2010年(平成22年)6月15日発行(通巻203号) 年10回発行(1.3.4.5.6.7.9.10.11.12月)

ISSN 0289-1824

URL :http ://www.rsj.or.jp/

Journal of the Robotics Society of Japan

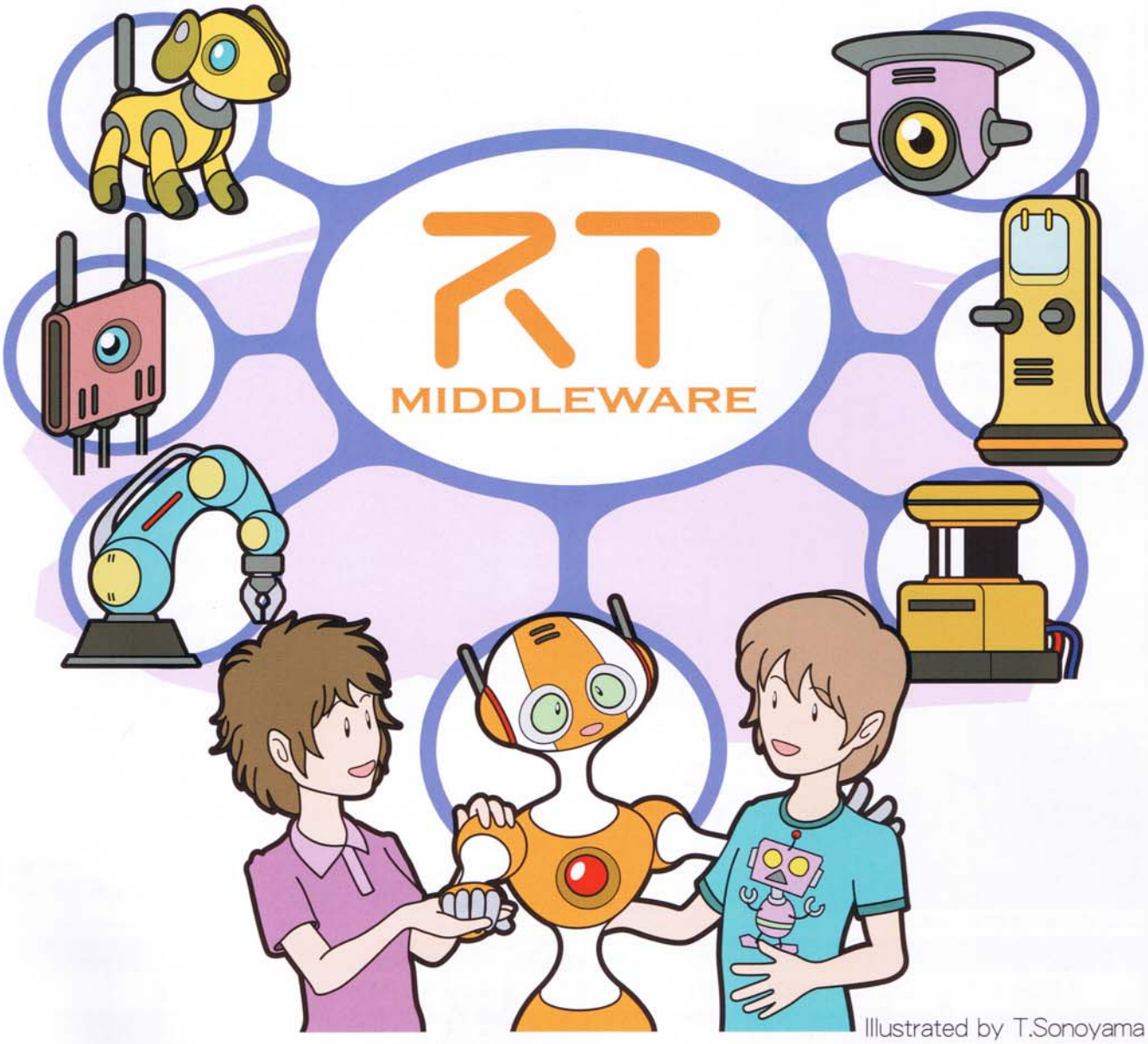
日本ロボット学会誌

RSJ

June 2010

Vol.28 No.5

[特集]
使えるRTミドルウェア



Illustrated by T.Sonoyama

RSJ 社団法人 日本ロボット学会

解 説

初心者のためのRTミドルウェア入門
—OpenRTM-aist-1.0とその使い方—

Introduction to RT-Middleware for Beginners
—OpenRTM-aist-1.0 and How to Use It—

安 藤 慶 昭* *独立行政法人産業技術総合研究所
Noriaki Ando* *National Institute of Advanced Industrial Science and Technology

