実システムへのWebサービス技術適用性検証

製造・流通システム事業部
製造・流通システム開発部

奥村　公治

Koji　Okumura

エンタープライズソリューション事業部
ECM・EIPソリューション部
ITスペシャリスト

金子　文徳

Fuminori　Kaneko

今、経済環境は、激しく変化し、より一層厳しくなっている。この中で各企業は、タイムリ且つ安価に変化に対応できる情報システムを模索している。WebサービスはSOAP、UDDI、WSDLをはじめとするオープンな技術をベースとして、この課題を解決できる有力な手段になりつつある。そこで当社では、技術力強化のひとつのテーマとして、Webサービス技術を実システムに適用する上で、考慮すべき点は何であるかを調査した。Webサービス技術は、サーバ間の通信としてすぐに且つ安価で使用できる技術である。しかし、実際にシステムを構築する場合には、既存の他技術と比べてもいろいろな機能を作成しなければならない。そこで、模擬システムを作成し、必要とされる機能をどのように実装するかの利用技術を蓄積してきた。

本文で、模擬システムの内容と課題、実装結果の評価などについて報告する。なお、本文の５章は、日本IBM株式会社主催の合同プロジェクトによる成果である。

IBM合同プロジェクト参加企業（７社）

日本アイ・ビー・エム株式会社
株式会社ソルネット
株式会社ティージー情報ネットワーク
日進ソフトウェア株式会社
株式会社ライトウェル
株式会社ラック
株式会社エクサ

1. はじめに

現行Web技術の限界を打ち破る新しい標準化技術であるWSDL（Web Service Description Language）、SOAP（Simple Object Access Protocol）、UDDI（Universal Description, Discovery and Integration）から成る「Webサービスの実装基盤（以降、「新技術」と表す。）」整備が着々と進み、さらなる企業間連携の効率化促進や新しいビジネスモデル誕生の可能性が高まっている。

「新技術」利用知識を深めた企業はまだ少なく、顧客およびSI'erは競争力拡大の起爆剤として「新技術」適用に先手必勝を掲げた企業と、「新技術」適用に慎重な企業とに分かれている。熾烈な競争の中で勇み足とならないためには、「新技術」適用性の正確な把握が必要である。当社では、実用規模の模擬システム構築により「新技術」適用の有効性についての実証実験を行った。本文でその検証内容と検証結果について詳述する。

2. Webサービスの可能性

Webサービスについては、種々の論文や記事などで詳しく説明があるので、サービス利用者（リクエスタ）、サービス提供者（プロバイダ）、サービス紹介者（ブローカ）が、何をするか、また、標準であるWSDL,SOAP,UDDIについての説明は割愛する。

ここでは、Webサービスの可能性について、既存技術との対比の上で、詳しく検討する。

Webサービスの可能性については、次のものが挙げられる。

① Webサイトの連携
② アプリケーションの統合
③ ビジネスプロセスの統合

これらに関係する既存の技術は、HTML、CORBA、EAIであり、これらとWebサービスを比較する。

2. 1. HTML（Hyper Text Markup Language）との比較

現行のHTMLベースの連携では、表示タグとデータタグとの明確な分離を言語上、行うことができない。このため提供者（プロバイダ）側ですべての処理をする必要があり、特に、利用者（リクエスタ）側の表示形式にあわせるカスタマイズは大きな負荷となる。

また、利用者による表示の変更は提供者へも変更が波及し、顧客要求に応じた柔軟な対応が難しい。

一方、Webサービスでは、表示部は利用者がロジック部は提供者が持つため、利用者による自由なサイト構築ができる。同時にインタフェースが統一されるため、提供者は複数チャネルへの提供がコスト増を伴わずに実現できる。また、利用者も複数の提供者との接続がはかれる。詳細は、４章にて説明する。

2. 2. CORBA（Common Object Request Broker Architecture）との比較

CORBAとWebサービスのコンセプトと比較を図１、表１に示す。いずれもRPC（Remote Procedure Call）を具現化するものであり、両者とも似通った技術である。しかし、CORBAの方が以前からある技術であるため、トランザクション管理、セキュリティ仕様を既にもっている。また、通信もテキストだけでなくバイナリで行えるため、ネットワークに負荷をかけないものになっている。ただし、実装には訓練と専門知識、プログラミングが必要である。

一方、Webサービスはインターネット向きであると同時に、インターネット技術対応のサーバやネットワーク基盤があれば実装できる。プログラミングについても、オブジェクト指向以外に特別な専門知識を必要としない。

表１　CORBAとWebサービスの比較

	CORBA	Webサービス
相互接続性		
異種プラットフォーム間	○	○
異種開発言語間	△	○
トランザクション	あり	発展途上
セキュリティ	あり	発展途上
パフォーマンス		
送受信パケットサイズ	小	大
通信処理	軽い	重い
XML文書	×	○
バイナリ通信	あり	なし
開発生産性		
インタフェース実装コスト	大	小
ミドルウェア価格	大	小～大
安定性	○	△
ツールの種類	少	多
運用保守性		
サーバ	CORBAサーバ	Webサーバ
インタフェース変更	影響大	影響小
TCO	高価	安価

