2007－DD－59 (3)
2007／1／26

Webサービスにおける エージェントサービスパターンの構築

登川 誉史行 †, 大沢 英一 ††

† 公立はこだて未来大学大学院システム情報科学研究科 †† 公立はこだて未来大学，情報アーキテクチャ学科

電子商取引 (e コマース) において，Web サービスを顧客ユーザが利用する場合，Web サービスが配置されるサーバへリクエストをするリクエストコードが必要となる．しかし，リクエストコードを生成するためには，Web サービス周辺技術の習得，サービス利用方法の把握と Web サービスリクエストコード記述法など，その全てを自らの手で行わなくてはならない．本研究では，ユーザの負荷軽減を目指し，ユーザが行いたい Web サービスのパターン判別をエージェントサービスパターン (ASP) と位置付け，ASP から動的にリクエストコードを生成する手法を提案する．

Building Agent Service Pattern for Web Services

Yoshiyuki Noborikawa † Ei-ichi Osawa ††

† Graduate School of Systems Information Science, Future University-Hakodate

†† School of Systems Information Science,Department of Media Architecture, Future University-Hakodate

In this paper we present Agent Service Pattern (ASP) that generates Web service request codes either in SOAP or REST from WSDL files. The most of recent B2C services depend on specific Web application programs, however, Web services technology that enables more flexible and advanced services have gradually become available at many Web sites. To make use of Web services, users (i.e. customers) need to write service request codes, which requires large knowledge related to Web service technology and related areas. ASP supports a user by means of automatic generation of the Web service request codes from WSDL files which the user specifies.

1 はじめに

近年，情報技術の発展と個人用途 PC の爆発的普及により，主に企業間取引 (BtoB) で行われてきた電子商取引 (e コマース) が，企業対顧客取引 (BtoC) でも盛んに行われるようになってきた．BtoC により，オンラインショッピングやインターネットオークション，インターネット上での株取引など，新たな e コマースの形態が出現した．現在では，BtoC の e コマースは，Amazon.com や Yahoo！オークションをはじめとする Web アプリケーション上での取引が主流となっている．

一方，BtoB の e コマースでは，各々がもつ Web アプリケーションを，インターネットを介して相互運用・連携させる技術である Web サービスが使われるようになってきた．Web サービスとは，SOAP・WSDL・UDDI などの基幹技術によって実現される技術体系の総称である．

Web サービス技術の利便性は，BtoC でも注目される技術となった．実際，上記であげた Amazon.com や Yahoo!なども一般ユーザ向けの Web サービスを提供している．しかし，企業側が提供する Web サービスをユーザが利用するためには，知識の習得やコード生成などをユーザに課すという問題がある．

また，これまでの Web サービスの方式そのものに関する問題もある．Web サービスは従来の SOAP・WSDL・UDDI を用いるのが主流であったが，REST と呼ばれる，シンプルに Web サービ

スを実現できるアーキテクチャスタイルが見直されている．実際，Web サービスを提供している企業は，SOAP 方式と REST 方式の両方をサポートしている．Amazon Web Service(AWS) では [7]，REST と SOAP の API を提供しており，API を用いることで，両方式で Web サービスを利用することができる．

しかし，Yahoo!や Google など，どちらかの方式でのみ Web サービスを提供している企業もある．よって，SOAP 方式の Web サービスが，REST 方式の Web サービスを状況に応じて使い分ける方法が望まれている．

そこで，本研究では，BtoC の e コマースにおける，Web サービスユーザの負荷軽減を目指し，エージェントサービスパターン (ASP) を用い，動的に SOAP，REST に準拠した Web サービスコードを生成する手法を提案する．ASP とは，ユーザが要求したい Web サービスのインタフェースが記述されている WSDL ファイルから，サービスのパターンを決定するモジュールのことである．ASP を用いることによって，ユーザが提示した WSDL に柔軟に対応し，効率よく Web サービスリクエストコードを生成することが可能となる．

以下，本論文では，2 章で SOAP/WSDL と REST について紹介し，3 章で関連研究を示す．4 章では，本研究で構築した ASP を用いた Web サービスリクエストコード生成システムについて述べ，5 章で実験評価を行う．6 章で今後の展望とまとめについて述べる．

