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
NS2004-318, IN2004-318(2005-2)

Webサービスアプリケーションのプロトタイピングおよび性能評価のためのシステム開発

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サービスの迅速なプロトタイピングを行うための汎用的なシステム WS-PROVE（Web Service PROtotyping Validation Environment）を提案する．WS-PROVEは，まだ実装が確定していない新規のWebサービスをダミーWebサービスとして抽象化し，別のダミーWebサービス，または，既存のWebサービスと任意のトポロジーで連携させることができる．また，応答時間やネットワーク遅延を設定すると，Webサービスを動的接続し，各サービスあるいは統合サービス全体の処理時間を計測できる．本稿では，Webサービスアプリケーションの性能プロトタイピングシステムの要件を整理し，WS-PROVE の設計，実装，評価を行う．また，WS-PROVE を用いて，いくつかの Web サービスアプリケーションのプロトタイピングおよび性能見積もりを行い，WS-PROVE の有効性を示す．

キーワード Webサービス，Webサービスアプリケーション，プロトタイプ，プロトタイピング

WS-PROVE: A System for Rapid Prototyping and Performance Evaluation of Web Service Applications

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 This paper proposes a general framework, called WS-PROVE (Web Service PROtotyping Validation Environment), for rapid prototyping of Web service applications. In a requirement definition stage, WS-PROVE first abstracts a Web service under development as a dummy, and then integrate the dummy with other dummies or the existing ready-made Web services in arbitrary topologies. For a given configuration including the integration topology, processing time for each service and the network delay, WS-PROVE dynamically connects the Web services. Then, it measures the response time for each service and the whole integrated services. In this paper, we first clarify system requirements of the rapid prototyping system for Web service applications. Then, we present design and implementation of WS-PROVE. Additionally, we conduct a case study to prototype some Web service applications to demonstrate the effectiveness of WS-PROVE.

Keyword Web Services, Web Service Applications, Prototype, Prototyping

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OSやアクセスデバイスの種類を問わずにネットワーク経由で接続・連携するための枠組みである[9]．異機種分散環境におけるソフトウェアシステムの各機能をサービスという単位で公開し，共通のインタフェースを用いて疎結合するサービス指向アーキテクチャ（SOA）の実現手段として注目を集めている．

近年，システムの大規模化とネットワークの発達に伴い，既存システムの再利用と分散処理が重要な課題となり，Webサービス技術が実際に使われ始めている．例えば，企業内の異機種の基幹システムをWebサービス化して連携・統合する[10]，レガシーな社内システムを付加価値サービスとして外部公開し企業間システムを実現する[3]，また，Webアプリケーションをサービス化して統合ポータルを構築する[11]といった事例がある．

しかし，Webサービスは比較的新しい技術であるため，Webサービスを用いたシステムの体系的な開発方法論やベストプラクティス，キラーアプリケーション等は，未だ十分明らかになっていない．また，従来のアプリケーションと異なり，Webサービスでは，サービス連携にかかるサーバのミドルウェア部（XML，WSDL，SOAP/HTTPの処理等）のオーバーヘッドとサーバ間のネットワークを陽に考慮しなければならない．従って，Webサービスアプリケーションでは，特に，**サービス連携部分の効率**や**信頼性**といった**非機能要求**が重要となる．しかし，現状，これらの非機能要求は，実装後のテストで初めて評価され，システムの要求定義段階では取得できない．結果として，開発の早期段階において，Webサービスを用いるべきかどうかの判断は，開発者の経験に基づいてアドホックに行わざるをえず，開発の大きな後戻りを生みかねない．

この問題に対処するため，我々は，本研究において，Webサービスの迅速なプロトタイピング（Rapid Prototyping）[6]を行うための汎用的なシステム**WS-PROVE**（Web Service PROtotyping Validation Environment）を提案する．WS-PROVEの目的は，開発の要求段階において，Webサービス連携部分の非機能要求（現段階では主に性能）を評価可能にすることである*．

WS-PROVEは，まだ実装が確定していない新規のWebサービスを**ダミーWebサービス**として抽象化し，別のダミーWebサービス，または，既存のWebサービスと**任意のトポロジー**で連携させることができる．全体のサービス連携トポロジーと各ダミーWebサービスに対する応答時間や負荷を設定すると，WS-PROVEはダミー・既存Webサービスを設定に従って動的接続し，各サービスあるいは統合サービス全体の応答時間を計測する．Webサービスの接続・連携は，実際のミドルウェアおよびネットワークを介して行われるため，シミュレーションによる計測見積もりより正確な性能評価が可能となる．本稿では，Webサービスアプリケーションのプロトタイピングシステムの要件を整理し，WS-PROVEの設計，実装，評価を行う．またWS-PROVEを用いて，いくつかのWebサービスアプリケーションのプロトタイピングおよび性能見積もりを行い，WS-PROVEの有効性を示す．

