Webサービス本格活用のための設計ポイント

野間克司
福原信貴

要旨

Webサービスはインターネット上に分散するシステムを連携する仕組みであり、SOAP、WSDL、UDDIといった標準化された技術を利用して、システム連携を実現する。将来的にはWebサービスの活用により、企業内システムの共用化／一元化、及び、外部提供サービスの活用が進み、企業はIT投資をよりコアコンピタンス業務に集中させることが可能になると考えられている。一部の先行企業においては既にWebサービスの導入が始まっているが、内容的にはまだ試験導入段階のものが多い。本論では、セキュリティやトランザクション管理などについての課題が指摘されている中で、どのような箇所からWebサービスの適用を開始し、設計／開発時にはどのような点に留意すればよいかについて明らかにする。

キーワード：Webサービス、SOAP、WSDL、UDDI、XML、再利用、セキュリティ、セレクティブアウトソーシング

The Web Services are a system which links up the scattered systems on the Internet, leveraging standardized technology such as SOAP, WSDL and UDDI. It is a general thought that in the future business enterprises will be able to concentrate investments for IT more on their core competences operations as the result of sharing and/or centralizing of in-company systems as well as utilizing outsourced services by the Web Services. Some precedential corporations have already started implementing the Web Services, however, much of the content is yet within the extent of experiment. As some issues regarding security and transaction managements, etc. have been brought up, in this paper we will state where you should start the Web Services application from and what kind of points need to be examined during designing and/or development.

Keywords：Web Services, SOAP, WSDL, UDDI, XML, Reuse, Security, Selective Outsourcing

１．はじめに

インターネット上に分散するシステム同士をつなぐ新しい接続方式としてWebサービスが注目されている。従来からのWebシステムは、Webブラウザを使用して人が利用することを前提としており、システム同士で接続することを想定して設計されていなかった。しかし、Webシステムが増えるにつれて、Webシステム上のアプリケーション同士を簡単に接続し、有効活用したいというニーズが強まってきた。このような状況の中で新たに登場したモデルがWebサービスである。

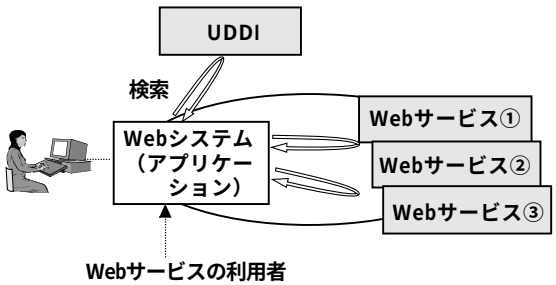

図１　Webサービスの利用イメージ

Webサービスは、インターネット上で公開された多様なサービス（アプリケーション）をニーズにあわせて容易に検索・発見し、結合するだけで、簡単に利用できるようにすることを目的としている。Webサービスの基本モデルは、サービスを提供する「サービスプロバイダ」、サービスを利用する「サービスリクエスタ」、サービスの登録・検索サービス提供する「サービスブローカ」の三者から構成される。Webサービスを実現するための技術要素としては、接続プロトコルのSOAP（Simple Object Access Protocol、以降「SOAP」）、サービス情報を記述するためのWSDL（Web Services Description Language、以降「WSDL」）、サービスブローカであるUDDI（Universal Description, Discovery, and Integration、以降「UDDI」）の三つが基本になっている。これらの技術は全く新規に作られたものではなく、従来からのインターネット技術であるTCP/IP、HTTP、XMLをベースにして作られたものであり、サービス連携を実現するために上位のプロトコルとして策定されたものである。

Webサービスは、インターネット上に公開されたシステムを連携する場合だけでなく、企業内のシステムを連携させる場合にも非常に有効な手段となる。Webサービスの活用により、企業内のシステムやデータの共用化／再利用化が進められるからである。

