インタラクティブ・ロボット基本ソフトウェアの開発

Development of a Programming Environment for Interactive Robots

原　功 1)　本村　陽一 1)　麻生　英樹 1)　河村　進 2)
Isao Hara　Yoichi Motomura　Hideki Asoh　Susumu Kawamura

1) 産業技術総合研究所　情報処理研究部門（〒305-8568 茨城県つくば市梅園 1−1−1　中央第２　E-mail: {isao-hara/y.motomura/h.asoh}@aist.go.jp ）
2) 株式会社カーネル（〒305-0004 茨城県つくば市柴崎字小太郎 68−17　E-mail: kernel@po.iijnet.or.jp ）

ABSTRACT. Various kinds of robots/appliances are now entering into our daily life. Smooth and flexible interaction between humans and such devices are becoming more and more important. In order to facilitate implementing complicated but natural interaction, we have developed a programming environment for interactive robots. The software includes a library for sensor/actuator modules, a module for editing and simulating motion control scripts, and a module for editing and handling probabilistic interactive scripts. We also implemented some vision and speech recognition/generation modules using the library as components for interactions.

１．背景

ペットロボットや携帯端末に代表されるように、様々な種類のロボットや情報アプライアンスが、われわれの日常生活に浸透しつつある。それに伴い、ユーザである人間とそうした多様なロボットやアプライアンスとの間の自然なインタラクションを効率よく実現することが次世代のインタフェースにかかわるソフトウェア開発の重要な課題のひとつになっている。

ロボットや携帯端末などのためのソフトウェアの開発は、これまで、機種依存性やタスク依存性が高いものであったが、今後の開発ニーズの爆発的な増加に対処するためには、汎用性の高い枠組みでのソフトウェア開発が望まれる。そのためには、使い易いソフトウェア開発環境を、以下の二つの意味での汎用性が高い形で実現することが求められている。

- 開発環境自体の汎用性
- その環境で開発したプログラムの汎用性

また、ユーザと情報システムとの間のインタラクションを円滑にするためには、システムがユーザの状況や意図を理解しながら応答することが望まれる。しかしながら、人間の状況や意図は多くの場合直接観測できず、不確実性が高い。

この問題を回避するために、現在の多くのインタラクティブシステムでは、定型的なパターンに沿ったインタラクションにユーザを誘導する、システム主導のインタラクションが採用されている。この方法によれば、ユーザの状況や意図の不確定性をあらかじめ回避することができるが、結果として、ユーザにとっては冗長な入力を強制されるなど、使いにくいものになっていることが多い。この問題を解決して、インタラクションをより柔軟で制約の少ないものにするためには、そうした不確実性に対処するためのメカニズムが必要とされる[1, 2, 3,4]。

２．目的

このような背景のもと、われわれは、自律移動対話ロボットのためのソフトウェアの開発を行ってきたが[5]、そうした中で、統合的なソフトウェア開発環境の必要性を痛感した。そこで、平成１２年度未踏ソフトウェア創造事業において「パーソナルロボット用アプリケーション開発環境」の開発を提案し、採択された。平成１３年度には引き続き「インタラクティブ・ロボット基本ソフトウェアの開発」を提案し、採択された。本開発の目的は、人間とインタラクティブ・ロボットとの間の自然でストレスの少ないインタラクションを効率よく実現するための統合的なソフトウェア開発環境を構築することである。

ロボットをプログラムするためのプログラミング言語やソフトウェア開発環境はこれまでにもいろいろ開発されている。たとえば、松井らはオブジェクト指向 Lisp での一種である EusLisp を開発した[6]。SONY は Open-R というソフトウェアアーキテクチャを提唱している[7]。また、最近では、ヒューマノイドロボット用のソフトウェア開発プラットフォームも開発されている[8]。これらに対して、既に述べたように、本開発では、

- 多様な構造のロボットに対応できる汎用性

を持たせることをめざした。そのために、開発環境を複数の階層的なモジュールに分割し、一部のモジュールを入れ替えるだけで、多様な構造のロボットに対応できるような設計を行った。また、この開発環境を用いて作成するソフトウェア自体もできるだけ再利用可能になるように、モジュール性を高めた設計を行った。

