マルチメディア通信と分散処理　109－17
電子化知的財産・社会基盤　17－17
（２００２．９．１３）

Java と.NET における Web サービスの構築と相互利用実証

浅田　孝利　　野澤　寛将　　長橋　和哉　　栗原　潤　　小泉　寿男
東京電機大学大学院理工学研究科

Web サービス・システムは，インターネット上で公開されたサービスを連携させる新しい情報システムアーキテクチャと言える．Web サービス・システムの代表的なものとして，Java プラットフォームベースのものと.NET プラットフォームベースのものが挙げられる．しかし，各プラットフォーム間でサービスの相互利用性が確保できないと、利用可能なサービスが限定され，Web サービスを導入するメリットが得られなくなってしまう．本稿では，Java プラットフォームおよび.NET プラットフォームでの Web サービス構築法，利用法について述べ，各プラットフォーム間での相互利用性について実証結果を論じる．また XML ベースの SOAP のレスポンスタイムおよびサービスの粒度に関して，測定結果を述べる．

Implementation of Web Services of Java Technology and .NET Technology and its interoperability

Takatoshi Asada　　Hironobu Nozawa　　Kazuya Nagahashi
Jun Kurihara　　Hisao Koizumi
Department of Computers and Systems, Graduate School of Tokyo Denki University

Web Services architecture is viewed as an innovative information system architecture which provides a standard means of communication among services on the Web. There are two typical Web Services architectures, one is based on Java platform and the other is based on .NET platform. If the interoperability of services between those platforms is not secured, the services that can be used will be limited and the merit of introducing Web Services will be no longer obtained. In this paper, we discuss the method of constructing and utilizing Web Services both on the Java platform and the .NET platform based on the experimental results of interoperability between both platforms, and also we describe the result of measurement of the response time of SOAP in connection with the granularity of services.

1.　はじめに

現在の Web サイトは，利用者である人間がアクセスし，情報を入手・閲覧するものであるが，Web サービスはコンピュータシステムがアクセスするものであり，インターネットを利用した企業間システムの連携、取引に大きな影響を与える可能性がある[1]．Web サービスについては，いくつかの技術解説書も刊行されており[1][4][5]，プロトタイプシステム開発の段階から実システム構築の段階へ進みつつある．

Web サービスは XML を使って RPC やメッセージングの機構を実現するための仕様である

SOAP，サービスを登録・検索するための仕組みである UDDI，サービスのインターフェース情報を記述する WSDL を基盤技術とするものである[2]．このような Web サービスの技術は，情報システムの視点からみるとインターネットをベースとする情報システムアーキテクチャの構成とその構築技術である．また Web サービスは，ソフトウェア工学の面からみてインターネット上に存在する機能の再利用するという実用的な新しい概念である．

筆者らは，Web サービス構築の効果的な技法の実現を目指している．そのためにまず，大学の研究室環境下で Web サービスを構築し，具体的な評価を行い，従来からの Web コンピューティング技術の延長として Web サービス・システムの動作確認をする必要がある．そのための第１ステップとして，3 層からなる Web コンピューティング構成上に EC ビジネスモデルの１つとして部品調達支援システムのプロトタイプを構築し，その機能の一部を Web サービスによって再構築し，評価を行った[3]．第２ステップは本稿で述べる Java 環境下での Web サービスの構築と.NET 環境での構築および相互利用の評価である．Web サービス実装環境として The Mind Electric 社の GLUE，Microsoft 社の.NET を使用している[4][5]．Web サービスの特徴として，プラットフォーム非依存があるが，多種多様な開発ツール同士での相互利用性というものは完全に保証されているとは言えない．また Web サービスの通信プロトコルである SOAP も JavaRMI や CORBA に比べて，XML の解析が必要であるため，パフォーマンスの低下という問題が挙げられる[6][7]．

本稿では，Java/.NET プラットフォーム上で簡単な Web サービスを構築し，その相互利用性について実証し，その結果を述べる．また SOAP のパフォーマンスについては，ローカル呼び出し及びインターネットを介した呼び出しをした場合のレスポンスタイムを測定し，その結果を述べる．