2 SOAP/WSDL と REST

本章では，SOAP および REST について説明し，両者の違いについて述べる．

2.1 SOAP

SOAP(Simple Object Access Protocol) は，ユーザとプロバイダ (Web サービス提供者) 間で Web サービスを利用するためのプロトコルである [1]．SOAP は XML ベースの記述仕様を持ち，どのようなトランスポートプロトコル (HTTP や SMTP など) で Web サービスを利用するかには依存しない．SOAP を用いた Web サービスで，主に利用されるのは RPC(Remote Procedure Call) である．RPC とは，リモートにあるコンピュータのメソッドを，ローカルにあるコンピュータ内に呼び出すことである．また，別の利用法として，XML 文書などのメッセージ交換がある．さらに，SOAP は画像などのバイナリデータを MIME(Multipurpose Internet Mail Extensions) 形式で添付することが可能である．

2.2 WSDL

WSDL(Web Services Description Language) は，W3C で定義された，Web サービスインタフェースを記述する言語である．Web サービスがどこにあり，どのようにして使うのかといった，インタフェースが記述されている．現在，WSDL1.1 が多くの Web サービスで利用されている．

WSDL のデータ構造を Fig. 1 に示す．definition 要素をルート要素とし，子要素として types 要素，message 要素，portType 要素，binding 要素，service 要素の 5 つの要素がある．さらに portType 要素と binding 要素で定義されている operation 要素，service 要素の子要素である port 要素がある．それぞれの要素は互いに参照し合うことで，1 つのサービスを記述している．

WSDL は，拡張要素を追加することで，SOAP や HTTP を扱うことが可能となる (詳細は 4.1.1 節で述べる)．

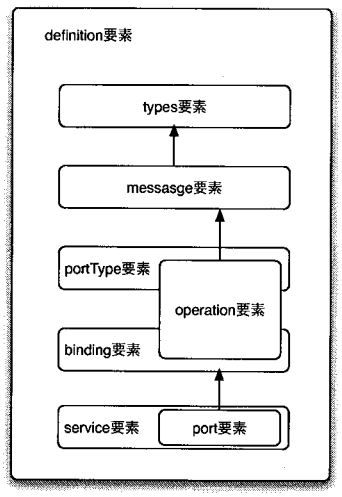

Fig. 1 WSDL の構造

2.3 REST

REST(Representational State Transfer) は，Roy Fielding が提唱したネットワークアーキテクチャスタイルである [2]．当初は，アーキテクチャとしての原則や制約を指す用語であった (狭義の意味)．しかし，HTTP で XML をやり取りする，Web サービスのプロトコルのような意味として使われるようになった (広義の意味)．REST の特徴は，HTTP を介してリソース (情報) を URI(Uniform Resource Identifier) により，参照できるということである．リソースは，グローバルな識別子 (URI) を持ち，リソースを取得したいユーザは HTTP を使う．HTTP は，GET，PUT，DELETE，POST，GET という 4 つのメソッドを使うことができる．各メソッドの説明を以下に示す．

- GET：リソースを取得する
- PUT：リソースを更新する
- DELETE：リソースを削除する
- POST：リソースを作成する

2.4 SOAP と REST

Fig2. 2 は，SOAP と REST の違いを表している．図の左端に並んでいる内部が空白の四角形は，ブラウザなどの情報やサービスを利用するユーザを表している．図の右端にある四角形は，情報やサービスを表す．REST では，URI と各情報やサービスが，1 対 1 で結ばれている．例えば，HTTP GET で URI 1 を参照したときは，entrylist が得られる．

一方，SOAP では，1 つの URI に対して，メソッドを用いて情報やサービスを取得する．例えば，HTTP POST で URI 1 を指定し，メソッド getEntryList() を Endpoint で呼び出すことで，entrylist が得られる．

3 関連研究

本章では，当該分野のこれまでの研究の状況を述べる．

石崎ら [4] は，過去に合成した BPEL を再利用し，合成の効率化を図ることで，BPEL 合成のプランニングコストや処理時間を抑える方法を提案した．しかし，この研究では，Web サービス要求のためのコードを生成するところまでは検討されていない．本研究においては，WSDL ファイルから，サービス要求のためのコードを動的に生成することができる．

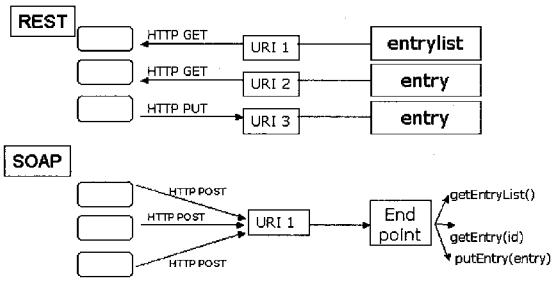

Fig. 2 REST と SOAP の違い [3] より転載]

