No. 04-4

ROBOMEC'04

2004 JSME Conference on Robotics and Mechatronics

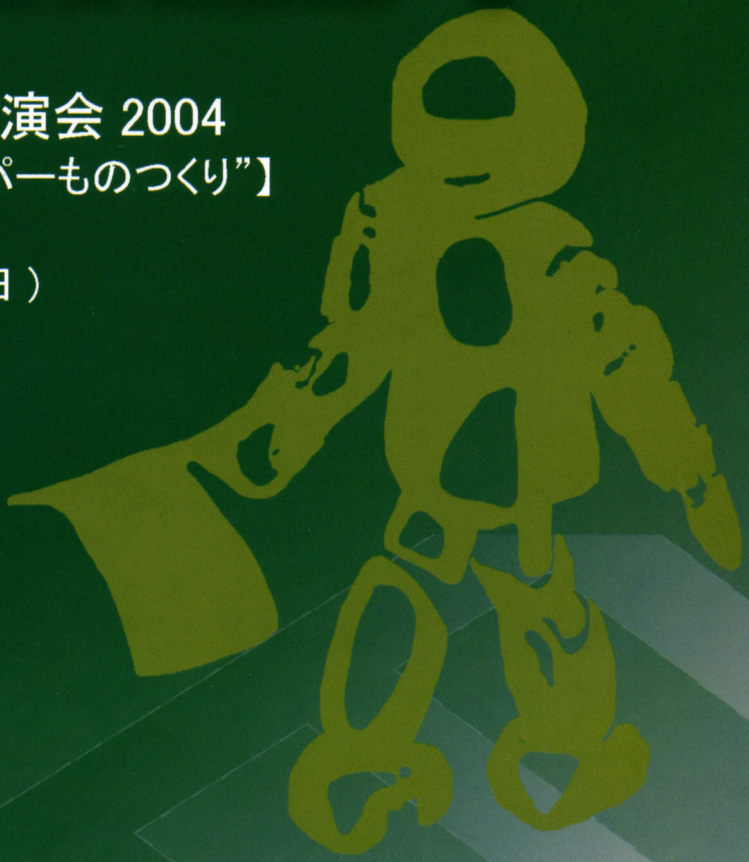
ロボティクス・メカトロニクス講演会 2004

【安心・安全な環境のための“スーパーものつくり”】

2004 年 6 月 18 日（金）～ 20 日（日）

名城大学（名古屋市天白区）

講演概要集



主催：社団法人 日本機械学会
ロボティクス・メカトロニクス部門

RT コンポーネントによるマニピュレータ制御システム構築
ー RT ミドルウェアの基本機能に関する研究開発（その５）ー

Development of Manipulator Control System with RT Components
- R & D of RT Middleware Fundamental Functions (Part 5) -

○正　北垣高成　末廣尚士　神徳徹雄　尹祐根　安藤慶昭　(産総研)

Intelligent Systems Institute,
National Institute of Advanced Industrial Science and Technology (AIST)
○ K. KITAGAKI, T. SUEHIRO, T. KOTOKU, W.K.YOON, and N. ANDO
mailto: k.kitagaki@aist.go.jp

Abstract: *Modularization of robot components is a key technology for the efficient construction of robot systems. Middleware for the robot system is necessary to establish the modularization. We have been developing such kind of middleware as RT middleware. This paper introduces an application for a manipulator control system with our RT software component framework.*

Key words: *RT (Robot Technology) middleware, RT component, Robot system architecture, CORBA*

1　はじめに

ロボットシステムを構成する要素をモジュール化し、部品として自由に組み合わせることで新たな機能を持つロボットシステムを、容易に構築可能とするシステム構築手法が確立されれば、ロボットの実用化、製品化の効率向上に大きく寄与すると考えられる。筆者らは、これまで開発してきたロボットシステムの構築において得られた知見を元に、そのようなモジュール化を実現するためのソフトウェア基盤技術としてロボット用ミドルウェア（RT ミドルウェア）の開発を進めている [1]-[5]。

RT ミドルウェアは、既存のロボット技術を部品化し再利用を促進するための
・コンポーネントフレームワーク、
・標準的に再利用されるソフトウェア部品群、
・ライブラリ群、
・標準サービス群
などから構成される [5]。

