社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
NS2004-99,IN2004-58,CS2004-54(2004-09)

複数 Web サービス連携手法の実験的評価

石井 健一　串戸 洋平　井垣 宏　中村 匡秀　松本 健一

奈良先端科学技術大学院大学 情報科学研究科 〒630-0192 奈良県生駒市高山町 8916-5

E-mail: {keni-i, youhei-k, hiro-iga, masa-n, matumoto}@is.naist.jp

あらまし Web サービスは，複数のサービスの連携，再利用を強く意識したサービス指向アーキテクチャの考えに基づいており，付加価値を高めたサービス提供を実現するインフラとして注目されている．本稿では，複数の Web サービスが連携して新たな統合サービスを実現する場合の呼び出し関係に着目し，Web サービスアプリケーションの性能を評価する．具体的には，Web サービスの呼び出し関係をトポロジーとして性質付けし，トポロジーの違いと Web サービスアプリケーションの総合的な性能との関連を調べる．その結果，連携する Web サービスの総数が等しい場合には，クライアントアプリケーションの応答時間に顕著な差が見られないが，トポロジーの与え方によっては，各サービスの応答時間が偏ることがわかった．Web サービスの応答時間は，直接的および間接的に呼び出す Web サービスの数が多いほど増加し，この数は，統合サービス内の性能ボトルネックを表す指標となることがわかった．

キーワード Web サービス，サービス連携，連携方式，Web サービスアプリケーション，トポロジー

An experimental evaluation of topologies integrating Web services

Ken-ichi ISHII　Youhei KUSHIDO　Hiroshi IGAKI　Masahide NAKAMURA　and

Ken-ichi MATSUMOTO

Graduate School of Information Science, Nara Institute of Science and Technology

8916-5 Takayama, Ikoma, Nara, 630-0192 Japan

E-mail: {keni-i, youhei-k, hiro-iga, masa-n, matumoto}@is.naist.jp

Abstract Web services are a powerful service-oriented infrastructure to provide various value-added services, which specifically emphasizes the integration and reuse of the service components. The paper evaluates the performance effect of the service integration, focusing on the call relationships among multiple Web services. Specifically, we characterize the call relationships as topologies of the integrated Web services, and investigates the correlation between a topology and the performance. The experimental evaluation shows that; when the number of Web services is equal, the difference of the topology does not yield any impact on total response time of the client application. However, response time of each service varies depending on the given topology. Also, the total number of services to which a Web service performs direct or indirect calls significantly affects the response time of the Web service. This number is shown to be a reasonable metric to indicate the bottleneck in the whole integrated service.

Keyword Web Services, Service Integration, Integration Schemes, Web Service Applications, Topology

1. はじめに

Webサービスは，自己記述型の検索可能で汎用的なサービスをプラットフォームやアクセスするデバイスの種類を問わずに接続するための枠組みである[16]. Webサービスのインターフェースは，厳密に型付けされ（WSDL），サービス間の通信には，標準的なプロトコル（HTTP/SOAP）が用いられる． これにより，サービス間の疎結合が実現し，複数のサービスを連携，再利用した統合サービスの開発が容易になるとされている．現在，Webサービスを用いたシステムがいくつか開発され，公開されている[1][4][18][19]. しかし，Webサービスは比較的新しい技術であるため，現在運用されているシステムは，試験段階のものや実地運用が開始されてから日の浅いものが多い[12]. また，Webサービスの開発方法論も未だに体系化されておらず，Webサービスアプリケーションの特長である，サービス連携や再利用性を評価する体系的な手法も存在しない．

本稿の目的は，複数のWebサービスが連携して新たな統合サービスを実現する場合の呼び出し関係に着目し，Webサービスアプリケーションの性能を評価することである．具体的には，Webサービスの呼び出し関係をトポロジーとして性質付けし，トポロジーの違いとWebサービスアプリケーションの総合的な性能との関連を調べる．

性能評価においては，与えられたトポロジーに従ってWebサービスを動的に統合させるシミュレーションシステムを実装して，14種類のトポロジーを計測した．その結果，連携するWebサービスの総数が等しい場合には，クライアントアプリケーションの応答時間に顕著な差が見られないが，トポロジーの与え方によっては，各サービスの応答時間が偏ることがわかった．Webサービスの応答時間は，直接的および間接的に呼び出すWebサービスの数が多いほど増加し，この数は統合サービス内の性能ボトルネックを表す指標となることがわかった．