また、柔軟なインタラクションをプログラミング可能にするために

- 非決定的制御機構の導入

を行った。具体的には、上位レベルの制御プログラムであるインタラクション・スクリプトが非決定的（確率的）

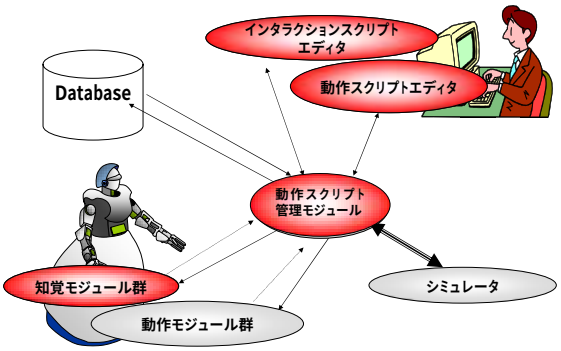

図 1 開発環境の概要

に動作することをめざした。

汎用性および普及の容易さを考慮してプラットフォームとなる OS としては Linux を採用した。実時間 Linux への対応も容易であると考えている。

３．開発したソフトウェア

今回の開発では、インタラクティブ・ロボット用のソフトウェアを、大きく

- 動素・感覚素プログラム
- 動作スクリプト
- インタラクション・スクリプト

の３つの階層に分けた。

動素・感覚素プログラムは、ロボットのアクチュエータやセンサに最も近い（下位の）レベルのプログラムであり、モータを制御して要素的な動き（たとえば腕の関節を９０度回転させるなど）や要素的な感覚（たとえば、人間の顔の検出・認識や、音声認識など）を実現するものである。ハードウェア依存性が高く、また、ロボットの構造に対する依存性も高い部分であるが、できるだけ汎用性の高いものになるように、ライブラリ化を行っている。

次の動作スクリプトは、動素や感覚素を組み合わせたインタラクションの要素となる一連の動き(たとえば「手を振る」など）を記述するものである。このレベルは、ロボットの構造に対する依存性は高いが、ハードウェア依存性は少ない。

最後のインタラクション・スクリプトは、動作スクリプトを組み合わせた、ユーザとのインタラクション（たとえば、人を見つけて、挨拶して、用件を尋ねて、実行して、返答する）を記述するものである。このレベルでは、ロボットの構造に対する依存性もある程度は少なくできると期待される。

このような階層構造を取ることによって、作成したソフトウェアは、下位のモジュールを変更するだけで、多様な構造のロボットに対応できるようになることが期待できる。

この階層構造に応じて、ソフトウェア開発環境も

- 動素・感覚素プログラム開発用ライブラリ
- 動作スクリプト編集・シミュレーションモジュール
- インタラクション・スクリプト編集・コントロールモジュール

の３つから構成される設計とした。図 1に開発環境全体の概要を示す。以下、各々について詳しく述べる。

（１） 動素・感覚素プログラム開発用ライブラリ

動素・感覚素プログラムの作成のためのライブラリとしては、平成１２年度未踏ソフトウェア創造事業「パーソナルロボット用基本ソフトウェアの開発」の成果物の一部である、非同期・イベント駆動型プログラムを作成するためのライブラリを用いる[9]。このライブラリを用いることによって、各動素、感覚素を実現するプログラムをモジュール性が高い形で、かつ、モジュール間の通信機能や、上位モジュールとの通信機能を含めて、容易に開発することができる。図 2に、このライブラリを用いて開発することができるイベント駆動型モジュールの構造を示した。

平成１２年度には、このライブラリを利用して人間型ロボット Tmsuk IV 用の動素モジュールを作成したが、今年度の開発では、これに加えて、視覚モジュール、音声認識モジュール、音声発話モジュールを作成し、それぞれ以下のような機能を実装した。これらのモジュールは、具体的なインタラクションを実現するための汎用的な部品として用いることができる。また、必要な機能の

