項目
(a) 対話シナリオの作成が簡単である．
(b) 対話シナリオが見やすい．
(c) 対話シナリオを編集しやすい．
(d) 使用した方法は取っ付きやすい．
(e) 対話シナリオの作成に時間がかからない．

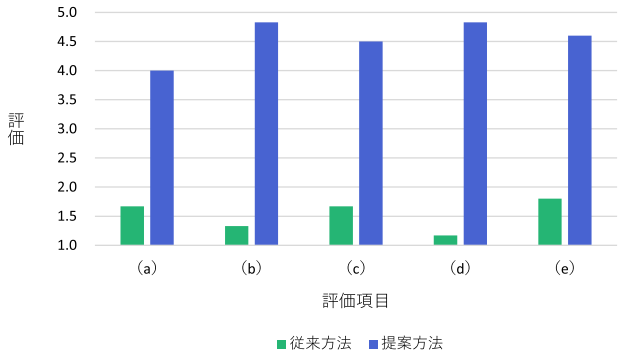

図 30　対話シナリオの編集に関するアンケート結果
Fig. 30　Questionnaire results on editing interaction scenarios.

6.1.2　結果と考察

対話シナリオの編集に関する評価結果を**表 1** に，5 段階評価アンケートの平均結果を**図 30** に示す．

6.1.2.1　要求仕様 2 に対する評価

従来方法に比べて，提案方法の方が，すべての課題で正解者数が多くなった．今回，従来方法での間違いで一番多

表 1　対話シナリオの編集に関する評価結果
Table 1　Evaluation results for editing interaction scenarios.

	従来方法			提案方法		
課題	正解者数	平均所要時間	正解者の平均所要時間	正解者数	平均所要時間	正解者の平均所要時間
(a)	3 人	8 分 50 秒	10 分 49 秒	6 人	3 分 27 秒	3 分 27 秒
(b)	2 人	4 分 39 秒	5 分 02 秒	3 人	4 分 26 秒	4 分 38 秒
(c)	2 人	10 分 03 秒	9 分 54 秒	2 人	6 分 01 秒	3 分 50 秒
全課題		7 分 51 秒	8 分 35 秒		4 分 38 秒	3 分 58 秒

かったのが，状態番号の計算ミスや記述ミスであった．また，遷移条件コマンドと出力コマンドの順番が 1 文だけ逆になっている解答もあった．一方，提案方法では状態番号の計算は自動で行われ，ユーザのキーボード入力も最小限に抑えているため，従来方法で起きるような簡単なミスが減り，正解率が上がったと考える．

また，正解した人の全課題の平均所要時間は，従来方法では 8 分 35 秒，提案方法では 3 分 58 秒と 46.2%削減となった．提案方法では，対話シナリオ作成時の状態番号の計算ミスやコマンドの入力ミスの恐れがなくなり，慎重に対話シナリオを作成する必要がなくなったため，所要時間が短くなったと考える．

以上より，提案システムを用いることで，対話シナリオの記述のミスを減らすことができ，ユーザが意図する対話シナリオをより短い時間で作成することができるようになったといえる．

6.1.2.2　要求仕様 3 に対する評価

図 30 より，どの項目においても，提案方法の方が良い評価を得ていることが分かる．特に，従来方法に比べて，提案方法では，対話シナリオが見やすく，取っ付きやすくなるとユーザが感じることが分かった．対話シナリオの編集においても，提案システムを用いることで，対話シナリオの把握が容易になり，対話シナリオの作成が手軽になったといえる．

6.1.2.3　今後の課題

課題 (b) で提案方法でも正解者数が少ない理由は次のように考えられる．課題 (b) は分岐構造を持つ対話シナリオの作成であったが，評価実験を行ったプロトタイプシステムでは，分岐ブロックは**図 31** のように，分岐していない形でツールボックスに表示されており，ユーザが分岐ブロックの Mutators 機能に気づかず，分岐ブロックを縦に 2 つ並べて使用していたため正しい対話シナリオが作成されなかった．そこで，この結果を受けて，現在のシステムでは分岐ブロックは**図 32** のように，初めから二股で表示し，分岐構造について分かりやすいように改善した．前述の被験者とは異なる大学生 6 人が現在のシステムで課題 (b) の対話シナリオを作成した結果を**表 2** に示す．現在のシステムを用いることで，正解者数は増加し，平均所要時間は短縮した．不正解であった 2 人も，分岐構造については正しい対話シナリオを作成しており，現在のシステムの方が分岐構造について分かりやすいシステムであるといえる．

図 31　分岐ブロック（分岐なし）
Fig. 31　The FUNCTION Block (No branch).

