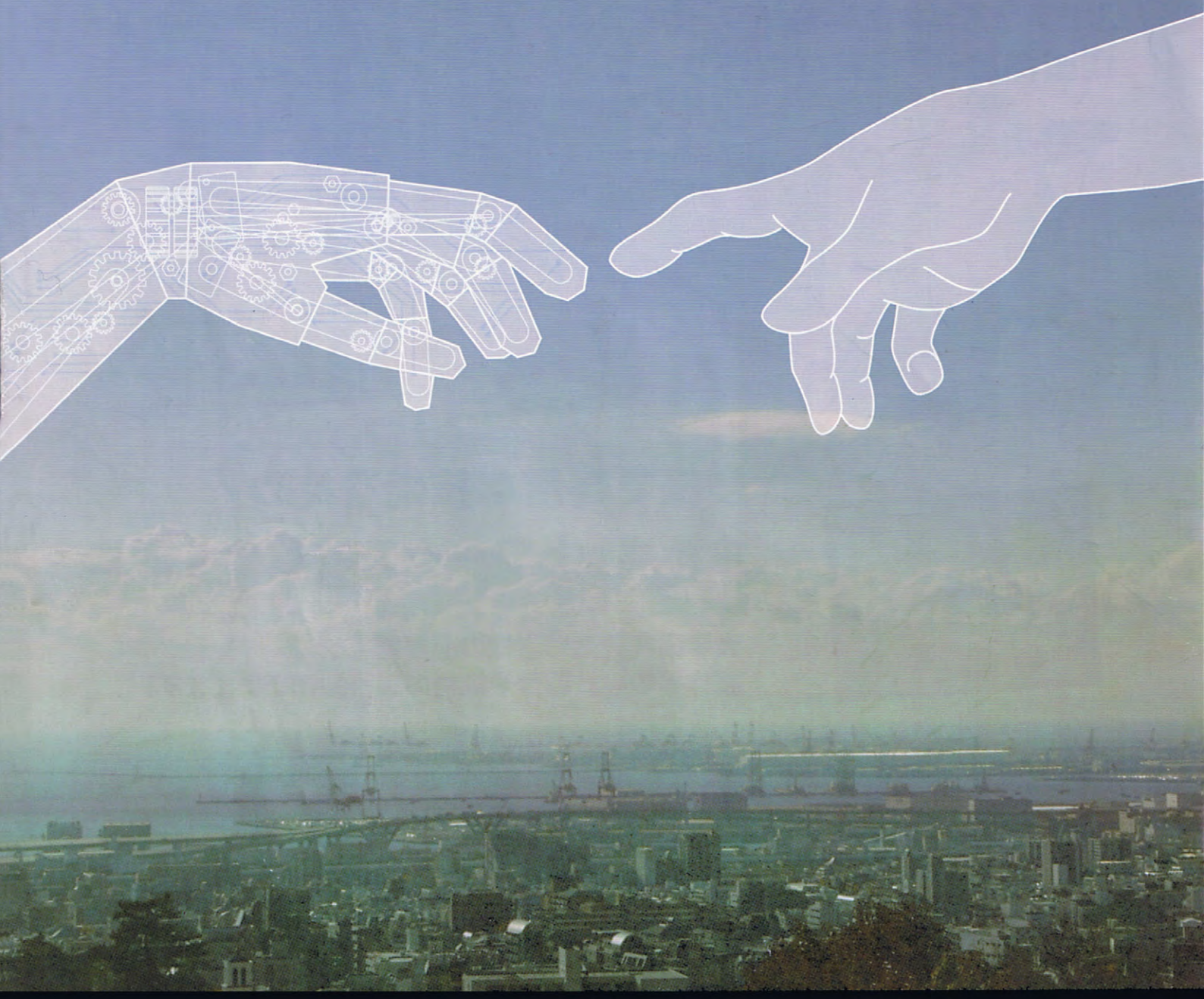
講演概要集

RSJ



第26回
日本ロボット学会学術講演会

THE 26TH ANNUAL CONFERENCE OF THE ROBOTICS SOCIETY OF JAPAN

[会期] 2008年9月9～11日 [会場] 神戸大学 [主催] 社団法人 日本ロボット学会 [共催] 神戸大学工学部・工学研究科

RTコンポーネントによるOpenHRP3制御ソフトウエア開発
–RTミドルウエアを基盤とした分散コンポーネント型ロボットシミュレータ–

○安藤 慶昭, 中岡 慎一郎, 神徳 徹雄, 比留川博久 (産総研)