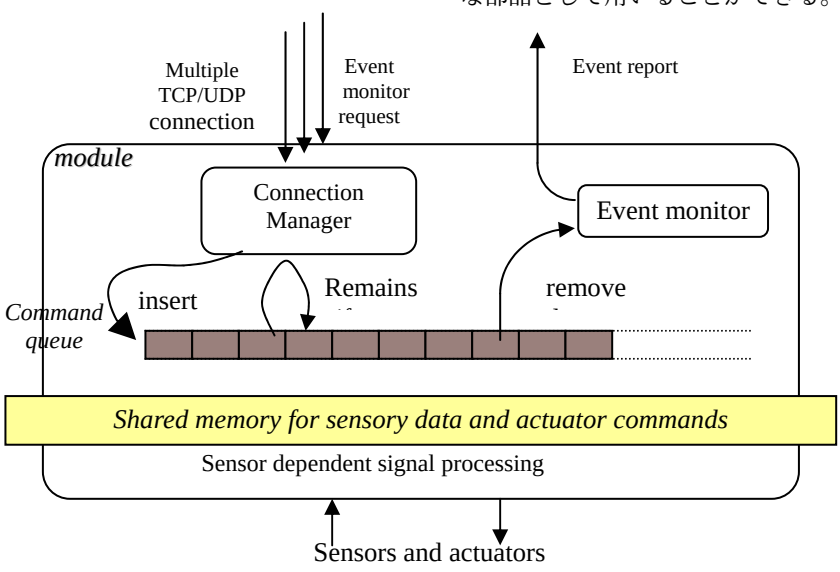

図 2 イベント駆動型ソフトウェアモジュールの構造

追加や修正も容易に行うことができる。

a)　視覚モジュール

視覚モジュールは、IEEE1394 で接続される小型カメラ（PointGray 社の FireFly）を入力デバイスとして、IPL(Intel Image Processing Library)および OpenCV(Open Source Computer Vision Library)を用いて作成された。上位モジュールから TCP/IP 接続を通して呼び出し可能な機能として、以下のような機能を実装した。

- オブジェクト移動方向検出
  コマンド：REPORT_DIRECTION
  引数：閾値
  戻り値：画像中の最大のオブジェクトの移動を８方向（上・下・右・左・右上・左上・右下・左下）のいずれかで検出したときにイベント発生を通知する。
- 画像中に存在する顔領域の発見
  コマンド：DETECT_FACE
  引数：閾値
  戻り値：画像中の顔（両眼のテンプレートとマッチするオブジェクト）を検出したときにイベント発生を通知する。
- はい・いいえの検出
  コマンド：DETECT_YES_OR_NO
  引数：閾値
  戻り値：顔が発見されていることを前提として、その領域の水平および垂直方向の動きにもとづき、「はい」か「いいえ」を認識し、イベント発生を通知する。
- バイバイの検出
  コマンド：DETECT_BYE_BYE
  引数：閾値
  戻り値：画面内でバイバイ（手を振る）動作を検出したときにイベント発生を通知する。

b)　音声認識モジュール

音声認識モジュールは、電総研で開発された、ネットワーク対応不特定話者日本語連続音声認識エンジン RVCP niNja の出力を受け取り、他のモジュールからの要求があったときに、認識結果と評価値（尤度）を返す機能を持つ

- 音声認識
  コマンド：LISTEN_SPEECH
  引数：音声認識エンジンの ID
  戻り値：音声認識エンジンからの認識結果文字列と評価値を通知する。

c)　音声発話モジュール

音声発話モジュールは、富士通製の音声合成エンジンを用いた Linux 版日本語音声合成ライブラリを利用して、以下の機能を実現している。

- ローマ字記述の日本語文章の発話
  コマンド：TALK
  引数：ローマ字で記述された日本語文章
  戻り値：発話終了時にイベントを通知し 0 を返す。
- ファイルに保存された日本語文章の発話
  コマンド：TALK_FILE
  引数：ファイル名
  戻り値：ファイルが存在すれば発話終了時にイベントを通知し、0 を返す。ファイルが存在しなければ即座に－1 を返す。
- 発話音声ピッチの切り替え
  コマンド：TALKSEX
  引数：0（男性）、1（女性）
  戻り値：ピッチを切り替えて、現在のモード（0 か 1）を返す。

