解 説

分散コンポーネント型ロボットシミュレータ・アーキテクチャ —RT コンポーネントを用いた実機と可換な制御ソフトウェア開発機能—

Distributed Component based Robot Simulator Architecture
—Concurrent Controller Module Development Environment based on RT-Component—

安藤慶昭* 中岡慎一郎* 神徳徹雄* *独立行政法人産業技術総合研究所
Noriaki Ando*, Shin'ichiro Nakaoka* *and* Tetsuo Kotoku* *National Institute of Advanced Industrial Science and Technology

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て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" [1] 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" [2] 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

原稿受付 2008 年 4 月 24 日
キーワード：Simulator, RT-Middleware, RT-Component
*〒 305–8568 つくば市梅園 1–1–1
*Tsukuba–shi, Ibaraki

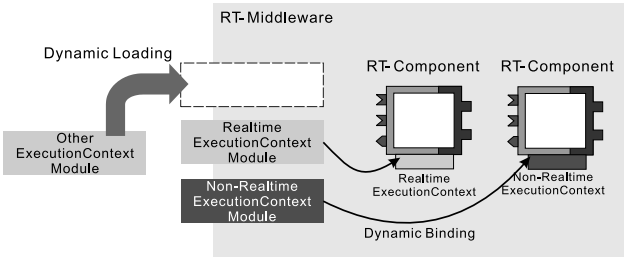


図 1 実行コンテキストの動的ロードと関連付け

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．

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，**図 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，**図 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切り替えるための実行コンテキストの拡張を行った．

3.1 シミュレータ実行コンテキスト

図 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．

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三つの部分からなる．

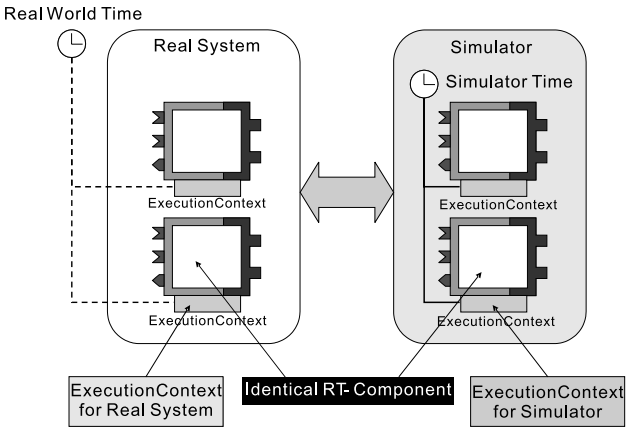


図 2 シミュレータ用拡張実行コンテキスト

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

図 3 実行コンテキストの拡張インターフェース定義

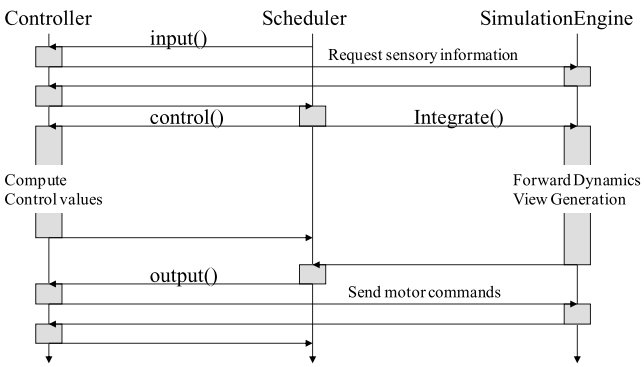


図 4 OpenHRP3 内の呼び出しシーケンス

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

図 5 OpenHRP コントローラインターフェース定義

これら三つの部分の呼び出し関係と処理の流れを**図 4** に示す．

コントローラ部は**図 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ごと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．

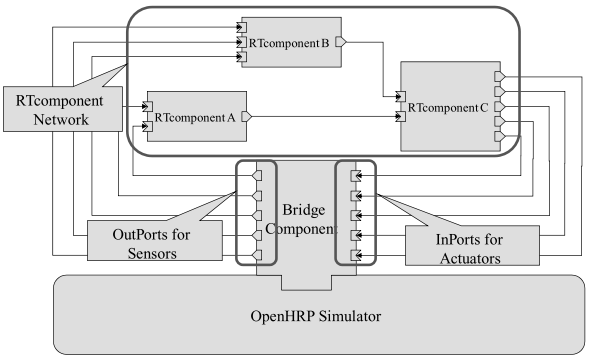


図 6 RT ミドルウエアに基づく OpenHRP3 の構造

`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一度だけ実行する．

`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**図 6** に示す．

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，図 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．**図 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
```

図 7 rtc-template による雛型コードの生成

```
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];
    // 一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;
}
```

図 8 PD コントローラ RT コンポーネントの実装例

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()` 関数は実行周期ごと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．**図 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 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 RTC と，共有オブジェクト形式のローダブルモジュール 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,

表 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%

(ii) スタンドアロン RTC コントローラ，(iii) ローダブルモジュール 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**表 1** に示す．

表中の従来型コントローラは，OpenHRP2 で利用されていた図 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分かる．スタンドアロン RTC 型コントローラは，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 RTC，ローダブルモジュール 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分かる．

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，いったんコントローラを 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．

謝 辞 本研究は，文部科学省の科学技術振興調整費による「科学技術連携施策群の効果的・効率的な推進」の一環として実施したものである．ここに感謝の意を表する．また，本研究の実施中に急逝された科学技術連携施策群次世代ロボット主監故谷江和雄教授に感謝するとともに，謹んで哀悼の意を表する．

参 考 文 献

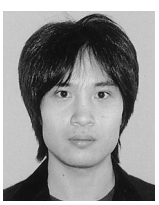
[1] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and W.-K. Yoon: "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp.3555–3560, 2005.
[2] 中岡慎一郎，山野辺夏樹，比留川博久，山根克，川角祐一郎："分散コンポーネント型ロボットシミュレータ OpenHRP3"，日本ロボット学会誌，vol.26, no.5, pp.399–406, 2008.
[3] Object Management Group: Robotic Technology Component Specification Version 1.0, formal/2008-04-04.
[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 CD-ROM，2I14, 2007.

安藤慶昭（Noriaki Ando）

1997 年 3 月京都大学工学部電子工学科卒業．2002 年 3 月東京大学大学院工学系研究科電子情報工学専攻博士課程修了．2002 年 4 月～2003 年 3 月東京大学生産技術研究所学術研究支援員．2003 年 4 月より独立行政法人産業技術総合研究所研究員．ロボットソフトウエアアーキテクチャ，RT ミドルウエアの研究に従事．博士（工学）．（日本ロボット学会正会員）

神徳徹雄（Tetsuo Kotoku）

1988 年東京大学工学部計数工学科修士課程修了．同年，通商産業省機械技術研究所入所．博士（工学）．現在，独立行政法人産業技術研究所知能システム研究部門研究グループ長．ロボット技術の共有と蓄積を図り，ソフトウェア開発効率を高める RT ミドルウェアの研究に従事．（日本ロボット学会正会員）



中岡慎一郎（Shin'ichiro Nakaoka）

2001 年東京工業大学理学部情報科学科卒業．2006 年東京大学大学院情報理工学系研究科コンピュータ科学専攻博士課程修了．博士（情報理工学）．2006 年 4 月より独立行政法人産業技術総合研究所知能システム研究部門研究員．現在に至る．ヒューマノイドロボットによる動き提示，動力学シミュレータ，ロボットソフトウェアプラットフォームの研究に従事．（日本ロボット学会正会員）