Matthew ら [5] は，WSIF API を用いることで，特定のバインディングに依存せず，WSDL から直接 Web サービスを要求可能にするコード生成方法を提案した．しかし，REST に対応したコード生成はサポートしていない．本研究においては，REST，SOAP に対応したコードを生成することができる．

4 ASP を用いた Web サービスリクエストコード生成システム

本研究では，ASP を用いて，ユーザが行う Web サービスのパターンを判断し，それに伴い自動的に Web サービスリクエストコードを生成するシステムを構築する．本章では，はじめに ASP について説明し，次にシステム概要，最後にコード生成エージェントの詳細について述べる．

4.1 ASP(エージェントサービスパターン)

エージェントサービスパターン (ASP) とは，ユーザが要求するサービスのパターンをもとに，エージェントが生成するコードパターンを判断するモジュールである．本研究で構築するシステムでは，エージェントはユーザが選択する WSDL ファイルから，ASP 判断アルゴリズムによって生成するコードパターンを決める．ASP を用いることによって，ユーザが要求するサービスに柔軟に対応

したWebサービスリクエストコードを作成することが可能となる．

4.1.1 WSDLの拡張

WSDL1.1では，SOAPとHTTPへの対応として，SOAPBindingとHTTP GET& POST Bindingの仕様が提案されている．SOAPBindingには，以下の拡張要素がある．

- soap:binding：SOAP形式にバインドされることを示す．SOAPを使用する時は，この要素が存在しなければならない
- soap:operation：どのような操作を行うかを示す
- soap:body：SOAPメッセージのBody要素を示す
- soap:header：SOAPエンベロープのHeader要素を示す
- soap:address：SOAPにバインドするポートを示す

HTTP GET& POST Bindingには，以下の拡張要素がある．

- http:address：利用するポートのURIアドレスを示す
- http:binding：HTTP形式にバインドされることを示す
- http:operation：操作の相対URIを示す

これら拡張要素の中で，注目すべきは，soap:binding，soap:addressそしてhttp:bindingである．この要素はWSDL文書中で，階層がもっとも浅い子要素であるからである．つまり，WSDL文書中に，3つの要素のどれかが記述されていれば，その要素を元に，その文書がSOAP拡張であるか，HTTP拡張であるかの判別は可能である．ASPはこの3つの要素から，コードパターンを判別する．

4.1.2 ASPアルゴリズム

アルゴリズム1に，ASPアルゴリズムを示す．このアルゴリズムは，与えられたWSDLから，saop:binding要素，soap:address要素，そしてhttpbinding要素が存在するかを判定し，HTTP_ BINDING(REST)パターンか，SOAP_ BINING(SOAP)パターンかを決定する．

このアルゴリズムでは，メソッドとして *READ*(), *GET*(), *SET*() を用いる．*READ*(*wsdlfile*) は，引数で指定した *wsdlfile* から文書を読み込む．*GET*(*Q*, *a*) は，要素 *Q* から子要素 *a* を探索し，子要素 *a* 返す．*SET*(*PATTERN*) は，*PATTERN* を決定パターンとして設定する．

このASPアルゴリズムにより，サービスパターンを決定し，パターンに沿ったコード生成が可能になる．

アルゴリズム1　ASPアルゴリズム

```
1  procedure Agent Service Pattern (wsdl)
2  begin
3    Def = READ(wsdl);
4    binding = GET(Def, binding);
5    while bindingHasNext ≠ null
6      binding = nextBinding;
7      soapBinding = GET(binding, soapBinding);
8      httpBinding = GET(binding, httpBinding);
9      if soapBinding ≠ null
10       SET(SOAP_BINDING);
11     end
12     else if httpBinding ≠ null
13       SET(HTTP_BINDING);
14     end
15   end
16   service = GET(Def, service);
17   while serviceHasNext ≠ null
18     service = nextService;
19     port = GET(service, port);
20     while portHasNext ≠ null
21       port = nextPort;
22       soapAddress = GET(port, soapAddress);
23       if soapAddress ≠ null
24         SET(SOAP_BINDING);
25       end
26     end
27   end
28 end
```