（2）　**動作スクリプトの編集・シミュレーション**

動作スクリプト編集･シミュレーションモジュールは、ロボットの動作（ポーズおよびその系列）を作成、記録、実行するためのエディタおよび動作スクリプトをプレビューするためのシミュレーション機能を持つ。

本ソフトウェアは、平成１２年度に作成した、動作系列生成・記録用モジュール[9]を XML 対応に拡張し、シミュレーション機能を追加したものである。このソフトウェアの主なクラス構成を図 3に示す。

GUI 状態表示パネルは、キーボードを用いたデータ入力のためのインタフェースである。この他に、データ入力用のインタフェースとして、ジョイスティック（または、ゲームパッド）および実ロボットと同等の TCP ソケットを持つ。これらのインタフェースを介して、ロボットのモデルに対して指示を送ることによって、３D View に表示されたロボット画像を制御できる。3DView の画面および GUI パネルを図 4図 5に示す。

このソフトウェアを用いて、ユーザは、以下のような手順で、ロボットの動作を画面上で確認しながら、ロボットの一連の動作を記述する動作スクリプトを作成・編集・命名・保存することができる。

a)　キーボードによる動作スクリプトの作成

前提条件：

Java3D が動作する環境であること。

メインフロー：

1. 動作スクリプトを作成するユーザは、キーボードを用いて、シミュレータ内のロボットの両手、頭の姿勢を操作（キーバインドは、別途コンフィグレーションファイルによって指定する）する。
2. シミュレータ内の動作スタックへその姿勢を記録する。
3. 1,2 の操作を繰り返すことで、動作スタック内に一連の動作列が記憶する。
4. この動作列が、正しく動作しているかどうかを、プレイボタンを押すことで確認する。
5. 正しい動作をしている場合は、セーブボタンによってファイルまたは、データベース内に名前をつけて保存する。
6. 4 で動作が正しくない場合には、エディタを起動することで動作スタックを編集し、正しい動作に変更するか、もしくは、現在の動作スタックを破棄し、最初から動作列の作成を行う。