2. ソフトウェアプロトタイピング

2.1. ソフトウェアライフサイクルモデル

ソフトウェア工学では，ソフトウェアの開発や運用・保守段階がどのように進んでいくかモデル化したものを**ソフトウェアライフサイクルモデル**（software life cycle model）と呼ぶ[6]．伝統的なソフトウェアライフサイクルモデルには，**ウォータフォールモデル**（waterfall model）がある．ウォータフォールモデルは，ソフトウェア開発工程を要求，設計，テスト，導入，運用，保守といった諸段階に分け，重複や反復が無いように各工程を実行するモデルである（図1）．

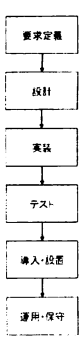

図1 ウォータフォールモデル

ウォータフォールモデルには，各工程を分業化できる利点がある．しかし，工程の後戻りは，一般的に莫大な工数がかかるため，各工程は，できる限り完全に完了する必要がある．特に，ライフサイクルの一番目にあたる要求定義は，完全に行わないと後続の工程からの後戻りが頻繁に発生するため，慎重を期する必要がある．これらの問題点を解決するために，新たなソフトウェアライフサイクルモデルがいくつか提案されている[8]．その1つに**ラピッドプロトタイピング**（rapid prototyping）がある．

---

* 一般にプロトタイピングでは，ユーザインタフェースなどの機能要求を扱う場合が多い．しかし，本研究では，Webサービスで重要となるサービス連携部の性能という非機能要求について，**性能プロトタイピング**[6]を行う．

2.2. ラピッドプロトタイピング

ラピッドプロトタイピングは，利用者の要求を正確に定義することを目的として，開発の初期段階でソフトウェアプロトタイプ(software prototype)を開発し，利用者に試用させることによって要求を正確に定義するモデルである（図 2).

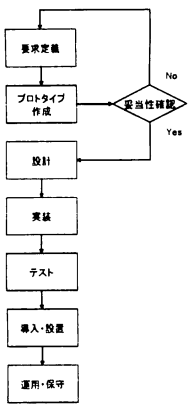

図 2 ラピッドプロトタイピング

ソフトウェアプロトタイプでは，最終製品が備えるべき機能の詳細までは実装せず，ユーザインタフェースや主な機能だけが実現される．また，ソフトウェアプロトタイプでは,開発の迅速さが要求されるために，一般には，非機能要求である安全性や信頼性，効率，性能などが無視される．本稿では，ソフトウェアプロトタイプを単にプロトタイプ，プロトタイプを開発することをプロトタイピングと呼ぶ.

プロトタイピングは，プロトタイプの対象，プロトタイピングのタイミング，プロトタイプと最終製品の関係によって分類される.

まず，プロトタイピングは，プロトタイプの対象によって，**ユーザインタフェースプロトタイピング**と**機能型プロトタイピング**（functional prototyping）に分類される．対象がユーザインタフェースである場合，ユーザインタフェースプロトタイピング，対象がユーザインタフェースと機能（計算機能・処理機能）である場合，機能型プロトタイピングに分類される.

次に，プロトタイピングのタイミングによって，**発見型プロトタイピング**（exploratory prototyping）と**経験型プロトタイピング**（experimental prototyping）に分類される.ユーザによる要求の妥当性確認を目標にし，要求定義段階で行われる場合，発見型プロトタイピングに分類される．設計段階で採用した技法によって，目標とする性能が達成できるか，ハードウェアは適切かどうか開発者が確認する段階で行われる場合，経験型プロトタイピングに分類される．また，経験型プロトタイピングは，性能を確認する**性能型プロトタイピング**（performance prototyping）とハードウェアを確認する**ハードウェアプロトタイピング**（hardware prototyping）に分類される.

最後に，ソフトウェアプロトタイプと最終製品の関係によって,**使い捨て型プロトタイピング**(throw-away prototyping）と**進化型（成長型）プロトタイピング**（evolutional prototyping)に分類される．この分類は，ユーザによる要求の妥当性確認後のプロトタイプがどのように使われるかによって行われる．要求の妥当性確認後のプロトタイプを破棄する場合，使い捨て型プロトタイピング，要求の妥当性確認後のプロトタイプを破棄せず，ベースとして開発を進める場合，進化型プロトタイピングに分類される．プロトタイピングの分類を表 1 に示す.