1. はじめに

RTミドルウェアは，（独）新エネルギー・産業技術総合開発機構（NEDO）の21世紀ロボットチャレンジプログラム（2002年度〜2004年度）において，（独）産業技術総合研究所（産総研），松下電工（現パナソニック電工株式会社），（社）日本ロボット工業会により研究・開発・標準化が行われた，ロボットシステムを効率よく構築するためのソフトウェアプラットフォームである．その後，いくつかの国家プロジェクトにおいて利用され，現在，NEDOの次世代ロボット知能化技術開発プロジェクトにおいて，RTミドルウェアを基盤としたソフトウェアプラットフォームと，それらを利用した知能モジュール群の研究・開発が進められている．

最初のプロジェクトの成果として，RTミドルウェアの参照実装OpenRTM-aist-0.2およびそのインターフェース仕様が公開された．その後，国際標準化団体OMG（Object Manegement Group：詳細は後述[1]）においてRTCインターフェース仕様の標準化が進められ，2008年4月にOMG公式標準仕様として公開された[2]．この標準に準拠したRTミドルウェア実装OpenRTM-aist-1.0が2010年1月にリリースされた[3]．

本稿では，RTミドルウェアあるいはOpenRTM-aistに触れたことのない読者を対象に，インストールからRTCの作成方法，システムの構築方法について，新たにリリースされたOpenRTM-aist-1.0を対象に解説する．誌面の都合上，十分な解説は難しいが，詳細はWeb[3]ページや書籍[4][5]などを各自参照されたい．

2. RTミドルウェア

RTミドルウェア（RT-Middleware: RTM）とは，ロボット機能要素（RT機能要素）をソフトウェアモジュール（RT-Component: RTC）化し，これらを組み合わせロボットシステムを構築するためのソフトウェアプラットフォームである．

RT機能要素とは，ある機能を提供するロボット構成要素で，例えばサーボモータやセンサ，カメラといった単機能のものや，これらデバイスの組み合わせにより実現される移動台車，アーム等といった，まとまった機能を提供する要素を指す．また，ハードウェアに直接的に結びついているものだけでなく，制御や画像処理のアルゴリズムなど，ソフトウェアのみの構成要素もその一種と考えることができる．

2.1 モジュール分割と統合

モジュール化されたRT機能要素を階層的に組み合わせ，ロボットシステムを統合する際に基盤となるのがRTミドルウェアである．RTミドルウェアでは，ソフトウェアモジュール化したRT機能要素をRTコンポーネント（RT-Component: RTC）と呼ぶ．

図1のように，分割されたコンポーネントはそれぞれポートと呼ばれる他のコンポーネントと通信するインターフェースを持つ．ポートを介してデータやコマンドのやり取りを行い，全体として一つのまとまった機能を実現する．

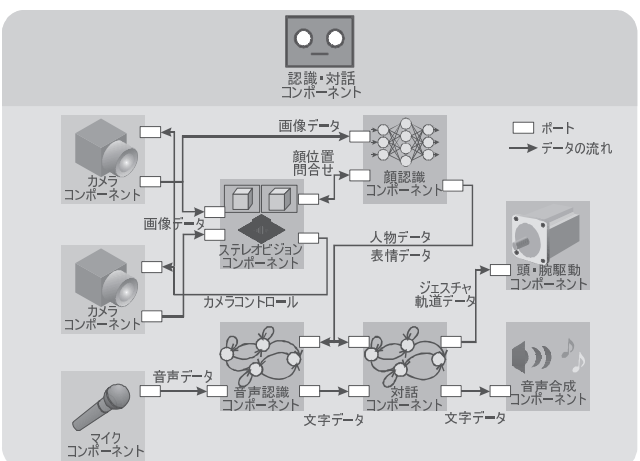


図1 対話サブシステムのコンポーネントによる構成例

原稿受付 2010年4月5日
キーワード：RT-Middleware, RT-Component, Standard
*〒305–8568 つくば市梅園 1–1–1
*Tsukuba–shi, Ibaraki