2. Webサービス連携トポロジー

2.1. Webサービス連携トポロジー

Webサービスは，複数のサービスの連携，再利用を強く意識したサービス指向アーキテクチャの考えに基づいている．Webサービスアプリケーションにおいて，1つのクライアントアプリケーション（以降 **CA**）が複数のWebサービス（以降 **WS**）と連携する場合，CA-WS間，またはWS-WS間において，様々なサービス呼び出し形態（連携方式）が考えられる．この連携方式をWebサービス連携トポロジー（以降 **WSトポロジー**）と呼ぶ．

複数WSの連携における最小単位として，1つのCAと2つのWSの連携を考える．まず，CAが，WS1にサービスを要求し，その応答を一旦受け取ってから，CAが，WS2にサービスを要求する方式が考えられる．本稿では，このような方式を**リダイレクト型**と呼ぶ．図 1は，リダイレクト型トポロジーをUMLのコラボレーション図に基づいて表現したものである．

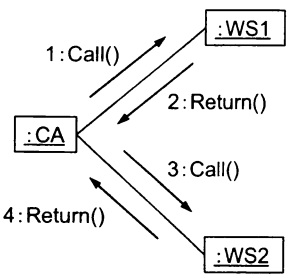

図 1 リダイレクト型

もう1つの最小単位の連携方式として，CAが，WS1にサービスを要求すると，そのWS1が，CAの代わりにWS2にサービスを要求する方式が考えられる．本稿では，このような方式を**プロキシ型**と呼ぶ（図 2）．

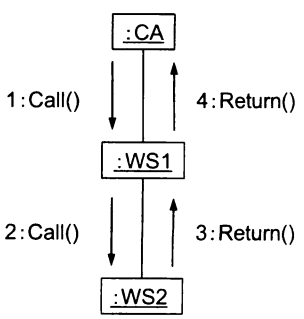

図 2 プロキシ型

すべてのWSトポロジーは，リダイレクト型とプロキシ型の組み合わせで表現できる．以降では，WSトポロジーの記法と評価するWSトポロジーについて述べる．

2.2. WSトポロジー式

ここで，任意のWSトポロジーを表現するための記法，WSトポロジー式（以降**トポロジー式**）を導入する．図 3にBNF（Backus Naur Form）による文法定義を示す．

トポロジー式の意味を簡単に述べる．全てのトポロジー式はCA，すなわちクライアントアプリケーションから始まる．これはサービスの開始がCAによってトリガされることを表す．式中の「;」は2つのWSの直列的な逐次呼び出しを表し，プロキシ型連携を形成する．また，「+」は同一オブジェクトが2つのWSを順番に呼び出すことを意味し，リダイレクト型連携を

表す．「()」は式における優先度を制御するために用いられる．

例として，図 1および図 2のトポロジーT1, T2 は，それぞれ，T1 = CA; (WS1 + WS2), T2 = CA; WS1; WS2 で表現される．

```
WS トポロジー:
    CA ';' 式
式:
    プロキシ式
    リダイレクト式
    '(' 式 ')'
プロキシ式:
    Web サービス
    Web サービス ';' 式
リダイレクト式:
    Web サービス '+' 式
Web サービス:
    "WS[0-9]+"
```

図 3　WS トポロジー式

2.3. 評価する WS トポロジー

1 つの CA と連携する WS の総数を n とする場合，本稿では，n=2, 3, 4 の全ての WS トポロジーを評価する．n が与えられた時，可能な WS トポロジーの総数を T(n)とすると，$T(n) = 2^{(n-1)}$ となる．図 4に実験で用いる全ての WS トポロジーのコラボレーション図を示す．

