2004

第22回日本ロボット学会学術講演会

講演概要集

2004.9.15-17
会場：岐阜大学

ROBOTICS

The 22nd Annual Conference of The Robotics Society of Japan

RSJ

主催　(社)日本ロボット学会

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 ミドルウェアの基本機能に関する研究開発 (その 8)–

○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)

Implementation of RT composit components and a component manager – R & D of RT Middleware Fundamental Functions (Part 8) –

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

Abstract— We have developed a framework of RT-component which promotes application of Robot Technology (RT) in various field. In robot application, it is indispensable that two or more RT-components synchronize and work in the real time. In this paper, we propose an architecture to realize composit components working in a real-time thread and a component manager that manages components and resources.

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

1. はじめに

ロボット要素技術 (RT) のソフトウエアモジュール化とその再利用性を高める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いる。RT ミドルウエアは、種々のモジュールの再利用を促進し、その様々な組み合わせで、より複雑な機能を実現するシステムを構築し、ロボット技術の新たな応用分野を開拓することを目指している。

RT ミドルウエアは、既存のロボット技術を部品化し再利用を促進するための

- RT コンポーネントフレームワーク、
- RT コンポーネント、
- 標準的に再利用されるソフトウエア部品群、
- ライブラリ群、
- 標準サービス群

などから構成される (図 1)。

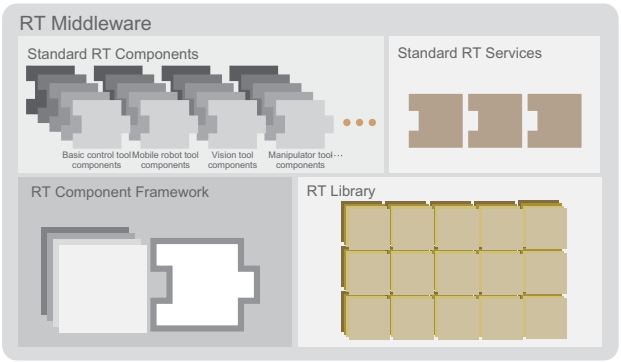

Fig.1 RT ミドルウエア

本稿では、上記のうち、コンポーネントフレームワーク、特にその中の複合コンポーネントのリアルタイム化とその機能をサポートするコンポーネントマネージャについて議論を行う。

2. RT コンポーネント

図 2 に RT コンポーネントのアーキテクチャを示す。RT コンポーネントは分散オブジェクト技術を用いて実装される。RT ミドルウエアは言語、OS 非依存なプラットフォームを指向しているため、現在は CORBA を用いて実装が進められている。

分散オブジェクトに対する RT コンポーネントの特長は、

- 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」を持つ。
- 入出力データストリームの相互接続のための共通なインターフェースを持つ。
- 複数 CPU 間のコンポーネント同期機構を持つ。
- コンポーネント・オブジェクトの管理機構を持つ。

等が挙げられる。

2·1 アクティビティ

通常、分散オブジェクトはオペレーションに対して処理を行いその結果を返すといったパッシブな使用が一般的である。一方ロボット等の制御においては制御部分や処理部分が常に動作しているものが一般的であり、同時に入力データ、出力データは連続したストリームとしてやり取りされるべきものが多い。

2·2 入出力データストリーム

通常の分散オブジェクトでは、データのやり取りは基本的にメソッド呼び出しを介して行われる。これに対して、ロボットの制御における入出力データストリームは、通常メソッドよりもデータ型が重要である。例えば、6 自由度を持つジョイスティックがあり、これが出力する 6 個の double 型データを用いて様々な対象 (マニピュレータ、移動ロボット、ヒューマノイド等) を様々な方式 (位置制御、速度制御等) で制御したい場合を考える。ジョイスティックはあらゆるタイプの制御対象のメソッド (インターフェース) を予め知っていな

ければならない。送り手が 6 個の double 型データを送ることができ、受け手も 6 個の double 型データを処理して動作することができれば十分である。当然、各々に適切な係数を掛けるなどして調整する必要があるが、それはそれぞれのコンポーネント内部での問題である。したがって、RT コンポーネントはデータ型が同じであれば常に接続可能な InPort/OutPort と呼ばれるデータストリームポートを持つ。

2·3 複数 CPU 間コンポーネント同期機構

多くのコンポーネントが協調して動作する場合、協調の度合いの疎密により種々の協調機構が必要となる。疎な連携は入出力データストリームで可能である。一方、密な連携においては、例えばコンポーネント A,B,C のアクティビティが、必ず A→B→C の順で同期的に動作させたいといった場合も考えられる。これを実現するために RT コンポーネントではアクティビティの同期処理機構も提供している。

