OpenRTM-aist による Web サービスサーバ型 RT コンポーネントの開発

○山野辺 夏樹, 安藤 慶昭, 神徳 徹雄（(独）産業技術総合研究所）

A Web Service Server RT-Component on OpenRTM-aist

○ Natsuki YAMANOBE, Noriaki ANDO, and Tetsuo KOTOKU
National Institute of Advanced Industrial Science and Technology (AIST)

Abstract— In this paper, an RT-Component which provides Web services is developed based on OpenRTM-aist-0.4.0. This component acts as a gateway between RT-Middleware and SOAP. As an exmanple, we implement a Web service server RT-Component which returns data input from its InPort according to the request of SOAP client.

1. はじめに

日常生活を支援する次世代のロボットシステムでは，ユーザの多様で高度なニーズに応えるため，様々な機器やロボット間の相互連携や，ネットワークを介したシステム全体としてのサービスの提供が要求される．

ロボットの機能要素をモジュール化して，それらを組み合わせることにより，ロボットシステムを容易に構築するための共通プラットフォームとして，RT ミドルウエアの研究開発が行われている[1]．インタフェースを共通化し RT コンポーネントとしてモジュール化することで，様々な RT 製品の相互利用や，コンポーネントの再利用によるシステム開発効率の向上が見込まれる．一方，通信ネットワークを介した情報サービスに関しては，SOAP を利用した Web サービスが主流となりつつあり，両プロトコルでの相互接続が可能になれば，ロボットシステムの提供するサービスの多様化・充実が期待される．

著者らはこれまでに，Web サービスを利用するクライアント型の RT コンポーネントについては構築し，その作成方法を紹介した[2]．本稿では，SOAP を利用した情報の送受信を可能とする Web サービスサーバ型の RT コンポーネントを作成したので，それについて紹介する．

2. Web サービスサーバ型コンポーネント

Web サービスサーバ型コンポーネントが存在すれば，通信ネットワークを介して，RT コンポーネント化されたロボットシステムや機能部品の情報を取得したり，それらの遠隔操作が可能となる（Fig. 1）．このとき，サーバ型コンポーネントが提供するサービスを利用するクライアント側は，サービス提供先が RT コンポーネントであることを気にする必要はない．WSDL でサービスのインタフェースを定義し，各コンポーネントが配布すれば，WSDL をもとにクライアントプログラムを自由に作成することが可能である．

サーバ型コンポーネントを利用したアプリケーション例としては，見守りサービスが挙げられる．カメラなどの複数の計測機器は RT ミドルウエアで連携して見守りを実現する．サーバ型コンポーネントはいくつかの機器と接続し，各機器のデータや画像を渡すサービスを持てば，ユーザ側は通信ネットワークを介して

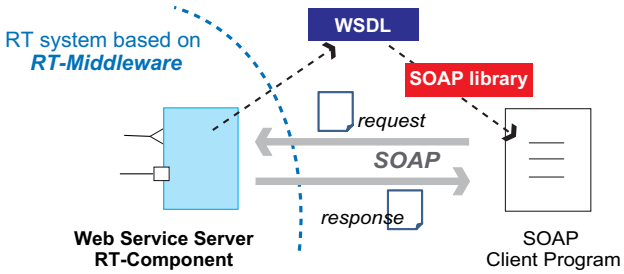

Fig.1 Web Service Server RT-Component.

必要なときに必要な情報を取得することが可能になる．

3. サーバ型コンポーネントの作成方法

本章では，RT ミドルウエアの実装の一つである OpenRTM-aist-0.4.0[3),4)]を用いたサーバ型 RT コンポーネントの作成方法について具体的に説明する．サーバ型コンポーネントは，SOAP のサーバプログラムを RT コンポーネントとしてラッピングすることで構築できる．SOAP サーバ/クライアントを構築するライブラリはさまざま存在するが，本稿では，C/C++向けのフリーの汎用 SOAP ライブラリ gSOAP[5)]を利用することとする．gSOAP では，実行ファイル単独で SOAP サーバの機能を実現できるため，RT コンポーネント化が容易である．また，WSDL の生成や SSL を利用した SOAP 通信機能も持っている．

3·1 SOAP サービスのインタフェース定義

まず，サーバ型コンポーネントで提供するサービスのメソッドおよび使用するデータ型を定義するヘッダファイルを作成する．Web サービスでは XML ベースの SOAP メッセージを送受信し，サーバ/クライアント間の連携が行われる．送受信されるデータは XML Schema 仕様で規定されたデータ型に限られるため，C/C++と XML Schema とのデータ型の対応付けを明記する必要がある．例えば，`double` 型の変数を受け取り，3 倍して解を返すようなサービスを持つ場合，以下のようなヘッダファイルを作成する．

```
typedef double xsd__double;
int ns__triple(xsd__double a, xsd__double &result);
```

2 行目がメソッド定義であり，’__’ で区切られた左側はサービスが定義される名前空間を，右側はメソッド名を表す．また戻り値は参照 & を付けて引数で定義する．

3·2 スケルトンの生成

作成したヘッダファイルから，gSOAP の soapcpp2 ライブラリを用いてスケルトンのソースファイル群を生成する．このとき，SOAP/C++のシリアライザ・デシリアライザ，スケルトンのヘッダ/ソースファイルなどが生成され，サービスのインタフェースを定義した WSDL も生成される．