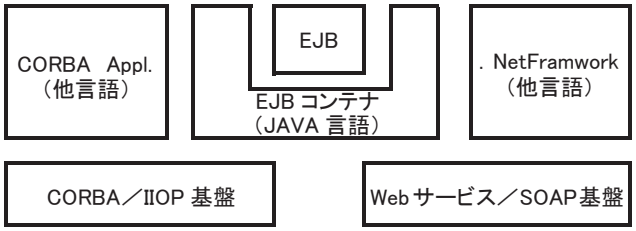

図１　CORBAとWebサービス

2. 3. EAI (Enterprise Application Integration) との比較

EAIツールの真価は、まずアプリケーション統合を自律化し可視化するというところにある。EAIサーバ上へアプリケーション統合にかかわるすべての機能と情報を集中させることで、統合の関係や構成をEAIサーバ上で完結させることができる。個々のアプリケーションは、他システムとの連携や統合の関係を意識する必要がなく、その関係はEAIサーバを修正するだけで追加・変更することができる。

EAIツールの多くは、アプリケーション連携を担うだけではなく、ビジネスプロセスの管理と自動化を実現するBPM（Business Process Management）機能を装備している（図２参照）。

現在のところWebサービスには、これに相当する機能はない。しかし、ビジネスの機能をサービスとしてネットワーク上に公開し、それを他者が使用することに適した技術である。このため、新しいビジネスプロセスを定義して、それに必要なサービスを組み合わせていくことでビジネスプロセスの統合が出来るようになる（表２参照）。

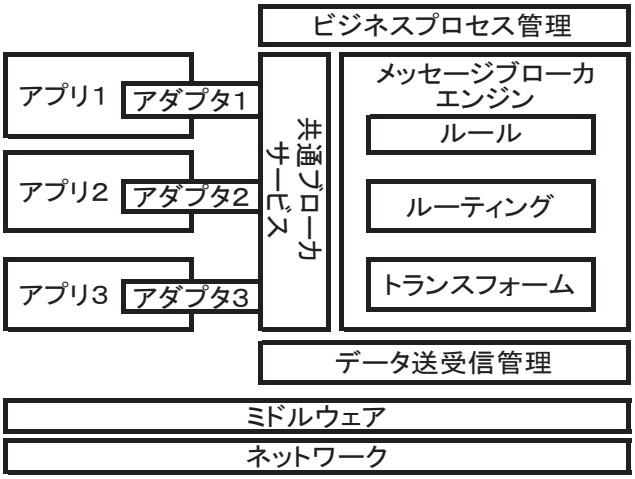

図２　EAIの機能模式図

表２　EAIとWebサービスの比較

	EAI	Webサービス
相互接続性		
サーバ（リクエスタ）	自律	作り込み
他システム	アダプタ	プロバイダ化
異種プラットフォーム間	ツール依存	○
トランザクション	あり	発展途上
セキュリティ	あり	発展途上
BPM機能	あり	発展途上
開発生産性		
安定性	○	△
ツールの種類	ツール依存	多
運用保守性		
インタフェース変更	サーバで吸収	影響小
TCO	高価	安価

2. 4. Webサービスの課題

既存技術からWebサービスの機能を比較すると、次の点で何らかの対応が必要となる。

① トランザクション管理
② セキュリティ
③ サービスの統合

これらの対応は、標準化の推進、アプリーションの中に独自構築、SOAPヘッダの拡張などで対応できるとされている。そこで今回の技術検証のおいては、それらについてどのように対処し、模擬システムの中に実装したかを以下に述べる。

3. Webサービスの仕組み

模擬システムを理解するために、Webサービスの仕組みについて説明する。図３は、JAVA/AXISベースのRPCを示す模式図である。この模式図の中で、Webサービス化の手順と１回のHTTP通信の要求と応答を示す。

3. 1. プロバイダサーバ（サービス提供者側）

通常通りにプログラムを作成する。公開したいメソッドを決める。この単位をサービス粒度といい、どの程度の大きさにするかは、ノウハウである。後は、AXISが搭載されているサーバの然るべきディレクトリに中にこのプログラムを置き、server-config.wsdd（WSDD：Web Service Deployment Descriptor）ファイルの中に、Webサービス名とこのプログラム名、公開するメッソドを記入する。これで終了である。