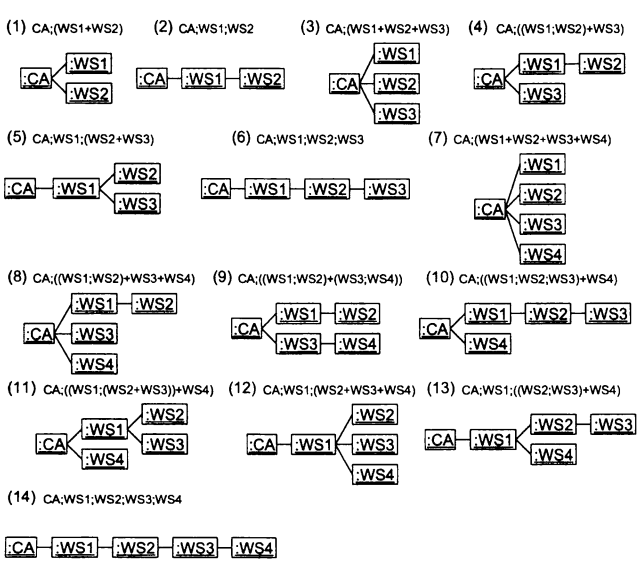

図 4　WS トポロジーのコラボレーション図（n=2,3,4）

各 WS トポロジーには,識別子として，それぞれ(1)～(14)の番号を付与している．また，WS トポロジーに含まれる WS には，実行される順番に番号が付加されている．例として，図 4の(3)は，CA が WS1,WS2,WS3 に対して処理を順次要求する，

3. WS 連携シミュレーションシステムの開発

我々は，WS トポロジーの違いが統合サービスの性能にどのような影響を与えるかを評価するため，任意の WS トポロジーをシミュレートできる WS シミュレーションシステム（以降 **WSS システム**）を開発した．本章では，WSS システムの概要について述べる．

3.1. WSS システム概要

WSS システムは，n 個の WS を任意の WS トポロジーに従って動的に統合し，統合された WS 全体(以降**統合 WS**)の性能評価を行うためのシステムである．

ユーザが WS トポロジー式を与えると，各 WS について，それが自ら呼び出す次の WS（以降**次段 WS**）の集合とその呼び出し順序が決定する．WSS システムでは，この次段 WS 集合と呼び出し順に関する情報を，各 WS が参照する定義ファイル（以降 **WS 定義ファイル**）として保持する．各 WS は，実行時に WS 定義ファイルを参照し，次段 WS を動的に決定，呼び出しを行う．こうして，与えられた WS トポロジーに従った統合 WS が実行される．WSS システムは，その実行中に，統合 WS と各 WS の性能を計測する．

3.2. WSS システム設計

開発した WSS システムは，主に，**WS 定義ファイル**，**連携 WS** および**連携 CA** の 3 つの構成要素から成る．以降，これらについて説明する．

3.2.1. WS 定義ファイル

WS トポロジー（式） は，複数の WS 間の大域的な呼び出し関係を表したものである．しかし，実際にトポロジーに従った統合 WS を実現するには，各 WS が局所的にどの WS を呼び出せばよいのかを知る必要がある． WS 定義ファイルは，各連携 WS および連携 CA の局所的な動作を定義する．具体的には，次段の連携 WS 呼び出しに必要となる以下の情報を格納している．

① 呼び出す WS 名
② 呼び出す WS のメソッド名
③ メソッドに渡すパラメータ情報

例として図 4(13)の WS トポロジーを考える．CA は，WS1 を呼び出す．そして，WS1 は，WS2 および WS4 をこの順で呼び出し，WS2 は，WS3 を呼び出す．WS3 は，何も呼び出さない．このような各 WS が行う動作を WS 定義ファイルに記述する（図 5参照）．

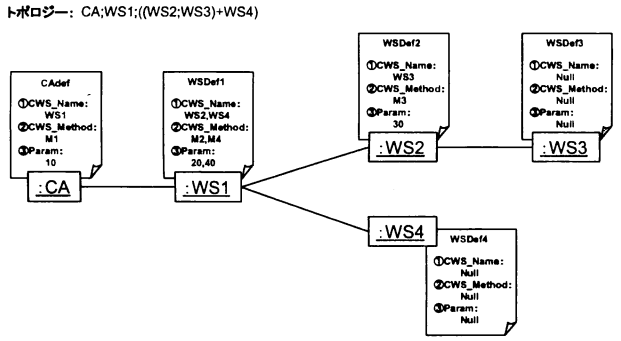

図 5 WS トポロジーから生成される WS 定義ファイル

3.2.2. 連携 WS

連携 WS は，WSS システム実行時に，次段の連携 WS を WS 定義ファイルにより動的に決定し，呼び出しを行う WS である．複数の連携 WS を呼び出す場合は，現在呼び出し中の連携 WS の応答を受け取ってから，次の連携 WS を呼び出す方式（同期的呼び出し）を採用する．また，連携 WS は，処理を依頼された時刻，各次段 WS を呼び出した時刻，処理が終了した時刻にログを書き出す．