Webサービスの実装をサポートする製品やツールも出揃いつつあり、先行ユーザ企業においては導入が始まっているが、位置付けとしては、試験的な導入によりWebサービスをどのように適用すればよいのかを見極めている段階といえる。

これは、Webサービスについて、

・セキュリティやトランザクション等について、仕様が十分に固まっていない箇所がある。
・UDDIサービスの登録内容についての保

障の考え方が明確になっていない。

といった仕様に関する課題が指摘されている状況であり、企業においては、

・Webサービスをどのようなシステムから適用していけばよいかわからない。
・障害対応、拡張対応、性能対策、テスト方法等の方式設計手法がよくわからない。
・どのようなシステム設計を行えば再利用されやすいシステムを作れるのかよくわからない。

といった、開発段階での懸念が解消されていない状況にあるからである。

NRI（野村総合研究所）では、Webサービスの本格的な企業システムでの実用化が始まる前に、これらのWebサービス構築時における懸念の解消を目的として、技術的な検証を行い、設計時における留意事項をまとめ、Webサービス適用のための方針作りを行ってきた。

本論では、このWebサービスの主要技術について簡単に紹介した後、Webサービスをどのように適用していけばよいかについて説明し、方式設計とアプリケーション設計各々の観点から留意すべき点を明らかにしていく。

2．Webサービスの主要技術

最初にWebサービスの主要技術要素であるSOAP、WSDL、UDDIについての概略を説明する。

（1）SOAP

Webサービス関連技術において、最も重要な技術要素といえるのがSOAPである。

既存のWebシステムは、人とシステムが会話することを前提に作られており、Webブラウザからのリクエストに対して、その結果をHTML形式で返す仕組みになっている。HTMLは、結果を人が見て理解できるようにレイアウト情報＋データから成り立っているが、SOAPの場合、データとそのデータの意味づけ（メタ情報）で構成されるXML形式の情報を提供する。