4.2 システム概要

本研究で提案する ASP を用いた，動的 Web サービスリクエストコード生成システムの概要を Fig. 3 に示した．以下では各要素について説明する．

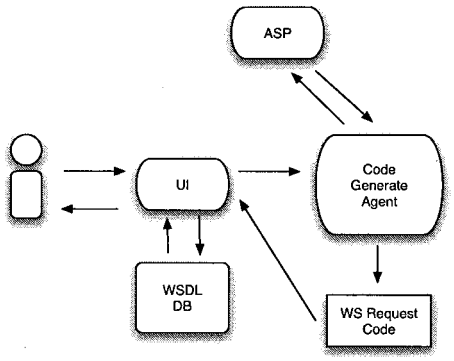

Fig. 3　システム概要

4.2.1 ASP(エージェントサービスパターン)

このモジュールでは，コード生成エージェントから取得した WSDL ファイルの要素から，ASP を用いた判断アルゴリズムによって，生成するコードパターンを決める．

4.2.2 UI(ユーザインタフェース)

本システムを利用する際のインタフェースとなる GUI で構築したモジュールである．ユーザは WSDL データベースから利用するサービスを選択するだけで，Web サービスコードを生成することができる．ユーザが UI を通して制御できるのは，WSDL データベースからの WSDL ファイル選択・決定，選択した WSDL ファイルから Web サービスコードの生成である．

4.2.3 WSDL データベース

このデータベースには利用可能な WSDL ファイルが格納されている．ユーザはこのデータベースからどのサービスを利用するかを選択する．

4.2.4 Code Generate Agent(コード生成エージェント)

コード生成エージェントは，管理モジュール，SOAP 生成モジュール，REST 生成モジュールを持つ．インタフェースモジュールは UI から，ユーザが選択したサービスの WSDL ファイルを受け取り，ASP にサービス内容を渡す．ASP から返ってきたサービスパターンに沿って，SOAP または REST 生成モジュールに WSDL ファイルを渡し，Web サービスリクエストコードを生成する．

4.3 Code Generate Agent(コード生成エージェント) の詳細

コード生成エージェントは管理モジュール，SOAP 生成モジュール，REST 生成モジュールにより構成されている (Fig. 4)．

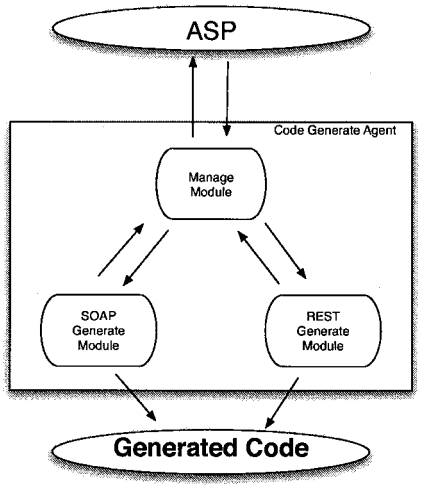

Fig. 4　コード生成エージェントの詳細図

管理エージェントは，UI から WSDL ファイルを取得し，ASP へ渡し，ASP の判断アルゴリズムによって選択されたコードパターン (SOAP 型か REST 型) を得る．得られたコードパターンから，SOAP 生成モジュールか REST 生成モジュールへ WSDL ファイルを渡す．

SOAP 生成モジュールおよび REST 生成モジュールは与えられた WSDL ファイルから，それぞれのパターンの Web サービスコードを生成する．

SOAP 生成モジュールでは，与えられた WSDL ファイルを AXIS ミドルウェアの WSDL2Java を用いて，SOAP 通信を行うための Web サービスプロキシコードを生成する．具体的には，Table1 で示す対応に沿って，Java クラス (Web サービスプロキシコード) が生成される．

Table 1　WSDL2Java の対応表

WSDL	WSDL2Java
Service 要素	<Service 要素名>.java
Service 要素	<Service 要素名>Locator.java
Port 要素	<Port 要素名>.java
Binding 要素	<Binding 要素名>.java
Binding 要素	<Binding 要素名>Stub.java

REST 生成モジュールでは，与えられた WSDL ファイルの http:binding 要素の verb 属性の値を抽出する (HTTP GET/POST などを行うメソッド)．また，http:address 要素の location 属性の値から，URI を抽出する．http:operation 要素の location 属性から値を抽出する．得られた verb 属性値，http:address 要素の location 属性値，http:operation 要素の location 属性値から REST パターンのリクエストコードが生成される．