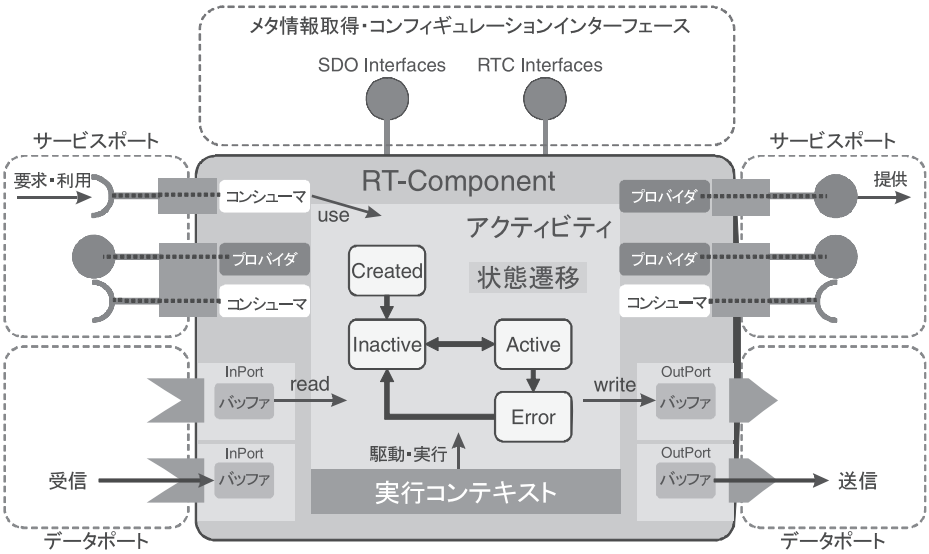


図 2 RT コンポーネントのアーキテクチャ

モジュール化することで，モジュール単位での並行開発，再利用，交換や更新，分離等が可能になるため，複雑さの軽減，開発効率の向上や，システムの柔軟性・拡張性・安定性の向上が期待できる．

2.2 RTC アーキテクチャ

RT システムでは，低レベルのセンサ処理やアクチュエータ制御から，高レベルの認識，判断，行動制御など，様々なレベルの処理を連携して行う必要がある．低レベルの制御プログラムには，速度やリアルタイム性の要求を満たす言語が求められる一方で，高レベルのプログラムには，抽象度や記述力の高い言語が求められる．また，現在の RT システムは，複数の CPU で構成されるケースが増えており，並列制御やネットワークを介した連携機能が要求される [6].

以上の要件から，これらの機能要素をモジュール化するためには，様々な粒度でモジュール化が可能で，かつ，多様な言語，OS 上で動作する，分散コンポーネント型のフレームワークが必要である．RT ミドルウェアおよび RT コンポーネントアーキテクチャは，こうした要求を満たすように設計されている．

図 2 に RTC の基本的なアーキテクチャを示す．RTC の主な機能としては以下のものがある．

メタ情報取得：RTC はメタ情報取得のためのインターフェース（イントロスペクション†機能）を持つ．実行時の動的なシステム構成変更等に利用される．

アクティビティ：コンポーネントの主たる機能を実行する部分．RTC の統一的管理のため，Inactive（OFF 状態），Active（ON 状態），Error（エラー状態）といった共通の状態遷移が決められている．RTC 開発者は，主にそれぞれの状態やイベントに割り当てられた関数（コールバック関数）に実現したい機能を実装することで RTC を作成する．

実行コンテキスト：アクティビティの関数は，実行コンテキスト（Execution Context: EC）と呼ばれるスレッドにより実行される．EC は，RTC に対して動的にアタッチ・デタッチ可能で，一つの EC を複数の RTC に対してアタッチし，直列に同期的に実行させたり，リアルタイム実行可能な EC と入れ替えることで RTC の実行をリアルタイム化することも可能である [7].