b)　ジョイスティックまたはゲームパッドによる動作スクリプトの作成

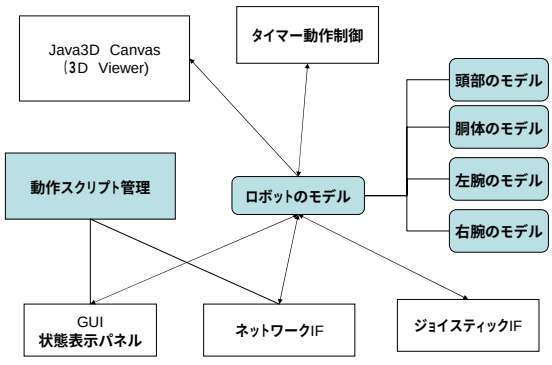

図 3 動作スクリプト編集・動作シミュレータの構成

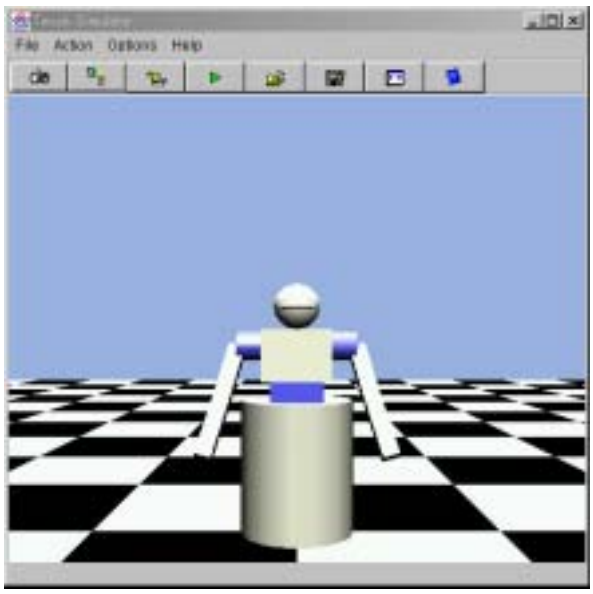

図 4 シミュレータの３DView 画面

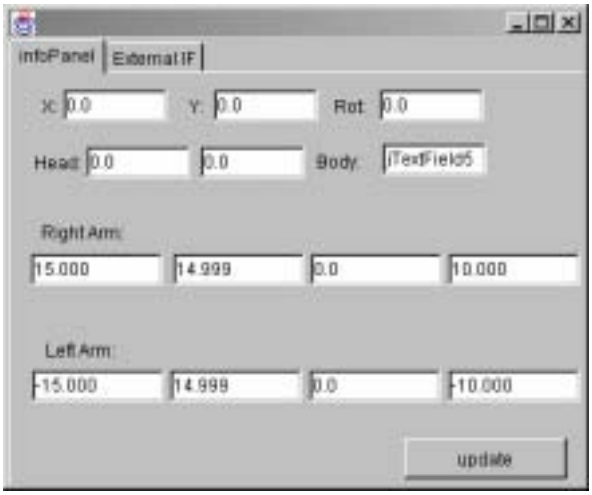

図 5 シミュレータの GUI 状態表示パネル

前提条件：

ジョイスティックまたは、ゲームパッドが使用可能であり、Java3D が動作すること。

メインフロー：

1. キーボードの場合と同様にシミュレータ内のロボットのポーズを、ジョイスティックを用いて指定し、記憶させる。ゲームパッドのキーバインディングは、コンフィグレーションファイルによって指定されたものを使用する。
2. 記憶された動作列を、プレビューし、正しい動作であれば、それをファイルまたは、データベース内に名前をつけて保存する。

c) ネットワークインターフェースを介した動作スクリプトの作成。

前提条件：

シミュレータをサーバーとして起動し、クライアントプログラム（telnet など）を起動すること。

メインフロー：

1. クライアントプログラムから、MOVE_JOINT や MOVE_HEAD などのコマンドを発行し、シミュレータ内部のロボットを操作する。ここで用いているコマンドは、実際のロボットと同様に非同期・イベント駆動型に動作するものであり、実装されているコマンドの詳細については[9]を参照。
2. 目標の姿勢をとった状態で、REGISTER コマンドを用いて、動作スタックに、その姿勢を記憶させる。
3. PLAY コマンドで、動作スタックの動作をプレビューし、その動作列を、NAME コマンドを用いて、動作スクリプトとしてデータベースに保存する。

このようにして作成された動作スクリプトは、XML 文書としてデータベースあるいはファイル内に保存される。ここまでで作成した動作スクリプトは、全体の姿勢を、動作列として保存したものであるが、動作スクリプトを直接外部エディタ等で編集することもできる。この場合には、拡張タグを用いることで、姿勢ベクトルの列だけではなく、動作スクリプト名の列を書くことで、動作スクリプトを記述することも可能である。

図 6に動作スクリプトのフォーマットを示す。動作スクリプトの各要素は以下のとおりである。

- 要素名：actionscript
  この要素は、動作スクリプト内の最上位要素であり、このタグ内の内容を 1 つの動作スクリプトとして定義している。また、属性として、Name を持ち、それぞれの動作スクリプトの名前を定義する。
- 要素名：action
  動作スクリプトの下位要素であり、動作の基本となるロボットの 1 つのポーズ（姿勢）または、動作スクリプトを定義する。
- 要素名：pose
  動作スクリプトの最下位の要素であり、ロボットの関節角ベクトルで表されたものであり、最も基本的な要素となる。
- 要素名：name
  Pose と同様に動作スクリプトの最下位の要素であり、既存の動作スクリプト名で表されたものである。
- 要素名：head
  Action 要素の中で用いられ、頭部の動作（または姿勢）を記述したものであり、Action で定義された動作（または姿勢を）上書きする形で定義することができる。このタグの内容として、上記の Pose タグまたは Name タグで、その動作を指定する。
- 要素名：leftarm
  Action 要素の中で用いられ、左腕部の動作（または姿勢）を記述したものであり、Action で定義された動作（または姿勢を）上書きする形で定義することができる。このタグの内容として、上記の Pose タグまたは Name タグで、その動作を指定する

```
<?xml versions=”1.0” ?>
<actionscript  name=”名前”>
  <action>
     <pose>姿勢</pose> または
          <name>動作スクリプト名</name>
     <head>頭の動作</head>
      <body>体の動作</body>
      <leftarm>左手の動作</leftarm>
      <rightarm>右手の動作</rightarm>
  </action>
  <action> …..</action> …..
 </actionscript>
```