2. Web サービス構築と動作確認

研究室内での Web サービス動作環境を図１に示す．

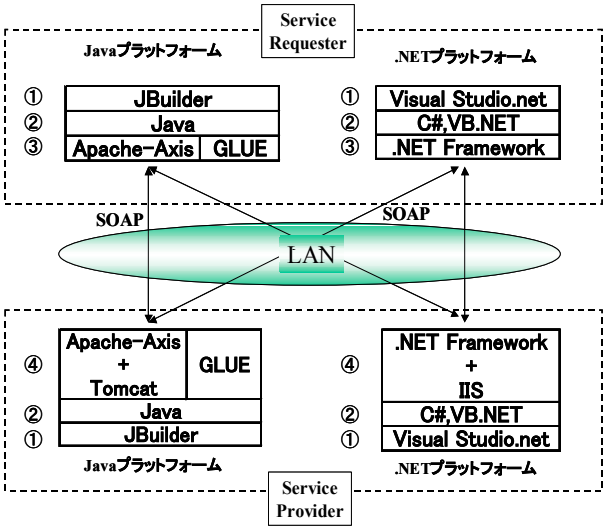

図１　研究室内動作環境

図 1 の①は開発ツール，②は開発言語，③は SOAP 実装，④は③＋サーバ機能を示している．この環境下のもと，Java および.NET プラットフォーム上での Web サービスの構築法および利用法を述べる．

構築するサービス内容としては，単純型データを結果とする「四則演算サービス」，複合型データ（配列，構造体）を結果とする「郵便番号検索サービス」を挙げる．

2.1　構築するサービス内容

(1) 単純型データのやり取り

四則演算 Web サービスは引数(double 型)を２つ与え和・差・積・商を double 型で返すサービスである．

(2) 複合型データのやり取り（配列）

郵便番号検索サービスは以下の３つのメソッドを提供する．

- String 型７桁の郵便番号から String 型住所を取得
- String 型住所から String 型 7 桁の郵便番号を取得
- String 型郵便番号の一部から String 配列型住所を取得

(3) 複合型データのやり取り（構造体）

郵便番号から以下に示す Address 型を取得

```
public class Address｛
  public string Ken;//都道府県名
  public string shiku;//市町村名
  public string Choson;//町域名
  public string zipCode;//郵便番号｝
```

リスト１　Address 型

2.2 Java プラットフォームでの構築

(1) 構築環境

アプリケーションサーバに GLUE Standard Edition2.3.1，開発ツールに JBuilder6 Personal Edition を使用し，開発言語は Java で構築した.

(2) サービス公開までの手順

① 公開するサービスのクラスファイルを作成する.

② 公開するクラスの指定する commands.xml を作成する

```
<commands>
  <invoke>
  electric.registry.Registry.publishInstance
  (“urn:サービス名”,”クラス名”)
  </invoke>
</commands>
```

リスト２　commands..xml

③ GLUE サーバを実行する

④ WSDL が自動生成される

(3) 公開されたサービスの動作確認手順

① 公開されている WSDL の URL を取得する

② ①の URL から WSDL を LOAD する

③ GLUE によりプロキシプログラムが自動生成される
(接続するためのクラス, 接続するためのインターフェース)

④ ③で作成されたプロキシプログラムを利用し，サービスにアクセスするためのクライアントプログラムを作成する

③ サービス利用

2.3 .NET プラットフォームでの構築と動作確認

(1) 構築環境

.NET Framework SDK，.NET Framework SP1 を使用し，開発言語は C#を用いた.

(2) サービス公開までの手順

① Web サービスとして発行するクラスを記述しサービスとして公開するメソッドを[WebMethod]属性とする.

② ①で作成したファイルを IIS の仮想ディレクトリに配置する

③ ②にアクセスしたときに自動コンパイル・実行され，WSDL が自動生成される.

(3) 公開されたサービスの動作確認手順

① 公開された WSDL から wsdl.exe を利用し，プロキシプログラムを生成する

② ①からサービスにアクセスするためのクライアントプログラムを作成する

③ サービス利用

3. Java/.NET による Web サービス相互利用実証

2.1 で述べた Web サービスを用いて Java クライアントプログラムから.NET Web サービスおよび C#クライアントプログラムから Java Web サービスでの相互利用性について述べる.