本稿では、これまで提案してきたコンポーネントフレームワークに基づき構築された力制御システムを紹介する。

2　RT ソフトウェアコンポーネント

分散オブジェクトは明示的なインタフェースを持つサーバオブジェクトであり、パッシブな動作の側面のみをサポートする。これに対し、ロボットの要素の多くはサーボなど、それ自身のタスクを持つだけでなく、必要なデータを自ら収集したり、またイベントの発生などの通知を行ったりなどの固有の処理 (アクティビティ) を持っている。その固有の処理の中では、自身のインタフェースで要求されるデータを随時書き換えたり、他の RT モジュールへクライアントとして要求を出したり、などが行われることになる。

ロボットの要素を部品として再利用できる形でモジュール化するためには、上記の機能をひとまとまりとした単位で考える必要がある。これをオブジェクトと区別して「RT ソフトウェアコンポーネント」(以降、簡単のため RT コンポーネントもしくはコンポーネントと記す) と呼びロボットの要素のモジュール化の単位とすることを筆者らは提案している。

3　RT コンポーネントの構成

筆者らのフレームワークでは、コンポーネント自身がアクティブに動作し、相互に通信しながらシステムの機能を実現するタイプのモジュール化を実現する。１つの RT コンポーネントは、以下の 3 種類の CORBA オブジェクトから構成される (Fig.1)。

(1) RtComponent オブジェクト

RT コンポーネントの本体。コンポーネントのコマンドインタフェースはここで定義される。RtComponent では、様々なコンポーネントに共通なものを定義する。個々のカテゴリーのコンポーネントは RtComponent を継承して作られる。ただし、複雑な多重継承をさけるために実装の方は RtComponent の実装を継承せずに個別にする方が良い。実装部はアクティビティ部を別スレッドとして持つ。入力ポート、出力ポートは、各々 InPort, OutPort の“実装”を内部で生成することで実現する。それらもまた別スレッドで実行される。

(2) InPort オブジェクト

RT コンポーネントの入力ポート。IDL で公開された put 操作により外部からデータを受け取る。受け取った後の処理はコンポーネントごとに異なるが数種類のパターンに分けられると考えている。これらの違いは、それぞれの“実装”を用意することで吸収する。

(3) OutPort オブジェクト

RT コンポーネントの出力ポート。IDL で公開された get 操作により外部からの pull 型アクセスを受け付ける。また、InPort のオブジェクトリファレンスを引数にして subscribe することにより、指定された InPort に対してアクティビティ部でのデータ生成に応じて push 型のデータ出力を行う。

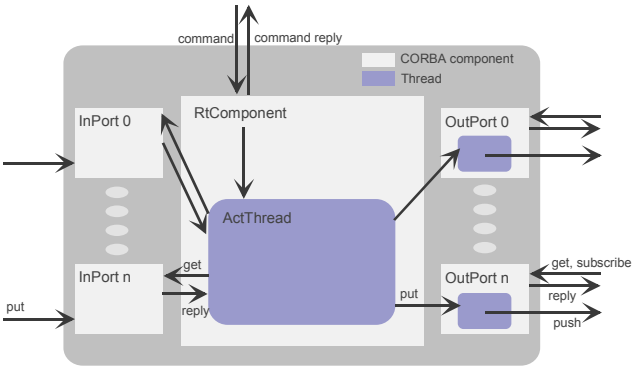


Fig.1　Archtecture of RT component

```
1: module RTM {
2: // Data type definition
3:   struct Time {
4:     unsigned long sec;     // sec
5:     unsigned long nsec;    // nano sec
6:   };
7:   typedef float Float6[6];
8:   struct TimedFloat6 {
9:     Time tm;
10:    Float6 val;
11:  };
12:  ・・・・・
13:// Interface definition
14:  enum ReqType {once, repeated};
15:
16:  interface InPort {
17:    void put(in any value);
18:    readonly attribute string name;
19:    readonly attribute CORBA::TypeCode port_type;
20:  };
21:
22:  interface OutPort{
23:    any get();
24:    boolean subscribe(in ReqType r_type,
25:                      in InPort i_port);
26:    boolean unsubscribe(in ReqType r_type,
27:                        in InPort i_port);
28:    readonly attribute string name;
29:    readonly attribute CORBA::TypeCode port_type;
30:  };
31:
32:  typedef sequence<InPort> InPortList;
33:  typedef sequence<OutPort> OutPortList;
34:
35:  interface RtComponent {
36:    void on();
37:    void off();
38:    InPortList in_ports();
39:    OutPortList out_ports();
40:  };
41:};
```

