パーソナルロボット用アプリケーション開発環境

An Application Development Environment for Personal Robots

麻生　英樹　　原　功　　本村　陽一
Hideki Asoh　Isao Hara　Yoichi Motomura

産業技術総合研究所　情報処理研究部門　メディアインタラクショングループ
〒305-8568 茨城県つくば市梅園１－１－１中央第２
E-mail: h.asoh/isao-hara/y.motomura@aist.go.jp

ABSTRACT. Personal robots such as pet robots are becoming popular. In order to accelerate developing various applications on various personal robots, programming environment is necessary. We are developing an application development environment for personal robots. Current system includes C libraries for creating elemental function agents, and Java class libraries for creating integrated function agents. Using the libraries a robot daemon program and a control panel program was implemented as a sample application.

１．背景

ロボットはこれまで、主に、工場をはじめとして、発電所、建設現場、宇宙、海洋、極地などといった状況で、人間とは隔離され、人間を上回る作業精度、作業効率、パワー、耐久性などを実現するために用いられてきた。しかしながら、近年のペットロボットの商品化などに象徴されるように、今後１０年くらいの間に、ロボットの活躍の場は家庭やオフィス、病院など、人間の日常生活の場へと拡大し、普通の人が日常的になんらかの形でのロボットを利用し、ロボットとインタラクトするようになることが予想されている。

こうした状況は、コンピュータにおけるメインフレームからパーソナルコンピュータへの移行と対比して考えることができる。メインフレームとパーソナルコンピュータとが同じ「コンピュータ」と呼ばれながらも、そのアプリケーションや要求仕様において全く異なる機器であるように、従来のロボットとパーソナルロボットとでは、ロボットとしての仕様や想定されるタスクが大きく異なる。たとえば、パーソナルロボットに必要とされる機能は、精密なマニピュレーション能力や強大なパワーではなく、むしろ、実環境において人間と柔軟にインタラクトするための機能、すなわち、人間を発見し、人間の発する音声やしぐさ、表情を通じた働きかけを理解する機能、適切な間合いを取りつつ人間と一緒に移動するための自律移動機能、音声やジェスチャ、表情などの身体的な機構を用いた情報表出機能などである。

したがって、パーソナルコンピュータのための OS やミドルウェア、アプリケーションが必要とされたように、パーソナルロボットにはそのための OS やミドルウェア、アプリケーションの開発が必要とされている。また、それらのソフトウェアは、従来のロボット制御ソフトウェアのようにロボット個別的なものではなく、「パーソナルロボット」に必要とされる機能をカバーするような汎用性を持つとともに、具体的なセンサや機械デバイスに依存する部分をできるだけモジュール化した形で実現されていることが望ましい。

我々は、これまでに、オフィスロボット「事情通」として、オフィスのような日常的な環境で、移動し、人間と音声対話などのインタラクションを行うロボットを開発してきた[1,2]。そこでの経験からも、そうしたインタラクティブなロボットのためのさまざまな機能を統合するソフトウェア開発のむずかしさとそれを支援する環境の重要性、必要性を認識した。

２．目的

本開発の目的は、第一義的には、多様なパーソナルロボットの開発と普及を促進することである。そのためには、さまざまなパーソナルロボットのための種々のアプリケーションを、ハードウェア依存性少なく開発できるようにすることが必要である。その目的に向けた第一歩として、本開発では、比較的汎用なロボットハードウェア上で、モジュール化されたセンサや機械デバイス依存のドライバモジュールと、機器依存部分と非依存部分とを切り分けたミドルウェアとを組み合わせたアプリケーション開発環境の開発を行った。

３．開発の内容

パーソナルロボットのアプリケーションプログラムは、ロボットに搭載されている各種のセンサからの情報にもとづいてイベントを検出し、現在の状態とあわせて適切なアクションを起動し、その結果を、また、センサによってモニタリングする。すなわち、センサイベントの監視とイベント発生に対応するアクション起動を繰り返す。そこにおけるミドルウェアの主な役割は、適切なタイミングで、適切な抽象レベルでのイベント監視を行って、その結果をアプリケーションに渡すことと、アプリケーションからの抽象的な行動依頼をデバイスレベルにブレークダウンして実行することである。