図 32　分岐ブロック（分岐 2 つ）
Fig. 32　The FUNCTION_CALL Block (Two branches).

表 2　課題 (b) の再評価結果
Table 2　Evaluation results for task (b).

ツールボックスのブロック	図 31（分岐なし）	図 32（分岐 2 つ）
正解者数	3 人	4 人
平均所要時間	4 分 26 秒	3 分 46 秒
正解者の平均所要時間	4 分 36 秒	3 分 39 秒

また，課題 (c) で提案方法でも正解者数が少ない理由は次のように考えられる．課題 (c) は変数を用いてランダムに遷移する対話シナリオの作成であったが，ランダムに遷移させる方法が分からず，課題を途中で諦めるユーザも多かった．そこで，この結果を受けて，ランダムに遷移するなど難しい状態遷移の構造に関しては，1 つのブロックで複雑な対話シナリオを再現できるブロックをあらかじめ用意することにした．課題 (b) の再評価実験と同じ被験者が，ランダムに遷移する状態遷移の構造について簡単に再現できるブロックを新たに用意したシステムで，課題 (c) の対話シナリオを作成した結果を**表 3** に示す．再評価実験の結果，正解者数は 3 人増えた．正解者の平均所要時間は 3 分 50 秒から 5 分 48 秒と 51%増加したが，ブロックを簡単にしたことで，被験者が途中で解答を諦めることなく，時間をかけて正しい対話シナリオを作成したためであると考える．

新たに実装したシステムでも，すべての課題において被験者全員が正解することはなかった．今後，被験者の解答から，さらなる改善や機能の追加を検討し，ユーザが意図

表 3　課題 (c) の再評価結果
Table 3　Evaluation results for task (c).

ランダムに遷移する状態遷移の構造	複数のブロックを組み合わせて記述	新たに用意した 1 つのブロックで記述
正解者数	2 人	5 人
平均所要時間	6 分 01 秒	5 分 30 秒
正解者の平均所要時間	3 分 50 秒	5 分 48 秒

したとおりの対話シナリオの作成ができるシステムの開発を目指す．

6.2　対話シナリオの把握に関する評価

6.2.1　実験方法

対話シナリオから対話の内容を推測し，その推測結果と実験後のアンケートから評価を行う．この評価実験の目的と被験者，実験の手順，アンケート項目を以下に示す．

(1) 実験の目的

以下の 2 つの方法の比較から，対話シナリオの把握における提案システムの有効性を評価すること．

従来方法　テキストで記述した対話シナリオから対話の内容を推測する．

提案方法　提案システムを用いて記述した対話シナリオから対話の内容を推測する．

(2) 被験者

6.1 節と同様．

(3) 実験の手順

6.1 節と同様．ただし，課題は以下の 2 つである．

- 課題
  - (a) 順次構造の対話シナリオの内容の推測
  - (b) 分岐構造を持つ対話シナリオの内容の推測

(4) アンケート項目

(a) 対話内容の推測が容易にできる．
(b) 対話シナリオが見やすい．
(c) 対話内容が分かりやすい．
(d) 使用した方法は取っ付きやすい．
(e) 対話内容の推測に時間がかからない．

6.2.2　結果と考察

5 段階評価アンケートの平均結果を図 **33** に示す．

6.2.2.1　要求仕様 3 に対する評価

図 33 より，どの項目においても，提案方法の方が良い評価を得ていることが分かる．提案システムを用いることで，対話内容が分かりやすく，対話内容の推測に時間がかからないとユーザが感じることが分かった．

また，被験者の解答を見ると，従来方法よりも提案方法の方が対話内容を具体的に推測しているものが多かった．このことから，提案システムを用いることで対話シナリオの細かい点まで注目でき，詳しく対話内容を把握できるといえる．

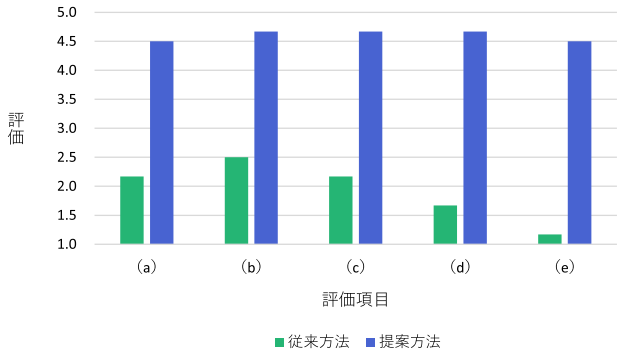

図 33　対話シナリオの把握に関するアンケート結果
Fig. 33　Questionnaire results on grasping interaction scenarios.