表 1 プロトタイピングの分類

分類方法	分類		説明	
対象	ユーザインタフェース		ユーザインタフェースが対象	
	機能型		ユーザインタフェースと機能が対象	
タイミング	発見型		要求定義段階	
	経験型	性能型	実装段階	性能を確認
		ハードウェア		ハードウェアを確認
最終製品との関係	使い捨て型		プロトタイプを破棄する	
	進化型(成長型)		プロトタイプを破棄しない	

3. Web サービスプロトタイピングシステムの開発

我々は，Web サービスアプリケーションにおけるプロトタイピングを実現するため，Web サービスプロトタイピングシステム（以降 **WS-PROVE: Web Service PROtotyping and Validation Environment**）を開発した．本章では，WS-PROVE の概要について述べる.

3.1. システム要件

本項では，WS-PROVE のシステム要件を明確にする.

(R1)連携 Web サービスの変更

Web サービスアプリケーションには，ネットワーク上に分散する Web サービスとの柔軟な連携が望まれる．そのため，システムは，既存の Web サービス，新規に構築する Web サービスに関わらず，プロトタイプ対象のサービスと容易に連携指示できる必要がある．これは，Web サービスの特徴である疎結合性によるものである．システムには，サービスの変更や再構築の効率性を高めることが要求される.

(R2)トポロジーの変更

複数の Web サービスを利用し，Web サービスアプリケーションを実現する際,Web サービスの連携方法は，複数存在する．そのため，システムは，Web サービスの**連携トポロジー**を容易に変更できる必要がある.

(R3)性能の確認

Web サービスアプリケーションの性能評価には，Web サービスのミドルウェア部およびネットワーク部

のオーバーヘッドが無視できない．そのため，システムでは，性能面を見積もれることが必要である．具体的には，応答時間を計測できること，ネットワークの遅延を容易に設定できることが必要である．

(R4)システムの使いやすさ

Webサービスアプリケーションの開発には，経験や知識が必要とされる．そのため，システムでは，設定を簡単にでき，最低限の知識で利用できることが重要である．また，プロトタイピングが開発の初期段階で行われるために，システムには，プロトタイプを短時間で構築できることが望まれる．

3.2. システム設計

WS-PROVEは，Webサービス（以降WS）を任意の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）の性能を計測する．WS-PROVEは，主に**WS定義ファイル**，**連携WS**および連携クライアントアプリケーション（以降**連携CA**）の3つから構成される．

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（以降次段WS）呼び出しに必要となる情報（呼び出すWS名，呼び出すWSのメソッド名，メソッドに渡すパラメータ情報）を格納している．

連携WSは，プロトタイプの対象となる開発中のWebサービスをシミュレートする．次段WSの情報をWS定義ファイルから実行時に取得し，動的呼び出しを行う．また，WS定義ファイルから処理時間を取得し，一定時間スリープすることで完成版の処理のシミュレートを行う．複数の次段WSを呼び出す場合は，現在呼び出し中の次段WSの応答を受け取ってから，次の次段WSを呼び出す方式（同期的呼び出し）[4]を採用する．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．

連携CAは，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，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次段WSを呼び出す．連携CAは，実行時刻に関するログを書き出す．WS定義ファイル，連携WS，連携CAのより詳細な情報は，文献[4]を参照されたい．システム概要図を図3に示す．WS-PROVEは，入力としてWS定義ファイルが与えられると連携CA，連携WSを動的に実行し計測結果を出力する．

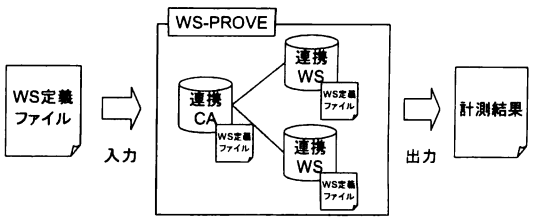

図 3　システム概要図

3.3. システム実装

第3.2節でのシステム設計を基にWS-PROVE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と2つの連携WSの統合WSを例にしたシステム詳細図を図4に示す．これは，2つのWebサービスWS1，WS2および1つのCAをプロトタイプするシナリオである．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[4]に基づくアプローチを採った．これは，呼び出す可能性のある全てのWSを起動するDLL（Dynamic Link Library）とダミーのWSをあらかじめ作成しておく．DLLは，プロトタイプの対象となるWSのWSDL（Web Services Description Language）ファイルから生成する．次に，ダミー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（図4(2)(7)）目的の連携WSを呼び出す(図4(3) (8))というアプローチである．