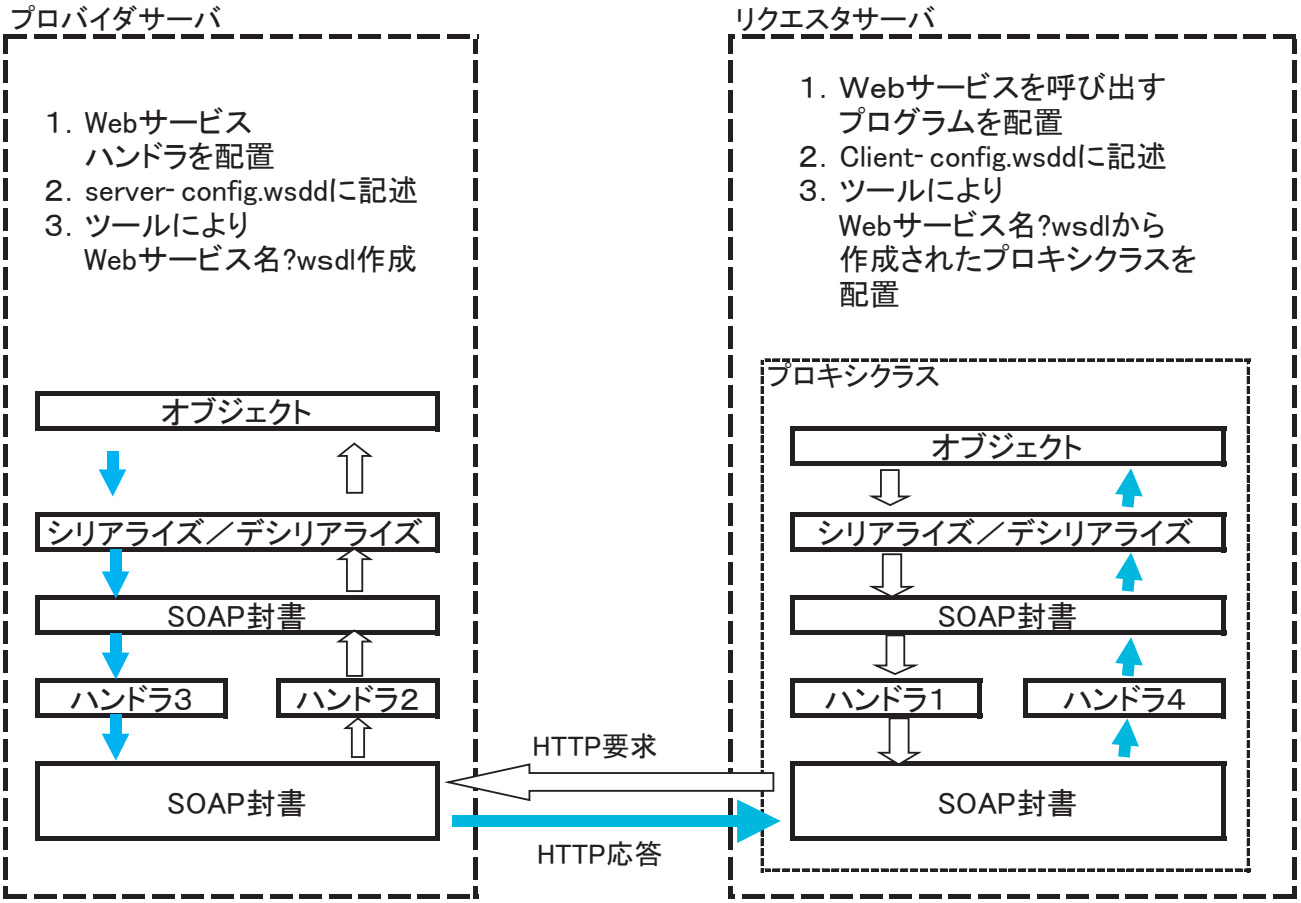

図３　Ｗｅサービス模式図

この裏では、外部からURI=XXXXXXXXXXXXXX/Webサービス名?WSDLで参照できるように、WSDLが作成されている。したがって、EJB（Enterprise JavaBeans）があれば、server-config.wsddに記述するだけで、Webサービスとして公開できる。

通信に使用されるSOAP封書に対しては、ハンドラというプログラムを作成できる。このハンドラのために、SOAP封書に対するAPIが用意されていて、これを用いることでSOAP封書に対する加工が行える。このハンドラプログラムも上記のディレクトリの中に置かれ、名前は、server-config.wsddに定義される。この定義の中で、SOAP封書を受け取った時、出す時など、きめ細かくハンドラの適用を定義する。

3.2.　リクエスタサーバ（サービス利用者側）

通常通りにプログラムを作成する。そこで、公開されているメソッドが必要な場合、Webサービスを使用する。実際には、上記のURI=XXXXXXXXXXXXXX/Webサービス名?WSDLをAXISツールに渡し、そのツールがプロキシクラスを自動生成（スタブの生成）する。

このプロキシクラス（プログラム）の中に、メソッドが定義されているので、自由に使用することができる。ハンドラに関しても、前述と同様な方法が適用できる。

3.3.　リクエスタサーバとプロバイダサーバとの繋がり（SOAP通信）

リクエスタサーバとプロバイダサーバには、同じオブジェクトが存在している必要がある。このためには、オブジェクトをSOAP封書にし、また同じオブジェクトに再現できれば良い。これを行うものが、シリアライザ（オブジェクト⇒SOAP封書）とデシリアライザ（SOAP封書⇒オブジェクト）である。

一方向の通信の中で、オブジェクト⇒シリアライザ⇒SOAP封書⇒ハンドラ⇒SOAP通信⇒ハンドラ⇒SOAP封

書⇒デシリアライザー⇒オブジェクトの流れが行われている。したがって、プログラムを書く人にとって、良いか悪いかは別にしてネットワークのことは完全に隠されてしまうことになる。