Development Method for OpenHRP3 Robot Controller by RT-Component
–A Distributed Component based Robot Simulator by RT-Middleware–

*Noriaki ANDO, Shin'nichiro NAKAOKA, Tetsuo KOTOKU, Hirohisa HIRUKAWA (AIST)

Abstract—In order to increase the efficiency of robot development, the establishment of the software platform, which supports modular robotic software development, is a matter of urgency. The RT-Middleware is one of the platforms based on the distributed object technology and supports the construction of various robotic systems by the integration of robotic elements called RT-Components. However, there are few simulators which are compatible with the RT-Middleware. We developed a robotic simulator, OpenHRP3 (Open architecture Human-centered Robotics Platform 3), which guarantees the consistency of the RT-Components. In this paper, we introduce the development of the RT-Components by using OpenHRP3.

Key Words: Robot Simulator, RT-Middleware, RT-Component, Common Platform Technology, OpenHRP3

1. はじめに

ソフトウエアのモジュール化と再利用を促進し，次世代ロボットの開発を効率化することを目的とし，RTミドルウエア “OpenRTM-aist” [2] を基盤とした，分散コンポーネント型ロボットシミュレータ “OpenHRP3 (Open architecture Human-centered Robotics Platform 3)” [1] を開発した．

RTミドルウエア (RTM) は，RTコンポーネント (RTC) と呼ばれるモジュール単位でシステムを構成する，コンポーネント指向開発のためのプラットフォームである．一方，OpenHRP3 の前身である OpenHRP2 は，主にヒューマノイドロボットの制御ソフトウエア開発，動力学シミュレーションのためのプラットフォームであり，これまで両者は別々に開発されてきた．

OpenHRP3 ではシミュレーション対象を拡大するとともに，制御ソフトウエアやシミュレーション対象をRTコンポーネント化することで，システム構成の柔軟性向上，ソフトウエアの再利用性の向上を図った．これにより，RTコンポーネントとして実装されたロボットの制御ソフトウエアをシミュレータ上で検証し，検証済みのRTコンポーネントを再コンパイルすることなしに実機に適用可能となる．

本稿では，実機と可換な制御ソフトウェア開発機能を実現するためにRTミドルウェアに対して行った拡張と，それを用いて実現した OpenHRP3 の制御ソフトウェア開発機能について紹介する．

2. RTミドルウエア

RTミドルウエアはRTコンポーネントと呼ばれるモジュールの組み合わせにより，ロボットシステムを効率よく構築するためのプラットフォームである．RTコンポーネントのインターフェース仕様は， OMG (Object Management Group) で RTC (Robotic Technology Component) Specification として標準化されている [3].

2·1 RTコンポーネント

RTコンポーネントは，

- メタ情報取得のための標準インターフェース
- ロジックを実行するアクティビティ
- データ送受信を行うデータポート (InPort/OutPort)
- サービスの提供・利用を行うサービスポート
- パラメータ設定のためのコンフィギュレーションインターフェース

を備えたフレームワークに，コアロジックと呼ばれる開発者独自のロジックを組み込むことにより作成されるソフトウエアコンポーネントである．

RTコンポーネントは一種の CORBA (Common Object Request Broker Architecture) オブジェクトであり，RTミドルウエアはその生成，実行，管理，削除等のライフサイクルの管理や，ロギング，コンフィギュレーション等の共通機能の提供を行う．システムは通常複数の RTコンポーネントから構成され，CORBA が提供するネットワーク透過性によりネットワーク上に自由に配置することできる．RTコンポーネント同士

は，実行時にデータポートやサービスポートを介して接続され，相互にデータやコマンドの通信を行うことでシステム全体が動作する．コアロジックのパラメータを適切に設定しておけば，実行時にコンフィギュレーションインターフェースを介して変更することができ，同一の RT コンポーネントを再コンパイルすることなしに，様々な用途に再利用することができる．

2·2 実行コンテキスト

通常のプログラムは，起動されるとプロセスのメインスレッドや，プロセス内で生成されたスレッドにより処理が実行される．これに対して RT コンポーネントでは，スレッドを抽象的に表現した実行コンテキスト (ExecutionContext) と呼ばれるオブジェクトが，スレッドに相当する実行の主体となる．実行コンテキストは実行周期や状態マシンを持ち，状態に応じて RT コンポーネントのアクティビティ内のコールバック関数を呼ぶ．コンポーネント開発者はロボット制御等のアルゴリズムをこのコールバック関数に記述することで RT コンポーネントを実装する．

