複合状態編集機能を有する 音声対話コンテンツ編集システム

林 晃大[1] 山本 大介[1] 高橋 直久[1]

概要：本研究室では Android 版 MMDAgent を用いて携帯端末向け音声対話システムの研究を行っている．MMDAgent は，ユーザ自身が任意に対話シナリオをカスタマイズすることができるという特徴をもつ．FST(Finite State Transducer) ファイルと呼ばれる外部ファイルに対話シナリオを記述し，起動時に FST ファイルを読み込ませるという方法で対話シナリオのカスタマイズを行う．また，本研究室では FST ファイルの編集が不慣れなユーザでも編集できるように，MMDAgent の対話シナリオをタブレットを用いてタッチ操作で追加・編集できるシステム EFDE を提案してきた．しかし，EFDE は，FST テンプレートを用いることで定型的な状態遷移を一つの複合状態と扱うことが可能になり，みかけの状態数を削減することができる利点がある．その一方で，非定型的な処理や，割り込みなどを伴う状態遷移を記述しようとすると，状態数が極端に増えてしまう課題があった．これらの課題を解決するために，UML における exit point の概念を実装した exit point 付き複合状態モデルを提案する．提案システムは複合状態の内部に exit point を設定可能，内部を状態遷移図に基づいて編集可能にすることで従来システムより自由度の高い編集が可能という特徴を持つ．さらに，提案システムに基づいて実装したプロトタイプシステムを評価実験を行い，提案システムの有用性を評価した．

Voice Interaction Contents Editor using Composite State Editing Function

KODAI HAYASHI[1] DAISUKE YAMAMOTO[1] NAOHISA TAKAHASHI[1]

1. はじめに

近年，Apple の Siri[1] や NTT ドコモのしゃべってコンシェル [2] などの携帯端末向け音声対話システムが普及しつつある．

そこで，我々の研究室ではパソコン向け音声インタラクション構築ツールキットの MMDAgent[3]，[4] を Android[5] 向けに移植した Android 版 MMDAgent[6] を開発し，Android 版 MMDAgent を用いて携帯端末向け音声対話システムの研究を行っている．

MMDAgent は，対話シナリオを編集することで，ユーザ自身が対話内容を自由に設定できるという特徴をもつ．FST ファイルと呼ばれる外部ファイルに記述し，起動時に MMDAgent に FST ファイルを読み込ませることで記述した対話を行うことができる．対話シナリオとは，「ユーザがこう言ったら，キャラクタはこう返してこういう動作をする」といった対話の流れや台本のようなものを意味する．また，FST をテキストで記述する際の記述言語を FST スクリプトと呼び，FST スクリプトが記述されたファイルを FST ファイルと呼ぶ．

FST ファイルの状態遷移を理解するためには状態番号を目でたどって読む必要がある．FST ファイルでは，同じ状態番号の遷移でも行を大きくまたいで遷移を記述することが可能で，そのような記述を避けていても複雑な状態遷移になればそうせざるを得ない場合が出てくる．そのような場合，状態遷移を理解しづらくなる．そこで，我々の研究室では状態遷移を直感的にわかりやすいように，状態遷移図で MMDAgent の対話シナリオをタブレットを用いてタッチ操作で編集，作成できるシステム EFDE[7] を提案

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

した．EFDE の詳細は次節で述べる．

しかし，EFDE は状態数, 遷移数が増加するとタブレットのタッチ操作での編集が難しくなる．点検作業での音声対話といったようなユーザ割り込みを含むような複雑な対話シナリオを編集すると作成する状態遷移図が煩雑になり対話シナリオの編集が難しくなるという課題がある．また，テキストエディタで編集する場合でも，先程述べたように複雑な状態遷移になるため，状態遷移が理解しにくいという課題がある．これらの課題に対して，本研究では，複合状態を利用して状態をグループ化し階層化することで状態遷移図を見やすく整理できるのではと考え，複合状態内部を状態遷移図に基づいて編集可能な exit point[8] 付き複合状態モデルを提案する．

2. 研究背景

2.1 Android 版 MMDAgent

Android 版 MMDAgent とは，音声インタラクション構築ツールキットの MMDAgent を Android 版端末向けに移植したものである．MMDAgent は，名古屋工業大学国際音声技術研究所で開発された音声対話システム構築ツールキットであり，音声認識，音声合成，3D モデルの描画と物理演算などの機能を統合したシステムである．音声認識では Julius[9] を，音声合成では Open JTalk[10] を，3D モデルでは MikuMikuDance[11] 形式を，物理演算では Bullet Physics[12] を採用している．