4. Webサービス技術検証と検証結果（その１）

4.1. 宿検索予約システム

検証用模擬システムとして、旅行業者向けフロントエンドシステムに近い規模のデータベース構成、処理ロジック、表示画面遷移および画面構成により、宿検索予約システムを構築した。

新技術の適用により、Webサービスの仕組みついての知見を深めると同時に、下記課題の解決策について「動作安定性」「開発生産性」「実行時パフォーマンス」の観点から、有効性を検証した。

①複合型の引数、返値を持つメソッドのWebサービス公開と利用

②複数サイトに存在するWebサービスの集約的な利用（UDDI利用）

③アプリケーションの中にトランザクション管理の実装

図４に現状想定できるWebサイトのイメージを、図５に模擬システムにおけるWebサイトのイメージを示す。

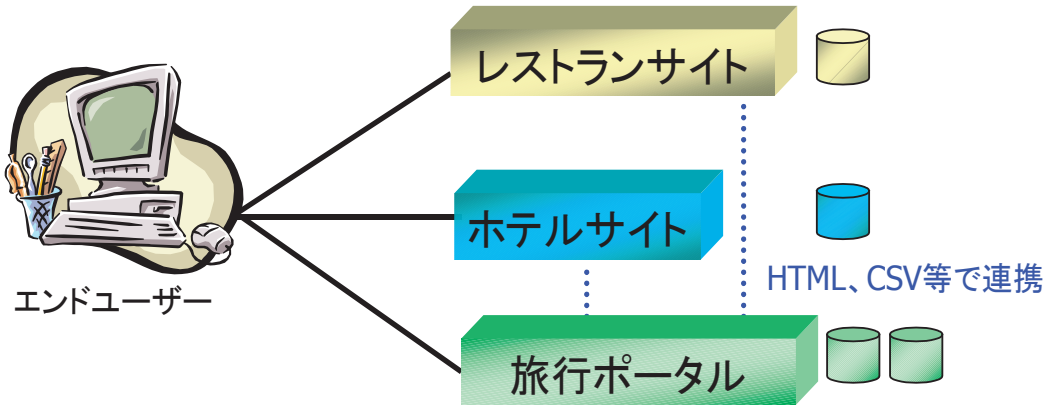

図４　現状のWebサイト連携イメージ

連携しない場合は、全て自前で準備

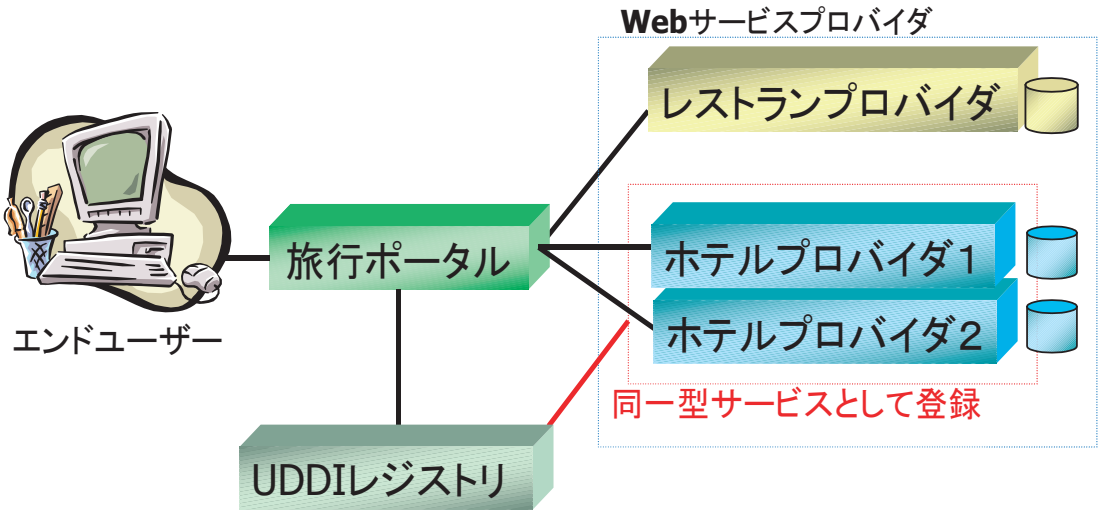

図５　Webサービスを使用したWebサイト連携（模擬システム全体図）

4.2. UDDIを用いた宿検索サービス集約

Ｗｅｂサービスの公開と利用に関して、Ｗｅｂサービスのネーミングサービスである UDDIレジストリを利用すれば、接続形態を一対多の動的バインディングとすることができるので、より大きな利用価値が期待される。今回の実装では、次の手順でUDDIを利用した。

① サービス提供者は、統一インタフェースでWebサービス公開する。

② 提供サイトの「サイト情報」、「サービス情報」、「インタフェースの実在場所」をUDDIレジストリ構造（図６参照）の「businessEntity」、「businessService」、「bindingTemplate」として登録する。この時、参照先となる「tModel」のキーを同一値で登録する。

