CONFERENCE DIGEST

ロボティクス・メカトロニクス講演会2010

2010 JSME Conference on Robotics and Mechatronics

ROBOMEC2010 in ASAHIKAWA

ロボティクス・メカトロニクス・フロンティア・ビッグバン

Robotics・Mechatornics・Frontier・Big-Bang

June 13 Sun. - 16 Wed., 2010

Asahikawa TAISETSU Arena

主催　社団法人 日本機械学会 ロボティクス・メカトロニクス部門

The Japan Society of Mechanical Engineers, Robotics and Mechatronics Division

OpenRTM-aist-1.0 における新しいサービスポートの実装

Implementation of new Service Port in the OpenRTM-aist-1.0

○正 安藤 慶昭 (産総研)　栗原 眞二 (産総研)　片見 剛人 (富士ソフト)　坂本 武志 (テクノロジックアート)

Noriaki ANDO, National Institute of Advanced Industrial Science and Technology, n-ando@aist.go.jp
Shinji KURIHARA, National Institute of Advanced Industrial Science and Technology
Tsuyoto KATAMI, Fujisoft Incorporated
Takeshi SAKAMOTO, Technologic Arts, Incorporated

The OMG Robotic Component Specification[1] defines RT-Component model including "Port". RT-Middleware (RTM): OpenRTM-aist has provided "Service Port" for service oriented communication between RT-Components (RTCs). In this paper, new explicit connection mechanism between "provided interfaces" and "required interfaces" is proposed. A new naming rule for interfaces is introduced, and new connection process is implemented. RTSystemEditor, which is a tool for RTCs manipulation, is improved for this new feature.

Key Words: RT-Middleware, RT-Component, service interface

1. はじめに

著者らが開発する RT ミドルウエア (RTM) : OpenRTM-aist は，ロボットの機能要素を RT コンポーネント (RTC) と呼ぶ構成要素としてモジュール化し，それらを組み合わせることにより容易にロボットシステムを構築するためのフレームワークである．RTC のモデルは国際標準化団体 OMG (Object Management Group) において Robotic Technology Component Specification[1] として標準化されている．

標準 [1] では, RTC は他の RTC と相互作用する端点としての UML[2] のコンポーネント定義に準じるポート (Port) が定義されている．ポートは何らかの機能を外部に対して提供する, 提供インターフェース (Provided Interfaces) と, 外部の機能を利用する要求インターフェース (Required Interfaces) のいずれか, また両方を複数持つことができる．

OpenRTM-aist では, データ指向の通信に特化したポートとしてデータポート, その他一般のポートとしてサービスポートをそれぞれフレームワークとして提供している．サービスポートは, ユーザ定義の任意のインターフェースをポートに追加することが可能であり, 開発者は RTC 内部の詳細な機能へのアクセスを提供するために, 任意の数・種類の提供インターフェース (OpenRTM-aist ではプロバイダとも呼ぶ) を追加することができる．また, 他の RTC の提供インターフェースを利用するために, 任意の数・種類の要求インターフェース (OpenRTM-aist ではコンシューマとも呼ぶ) を追加することができる．

以前のサービスポートにおいては, コンパイル時に静的に決められたインターフェース名に従って, 要求インターフェースと提供インターフェースの接続が行われていた．しかしながら, 多様な RTC 作成者が作成する RTC 間では, サービスインターフェースの接続ルールを予め決めることは難しく, ポートの接続時にインターフェース間の関連付けを動的に行う方法が必要とされていた．本稿では, OpenRTM-aist-1.0 で導入された, これらサービスポートのインターフェース接続を詳細に制御する方法について示す．

2. サービスポート

サービスポートはユーザ定義のサービスおよびそれを利用するインターフェースの端点を提供するポートである．RTC は, ポートを介してユーザが定義したサービスインターフェースを持つことができ, これを UML の用語で提供 (Provided) インターフェースと呼ぶ [2]．また, 他の提供インターフェースを利用するためのインタフェースを持つことができ, これを要求 (Required) インターフェースと呼ぶ [2]．

サービスポートは任意の数の提供・要求インターフェースを保持することができる．ポート同士を接続する際に対応する提供インターフェースと要求インターフェースを適切に関連付ける必要がある．

```
// in header
RTC::CorbaPort m_port0;
MyService_impl m_mysvc0;
RTC::CorbaConsumer<YourService> m_cons0;

// in implementation
m_port0.registerProvider("MyService0", "Generic",
                                    m_mysvc0);
m_port0.registerConsumer("YourService0", "Generic",
                            m_cons0 );

m_cons0->your_service_function();

m_cons1->my_service_function();
```

Fig.1 An example of a service port implementation.