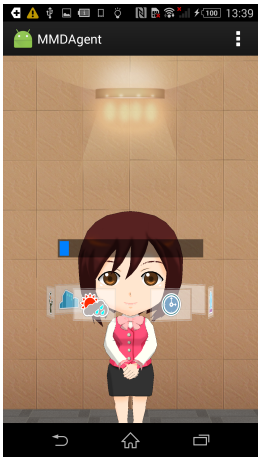

図 1 Android 版 MMDAgent

MMDAgent は対話シナリオの記述形式に有限状態オートマトンの一種である有限状態トランスデューサ (FST) を採用している．FST をテキストで記述する際の記述言語を FST スクリプトと呼び，FST スクリプトが記述されたファイルを FST ファイルと呼ぶ．この FST ファイルを編集することによって対話シナリオのカスタマイズが可能となっている．

2.2 FST ファイル

FST ファイルの例を図 2 に示す．FST ファイルは図 2 の左から順に状態番号，次状態番号，遷移条件，遷移時に出力されるコマンドの 4 つの組から構成される．これらの要素は 1 行で記述され，各要素間は空白またはタブ文字で区切られる．図 2 では状態番号が 0 の際に，「こんにちは」という音声を認識した場合，状態番号 10 へと遷移し，3D キャラクタが greet.vmd で指定された動作 (今回はお辞儀) を行う．さらに，状態番号 11 へと遷移し，「こんにちは」と音声合成を行い，状態番号 12 へと遷移する．

```
0  10  RECOG_EVENT_STOP|こんにちは   <eps>
10 11  <eps>                        MOTION_ADD|mei|greet|greet.vmd
11 12  <eps>                        SYNTH_START|mei|normal|こんにちは
12 0   SYNTH_EVENT_STOP|mei         <eps>
```

図 2 FST スクリプトの例

2.3 EFDE

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って対話スクリプトを作成できるシステムであり，作成する状態遷移図のモデルとして，複合状態を導入した状態遷移モデルを採用している．複合状態とは内部遷移を持つ状態のことである．また，あらかじめ FST テンプレートの定義に外部ファイルを用いて入力フォームの構成，基礎となる状態遷移，入力データから状態遷移図への変換方法を定義することで，作成したい処理に近い FST テンプレートを選択し，データを入力することで手軽に編集が可能である．

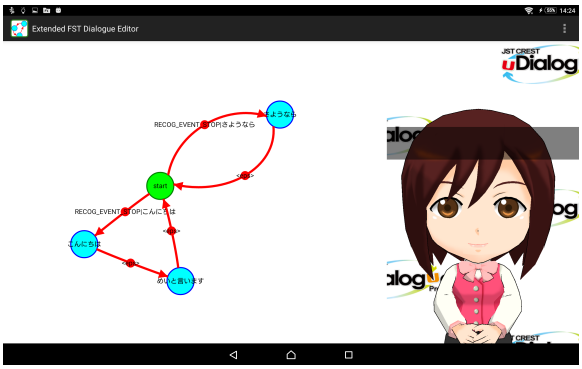

図 3 EFDE

2.4 EFDE の問題点

EFDE は状態数, 遷移数が増加するとタブレットのタッチ操作での編集が難しくなる．複合状態の編集方法があらかじめ定義された FST テンプレートを用いた方法のみであることから複合状態の内部状態に複合状態を設定できない．また，複合状態内部を状態遷移図に基づいて編集できない．そのため，図 4 のような点検作業での音声対話と

いったようなユーザ割り込みを含むような対話シナリオを編集すると，状態遷移が複雑になる．ユーザ割り込みとは，一連の動作 (この場合は点検作業) を強制的に終了させる遷移である．

しかし，複合状態の編集方法があらかじめ定義された FST テンプレートを用いた方法のみであることから，複合状態を利用して状態のグループ化ができないため，青枠で囲まれた状態を点検というグループでグループ化することができない．ユーザ割り込みを表現するためには，青枠で囲まれた各状態ごとに遷移を接続しなければならない．そのため，作成する状態遷移図が煩雑になり対話シナリオの編集が難しくなるという課題がある．

図 4 は，丸が状態，矢印が遷移である．また，状態に書かれている文章が 3D キャラクタが読み上げる内容で，遷移に書かれている文章は遷移条件である．

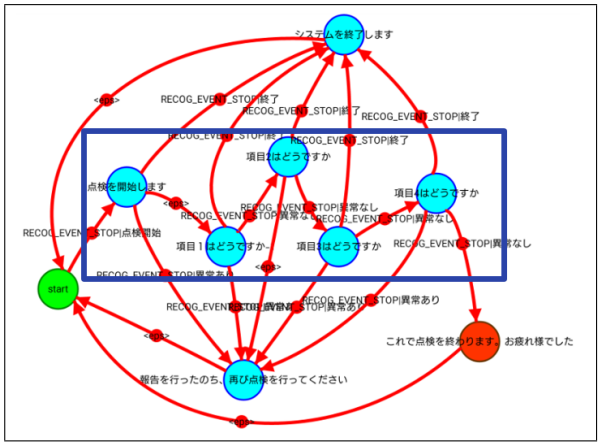

図 4　EFDE での点検作業の状態遷移図

そこで，複合状態内部を状態遷移図に基づいて編集可能な exit point 付き複合状態があると，状態をグループ化することで，一度に表示する状態，遷移数が減り，状態遷移図が見やすくなる．また，処理の流れごとに状態をグループ化できるため，状態遷移図を見て大まかな動作の流れが確認可能である．