相互利用実証環境を図 2 に示す．また実証結果を表 1 に示す.

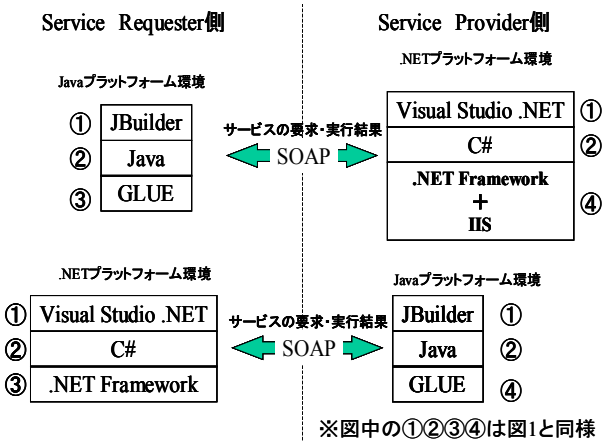

図２　相互利用実証環境図

実証 1： double 型のデータを結果とする Java と.NET 間の相互利用実証

(a) Java クライアントで.NET Web サービスを利用した場合の実証を行った.

2.3-(2)の手順に沿って，Web サービスとして公開する．そして 2.2-(3)の手順に沿って公開されたサービスを利用した．サービスの実行結果として SOAP レスポンスをリスト 3 に示す.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
        <mulResponse xmlns="http://tak.ip.net/WebServices">
                <mulResult>9.2465096069933264</mulResult>
        </mulResponse>
</soap:Body>
</soap:Envelope>
```

リスト 3　実証 1 (a) の SOAP レスポンス

(b) C#クライアントで Java　Web サービスを利用した場合の実証を行った.

2.2-(3)の手順に沿って，Web サービスとして公開する．そして 2.3-(2)の手順に沿って公開されたサービスを利用した．サービスの実行結果として SOAP レスポンスをリスト 4 に示す.

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
 xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
 soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
        <soap:Body>
          <n:divResponse xmlns:n='http://tempuri.org/mathmath'>
           <Result xsi:type='xsd:double'>0.9465472009121418</Result>
          </n:divResponse>
        </soap:Body>
</soap:Envelope>
```

リスト 4　実証 1 (b) の SOAP レスポンス

実証 2： String 配列のデータを結果とする Java と.NET 間の相互利用実証

(a) Java クライアントで.NET Web サービスを利用した場合の実証を行った.

同様の手順に沿って，サービスを公開し，公開されたサービスの実行結果として SOAP レスポンスをリスト 5 に示す.

```
?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
   <getAddressesByZipCodeResponse xmlns=
      "http://kazuya.homeip.net/">
       <getAddressesByZipCodeResult>
              <string>東京都港区海岸（３丁目）</string>
              <string>東京都港区芝浦（２～４丁目）</string>
       </getAddressesByZipCodeResult>
   </getAddressesByZipCodeResponse>
</soap:Body>
</soap:Envelope>
```

リスト 5　実証 2 (a) の SOAP レスポンス

(b) C#クライアントで Java　Web サービスを利用した場合の実証を行った.

同様の手順に沿って，サービスを公開し，公開されたサービスの実行結果として SOAP レスポンスをリスト 6 に示す.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
 <soap:Body>
   <n:getAddressesByZipCodeResponse
    xmlns:n='http://tempuri.org/AddressSearch'>
     <Result href='#id0'>
     </Result>
   </n:getAddressesByZipCodeResponse>
   <id0 id='id0' soapenc:root='0' xsi:type='soapenc:Array'
    soapenc:arrayType='xsd:string[2]'>
      <i xsi:type='xsd:string'>東京都新宿区新宿1600022</i>
        <i xsi:type='xsd:string'>埼玉県比企郡鳩山町石坂3500311</i>
   </id0>
 </soap:Body>
</soap:Envelope>
```

リスト 6　実証 2 (b) の SOAP レスポンス

実証 3： 構造体を結果とする Java⇔.NET 間の相互利用実証

(a) Java クライアントで.NET Web サービスを利用した場合の実証を行った.

同様の手順に沿って，サービスを公開し，公開されたサービスの実行結果として SOAP レスポンスをリスト 7 に示す.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
  <getAddressClassesByZipCodeResponse
    xmlns="http://kazuya.homeip.net/">
          <getAddressClassesByZipCodeResult>
                  <Address>
                  <Ken>東京都</Ken>
                  <Shiku>港区</Shiku>
                  <Choson>芝浦（2～4丁目）</Choson>
                  <zipCode>1080023</zipCode>
                  </Address>
          </getAddressClassesByZipCodeResult>
  </getAddressClassesByZipCodeResponse>
</soap:Body>
</soap:Envelope>
```