Fig.2　IDL for RtComponent

これらの IDL 定義を Fig.2 に示す。データ型の定義に続き、上記 3 種の CORBA オブジェクトのインタフェースが定義されている。入出力ポートは、any 型のデータを用いることで IDL 定義を各々 1 つですむようにした。通信や処理の効率を考えると any 型でない方が良いのだが、その場合、ポートのデータ型に応じて異なる IDL 定義が必要となる。

4　力サーボシステム

RT コンポーネントを用いたマニピュレータ制御システムのひとつとして、シンプルな力制御を実現するサーボシステムを構築した。基本要素コンポーネントは、目標値生成、センサ、コントローラ、アクチュエータの 4 つである。

4.1　基本要素コンポーネントの機能と構成

Fig.3 に本システムにおける基本要素コンポーネント間のデータの流れを示す。アクチュエータコンポーネントは慣性空間の速度制御指令により動作するマニピュレータであり、入力は速度指令値である。コントローラコンポーネントは単純な比例制御アルゴリズムの実装されたコントローラであり、目標値生成コンポーネントからの目標値とセンサコンポーネントからのセンサデータの偏差にゲインをかけ速度指令値を出力する。センサコンポーネントはマニピュレータの手先に装備された力覚センサで観測される力覚データを出力する。目標値生成コンポーネントは力覚センサを利用したジョイスティックであり、オペレータが加えた力を目標値として出力する。なお、本システムにおける入出力データ型は TimedFloat6(Fig.2 のリスト上部参照) である。この構造体は、タイムスタンプと float 6 個で構成される。

4.2　サーボコンポーネント

それぞれの OutPort に対し subscribe オペレーションを適切に発行し、それぞれのコンポーネントを Port を介して接続することで、システムを構成することがで

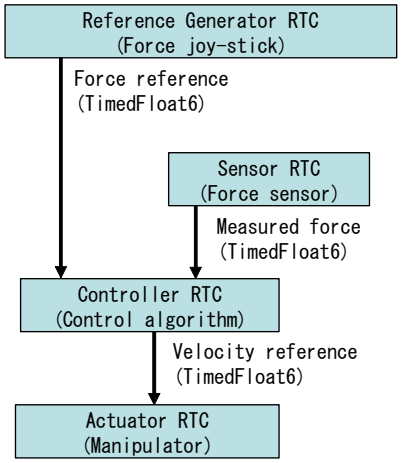


Fig.3　Structure of a manipulator servoing system

きる。RT コンポーネントブラウザ [5] などを用いれば手動で接続が可能である。しかし、本システムのようなサーボ系を構築する場合には、利用するコンポーネントはセンサ、コントローラ、アクチュエータであり、接続状態も決まっており、ひとつのパターンと考えられる。あらかじめ、RT コンポーネントとして準備しておくことは有用である。サーボコンポーネントはそのような管理機能を持つ複合コンポーネントである。Fig.4 にサーボコンポーネント機能の模式図を示す。本コンポーネントは外部から、管理下のセンサ、コントローラ、アクチュエータコンポーネントを隠し、サーボへの目標値を入力するための Inport を持つ。

5　コンポーネントの実装

それぞれのコンポーネントの実装について示す。なお、それぞれのプログラムリストは、RT コンポーネントとしての共通的な部分を除いた固有の実行部分を抜粋したものである。また、枠で囲まれた部分は実際の機能の記述部分である。

5.1　目標値生成コンポーネント

目標値は様々な手法で生成されるものであるが、本システムでは、力覚センサを用いたジョイスティックにより生成した。プログラムは次に述べるセンサコンポーネントとほぼ同様である。

5.2　センサコンポーネント (Fig.5)

2 行目で力覚センサからのデータを取得し、3,4 行目で OutPort に出力する構造体 (TimedFloat6 out_val) に力覚データを格納する。5,6 行目でタイムスタンプを同様に格納する。7 行目で any 型に変換し、8 行目で OutPort に put する。

5.3　コントローラコンポーネント (Fig.6)

2 ～ 4 行目で、センサコンポーネントの OutPort からのデータを取得し、any 型から TimedFloat6 に変換する。5 ～ 7 行目で、同様に目標値生成コンポーネントの OutPort からのデータを TimedFloat6 に変換する。

9 ～ 12 行目は、シンプルな比例制御 (偏差にゲインをかける) の実装部分である。13,14 行目でタイムスタンプの格納、15 行目で any 型に変換し、16 行目で OutPort に put する。