6.3　ユーザビリティに関する評価

6.3.1　実験方法

提案システムを使用して対話シナリオを作成し，実験後のアンケートから評価を行う．ユーザビリティの評価尺度としては SUS（System Usability Scale）[21] を使用する．SUS とは，1986 年にイギリスの John Brook がビデオ端末の開発にあたってユーザの満足度を測定するために作成した心理尺度で，システムの満足度評価における標準となっている．SUS のスコアは 100 点満点で算出され，平均スコアは 68 である．この評価実験の目的と被験者，実験の手順，アンケート項目を以下に示す．

(1) 実験の目的

一般ユーザにより，提案システムのユーザビリティを評価すること．

(2) 被験者

大学の公開講座に参加した 12 歳から 72 歳までの一般ユーザ 41 人

(3) 実験の手順

大学の公開講座にて実験を行ったため，被験者には提案システムだけでなく，MMDAgent の利用も体験してもらった．また，被験者は実験中でも自由に質問ができる．実験の手順を以下に示す．

(a) MMDAgent の使用方法の説明
(b) MMDAgent の利用体験
(c) 提案システムの使用方法の説明
(d) 提案システムを用いて対話シナリオを作成する，5 つの演習課題に取り組む．
(e) 作成した対話シナリオを用いて，MMDAgent を

表 4　提案システムの SUS スコア
Table 4　SUS score of proposal system.

被験者	SUS スコア
10 代	70.4
20 代	-
30 代	-
40 代	55.0
50 代	70.0
60 代	17.5
70 代	50.0
全年代	67.8

動かす．

(f) 演習課題が終わった人は，自由に提案システムと MMDAgent を使用する．

(g) アンケートを実施

(4) アンケート項目

(a) 提案システムをまた利用したいと思う．

(b) 提案システムは複雑だと感じた．

(c) 提案システムは使いやすいと思った．

(d) 提案システムを利用するには専門家のサポートが必要だと思った．

(e) 提案システムには様々な機能がうまくまとまっていると感じた．

(f) 提案システムにはちぐはぐな点が多いと思った．

(g) 提案システムの利用方法をたいていの人はすぐに理解すると思う．

(h) 提案システムはとても操作しにくいと感じた．

(i) 提案システムを使いこなせる自信がある．

(j) 提案システムを使用する前に知っておくべきことが多くあると思う．

6.3.2　結果と考察

提案システムの SUS のスコアを**表 4** に示す．なお，今回の評価実験の被験者の中に 20 代と 30 代の人はいなかった．

6.3.2.1　要求仕様 1 に対する評価

提案システムの SUS のスコアは 67.8 であった．このことから，提案システムのユーザビリティは標準的で，一般ユーザでも利用できるシステムであることが分かった．

被験者の年代別の SUS スコアを見てみると，10 代の SUS スコアが一番高く，70.4 である．提案システムを使用している様子を観察していても，頭で考えてから対話シナリオを作成するのではなく，様々なブロックを使って対話シナリオを作成し，MMDAgent で動作を試す人が多かった．また，分からないことがあったときやエラーが出たときに積極的に質問する人が多く，これらのことから評価実験の終盤では提案システムを使いこなしている人が多く，SUS スコアも高くなったのだと考える．一方で，60 代の被験者については，評価実験の参加者が 1 人しかいなかったため，偏った結果となった．他の年代も 10 代に比べて SUS スコアが低く，アンケート項目 (d) で特に悪い評価を得ていた．そのため，人のサポートがなくても提案システムを使用している間に利用方法を習得できるような工夫が必要であると考える．