3.2.3. 連携 CA

連携 CA は，WSS システムの動作をトリガする CA をシミュレートするアプリケーションである．連携 WS との本質的な違いは，WS トポロジーにおいて，最初に実行されることだけで，連携 WS の機能を内包する．つまり，連携 CA は，連携 WS と同様に，与えられた WS 定義ファイルを元に動的に連携 WS を呼び出し，実行時刻に関するログを書き出す．評価実験用に，統合サービスのシミュレーション回数を指定できる機能も備えている．

3.3. 実装環境

3.2 節でのシステム設計を基に，WSS システムを実装した．実装環境は，以下の通りである．

言語：Microsoft Visual C# .NET
プラットフォーム：Microsoft .NET Framework 1.1
Web サーバ：IIS(Internet Information Services)5.1

連携 CA は，コンソールアプリケーションとして実装した．また，連携 WS の動的呼び出し部の実装においては，実行時バインドに基づくアプローチ[22]を採った．これは，まず，呼び出す可能性のある全ての WS の WSDL ファイルから，その WS を起動するための DLL を予め作成しておき，連携 WS は実行時に WS 定義ファイルを参照して，適切な DLL を動的リンクして次段 WS を呼び出すというアプローチである．

4. 評価実験

本章では，実装した WSS システムを用いて，様々な WS トポロジーに対する性能評価を行う．実験用 Web サーバのスペックは，以下の通りである．

CPU：Pentium 4 2.4GHz
メモリ：512MB
OS：Windows XP Professional SP1
プラットフォーム：Microsoft .NET Framework 1.1
Web サーバ：IIS(Internet Information Services)5.1

一般的には，複数の WS 間の通信には，ネットワークのオーバーヘッドがかかる．しかし，本研究では，純粋にトポロジーの影響のみを評価するため，全ての連携 WS を 1 台のサーバ内に配置し，通信遅延による影響を排除した．

4.1. 評価項目の検討

実験では，図 4に示した各 WS トポロジーにおける連携 CA と連携 WS の応答時間を計測する．応答時間とは，連携 CA および連携 WS が処理を開始してから処理を終えるまでの時間である．サービス 1 回の応答時間は極めて短いため，連携 CA で繰り返し回数を 1000 回に設定して応答時間を計測した．この結果から，連携 CA および連携 WS それぞれの応答時間の合計と連携 WS の応答時間の分散を得る．

さらに，各 WS トポロジーにおける**直接 WS 呼び出し数**と**間接 WS 呼び出し数**を計測する．直接 WS 呼び出し数とは，連携 CA あるいは連携 WS が直接呼び出す連携 WS の数である．間接 WS 呼び出し数とは，連携 CA あるいは連携 WS が直接呼び出す連携 WS 以降に関係するすべての連携 WS の合計数である．

4.2. 比較評価

応答時間の計測結果を表 1，直接・間接 WS 呼び出し数を表 2に示す．

連携 CA の応答時間は，連携 WS の数が等しい WS トポロジー間において，ほとんど差が認められなかった．また，連携 CA の応答時間は，WS トポロジーの連携 WS 数が 1 つ増すごとに約 15 秒増加した結果となった．この結果から，連携 WS の数が等しい WS トポロジー同士では，システム全体の処理時間に差がほとんど無く，システム全体の処理時間は，連携 CA の利用する連携 WS 数と共に増加することがわかる．つまり，連携 CA の利用する WS 数は，統合サービス全体の効率性に大きな影響を与える．

連携 WS それぞれの応答時間は，WS トポロジー毎に偏りが見られる．連携 WS の応答時間は，直接 WS 呼び出し数と間接 WS 呼び出し数の合計（呼び出し総数と呼ぶ）が多い連携 WS ほど，長くなる傾向が認め

られた．この実験では，呼び出し総数が1増えるたびに，WSの応答時間が約15秒長くなった．これは，連携WSが同期的にWSを呼び出すため，呼び出したWSから処理結果を受け取るまでの間，時間を待たなければならないからである．呼び出し総数はWSトポロジーに依存し，大きな呼び出し総数を持つWSは応答時間が長くなり，高負荷時には未処理のトランザクションが停滞しやすくなる．結果的に，統合サービスのボトルネックとなる．

