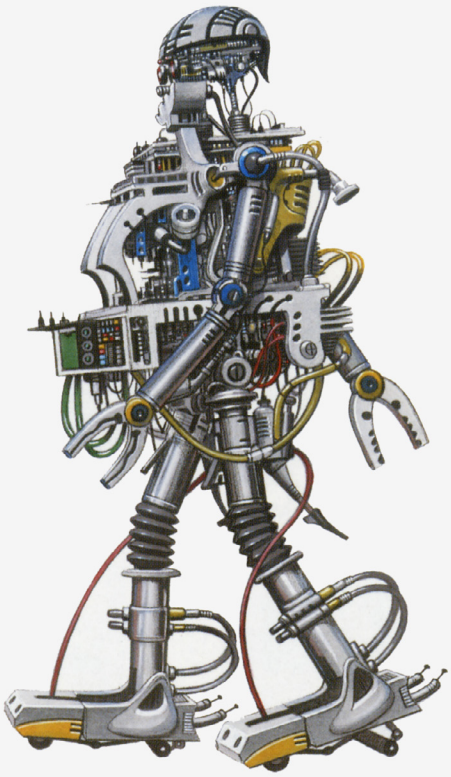
# 第23回日本ロボット学会
# 学術講演会

## The 23rd Annual Conference of
## THE ROBOTICS SOCIETY OF JAPAN

# 講演概要集

2005年9月15-17日
会場：慶應義塾大学

RSJ



主催 (社)日本ロボット学会

# RT ミドルウエアによる知能化空間のシステムデザイン

○安藤慶昭 (産総研), 新妻実保子 (東大), 都島良久 (東大), 橋本秀紀 (東大)

## System Design of Intelligent Environment using RT-Middleware

*Noriaki ANDO (AIST), Mihoko NIITSUMA (Univ. of Tokyo),
Yoshihisa TOSHIMA (Univ. of Tokyo), Hideki HASHIMOTO (Univ. of Tokyo)

**Abstract**— In this paper, we discuss about system integration method by using RT-Middleware. "Intelligent Environment", which is a typical networked robotics system, is a potential application of RT and RT-Middleware. A variety of devices, sensors and robots, which are connected to network, are distributed in the "Intelligent Environment". Therefore application software, which integrate these devices, should be flexible and extensible. To realize "Intelligent Environment", some function of software platform will be analyzed and abstracted, and a part of this function is implemented based on RT-Middleware framework.

**Key Words:** RT-Middleware, system design, integration, intelligent environment

## 1. はじめに

本稿では、ロボットソフトウエアプラットフォームである RT ミドルウエアによるシステム構築の考え方に基づいた、知能化空間のシステム構築法に関して議論を行う。

RT ミドルウエアは、RT (Robot Technology) 要素のソフトウエアモジュール化を容易にし、システムインテグレーションを効率的に行うための基本機能を備えたソフトウエアプラットフォームである [1, 2]。このモジュール化されたソフトウエア部品は RT コンポーネントと呼ばれ、RT ミドルウエア上では、これら RT コンポーネントを組み合わせることによりシステムを構築する。また、図 1 に示すように、システムの構築を体系的に行う方法論を確立するために、実装部分の統一的記述をサポートするツールとしての側面もある [3]。

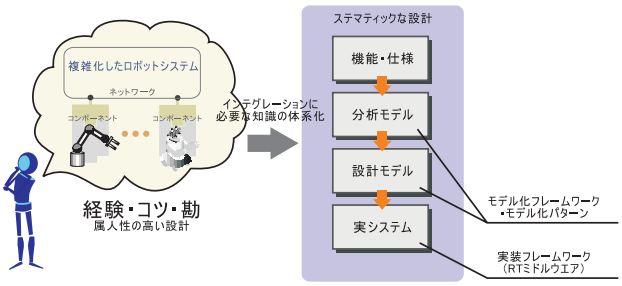



**Fig.**1 RT ミドルウエアによる体系化されたシステム構築.

RT ミドルウエアは、RT を単体のロボットのみならず、その技術要素レベルで様々な用途に応用するためのプラットフォームを提供することを目指している。特に、実生活空間の知能化やユビキタス化といった分野は、RT および RT ミドルウエアの本格的応用が期待される分野の一つである。