以上より，SUS スコアは年代によってばらつきがあり，提案システムのユーザビリティは改善が必要であるといえる．しかし，評価実験では，FST 形式について説明していないが，すべての被験者が簡単な構造の対話シナリオを作成することができた．これらのことから，提案システムを用いることで，FST 形式の構文について知識がない一般ユーザでも対話シナリオを作成することができるといえる．

6.3.2.2　今後の課題

今回の評価実験は，大学の公開講座の中で行い，システムの操作をユーザに体感してもらうことを目的としていたため，提案システムと MMDAgent の連携もユーザ自身が行った．そのため，提案システムで作成した対話シナリオを MMDAgent で動作させる方法が一般ユーザにとっては複雑で，SUS のスコアがあまり高くならなかったのだと考える．このことから，今後の課題として，提案システムと MMDAgent のシームレスな連携が必要であるといえる．提案システムのなかで，ユーザがボタンを押すだけで MMDAgent が起動し，作成した対話シナリオを実行する仕組みを実現したいと思う．

7.　おわりに

本論文では，一般ユーザによる対話シナリオの編集を可能にするため，ブロック型ビジュアルプログラミング機能を有する音声対話シナリオ編集システムを提案し，その実現法について述べた．提案システムでは，Google Blockly を用いて，対話の内容を「ブロック」で表し，パズルのように組み合わせることで対話シナリオを作成できるようにした．対話シナリオの作成では，音声対話システムのエージェントとそのユーザの「対話」に沿って記述することで，プログラミングや FST 形式に馴染みのないユーザでも，直観的に操作できるようにした．

また，提案システムに基づきプロトタイプシステムを作成し，提案システムの有効性とユーザビリティに関する評価を行った．提案システムの有効性に関する評価では，対話シナリオの編集と把握について，被験者（大学生）6 人に対して，テキストエディタで対話シナリオを記述する従来方法と提案システムを用いて記述する提案方法を比較した．この結果，提案システムを用いることで，短い時間で対話シナリオの編集や内容の把握ができ，ユーザが取っ付きやすいと感じることが分かった．また，ユーザビリティに関する評価実験では，被験者（大学の公開講座に参加した 12 歳から 72 歳までの一般ユーザ）41 人に，提案システムを使用してもらい，評価尺度として SUS を用いて，ユーザビリティの評価を行った．提案システムの SUS のスコ

アは 67.8 であり，一般ユーザでも利用できるシステムであることが分かった．

今後は，一般ユーザによる評価を続けて行い，一般ユーザによる対話シナリオの編集が可能となったことで，対話シナリオのバリエーションが増加したか検証する．また，提案システムを名古屋工業大学の正門前にある音声対話システムを搭載したデジタルサイネージに実装し，デジタルサイネージの利用者が自由に対話シナリオを編集できるシステムの開発を目指す．利用者が「面白い」,「欲しい」と思う対話シナリオを登録することで，その他の多くの利用者にとって便利なデジタルサイネージになると考えられる．