こうしたソフトウェアは従来から、ロボット個別的な形では作成されてきているが、今回の開発では、「パーソナルロボット」と言えるようなカテゴリのロボットに

対してできる限り汎用的に利用できるものとなることを考慮した。また、異なるロボットプラットフォームや異なる機械デバイス・センサ間の移植が容易であるようなソフトウェアを実現することを考慮した。

パーソナルロボットのハードウェア構成としては、通常の PC に各種センサやアクチュエータのインタフェースボードが接続されるという形態を想定した。当面はこれが最もコストが安い実現方法として特に研究用には広く用いられると考えられるためである。実際の開発では、こうした形態のロボットとして、（株）テムザックが販売しているテムザック IV（図１） を開発のベースおよび評価用に使用した。テムザック IV は、双腕を有する人間型の移動ロボットで、移動のための自由度２、腕の自由度７ｘ２､指の自由度３ｘ２､胴体の自由度１、首の自由度２の合計２５自由度の可動部分を持つ。それぞれの自由度は、モーターによって駆動され、ポテンショメーターによって各軸の回転角度を検知することができる。

このロボットを選んだ理由は、

・インタラクションのための表現力が大きいこと。
・各自由度の駆動およびセンシングが入手しやすい市販部品によって実装されており、制御等がむずかしくないこと。

である。また、本開発の開始時には、同程度の表現力すなわち、表現自由度を持つロボットで市販されているものとしては唯一のものであった。

テムザック IV のソフトウェアとしては遠隔操縦のためのものが Windows ベースで開発され、添付されているが、今回の目的のためには使えないため、デバイスドライバも含めて開発を行う必要があった。

ＯＳには Linux を用いた。これは、パーソナルロボットの研究から商品開発に至るまで幅広いユーザに利用されるとともに、ユーザの手によって成長してゆくことができるようなパブリックドメインのミドルウェアを実現することを目的としたためである。さらに、Linux 上でのロボットアプリケーション開発を容易にすることによって､センサやアクチュエータメーカが Linux 対応のデバイスドライバを積極的にサポートするようになることも狙いの一つである｡実時間 Linux の利用も検討したが、今回はまず通常の Linux 上で開発を進めた。実時間 Linux への移行には大きな問題点はないと考えられる。

これらのベースの上で、具体的には、以下のようなソフトウェアの開発を行った。

・デバイスドライバ
ロボットに固有のデバイスを利用するためのデバイスドライバである。本開発の主たるターゲットではないが、ミドルウェアの動作をロボット上で確認するために必要であるため、開発を行った。
・パーソナルロボット要素機能ライブラリ
パーソナルロボットのアプリケーション作成に必要とされる要素機能を抽象化した形で提供する要素機能エージェント（デーモン）の実装を支援するためのライブラリである。デバイス依存部分と、モジュール間の通信や機能実行管理などの共通部分、機能依存部分とを切り分けて、できるだけ汎用性や再利用性の高い形で実装した。
・パーソナルロボット統合機能開発環境
要素機能を組み合わせた統合機能タスクを記述・実行するための開発環境である。要素機能の利用を制御するための Java クラスとして実装した。
・機能統合サンプルアプリケーション
開発したライブラリおよびクラスを利用したサンプルアプリケーションとして、テムザック IV 用のロボットデーモンプログラムおよび、ロボットデーモンプログラムと通信してティーチング・プレイバックによる動作生成を行うための操作パネルプログラムを作成した。

図１　テムザック IV

４．開発の成果

（１）ロボット制御デバイス用デバイスドライバ

機能検証用のロボットであるテムザック IV に使用されているモーターおよびセンサ制御ボードのための Linux（kernel ver.2.2.X 以降）用のデバイスドライバ群として、