データポート：連続的なデータの送受信を行うためのデータ指向ポート．入力ポート（InPort）と出力ポート（OutPort）の 2 種類があり，同じデータ型同士なら，言語や OS が異なっていても，ネットワークを介して接続・通信することができる．

サービスポート：コマンドレベルの詳細な機能の提供・利用を行う，ユーザ定義のプロバイダ（提供（Provided）インターフェース）や，コンシューマ（要求（Required）インターフェース）を持たせることができるポート††．データポート同様，言語，OS が異なっていてもインターフェース型が同じであれば，接続し関数を呼び出すことができる．

コンフィギュレーション：ユーザ定義のパラメータを，実行時に外部から変更するための機能．複数のパラメータセットを持ち，それらを一斉に入れ替えることができる．パラメータをあらかじめ変更可能にしておくこ

†Introspection：日本語で内省と訳される，オブジェクトやコンポーネントのメタ情報を取得する仕組みのこと．定義は一定していないが，Java におけるリフレクションと類似の機能．OMG RTC 仕様 [2] ではこの機能を introspection と定義している．

††外部に機能を提供するインターフェースを Provided Interface，外部の機能を要求・利用するインターフェースを Required Interface と呼ぶ [8].

表 1 OMG RTC 準拠のミドルウェア一覧

名称	ベンダー名	概要
OpenRTM-aist	産総研	C++, Python, Java の三種類.
OpenRTM.NET	（株）セック	.NET による実装，OpenRTM-aist と互換性あり [4].
OPRoS Project	韓国 ETRI	独自ミドルウェア上の実装.
PALRO（パルロ）	富士ソフト株式会社	小型人型ロボット PARLO（パルロ）の制御ソフトウェアが C++ 言語レベルで互換.
GostaiRTC	GOSTAI/Thales	OMG RTC [2] Local PSM に準拠.

とで，RTC を様々なシステムで再利用可能にする.

一般に，RT システムにおける低レベル部分では，サーボなど粒度が細かくデータ指向の密結合なサブシステムが主体であり，判断や振る舞いを決める高レベルの部分では，粒度の粗いサービス指向のサブシステムが主体となる．RTC では，こうした多様な粒度のモジュール化を共通のフレームワークで実現しているため，階層化フレームワークで問題となる，階層間の結合は問題とならない.

異なる言語，および OS 上の RTC 間の透過的連携は，分散オブジェクトミドルウェアの標準仕様である CORBA（Common Object Request Broker Architectur）[9] を利用することで実現されている.

2.3 RTC 標準仕様

OMG（Object Management Group）[1] は 1989 年に設立されたソフトウェア標準化団体であり，分散オブジェクトミドルウェア：CORBA（Common Object Request Broker Architecture）[9]，ソフトウェアモデリング言語：UML（Unified Modeling Language）[8] をはじめとして，様々な分野のソフトウェア標準を策定・管理している組織として知られている.

RTC のインターフェース仕様も，CORBA 同様 OMG において，産総研と米国ミドルウェアベンダ RTI（Real Time Innovations）により標準化され，RTC（Robotic Technology Component）Specification [2] として 2008 年 4 月に公式リリースされた.

標準化のメリットとして，標準に基づき多くのベンダ，開発者が自由にミドルウェアを実装することができる点が挙げられる．現在，OMG RTC 仕様に準拠，または一部準拠するミドルウェアとして，**表 1** に示す 7 種類が存在する.

このうち，通信を介して互換性がある物は，OpenRTM-aist と OpenRTM.NET のみであるが，他の実装とは内部モデルが同一であるため，ブリッジ等を作り連携させた際にも，全体としての整合性に矛盾が生じることは少ない．また，複数の組織による多様な実装が存在することで，用途に応じて適切な言語やライセンスを選択することも可能であり，ミドルウェアそのものの永続性も高くなる.

3. RTC 開 発

本章では，RT ミドルウェア：OpenRTM-aist を利用して，RT コンポーネントの開発方法について述べる.