しかし，EFDE では exit point 付き複合状態モデルの実装に当たり，以下の 3 つの問題がある．

(1) 複合状態内部を状態遷移図に基づいて編集することができない
(2) exit point 付き複合状態を複合状態を持たない状態遷移に展開できない
(3) exit point 付き複合状態を編集するために編集画面切替機能を持つ編集インタフェ-スを持たない

3.　提案システムの概要

提案システムでは図 5 ように，図 4 の青枠で囲んだ状態を exit point 付き複合状態 (図 5 の四角で描画された状態) の内部を状態遷移図に基づいて編集可能にし，グループ化可能にした．また，exit point 付き複合状態の処理を終わらせるユーザ割り込みの遷移 (図 5 の exit point 付き複合状態からの赤い矢印．以降，強制遷移と呼ぶ) を 1 本の遷移で表現可能にした．提案システムでは，これらにより状態遷移図を見やすくしようと考えた．

図 5 の exit point 付き複合状態からの始点が黒丸の赤い矢印は複合状態の処理終了時の遷移あり，二重丸に描画されている状態は exit point に対応する状態である．

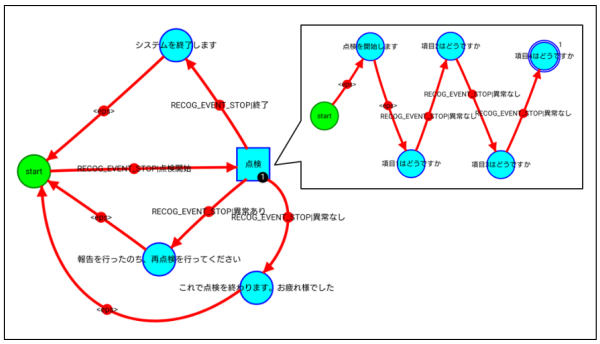

図 5　提案システムでの点検作業の状態遷移図

3.1　提案システムの機能

2.4 節で述べた 3 つの問題点を解決するために，提案システムで従来システムに加えた 2 つの新たな機能と変更を加えた 1 つの機能について説明する．

exit point 付き複合状態登録機能

2.4 節の (1) を解決するために，exit point 付き複合状態登録機能では，状態遷移図に exit point 付き複合状態を追加した際に，追加した exit point 付き複合状態の状態遷移図データを新たに作成することで，複合状態内部を状態遷移図に基づいて編集できるようにした．状態遷移図データとは，状態遷移図を記憶するデータである．また，追加した状態遷移図データを木構造に登録する機能である．exit point 付き複合状態を削除する際には，削除する exit point 付き複合状態の状態遷移図データを木構造から削除する．

exit point 付き複合状態の展開方法

exit point 付き複合状態モデルを導入したことにより，複合状態の処理を強制的に終了させる遷移である強制遷移，exit point 付き複合状態の処理終了時の遷移である exit point からの遷移が追加され，exit point 付き複合状態の展開方法が FST 生成機能に追加された．

2.4 節の (2) を解決するために，FST 生成機能では，複合状態，exit point 付き複合状態の判定，遷移の種類の判定を行った後，各複合状態の展開が行われている．

編集画面切替機能

2.4 節の (3) を解決するために，編集画面切替機能は，ユーザによって編集画面切り替えの要求があった際に，ユーザが選択した exit point 付き複合状態の状態遷移図データを選択し，編集画面に読み込み，切り替えを実行する機能である．編集画面の切り替えが実行可能な状態遷移図は，現在の状態遷移図データの親または子の関係にある状態遷移図データのみである．

3.2 提案システムの構成

提案システムのシステム構成図を図 6 に示す．提案システムでは，従来システムである EFDE に加えて exit point 付き複合状態登録機能，編集画面切替機能，FST 生成機能に exit point 付き複合状態の展開方法を追加した．まず状態遷移図作成機能で exit point 付き複合状態を追加すると，exit point 付き複合状態登録機能によって追加した exit point 複合状態の状態遷移図データが新たに作成され，状態遷移図データを管理する木構造に登録される．状態遷移図作成機能で編集画面の切り替えをすると，編集画面切替機能によって木構造から要求された状態遷移図データを選択し，編集画面に読み込み，切り替えが実行される．

