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)-

System Development Method using RT Components
R & D of RT Middleware Fundamental Functions (Part 14)

○安藤慶昭 (産総研), 末廣尚士 (産総研), 北垣高成 (産総研)
神徳徹雄 (産総研), 尹祐根 (産総研)

*Noriaki ANDO (AIST), Takashi SUEHIRO (AIST), Kosei KITAGAKI (AIST),
Tetsuo KOTOKU (AIST) and Woo-keun YOON (AIST)

Abstract— In this paper, we discuss about system development methodology by using RT-Components on RT-Middleware. RT-Middleware has been developed for robotic software platform which promotes application of Robot Technology (RT) in various field. We developed manipulator tele-operation system as RT-Component based systems. And through this development process, which is composed of "analysis", "design" and implementation, we tried to derive systematic RT-system development methodology and discussed about it.

Key Words: RT(Robot Technology), software component, middleware, robot system

1. はじめに

近年、社会的要請からロボット機能要素 (RT[1]: RobotTechnology[1)]) をロボットのみならず、実生活空間の知能化やユビキタス化に適用し、本格的応用を模索する動きが盛んになってきた。また、ロボット工学においては個々の機能の研究と共に、これまでロボット工学が積み上げてきた様々な知見をいかに組み合わせシステムを構築するかといった、インテグレーション手法が重要となりつつある。

こういった背景から、属人性を排除したシステムインテグレーションを体系的に実践する知識と、同時にこれをサポートするプラットフォームの必要性が高まってきた (図 1)。

著者らは、RT のソフトウエアモジュール化を実現し、システムインテグレーションを効率的に行うためのソフトウエアプラットフォーム「RT ミドルウエア」の基本機能に関する研究開発を行ってきた。

これまで RT ミドルウエアのソフトウエア部品：RT コンポーネントのアーキテクチャを提案し [2, 3, 4, 5, 6]、RT コンポーネントを用いて実際にいくつかのシステムを構築しその有用性を示した [7]。

本稿では、RT コンポーネントによりシステムを構築するための方法論について議論を行う。始めに、システム構築を実装レベルでサポートするために開発されたプラットフォーム「RT ミドルウエア」およびそのソフトウエア部品である「RT コンポーネント」について概要を説明する。次に、RT コンポーネントを使用してシステムを構成する方法について、具体的な例を挙げて、システム仕様の分析、設計、実装を行いながら検討を加える。最後に、システム構築を系統的に行う方法について議論する。

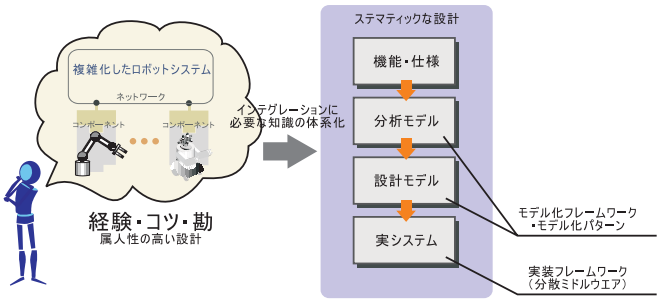

Fig.1 勘に頼る設計からシステマティックな設計へ.

[1]RT (Robotic Technology) とは、「ロボット技術を活用した、実世界に働きかける機能を持つ知能化システム」に関する技術の総称である。移動ロボット、マニピュレータなどロボット単体のみならず、知能化空間など一見ロボットには見えないシステムも、RT の集合体とみなす [1)]。

2. 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。その一方で、解析の結果やそこから得られた機能を実世界に適用するには、これらを再統合 (シンセシス) する必要がある。現在のロボット工学においては、このシンセシスの知識は開発者や研究者の経験や勘に依存しており、体系化されているとはいい難い。

2·1 RT システム開発

ロボットシステムの開発フローは、ソフトウエアシステム開発のそれを参考にするならば図 2 のように考えることができる。

研究者や開発者が経験と勘によって行ってきた、ロボットシステムを構築するために必要な知識を大別すれば、システムを分析するための知識、設計に関する知識、実装に関する知識に分けることができる。