本稿では、RT ミドルウエアに基づき、コンポーネント指向により知能化空間を設計し、実現する際に必要となる機能、考慮すべき事項について分析を行う。その上で、従来の RT ミドルウエアにプラットフォームとして不足しており、追加すべき機能、フレームワークを提供すべき部分などを明らかにする。この議論から抽出された追加すべき機能のうち、本稿では特に、組込機器等のリソースの乏しいデバイスをミドルウエアネットワークに参加させる方法について、一つの方法を提案する。

## 2. 知能化空間のコンポーネント指向開発

ロボット研究はこれまで、機械機能の解析 (アナリシス) および比較的単純な機能の実現が重点的に行われてきた。これに対して知能化空間では、空間内にセンサやアクチュエータ、あるいはロボット技術を導入し、既存 RT 要素のインテグレーションにより、日常生活空間でいかに人間の役に立つ機能を実現し、物理的な支援も含めた QOL(Quality of Life) 向上に役立てるかに主眼を置いている。すなわち、これまでロボット工学により培われてきた多種多様な技術を、如何にインテグレーションし機能を実現するか、ということが主要な問題の一つとなる。

知能化空間には、多くのセンサやアクチュエータ、ロボットなどが分散配置され、それらがネットワークにより結びついたシステムとなる (図 2)。これらを管理・統合するソフトウエアは、柔軟かつ拡張性の高いものである必要がある。

一方、RT ミドルウエアは分散オブジェクトミドルウエアをベースにしたロボットソフトウエアプラットフォームであり、ネットワーク指向かつコンポーネント指向のこういったアプリケーションを構築するのに適している [3]。

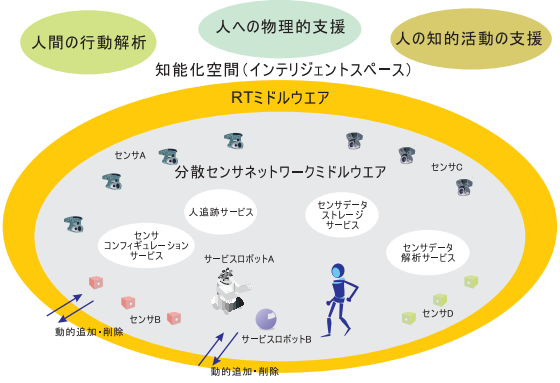



**Fig.**2 RT-Middleware による知能化空間.

## 3. システムの機能分析

知能化空間を RT システムとして構築するために必要となる、共通基盤機能の分析を行う。すなわち、知能化空間のためのプラットフォームに必要とされる機能を分析し、現在の RT ミドルウエアに追加すべき機能を抽出する。これから議論する RT ミドルウエアは、知能化空間内で必要となる様々な機能を設計・実装する上で、十分な機能を有するプラットフォームでなければならない。

具体的な機能が決定していない段階で、プラットフォームに必要とされる機能を完全に抽出することは困難である。しかしながら、様々な機能をコンポーネント指向で設計・実装を行うと仮定することで、必要な機能の多くの部分はパターンに当てはめることができ、柔軟かつ拡張性の高いプラットフォーム構築に対する指針が得られるものと考えられる。

多数のデバイスがネットワークで協調動作する知能化空間のシステムにおいてプラットフォームに必要とされる機能は以下のようなものが挙げられる。

- デバイス、コンポーネントの動的追加・削除
- デバイス群のネットワークによる統合・協調制御
- デバイス間・コンポーネント間の密結合
- デバイス間・コンポーネント間の疎結合
- 様々なアプリケーションシナリオの実行
- 軽量プロトコルの提供
- RTM と他の軽量プロトコルのブリッジの提供