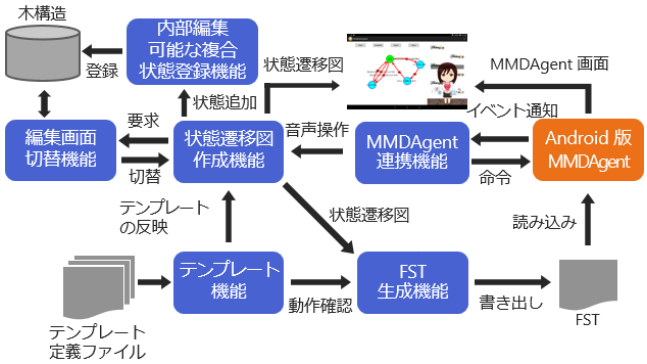

図 6　提案システムの構成図

4. 実現方法

ここでは，提案システムの実現法について述べる．4.1 節では exit point 付き複合状態モデルについて述べる．また 4.2 節では exit point 付き複合状態の展開方法について述べ，4.3 節では編集画面切替機能の実装について述べる．

4.1 exit point 付き複合状態

exit point 付き複合状態モデルを導入するためには，従来システムで実装されている複合状態と区別し，exit point 付き複合状態内部を状態遷移図に基づいて編集可能にするという課題を解決する必要がある．この節では従来システムで実装されている複合状態について説明した後，課題の解決策について述べる．また，exit point と exit point 付き複合状態の処理終了時の遷移を表現した exit point からの遷移，複合状態の処理を強制的に終了させる遷移を表現した強制遷移について説明する．

4.1.1 複合状態

複合状態は UML のステートマシン図やハレルが提唱したステートチャート [13] に採用されている．複合状態とは内部に状態を持つ状態のことであり，複合状態を導入することによって状態を階層的に表現することができる．また一度に画面に表示すべき状態数を減らすことが可能である．複合状態編集方法は，テンプレート定義ファイルと呼ばれる入力フォームと変数が存在するテンプレートが定義されているファイルを選択し，入力フォームからデータを入力することで複合状態の編集が可能である．

4.1.2 exit point 付き複合状態

exit point 付き複合状態とは，複合状態内部を状態遷移図に基づいて編集可能な複合状態である．提案システムでは exit point 付き複合状態ごとに状態遷移図データを作成し，4.3 節で説明する編集画面切替機能を利用して各 exit point 付き複合状態を編集する．また，複合状態と exit point 付き複合状態を区別するために，4.1.5 節で説明した複合状態を扱うクラス Module State に追加した int 型変数 com を利用し，com = 0 なら複合状態，com != 0 なら exit point 付き複合状態と区別する．

4.1.3 exit point

exit point は UML のステートマシン図で採用されている．exit point とは複合状態の処理が終了したことを示す仮想状態であり，複合状態の出口を表している．

4.1.4 exit point により追加される遷移

提案システムでは exit point 付き複合状態モデルの導入により，exit point からの遷移，強制遷移を表現することができる．exit point からの遷移は，複合状態内部の exit point に対応する状態からの遷移であり，複合状態の処理終了時の遷移を表現している．強制遷移は複合状態の全ての内部状態からの遷移であり，複合状態の処理を強制的に終了させる遷移を表現している．これら 2 つの遷移を実装するために 4.1.5 節で説明した遷移を扱うクラス Transition に追加した int 型変数 t_exit を利用し，exit point に登録されている複合状態からの遷移で t_exit = -1 なら強制遷移，t_exit>0 なら exit point からの遷移と区別した．exit point 付き複合状態からの遷移を追加した段階では，exit point に登録されている複合状態からの遷移は強制遷移となっており，タブレットのタッチ操作によって exit point からの遷移として切り替えることが可能である．

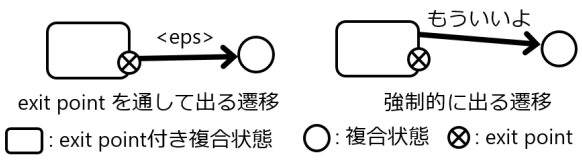

図 7　二種類の遷移

4.1.5 exit point 付き複合状態のデータ構造

exit point 付き複合状態を管理するデータ構造は，複合状態を扱うクラス Module State，状態を扱うクラス State，遷移を扱うクラス Transition の変数は表 1，2，3 のように定義されている．Moudle State クラスの変数 com はその状態が複合状態か exit point 付き複合状態であるかを判別する変数であり，com = 0 なら複合状態，com >0 なら exit point 付き複合状態と区別している．また，Moudle State クラスの変数 exit はその状態が exit point であるかを判別する変数であり，s_exit = 0 なら exit point に対応しない複合状態，s_exit >0 なら exit point に対応する複合状態と区別している．Transition クラスの変数 t_exit はその遷移が exit point からの遷移か強制遷移であるかを判別する変数であり，t_exit >0 なら exit point からの遷移，t_exit = -1 なら強制遷移と区別している．