5　実験・評価

本研究で提案した，ASP を用いたコード生成システムを Java で実装し，システムの動作・性能を評価するための実験を行った．まず，SOAP-RPC を行う WSDL ファイルと，HTTP-POST を行う WSDL ファイルを用意し，実際に本システムを用いて，コードを生成させる実験を行った．次に，本システムの性能を評価するために，メモリ使用量・処理時間の測定実験を行った．

5.1　実験環境

実験は，以下の環境下で行った．

- PC : PowerMac G5
- CPU : 2.5GHz PowerPC G5
- OS : MacOSX10.4.8
- メモリ : 2.5GB DDR SDRAM

5.2　システム動作実験

本実験では，用意した WSDL ファイルから，記述されているバインディングに，適したコード生成が可能かを評価する．

5.2.1　実験設定

SOAP-RPC を行う WSDL ファイル (SOAP-WSDL) として，Fig. 5 に示すものを，HTTP-POST を行う WSDL ファイル (REST-WSDL) として Fig. 6 を用意した．この WSDL が記述しているサービスは，AXIS2 1.1[8] のサンプルに入っている．このサービスは，現在の AXIS のバージョン情報を取得するサービスである．この実験では，本システムを用い，SOAP-WSDL から Web サービスプロクシコードの生成と，WSDL-REST から REST リクエストコードの生成を行わせた．

```
<wsdl:definitions xmlns:ns1="http://org.apache.axis2/xsd"
・・・中略・・・
targetNamespace="http://org.apache.axis2/">

<wsdl:types>
        ・・・中略・・・
</wsdl:types>

<wsdl:message name="getVersionRequestMessage">
        ・・・中略・・・
</wsdl:message>

<wsdl:portType name="versionPortType">
        ・・・中略・・・
</wsdl:portType>

<wsdl:binding name="versionBinding" type="tns:versionPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document" />
        <wsdl:operation name="getVersion">
                <soap:operation soapAction="getVersion" style="document" />
                ・・・中略・・・
        </wsdl:operation>
</wsdl:binding>

<wsdl:service name="version">
        <wsdl:port name="versionPortType"
                binding="tns:versionBinding">
                <soap:address
                location="http://localhost:8080/axis2/services/version" />
        </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Fig. 5　SOAP-WSDL

5.2.2　結果

実験の結果，本システムに SOAP-WSDL を与えた場合は，Table　2 に示した Web サービスプロクシコードを生成した．この結果から，システム内において，WSDL ファイルが適切に WSDL2Java へと渡されていることがわかる．また，REST-WSDL を与えた場合は，Table 3 に示した REST リクエストコードを生成した．この結果より，与えられた WSDL から，http:binding 要素の verb 属性値，http:address 要素の location 属性値，http:operation 要素の location 属性を抽出し，REST パターンのリクエストコードが生成されたことがわかる．

```
<wsdl:definitions xmlns:nsl="http://org.apache.axis2/xsd"
・・・中略・・・
targetNamespace="http://org.apache.axis2/">

<wsdl:types>
        ・・・中略・・・
</wsdl:types>

<wsdl:message name="getVersionRequestMessage">
        ・・・中略・・・
</wsdl:message>

<wsdl:portType name="versionPortType">
        <wsdl:operation name="getVersion">
                <wsdl:input message="tns:getVersionRequestMessage" />
                <wsdl:output message="tns:getVersionResponseMessage" />
        </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="versionBinding" type="tns:versionPort">
        <http:binding verb="POST" />
        <wsdl:operation name="getVersion">
                <http:operation location="getVersion" />
                        ・・・中略・・・
        </wsdl:operation>
</wsdl:binding>

<wsdl:service name="version">
        <wsdl:port name="versionPortType"
                binding="tns:versionBinding">
                <http:address
                        location="http://localhost:8080/axis2/rest/services/" />
        </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Fig. 6　REST-WSDL

5.3　性能評価実験

本実験では，用意したWSDLファイルからコード生成を行う際の，メモリ使用量および処理時間の測定実験を行った．

5.3.1　実験設定

処理するWSDLファイルのサイズを，SOAP-WSDLファイル(1.73KByte)，REST-WSDLファイル(1.54KByte)に設定[1]し，各ファイルについて，コード生成に費やすメモリ使用量・処理時間を測定した．

メモリ使用量測定では，WSDLファイルを10回処理し，その間のメモリ使用量を測定した．処理時間測定では，WSDLファイルを1000回処理させ，コード生成までの時間を測定した．