リスト7　実証３(a)の SOAP レスポンス

(b) C#クライアントで Java　Web サービスを利用した場合の実証を行った.

同様の手順に沿って，サービスを公開し，公開されたサービスの実行結果として SOAP レスポンスをリスト８に示す.

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
xmlns:ns5='http://www.themindelectric.com/package/'>
 <soap:Body>
 <n:getAddressClassByZipCodeResponse
  xmlns:n='http://tempuri.org/AddressSearch'>
   <Result href='#id0'></Result>
 </n:getAddressClassByZipCodeResponse>
  <id0 id='id0' soapenc:root='0' xsi:type='ns5:Address'>
         <Ken xsi:type='xsd:string'>東京都</Ken>
         <Shiku xsi:type='xsd:string'>新宿区</Shiku>
         <Choson xsi:type='xsd:string'>新宿</Choson>
         <zipCode xsi:type='xsd:string'>1600022</zipCode>
  </id0>
 </soap:Body>
</soap:Envelope>
```

リスト８　実証３(b)の SOAP レスポンス

表１　実証結果

	実証1	実証2	実証3
Javaクライアント⇔.NET Webサービス	利用可能	利用可能	利用可能
C#クライアント⇔Java Webサービス	利用可能	利用可能	利用可能

以上の結果から研究室内 Web サービス動作環境下での Java と.NET の相互利用が可能であることがわかった.

4. SOAP のパフォーマンスに関する考察

ローカル呼び出しの場合とインターネットを介した呼び出しの場合の二つに分け，SOAP データ量を変化させたときのレスポンスタイムを測定し，その結果を図４，図５に示す.

4.1 測定環境

測定環境を図３に示す.

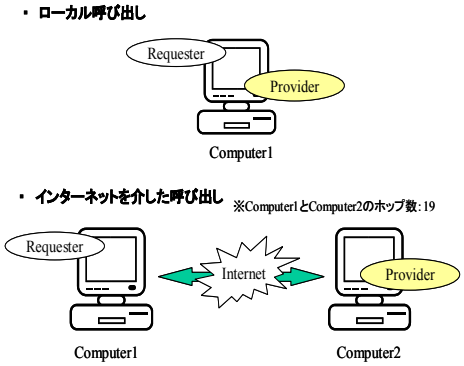

図３　SOAP パフォーマンス測定環境図

Computer１及び Computer２のスペックをまとめたものを表２に示す.

表２　マシンスペック表

	Computer1	Computer2
CPU	Intel Celeron500MHz (*2)	Intel Pentium4 2.2GHz
Memmory	512MB	512MB
OS	Windows XP	Windows XP
Network	ADSL(1.4Mbps)	10Mbps
Web Server	IIS5.0& .NET Framework	IIS5.0& .NET Framework

4.2 測定方法

プログラムは Visual Studio.NET で作成した.

レスポンス量（プロバイダからリクエスタへのデータ量）を変化させレスポンスタイムを測定する.

- プロバイダ側のプログラムは引数によってレスポンス量が変化する．レスポンス内容は単純な文字列を返す.
- リクエスタ側のプログラムがプロバイダ側にレスポンス量を指定し，取得要求を送信する．レスポンス要求量を 1Kbyte～10Kbyte まで 1Kbyte づつ増加させたときのレスポンスタイムを 100 回計測しその平均値を測定する.