さらに、分析はシステムが要求する仕様を実際の技術レベルに即した形で分析を行うための知識 (分析パターン) と枠組み (分析モデルフレームワーク) に分けられる。また、設計や実装に関しても設計をより現実的なレベルに落とし込む知識 (設計パターン) および枠組み (設計フレームワーク) から成る。

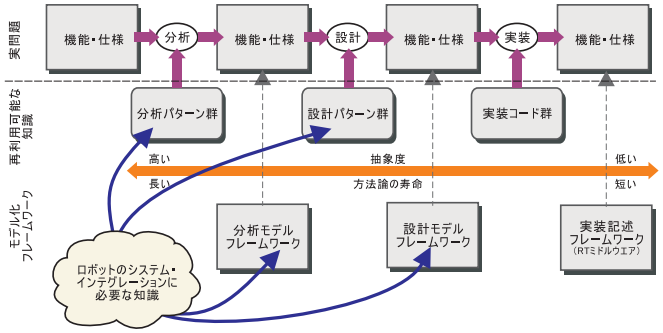

Fig.2 RT システム設計.

図 2 に示すように、実装に近いものほど抽象度が低く、一般に寿命が短い。したがって、実装レベルに近いフレームワークだけでは技術が陳腐化すればすぐに使えなくなる可能性がある。逆に、より一般化されたパターンやモデル化フレームワークを獲得することができれば、技術的進歩により実装技術が変化しても、すぐに新しい技術を用いたロボットシステム開発を行うことができる。

2·2 RT ミドルウエア・RT コンポーネント

RT ミドルウエアは図 2 の実装記述を支援するフレームワークおよびツール群の総称である。RT コンポーネントは、フレーム

ワークを用いて記述されたソフトウエア部品を指す。以下、RT コンポーネントについてそのオブジェクトモデルの概要を述べる。

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を用いて実装される。現在開発中の RT ミドルウエアは言語、OS 非依存なプラットフォームを目指しており、CORBA を用いて実装が進められている。

通常の分散オブジェクトに対する RT コンポーネントの特長は以下のとおりである。

- コンポーネント自体が常に動作し続ける「アクティビティ (Activity) 」を持つ。
- 入出力の相互接続互換性を保証する共通インターフェース (InPort/OutPort) を持つ。
- アクティビティはコンポーネント間の互換性を保証する共通の状態遷移を持つ。

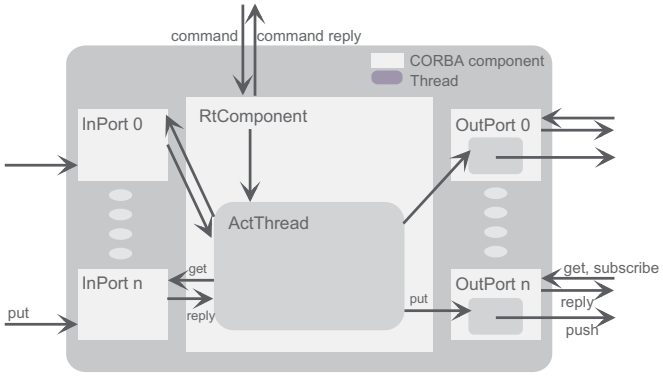

Fig.3 RT コンポーネント.

通常の分散オブジェクトでは、オペレーションに対して何らかの処理を行い、その結果を返すといった動作が一般的である。RT コンポーネントはコンポーネント自体が常に動作し続けるアクティビティを持ち、これがロボット等デバイス制御の主体となる。

一方、ロボットの制御においては、機能ブロックをデータを「受け取り」、「処理」し、結果を「出力する」機能単位とするとシステム構成が行いやすい場合が多い。RT コンポーネントはデータの受け渡しを行う共通インターフェースとして InPort/OutPort とよばれる入出力ポートを持つ。