本システムの実行手順は，以下の通りである．

1. 統合WS開発に必要なWSを準備する．新規にWSを作成，あるいは，既存のWSを検索し，発見する．
2. 各WSのWSDLからプロキシクラスのソースを生成する．
3. 各プロキシクラスのソースコードからDLLを生成する．
4. ダミーWSを必要な数だけ作成する．
5. WS定義ファイルを作成する．
6. WS定義ファイルを入力し，連携CAを実行する．

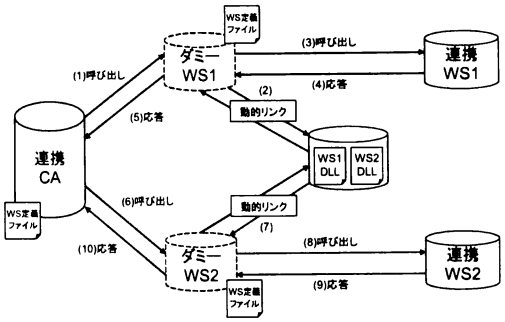

図 4 システム詳細図

4. ケーススタディ

本章では，実装した WS-PROVE を用いて，4 つのトポロジーを持つ Web サービスアプリケーションをプロトタイピングし，システムの有効性を実証する．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

実験では，連携 CA，すべての連携 WS を 1 台のサーバ内に配置する．

4.1. プロトタイピングのシナリオ

実験では，図 5 に示した 4 つのトポロジーのプロトタイピングを行う．連携 CA・各連携 WS には，WS 定義ファイルによって処理時間を設定する．処理時間とは，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．また，連携 CA と連携 WS，連携 WS と連携 WS の間には，ネットワーク遅延を設定する．処理時間，ネットワーク遅延の設定は，以下のように行う．

E1　処理時間：100 ミリ秒
　　ネットワーク遅延：なし
E2　処理時間：100 ミリ秒
　　ネットワーク遅延：表 3 に基づく（例：CA，WS6 間・300 ミリ秒）
E3　処理時間：表 2 に基づく
　　ネットワーク遅延：なし
E4　処理時間：表 2 に基づく
　　ネットワーク遅延：表 3 に基づく

表 2 処理時間の設定（ミリ秒）

CA	WS1	WS2	WS3	WS4	WS5	WS6
100	200	300	400	500	600	700

表 3 ネットワーク遅延の設定（ミリ秒）

	WS1	WS2	WS3	WS4	WS5	WS6
CA	50	100	150	200	250	300
WS1		50	100	150	200	250
WS2			50	100	150	200
WS3				50	100	150
WS4					50	100
WS5						50

各トポロジーに E1 から E4 までをそれぞれ適用し，1 つのトポロジーにつき 4 種類の実験を行う．サービス 1 回の処理時間は，極めて短いため，連携 CA で繰り返し回数を 100 回に設定して処理時間を計測した．計測結果から，連携 CA および連携 WS それぞれの処理時間の合計と平均を得る．

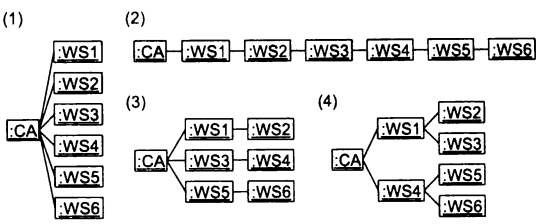

図 5 実験用トポロジー

4.2. プロトタイプの評価

システムは，すべての実験用トポロジーの性能を計測することができた．我々は，先行研究[4]において，1 つの連携 CA と 2，3，4 個の連携 WS で構成されるトポロジーでもシステムの動作を確認している．この結果からシステムは，様々なトポロジーに対応できる．

E1，E3 の動作時間を表 4，表 5 に示す．動作時間とは，連携 CA および連携 WS の個別の処理時間であり，繰り返し回数 100 回の平均値となっている．

表 4 E1 動作時間（ミリ秒）

	(1)	(2)	(3)	(4)
CA	796.70	805.55	805.87	799.10
WS1	106.65	692.66	224.21	337.14
WS2	103.75	575.67	107.56	107.25
WS3	104.53	458.46	224.62	103.58
WS4	103.97	341.14	107.31	337.76
WS5	105.09	224.31	224.77	107.17
WS6	105.10	107.35	107.31	103.68

表 5 E3 動作時間（ミリ秒）

	(1)	(2)	(3)	(4)
CA	2872.81	2884.73	2884.18	2874.22
WS1	200.45	2771.18	520.88	930.40
WS2	307.02	2560.32	310.42	310.28
WS3	400.00	2240.13	912.41	400.11
WS4	494.62	1826.11	497.95	1821.35
WS5	604.57	1318.22	1318.43	607.19
WS6	698.56	701.09	701.11	696.37