HTMLが広く利用されるようになった理由の一つは、標準化団体によって標準化された仕様であり、手軽に誰でも利用できたとい

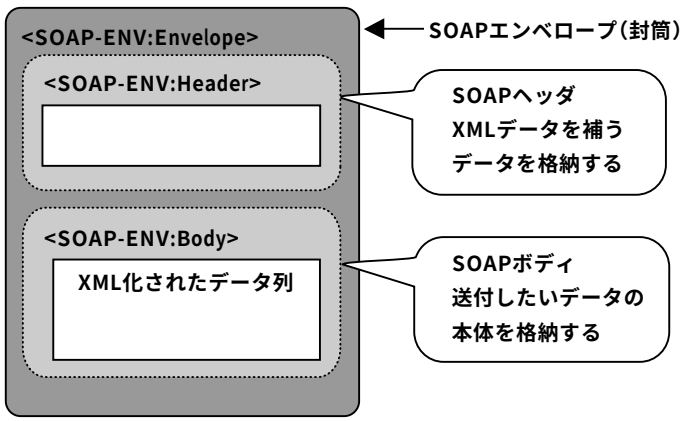

図２　SOAPメッセージの構造

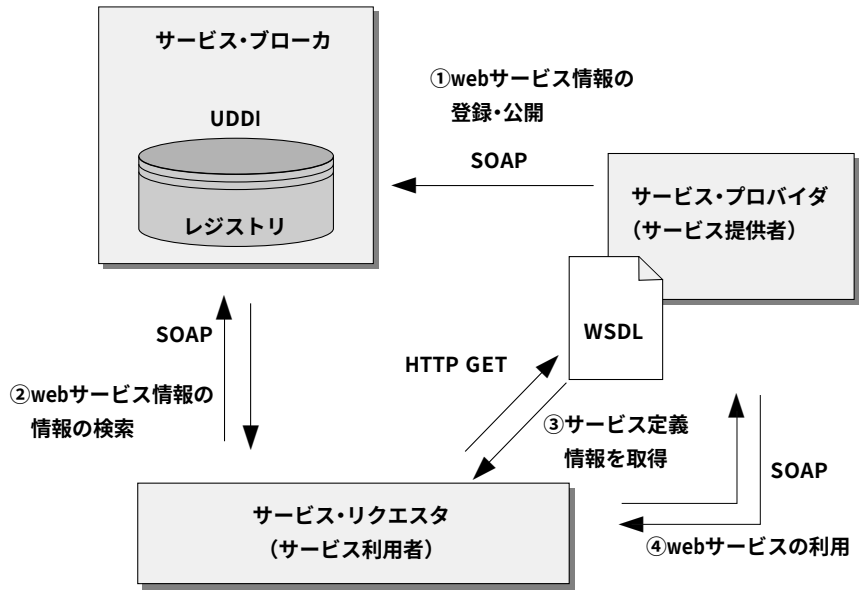

図3　SOAP、WSDL、UDDIの関係

うことであるが、SOAPにも同様のことがあてはまる。SOAPはWWW関連の標準化団体W3C（World Wide Web Consortium）によって制定された仕様である。

SOAPは仕様として規定している部分は少なく、構造的にはユーザ定義情報を格納する「SOAPボディ」、付加的に処理に必要なデータを格納する「SOAPヘッダ」、そして全体を包含する「SOAPエンベロープ」からなっている。「SOAPボディ」中に格納するデータの構造は、ユーザーが自由に定めることができる。メッセージとしてXML文書を用いることも当然可能であるが、XMLを意識しないでも簡単に実装できるように、RPCとして利用する場合の利用方法についても仕様書に示されている。このRPCモデルを多くのSOAPミドルウェア実装製品がサポートしているが、仕様の解釈の違いから、接続性の問題を発生させることがあるので、注意が必要になっている。SOAP仕様は、現在V1.2を制定中である。

（2）WSDL

システム同士を接続するには、お互いが事前に接続インタフェースを取り決めておく必要がある。Webサービスについても同様であり、このインタフェースを記述するために制定された仕様がWSDLである。この定義情報にはWebサービスとして公開されているメソッド（RPCの場合）、あるいはやり取りされるXMLの構造（XMLメッセージングの場合）、アクセスポイント情報等が記述される。WSDLにはインタフェース定義情報が過不足なく記述されているため、WSDLを元に実装の雛形を作成することも可能である。このため、サービスへの要求を出すWebサービスリクエスタ、あるいはサービスを提供するWebサービスプロバイダのスケルトンコード（スタブやプロキシと呼ばれるもの）を

作成できるような製品も出てきており、開発の効率化が図られるようになっている。しかしながら、実際には製品によって仕様上の解釈の違いや、サポートしているXML構造の記述方法に違いがあり、製品間での相互運用性は完全には実現できていない状態である。

WSDLの仕様はV1.2を制定中であり、多くの実装はV1.1を元に作成されている。

（3）UDDI

Webサービスでの提供サイトが数多くでてくれば、当然それを検索・登録する技術とそのサイトが必要になる。Webシステムを利用するときに、アクセスしたいWebサイト先を探すために、検索サイトを使うのと同様である。それがUDDIである。UDDIを利用して、公開されているWebサービスを検索し、そのWebサービス用のWSDLを取得することによりWebサービスの結合が可能になる。

UDDIは全世界でグローバルに利用できるサービス情報のレポジトリとして機能する。いくつかのUDDIサイトが既に公開されているが、それぞれのサイトは所有するデータをお互いにレプリケーションする仕組みになっているため、どの公開サイトからでも同じ情報を検索できる。

また、このグローバルなUDDIとは別に、企業内や業界内といった閉じた世界で利用する「プライベートな」UDDIを作成することも可能である。後にも述べるが、グローバルなUDDIで検索できるWebサービスに対して、そのサービス内容について保障する仕組みが提供されていないこともあり、現在は「プライベートな」形でのUDDIの活用が注目されている。