表 1　複合状態を扱うクラス Module State

変数名	型	説明
child_list	ArrayList<State>	内部の状態のリスト
transition_list	ArrayList<Transition>	内部の遷移のリスト
description	String	複合状態の説明
start	State	内部の開始状態を表す要素
end_states	ArrayList<State>	内部の終了状態のリスト
com	int	状態の種類を判別する要素
s_exit	int	exit point を判別する要素

表 2　状態を扱うクラス State

変数名	型	説明
state_name	String	状態名を表す要素

表 3　遷移を扱うクラス Transition

変数名	型	説明
prestate	State	遷移前状態を表す要素
nextstate	State	遷移先状態を表す要素
event	String	遷移条件を表す要素
command	String	出力を表す要素
t_exit	int	遷移の種類を判別する要素

4.2 exit point 付き複合状態の展開

従来システムでは作成した状態遷移図を FST ファイルに書き出すために，状態遷移図の各状態に重複しないように状態番号を割り当て，複合状態の展開を行いながら FST ファイルへの書き出しをしていた．提案システムでもこの手順に大きな変更はないが，今回 exit point 付き複合状態モデルを導入したことにより追加される遷移を状態遷移図から FST ファイルに書き出すために，新たに複合状態の展開方法を追加した．この節では新たに追加した展開方法の判別，展開方法について説明する．

4.2.1 展開方法の判別

従来システムでは複合状態のみであったため，展開方法は複合状態に接続された遷移の遷移条件によって決定されていた．しかし，提案システムでは複合状態に加え，exit point 付き複合状態を追加するため展開方法が新たに追加される．そこで図 8 のように状態，遷移の判定することで展開方法の判別を行った．複合状態の展開方法の判別の流れを以下に示す．

手順 1　ある複合状態 C に対して複合状態の種別の判定を行う

手順 1-1　複合状態 C から接続している遷移を T とする

手順 1-2　複合状態 C の int 型変数 com の値を取得する

手順 1-3　手順 1-2 で取得した値が 0 なら手順 2 を実行し，0 以外なら手順 3 を実行する

手順 2　複合状態 C からの遷移 T の判定を行う

手順 2-1　複合状態 C から接続している遷移 T の String 型変数 event の値を取得する

手順 2-2　手順 2-1 で取得した値が <eps> なら展開方法として「終了状態のみに接続」，<eps> 以外なら展開方法として「すべての内部状態に接続」を行う．

手順 3　exit point 付き複合状態 C からの遷移 T の判定を行う

手順 3-1　取得した exit point 付き複合状態 C から接続している遷移 T の int 型変数 t_exit の値を取得する

手順 3-2　手順 3-1 で取得した値が −1 なら展開方法として「すべての内部状態に接続」，0 より大きいなら展開方法として「取得した値に対応する exit point である状態のみに接続」を行う．

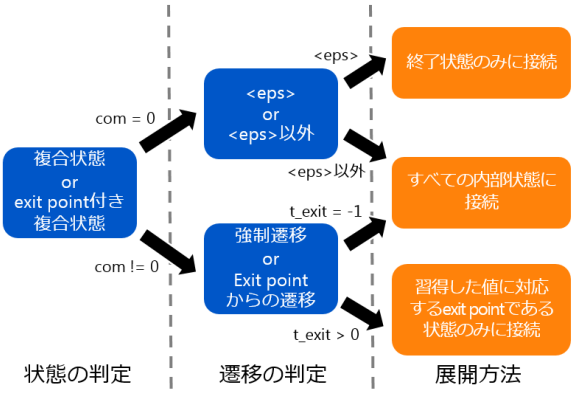

図 8　展開方法の判別

4.2.2 複合状態の展開方法

従来システムと同様に作成した状態遷移図を FST ファイルに書き出すためには，ミーリ・マシンの状態遷移図になるように複合状態を展開しなければならない．展開の方

法は UML のステートマシン図と基本的には同様である．図 9 のように exit point 付き複合状態に接続している遷移が exit point からの遷移であったときは，exit point に登録されている状態にのみつなぐ．次に exit point 付き複合状態に接続している遷移が強制遷移であったときは，図 10 のように exit point 付き複合状態に接続している遷移をすべての内部状態につなぐ．これらによって exit point を導入することで exit point 付き複合状態の処理終了時の遷移を表現した exit point からの遷移，ユーザ割り込みによる遷移を表現した強制遷移を表現することができる．