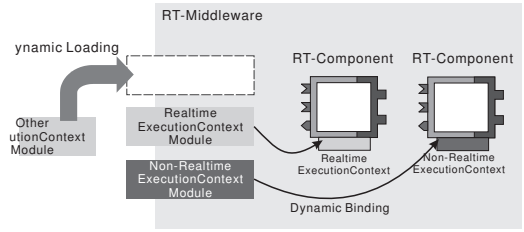


Fig.1 実行コンテキストの動的ロードと関連付け．

RT コンポーネントが生成されると，通常一つの実行コンテキストが RT コンポーネントに関連付けられ，コアロジックの実行が開始される．RT ミドルウエアでは，Fig. 1 に示すように，動的リンク可能なモジュールとして作成された複数種類の実行コンテキストを，コンポーネント生成時に選択的に関連付けることができる．例えば，非リアルタイム実行コンテキストをリアルタイム実行コンテキストに実行時に動的に切り替え特定の RT コンポーネントをリアルタイム実行することも可能である [4]．

3. シミュレータ・アーキテクチャ

現実のシステムでは，各コンポーネントにおける時間の進み方は，現実の時間の進み方と同じである．一方シミュレータにおいては，シミュレーション対象や対象を制御するコントローラが，シミュレータの管理する時間 (シミュレータ時間と呼ぶ) で動作する必要がある．そこで，Fig. 2 に示すように，同一のコンポーネントで現実時間での動作とシミュレータ時間での動作をシームレスに実現するための実行コンテキストの拡張を行った．

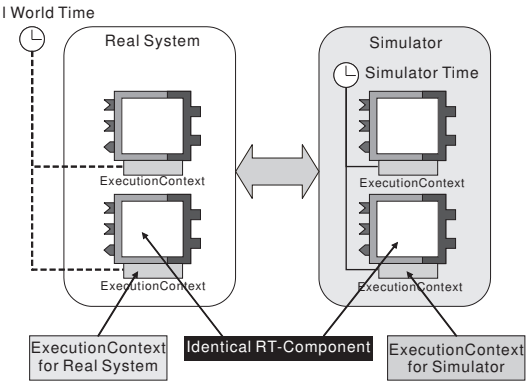


Fig.2 シミュレータ用拡張実行コンテキスト．

3·1 シミュレータ実行コンテキスト

Fig. 3 に示す CORBA IDL (Interface Definition Language) 定義のように，現実時間で動作させる際に使用する実行コンテキストの標準インターフェース (ExecutionContextService) を，シミュレータ用に拡張し，シミュレータ時間で時刻を進めるためのオペレーション (`tick()`) を追加した実行コンテキストを作成した．

```
interface ExtTrigExecutionContextService
  : ExecutionContextService
{
  void tick();
};
```

Fig.3 実行コンテキストの拡張インターフェース定義．

現実の時間刻みでロジックを駆動する代わりに，シミュレータが供給するクロック信号 (=`tick()` オペレーション呼び出し) により，関係する RT コンポーネントのロジックが駆動される．これにより，同一の RT コンポーネントを内部の変更や再コンパイルを行うことなく，実システムとシミュレータシステムの両方に利用することができる．

3·2 ブリッジコンポーネント

シミュレータ内部は大きく分けて，順動力学計算や視野画像生成を行うシミュレーションエンジン部とロボットの制御量を計算するコントローラ部，それらの動作タイミングをシミュレーション世界の時間に合わせて制御するスケジューラ部の 3 つ部分からなる．

これら 3 つの部分の呼び出し関係と処理の流れを Fig. 4 に示す．

コントローラ部は Fig. 5 に示す IDL によって定義されるインタフェースを持っており，これらのインタフェースはスケジューラ部によって制御周期毎に一度呼び出される．

各オペレーションで行われる処理は以下の通りである．

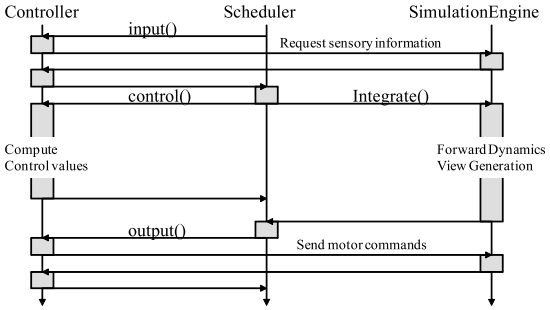


Fig.4 OpenHRP3 内の呼び出しシーケンス.

```
interface Controller
{
  // 略
  void input();
  void control();
  void output();
  // 略
};
```

Fig.5 OpenHRP コントローラインターフェース定義.