③ DIへはtModelキーによる問い合わせを行い、レジストリ中に存在するサービス提供者の実在場所を動的に取得する。

サービス利用者である宿検索ポータルサイトでは、統一インタフェースのWSDLより汎用ツールを用いて実装言語用のプロキシクラスコードを自動生成後、バインディング情報部分はUDDIから動的に取得する情報で実行時に置き換えができるように実装した。

図７は、宿検索ポータルサイトのフロー図を示す。

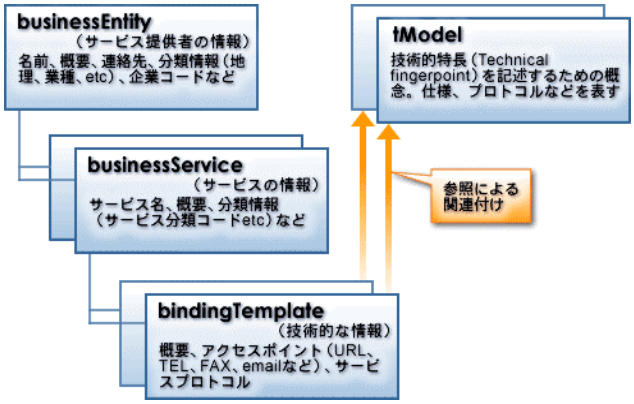

図６　UDDIレジストリ構造

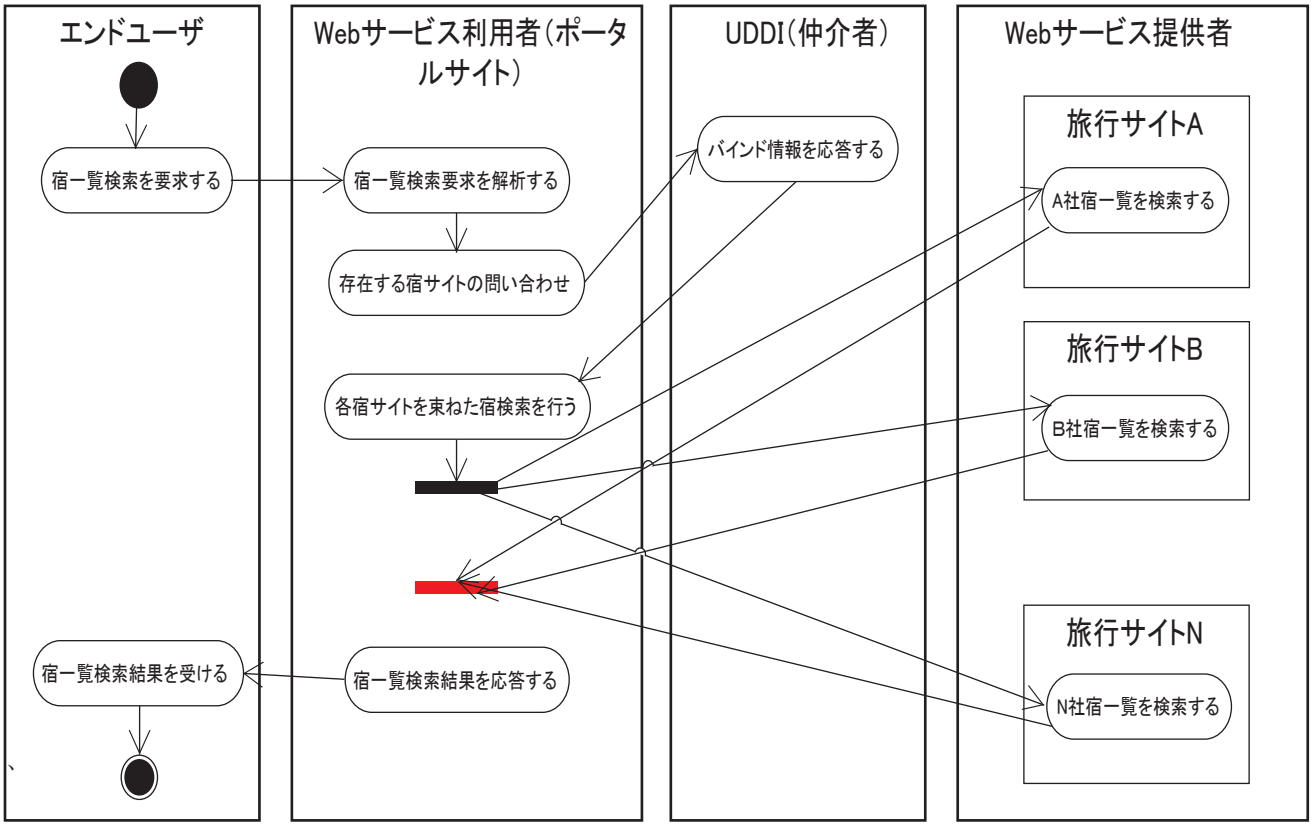

図７　限定的な動的バインディング例

4.3. 統合予約とトランザクション管理

Web公開されたサービスを組み合わせて一つの大きなサービスを構成する場合、部分サービスの実行結果に従って、次処理の分岐管理およびサイト間の状態管理を行うような全体トランザクション管理の仕組みが必要になる。