また、様々な機能を持つコンポーネントでも、それらが共通のソフトウエア部品として扱うことが出来るように、コンポーネントとしての状態遷移が規定されている。その規定に従ってコンポーネントを構成することにより、様々な粒度のコンポーネントでも同様に扱うことが出来るようになっている。また、この状態遷移を規定することにより、コンポーネントを入れ子にした複合コンポーネントを実現することが可能となる。

2·3 ソフトウエアの再利用

RT ミドルウエアは、ソフトウエアの再利用性を向上させ、システム構築の効率を高めるのが目的の一つである。ここで言う再利用とは 2 つの側面がある。

一つは、既存のソフトウエア資産の再利用である。RT ミドルウエアでは、これまで作成されたソフトウエア資産を再利用し、容易に RT コンポーネント化するためのフレームワークを提供している。こうしたフレームワークの存在は、ソフトウエアの再利用の観点から非常に重要となる。

もう一つは、コンポーネントレベルでの再利用である。RT コンポーネント化されたソフトウエア部品は、コンポーネント化する過程で仕様やインターフェースが明確化さるので、第三者が利用する場合にもコンポーネント内部について深く知らなくとも新たなシステムに組み込むことが出来る。

こうした 2 つの再利用性が向上することにより、システム構築の効率が上がり、実際のシステム構築においてより本質的な部分に注力できるようになるため、高度で複雑なシステム構築が容易に行えるようになるものと考えられる。

Table 1 システムに使用するハードウエア.

マニピュレータ	
マニピュレータ 1	三菱重工製 PA10
マニピュレータ 2	川田工業製 HRP2 プロメテ (アーム部)
ヒューマンインターフェースデバイス	
ジョイスティック	ニッタ 6 軸力覚センサ IFS
ジョイパッド	PlayStation 用 アナログコントローラ

3. システム構築例

本稿では、あるシステムを実現する手順を例にとり、コンポーネントの組み合わせによりシステムを実際に構築する。具体例として、マニピュレータの遠隔操作システムを挙げる。

3·1 システム分析

システム構築に際し、システムにおいて必要とされる機能を仕様として以下に挙げる。

- ジョイスティックを使用しマニピュレータの手先位置姿勢を制御する
- マニピュレータは速度指令を受け取るものとする
- ジョイスティックは 2 種類存在する
- マニピュレータは 2 種類存在する
- それぞれを任意の組み合わせで接続できるようにする
- ネットワーク経由での通信に関してはリアルタイム性は考慮しない

なお、システム構築にのためのハードウエアは表 1 に示すものを使用した。

3·2 システム設計

上記の仕様から、システム構築に必要なソフトウエアモジュールをコンポーネントとして分類する。各コンポーネントは、コンポーネントレベルでの再利用性を高めるために、以下の指針に則って粒度を設定する。

- コンポーネントには最小機能のみ持たせる。
- 送受信データには汎用性を持たせる。
- 座標系、単位系は汎用的なものに設定する。
- コンポーネント間の依存性は出来るだけ減らす。
- InPort/OutPort で送受信するデータは出来るだけ小さくする。
- 上記を満たすようにコンポーネントの粒度を設定する。

上記の指針より、本システム構成に必要なコンポーネントを以下のように設定した。(括弧内はコンポーネント名。)

- PA10 マニピュレータコンポーネント (PA10)
- HRP2 アームコンポーネント (HRP2Arm)
- ジョイスティックコンポーネント (Joystick)
- ジョイパッドコンポーネント (Joypad)
- ゲイン調整用スライダコンポーネント (Slider)
- コントローラコンポーネント (Controller)

各コンポーネント間で送受信するデータの形式は表 2 に示す型とした。

実際にコンポーネントを使用して、システムを構築した場合のイメージを図 4, 5 に示す。Joypad を使用してマニピュレータを操作するシステムにおいては、Joypad から出力される手先速度指令値をそのままマニピュレータに入力している。一方、Joystic を使用したシステムにおいては、力-速度変換のためのコントロー

Table 2 使用するコンポーネントとその入出力.