図 6 動作スクリプトのフォーマット

```
<?xml version="1.0" ?>
<actionscript name="byebye" count="1">
  <action>   <name>byebye2</name>  </action>
 <action>
  <name>stop</name>
  <letarm><name>byebye2</name></leftarm>
 </action>
 <action>
  <name>byebye2</name>
  <rightarm><name>stop</name></rightarm>
 </action>
</actionscript>
```

図 7 動作スクリプトの例

- 要素名：rightarm
  Action 要素の中で用いられ、右腕部の動作（または姿勢）を記述したものであり、Action で定義された動作（または姿勢を）上書きする形で定義することができる。このタグの内容として、上記の Pose タグまたは Name タグで、その動作を指定する。

図 7に上記のようにして作成した動作スクリプトの例を次に示す。この例は、両手を振ったあと、左手で 2 度振る場合の動作スクリプトである。記述されている byebye2 は、両手を挙げて振る動作であり、stop は、その場で停止する動作を表している。左手だけを振るという動作は、二通りの方法で記述されている。

（3） インタラクティブ・スクリプト編集・制御

インタラクティブ・ロボットを制御するための最も上位のプログラムであるインタラクション･スクリプトは、確率的な状態遷移を行うネットワークと各状態で実行される Agent オブジェクトとから成る。今回開発したインタラクティブ・スクリプト編集・コントロールソフトウェアは、状態遷移図の作成、Agent オブジェクトの定義、状態遷移、条件付確率の学習、動作実行時の状態遷移コントロールを行うものである。

インタラクション・スクリプトの作成にあたっては、UML(Unified Modeling Language)[10]と同様の手順に従い、システムの要件を表すインタラクションシナリオからユースケースを経て、状態遷移図を作成する。この場合の状態とは、ある特定の状況を仮定し、その上で動作する Agent のことである。ただし一般の状態遷移図を非決定的に拡張し、確率的な状態遷移を許すものになっている。この状態遷移確率および効用をパラメータとして事例から再設定することで、ロボットの挙動がより適切になるように学習することも可能にしている。図 8に状態遷移ネットワークのイメージを示す。

以下では、インタラクション・スクリプトの各要素についてより詳しく説明する。

a） ノード

ある状態での動作に対応する機能モジュールをノードとして定義する。ノードはチャネルからの入力と、前の状態（ノード）から引き継いだ状態変数を受け、適切なアクションを実行し、チャネルに対する出力動作と、内部状態変数値の更新を行い、次の状態遷移先を決定する。（この状態遷移先の決定は意思決定論的に行う。すなわち効用値として次の遷移ノードの中の値を参照し、現在の状況の元で期待効用が最大のものを選ぶことで最適な決定を行う。）