こうした動作を検証するために単純化したモデルとして、以下のサイト間トランザクション構成を設定する。ポータルサイトとして「旅行商品の統合予約」を提供するものとし、宿サイトの「宿予約」とレストランサイトの「レストラン予約」を組み合わせて一つの予約とするような状態管理を行う。たとえば「レストラン予約」が成功しても「宿予約」が失敗すれば、ポータルサイトがレストランサイトに対して「レストラン予約キャンセル」を要求し、全体の整合性を制御する（図８参照）。

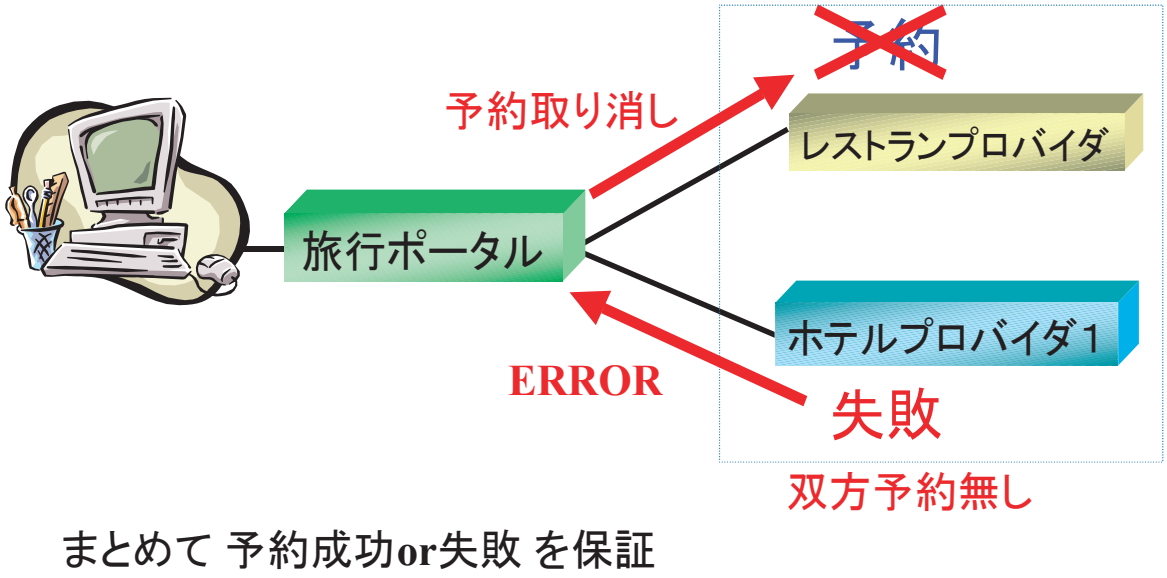

図８　トランザクションの整合性保証

4.4. 検証結果

今回開発した模擬システムは、下記の環境にて検証を行った。

- Windows2000+SP2
- WebSphere Application Server V4. 03AE
- （ApacheAXIS/WSIF用JARを別途組み込み）
- DB2 V7. 2
- IBM UDDI V1. 1. 1
- WebSphere Studio Application Developer Integration Edition V4. 1. 1

（１）検索機能のWebサービス化と高い接続性

実用の宿検索は「宿一覧検索」と「宿詳細表示」の２段階に分かれる。宿一覧検索の応答データはリストデータであり、各宿が複数プランを包含する場合、データ構造はさらに複雑になる。このような複雑な型である複合型を出力に持つサービスであっても、実際はオブジェクトクラスの形式で出力されるため、ローカルに存在するコンポーネントと同じように利用できる。

今回、提供者側で公開した複雑な応答データを返す宿一覧検索Webサービスにおいても、安定した通信動作を確認できた。上記JAVA言語用ツールはWSDLとJAVAコードとを相互変換するが、通信部分は言語に依存しないXMLベースに変換されるため、WindowsやC#など、どんなプラットフォーム、プログラム言語にも依存しない通信が可能となる。したがって、言語などに依存する場合に必要であった通信処理部のコストと期間に対して９割以上削減可能となり、その分のマンパワーを有効な領域へ向けることができる。この点においては顕著な生産性向上が見込める。

（２）データと表示の分離による接続柔軟性の向上

XMLをベースとしたWebサービスをインタフェースに適用することで、サービス要求に対する応答はシステムリーダブルなデータのみとなる。データ部と表示部の依存性がなくなるので、利用者側は自由自在に表示部をカスタマイズでき、サービス呼出によりデータを容易に取得できる。提供者側はWebサービス公開メソッドについてイニシャルコストを投資しただけで、次回以降の新たな利用者への対応コストが大幅に削減できる。さらに、サイト連携を迅速かつ低コストで実現するUDDIの利用もおこなえる。今回のUDDI利用は予め統一インタフェースを設定し、規定のtModelキーを登録するという限定的な使い方を前提にしている。このような使用方法であっても接続拡大のために役立つと認められる。

（３）パフォーマンス面の考慮

公開サービスの単位を細かくすればパラメータ条件は単純となるが、呼出回数が増えるため必然的にパフォーマンスが落ちる。一方、単位を粗くすれば呼出回数は減らせるが、パラメータ条件指定が複雑になるという関係があり、インタフェース設計時の重要検討課題となる。