なおUDDIは、Webサービスの登録・検索専用に作られたわけではなく、組織の電話での連絡先やFaxでの連絡先なども登録するように考えられた、より汎用的なレポジトリを目指している。SOAPを用いたAPIを利用しての検索や登録が可能であるとともに、Webブラウザからもアクセスできるようになっている。

2002年７月にバージョン３の仕様が公開され、機能面での向上が図られた。しかしながら、サービス保障の課題解消まではいたっていない。

UDDI Ver3.0で追加された主な機能

- **マルチレジストリのサポート**
  －複数のレジストリへ異なる情報の登録が可能に
- **検索性の向上**
  －問い合わせ構造のネスティング、検索用の修飾語の拡張やソート順序の指定が可能に
- **WSDLの対応**
  －OverviewDoc, binding Templatesのinstance Detailsで複数の情報記述をサポート
- **SubscriptionalAPIで同期－非同期の通知サポート**
- **インターネット、エクストラネット、イントラネット、開発など状況に応じてポリシーを設定可能に**
- **キーとしてUUIDの代わりに人が読みやすいURIベースのキーが利用可能に**

３．Webサービスの特徴と適用

（１）Webサービスの特徴

Webサービスの特徴をあげると次のようなことがいえる。

①TCP/IP、HTTP、XML等の既存インターネット技術がベースであり、全く新規につくられた技術ではない。

②SOAP、WSDL、UDDIを始めとしてセキュリティ、トランザクション等についてWebサービスに関わる標準化が盛んに行われている。

③従来の分散オブジェクト技術であるCORBAやDCOMと違い、プラットフォームやツールに依存することなく実装が可能である。

④ほとんどのITベンダがサポートを表明し、対応製品をリリースしている。

これらの特徴をまとめると、「Webサービスは、全くの新規技術ではないため導入の敷居も低く、SOAPやWSDLなどの標準仕様に皆が準拠することで、既存のシステムも含めて、容易にサービス連携を実現できるものである」といえる。

一方で、現在のWebサービスには残された課題がまだ多いともいわれている。例えば、

・グローバルなUDDIを用いて動的に未知のWebサービスに接続し利用することは、技術的には可能であるが、実際にはサービス保証の仕組みがないなど、ビジネス面での環境が整っていない。

・トランザクション管理の仕組みがないため、複雑なトランザクション処理システムへの適用は難しい。

・Webサービスの連携による業務プロセスの実現を目標としているが、現時点ではフロー制御の仕様が統一されておらず、外部の接続相手との複雑な連携処理の実現は難しい。

といった内容である。

こうした状況の中で、Webサービスを企業システムにどのような形で適用していくのがよいであろうか。以下にその例を示す。

（２）Webサービスの適用

①ポータルサイトへの情報提供

Webシステムが、情報提供システムとして世の中に広まったように、Webサービスもリスクの低い無償の情報提供型サービスから導入していく。適用例としては、企業内ポータルや一般消費者向けポータルサイトに向けてのニュース提供サービス、検索サービス等が考えられる。検索エンジンのGoogle社が提供する検索Webサービス「Google Web APIs」がその一例である。

②取引先とのデータ交換

既存取引先とのデータ交換システムとしてEDIが導入されてきているが、この取引先との接続にWebサービスを活用することが可能である。この接続にWebサービスが適している理由としては、

・企業間での取引であり、標準化された仕様を使うことにより、両社間での仕様の取り決めが容易になる。
・接続相手が特定(限定)できるのでセキュリティ上のリスクが小さく、サービス保障の面からも従来からの取引契約の延長として検討しやすい。
・FTP等を利用したファイル転送方式よりも、リアルタイム性をあげることができる。