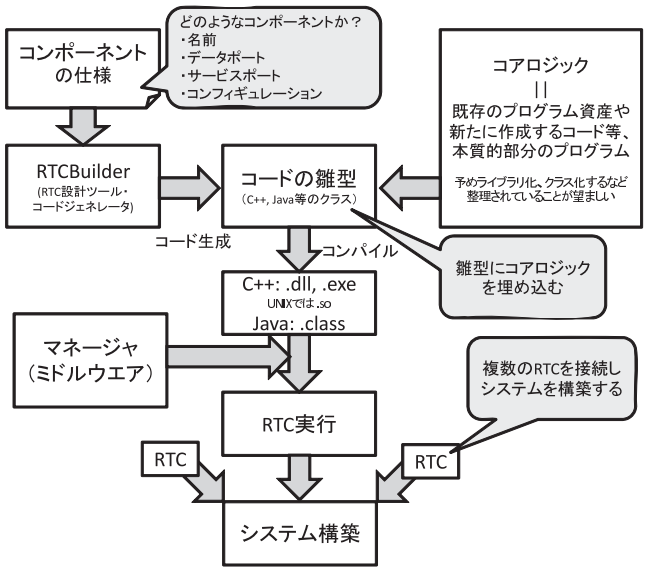


図 3 RTC および RT システム開発の流れ

3.1 開発の流れ

OpenRTM-aist はコンポーネント化のためのフレームワークと，コンポーネントを管理・実行するためのミドルウェアから構成される.

OpenRTM-aist は，コンポーネントを開発したいユーザ（コンポーネントデベロッパ）が持つ既存のソフトウェア資産，あるいは新たに作成したソフトウェアを容易に RTC 化するためのフレームワークを提供する．コンポーネント作成の大まかな流れを**図 3** に示す.

上述したように，RT コンポーネントが持つ共通インターフェースに関するコード，他のコンポーネントとのデータのやり取りの処理などは，RT コンポーネントフレームワークにより隠蔽されている．これらの処理は共通であるため，多くの部分はライブラリ化や自動生成が可能である．OpenRTM-aist では RTC の雛型を生成するためのツールとして RTCBuilder を提供している.

RTC 開発者は，自分が開発した既存のプログラムをコンポーネントフレームワークに組み込むことで RT コンポーネントを作成し，複数の RTC を組み合わせてロボットシステムを構築する．既存のソフトウェア資源をソフトウェア部品である RT コンポーネントとして作成しておけば，

表 2 Windows での言語ごとのインストール方法概要

開発環境	C++	RTC を開発するには，Microsoft Visual C++ が必要．Visual C++ Express は無料で利用可能.
	Python	Python をインストールする．バージョン 2.6 推奨.
	Java	Sun が配布する最新の JDK（Java SE Development Kit）をインストールする.
OpenRTM-aist 本体	各言語共通	ダウンロードした msi（Microsoft Windows Installer）を起動し，指示に従いインストールする．C++版は使用する Visual C++のバージョンごとに msi が存在する.
Eclipse ツール	各言語共通	ダウンロードした zip ファイルを展開するのみで，特別なインストール作業は不要．eclipse.exe を起動する.

様々な場面での再利用が容易になる．作成された RTC は，ネットワーク上の適当な場所に配置し，任意の場所から利用可能である.

RTC フレームワークに則って作成された RTC は大きく分けて 2 種類の形態がある．スタンドアロン RTC（Standalone RT-Component）は，単一の実行形式のバイナリである．ローダブルモジュール RTC（Loadable Module RT-Component）は動的にロード可能なバイナリファイルであり，1 プロセスで複数種類の RTC を同時起動する際等に利用される.

3.2 OpenRTM-aist のインストール

実際の開発の具体例に入る前に，OpenRTM-aist のインストールについて述べる．OpenRTM-aist は公式 Web サイト [3] から，C++版，Java 版，Python 版をそれぞれダウンロードできる．ソースコードおよび，Windows インストーラ，各種 Linux パッケージ版が公開されている．サンプルを動作させるだけであれば，C++版 OpenRTM-aist の Windows インストーラのみをダウンロードしインストールするのが最も簡単である．**表 2** に OS，言語ごとのインストール方法の概要を示すが，詳細は Web ページ [3] のマニュアルおよびダウンロードページを参照されたい.

3.3 RTCBuilder による雛型コードの生成

RTCBuilder は RT コンポーネントの雛型コードを自動生成する開発ツールである（**図 4**）．RTC の基本プロファイルやデータポート，サービスポート，コンフィギュレーションに関する情報を入力することで大半のコードが自動生成される．対応している言語は，C++，Java，Python，C#である．コンポーネントを作成する前に，おおよそ以下のことを決めておくとよい.