・A/D ボード用ドライバ
・D/A ボード用ドライバ
・DO ボード用ドライバ

の３種類のデバイスドライバを作成した。それぞれ、デバイスをオープン・クローズし、値の読み書きを行う機能を持つが、テムザック IV 上での機能は

A/D ボード：各関節に接続されたポテンショメータの値の読み取り
D/A ボード：各関節の駆動用モータの速度指令の設定
DO ボード：各関節のモータの始動/停止および回転の正逆の指令

である。

（２）パーソナルロボット要素機能ライブラリ

上位モジュールからの要求に従ってデバイスドライバを使用してテムザック IV の各関節の角度を並行的に制御するソフトウェアモジュールである。開発の目的のひとつである高い移植性を確保するために、

・ロボットに固有な部分(libtmsuk)
・ロボットに汎用的な部分(agentlib)

の二つに切り分けて実装されている。

a) ロボット固有な要素機能ライブラリ

ロボットに固有な部分は、デバイスドライバを使って各関節に指令を出すためのライブラリである。本プロジェクトで用いたプラットホームとなるテムザック　IV

に固有の機能を実装したライブラリ(libtmsuk.so)の開発行った。このライブラリでは、ロボットの初期化と各モータのコントロールを行う関数や各関節の読みを得るための関数等を提供する。このライブラリで提供する主な関数を以下に示す。

・initRobot()
記述：int intRobot()
説明：ロボット制御に必要なデバイスのオープンと初期化。ロボットのモータドライブのために、A/D, D/A, DO の各デバイスをオープンし、設定を行う。処理に成功すれば、1 を返し、失敗すれば、NULL を返す。

・load_cfg()
記述：int load_cfg(char *name)
説明：各関節の動作範囲、初期値等をファイルから読み込み設定。name という設定ファイルをオープンし、ロボットの動作範囲、初期値となる、関節角（ポテンショメータの読み）を設定する。
設定ファイルの内容は、
```
#
#   Min Main Init
#
600 3800 2200   # Head R/L
1000 2000 1000 # Head U/D
 ……
```
のような形式をもち、各行に最小値、最大値、初期値が空白で区切られた整数の列が記述したものである。このファイルでは、＃より後の文字列は、コメントとして認識される。

・updataState()
記述：void updateState();
説明：この関数が呼ばれると A/D ドライバを介してポテンショメータの値を読み込み、State という変数へセットする。State は、
```
unsigned int State[AD_CHANNELS];
```
として、ライブラリ内で定義されている。外部プログラムからこの値を参照するためには、
```
extern unsigned int State[AD_CHANNELS];
```
と記述すればよい。

・moveJoint()
記述：int moveJoint(int n, int val);
説明：n 番目の関節に対して動作目標値 val を設定する。n は、関節の場所を示す整数値であり、移動目標のポテンショメータの読みである。この関数によってモータの状態（正/逆回転、停止）が配列 JointState[]に設定される。
関節とその場所を表す整数値との対応は、tmsuk.h 内に記述されている。

・start_motor()
記述：void start_motor();
説明：モータの状態（正/逆回転、停止）が配列 JointState[]の記述にしたがって、関節のモータを駆動/停止させる。なお、この関数が呼ばれるまで、モータは、始動/停止されない。

b) ロボット汎用的なライブラリ

一方、ロボットに汎用的な部分は今回の開発の要のひとつであり、以下の機能を持つ要素機能エージェント（デーモン）を実現するためのものである。

・TCP/IP ソケットを使った上位モジュールとの通信
・上位モジュールからのコマンドメッセージのパージング
・複数の要求コマンドの並行的な実行管理
・複数エージェント間通信のための共有メモリ管理

下位のモジュールと上位のモジュールとが別のマシン上で動くことを可能にするために、上位のモジュールとは TCP/IP ソケットを通じて通信を行い、メッセージの授受を行うように設計を行った。これによって、

・ネットワークを介してロボットをコントロールし、ロボットとインタラクションするようなアプリケーション
・複数の CPU での実行を必要とするような複雑なインタラクション