等があげられる。

③社内システムにおける共通サービス

企業においては、業務拡張、改善にあわせてシステムの追加開発が繰り返し行われてきている。その結果、類似機能をもつシステムが散在し、社員情報や顧客情報などのデータもシステム毎に保有して二重、三重の管理が必要になっているというケースがよく見受けられる。このような場合に、機能の共通化、データの一元化を実現する手段としてWebサービスを利用する。

既存システムをWebサービスとして接続する場合は、システムを新たに作り変えるのではなく、Webサービスのインタフェースを追加して既存システムをラッピングする方式をとることにより、効率的かつ早期に実現が可能になる。

④システム部品としての活用

Webサービスは、UDDIを利用してニーズにマッチしたWebサービスを検索し、結合して利用する、あるいは業務システムをWebサービス化することによりシステム連携を図るというのが基本的な考え方であるが、そのほかにも、次のような利用も可能である。

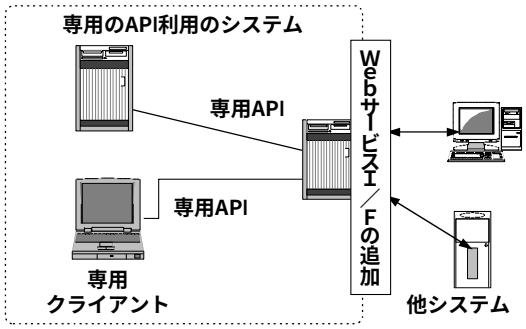

図4　社内システムへの適用

ⅰ）C/SシステムとWebシステムの融合

現在でのシステム構築の主流はWebシステムであるが、それ以前はクライアントサーバ（以下、C/S）型の二層構造システムが主流であった。C/S型からWebシステムへの移行を検討している企業が多いなか、伝票入力業務など、C/S型のリッチクライアントでの

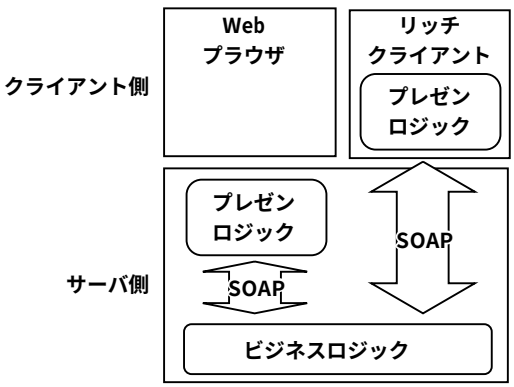

図5　C/SとWebシステムの併用

ユーザインタフェースのほうが業務効率が高く、Webシステムへ移行していないシステムも数多く残っている。このような場合にアプリケーションをWebサービス化してC/S間の通信をSOAP化するという方法がある。Webサービス化することにより、サーバ側のアプリケーションをWebシステムと併用することが可能になる。

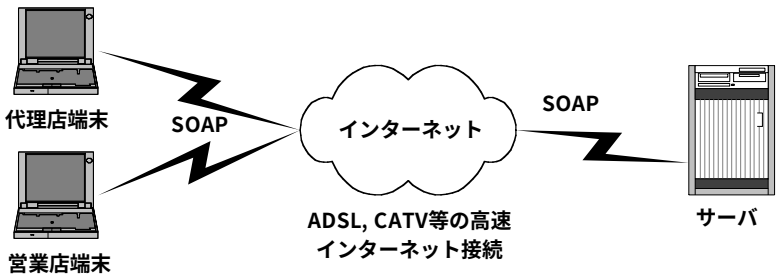

図6　SOAP接続

また、C/S型システムでマスタ情報や定義変更等により、頻繁にクライアントアプリケーションを更新する必要があるケースでは、該当箇所をWebサービス化することにより、クライアントアプリケーションの更新頻度を減らすことも可能になる。

ii）通信プロトコルとしてのSOAP利用

UDDIやWSDLは利用しないで、単にクライアントとサーバ間の通信にSOAPを利用するという手法もある。これは、SOAPがHTTP上で利用可能なことを活用して、インターネット経由でC/S接続を容易に実現したい場合に使える手法である。最近ではADSLやCATVインターネットの普及により、インターネット接続料金が以前と比べ格安になってきているため、営業店や販売代理店に設置するクライアントとセンターや本店にあるサーバとの接続にインターネットを利用したいというニーズが増えている。このC/S間通信プロトコルとしてSOAPを採用する。