- プロファイル（名前，カテゴリ名，バージョン等）
- データポート（InPort·OutPort，ポート名，データ型）
- サービスポート（ポート名，サービスインターフェース）
- コンフィギュレーション（変数の名前，変数の型）

Eclipse メニューの「ファイル」「新規」でダイアログを開き，「その他」から「RTCBUilder」プロジェクトを選択しプロジェクト名を入力すると，図 4 の画面が現れる．RTCBuilder には，「基本」「アクティビティ」「データポート」「サービスポート」「コンフィギュレーション」「ドキュメント生成」「言語・環境」「RTC.xml」のタブがあり，「基本」から「言語・環境」までのタブで順に必要に応じて項目を埋めていく．最後に，「基本」タブにある，Output Project で先ほど入力したプロジェクト名を選択し，「コード生成」ボタンを押すことで，雛型コードが生成される．生成されたコードは，Eclipse 起動時に指定したワークスペース内のプロジェクト名ディレクトリに保存される.

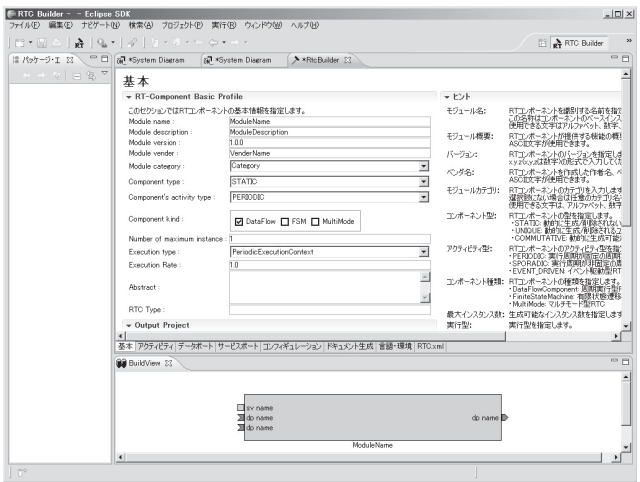


図 4 RTCBuilder

3.4 RTC の実装

RT コンポーネントのプログラムには通常のプログラムとは異なり，main 関数に直接処理を実装することはない．ここでは，例として C++版の実装について述べる.

RT コンポーネントは，ある基底クラスを継承した一つのクラスとして実装される．RT コンポーネントにおいてロジックが行う処理は，その基底クラスのメンバ関数（メソッド）をオーバーライドする形で記述する．例えば，初期化時に行う処理は，`onInitialize` 関数の中に，RTC がアクティブ時に周期的に処理したい内容は `onExecute` 関数に記述する.

図 5 は C++での実装例である．この例では，クラス宣言と実装が一体となっているが，実際にはヘッダファイル（.h）と実装ファイル（.cpp）に分割されてコードが生成される．MyLogic クラスのオブジェクト mylogic は，このコンポーネントが実際に行うコアロジックが実装されたクラスである.

```
class MyComponent
 : public DataflowComponentBase
{
public:
  // 初期化時に実行したい処理
  virtual ReturnCode_t onInitialize()
  {
    if (mylogic.init())
      return RTC::RTC_OK;
    return RTC::RTC_ERROR;
  }

  // 周期的に実行したい処理
  virtual ReturnCode_t onExecute(RTC::UniqueId ec_id)
  {
    if (mylogic.do_someting())
      return RTC::RTC_OK;
    RTC::RTC_ERROR;
  }

private:
  MyLogic mylogic;
  // ポート等の宣言
  //   :
};
```