を容易に実現することができる。

受け付けるコマンドメッセージは API にあたるものであるが、ロボットの機能の変化にあわせて自由に定義・拡張可能な仕様になっている。現段階でさまざまなロボットのための API を設計し固定することは適切ではなく、こうした仕組みによってさまざまな API を評価しながら、必要な機能を明らかにしてゆくことが望ましいと考えた。

複数の上位モジュールからの複数の要求の実行管理をするために、コマンドスタックを用いて、現在実行中のコマンドを保存し、一定の周期で繰り返し実行することを可能にしている。

さらに、複数の要素機能エージェントが、ロボットの異なる部分（たとえば右半身と左半身、目と耳と腕、など）を受け持つことも想定し、エージェント間の通信を行うための共有メモリを実装した。

このライブラリは、電子技術総合研究所で開発された、事情通ロボットの制御ソフトウェアである 小脳-大脳アーキテクチャ[2]の開発を踏まえて、より汎用的な応用に対応できるように新たに書き直したものである。

・要素機能エージェントの構成

エージェントは、1 つのコマンドスタックと TCP 接続のための通信ポートを持ち、poll システムコールにより、TCP 接続要求、TCP コマンド要求の 2 つを同時に待ち受ける。エージェントは TCP サーバであり、TCP 接続を複数受け付けることができる。コマンド要求はいったんコマンドスタックに蓄えられる。各エージェントは固有の周期時間での周期処理を行う。周期処理の中でコマンドスタック中に実行可能なコマンド要求があればそれを実行し、結果を reply メッセージとして sender に報告し、コマンドスタックから取り除く。（図２）

・エージェント間通信

エージェント間の情報交換は、共有メモリまたは TCP ソケットを通じて行う。共有メモリは、大量で高速な情報伝達のために用いることができる。事象の発生を共有メモリを用いて実現するためには、メインループ内で共有メモリ内を定期的に検査しなければならない。 TCP ソケットは、主にイベント伝達のために用意された通信

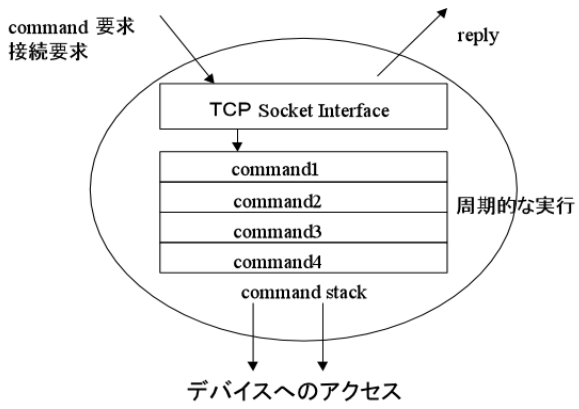

図２　要素機能エージェントの模式図

手段であり、異なるマシンに存在するエージェント間通信、非同期イベントの通信に用いる。
・コマンドメッセージの形式
　コマンドメッセージは (command　sender seq interval arg1 arg2 ...) という形式のテキストである。command　は文字列またはそれに対応する整数である。対応表は、エージェントプログラム内で定義される。定義の方法については後述する。sender は、コマンドを発行したプロセスの名前であり、seq はそのプロセス内でユニークに定義される整数値である。これらの 2 つの値は、コマンドを識別するために用い、主に retract コマンドでコマンド要求を取り下げるときに用いられる。arg1, arg2,...は、各コマンド特有の引数であり、整数、浮動小数または文字列とする。
　interval は、そのコマンドを実行する場合の最小のインターバル時間で単位は msec で指定する。これによって全体のイベント周期とは別にコマンドごとの実行周期を定義することができる。
・イベント報告
　エージェントがコマンド要求によりイベントの監視を続け、条件にあった場合、または、要求された動作処理を終えた瞬間、コマンドが送られてきたのと同一のコネクションを通じ reply メッセージを通知する。コマンド対して必ず 1 つの reply メッセージが生成され、 (reply sender seq result1 result2 ...)という形式のテキストで返される。
・エージェントの作成
　ライブラリを用いた具体的なエージェント（デーモン）の作成は次のステップで行うことができる。