WebサービスがXMLをベースとしている以上、現行のテキストやバイナリベースの通信プロトコルと比較して通信パケットサイズが膨らむ。標準化団体により考案中のSOAPバイナリ転送標準や通信路の容量増大で解決されていく可能性が高い。今回検証したケースでは、XMLスキーマタイプ指定の省略により通信パケット量を３割ほど削減させることができた。RPCベースの通信ではWSDL取得時にクラス内の型をあらかじめ認識できるため、SOAPパケット上からはクラス内の変数型についてはスキーマタイプを省略することができる。

5. Webサービス技術検証と検証結果（その２）

5.1. ビデオ協会支援システム

レンタルビデオ店向けサービスをビデオ協会のようなところが行う時、必要とされるサービス（ブラックリストサービス、認証／課金サービスなど）を他社からの提供で充当するビデオ協会支援システムを、模擬システムとして構築した。このシステムを用いて、下記の課題に関する解決策の有効性を「動作安定性」「実行時パフォーマンス」の観点から検証した。

①Webサービス公開の容易性

②複数サイトに存在するWebサービスの統合利用

③End-To-Endでのセキュリティの確保

図９に模擬システムの全体構成のイメージを示す。

5.2. Webサービスの統合利用

今回のケースでのサービス統合のシナリオは、次のようになっている。

レンタルビデオ店がビデオ協会にアクセスして顧客の情報を参照すると、ビデオ協会はサービスプロバイダで提供しているブラックリストサービスを呼び出す。

サービスプロバイダでは、アクセスしているレンタルビデオ店が登録されているか、認証サービスを呼び出して確認する。アクセスが認められると、サービスプロバイダは課金サービスを呼び出し、アクセスログを記録して、ブラックリストサービスを実行する。

結果をビデオ協会に返す前に課金サービスを呼び出して、

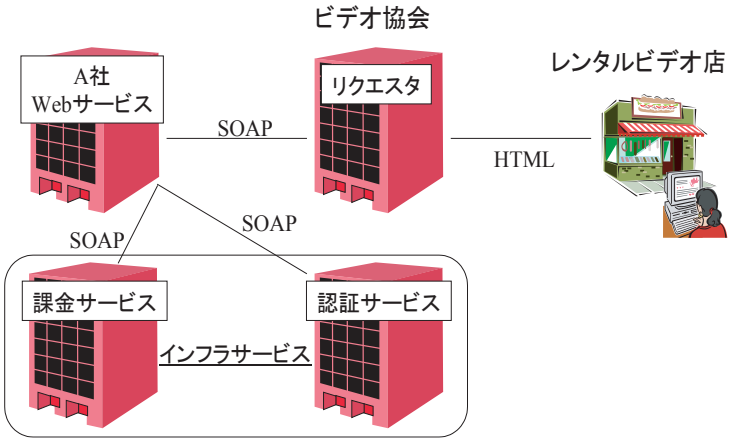

図９　模擬システム全体イメージ図

アクセスが終了したことを記録する。ビデオ協会に結果が届くと表示用に変換して、レンタルビデオ店に結果出力をする。なお、これら一連の処理は、レンタルビデオ店からの１回のHTTP要求により行われる。

