The 25th Annual Conference of
the Robotics Society of Japan

RSJ

第25回
日本ロボット学会
学術講演会

講演概要集

会期 ▶ 2007年9月13～15日
会場 ▶ 千葉工業大学津田沼キャンパス

主催 ▶ (社)日本ロボット学会

OpenRTM-aist-0.4.0によるヒューマノイドロボット HRP2のRTコンポーネント開発

○清水 昌幸　安藤 慶昭　尹 祐根　音田 弘　神徳 徹雄（産総研）

Development of RT-Components on Humanoid Robot HRP2 Using OpenRTM-aist-0.4.0

*Masayuki Shimizu, Noriaki Ando, Woo-Keun Yoon, Hiromu Onda, and Tetsuo Kotoku (AIST)

Abstract—This article presents how to develop RT-Components on a humanoid robot HRP2 using OpenRTM-aist-0.4.0. Key points on porting OpenRTM-aist-0.4.0 to HRP2's control system as well as programming RT-Components on the system are described. As an example, an RT-Component for bilateral teleoperation is developed.

Key Words: RT-Middleware, RT-Component, OpenRTM-aist, HRP2, Teleoperation.

1. 緒言

ロボット技術のモジュールベースの開発およびそれらの統合を容易に実現するためのソフトウェア共通基盤として，RTミドルウェアの研究開発が行われている[1][2]．RTミドルウェアでは，ロボットプログラムをRTコンポーネントという形でモジュール化することにより，プログラムの再利用性の向上や相互利用の促進を目指している．しかしながら，既存のロボットシステム上で，どのようにRTコンポーネントを作成すればよいかについては，それぞれのシステムに依存した部分もあるため，一概には言えない点も多い．様々なシステム上でのRTコンポーネントの開発を促進するためには，それぞれのシステムに対するRTコンポーネントの作成ノウハウを集約・蓄積し，それを広く公開する必要があると思われる．

本稿では，ヒューマノイドロボットHRP2[3]の制御プログラムをRTコンポーネント化する手法の一例を紹介する．以下では，RTミドルウェアとして，OpenRTM-aist-0.4.0[1]を利用し，HRP2の制御システム上でのRTコンポーネントの作成方法を具体的に説明する．

2. ヒューマノイドロボット HRP2の制御システム

本稿で利用するHRP2の制御システムは，ゼネラルロボティックス株式会社が提供しているもの[4]である．このシステムでは，CORBAを用いることにより，HRP2の各機能が分散オブジェクトとして実装されている．

このシステム上で，ユーザがHRP2の制御を行うためには，プラグインというモジュール形式で制御プログラムを実装する必要がある．各プラグインは，プラグインマネージャによって管理されており，現在のシステムでは，5ms毎に周期的に実行されるようになっている．なお，周期実行の実時間性は，ARTLinuxを制御OSとして採用することにより保証している．ユーザは，プラグイン内で周期的に実行される部分に自分の制御プログラムを挿入することにより，ロボットを制御することが可能となっている．そのように作成したユーザプラグインをプラグインマネージャにロードすることにより，ロードされた順番にプラグインが実行されるようになっている．

3. OpenRTM-aist-0.4.0のHRP2への移植

OpenRTM-aistは，RTミドルウェアの実装の一つであり，現在，バージョン 0.4.0が公開されている．OpenRTM-aist-0.4.0の詳細は，文献[1]で述べられているので，ここではその説明は省き，OpenRTM-aist-0.4.0をHRP2へ移植する方法を述べる．

3.1 CORBAの共有

OpenRTM-aistでは，ネットワーク上に分散したRTコンポーネントを組み合わせて利用するためにCORBAを用いている．また，先に説明したように，HRP2も様々な機能を相互利用するためにCORBAを用いている．したがって，OpenRTM-aistをHRP2のシステム上で利用するためには，HRP2のシステムとCORBAを共有する必要がある．

現在のところ，OpenRTM-aistの対応CORBA製品はomniORBのみであり，その他のCORBAを用いることはできない．現在，ゼネラルロボティックス社が提供しているHRP2のシステムは，CORBA製品としてIONA社のOrbix/Eを採用したものとomniORBを採用したものがあるので，OpenRTM-aistとCORBAを合わせるために，omniORBを採用したシステムを用いることにした．これにより，HRP2のシステムとOpenRTM-aistでCORBAを共有することができるようになった．