`input()` シミュレーションエンジン部から制御計算を行うのに必要なセンサ情報を取得する

`control()` センサ情報と内部状態に基づいてロボットの制御量を計算する

`output()` 計算した制御量をシミュレーションエンジン部に送る

RT コンポーネントとして実装された制御モジュールをシミュレーション世界で動作させるためにはこのコントローラ部に RT コンポーネントを接続する必要がある．そこでコントローラ部のインタフェースの実装を持ち，シミュレータ内のセンサデータを出力する OutPort と，アクチュエータに対するトルクを入力する InPort を備えた RT コンポーネントを開発した．これをブリッジコンポーネントと呼ぶ．ブリッジコンポーネントではコントローラ部のインタフェースをそれぞれ以下のように実装している．

`input()` シミュレーションエンジン部からセンサ情報を取得し，それらを OutPort に設定する．

`control()` 制御計算を行う RT コンポーネントでは外部トリガ付きの実行コンテキストを用い，このコンテキストの `tick()` オペレーションを呼び出して制御計算を 1 度だけ実行する．

`output()` InPort から制御計算の結果を読み出し，動力学計算サーバに設定する．

ブリッジコンポーネントを用いて制御用の RT コンポーネントを OpenHRP3 に接続した際の構成を Fig. 6 に示す.

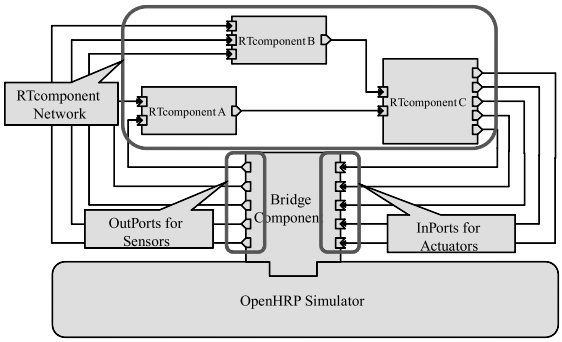


Fig.6 RT ミドルウエアに基づく OpenHRP3 の構造.

4. 制御ソフトウエアの開発

OpenHRP3 の利用者は，シミュレータ上のロボットを制御するため，Fig. 6 の RTcomponentA,B,C に相当する RT コンポーネントを開発することになる．OpenRTM-aist には RT コンポーネントのスケルトンコードを生成するコードジェネレータ rtc-tmplate が付属している．RT コンポーネントのプロファイルやポートの種類を指定すると，雛型コードが自動的に生成される．ここでは，シミュレーション対象を PD 制御するためのコントローラを RT コンポーネントとして作成する例を示す．Fig. 7 は rtc-template の実行例で，これにより angle という InPort 一つ，torque という OutPort 一つ備えた PDcontroller コンポーネントのスケルトンが生成される．

```
rtc-template -bcxx \
 --module-name=PDcontroller \
 --module-desc='PD controller component' \
 --module-version=0.1 --module-vendor=AIST \
 --module-category=Generic \
 --module-comp-type=DataFlowComponent \
 --module-act-type=SPORADIC \
 --module-max-inst=1 \
 --inport=angle:TimedDoubleSeq \
 --outport=torque:TimedDoubleSeq
```

Fig.7 rtc-template による雛型コードの生成.

生成されたスケルトンコードには，RT コンポーネントフレームワークが提供するアクティビティのコールバック関数が多数定義されており，必要とするコールバック関数を実装すればよい．`onExecute()` 関数は実行周期毎に呼ばれるコールバック関数で，主な処理はこの関数に記述する．Fig. 8 に PD コントローラの実装例を示す.