図 5 実装の例

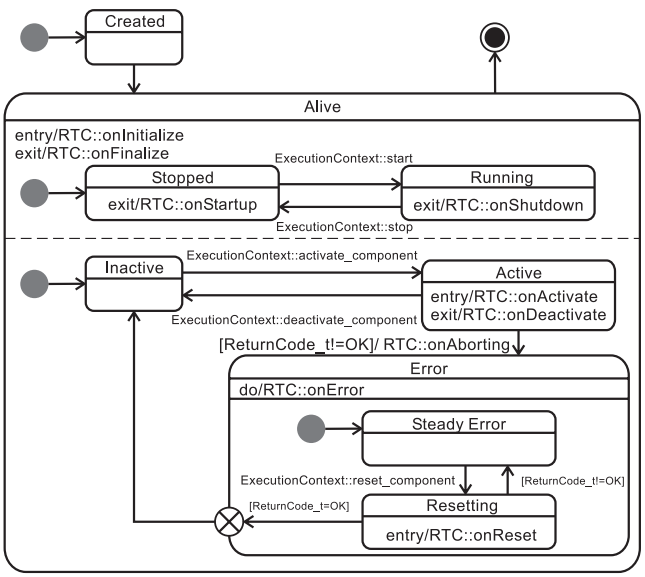


図 6 RTC ライフサイクル（UML ステートマシン図）

例では，非常に簡潔に mylogic の関数を呼ぶことで RTC が実装されているが，実際の実装でも，コアロジックをあらかじめこの程度簡単に利用できるようにクラス化しておき，RTC 内での呼び出しは最低限にすることが望ましい．

RTCBuilder により同時に生成される Makefile やプロジェクトファイルでこのコードをコンパイルすることで，実行ファイルと，共有オブジェクト（または DLL）が生成される．

3.5 RTC ライフサイクル

上述したように，RTC の実装では，あらかじめ決められた関数（コールバック関数）に処理を記述することで，コンポーネントを作成する．どのような関数があり，どういったタイミングで呼ばれるのかを知るためには，RTC の状態遷移，すなわちライフサイクルを理解する必要がある．**図 6** に，RTC の状態遷移図を示す．

表 3 RTC のコールバック関数一覧

関数名	概要
onInitialize	ライフサイクル初期化時に 1 度だけ呼ばれる．
onActivated	アクティブ化する際に 1 回呼ばれる．
onDeactivated	非アクティブ化する際に 1 回呼ばれる．
onExecute	アクティブ状態にあるとき周期的に呼ばれる．
onStateUpdate	onExecute の後に毎回呼ばれる．
onAborting	エラー状態に移行する際に 1 回呼ばれる．
onError	エラー状態にあるとき周期的に呼ばれる．
onReset	エラー状態から復帰する際に 1 回呼ばれる．
onShutdown	EC の駆動が停止する際に 1 回呼ばれる．
onStartup	EC の駆動が開始する際に 1 回呼ばれる．
onFinalize	ライフサイクル終了時に 1 度だけ呼ばれる．

コンポーネントは基本的に以下の状態を持つ．

- 生成状態（Created）
- 活動状態（Alive）
  - 非アクティブ状態（Inactive）
  - アクティブ状態（Active）
  - エラー状態（Error）
- 終了状態

これらの各状態や遷移時には，あらかじめ決められた関数（コールバック関数）が EC によって呼び出される．**表 3** に，コールバック関数とそれぞれが呼ばれるタイミングを記す．

4. システム開発

本章では，これまで述べた方法により作成した RTC を複数組み合わせて，システムを構築する手法について述べる．

4.1 ネームサービス

分散オブジェクトミドルウェアは，任意の場所の計算機上のオブジェクトに対して参照を保持するオブジェクトを介して透過的アクセスを提供する．この参照は CORBA では IOR（Interoperable Object Reference）と呼ばれ，実体はオブジェクトが存在する計算機のアドレスやポート，オブジェクト固有のキー等がエンコードされたものである．あるオブジェクトの IOR を別の計算機上のプログラムから利用する方法としては，ネットワーク上のサーバ上に登録する方法がある．これがネームサービスである．ネームサービスは CORBA で標準的に定義されているサービスの一つであり，OpenRTM-aist では，`rtm-naming` というラッパーコマンドが提供されている．

システムを起動する前に，RTC を登録するネームサーバを起動させる必要がある．また，各 RTC に対しては，ネームサーバの場所を設定ファイル “rtc.conf” に記載してあら