各トポロジーの終端に位置する連携 WS は，設定した

処理時間の-6～+11 ミリ秒の範囲に収まっている．また，E1 において，同一トポロジー内で同じ形のノードに位置する連携 WS では，動作時間がほぼ同じである．この結果から連携 CA と各連携 WS は，設定した処理時間通りに動作していることがわかる．次に，E2，E4 におけるトポロジー(3)のネットワーク遅延を表 6 に示す．

表 6　E2・E4 遅延（ミリ秒）

	(3)
WS1	61.70
WS2	101.75
WS3	148.73
WS4	195.56
WS5	242.27
WS6	304.59

	(3)
WS1	61.69
WS2	101.72
WS3	148.68
WS4	195.55
WS5	242.40
WS6	304.78

ネットワーク遅延は，設定の-8～+12 ミリ秒の範囲に収まっている．他のトポロジーの遅延についても同様の結果が得られた．これらの結果から設定したネットワーク遅延通りに動作していることがわかる．次に，E2，E4 の動作時間を表 7，表 8 に示す．

表 7　E2 動作時間（ミリ秒）

	(1)	(2)	(3)	(4)
CA	1868.20	1181.22	1433.49	1385.17
WS1	107.55	1005.65	286.87	505.84
WS2	107.55	826.05	107.47	107.60
WS3	107.62	646.48	287.02	107.60
WS4	107.51	466.50	107.73	505.76
WS5	107.53	287.07	287.00	107.68
WS6	107.53	107.56	107.47	107.59

表 8　E4 動作時間（ミリ秒）

	(1)	(2)	(3)	(4)
CA	3946.32	3259.40	3514.26	3463.70
WS1	201.37	3083.96	583.83	1099.52
WS2	310.62	2810.42	310.75	310.75
WS3	404.46	2427.75	974.40	404.48
WS4	498.20	1951.10	498.24	1990.09
WS5	607.61	1380.93	1380.78	607.63
WS6	701.25	701.34	701.45	701.42

遅延を考慮しても，各トポロジーの終端に位置する連携 WS は，設定した処理時間の-3～+11 ミリ秒の範囲に収まっている．また，E2 においても，同一トポロジー内で同じ形のノードに位置する連携 WS では，動作時間がほぼ同じである．これらの結果からも連携 CA と各連携 WS は，設定した処理時間通りに動作していることがわかる．

5. 考察と今後の課題

開発した WS-PROVE を第 3.1 節の要件(R1)～(R4)の実現という観点から評価する．(R1)については，システムが特定の WS に限らず利用できることや WS 定義ファイルによって変更が可能なことから実現されている．(R2)については，プロトタイプのトポロジーを WS 定義ファイルによって自由に変更できることで実現されている．(R3)については，第 4.2 節の結果からもわかるように，満足な性能を確認できていることから十分である．(R4)については，システムに GUI が実装されていないことやダミーWS を手動で作成しなければならないなどシステムの一部が自動化されていないことから不十分である．

従来のプロトタイピングでは，開発の迅速さが要求されるために性能が無視される．しかし，Web サービスアプリケーションにおけるプロトタイピングでは，性能を重視しなければならない．これは，従来のプロトタイピングが Web サービスアプリケーションに適用できないことを示唆している．今後，Web サービスアプリケーションへ適用可能なプロトタイピングモデルの提案が望まれる．現状では，性能を考慮してプロトタイピングを行うならば，WS-PROVE が機能型プロトタイピングに当てはまることになる．

今後の課題としては，実現されていない要件についてシステムへの実装を進める．具体的には，作業の自動化を含め，機能の充実を図り，使いやすいシステムにすることが挙げられる．将来的には，Web サービスアプリケーション開発方法論についての研究へとつなげていきたい．

文　献

[1] 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.

[2] 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.

[3] GeOap，http://www.geoap.jp

[4] 石井健一，串戸洋平，井垣宏，中村匡秀，松本健一，“複数 Web サービス連携手法の実験的評価”，信学技報，IN2004-58，pp.75-80，Sep.2004.

[5] 石川冬樹，田原康之，吉岡信和，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.

[6] 片山卓也，土居範久，鳥居宏次，ソフトウェア工学大辞典，朝倉書店，1998.

[7] 中島毅，田村直樹，岡田和久，和田雄次，“顧客対話時の要求獲得と確認を支援するプロトタイピングツール：ROAD/EE”，電子情報通信学会論文誌，Vol.J85-D-I，No.8，pp.725-740，Aug.2002.

[8] 大西淳，郷健太郎，要求工学，共立出版，2002.

[9] 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.

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

[11] 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/