5.4　アクチュエータコンポーネント (Fig.7)

3 ～ 5 行目でコントローラからのデータを取得し配列 outVel に代入する。6 ～ 11 行目は目標速度のリミッタ部分である。12 行目でマニピュレータに速度指令を発行する。

5.5　サーボコンポーネント

サーボコンポーネントは、センサ、コントローラ、アクチュエータの各コンポーネントのオブジェクトリファレンスから、それぞれの Port のオブジェクトリファレンスを獲得し、適切に subscribe する (Fig.4)。

本コンポーネントは、オペレーション on を受けると各コンポーネントの Port を接続し、次にそれぞれのコンポーネントにオペレーション on を発行する。Port 接続のプログラムリストを Fig.8 に示す。

1 行目でセンサコンポーネントの OutPort のオブジェ

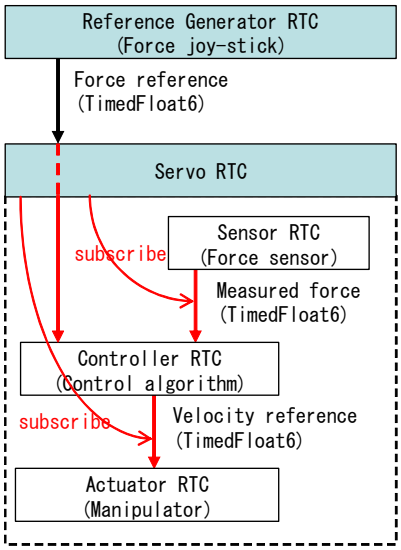


Fig.4　Function of the servo RTC

```
1: // Obtain sensor data
2: nittaFts->forceAtFsInFs(frc);
3: for (int i = 0; i < 6; i++)
4:     parent->out_val.val[i] = frc[i];
5: get_time(&parent->out_val.tm.sec,
6:          &parent->out_val.tm.nsec);
7: parent->any_buf <<= parent->out_val;
8: parent->o_port_i->put(parent->any_buf.in());
```

Fig.5　Main part of the sensor component

```
1: RTM::TimedFloat6 * tmp;
2: parent->any_buf = parent->sen_port_i->get();
3: parent->any_buf >>= tmp;
4: parent-> sen_val.val = tmp->val;
5: parent->any_buf = parent->ref_port_i->get();
6: parent->any_buf >>= tmp;
7: parent-> ref_val.val = tmp->val;
8: // Control algorithm
9: for (int i=0; i < 6; i++)
10:  parent->out_val.val[i] =
11:      gain[i] * (parent->ref_val.val[i]
12:                 + parent->sen_val.val[i]);
13:get_time(&parent->out_val.tm.sec,
14:         &parent->out_val.tm.nsec);
15:parent->any_buf <<= parent->out_val;
16:parent->o_port_i->put(parent->any_buf.in());
```

Fig.6　Main part of the controller component

```
1: RTM::TimedFloat6 * tmp;
2: float outVel[6];
3: parent->in_buf = parent->cnt_port_i->get();
4: parent->in_buf >>= tmp;
5: outVel = tmp->val;
6: for (int i = 0; i < 6; i++) { // Limitter
7:   if (outVel[i] > limitter[i])
8:     outVel[i] = limitter[i];
9:   else if (outVel[i] < -limitter[i])
10:    outVel[i] = -limitter[i];
11:}
12:m_palib->orderVelocity(outVel); //Control Arm
```

Fig.7　Main part of the actuator component

クトリファレンスのリストを取得し、2 行目で 0 番目の OutPort のオブジェクトリファレンスを取得する。3,4 行目でアクチュエータコンポーネントの 0 番目の InPort のオブジェクトリファレンスを取得、7 ～ 11 行目でコントローラコンポーネントの 0,1 番目 (センサ入力、目標値入力用) の InPort ならびに 0 番目の OutPort のオブジェクトリファレンスを取得する。13行目では、自分の 0 番目の InPort にコントローラの 1 番目の InPort のオブジェクトリファレンスを代入する。これは、外部からサーボコンポーネントの 0 番目の InPort にアクセスするとコントローラコンポーネントの 1 番目 (目標値入力用) の InPort に直接アクセスできることを意味している。15 行目で、センサコンポーネントの OutPort にコントローラの InPort のオブジェクトリファレンスを渡し subscribe する。16 行目では、同様に、コントローラコンポーネントの OutPort にアクチュエータの InPort のオブジェクトリファレンスを渡し subscribe する。