3·3 RT コンポーネント化

生成したソースファイルを利用して，サーバ型コンポーネントを作成する．SOAP サーバに必要な機能を RT コンポーネントのアクティビティに対応付けると，

- onActivate: サーバの起動
- onExecute: コネクションの受付，サービスの提供
- onDeactivate: サーバの終了処理

となる．rtc-template を利用して生成した RT コンポーネントの実装ファイルにスケルトンのヘッダファイルを読み込み，メンバ変数に SOAP の実行環境を追加 (`struct soap soap;`) して，上記のアクティビティは以下のように実装する．

onActivate

```
soap_init(&soap); //SOAP 実行環境の初期化
SOCKET m=soap_bind(&soap, "host", port, 100);
                    //host の port 番ポートでサーバ起動
```

onExecute

```
SOCKET s=soap_accept(&soap); //コネクションの受付
soap_destroy(&soap); soap_end(&soap);
                              //通信関連のデータ解放
```

onDeactivate

```
soap_destroy(&soap); soap_end(&soap);
soap_done(&soap); //SOAP 実行環境の解放
```

最後に，提供するサービスのメソッドを実装し，RT コンポーネントのソースファイルと生成したスケルトン，gSOAP の提供する SOAP ランタイムライブラリをリンクしてコンパイルすることで，サーバ型 RT コンポーネントが作成される．

4. 実装例

サーバ型コンポーネントの実装例として，InPort に入力されたデータを出力するサービスを持つコンポーネントを作成した．InPort の入力データは今回単純に整数値としたが，計測機器などと接続しそれらのデータを外部に提供するサーバ型 RT コンポーネントの簡易版と考えられる．

サービスの定義ファイルは以下のように作成し，スケルトンのソースを生成する．

```
typedef long xsd__int;
int ns__InPortData(xsd__int &result);
```

rtc-template を用いて，InPort を一つ持つ RT コンポーネントを生成し，アクティブティなどを前章で示したように実装する．ここで，クラスが異なるため SOAP サービス側から RT コンポーネントのポートには直接

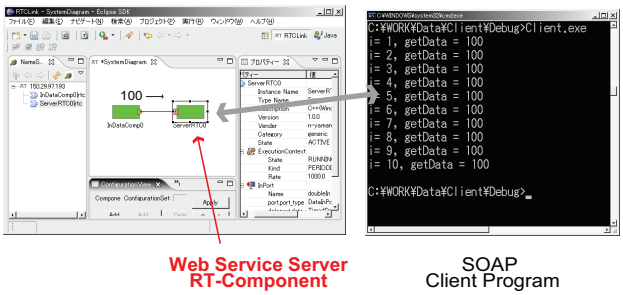

Fig.2 Result of Web Service Server RT-Component.

アクセスすることができないため，以下のように追加する．まず，RT コンポーネントへのポインタを取得するために，ポインタ取得用のメソッドを SOAP サービス側で以下のように実装し，コンポーネントのコンストラクタ内で呼び出されるようにする．

```
ServerRTC *g_RTCptr;
void setRTCptr(ServerRTC *RTCptr){
   g_RTCptr = RTCptr;
}
```

次に，ポートのデータを間接的に取得するためのメソッドをコンポーネント側に以下のように実装し，これを用いて InPort のデータを取得する．

```
public:
long ServerRTC::get_data(){
   m_inIn.read();      //m_inIn InPort のインスタンス
   return m_in.data;  //m_in バッファ
}
```

Fig. 2 に作成したサーバ型コンポーネントの実行結果を示す．データを出力する側のコンポーネントは常に 100 を出力するように実装した．クライアントプログラムは，サーバ型コンポーネントの WSDL をもとにコンポーネントからデータを 10 回取得するように実装し，きちんとデータが取得できていることを確認した．

5. おわりに

本稿では，Web サービスサーバ型の RT コンポーネントを作成し，その実装方法を具体的に説明した．Web サービスサーバ型のコンポーネントにより，ユーザはサービス提供先が RT コンポーネントであることを気にする必要なく，通信ネットワークを介して，RT コンポーネント化されたロボットシステムからの情報取得や遠隔操作が可能となる．なお，今回作成したサーバ型コンポーネントについては公開する予定である．

参考文献

1) N. Ando et al.: “RT-Middleware: Distributed Component Middleware for RT (Robot Technology),” IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 3555–3560, 2005.
2) 山野辺 他: “天気情報取得コンポーネントの開発 –RT ミドルウエアグループとロボットサービスイニシアチブとの連携–”, 日本ロボット学会学術講演会, 2I13, 2007.
3) OpenRTM-aist Official Web Site, http://www.is.aist.go.jp/rt/OpenRTM-aist/
4) 安藤 他: “OMG RTC 標準仕様に準拠した RT ミドルウエアの実装”, ロボティクスメカトロニクス講演会 2007, 1P1-A02, 2007.
5) gSOAP Web Site, http://www.cs.fsu.edu/~engelen/soap.html