HRP2のシステムでは，ORB関連の起動および終了処理は，プラグインマネージャが行っている．それゆえ，OpenRTM-aistをHRP2のシステムで利用する場合，OpenRTM-aistはORB関連の処理をする必要がない．ところが，OpenRTM-aist-0.4.0では，デフォルトでORBの起動および終了処理をするように実装されており，プラグインマネージャのORBの処理と干渉してしまう問題が生じる．そこで，OpenRTM-aistのORB関連の起動および終了処理部分を改造し，それらの処理を行わないように変更した．この変更によって，HRP2のシステムとOpenRTM-aistとでCOBRAを問題なく共有することができるようになった．

3.2 ARTLinuxによる実時間処理

OpenRTM-aist-0.4.0は，デフォルトでは，RTコンポーネントの実行周期やデータポート間のデータの送受信周期を実時間で制御することができない．しかしながら，RTコンポーネントが外部から得た位置や力のデー

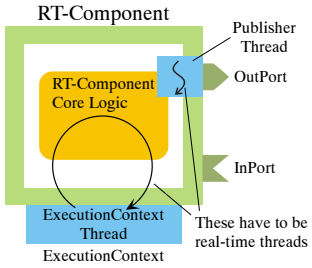

Fig.1 RT-Component architecture.

タを基に HRP2 を制御する場合，そのデータの処理や送受信を実時間で行う必要がある．OpenRTM-aist では，RT コンポーネントの実行周期は，実行コンテキストが管理しているため，RT コンポーネントの周期実行を実時間で行うためには，実行コンテキスト自体を実時間化する必要がある．また，OpenRTM-aist-0.4.0 では，データポート間の送受信のタイミングを制御する仕組みとして，三つのタイプが選択できるようになっている．これらの内，周期的にデータの送受信を実行するためのタイプとして，“Periodic” というものが用意されている．“Periodic” タイプの送受信の場合，**Fig. 1** に示すように，そのタイミングは Publisher というもので管理されているため，ポート間のデータ送受信処理を実時間化するためには，Publisher を実時間化する必要がある．

HRP2 では，その制御実行の実時間性を保証するために，実時間 OS である ARTLinux を用いている．ARTLinux では，システムのハードウェアタイマに同期させて処理を実行するための仕組みが提供されており，その機能をユーザ空間のスレッド単位で用いることができる．RT コンポーネントの実行コンテキストや”Periodic” タイプの Publisher は，それぞれ独立したスレッドとして実装されているため，ARTLinux の同期機能を用いることにより，それらのスレッド処理を実時間で行うことが可能となる．

現在の OpenRTM-aist-0.4.0 では，実行コンテキストは外部のものと動的に差し替えできるような仕組みが提供されているが，Publisher は外部から変更できないようになっている．そこで，OpenRTM-aist-0.4.0 の実行コンテキストと Publisher のスレッドに ARTLinux の API を直接追加し，それぞれのスレッドがデフォルトで実時間化されるように改造した．これにより，データポートのデータ送受信処理やコンポーネントの処理が実時間で行えるようになった．

4. HRP2 制御用 RT コンポーネント作成

4·1 アーキテクチャ

HRP2 の制御システムでは，ユーザのプログラムは，プラグイン形式で実装しなければならない．各プラグインは，その生成時に実行されるコンストラクタ，制御開始時に実行される setup()，制御中に周期的に実行される control()，制御終了時に実行される cleanup()，プラグインを破棄するときに実行されるデストラクタ，の主に五つの関数から構成されている．ユーザはそれぞれの関数を実装することにより，ロボットを制御することができるようになっている．

OpenRTM-aist-0.4.0 では，RT コンポーネントはマネージャ（RT コンポーネントマネージャ，以下 RTC マネージャと呼ぶ）によって管理されている．RT コンポーネントの生成や破棄などの処理は，このマネージャが担当しているため，RT コンポーネントを扱うためには，RTC マネージャが必須である．したがって，RTC マネージャ関連の処理を行うプラグイン（以下 RTC マネージャプラグインと呼ぶ）と RT コンポーネントの処理を行うプラグイン（以下 RTC プラグインと呼ぶ）を実装する必要がある．RTC マネージャプラグインは，どの RTC プラグインよりも先に生成されなければならないため，**Fig. 2** に示すように，一番初めにプラグインマネージャにロードし，その後，RTC プラグインをロードするという順番になる．以下では，RTC マネージャプラグインおよび RTC プラグインの実装方法を説明する．

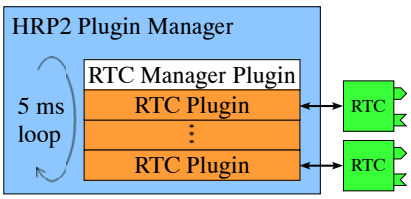

Fig.2 Controller architecture.