この内、「デバイス、コンポーネントの動的追加・削除」、「デバイス群のネットワークによる統合・協調制御」、「デバイス間・コンポーネント間の密結合」は現状の RT ミドルウエアの機能でほぼ実現することができる。以下に、新たに設計・実装する必要のある他の機能について述べる。

### 3·1 デバイス間・コンポーネント間の疎結合

現在の RT ミドルウエアでは、コンポーネント間の結合は主に InPort/OutPort による密な結合を主体としている [1, 2]。一方、コンポーネントに対してユーザが定義した単発的なコマンドを実行させるフレームワークは備わっておらず、これが必要な場合はユーザが独自に実装する必要があった。そこで、ユーザ定義のコマンドをコンポーネントが提供する「サービス」として、容易に定義・実装できるフレームワークを提供し、RT コンポーネントのインターフェース定義を拡張した。以下に、インターフェース定義の一部を示す [1]。詳細はページの都合上割愛する。

```
interface RTComponent {
    :
  ServiceProfileList get_service_profiles ();
  ServiceProfile get_service_profile (in string id);
  RTCService get_service (in string id);
};
```

### 3·2 様々なアプリケーションシナリオの実行

知能化空間においては、内部の人間に対して情報的・物理的サービスを提供するために、一連の動作シーケンスを実行する必要がある。こうした、動作シナリオには抽象レベルでの記述性に優れたスクリプト言語を用いることにより容易に実現できる。RT ミドルウエアでは、記述のためのスクリプト言語として Python をサポートしているが、コンポーネント群を統合し一連の動作を記述することができるフレームワークは備わっておらず、シナリオ記述フレームワーク及び、これを実行するサービスを提供する必要がある。

[1] ここに示したインターフェース記述は産総研版実装 OpenRTM-aist においては名称・仕様等が変更される可能性がある。

### 3·3 軽量プロトコルとブリッジ

知能化空間においては、リソースの乏しいマイコンにより実装された、センサ、無線ネットワークノードや RFID 等のデバイスが多数用いられる場合がある [4]。これらのデバイスでは、現在の CORBA に依存した RT ミドルウエアをそのまま実装することはできない。

これに対して、RT コンポーネントのモデルを、マイコン等のデバイスに実装可能な形式に移植し、それらと、RT ミドルウエアとのブリッジを提供することにより、RT ミドルウエアネットワークに参加させる方法が考えられる。

## 4. RT コンポーネント

ここではまず、RT コンポーネントのオブジェクトモデルについて整理する。

図 3 に RT コンポーネントのアーキテクチャ・ブロック図を示す。RT コンポーネントは RT ミドルウエアにおけるソフトウエアモジュール化の単位であり、ネットワーク上に分散されたコンポーネントへの透過的アクセスを実現するために、分散オブジェクト技術 (CORBA) を用いて実装さている。

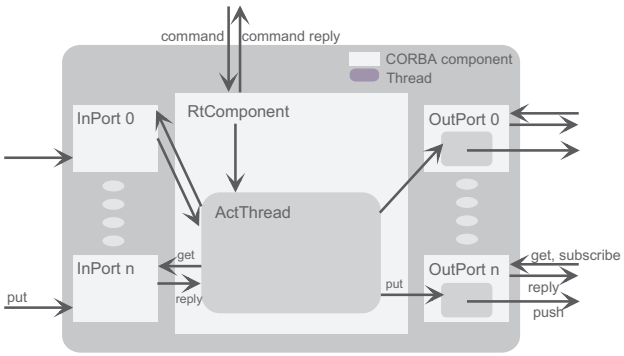



**Fig.3** RT コンポーネント.

### 4·1 RTComponent

RT コンポーネントの本体であり、基本的なインターフェースおよび入出力を行う InPort/OutPort オブジェクトを 0 個以上持つ。処理を行うコアロジックを持ち、外部または内部からのイベントに応じて内部状態を遷移させる。これをアクティビティと呼び、他とは独立したスレッドに割り当てられ以下に挙げる入出力とは独立に処理が行われる。