2·4 コンポーネント管理機構

RT コンポーネントはネットワーク上に分散して配置されるため、コンポーネント自身が他のコンポーネントをネットワーク上から検索し制御する必要がある。RT コンポーネントではこういった機能をコンポーネントマネージャを通して提供する。これにより、現在 CORBA を用いて実装を行っているが、コンポーネント側からはこれらの操作は隠蔽され、コンポーネント開発者はコアロジックにのみ注力できるような機構となっている。そのほか、コンポーネントの動的なロード・アンロード、コンフィギュレーションなどをサポートしており、多言語へのブリッジなどの機構を提供する予定である。

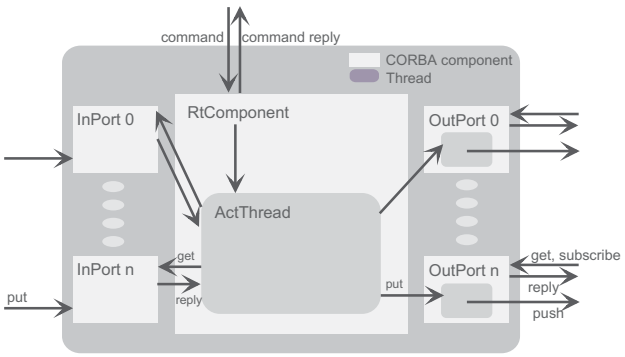

Fig.2 RT コンポーネント

3. RTComponent アーキテクチャ

RT コンポーネントはいくつかのオブジェクトから構成されており、大きく分けると以下の 3 つの部分から成る。

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。処理は周期動作・非周期動作に分けれら、また同時にハードリアルタイム・ソフトリアルタイム、あるいは非リアルタイム処理される。

InPort 他の RT コンポーネントからの出力をを受け取りハンドリングする入力ポート。データのストリームを受け取る必要がある RT コンポーネントはこの InPort を持つ。コアロジックは連続的に送られてくるデータに対して処理を行うため、基本的に周期動作を行うコンポーネントのみがこの InPort を持つ。我々は InPort の動作を数種類提案しており [2]、それぞれの “実装” を用意することで実現する。

OutPort オブジェクト 他の RT コンポーネントへ処理結果のデータストリームを渡す出力ポート。受け取る側のコンポーネントから値を取得する pull 型のデータ出力と、サブスクライブすることにより受け取り側へ能動的にデータを送る push 型の動作がある。

ユーザは任意の入出力ポートを定義し、アクティビティに主たる処理を記述することにより、既存のソフトウエア資源を容易にコンポーネント化することが出来る。

3·1 アクティビティ状態遷移

RT コンポーネントのアクティビティは、`init`, `ready`, `active`, `error`, `terminate` の 5 つの状態を持つ。図 3 にアクティビティ部の状態遷移図 (UML のステートチャート) を示す。

init 初期化状態。ここでコンポーネントの各種初期化を行う。たとえば、センサ、モータ等であればドライバ等の初期化は init 状態で行う。またロジックのみのコンポーネントなども、ここで変数初期化・初期状態の設定を行う。

ready 待機状態。この状態ではすぐにアクティブ状態に移行可能であるが実際には処理を行わない。モータ等ではサーボが ON の状態かつ安全に停止している状態は ready 状態にあたる。

active アクティブ状態。主たる処理を行う状態であり、コンポーネントは他のコンポーネントからのデータを受け取り、処理しあるいは出力することができる。

error エラー状態。何らかのエラーを検出するとこの状態に遷移する。他のコンポーネントとの通信は停止される。

terminate 終了状態。適切な終了処理が行われた後にコンポーネントは停止する。ハードウエアを扱うコンポーネントではこれを安全に停止させリソースを開放する。

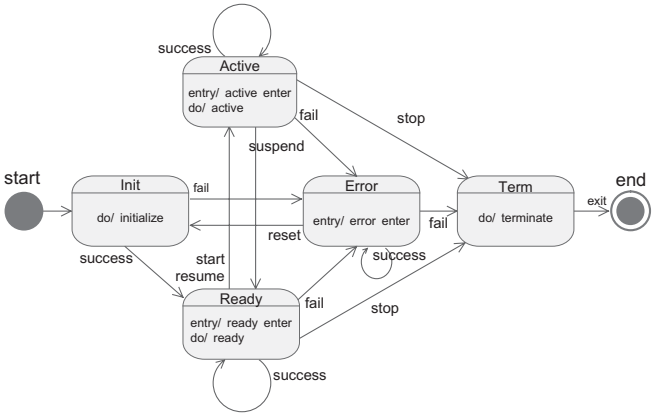

Fig.3 アクティビティのステートチャート

コンポーネント開発者は自分が実装したい対象について上記の状態を当てはめ、それぞれの状態での処理をプログラムに記述することになる。アクティビティは、自らまたは外部からのイベントにより状態遷移を行う。