表 1 応答時間の計測結果（秒）

WSトポロジー	CA	WS1	WS2	WS3	WS4	分散
1	35.870	6.052	5.897			0.006
2	36.031	21.475	5.863			60.934
3	51.480	6.051	5.962	5.919		0.003
4	51.527	21.512	5.937	5.919		53.969
5	51.418	36.905	5.909	5.840		213.971
6	51.666	37.172	21.314	5.820		163.834
7	67.119	6.138	5.896	5.910	5.937	0.010
8	67.269	21.776	6.153	5.885	5.898	46.803
9	67.597	21.643	5.943	21.444	5.991	60.662
10	67.006	37.061	21.338	5.901	5.922	166.484
11	66.875	36.953	5.932	5.890	5.877	180.804
12	66.738	52.231	5.902	5.861	5.860	402.918
13	67.067	52.610	21.473	6.002	5.908	362.787
14	67.503	53.032	37.336	21.701	6.034	306.657

表 2 直接・間接WS呼び出し数

WSトポロジー	CA			WS1			WS2			WS3			WS4		
	直接	間接	合計	直接	間接	合計	直接	間接	合計	直接	間接	合計	直接	間接	合計
1	2	0	2	0	0	0	0	0	0						
2	1	1	2	1	0	1	0	0	0						
3	3	0	3	0	0	0	0	0	0	0	0	0			
4	2	1	3	1	0	1	0	0	0	0	0	0			
5	1	2	3	2	0	2	0	0	0	0	0	0			
6	1	2	3	1	1	2	1	0	1	0	0	0			
7	4	0	4	0	0	0	0	0	0	0	0	0	0	0	0
8	3	1	4	1	0	1	0	0	0	0	0	0	0	0	0
9	2	2	4	1	0	1	0	0	0	1	0	1	0	0	0
10	2	2	4	1	1	2	1	0	1	0	0	0	0	0	0
11	2	2	4	2	0	2	0	0	0	0	0	0	0	0	0
12	1	3	4	3	0	3	0	0	0	0	0	0	0	0	0
13	1	3	4	2	1	3	1	0	1	0	0	0	0	0	0
14	1	3	4	1	2	3	1	1	2	1	0	1	0	0	0

連携WSの応答時間の分散は，分散の値が低いほど各WSトポロジーにおける連携WSの応答時間が似通っていることを意味し，連携WSにかかる負荷が分散されていることを示す．各WSトポロジーを計測結果から求められた分散の度合いにより，以下の4つのグループに分類する．

G1（分散値無≒0）：1,3,7
G2（分散値小 46～61）：2,4,8,9
G3（分散値中 163～214）：5,6,10,11
G4（分散値大 306～403）：12,13,14

この応答時間の分散とトポロジーの関連について考察する．表2において，各グループ内のトポロジーの呼び出し総数に着目する．各トポロジーtにおける連携WSの呼び出し総数の最大値を，MAX(t)とする．すると，同じグループに属するトポロジーt1, t2について，MAX(t1)=MAX(t2)になることがわかる．例えば，G2については，MAX(2)=MAX(4)=MAX(8)=MAX(9)=1となる．同様に，G1,G3,G4については，それぞれ0, 2, 3となる．このことから，各WSの呼び出し総数の最大値が低く抑えられるトポロジーほど，偏りのないバランスがとれた応答時間が得られることがわかる．

4.3. 考察

実験では，連携WSの数が等しいWSトポロジー間において，連携CAの応答時間に顕著な差が見られないことがわかった．しかし，連携WSの応答時間は，呼び出し総数が多い連携WSほど，長くなることがわかった．よって，統合サービスを構成する各連携WSの負担を減らしたい場合は，各WSの呼び出し総数を減らすようなWSトポロジーを選択することが望ましいと思われる．例えば，統合サービスを構成するすべての連携WSの負担を最小にしたい場合は，各WSの呼び出し総数を0にすれば良い．つまり，統合サービスをリダイレクト型のWSトポロジーで構成すればよいことになる．これは，表1に示される純粋なリダイレクト型であるWSトポロジー（1,3,7）の分散の値が0に近似していることからも明らかである．