4·2 RTC マネージャプラグイン

RTC マネージャプラグインは，どの RTC プラグインよりも先に RTC マネージャを生成・起動し，RTC マネージャが RT コンポーネントの管理をできる状態にしなければならない．そのため，RTC マネージャプラグインのコンストラクタで RTC マネージャを生成・起動する必要がある．OpenRTM-aist-0.4.0 では，コンストラクタで以下のようなコードを実行することにより，RTC マネージャの生成・起動が完了する．

```
RTC::Manager* rtc_mgr;
rtc_mgr = RTC::Manager::init(0,NULL);
rtc_mgr->activateManager();
```

コンストラクタで RTC マネージャを起動すれば，制御中に行うべき処理は何もないため，RTC マネージャプラグインの setup()，control()，cleanup() は必要ない．

最後に，全ての RTC プラグインの実行を終了するときは，RTC マネージャも終了させなければならない．この場合，RTC マネージャは，全ての RT コンポーネントを終了させた後で，一番最後に終了しなければならないため，RTC マネージャプラグインでは，デストラクタで RTC マネージャの終了処理をする必要がある．OpenRTM-aist-0.4.0 では，デストラクタで以下のコードを実行することにより，RTC マネージャが終了する．

```
rtc_mgr->shutdown();
```

先に述べたように，デフォルトの OpenRTM-aist-0.4.0 では，shutdown() で ORB も終了させるため，ORB を終了させないように shutdown() を変更する必要がある．

4·3 RTC プラグイン

RTC プラグインでは，RT コンポーネントを生成し，ポートを介して外部とデータをやりとりしたり，サービスを提供・利用したりする．RT コンポーネント本体は，プラグインとは別に用意する必要があり，rtc-template[2]

などのツールを用いると容易に作成することができるため，ここではその方法は省略する．

いま，rtc-template で生成された Sample という名前の RT コンポーネントがあると仮定し，これを RTC プラグインから生成する方法を初めに説明する．Sample コンポーネントを生成するためには，プラグイン内で以下のようなコードを実行すればよい．

```
Sample* rtc;
RTC::Properties profile(sample_spec);
RTC::Manager::instance().registerFactory(
        profile,
        RTC::Create<Sample>,
        RTC::Delete<Sample>);
rtc = (Sample *)
  RTC::Manager::instance().createComponent(
        "Sample");
```

上記において，createComponent() を実行すると，プラグインのスレッドから実行コンテキストのスレッドが起動し，そのスレッド上でコンポーネントが動作することになる．また，createComponent() の戻り値 rtc が，Sample コンポーネントクラスへのポインタとなるため，Sample クラスのパブリックなメンバ変数や関数へは，このポインタを介してアクセスすることができる．

次に，コンポーネントに登録されたポートを介して，外部とデータやサービスの授受を行う方法を説明する．RT コンポーネントが実行されるスレッドは，HRP2 のプラグインが実行されるスレッドとは別のものであるため，プラグインから直接ポートへのアクセスはできない．そこで，ポートへ直接または間接的にアクセスするためのメソッドをコンポーネントに付加する．例えば，TimedLong 型のデータをバッファ m_in に受け取る m_inIn という名前のデータ入力ポートから直接的にデータを取得するためには，コンポーネントのパブリックな関数として，以下のようなものを追加する．

```
public:
  void Sample::get_data(long *data){
     m_inIn.read();
     *data = m_in.data;
  }
```

プラグインでは，この関数を呼び出すことにより，データ入力ポートからデータを取得することができる．

データ出力ポートやサービスポートの場合も同様に，ポートに直接的にアクセスするメソッド，またはポートによって得られるデータやサービスに間接的にアクセスするメソッドを用意することによって，プラグインからポートを介してデータやサービスの授受ができる．

最後に，コンポーネントを終了させる方法を説明する．コンポーネントをプラグインから終了させるためには，プラグイン内で以下のコードを実行すればよい．

```
rtc->exit();
```

5. 実装例

5·1 遠隔操作用 RT コンポーネント

ここでは，上述の手法を用いた RT コンポーネントの実装例として，HRP2 のアームをスレーブアームとする遠隔操作用 RT コンポーネントを紹介する．以下では，文献 [5] に示されているバイラテラル遠隔操作システムのスレーブ側の機能を実現する RT コンポーネントの作成方法を示す．