`m_angleIn` と `m_torqueOut` はそれぞれ，シミュレータのブリッジコンポーネントに接続される InPort および OutPort である．InPort にはシミュレーションエンジン部から取得されたセンサ情報が入力され，シミュレータの動力学エンジンに対するトルク入力は OutPort

```
RTC::ReturnCode_t
PDcontroller::onExecute(RTC::UniqueId ec_id)
{
  // 関節角現在値読込
  m_angleIn.read();
  for (int i = 0; i < DOF; i++) {
    // 関節角目標値 (q_ref) を与える
    // 関節角速度目標値 (dq_ref) を与える
    // InPort からのデータを読み込む
    double q = m_angle.data[i];
    // ひとつ前のデータから角速度を算出
    double dq = (q - qold[i]) / TIMESTEP;
    qold[i] = q;
    // トルク指令値を算出する
    m_torque.data[i] = -(q - q_ref) * Pgain[i]

  -(dq - dq_ref) * Dgain[i];
  }
  // OutPort に書き込む
  m_torqueOut.write();
  return RTC::RTC_OK;
}
```

Fig.8 PD コントローラ RT コンポーネントの実装例.

から出力される.

実装コードを雛型生成時に生成された Makefile を利用してコンパイルすると，実行形式のスタンドアロン型 RTC と，共有オブジェクト形式のローダブルモジュール型 RTC が生成される.

4·1 性能評価

OpenHRP に含まれるロボット歩行シミュレーションのサンプルを用い，i) 従来型 (OpenHRP2) コントローラ, ii) スタンドアロン型 RTC コントローラ, iii) ローダブルモジュール型 RTC コントローラ，について完了時間を計測し比較を行った．性能評価結果を Table 1 に示す.

表中の従来型コントローラは，OpenHRP2 で利用されていた Fig. 5 のコントローラインターフェースを実装する形で作成した．RTC 用データ型への変換処理がないため最も高速に実行されることがわかる．スタンドアロン型 RTC コントローラは，別プロセスとしてコントローラを実行した場合の結果である．シミュレーションエンジンとのプロセス間通信が行われるため，三つの中で最も遅くなる．ローダブルモジュール型 RTC のコントローラは，RT コンポーネントをシミュレーションエンジンと同一プロセス内にロードしシミュレーションを実行した結果である．プロセス間通信がなくなる分，スタンドアロン RTC コントローラより高速に動作した．以上，スタンドアロン型 RTC，ローダブルモジュール型 RTC いずれの場合も，以前のコントローラと比較して 10% 以内速度低下に抑えられていることがわかる.

Table 1 RTC コントローラの性能評価

コントローラの種類	実行時間 [sec]	増加率 [%]
従来型	42.9944	–
スタンドアロン型 RTC	46.8900	9%
ローダブルモジュール型 RTC	44.7834	5%

5. おわりに

本稿では，実機とシミュレータ間で可換な制御ソフトウェア開発環境を提供する RT ミドルウェアを基盤とした分散コンポーネント型ロボットシミュレータ OpenHRP3 のアーキテクチャを紹介した．RT コンポーネントのロジックを実行する実行コンテキストに対して拡張を行い，それを用いて実現した OpenHRP3 の制御ソフトウェア開発機能について述べた.

ロボットコントローラの開発者は，一旦コントローラを RT コンポーネントとして開発し，シミュレータ上で検証し動作することを確認すれば，同一のコントローラ RT コンポーネントを実機上でも再コンパイルすることなく動作させることが可能となる．また, コントローラを RTC 化することにより, RT ミドルウエアの各種ツール，既存の RT コンポーネントを利用することが可能となった．これにより，コントローラコンポーネントの作成，シミュレータによる検証および実機での実行をシームレスに行うことができ，ロボット開発の効率化が期待できる.

[1] 中岡慎一郎，山野辺夏樹，比留川博久，山根克，川角祐一郎, “分散コンポーネント型ロボットシミュレータ OpenHRP3”，日本ロボット学会誌，本特集号.

[2] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun Yoon, “RT-Middleware: Distributed Component Middleware for RT (Robot Technology)”, 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3555-3560, 2005.

[3] Catalog of OMG Domain Specifications, “Robotic Technology Component (RTC)”, http://www.omg.org/technology/documents/domain_spec_catalog.htm

[4] 安藤 慶昭, 清水 昌幸, 尹 祐根, 神徳 徹雄, “RT コンポーネントのリアルタイム化を実現する実行コンテキストの拡張”, 第 25 回 日本ロボット学会学術講演会予稿集, p.2I14, 2007