5. おわりに

本稿では，複数のWSを連携するシミュレーションシステムの設計・実装を行い，各WSトポロジーの性能評価を行った．

今後の課題としては，各WSトポロジーを実際の運用環境に近い形で性能評価すること，実装環境の違い（.NETとJAVAの比較）によるオーバーヘッドを計測すること，実際のWS適用事例を調査・比較し，分析対象として有用なWSトポロジーを特定するである．また，将来的には，WSアプリケーション開発方法論についての研究へつなげていきたい．

文 献

[1] Amazon Web Services, http://www.amazon.com/gp/browse.html/104-0877510-2922306?node=3435361

[2] 青山幹雄，“Webサービス技術とWebサービスネットワーク”，信学技報，IN2002-163，pp.47-52，Jan.2003.

[3] 福田直樹，肥塚八尋，和泉憲明，山口高平，“オントロジーに基づくWebサービスの自動連携”，信学技報，KBSE2004-3，pp.13-18，May.2004.

[4] Google Web APIs, http://www.google.com/apis/

[5] 本多泰理，矢田健，山田博司，“トラヒックフローを考慮したWebサービスのネットワーク構成法の一検討”，信学会総合大会講演論文集，2003-3，pp.354，Mar.2003.

[6] 本多泰理，“Webサービスのトランザクション処理の性能設計技術に関する一検討”，信学会総合大会講演論文集，2004-3，pp.235，Mar.2004.

[7] 石川冬樹，田原康之，古岡信和，本位田真一，“Webサービス連携のためのモバイルエージェント動作記述”，情報処理学会論文誌，Vol.45，No.6，pp.1614-1629，June.2004.

[8] 和泉憲明，幸島明夫，車谷浩一，福田直樹，山口高平，“多粒度リポジトリに基づくWebサービス

のビジネスモデル駆動連携”，信学技報，KBSE2002-32，pp.43-48，Jan.2003.

[9] 近藤秀和，村岡洋一，“Web サービスを利用した次世代型オンラインブックマークシステム”，信学技報，DE2004-4，pp.19-24，June.2004.

[10]越田高志，植村俊亮，“WSIF における Web サービス・メソッドの自動設定”，信学会総合大会講演論文集，2004-3，pp.120，Mar.2004.

[11]来間啓伸，本位田真一，“ポリシに基づく Web サービス・コミュニティ連合のモデル”，情報処理学会論文誌，Vol.45，No.6，pp.1593-1602，June.2004.

[12]村山隆彦，由良俊介，“Web サービスのビジネス運用に向けて”，信学技報，IN2003-166，pp.37-42，Jan.2004.

[13]西村徹，堀川桂太郎，“Web サービスにおける境界条件の検査”，信学会総合大会講演論文集，2003-6，pp.234，Mar.2003.

[14]大野邦夫，“オフィス文書と XML”，信学技報，IN2003-167，pp.43-50，Jan.2004.

[15]山登康次，武本充治，島本憲夫，“ユビキタス環境に適したサービス連携フレームワークにおけるサービステンプレート生成手法”，信学技報，IN2003-169，pp.1-6，Jan.2004.

[16]寺口正義，山口裕美，西海聡子，伊藤貴之，“ストリーミング処理による Web サービスおよび Web サービスセキュリティの軽量実装”，情報処理学会論文誌，Vol.45，No.6，pp.1603-1613，June.2004.

[17]寺井公一，和泉憲明，山口高平，“オントロジーとパターンに基づく Web サービス連携”，信学技報，KBSE2003-39，pp.35-40，Jan.2004.

[18]XML Web サービス対応 Business Travel System パイロット版，http://net.est.co.jp/jtb/about/

[19]XML Web サービス対応三省堂デイリーコンサイス体験版，http://www.btonic.com/ws/

[20]山本晃治，上原忠弘，松尾明彦，大橋恭子，猪股順二，松田友隆，山本里枝子，“XML アプリケーション開発のためのパターン体系の開発”，情報処理学会論文誌，Vol.45，No.6，pp.1584-1592，June.2004.

[21]山岡和雄，佐藤厚，山本茂樹，瀧口修，岡庭祐，小原利光，宮下慎一，“障害管理用統合 NMS 構築における Web サービスの適用事例”，信学技報，IN2003-160，pp.1-6，Jan.2004.

[22].NET XML Web Services Repertory，“Dynamically invoke XML Web Services (improved version)”，http://xmlwebservices.cc/index_Samples.htm