5·1.1 機能

初めに，作成する RT コンポーネントの機能を設計する．本稿におけるバイラテラル遠隔操作システムでは，以下の制御則に基づきマスタおよびスレーブアームを制御する．

$$\boldsymbol{V} = \boldsymbol{A}\left(\boldsymbol{F}_m + \boldsymbol{F}_s\right) \tag{1}$$

ここで，$\boldsymbol{V} \in \Re^6$ はマスタおよびスレーブアームの目標手先速度，$\boldsymbol{F}_m \in \Re^6$ はマスタアームの手先力，$\boldsymbol{F}_s \in \Re^6$ はスレーブアームの手先力，$\boldsymbol{A} \in \Re^{6\times 6}$ は正定アドミタンス行列である．現行の HRP2 の制御システムでは，下位レベルのコントローラとして位置制御コントローラが採用されているため，式 (1) で示される制御則は，実際には，以下のような制御則に変換して実現される．

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i + \boldsymbol{A}\left(\boldsymbol{F}_m + \boldsymbol{F}_s\right)\Delta t \tag{2}$$

ここで，Δt は制御周期，$\boldsymbol{X}_i \in \Re^6$ および $\boldsymbol{X}_{i+1} \in \Re^6$ は現時刻および 1 制御周期後の時刻における手先位置・姿勢である．

以上より，バイラテラル遠隔操作システムを構築するためには，マスタおよびスレーブアームの手先力を入力とし，1 制御周期後の目標手先位置・姿勢を出力とするコントローラと，その目標手先位置・姿勢を実現する位置制御コントローラが必要であることが分かる．本稿では，式 (2) の制御則に基づき目標位置・姿勢を計算するコントローラが外部の RT コンポーネントとして別に与えられるとし，スレーブアームである HRP2 の RT コンポーネントのみを設計する．この場合，HRP2 の RT コンポーネントは，以下の機能が必要となることがわかる．

- 目標手先位置・姿勢を外部から取得する．
- 得られた手先位置・姿勢を位置制御コントローラで実現する．
- アームの手先力を外部に出力する．

5·1.2 ポート

次に，上述の RT コンポーネントに必要なポートを設計する．スレーブアーム用遠隔操作 RT コンポーネントは，目標手先位置・姿勢の入力と手先力の出力が必要であるため，このコンポーネントに最低限必要なポートは，目標手先位置・姿勢の入力用ポートが一つ，手先力の出力用ポートが一つである．また，これらのポートでは，周期的にデータの入出力を行うため，サービスポートではなくデータポートとして実装することにする．

これらのポートは遠隔操作 RT コンポーネントに最低限必要なものであるが，それらとは別な種類のポートが必要な場合がある．実際，バイラテラル遠隔操作では，上で述べた以外の制御モードもあり，それらのモードをオペレータの指令によって切り替えるためには，オペレータの指令を受けるポートが必要である．一般に，この種のポートは，オペレータの指令の多様性のために，サービスポートとして実装した方がよい場合が多いが，今回の場合，制御モードが数種類に限定されるため，データ入力ポートとして実装することにする．

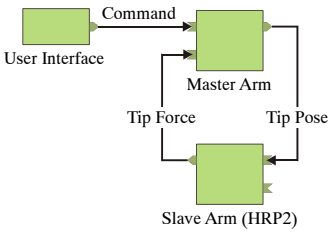

Fig.3 System diagram for bilateral teleoperation.

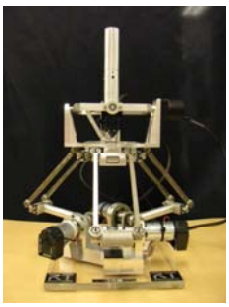

Fig.4 Haptic interface [7][8].

Fig.5 HRP2TIG.

以上をまとめると，今回設計したポートは以下の三つである．

(a) 目標手先位置入力用データ入力ポート
(b) 手先力出力用データ出力ポート
(c) オペレータ指令受信用データ入力ポート

5·1.3 データ型

最後に，ポートのデータ型を設計する．ここでは，上記の (a) および (b) のポートについて説明する．(a) の入力ポートと (b) の出力ポートのデータの種類は異なるが，ポートごとに異なるデータ型を用いると実装が複雑になるため，ここでは，入力ポートと出力ポートのデータ型を統一することにする．それぞれのポートのデータは，浮動小数点の配列で表すことができるため，入出力ポートで用いる共通のデータ型として，以下のようなものを定義して用いることにする．

```
struct BilateralSeq{
   Time tm;
   unsigned long opcode;
   sequence<float> data;
};
```