ISDN回線や専用線等を使って接続していた場合に比べ、費用削減効果が期待できるのに加え、高速での常時接続化が可能になることにより、システムの機能向上も期待できる。

4．方式に関わる設計ポイント

Webサービスを開発する場合、Webサービス実装サポート製品がよくできていることもあり、単純なWebサービスであれば、SOAPの仕様などを理解していなくても簡単に作成することが可能である。しかしながら実際にWebサービスを企業システムに適用するとなると、留意すべき検討ポイントが数多くでてくる。NRIではこれらのポイントを整理し、対策方法の検討をすすめてきた。4章では方式設計時における検討ポイント、5章にアプリケーション設計時における検討ポイントを説明する。

方式設計時に検討すべき点として、以下の項目があげられる。

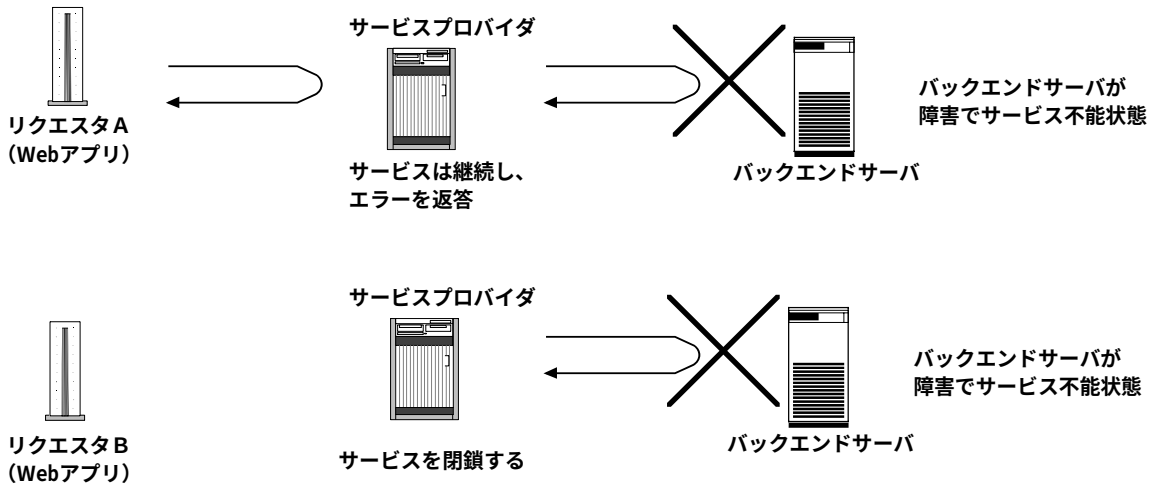

図７　障害時の対応（１）…プロバイダ側

・障害時の対応
・レスポンスタイム性能の確保
・拡張性／可用性の確保
・セキュリティ

以下に、これらの内容について具体的に説明していく。

（１）障害時の対応

Webサービスは、利用する側のサービスリクエスタと、提供者側のサービスプロバイダ間の接続が疎結合になるため、障害対策を入念に検討する必要がある。リクエスタサイドとプロバイダサイド双方で、障害としてどのようなケースがあり、万が一障害が起きた際には、どのような対応が必要となるかを予め確認しておく。例えば、