### 4·2 InPort/OutPort オブジェクト

InPort/OutPort は他のコンポーネントとのデータのやり取りを行う入出力ハンドリングオブジェクトである。RT コンポーネント内部のコアロジックの実行を妨げずに、非同期的に入出力を行う Publisher/Subscriber モデルに基づいている。非同期通信だけでなく、複合コンポーネントを利用することにより同期的入出力もサポートされている。

### 4·3 アクティビティ

さまざまなコンポーネントデベロッパによって作成された多数のコンポーネントを統一的に扱うには、コンポーネントのライフサイクルにおいて共通の状態遷

移を持たせると管理が行いやすい。共通の状態遷移を持たせ、その意味を予め規定しておくことにより、多数のコンポーネントの挙動を統一的に制御することが可能となる。

RT コンポーネントのアクティビティは、BORN, INITIALIZE, READY, STARTING, ACTIVE, STOPPING, ABORTING, ERROR, EXITING, FATALERROR, UNKNOWN の 11 の状態を持つ。図 4 にアクティビティ部の状態遷移図 (UML ステートチャート) を示す。

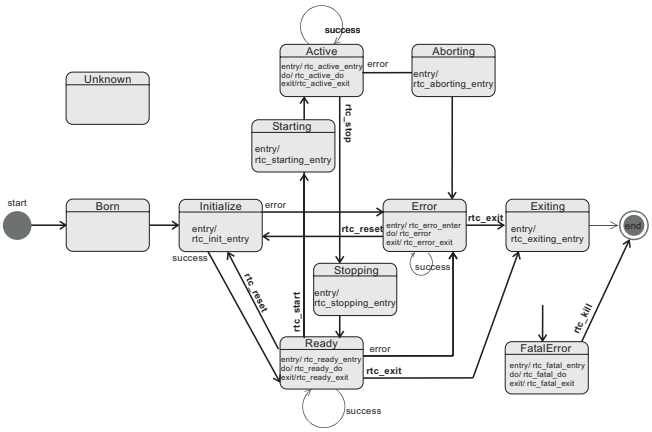



**Fig.**4 アクティビティの状態遷移図

## 5. 軽量版コンポーネント: RTC-Lite

ここでは、上述したマイコン等を用いた小型デバイスを、RT ミドルウエアネットワークに参加させるために実装した、軽量版コンポーネント “RTC-Lite (RT-Component Lite)” について言及する。

### 5·1 small RTUnit

知能化空間においては、カメラと画像処理によるセンシングといった高機能なセンサ以外にも、非常に多くのシンプルなセンサデバイスを分散配置しセンシングを行う方法も用いられる。また、RFID や無線ネットワークノードを利用したり、家電の ON/OFF 制御といった簡単な組込デバイスを利用したい場合が多数ある [4]。これらのデバイスの制御を、全て PC 程度の CPU を持つ機器を用いて行うことは現実的ではなく、また、こうした組込デバイスが RT ミドルウエアにより統合できなければ、プラットフォームとして十分な機能を提供しているとはいえない。

こうした組込デバイスを RT ミドルウエアネットワークで統合できることを実証するため、比較的簡単な IO を備え、小型で、ネットワーク接続可能な実験用制御ユニットとして、”small RTUnit” を開発した。small RTUnit は、入出力制御用マイコンとして PIC 16F877A、ネットワークインターフェースとして Lantronix 社の XPort を搭載したプログラマブルなネットワーク IO デバイスである (図 5)。

例えば、このユニットと焦電センサを用いることで、図 6 に示す人体感知センサユニットを実現することができる。さらに、このセンサを多数部屋の中に配置すれば、部屋の中の人の動きを捉えることができ、ロボットの制御等に利用することができる。図 7 は、このセンサユニットを用いた例として作成した人の動きを視覚化するアプリケーションである。