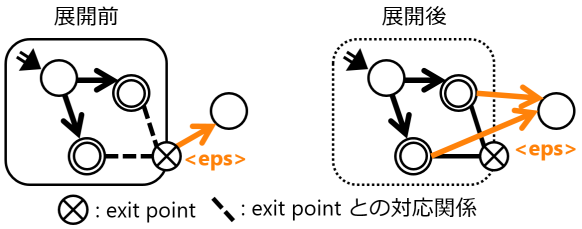

図 9　exit point からの遷移の展開方法

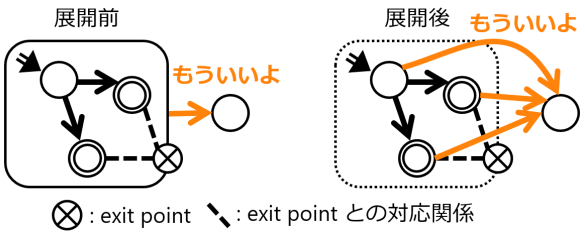

図 10　強制遷移の展開方法

4.3　編集画面切替機能

この節では，状態遷移図データについて説明した後，各 exit point 付き複合状態を状態遷移図に基づいて編集するために，ユーザが選択した exit point 付き複合状態に切り替える方法，管理方法について説明する．

4.3.1　状態遷移図データ

状態遷移図データは状態遷移図を記憶するデータである．状態遷移図データのデータ構造は 4.1.5 節で説明した複合状態を扱うクラス Module State，状態を扱うクラス State，遷移を扱うクラス Transition で構成されており，図 11 がデータ構造の例である．状態遷移図データを作成すると図の複合状態 root，main，状態 S が自動的に作成され，ユーザは main 以下の部分を編集することが可能である．

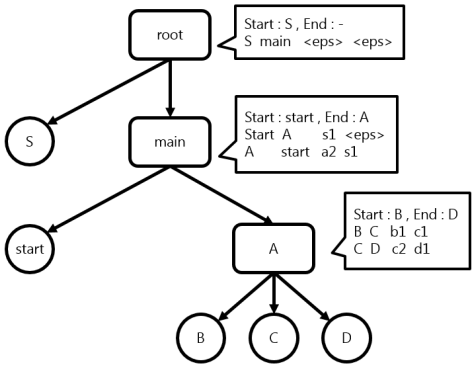

図 11　状態遷移図データのデータ構造例

4.3.2　編集画面管理方法

提案システムでは，現在編集画面に表示されている状態遷移図データの親または子の関係にある状態遷移図データに切り替え可能にするため，状態遷移図データの管理方法として木構造を採用している．提案システムの木構造では exit point 付き複合状態の状態遷移図データ，4.1.5 節で説明した int 型変数 com と同じ値を持つ int 型変数 tree_com をノードとして設定し，各ノードは親ノードを一つだけ，子ノードを複数もつ木構造である．

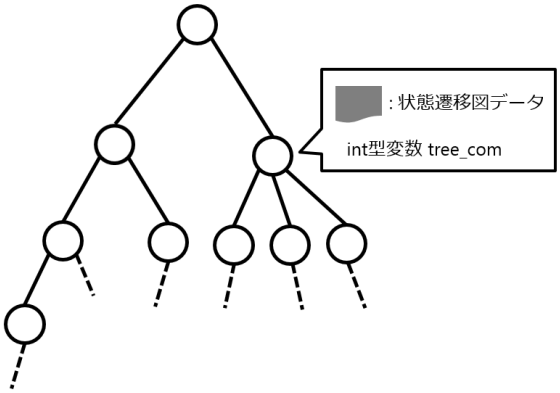

図 12　提案システムの木構造

4.3.3　編集画面切替機能

提案システムでは編集画面の切り替えは，現在表示されている状態遷移図データの親または子の関係にある状態遷移図データにのみ切り替えが可能である．

親ノードへの編集画面切替方法を 2 つの場合で説明する．現在のノードの親ノードが存在する場合，その親ノードの状態遷移図データを編集画面に読み込み，編集画面の切り替えを行い，現在のノードを親ノードに変更する．現在のノードの親ノードが存在しない，つまり現在のノードが根である場合，編集画面の切り替えが行えないため，何も行わずタブレット上にエラー文を表示させる．子ノードへの編集画面切替方法について説明する．ユーザが選択し

た exit point 付き複合状態の int 型変数 com を取得し，現在のノードの子ノードの中から int 型変数 tree com が一致するノードを検索する．次に一致した子ノードの状態遷移図データを編集画面に読み込み，編集画面の切り替えを行い，現在のノードを一致した子ノードにする．