認証・課金サービスは、ハンドラ機能を利用することで、ブラックリストサービスに影響を与えることなく、サービスの共通的な機能として作成した（図10参照）。

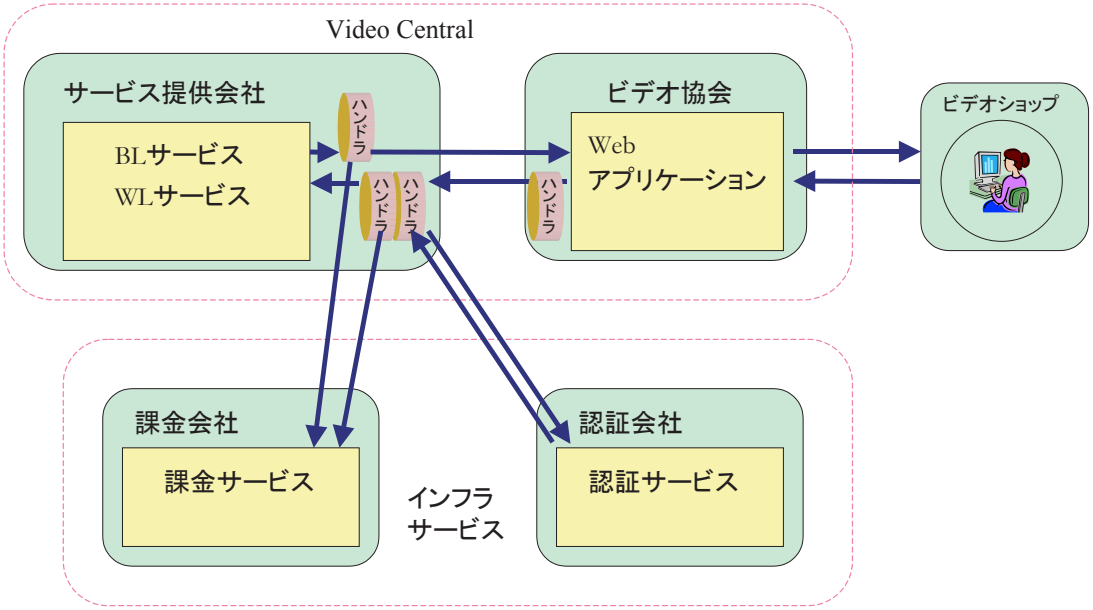

図 10　模擬システムにおけるサービス統合の仕組み

5.3.　End-To-Endでのセキュリティ確保

通常のＳＳＬでは通信路上のPoint-To-Pointのセキュリティしか確保できない。
したがって、Webサービスで複数のサーバ間でSOAP封書をやりとりする場合には、別途封書中のメッセージに対してEnd-To-Endのセキュリティを考慮する必要がある。今回はXML署名、XML暗号という技術を使用してこれを実現した。

実際には、ビデオ店のユーザ情報は、ビデオ協会と認証サービスがわかっていれば良い情報である。途中のBLリスト提供サービス会社では読めない方が良い。この情報は、SOAP封書のヘッダ部に格納していた。この部分に関して署名・暗号化を行い、通過するサイトでは、ユーザ情報が判らないようにした。

5.4.　検証結果

（１）Webサービス公開の容易性

ビデオ協会支援システムは、DBシステムの応用例として、サーバ／クライアント型で開発されていたものを、Webサービス化したものである。このソースコードは自由にダウンロードできるので、今回の模擬システムで必要なサービスは、すべて新しいメソッドとして作成し、上記に追加した。実際には、普通にメソッドの仕様を定義し、実装するだけであり、容易にWebサービスの作成・公開ができた。さらに、当初このアプリケーションはApache SOAP2.2を使用して作成されていたが、ハンドラを使用するためにApache Axisに変更した。この変更も容易であった。

（２）Webサービスの統合利用

ハンドラを利用することによって、一回のHTTP要求で、複数のサービスを統合することができた。この場合、注意すべき点は、あるサービスでエラーが起きたときの処理（例外処理）に関して対処が必要になる。ハンドラの動作を調べて、例外処理を行う必要がある。

（3）パフォーマンス面の考慮

今回の検証の中で、署名・暗号化の処理を行っている。この処理がパフォーマンスに与える影響について評価した。１回目の処理では、前処理などのためかなり処理時間を要した。しかし、２回目以降は、片方向で、0.5秒程度になった（暗号化⇒複合化）。

したがって、常時稼動しているサイトでは、１回目は無視できるので、特別な配慮の必要はない。通常、サイトにおけるパフォーマンス面の考慮（アクセス数、処理時間、サーバ数など）にはそれほど影響を与えないと判断した。

なお、処理時間は下記環境にて実測した。

- WindowsXP
- IBM　JDK1.3.0、XSS4J（2002/04/25版）
- CPU　Pentium4.1　1.8GHZ
- RAM　1GB
- HDD　IDE

6. 終わりに

Webサービス技術の本質は、企業内のコンピュータシステムと人的システムを含めた総体に対して外部への標準的な通信口を設けることにある。企業内部処理は隠蔽され、外部へは自動化されたコンピュータシステムとしてブラックボックスとなる。

したがって、一旦公開認知されると、外部に対して安定したサービスを提供する義務と責任を担うことになり、使いにくいサービス、安定供給できないサービスは企業名と共に淘汰されてしまう恐れを孕んでいる。

この事情から公開に関しては、検証作業を経て確実になった技術から段階的に適用を推進することが大事である。現在のWebサービスの適用に関して、つぎのように考えている。

① Webサービス公開と利用における静的バインディンは成熟レベルにあり、高い適用効果が見込めるため、適用実施は現実的である。ただし、インタフェースの公開は部分的な企業グループ内の範囲から始めて、徐々に広げていくアプローチが望ましい。

② 動的バインディングやトランザクション管理の利用は、現在進められている標準化活動（グローバルUDDIやトランザクション系標準）の推進と実装により容易になると想定できる。現状は既存ミドルウェアの併用と必要最小限のアプリ実装により補完する。

ビジネスプロセスの統合への適用は、既存のEAI機能と比べても時期尚早である。しかしEAIのハブ・スポーク的な統合に対して、サービスを集約してビジネスプロセスを構築する考えは現実に即しているので、標準化活動（BPEL4WS*、WS-Coordination、WS-Transaction）に合わせて、当社としての技術力を強化する予定である。

*BPEL4WS（Business Process Execution Language for Web Services）

参考文献

日本アイ・ビー・エム株式会社、JStartチーム、「最新Webサービスが分かる」、技術評論社、2002

＜問い合わせ先＞
製造・流通システム事業部
製造・流通システム開発部
製造・流通第５チーム
Tel.　044-246-5342　奥村　公治
E-mail：koji-okumura@exa-corp.co.jp

本文中の会社名、製品名、およびサービス名などはそれぞれ各社の商標または登録商標です。