コンポーネント内の実装においては，サービスポートは通常図 1 のように利用される． このように，サービスを提供したい提供インターフェースを実装したオブジェクトを `registerProvider()` で登録することにより，他のコンポーネントから利用可能にし, 他方, 要求インターフェースを `registerConsumer()` で登録することにより他のコンポーネントのサービスをコンポーネント内で利用することができる．

RT ミドルウエアの以前のバージョンにおいては, サービスの提供インターフェースと要求インターフェースの接続は, インターフェース型とインスタンス名が同じものが自動的に接続されるというルールで行われていた．

しかしながら, 多数のインターフェースを持つポート同士の接続や, ある種の条件において接続・非接続を選択する必要があるケース等, 上記の単純なルールのみでは対応できない場合がある．また, 異なる開発者が独立に開発することが前提である RTC において, ソースを変更することなくインスタンス名を一致させることは困難である．

2.1 インターフェース接続

ここで, 図 2 に示すインターフェース $\{A_{r1}, A_{r2}, A_{p1}, A_{p2}, A'_{p1}\}$ を持つコンポーネント同士の接続を仮定する．A_{p1} は, 型が A である提供インターフェースの第 1 番目のインスタンスであることを意味する．同様に, A_{r1} は A 型の要求インターフェースの第 1 番目のインスタンスである．また, A' は A を継承した A' 型インタフェースであることを意味する．従って, リスコフの置換原則 [3] に従えば, A'_{p1} と A_{r1} は接続可能であるが, 逆に, A_{p1} と図中には無いが A'_{r1} は接続できない．

図 2 のインターフェースを持つポート同士の例では, イン

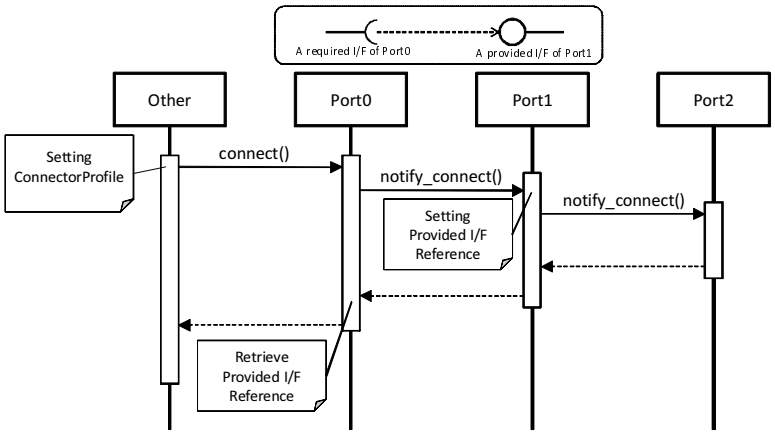

Fig.3 A sequence diagram for Ports connection.

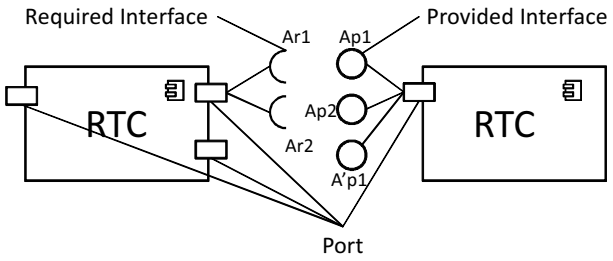

Fig.2 An example of Ports interface connections.

ターフェースは全て接続されるとすると 6 通り, 必ずしも接続する必要がない場合に, 接続しない条件を ϕ とすると, 以下の 13 通りの接続が可能である.

$$
\begin{aligned}
A_{c1} - \phi \quad &: \quad A_{c2} - \{\phi, A_{p1}, A_{p2}, A'_{p1}\} \\
A_{c1} - A_{p1} &: \quad A_{c2} - \{\phi, A_{p2}, A'_{p1}\} \\
A_{c1} - A_{p2} &: \quad A_{c2} - \{\phi, A_{p1}, A'_{p1}\} \\
A_{c1} - A'_{p1} &: \quad A_{c2} - \{\phi, A_{p1}, A_{p2}\}
\end{aligned}
$$

サービスポートはこのような多様な接続を自由に構成できる機能を提供する必要がある.

2.2 接続シーケンス

ポートの接続は, 図 3 に示す UML シーケンス図 [2] に従って行われる. OMG RTC 仕様によれば, 一つの接続には, 複数のポートを関連付けることができる [1].