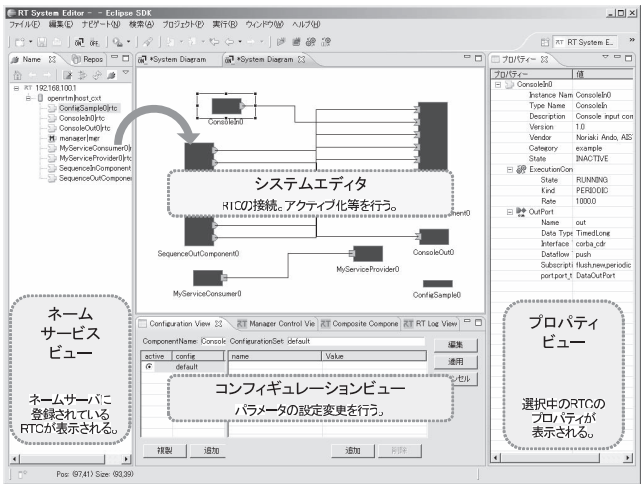


図 7 RTSystemEditor によるシステム構築

かじめ与える必要がある．例えば，ネームサーバをホスト名 “openrtm.mydomain.net” 上で起動した場合，すべての RTC には，以下のように記載した `rtc.conf` を与える必要がある．

```
corba.nameservers: openrtm.mydomain.net
```

また，ネームサーバは IP アドレスでも与えることができ，“,” で区切ることで複数のサーバに同時に RTC を登録することができる．ネームサーバは，通常長時間起動したままでよく，システムにおいて固定的であるので，設定ファイルを頻繁に書き換える必要はない．

4.2 RTSystemEditor によるシステム構築

作成されたいくつかの RTC を実行し，それらのポートを接続し，アクティブ化することでシステムが動作する．RTC 同士の接続や RTC に対してアクティブ化や非アクティブ化のコマンドを送りシステムを起動するためのツールとして RTSystemEditor が提供されている（**図 7**）．

RTC は起動されると，図 7 左の ネームサービスビューに現れる．ネームサービスビュー上の RTC を中央のエディタにドラッグアンドドロップすると，RTC がシステムエディタ内に示されるアイコンで表示される．長方形の辺上の凸部がポートを表しており，これら RTC 間のポートを接続することでシステムを構築する．また，画面中央下部には RTC のコンフィギュレーションビューが表示されており，ここで任意の RTC のパラメータを編集することができる．

システムを構築したら，エディタ上で右クリックし “All Activate” を選択することで，すべての RTC をアクティブ化可能である．また，エディタ上で右クリックし，“Save as” を選択することで，システムの構成情報を保存することができる．保存したシステム構成情報は，再度呼び出すことでシステムの接続情報，コンフィギュレーションの情報等を復元することが可能である．

現在のところ，システム構成情報を復元する際にはあらかじめ RTC を起動しておく必要があるが，将来的には RTC の起動から接続復元までが自動で行えるようになる予定である．

5. おわりに

本稿では，RT ミドルウェアの基本的概念および RT コンポーネントおよびシステムの実際の開発方法を解説した．RT ミドルウェアを用いることで，システムのモジュール化が促進され，柔軟かつ堅牢なシステム構築が容易となる．RT ミドルウェアを基盤とすることで，ロボットソフトウェアの共有，再利用が促進され，ロボットシステム開発の発展に繋がれば幸いである．

謝　辞　本研究の一部は，平成 21 年度，NEDO 次世代ロボット知能化技術開発プロジェクト（平成 19〜23 年度）の一環として実施したものである．また，OpenRTM-aist の開発にあたり，関係各位，メーリングリストや，その他コミュニティーの多くの方々から多大な助言をいただいた．ここに深く感謝の意を表する．

参考文献

[1] Object Managerment Group (OMG): `http://www.omg.org`
[2] Object Management Group: Robotic Technology Component Specification Version 1.0, formal/2008-04-04
[3] OpenRTM-aist Official Web Site: `http://www.openrtm.org`
[4] 長瀬雅之，中本啓之，池添明宏：はじめてのコンポーネント指向ロボットアプリケーション開発 RT ミドルウェア超入門．毎日コミュニケーションズ, 2008.
[5] 水川真，大原賢一，坂本武志：UML と RT ミドルウェアによるモデルベースロボットシステム開発．オーム社，2009.
[6] 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 (IROS2005), pp.3555–3560, 2005.
[7] 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.
[8] Object Management Group: OMG Unified Modeling Language (OMG UML), Superstructure, V2.2, formal/2009-02-02
[9] Object Management Group: Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, formal/2008-01-04

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〜2009 年 9 月独立行政法人産業技術総合研究所研究員．2009 年 10 月より独立行政法人産業技術総合研究所主任研究員．ネットワークロボティクス，ロボットソフトウェアアーキテクチャ，RT ミドルウェアの研究に従事．博士（工学）．（日本ロボット学会正会員）