全てのコンポーネントに同じ状態遷移を持たせることにより、複数のコンポーネントを結合させ同期的に動作させる「複合コンポーネント」(Composit Component) を構成するのが容易になる。すなわち、複数の密接に関係し同期的に動作するコンポーネントの塊をひとつのコンポーネントとして捉えることができるようになる。この複合コンポーネントは、多層に階層化された複雑システムや、複数のコンポーネントをリアルタイム動作させる際に有効である。

3·2 コード例

図 4 に、実際にコンポーネント開発者が C++言語でコンポーネントを作成することを想定したサンプルコードを示す。コンポーネント開発者は自分が作成するコンポーネントをひとつのクラスとして記述する。このクラスは、RTComponent の基底クラス RtcBase を継承し、RtcBase に定義されている純粋仮想関数を実装することで各状態における処理を実現する。また、入出力を扱う InPort/OutPort は InPort/OutPort クラステンプレートによりメンバ変数としてクラス内に保持し利用することができる。

4. 複数のコンポーネントの協調

これまで、上記のコンポーネントフレームワークに基づいて、我々はいくつかのコンポーネントを実際に作成し、実際のロボットを動作させるなどして検証を行ってきた [3]。ロボット用プログラムにおいては、各種処理をリアルタイム動作させる必要が様々な場面で発生する。我々は、いくつかのコンポーネントのアクティビティを ART-Linux を用いてリアルタイム化し動作させ、単一コンポーネントレベルでのリアルタイム化も容易に可能であることを確認した。

しかしながら、コンポーネントの粒度によっては複数のコンポーネントを多数同期させリアルタイム化する必要がある。たとえば、マニピュレータと力センサをそれぞれ個別のコンポーネントとし、これら二つのコンポーネントを用いてマニピュレータの力制御を行おうとする場合には、個々のコンポーネントを単にリアルタイム化するだけでは安定な力制御を行うことは出来ない。RT ミドルウエアは、まったく個別に作成されたコンポーネント同士を組み合わせて一つの RT システムが容易に構築することを目指しており、複数のコンポーネントを同期させてリアルタイムで動作させる必要がある。

```
using namespace RTM;

#include "RtcBase.h"
#include "RtcManager.h"

class SampleComponent
  : public RtcBase
{
 public:
  SampleComponent(RtcManager* manager)
  {
    (中略)
    // InPort OutPort の登録
    RegisterChannel(m_FtSensor);
    RegisterChannel(m_Velocity);
  }
  virtual ~SampleComponent(){};

  // 各状態における処理
  virtual bool init()   {/* 初期化処理 */;}
(中略)
  virtual bool error()  {/* エラー状態処理 */;}
  virtual bool term()   {/* 終了状態処理 */;}

 private:
  InputChannel<Float6> m_FtSensor;
  OutputChannel<Float6> m_Velocity;
};

extern "C" {
// ファクトリ
RtcBase* TestComponentNew(RtcManager* manager)
{
  return new TestComponent(manager);
}
void TestComponentDelete(RtcBase* p)
{
  delete p;
  return;
}
// ロード時のエントリポイント。
void  SampleComponentInit(RtcManager* manager)
{
  // モジュールのプロパティを定義する。
  RtcModuleProperty p;

  p.Name = "SampleComponent";
  p.Category = "SampleComponent";
  p.Vendor = "AIST, Jpana";
  p.Version = "1.0";
  :
  (中略)

  // マネージャにモジュールを登録する。
  manager->RegisterComponent(p,
                             TestComponentNew,
                             TestComponentDelete);
}
}
```

