National Institute of Advanced Industrial Science and Technology
AIST

RT MIDDLEWARE

RtcHandle

– Python環境から簡単にRTCを扱う方法–

（独）産業技術総合研究所
知能システム研究部門
末廣　尚士

問題の背景

OpenRTM-aistを用いてRTCを作ることはできるが、
- 作られたRTCを用いてロボットシステムを構築し、
- ロボットアプリケーションを実現する、

簡単な方法はまだ提供されていない。

たとえば、あるRTCを使うために、
- C++で別途RTCを作成して、
- その中でコマンドメニューを出して動作試験をする、

などをすることがある。

select menu(1:set param, 2:set destination, 3:move arm) >>

これは正しいやり方だろうか？

RtcHandleとは

- 概要:
  - そこにあるRTCをPython環境から簡単に扱うことができるPythonモジュール

- 特徴:
  - Python環境からのRTCの簡単操作
  - ロボットシステム構築の支援
  - RTCおよびロボットシステムのデバッグツール
  - RTCのプロトタイピング
  - ロボット作業アプリケーションの開発・実行

前提とするシステム

- OpenRTM-aist-0.4.2 (これは相手側の話)
  - OpenRTM-aist-0.4.2の未接続portのdisconnectに関するパッチが必要。
  - Python版、Java版（それぞれ0.4.1)についても基本的には確認済み。
  - rtc_handleを契機にバグが発見されることもある。
- Linux
  - OpenRTM-aist-python-0.4.1
  - Python2.4.4 (python-2.4.4-1.4vl4)
  - とりあえず確認はVine 4.2のみ。おそらくLinuxならそのまま動作する。
- Windows XPでも動作確認済み
  - Python2.5系
    - python-2.5.1.msi 、omniORBpy-3.1.msi 、OpenRTM-aist-Python2.4-0.4.1-RELEASE.msi
  - Python2.4系
    - python-2.4.4.msi 、omniORBpy-2.7.msi 、OpenRTM-aist-Python2.4-0.4.1-RELEASE.msi

基本クラス(1)

- **RtmEnv**
  orbやnaming serviceなど，OpenRTMの環境情報を保持する.
- **NameSpace**
  naming serviceとそこに登録されているcorbaオブジェクト，RTCおよび対応するRtcHandleを保持する.
- **RtcHandle**
  RTCの情報の保持およびその機能へのアクセスを提供する

基本クラス(2)

- **Port**
  RTCのPortに対応するクラス．ポートの種類によって以下の3つのサブクラスがある．このクラスを通してポートのサービスや入出力へのアクセスも直接行うことができる．
  - RtcService：サービスポート
  - RtcInport：入力ポート
  - RtcOutport：出力ポート
- **Connector**
  RTCのPort間の接続情報を管理するConnectorProfileに対応するクラス．接続Portの種類により2つのサブクラスがある．
  - ServiceConnector：サービスポートの接続
  - IOConnector：入出力ポートの接続

使用例(1)

- モジュールの読み込み

```
$ python
...
>>> from rtc_handle import *
```

- OpenRTM環境の構築

```
>>> env = RtmEnv(sys.argv,["localhost:9876"])
```

- RtcHandleの生成

```
>>> env.name_space['localhost:9876'].list_obj()
```

使用例(2)

- RtcHandleの取り出し

```
>>> pa10fk = env.name_space["localhost:9876"].rtc_handles["pa10fk0.rtc"]
>>> frm_ctrl = env.name_space["localhost:9876"].rtc_handles["frm_ctrl0.rtc"]
```

- Connectorの生成、接続

```
>>> con1 = IOConnector([pa10fk.outports["frame"], frm_ctrl.inports["ref_frm"]])
>>> con1.connect()
>>> con1.disconnect()
```

- RTCの活性化、不活性化

```
>>> pa10fk.activate()
>>> pa10fk.deactivate()
```

使用例(3)

- 出力ポートからの直接データ取得

```
>>> data1=pa10fk.outports["frame"].read()
>>> data1.data = [-0.7070..., ...]
```

- 入力ポートへの直接データ送信

```
>>> data2=[-0.7070..., ...]
>>> frm_ctrl.inorts["frame"].write(data2)
```

rtc_handleの利点

- pythonスクリプトでRTC利用環境が簡単に構築できる。
- インタラクティブにRTシステムを試すことができる。
- RTCを利用する「プログラム」が簡単に書ける。

- 以下はpythonとしての利点が大きいが、
  - python版OpenRTMはインストールが簡単。
  - python言語は難しくないし、ロボット制御にはとても便利なインタラクティブ環境を提供してくれている。

National Institute of Advanced Industrial Science and Technology
AIST

デモ

RT MIDDLEWARE

- 環境構築スクリプト実行
  - si2008.py
- RTCに関するインタラクティブな情報収集
  - env.con_list[1].disconnect()
  - a=vel_7dof.outports['th2'].read()
  - pa10disp.inports['th'].write(a.data)
  - env.con_list[1].connect()
- RTシステムのプログラム
  - go_to(move, pb)
  - demo(move, [pa, pb, pc])
- 終了

アーム動作関数のソース

```
pa = FRAME(xyzabc=[600.0, 0.0, 220.0, 0.0, pi, pi/4])
pb = FRAME(xyzabc=[600.0, 0.0, 500.0, 0.0, pi, pi/4])
pc = FRAME(xyzabc=[600.0, 300.0, 300.0, 0.0, pi, pi/4])
#
def demo(mv_rtc, pos_list) :
    for pos in pos_list :
      go_to(mv_rtc, pos)
#
def go_to(mv_rtc, pos) :
   target = c_frame(pos.mat.val,pos.vec.val)
   mv_rtc.services['ComMv'].provided['com_move'].ref.move_to(target)
```

アーム動作関数2のソース

```
def go_to2(ctrl_rtc, pos) :
    frm_data=range(12)
    for i in range(3) :
      for j in range(3) :
        frm_data[3*i+j]=pos.mat.val[i][j]
      frm_data[9+i]=pos.vec.val[i]
    ctrl_rtc.inports['ref_frm'].write(frm_data)
    time.sleep(0.5)
    state = 0
    print state
    while not state :
      tmp = ctrl_rtc.outports['state'].read().data
      if (tmp == 0 or tmp == 1) :
          if not state == tmp :
              state = tmp
              print state
      time.sleep(0.01)
```

National Institute of Advanced Industrial Science and Technology
AIST

まとめ

RT MIDDLEWARE

- そこにあるRTCをPython環境から簡単に扱うことができるPythonモジュールについて説明した。
- まだ完成形ではない。問題点もいろいろ。
  - 本当に使いやすい構造か？
  - 実RTCとの整合性を考えたとき、こういう静的なデータ構造を使うのが良いのか？
- 基本的にはRTCのプロキシオブジェクトなので、マッピングを決めてC++、Javaでも使えるようになれば便利になると思う。

- 多くの人に使ってもらうことでより良いものにしたい。
  - URL: http://staff.aist.go.jp/t.suehiro/rtm/
- それ以上に、このシステムの上にロボットアプリケーションのフレームワークを構築できると嬉しい。