6 実行

一般的な手順として、あらかじめ必要なコンポーネントを起動した上で、各 Port を接続し、各コンポーネントにオペレーションonを発行することで個々の機能が実行され、システム全体が機能する。なお、コンポーネント機能の動作中に Port 接続を変更することは禁止されていないため、コンポーネント開発時にはその点を十分考慮する必要がある。

本システムでは、まず、センサ、コントローラ、アクチュエータ、目標値生成の各基本要素コンポーネント、およびサーボコンポーネントを起動する。この時点では、コンポーネントの Port は接続されていない。次に、コンポーネントエディタなどで、目標値生成コンポーネントの OutPort をサーボコンポーネントの InPort に接続する。最後に、サーボコンポーネントと目標値生成コンポーネントオペレーションonを発行することで、システム全体が起動する。

本システムにより、オペレータの指令値に従ってマニピュレータを力制御できることを確認した。なお、マニピュレータとして利用した PA10 の入力受付周期

```
1: RTM::OutPortList_var opl = m_sen->out_ports();
2: RTM::OutPort_var op_sen = opl[0];
3:
4: RTM::InPortList_var ipl = m_mot->in_ports();
5: RTM::InPort_var ip_mot = ipl[0];
6:
7: ipl = m_cnt -> in_ports();
8: RTM::InPort_var ip_cnt0 = ipl[0];
9: RTM::InPort_var ip_cnt1 = ipl[1];
10:opl =  m_cnt -> out_ports();
11:RTM::OutPort_var op_cnt = opl[0];
12:
13:ip_list[0]=ip_cnt1;
14:
15:op_sen->subscribe(RTM::REPEATED, ip_cnt0.in());
16:op_cnt->subscribe(RTM::REPEATED, ip_mot.in());
```

Fig.8　Connection part of servo component

２ [ms] であるため、それぞれのコンポーネントも同様に２ [ms] 周期で動作させた。リアルタイム制御は ARTLinux により実現した。

7 おわりに

RT コンポーネントによるマニピュレータ制御システムの構築例を、力サーボシステムにより示した。

システム要素のコンポーネント化により、必要に応じて新たなコンポーネントを開発し接続を変更することで、新たな機能を持つシステムを容易に構築することが可能となる。例えば、遠隔マスタコンポーネントを開発し目標値生成コンポーネントとして接続するだけで、新たな遠隔マスタでマニピュレータを制御することが可能である。また、別の制御アルゴリズムを実装したコンポーネントを開発し、コントローラコンポーネントとして接続することで新たな制御系を実現することが可能となる。

以下に、今後の課題を示す。

・モジュールのインタフェース仕様

通信データの整合性がとられていなければモジュールの接続は容易ではなくなるため、モジュール化のメリットが失われる。したがって、データ型および属性についてのある程度の仕様を定める必要がある。ただし、あまりに厳格な仕様は普及を妨げることになるため、その点を留意する必要がある。

・リアルタイム性の記述

マニピュレータ制御においてリアルタイム性は必要不可欠である。ひとつのモジュール内における周期的なリアルタイム性だけではなく、複合モジュール化における周期的なリアルタイム性や割り込み応答性の記述方法が必要である。

・マニピュレータ動作スクリプトの記述

構築されたシステムを用いて、マニピュレータを動作させるためのスクリプトの記述についてのフレームワークが必要である。

なお、本研究は新エネルギー・産業技術総合開発機構（NEDO）の「ロボットの開発基盤となるソフトウェア上の基盤整備」事業の研究成果である。

[参考文献]

[1] 北垣、末廣、神徳、平井、谷江、RT ミドルウェア技術基盤の研究開発について , 第８回ロボティクスシンポジア、pp.487-492, 2003.

[2] 末廣、北垣、神徳、尹、安藤、RT 要素のモジュール化に関する検討、第 15 回ロボット学会学術講演会、1F27, 2003.

[3] 末廣、北垣、神徳、尹、安藤、RT コンポーネントの実装例、第 15 回ロボット学会学術講演会、1F28, 2003.

[4] 神徳、北垣、安藤、尹、末廣、RT ミドルウェアのソフトウェア開発支援機能の検討、第９回ロボティクス・シンポジア、pp.282-287, 2004.

[5] 安藤、末廣、北垣、神徳、尹、RT 要素のモジュール化および RT コンポーネントの実装、第９回ロボティクス・シンポジア、pp.288-293, 2004.