5.3.2　結果

メモリ使用量についての結果をFig 7に示した．SOAP-WSDL，REST-WSDLのメモリ使用量はほぼ同一の値となった．

処理時間の結果は，1000回処理時のSOAP-WSDL，REST-WSDLの差は約10msしかなかった．

Table 2　生成したWebサービスプロクシコード

GetVersion.java
Version.java
VersionBinding.java
VersionLocator.java
VersionPortType.java

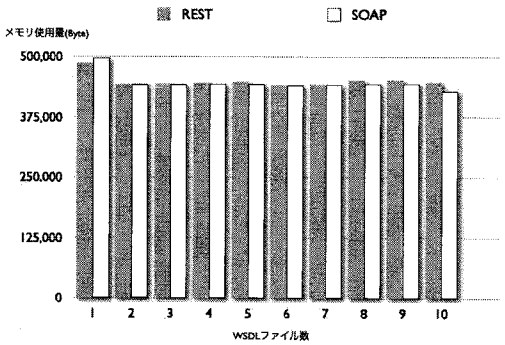

Fig. 7　メモリ使用量の測定結果

Table 3　生成したRESTリクエストコード

Port	versionPortType
Verb	POST
URL	http://localhost:8080/axis2/rest/services/getVersion

[1] WSDLの記述の特性上，SOAPBindingとHTTPBindingでは，記述量に差が生じる

5.4 考察

SOAP-WSDLファイルからの，Webサービスプロクシコード生成と，REST-WSDLファイルからの，RESTリクエストコード生成では，問題なくWSDLに記述されている通りの各コードを生成することができた．特に，RESTリクエストコード生成器は，独自の実装であるが，WSDLに記述されているPort数に応じたリクエストコードが生成ができる．

性能評価として，SOAP-WSDL処理，REST-WSDL処理の両者を比較してみると，システムのメモリ使用量に関してしては，ほぼ同等の性能を持つことがわかった．処理時間に関しても，最終的に10msの差しかなかった．よって，SOAP-WSDLとREST-WSDLのシステム処理能力はほぼ同等と考える．

6 今後の展望とまとめ

本論文では，BtoCのeコマースにおける，Webサービスユーザの負荷軽減を目指し，エージェントサービスパターン(ASP)を用い，動的にSOAP，RESTに準拠したWebサービスコードを生成する手法を提案した．ASPを用いることによって，ユーザが提示したWSDLに柔軟に対応し，効率よくWebサービスコードを生成することが可能となる．SOAP，REST方式が記述されたWSDLを使用した実験では，システムの動作確認，性能評価を行った．結果は，正常にコード生成に成功し，性能に関してはSOAP処理，REST処理，ともに同等のパフォーマンスを持っていることがわかった．

本論文では,ASPを用いたWebサービスリクエストコード生成システムの実装方法について述べた．我々は，ASPに全てのWSDLBinding拡張の判断機能を持たせることで，さらなるユーザ負荷軽減が可能になると考える．本論文で述べてきたように，Webサービスを一般ユーザが利用するためには，ユーザに多大な負荷を課してしまうという問題がある．ASPに，上記のような機能が実現できれば，システムがユーザとインタラクションをとりながら，ユーザの要望に沿ったWebサービスリクエストコードを生成できるようになる．

参考文献

1) W3C, "Simple Object Access Protocol(SOAP)1.1", http://www.w3c.org/2000/xp/Group

2) Roy Fielding, "Principled Design of the Modern Web Architecture", ACM Transactions on internet Technology, Vol. 2, No. 2, May 2002, Pages 115-150.

3) 山本陽平, "REST入門"発表資料, 画像電子学会VMA研究会, November 2005.

4) 石崎康太, 大沢英一, "BPELとOWL-Sを用いたWebサービスの自動合成に関する研究", 第68回情報処理学会全国大会論文集, 2006.

5) Matthew J. Duftler, Nirmal K. Mukhi, Aleksander Slominski and Sanjiva Weerawarana, "Web Services Invocation Framework (WSIF)", Proceedings of the OOPSLA Workshop on Object-Oriented Web Services, 2001.

6) "Amazon.com", http:/www.amazon.co.jp

7) "Amazon Web Service", http://www.amazon.com/gp/browse.html?node=3435361

8) "Apache Axis Project" , http://ws.apache.org/axis2/