このデータ型において，data は，目標手先位置・姿勢および手先力を格納するための配列であり，opcode は，データの種類を識別するための変数である．

5·1.4 実装

以上の設計を基に，HRP2 のプラグイン形式で遠隔操作 RT コンポーネントの実装を行った．RT コンポーネントでは，目標手先位置・姿勢データを入力ポートから読み込むためのメソッドと手先力を出力ポートに書き出すためのメソッドを実装した．HRP2 のプラグインでは，入力メソッドを介して外部から取得した目標手先位置・姿勢に基づいて位置制御コントローラへ指令を出力する処理とアームの手首に搭載された力センサから取得した手先力を出力メソッドを介して外部へ提供する処理を実装した．

5·2 バイラテラル遠隔操作システム

作成した HRP2 の遠隔操作 RT コンポーネントを用いて，バイラテラル遠隔操作システムを構築した．本システムでは，マスタアームの RT コンポーネントとして，文献 [6] に示されているものを用いた．このマスタアームコンポーネントは，式 (2) で表されるコントローラを内蔵しており，スレーブアームの手先力を入力として受け取り，スレーブアームの目標手先位置・姿勢を出力する．このコンポーネントと作成した HRP2 の RT コンポーネントを RtcLink[2] を用いて **Fig. 3** のように接続した．すなわち，マスタアームコンポーネントの出力ポート（目標手先位置・姿勢）と HRP2 コンポーネントの入力ポート，および HRP2 コンポーネントの出力ポート（手先力）とマスタアームコンポーネントの入力ポートをそれぞれ接続した．このときの接続タイプは”Periodic” を選択し，周期は 5ms とした．また，マスタアームコンポーネントの実行周期は 1ms，HRP2 コンポーネントの実行周期は 5ms とした．

以上で構築したシステムの有用性を検証するため，**Fig. 4** で示される 6 自由度ハプティックインタフェース [7][8] をマスタアーム，**Fig. 5** で示される HRP2 の右腕をスレーブアームとして遠隔操作実験を行った．マスタアームコンポーネントから出力された目標手先位置・姿勢を HRP2 コンポーネントで取得し，それを HRP2 の位置制御コントローラで実現することにより，バイラテラル遠隔操作が実現できることが確認できた．

6. 結言

本稿では，ヒューマノイドロボット HRP2 のシステム上で RT コンポーネントを構築するための方法を具体的に説明した．初めに，HRP2 の制御システムの概要を述べ，そのシステム上に OpenRTM-aist-0.4.0 を移植する方法を述べた．次に，HRP2 の制御システム上で RT コンポーネントを作成するための方法を説明した．最後に，提示した手法の実装例として，遠隔操作用 RT コンポーネントを開発し，その有用性を実験により検証した．なお，本稿では，HRP2 に OpenRTM-aist-0.4.0 を移植する際に OpenRTM-aist の本体を一部変更したが，その部分のプログラムは公開する予定である．

謝辞

本研究は，原子力委員会の評価に基づき，文部科学省原子力試験研究費により実施されたものである．

参考文献

[1] 安藤 他，“OMG RTC 標準仕様に準拠した RT ミドルウェアの実装”，ロボティクスメカトロニクス講演会 2007，1P1-A02，2007.
[2] 安藤 他，“RT ミドルウェアを基盤としたロボット統合開発環境”，ロボティクスメカトロニクス講演会 2007，1P1-A03，2007.
[3] 五十棲 他，“ヒューマノイドロボット HRP-2 の開発”，日本ロボット学会誌，vol.22，no.8，pp.1004–1012，2004.
[4] ゼネラルロボティックス株式会社，OpenHRP 制御ソフトウェア，http://www.generalrobotix.com/product/openhrp/products.htm
[5] 尹 他，“バイラテラル遠隔操作を利用したタスクスキルトランスファー手法”，日本ロボット学会誌，vol.25，no.1，pp.155–165，2007.
[6] 尹 他，“OpenRTM-aist による実時間制御を考慮した RT コンポーネント”，ロボティクスメカトロニクス講演会 2007，1P1-A01，2007.
[7] 妻木 他，“小型 6 自由度ハプティックインタフェースの開発”，日本ロボット学会誌，vol.23，no.4，pp.474–481，2005.
[8] 尹 他，“構造剛性解析に基づく改良型デルタ機構の設計”，日本機械学会論文集（C 編），vol.69，no.681，pp.1358–1365，2003.