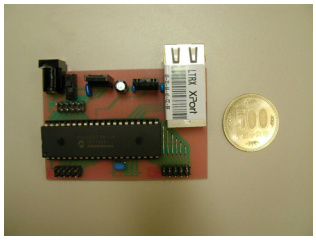


**Fig.**5 small RTUnit.

**Table** 1 small RTUnit 仕様.

| | |
|---|---|
| マイコン | PIC 16F877 20MHz |
| 通信インターフェース | Lantronix XPort |
| ロジック電源 | 5V |
| A/D | 8ch 10bit(逐次変換型, 35$\mu$s/ch) |
| DIO | 24ch (内 8ch は A/D と共用) |
| シリアル | 2ch (内 1ch は Xport との通信用) |
| その他 | 12V 電源コネクタ |

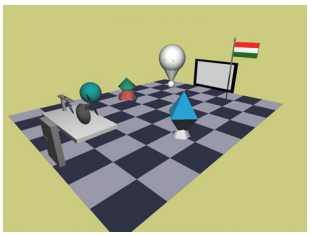


**Fig.**6 センサユニット.　**Fig.**7 センサ情報の視覚化例.

こうした組込デバイスが RT ミドルウエアにより統合することができれば、様々なアプリケーションを容易に構築できるだけでなく、ネットワークを通じて室内の他のロボットのセンサとしても容易に利用できる。

### 5·2 RTUnit 用軽量プロトコル

現在、RT ミドルウエアは CORBA 上に実装されているが、リソースの乏しい組込デバイスにおいて CORBA を利用するのは、困難であり、場合によっては不可能である。

そこで、従来の CORBA による RT コンポーネントの記述を、オブジェクトモデルとして抽出し、それをプラットフォーム (この場合組込デバイス) 独自のプロトコルで実装しなおすことにより、他の RT コンポーネントとの互換性および相互運用性を確保した。

具体的には、図 8 に示すように、デバイス上のプログラムと、サーバ上のプロキシコンポーネント間のプロトコルを、RT コンポーネントのオブジェクトモデルに基づき定義し、RTUnit があたかも 1 つの RT コンポーネントのように振る舞い、RT ミドルウエアにより統合可能にした。

上述したように、RT コンポーネントのオブジェクトモデルでは、以下の内部オブジェクトが存在する。

- アクティビティ (状態遷移をもつ)
- InPort (入力データハンドリング)
- OutPort (出力データハンドリング)
- コマンド (ユーザが定義できるコマンド)

上記に基づき、RTUnit と プロキシコンポーネントとの簡易プロトコルを定めた (図 9)。

プロキシコンポーネントは、外部からの RTUnit コンポーネントに対する要求を上記のプロトコルに変換し RTUnit に対して送る。あるいは、RTUnit からの

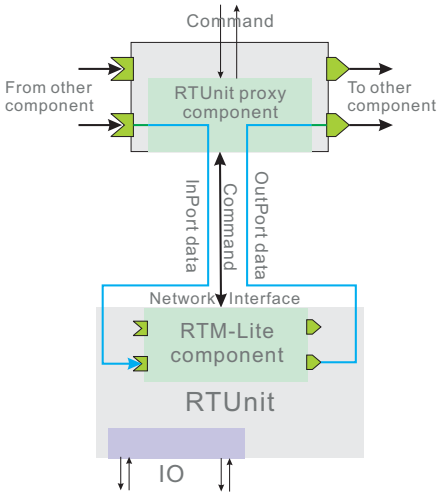



**Fig.**8 RTM-L コンポーネントと Proxy コンポーネント.


**Fig.**9 RTM-L メッセージ形式.

**Table** 2 RTM-Lite プロトコル

| | |
|---|---|
| Header | ヘッダ |
| Level0 | 上記の各オブジェクトを選択するコマンド |
| Level1 | 上記の各オブジェクトのコマンド |
| Data | 引数をマーシャリングしたデータ |
| Parity | パリティ |

データを RT ミドルウエア (CORBA) のプロトコルに変換し、他のコンポーネントやクライアントプログラムに対して送る。

これにより、CORBA により実装された従来の RT コンポーネントと、異なるプロトコルで実装された簡易コンポーネントが同一のミドルウエアで協調することができる。

### 5.3 テンプレートコードジェネレータ

RT ミドルウエアでは、コンポーネントの仕様 (各種プロファイルや InPort/OutPort の型・数) を与えることで、雛形コードを生成するテンプレートジェネレータを提供している。