接続開始時に, 図中 “Other” に相当するツール等が ConnectorProfile と呼ばれる接続に関する構造体に各種データを予めセットしたうえで, 接続の端点となるポート (図中 “Port0”) の `connect()` オペレーションに ConnectorProfile を与えることでシーケンスが開始される. 図 3 では, “Port0” が要求インターフェースを持ち, “Port1” が提供インターフェースを持っている. “Port1” に対する `notify_connect()` 呼び出し時に ConnectorProfile が持つ名前付き値リストである properties メンバーに提供インターフェースの参照のをセットする. セットされた参照は, 破線矢印で表される呼び出しの戻り時に, 要求インターフェースを持つ “Port0” によって ConnectorProfile の properties メンバーから取り出され参照がプロキシ (代理) オブジェクトにセットされ利用される.

OpenRTM-aist では, ConnectorProfile 構造体が持つ名前付き値リスト (properties) に, 提供インターフェースの CORBA オブジェクトリファレンス (IOR: Interoperable Object Reference) と呼ばれる文字列で表記された参照情報を, インターフェースの型名とインスタンス名をキーとし, 下記の様に格納していた.

```
<type_name>.<type_name>: IOR:1280636428603821048937 4....
```

コンシューマは, 自分と同一のインターフェース型名とインスタンス名のキーを ConnectorProfile から探し, プロバイダオブジェクトへの参照を暗黙的に取得していた. しかしながら, 上述のように任意の要求インターフェースに任意の提供インターフェースを対応づけるには, インターフェースの対応関係を明示的に指定する方法を実現する必要がある. 次節ではこの方法について考える.

2.3 インターフェース命名規則

上記のこれまでの方法の問題点の一つは, 要求インターフェースが暗黙的に, 自身に対応する提供インターフェースの参照を, 型名とインスタンス名のみの対応から静的に取得していたことである.

図 2 に示すような, 複雑なインターフェースを持つポート間の接続を詳細に制御するには, まず, インターフェースを区別する方法を与える必要がある. 例えば, 以前の `<type_name>.<type_name>` という形式では容易に衝突する可能性がある. そこで, ポートに付属するインターフェースの指定子のフォーマットを以下のように定める. インターフェースに関するプロパティをそれぞれ以下のように命名する.

`rtc_iname` RTC インスタンス名
`port_name` ポート名
`if_pol` インターフェース極性
`if_tname` インターフェース型名
`if_iname` インターフェースインスタンス名

インターフェースの指定子を以下の文字列名称で指定するものとする.

```
<rtc_iname>.port.<port_name>.<if_pol>.<if_tname>.<if_iname>
```

提供インタフェースのプロパティが以下の場合,

rtc_iname	=	MyComp0
port_name	=	mysvc
if_polarity	=	provided
if_tname	=	echo_interface
if_iname	=	echo_interface2

インターフェース指定子は

```
MyComp0.port.mysvc.provided.echo_interface.echo_interface2
```

のように記述される. また, 同様に要求インターフェースのプロパティが以下の場合,

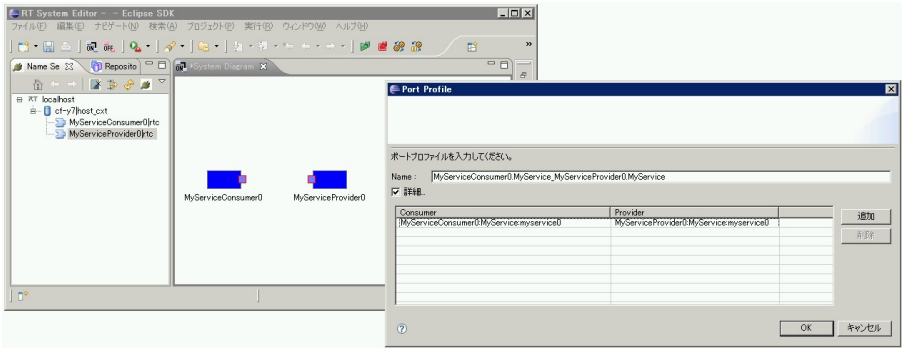

Fig.5 A service-ports connection dialog of RTSE.

rtc_iname	=	YourComp0
port_name	=	yoursvc
if_polarity	=	required
if_tname	=	hoge_interface
if_iname	=	hoge_interface1

インターフェース指定子は,

```
YourComp0.port.mysevc.required.hoge_interface.hoge_inteface1
```

のように記述することができる.