謝辞　本研究は，JSPS 科研費 19H04115，および，総務省 SCOPE の助成を受けたものです．この場を借りて，感謝の意を表します．

参考文献

[1] Lee, A., Oura, K. and Tokuda, K.: MMDAgent — A Fully Open-Source Toolkit for Voice Interaction Systems, *Proc. IEEE ICASSP2013*, pp.8382–8385 (2013).
[2] 馬場哲晃，島影圭佑，本多達也，田中浩也：あなたのためのデザイン：デジタルファブリケーションが可能にする身近な人の為の福祉機器プロダクト，研究報告アクセシビリティ（AAC），Vol.2017-AAC-4, No.12, pp.2432–2431 (2017).
[3] 古市瑞希，山本大介，高橋直久：プログラミング初心者のための Blockly を用いた音声対話シナリオ編集システム，マルチメディア，分散，協調とモバイル（DICOMO2019）シンポジウム，pp.423–432 (2019).
[4] Scratch, available from ⟨https://scratch.mit.edu/⟩ (accessed 2020-1-15).
[5] viscuit, available from ⟨https://www.viscuit.com/⟩ (accessed 2020-1-15).
[6] Google Blockly, available from ⟨https://developers.google.com/blockly/⟩ (accessed 2020-1-15).
[7] Nishimura, R., Yamamoto, D., Uchiya, T. and Takumi, I.: Development of a Dialogue Scenario Editor on a Web Browser for a Spoken Dialogue System, *Proc. 2nd International Conference on Human Agent Interaction*, pp.129–132, ACM Digital Library (2014).
[8] Nishimura, R., Yamamoto, D., Uchiya, T. and Takumi, I.: MMDAE: Dialog scenario editor for MMDAgent on the web browser, *ICT Express*, Vol.5, No.1, pp.47–51 (2019).
[9] Wakabayashi, K., Yamamoto, D. and Takahashi, N.: A Voice Dialog Editor Based on Finite State Transducer Using Composite State for Tablet Devices, *Proc. IEEE/ACS ICIS 2015*, Las vegas (2015).
[10] Wakabayashi, K., Yamamoto, D. and Takahashi, N.: A Voice Dialog Editor Based on Finite State Transducer Using Composite State for Tablet Devices, *Computer and Information Science 2015*, Studies in Computational Intelligence, Vol.614, pp.125–139 (2016).
[11] pepper, available from ⟨https://www.softbank.jp/robot/pepper/⟩ (accessed 2020-1-15).
[12] Choregraphe, available from ⟨https://www.softbankrobotics.com/jp/developer/document/#pepper⟩ (accessed 2020-1-15).
[13] コミュクラフト，入手先 ⟨https://commucraft.amivoice.com/⟩（参照 2020-1-15）.
[14] 光永法明，井芹威晴，吉田図夢：タブレット端末で動作する，マイコン用ビジュアルプログラミング環境 aiBlocks の開発，情報処理学会論文誌教育とコンピュータ（TCE），Vol.3, No.1, pp.53–63 (2017).
[15] 杉村　博，笹川雄司，藤田裕之，関家一雄，黄　啓新，一色正男：HEMS コントローラ開発者育成のための ECHONET Lite 教育コンテンツの開発，研究報告コンシューマ・デバイス&システム（CDS），Vol.2016-CDS-15, No.17, pp.1–4 (2016).
[16] ECHONET, available from ⟨https://echonet.jp/⟩ (accessed 2020-1-15).
[17] Repl-AI, available from ⟨https://repl-ai.jp/⟩ (accessed 2020-1-15).
[18] 大村　廉：物語記述によるホームネットワークアプリケーション開発手法の提案，情報処理学会論文誌 Vol.58, No.10, pp.1591–1605 (2017).
[19] Blockly Developer Tools, available from ⟨https://blockly-demo.appspot.com/static/demos/blockfactory/index.html⟩ (accessed 2020-1-15).
[20] Mutators, available from ⟨https://developers.google.com/blockly/guides/create-custom-blocks/web/mutators⟩ (accessed 2020-1-15).
[21] Brooke, J.: *SUS: A "quick and dirty" usability scale*, Jordan, P.W., Thomas, B., Weerdmeester, B.A. and McClelland, A.L. (Eds.), Usability Evaluation in Industry, London: Taylor and Francis (1996).

古市 瑞希　（学生会員）

1996 年生．2019 年名古屋工業大学工学部情報工学科卒業．同年名古屋工業大学大学院博士前期課程に進学し，情報工学専攻に在学．現在，音声対話システムの研究に従事．

山本 大介　（正会員）

2003 年名古屋大学工学部電気電子・情報工学卒業．2008 年同大学大学院情報科学研究科博士課程修了．2008 年名古屋工業大学工学研究科助教．2011 年同准教授．地理情報システム，マルチメディア，Web サービスに関する研究に従事．日本データベース学会，人工知能学会，日本バーチャルリアリティ学会，IEEE 各会員．博士（情報科学）．

高橋 直久　（正会員）

1974 年電気通信大学応用電子工学科卒業．1976 年電気通信大学大学院応用電子工学専攻修了．1976 年日本電信電話公社（現 NTT）武蔵野電気通信研究所入所．1987 年工学博士（東京工業大学）．NTT 基礎研究所，ソフトウェア研究所，未来ねっと研究所を経て，2001 年名古屋工業大学工学部電気情報工学科教授．2004 年同大学大学院工学研究科ながれ領域（情報工学専攻/情報工学科）教授．2017 年同大学名誉教授，国際音声言語技術研究所プロジェクト教授．現在，時空間情報処理，音声対話システム，ネットワーク診断システム，e ラーニング・システム等の研究に従事．電子情報通信学会，日本データベース学会，ACM 各会員．本会終身会員．