b） チャネル

動作機能を実現した下部モジュールとの接続を行う

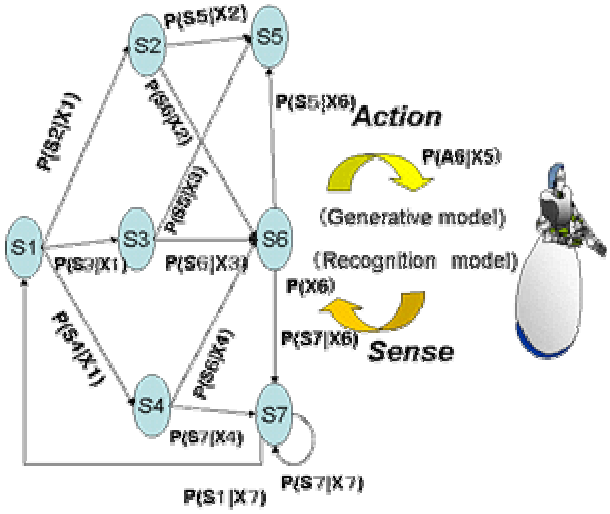

図 8 確率的状態遷移ネットワーク

入出力ストリームをあらかじめ、ポート番号、ホスト名により定義しておきこれをノードの入出力とする。このチャネルから入力を受ける場合は専用のスレッドにより常時入力を監視する。

c） Agent オブジェクト

各ノードでは Agent クラスを継承してユーザが生成した各種 Agent オブジェクトを一つ選択して保持する。（パッケージの中にいくつかのサンプルがあるのでユーザはこれを継承・修正するなどして拡張することができる。）ここで指定したオブジェクト(JAVA のクラスファイル)は JAVA のリフレクション機能によって、インタラクション・スクリプト編集時に初めて読み込まれ、実行時にインスタンスが生成される。したがって、編集時にソースコードを修正し、コンパイルして利用することも可能である。

Agent クラスの中では先に指定したチャネルからの入出力と隣接ノードから受け取ることのできる状態変数を参照できる。また状態遷移に伴って、以下の３つのメソッドが実行される。

void entryAction
void exitAction,
boolean duringAction

entryAction はノードに制御が移った時点で実行され、前の状態や入力の観測などを行う。exitAction はノード内の制御が終了する時点で実行され、次の遷移先の決定などを行う。duringAction はノード内の状態におけるメインループの中で実行され、現在の状態に留まる限り真値を返すことで動作が繰り返される。何らかの終了条件の判定を行い、次の状態へ遷移する時に false を返すことでループから脱出する。

d） 状態変数

現在の状態に関する情報を過不足なく表現した変数（ベクトル）たとえば動作を実行した結果についての場合分けなどを行うのに用いる。（例：Val=[Success, False]）変数の値は動作の実行によって確定されるが、動作の実行結果が不完全だったり、動作の実行が不可能な場合でも、次に述べる条件付確率による予測を行い、確率分布を計算することもできる。

e） CPT(Conditional Probability Table)

あるノードの状態変数と次のノードの状態変数の間に確率的な依存関係があるとき、これをシステムが記録した動作履歴中の頻度から条件付確率として求めておくこ

とができる。知覚動作による観測が不完全で状態変数が確定できないよう場合に、この条件付確率を使った確率推論によって予測を行うことができる。また動作実行を行う前におおまかな予測として実行結果の確率を利用することがある。

f) CNT(CouNt Table)

CPT を事例の頻度から求めるために、データベース中にある該当事例の件数を数えてテーブルとして格納する。これを個々の場合について正規化したものが条件付確率となる。

g) CPA(Conditional Probability Approximator)

CNT 内の頻度が少なかったり、場合によって頻度にばらつきが多い場合には CNT を単純に正規化しただけでは条件付確率の信頼性が著しく低いことが起こりえる。そのような場合に、線形回帰やニューラルネットワークなどの近似モデルを導入し、条件付確率を滑らかに近似することで信頼性を高める。

h) UT(UtilityTable)

その状態における変数の取りえる値のそれぞれについて効用値を割り当てる。

i) パス

状態遷移の始点・終点、範囲を指定するためにパスオブジェクトが導入されている。デフォルトでは最初に生成したノードが始点、最後に生成したノードが終点、すべてのノードが遷移の対象となる。ユーザが個別に始点、終点、範囲となるノード集合を指定することもできる。

システムがインタラクションスクリプトの実行を開始すると、パスで指定された始点から順に状態遷移図をたどって各ノードに対応する Agent を実行し、終点の Agent の実行が完了すると動作を停止する。また確率推論の実行の際にも計算する範囲を限定するためにパスオブジェクトを利用することができる。図 9に状態遷移ネットワークの構造、図 10に各ノードのエージェントの構造を示した。