3. 設計と実装

3.1 接続処理

ここでインターフェース記述子を簡単のために `<if_desc0>`, `<if_desc1>`, ... とする. また, ConnectorProfile の properties メンバーの 名前 (key) と 値 (value) を key: value のように記述するものとする.

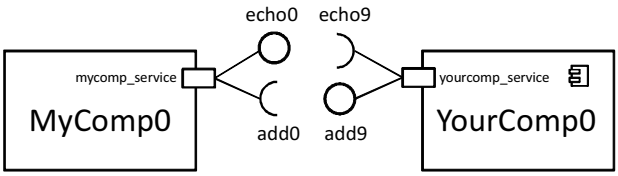

Fig.4 An example service interfaces.

いま, 図 4 に示す 2 つのコンポーネントのサービスポートを接続する場合を考える. すなわち, “MyComp0” RTC は “mycomp_service” ポートに, Echo 型の “echo0” 提供インターフェースと, Add 型の “add0” 要求インターフェースを持つ. 他方, “YourComp0” RTC は “yourcomp_service” ポートに, Echo 型の “echo0” 要求インターフェースと, Add 型の “add0” 提供インターフェースを持つ.

MyComp0 の echo0 と YourComp0 の echo9 , MyComp0 の add0 と YourComp0 の echo9 をそれぞれ対にして接続させるものと仮定する. この場合, ConnectorProfile の properties は以下のように設定する.

```
<add0>: <add9>
<echo9>: <echo0>
```

ただし, それぞれ

```
<add0>:MyComp0.port.mycomp_service.required.add.add0
<add9>:YourComp0.port.yourcomp_service.provided.add.add9
<echo0>:MyComp0.port.mycomp_service.provided.echo.echo0
<echo9>:YourComp0.port.yourcomp_service.required.echo.echo9
```

である.

上述したように, 提供インターフェースを持つポートでは, 自分の提供インターフェース記述子をキーとし, 値にインターフェースの参照を ConnectorProfile の properties メンバに設定する. 接続を構成する RTC 群のインスタンス名は重複しないとすれば, このインターフェース記述子は今接続しようとしているコネクタ群内においては一意であるため, 同じキーは 1 つしか存在しない.

一方, 要求インターフェースを持つポートでは, 自分の要求インターフェース記述子を key とする key-value ペア が存在するかどうか調べ, もし存在すれば, その value に設定された提供インターフェース指定子で指定される参照を, さらに ConnectorProfile の properties から探し, それを要求インターフェースのプロキシに設定する. 要求インターフェースに対して意図的に接続を行わない場合は, 予約文字列 “nil” または “null” を設定することでこれを実現する.

4. RTSystemEditor

OpenRTM-aist では, RT コンポーネントおよびシステムの構築・操作を行うための GUI ツールとして, RTSystemEditor (以降 RTSE) を提供している. RTSE は, オープンソースのソフトウエア開発環境 Eclipse のプラグインとして実装されているため, 他の Eclipse の数多くのプラグインと連携可能である.

RTSE では, コンポーネントのポート間の接続は, エディタ上にアイコン表示された RTC のポートをそれぞれドラッグアンドドロップすることにより行う. 接続の詳細指定を行うための図 5 に示すダイアログを新たに実装した. 接続を構成する際には, 右側の「追加」ボタンを押し, 接続を追加する. 表の左右にはそれぞれ要求インターフェースと提供インターフェース一覧が, それぞれドロップダウンリストとして表示され, 接続したいインターフェースを選択することで接続を構成する.

5. おわりに

本稿では, OpenRTM-aist-1.0 において導入された, 多数のインターフェースを持つ RTC ポート間の複雑な接続を実現する方法を示した. インターフェースの命名規則を導入し, 多数のインターフェースを明確に区別するとともに, 提供インターフェースと要求インターフェースの対応を接続時に指定する方法を実現した. 本稿では, 静的なインターフェースを前提とした接続のみを扱ったが, 提供インターフェース, 要求インターフェースともに, 接続時にインスタンスを動的に生成する接続も考えられる. 今後は, これらの動的インターフェースを扱う方法についても実現する予定である.

文献

[1] Object Management Group, Robotic Technology Component Specification Version 1.0, formal/2008-04-04, 2008
[2] Object Management Group, Unified Modeling Language: Superstructure Version 2.3, formal/2009-02-02, 2009
[3] Barbara Liskov and Jeannette M. Wing, “Family Values: A Behavioral Notion of Subtyping”, Carnegie Mellon University Technical Report: CS-93-187, 1993