コンポーネント名	入出力力	データ形式
PA10	入力	速度 [mm/s] (×3), 角速度 [rad/s] (×3)
HRP2Arm	入力	速度 [mm/s] (×3), 角速度 [rad/s] (×3)
Joystick	出力	力 [N] (×3), モーメント [Nm] (×3)
Joypad	出力	速度 [mm/s] (×3), 角速度 [rad/s] (×3)
Slider	出力	ゲイン [mm/s/N] (×3), [rad/s/Nm] (×3)
Controller	入力 出力	力 [N] (×3), モーメント [Nm] (×3) 速度 [mm/s] (×3), 角速度 [rad/s] (×3)

ラとそのゲインを調整するスライダからなる速度指令出力系から出る速度指令をマニピュレータに入力する形式となっている。

いずれの場合にも、マニピュレータは手先速度指令値を受け取り、指令値に従って手先速度を制御するだけでよい。

3·3 コンポーネントによる実装

以下に、設計に従って実装したコンポーネントについて述べる。

PA10 コンポーネント

既存の PA10 制御用クラスライブラリを利用し、InPort から受け取ったデータを元に速度制御モードで動作させるコンポーネントを作成した。コンポーネントフレームワークがあるため、既存のコードの再利用により新たなコードを殆んど追加することなくコンポーネントを作成することが出来た。

アクティビティ部分の擬似コードは以下のようになる。既存のクラスライブラリを使用しているため、コンポーネント化に必要なコード量はこのように非常に少なくて済む。

```
RtmRes rtc_active_do()
{
  // InPort から速度指令を読み込む
  m_RefVel = m_RefVelIn.read();

  // PA10 に速度指令を与える
  m_palib->orderVelocity(m_RefVel);

  return RTM_OK;
}
```

`m_RefVelIn` は InPort のインスタンスであり、`read()` により最新の値を変数 `m_RefVel` に読み込んでいる。`m_palib` はマニピュレータ PA10 の制御オブジェクトのインスタンスであり、`orderVelocity()` により速度指令値を与えている。実際には、このアクティビティ部分は ARTLINUX によりリアルタイム実行されており、2 ms の制御周期で実行されている。

HRP2Arm コンポーネント

ヒューマノイド HRP2 制御フレームワークでは、プラグイン形式で作成したユーザプログラムにより、アームの手先速度指令値を与えることが出来る。共有メモリを持たせた手先速度指令値を与えるプラグインを作成し、この共有メモリに対して InPort から受け取った速度指令値を書き込むコンポーネントを作成した。プラグインは既存のものを利用し、新たに共有メモリに値を書き込むコンポーネントを作成するだけで、アームをコンポーネントすることが出来ている。

Joystick

本システムで使用する Joystick はコンポーネントは、力センサを利用した 6 軸出力可能なジョイスティックである。既存の力センサ制御クラスを利用し、取得した力・トルク値を OutPort から出力するコンポーネントを作成した。

Joypad コンポーネント

Joypad を USB 変換ユニットを介して PC に接続し、Linux の汎用 joystick デバイスドライバにより、Joystick として認識させて使用した。Joystick コンポーネントでは Joystick デバイスから得た値を手先速度指令値に変換し OutPort から出力させるコンポーネントとして新たに作成した。

Slider コンポーネント

コントローラのゲインを外部から調整するために調整用のスライダを GUI として作成した。この GUI によるスライダコンポーネントは、GUI 作成が比較的簡単に行える Python (Tkinter: Python の Tk 拡張モジュール) を利用するために、RT ミドルウエアの Python 拡張を利用して作成した。

Controller コンポーネント

力・トルク入力を得て、マニピュレータの手先速度・角速度指令に変換するための制御コンポーネントである。力・トルク–速度・角速度変換は以下の式で表される簡単な制御側を実装した。

$$v_{ref} = K(f_{ref} - f_{ext}) \tag{1}$$

ここで、v_{ref} は手先速度・角速度、K はゲイン、f_{ref} は入力された力・トルク、f_{ext} は手先力センサの力・トルク値である。なお、本システムでは手先力センサは使用しない。