①ローカル呼び出しの場合

1つの計算機上でプロバイダとリクエスタが動作する.

②インターネットを介した呼び出しの場合

2つの計算機がインターネットを介して接続され,プロバイダとリクエスタが分散して動作する.

4.3 実証結果および考察

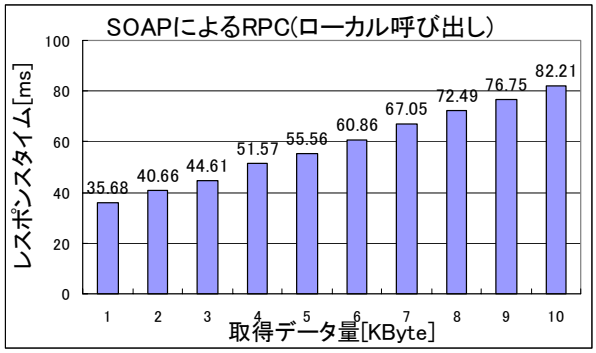

図4 ローカル呼び出し時の測定結果

取得データ量を 10 倍としたときレスポンスタイムは2.3倍になった

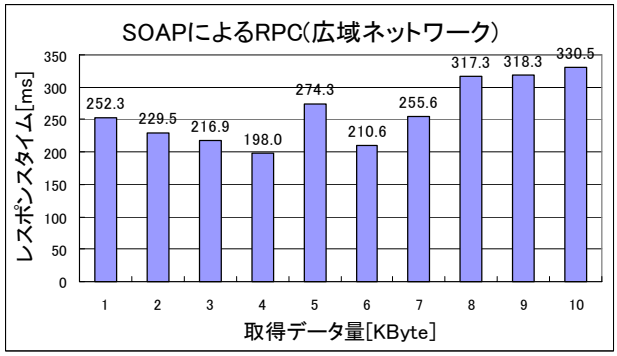

図5 インターネットを介した時の測定結果

取得データ量を 10 倍としたときレスポンスタイムは1.3倍になった.

測定の結果 SOAP のパフォーマンスはその時のネットワーク環境に依存することが分かる. またローカル呼び出しでは, SOAP メッセージのサイズとレスポンスタイムが比例関係になるが, インターネットを介した呼び出しでは, SOAP メッセージのサイズに依存しないレスポンスタイムとなる. よって Web サービスを用いたシステムのパフォーマンスをあげるためには, Web サービスの設計段階で, Web サービス自体の粒度を可能な限り大きくし, Web サービスを呼び出す回数を減らす設計を行う必要がある.

5. まとめと今後の課題

Web サービス・システムの代表的なものとして, Java プラットフォームベースのものと.NET プラットフォームベースのものについて, 大学研究室内でサービスを構築し相互利用実証を行った. 四則演算 Web サービス, 郵便番号検索 Web サービスを作成し, Java と.NET 間での相互利用実証を行った. また SOAP のパフォーマンスを測定し, 測定結果について考察した.

今後は, 在庫管理システムや調達システムなど SCM に着目したプロトタイプを構築し, 従来の Web コンピューティングによる構築と Web サービスによる構築との組み合わせ, また異なるプラットフォーム間での相互利用実証を行っていく. また SOAP パフォーマンスに関しては, CORBA や JavaRMI など既存の分散オブジェクト技術との比較を行っていく.

参考文献

[1] 嶋本正他著, “Web サービス完全構築ガイド”, 日経 BP 社,pp10-42,2001

[2] 浅田孝利他著, “Web サービスにおける EC ビジネスモデルの構築と評価”, IPSJ 第64回全国大会講演論文集(4) pp373-374,2002

[3] 栗原潤他著,”SOAP/WSDL/UDDI を基盤とした Web サービスによる EC モデルの実証と評価”, IPSJ 情報システムと社会環境 研究報告 No.79, pp53-60,2002

[4] 実森仁志, “Web サービス完全ガイド-Web サービス対応製品が出そろう”, 日経 BP 社, pp34-50,2002

[5] Graham Glass, “Web Services-Building Blocks for Distributed Systems ”,PRENTICE HALL Inc,pp191-214,2002

[6] Kennard Scribner,Mark C.Stiver, “SOAP 技術入門”, (株)ピアソン・エデュケーション, pp8-25,2001

[7] Irmen de Joug,”Web Services/SOAP and CORBA”,http://www.xs4all.nl/~irmen/comp/CORBA%20vs%20SOAP.html