４．まとめ

平成１３年度未踏ソフトウェア創造事業において開発した、インタラクティブ・ロボットのためのソフトウェア開発環境について述べた。最も下位の動作素・感覚素プログラムを開発するためのライブラリを用いて、汎用的に利用可能な部品として、視覚モジュール、音声認識モジュール、音声発話モジュールを開発した。また、それらを組み合わせるための動作スクリプトエディタおよびシミュレータを開発した。動作スクリプトは XML で記述され、シミュレータの３次元画面でプレビューすることができる。さらに、最も上位のプログラムであるインタラクション・スクリプトを編集・コントロールするためのモジュールを開発した。

本基本ソフトウェアは、高いレベルまでのモジュール性と抽象化、非決定的な行動決定機構や学習機構の組み込み、オブジェクト指向、状態遷移指向なプログラム開発モデルの導入、などの点で独自性が高いものであり、今後、具体的な使用を重ねて行くなかでさらに改良を進めてゆきたい。

５．参加企業及び機関

- 原功、本村陽一、麻生英樹（産業技術総合研究所）
- 株式会社カーネル
- 財団法人　日本産業技術振興協会

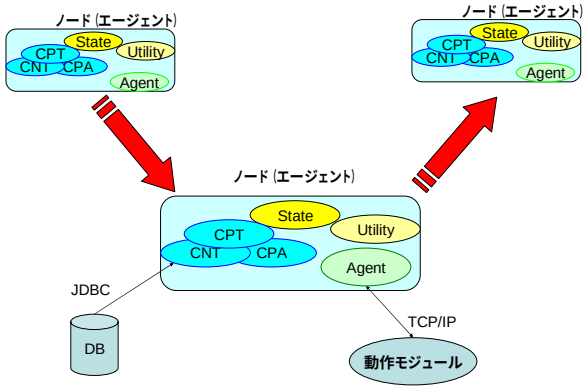

図 9 状態遷移ネットワークの構造

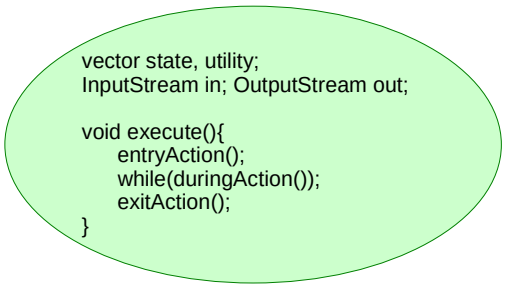

図 10 エージェントの構造

６．参考文献

[1] 原功、本村陽一：自律ロボットのための状況依存マルチエージェントアーキテクチャ，第５回ロボティックスシンポジア予稿集, pp.86-91 (2000)

[2] Y.Motomura, I.Hara, K.Itoh, H.Asoh, T.Sato: Task, Situation, and User Models for Personal Robots, IJCAI2001 Workshop on Reasoning with Uncertainty in Robotics (2001)

[3] 本村陽一、佐藤泰介：確率的タスクモデルによるインタラクティブシステムの制御と学習，2002 年人工知能学会全国大会予稿集，1B1-04　(2002)

[4] 原功：インタラクティブ・ロボットにおける行動制御機構，ロボティクス・メカトロニクス講演会 2002 予稿集 (掲載予定) (2002)

[5] T.Matsui, H.Asoh and I.Hara：An Event-Driven Architecture for Controlling Behaviours of the Office Conversant Mobile Robot, Jijo-2，Proceedings of 1997 IEEE International Conference on Robotics and Automations, pp.3367-3371 (1997)

[6] 松井俊浩、原功、中垣博文：EusLisp リファレンスマニュアル，電子技術総合研究所テクニカルレポート ETL-TR-95-19 (1995)

[7] Open-R SDK, http://www.aibo.com/openr/

[8] Open-HRP, http://www.is.aist.go.jp/humanoid/openhrp/Japanese/

[9] 麻生英樹、原功、本村陽一：パーソナルロボット用アプリケーション開発環境, ITX2001 (2001)

[10] UML1.3, http://www.omgi.org/technology/documents/spec/modeling.html