4. RTCLink によるコンポーネントの接続

RTCLink (図 6) は RT コンポーネントのアクティブ化、非アクティブ化をはじめとする制御および、InPort と OutPort 間の接続を管理する GUI である。RTCLink は主として、システム開発時における、低レベル (サーボレベル等) の試験的なコンポーネント間の接続や動作設定を制御するためのツールとして開発された。

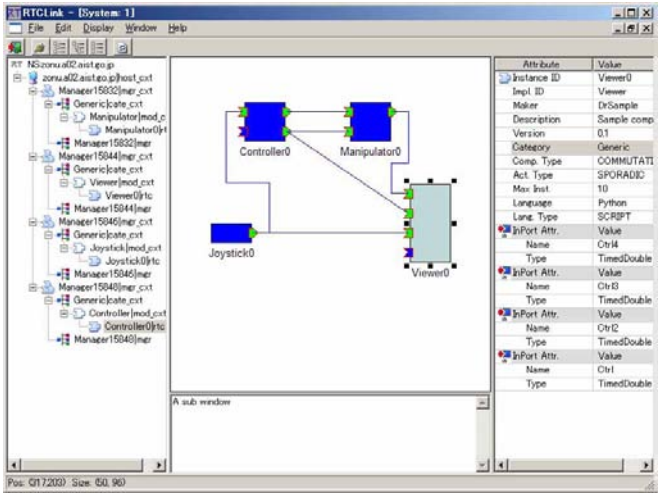

Fig.6 GUI ツール: “RTCLink”.

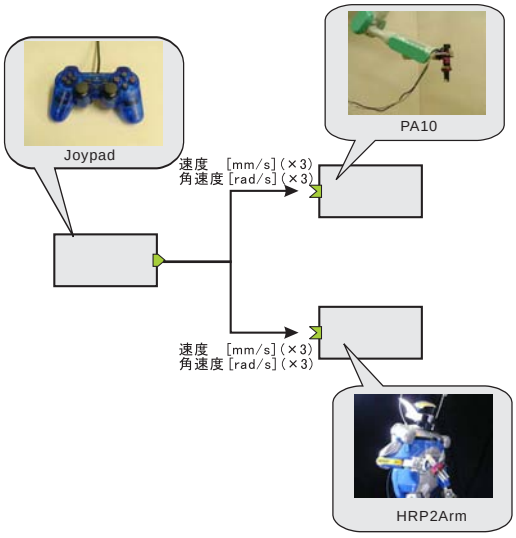

Fig.4 Joypad と PA10 / HRP2Arm によるシステム.

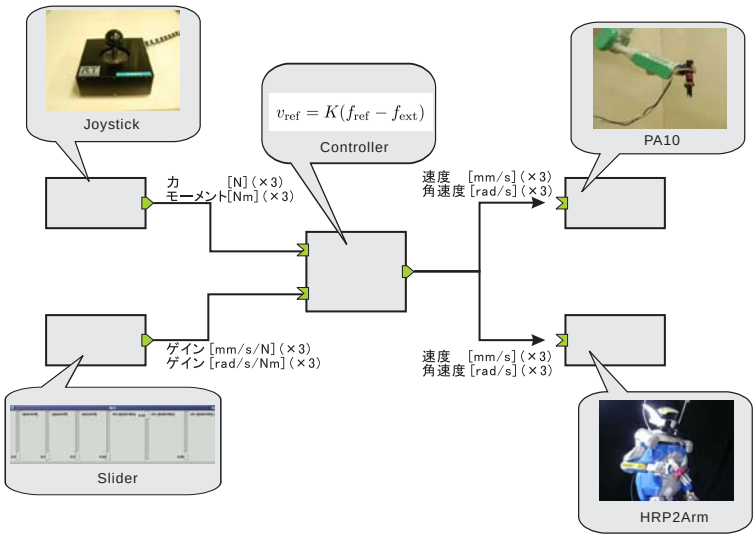

Fig.5 Joystick と PA10 / HRP2Arm によるシステム.