Fig.4 サンプルコード

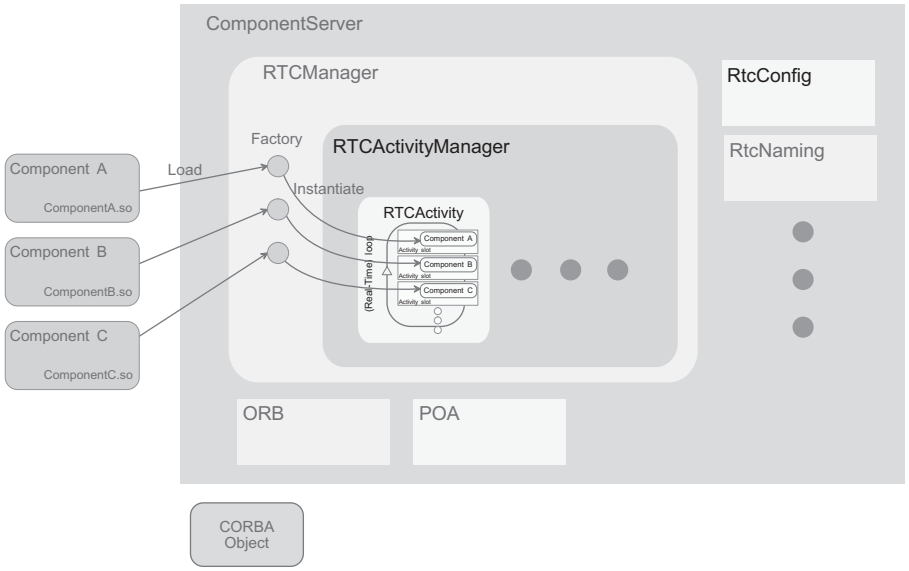

Fig.5 コンポーネントマネージャ

5. コンポーネントマネージャ

複数のコンポーネントのアクティビティを同期的に実行する方法として、我々はコンポーネントとアクティビティを分離し、複数のアクティビティを単一スレッドで実行する方法を提案する。そのための枠組みとして、コンポーネントマネージャを導入する。コンポーネントマネージャの機能で、本稿に関係あるものとしては以下のものが挙げられる。

- コンポーネントオブジェクトの動的な追加削除
- コンポーネントのインスタンス化
- アクティビティスレッドの制御

図 5 に RT コンポーネントマネージャの構成を示す。この例では、Component A, Component B, Component C がそれぞれマネージャにロードされ、同一のスレッドでループ実行されている様子を示す。

Component A, Component B, Component C はロードされると、マネージャによりインスタンス化される。インスタンス化された 3 つのコンポーネントは、マネージャが管理するスレッドリソースから、ひとつのスレッドを与えられ、シーケンシャルに実行される。

与えられたスレッドがリアルタイム実行可能なスレッドであり、Component A, Component B, Component C の実行時間の合計 (+呼び出しオーバーヘッド) が周期時間内であれば、リアルタイム実行可能となる。

マネージャはこのほかに、ローカルなコンフィギュレーションを管理する RtcConfig オブジェクト、コンポーネントの検索をサポートする RtcNaming オブジェクト等を持ち、これらのサービスをコンポーネントに対して提供する。

6. 実験

ここでは、アクティビティスレッドの制御機能として、コンポーネントを一つのリアルタイムスレッドで実行する枠組みのプロトタイプを ART-Linux 上に実装した。

実験では空のアクティビティをスレッド内に順次組み込み、本機構のオーバーヘッドを推定する簡単な実験を行った。1ms のリアルタイムループのスレッドに、順次アクティビティを組み込んだ結果、70 個の空のアクティビティを実行した際の 1000 回あたりの平均ループ時間は 999.713μs、標準偏差は 0.486μs であった。

実際の使用に際しては、各アクティビティの実行時間の合計が、ループ時間内である必要があり、同時実行されるコンポーネント数も数個程度であることが想定される。したがって、モノリシックにプログラムを構成した際との性能差はほとんど無視できるレベルであるものと考えられる。

7. おわりに

本稿では、複数の RT コンポーネントを複合コンポーネントとして利用するためのアーキテクチャについて検討を行った。また、プロトタイプを実装し、同一プロセス内で実行されるコンポーネントが単一リアルタイムスレッド内でリアルタイム動作させることにより有効性を確認した。

今後は、本稿で示した RT コンポーネントおよびコンポーネントマネージャを実際のロボットに対して適用し、実際の実時間制御等での利用可能性を検討する。また、コンポーネント利用の利便性を向上させる種々のサービスや、各種ツールを開発する予定である。同時に、外部に対してこれらをリリースし実際に様々なロボットに対して使ってもらうことでフィードバックを得てゆきたいと考えている。

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] 北垣、末廣、神徳、平井、谷江: “RT ミドルウェア技術基盤の研究開発について -ロボット機能発現のために必要な要素技術開発-”, ロボティックシンポジア予稿集, pp.487-492, 2003.

[3] 安藤, 末廣, 北垣, 神徳, 尹, 「RT 要素のモジュール化および RT コンポーネントの実装」, ロボティックシンポジア予稿集, pp.288-293, 2004.