1. agent.h の include
2. コマンドのエントリ
3. 共有メモリの確保
4. task_descriptor の初期化
5. TCP ポートの生成
6. Main Loop のコール

（３）パーソナルロボット統合機能開発環境
要素機能エージェントと通信してコマンドメッセージを送り、エージェントが提供する機能を利用した統合機能を開発するためのベースとなる Java クラスとして、
・ロボット制御用デーモンとの通信を行うためのクラス (RobotComm クラス)
・ロボット制御用デーモンからの応答メッセージを処理するためのクラス（Commander クラス）
の二つを開発した。これらのインスタンスを作成することで下位プログラムモジュールへのコマンドを容易に与え、下位モジュールの機能を統合した統合機能を開発することができる。
a) RobotComm クラス
下位ロボットプログラム用通信オブジェクト
Constructors
RobotComm(String name, int port, String cmd, CallbackObj callback)
によって通信のためのスレッドを生成し、コマンドを送信する。下位ロボットプログラムからの応答は、callback オブジェクトの response()というメソッドを起動する。
Values
CallbackObj　callbackObject
応答時に呼び出されるオブジェクト名
String command
ロボットへ送られるコマンド
Methods
Public void opnePort(String target, int port)
ロボットデーモンへの通信路作成。
Public void closePort()
ロボットデーモンへの通信路を閉じる。
Public void setCommand (String cmd)
ロボットへ送るコマンドを設定する。
Public void setCallbackObject(CallbackObj obj)
応答時に処理を行うオブジェクトの登録。
Public void sendMsg(String cmd)
コマンドをロボットへ送る。
Public Stiring recvMsg()
ロボットからの応答を得る。
b) CallbackObj インターフェース
RobotComm で下位ロボットプログラムの応答を処理するためのインターフェースクラス
Methods
Public void response(String str)
ロボットからの応答を処理する。
c) Commander クラス
下位ロボットプログラムへのコマンドを発行及び応答処理を行うためのオブジェクトクラス。CallbackObj インターフェースを実装している。
Constructors
Commander()
デフォルトのロボットへ接続
Commander(String robotName)
ロボットの名前を指定して接続
Commander(String robotName, int robotPort)
ロボットの名前、接続ポートを指定して接続
Values
String robotName
接続先のロボットの名前
int　robotPort
接続ポートの番号
String command
ロボットへ送るコマンド列
CallbackObj parent
応答を処理するためのオブジェクト名
Methods
Public void setRobotName(String name)
接続ロボットの登録
Public void setRobtPort(int port)
接続ポートの登録
Public void setCommand(String command)
ロボットへのコマンドの登録
Public void setParent(CallbackObj　parent)
応答を処理をディスパッチする
オブジェクトの登録
Public void exec()
登録されたコマンドをロボットへ送信する
Public void exec(String command)
Command で指定しているコマンドを
ロボットへ送信する

５．開発成果の検証と評価

（1）ロボット制御用デーモン

　開発した要素機能ライブラリを用いて、テムザック IV のすべての関節を並行に制御することができるプログラム、ロボット制御用デーモン (robotd) を作成した。上位プログラム群との通信のための TCP ソケットポートとして、7100 をデフォルトとして使用し、以下に示すコマンドを処理することができる。ロボット制御用デーモンへのコマンドは、前述した非同期・イベント駆動型要素機能ライブラリで定義されたコマンドフォーム

(command sender seq interval arg1 arg2 ...)