このツールを使用することにより、Joystick/Joypad コンポーネントと PA10/HRP2Arm コンポーネント間の接続を動的に切り替え操作することが出来ることが確認された。

なお、RT コンポーネントの接続情報は XML 形式で保存することが出来、またこの XML ファイルをロードすることにより接続を即座に再現することが出来る。

5. システム構築について

本稿で示したシステム構築例では、図 2 に示したシステム設計フローに従い、「分析」「設計」「実装」の工程に従いシステムを構築した。工程には分かれているものの、各工程における「仕様の抽出」「コンポーネントの設計」に関しては、著者らの経験に大きく依存しており、この段階でこれらを一般化するのは困難である。

しかしながら、「システム設計」工程においては、コンポーネント化するための目安としての指針を示すことが出来た。これは、ひとつには実装フレームワークが RT ミドルウエアとして規定されているため、設計の指針を示しやすかった為である。また、こうした指針は、対象とするシステムを幾つかの類型に分類することでパターン化することも可能であろう。幾つかの典型的なシステム設計パターンを再利用可能な形で蓄積することが出来れば、大規模なシステムを構築する際にも、設計にかかる労力を大いに削減できる可能性がある。さらに、設計に関してパターン化や指針が定まれば、分析においても同様のことができるようになる可能性がある。

上述した分析、設計を多様なレベル (システムの上位〜下位) や多様なフェーズ (仕様〜実装) で行うためには、これらを記述する共通の言語を持つ必要がある。ソフトウエア設計においては UML がその役割を担っており、ロボットシステム設計においても UML[8] あるいは Real-Time UML[9] といった記述法を参考にし、あるいは取り入れる必要があるのではないだろうか。

6. おわりに

本稿では、RT コンポーネントを利用したマニピュレータの遠隔操作システム構築を例に取り、システム構築の系統的な方法の構築に関して議論を行った。「分析」「設計」「実装」の工程に従ったシステム構築方法に基づいて、システムを構築することで、再利用性の高いコンポーネントと柔軟性のあるシステムが得られた。しかしながら、系統的システム構築に必要な「分析」「設計」に関するフレームワークの構築やパターンの獲得にはいたっていない。これらを構築・獲得するためには、より多くのロボットシステムに対して、RT ミドルウエアを適用し、その過程からこれらを抽出する作業が必要となるであろう。また、ここから同時に RT ミドルウエアにおいて必要な機能等も抽出されるものと考えられる。今後、多くの RT システムに対し RT ミドルウエアの適用を行い、機能の充実と共に、系統的システム構築のためのパターン抽出を行ってゆく予定である。

参考文献

1) 「21 世紀におけるロボット社会創造のための技術戦略調査報告書」, (社) 日本機械 工業連合会, (社) 日本ロボット工業会, 2001.
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 1)", 第 21 回 日本ロボット学会学術講演会予稿集, p.1F27, 2003.09
3) 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
4) 安藤 慶昭, 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, "RT 要素のモジュール化および RT コンポーネントの実装", 第 9 回 ロボティクスシンポジア, pp.288-293, 2004.03
5) 安藤 慶昭, 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, "RT 複合コンポーネントおよびリアルタイムコンポーネントの実装-RT ミドルウエアの基本機能に関する研究開発 (その 7)-", 日本機械学会 ロボティクス・メカトロニクス講演会 2004, p.1A1-L1-5, 2004.06
6) 安藤 慶昭, 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, "RT 複合コンポーネントおよびコンポーネントマネージャの実装− RT ミドルウエアの基本機能に関する研究開発（その 8）−", 日本機械学会 ロボティクス・メカトロニクス講演会 2004, p.1C26, 2004.09
7) 北垣 高成, 末廣 尚士, 神徳 徹雄, 尹 祐根, 安藤 慶昭, "RT コンポーネントによるマニピュレータ制御システム構築− RT ミドルウエアの基本機能に関する研究開発（その 5）−", 日本機械学会 ロボティクス・メカトロニクス講演会 2004, p.1A1-L1-6, 2004.06
8) http://www.omg.org/uml/
9) http://realtime.omg.org/