5. プロトタイプシステム

5.1 プロトタイプシステムの概要

プロトタイプシステムは，Android SDK[14] を利用し，Android アプリとして実装した．開発環境は Android Studio[15] を用いた．Android タブレット端末として Xperia Tablet Z4(SGP712JP/B)[16] を用いた．Android バージョンは 6.0.1 である．

今回は従来システムである EFDE に exit point 付き複合状態モデルを導入し，編集画面切替機能を実装したシステムをプロトタイプシステムとしている．

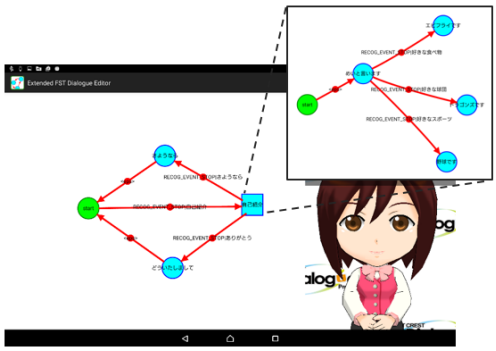

図 13 プロトタイプシステム

5.2 使用時の流れ

このプロトタイプシステムを使用して FST の編集をする流れは以下の通りである．

(1) プロトタイプシステムを起動する
(2) 状態や遷移を状態遷移図に追加していく
(3) 状態や遷移の設定を行う
(4) 各 exit point 付き複合状態の内部の状態遷移図を切り替える
(5) 2，3，4 を繰り返す
(6) execute ボタンを押し FST ファイルを書き出す

6. 評価実験

6.1 実験の目的

実験の目的は，提案システムが exit point 付き複合状態，exit point の導入によって，状態，遷移数が増加しても対話シナリオの編集が容易であるか，また複雑な対話シナリオが作りやすくなったかを実証することである．

6.2 実験方法

被験者 (本学学生 6 名) に，指定した会話ができるように Android 版 MMDAgent の対話シナリオを 2 つの手法を用いて編集してもらい，それぞれの状態遷移図の作成時間を測定する．その後 5 段階評価のアンケートに回答してもらう．またアンケートには自由コメント欄も用意した．編集してもらう手法は以下の 2 つである．被験者の半分は，先に手法 1 を後に手法 2 を行い，もう半分は逆の順番で行う．

6.3 アンケート

アンケートは 5 段階評価と自由コメントを用意した．アンケートの項目は表 4 の通りである．手法 1，手法 2 の両方に関して，それぞれの項目の評価をしてもらった．評価は 5 段階でしてもらい，評価基準は「1：あてはまらない，2：ややあてはまらない，3：どちらともいえない，4 :ややあてはまる，5：あてはまる」とした．アンケート項目を以下に示す．

表 4 アンケート項目

番号	質問内容
1	編集しやすい
2	対話シナリオが確認しやすい
3	状態遷移図が見やすい
4	思い通りの会話が作れる
5	複雑な会話が作れる
6	作業が面倒だ
7	間違えやすい

6.4 実験結果と考察

アンケートの各項目について評価の平均をとったグラフを図 14 に示す．項目 1 から項目 6 については数値が高いほど良い評価で，項目 7 から項目 9 については数値が高いほど悪い評価である．

ほとんどの項目について，提案システムを用いた手法 2 の方が良い評価を得ていることがわかる．項目 1，3 については提案システムの評価がどれも 4 以上と評価が特に高い．特に，項目 3 については従来システムの評価が 1.67，提案システムの評価が 4.50 となっており，従来システムを用いた手法 1 は状態遷移図が見づらいという結果が出た．この結果から，提案システムは exit point 付き複合状態，exit point を導入したことにより状態遷移図が見やすくなったと考えられる．また，項目 5 では手法 2 のほうがよい評価を得ていることから，従来システムより複雑な対話シナリオが作成できるようになったと考えられる．

項目 6 では手法 1，2 で差がなく，自由コメント欄で「ボタンが追加された分操作を覚えるのが大変」といった回答があり，作業が面倒だという点では手法 1 と変わらない評価になったと考えられる．

また，編集にかかった時間の平均を表 5 に示す．表 5 の通り被験者全員の平均で，従来システムを用いた手法にかかった時間は 5 分 27 秒，提案システムを用いた手法にかかった時間は 3 分 51 であり，1 分 36 秒早くなった．この結果より，従来システムで編集するより提案システムで編集する方がより速く編集できることが分かった．