を用いる。ただし、このコマンドフォームは、ライブラリ中のコマンドパーザを置き換えることで自由に変更することができる。すなわち、汎用的な体系である XML 形式のコマンド体系などに容易に変更することができる。また、robotd は、コマンドの追加・修正をユーザに応じてカスタマイズすることができる。受理可能なコマンドリストは main.c に記述されており、新規な機能を実装するとともに、このファイルを修正・コンパイルすることで新たなコマンドを追加生成することが可能である。

現在使用可能なコマンドは以下のとおりである。

PRINTSTATE
用法：(printState sender seq interval val)
説明：現在の関節角、目標関節角等を表示
val = 0 ：目標関節角の表示
1 ：下限の関節角の表示
2 ：初期値の関節角表示
3 ：上限の関節角表示
4 ：現在の関節角の表示

MOVEHAND
用法：(moveHand sender seq interval arm J1 J2 J3 J4)
説明：腕を指定角だけ移動させる
J1：肩前後、J2：肩左右、J3：肘左右、J4：肘前後
arm = 0 ： 右腕の動作指定
1 ： 左腕の動作指定
2 ： 両腕の動作指定

INITHAND
用法：(initHand sender seq 0)
説明：腕の各関節を初期状態に移動させる。

MEMORIZE
用法：(memorize sender seq 0)
説明：現在の状態を記憶されている状態列に追加する。

PLAY
用法：(play sender seq 0)
説明：記憶した状態列を順に実行する

STOP
用法：(stop sender seq 0)
説明：動作を停止させ、現在、コマンドスタックにある命令をクリアする。

PAUSE
用法：(pause sender seq 0)
説明：動作を一時停止させる。コマンドスタックをクリアしない。

RESUME
用法：(resume sender seq 0)
説明：動作の再開。

ACTION_CLS
用法：(action_cls sender seq 0)
説明:現在記憶している状態列をすべてクリアする。

ACTION
用法：(action sender seq interval actionName)
説明：引数 actionName で指定された状態列を呼び出し、実行する。

NAME
用法：(name sender seq interval actionName)
説明：
現在記憶している状態列に引数 actionName で指定された名前をつける。

SETJOINTS
用法：
(setJoint sender seq interval
rJ1 rJ2 rJ3 rJ4 lJ1 lJ2 lJ3 lJ4)
説明：各関節を指定した角度へ動作させる。
[rl]J[1,2,3,4] は、0 から１００までの値とし、各関節の動作可能範囲の%を与える。

QUERYACTION
用法：(queryAction sender seq 0)
説明：現在、ロボット内に記憶されている状態列の一覧を表示する。

また、汎用ライブラリ部分については、Nomadic 社の移動ロボットベース Nomad 200 上で、モータ駆動やセンサ情報からのイベント抽出などの要素機能を実装するためにも利用し、汎用的な動作を確認している。

（２）機能統合サンプルアプリケーション

上記の robotd と、開発した Java の Commander クラスとを用いて、サンプルプログラムとして、ティーチングプレイバック用操作パネルの実装を行った。このプログラムは、jdk1.2 以上で動作する Java アプリケーションである。

操作パネルは、
・ロボット関節角指定用操作パネル（図３）
・関節角情報表示パネル（図４）
・行動列生成・呼び出し用操作パネル（図５）
から構成されている。

a) ロボット関節指定用操作パネル

この操作パネル内のスライダによってロボットの各関節を指定し、Action ボタンを押すことで、動作コマンドが発行され、ロボットの首、胴、両腕を動かすことができる。また、Memorize ボタンによって現在の関節角の状態を下位ロボットプログラム内に記憶させることができる。

記憶された状態は、動作列として記憶され、後述する行動列生成・呼び出し用操作パネルでプレイバック動作を実行することができる。

b) 関節角情報表示パネル

この操作パネルは、ロボットデーモンの状態をログとして表示することができる。また、パネル上のボタンによって、現在の関節角や目標関節角などの情報を表示することができる。この情報表示パネル上では、他のパネルで実行されたコマンドに対する応答も表示することができる。

c) 行動列生成・呼び出し用操作パネル

この操作パネルは、前述したロボット関節角指定用操作パネルで記憶した行動列を扱うためのパネルである。このパネル上の Play ボタンによってフィードバック実行を行うことができる。また、現在ロボットデーモン内に記憶されている状態列（行動）に名前を付けて記憶（長期記憶）させることができる。また、ShowList ボタンによって以前記憶した行動列の一覧を呼び出し、セレクト領域で指定された行動を Play ボタンを用いて呼び出し・実行させることができる。この機能によって、ロボット内にマクロ行動を記述することが可能になる。