上述したように、本稿で提案した RTC-Lite は、従来の RT コンポーネントと同じモデルを採用している。したがって、RTC-Lite も同等のプロファイル情報により記述できる。

そこで、テンプレートジェネレータの付加機能として、small RTUnit 用のテンプレートジェネレータを作成した。図 10 に示すように、同一のプロファイル情報から、PC 上で動作する従来の RT コンポーネントと small RTUnit で動作する RTC-Lite のコードを生成することができる。

両者が生成するコードは共に、状態遷移ロジック (図 4) や InPort/OutPort の管理等は Frozen Spot (フレームワークにおいて固定された機能を提供する部分) としてすでに定義されたコードとして生成され、コンポーネント開発者は、各状態におけるロジックや InPort/OutPort のデータの扱いといった Hot Spot のみを記述すれば良い様にフレームワーク化されている。両者は内部の実装こそ異なるが、外部からコンポーネントとしてアクセスした場合には、同様の RT コンポーネントとして扱うことができる。

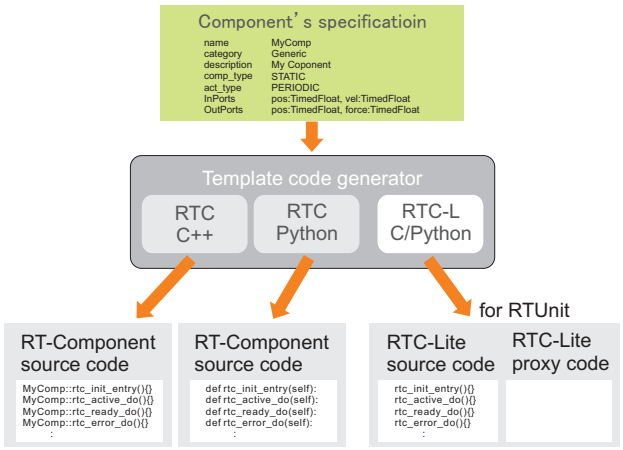



**Fig.**10 コードジェネレータ.

## 6. おわりに

本稿では、知能化空間のためのプラットフォームを RT ミドルウエアにより構築する手法について議論した。知能化空間のためのプラットフォームに必要とされる機能を分析し、現在の RT ミドルウエアに追加すべき機能を明らかにした。

これらの機能のうち、小型の組込デバイスを RT ミドルウエアネットワークに参加させるための方法として、簡易プロトコルとプロキシコンポーネントによる方法 (RTC-Lite) を提案した。RTC-Lite を従来の RT コンポーネントと同様のオブジェクトモデルとして構築することにより、コンポーネントの仕様を共通化でき、容易に RT ミドルウエアネットワークに参加させる方法を示した。

今後は、本稿で明らかになった追加すべき機能を順次実装し、実際に知能化空間構築に用いることで、システム構築の方法論の確立に役立てる予定である。

### 参考文献

[1] 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, 安藤 慶昭, "RT コンポーネントの実装例.RT ミドルウエアの基本機能に関する研究開発 (その 1)", 第 21 回 日本ロボット学会学術講演会予稿集, p.1F27, 2003.09

[2] 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, 安藤 慶昭, "RT コンポーネントの実装例.RT ミドルウエアの基本機能に関する研究開発 (その 2)", 第 21 回 日本ロボット学会学術講演会予稿集, p.1F28, 2003.09

[3] 安藤慶昭, 末廣尚士, 北垣高成, 神徳徹雄, 尹祐根, "RT コンポーネントによるシステム構築法 -RT ミドルウェアの基本機能に関する研究開発 (その 14)-", 日本機械学会ロボティクス・メカトロニクス講演会 2005, p.2A1N072, 2005.06

[4] 大原賢一, 大場光太郎, 金奉根, 谷川民生, 平井成興, 谷江和雄, "空間機能化のための空間機能モジュールの提案", 日本機械学会 ロボティクス・メカトロニクス講演会 2005, p.2A1N055, 2005.06