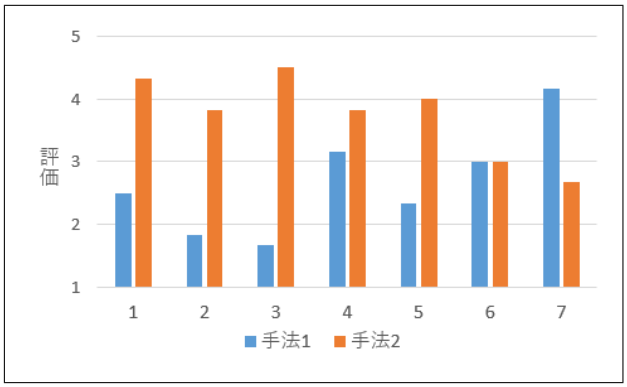

図 14　アンケート結果

表 5　編集にかかった時間平均

	手法 1 でかかった時間	手法 2 でかかった時間
平均時間	5 分 27 秒	3 分 51 秒

7. 関連研究

7.1 astah* professional

(株) チェンジビジョンのエンジニア向けのモデリングツール astah*professional[17] は，UML，ER 図，DFD，フローチャート，CRUD，要求図を描画できるモデリングツールである．ステートマシン図描画の場合，ユーザは編集画面上部の状態，遷移などを選択，配置することでステートマシン図を描画することができる．また，複合状態内部も編集可能であり，astah*professinal では複合状態内部を図のように直接描画することが可能である．

8. おわりに

本研究では，exit point 付き複合状態モデルを提案し，複合状態を状態遷移図に基づいて編集可能な EFDE の実現法について述べた．また，提案した実現法をもとにプロトタイプシステムを実装した．そして，評価実験として実際にプロトタイプシステムを利用してもらうことで，システムの有用性を示した．

評価実験により，状態遷移図が見やすいという項目については従来システムの評価が 1.67 となっており，従来システムでは状態遷移図が見づらいという結果が出た．この結果から，提案システムは exit point 付き複合状態モデルを導入したことにより状態遷移図が見やすくなり編集が容易になったと考えられる．また，複雑な会話が作れるという項目については提案システムのほうがよい評価を得ていることから，従来システムより複雑な対話シナリオが作成しやすくなったと考えられる．

今後の課題として，exit point 付き複合状態が増加すると，現在ユーザがどの exit point 付き複合状態を編集しているか，exit point 付き複合状態同士の関係を覚えておくことが大変であるため，ユーザがどの exit point 付き複合状態を編集しているかを確認できる機能を実装することが挙げられる．

謝辞

本研究は JSPS 科研費 26330136，25700009，科学技術振興機構 CREST，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] Siri
入手先 ⟨https://www.apple.com/jp/ios/siri/⟩ (2017.4.20 参照)

[2] しゃべってコンシェル
入手先 ⟨https://www.nttdocomo.co.jp/service/shabette_concier/⟩ (2017.4.20 参照)

[3] Akinobu Lee, Keiichiro Oura, Keiichi Tokuda, MMDAgent - A fully open-source toolkit for voice interaction systems, Proceedings of the ICASSP 2013, pp. 8382-8385, 2013.5.

[4] MMDAgent
入手先 ⟨http://www.mmdagent.jp/⟩ (2017.4.20 参照).

[5] Android
入手先 ⟨https://www.android.com/intl/ja/⟩ (2017.4.20 参照)

[6] 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，情報処理学会シンポジウムシリーズ IPSJ Symposium Series，Vol.2013，No.1，pp.675-680，2013.

[7] Keitaro Wakabayashi, Daisuke Yamamoto, Naohisa Takahashi, 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.

[8] IT 専科 状態マシン図（State Machine Diagram）
入手先 ⟨http://www.itsenka.com/contents/development/uml/state_machine.html⟩ (2017.4.20 参照).

[9] Julius
入手先 ⟨http://julius.osdn.jp/⟩ (2017.4.20 参照)

[10] Open JTalk
入手先 ⟨http://open-jtalk.sp.nitech.ac.jp/⟩ (2017.4.20 参照)

[11] MikuMikuDance
入手先 ⟨http://www.geocities.jp/higuchuu4/⟩ (2017.4.20 参照)

[12] Bullet Physics
入手先 ⟨http://bulletphysics.org⟩ (2017.4.20 参照)

[13] Harel,D.,Statechart: A visual formalism for complex systems，Science of Computer Programming，Vol.8，No.3，pp.231-274，1987.

[14] Android SDK
入手先 ⟨https://developer.android.com/studio/index.html⟩ (2017.4.20 参照)

[15] Android Studio
入手先 ⟨https://developer.android.com/studio/index.html⟩
（2017.4.20 参照）
[16] Xperia tablet Z4
入手先 ⟨http://www.sony.jp/tablet/products/Z4/⟩
（2017.4.20 参照）
[17] astah*professinal
入手先 ⟨http://astah.change-vision.com/ja/product/astah-professional.html⟩
（2017.4.20 参照）.