（３）ソフトウェアモジュールの機能確認

開発したモジュールが全体として実際に機能することを確認するために、いくつかの動作を作成登録し、実行できることを確認した。具体的には以下のような簡単な動作を作成した。個々の動作の登録は簡単であり、他にもいろいろなものを作成することができる。

・ばいばい（片手を振ってばいばいする）

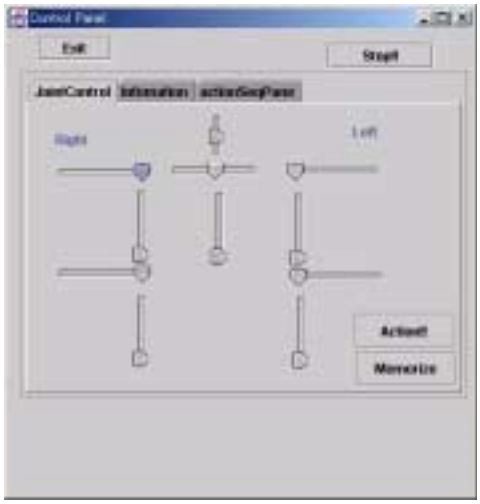

図３　関節角指定用操作パネル

図４　関節角情報表示パネル

図５　行動列生成・呼び出し用パネル

・ポインティング（ある方向を指さす）
・ダンス（双腕を振ってダンスする）

５．まとめと今後の課題

本開発では、パーソナルロボットのためのアプリケーション開発環境を、できるだけロボットに非依存な形で実装した。今回は基本的な機能の開発にとどまったが、提案した基本的な枠組みに沿って、要素機能エージェントの構築を支援するライブラリおよび機能統合エージェントの作成を支援するクラスライブラリを作成し、ティーチングプレイバック操作を実現するアプリケーションの実装を通じて、それらの有効性と動作を確認することができた。

今後の課題としては、今回開発した基本的な枠組みをさらに拡張して、各種のセンサ処理、認識プログラム等とも組み合わせて、パーソナルロボットに期待される幅広いアプリケーションをカバーするために必要とされる要素機能エージェント群および統合機能エージェント群を探索的に実装してゆくことがあげられる。そのためには、要素機能を階層化してゆくとともに、パーソナルロボットのための汎用的な API としてどのような抽象的な機能を用意する必要があるかの検討が必要である。

また、アプリケーション開発環境として使いやすいものにすることおよび、開発の効率を上げるために、実際のロボットの代用となるシミュレータ、統合機能を効率よく記述するためのスクリプト言語などの開発も必要である。

さらに、不確実な情報の多い実環境において、曖昧で複雑な振る舞いをする人間との柔軟なインタラクションを実現するためには、確率的な状況認知にもとづいて、状況依存な統合機能エージェント群を管理する仕組みを組み込んでゆくことが必要であると考えている[3][4]。

６．参加企業及び機関

（財）日本産業技術振興協会
（プロジェクト実施管理組織）

７．参考文献

[1] H.Asoh, et al.: Jijo-2: An office robot that communicates and learns, IEEE Intelligent Systems, vol.16, No.5, pp.46-55, SEPTEMBER/OCTOBER (2001).
[2] T.Matsui, H.Asoh, I.Hara and N.Otsu: An event driven architecture for controlling behaviours of the office conversant mobile robot, Jijo-2, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, pp.3367-3371 (1997).
[3] 本村, 原, 田中, 学習知能ロボットにおける状況依存エージェントの協調, 日本ソフトウェア科学会マルチエージェントと協調計算ワークショップ (1999).
[4] Y.Motomura, I.Hara, K.Itou, H.Asoh, and T. Sato: Task, situation, and user models for personal robots, IJCAI-2001 Workshop Working Notes, Reasoning with Uncertainty in Robotics, pp.51-56 (2001).