・サービスプロバイダ側のバックエンドシステムに障害が発生した場合、
　＞エラーを返す　又は、
　＞サービスを閉鎖する
・リクエスタが複数のサービスプロバイダに同時に要求を出すケースで、そのうちの１プロバイダに障害が発生した場合、
　＞一つでもエラーがあればすべてエラーにする　又は、
　＞障害を除いて結果を返す

等、システムの要件に応じて障害箇所毎に対応方法を確認しておく必要がある。

Webサービスはトランザクションの一貫性を保障できる仕様になっていないため、複数Webサービスを呼び出して処理が完結するような処理フローになるケースでは、エラー処理の扱いに特に注意する必要がある。

また、エラー発生時におけるWebサービス独自の対処方法として、

・予め用意しておいたミラーサイトでの代

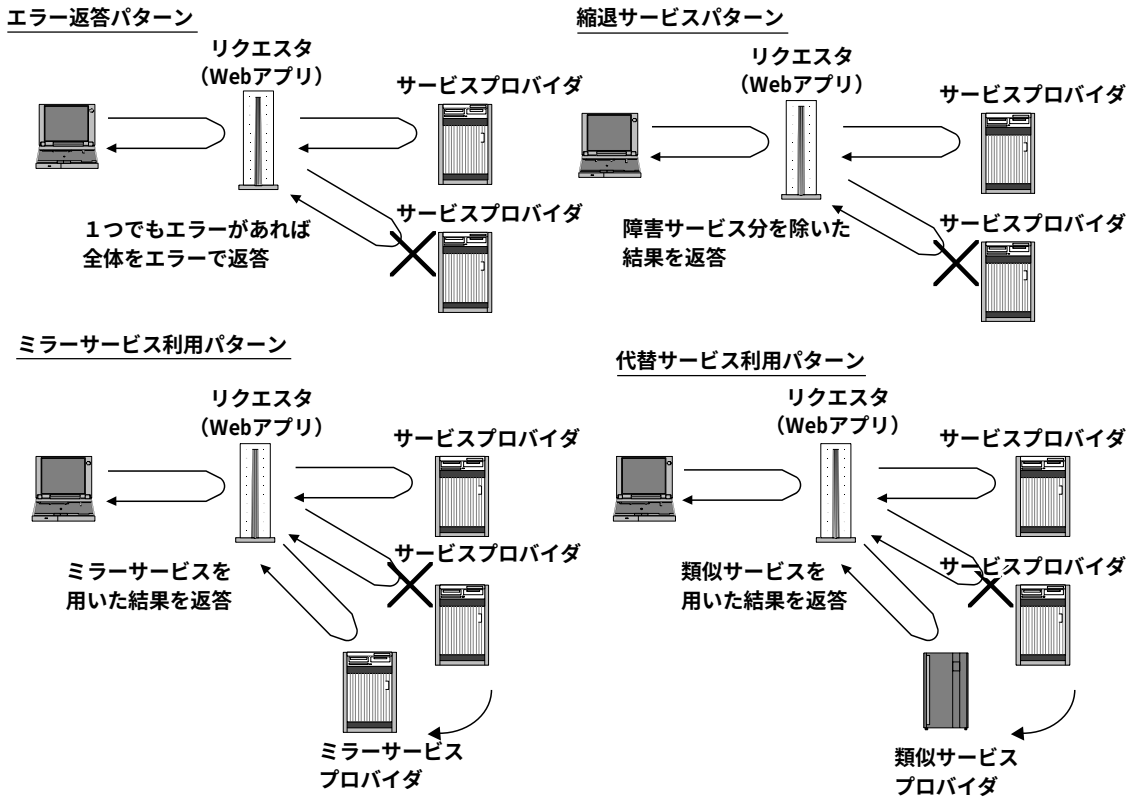

図８　障害時の対応（２）…リクエスタ側

替サービスを利用する方法。

・他社が提供する類似サービスを利用する方法。

もあげられる。ミラーサイトでの代替サービスは、プロバイダサービスを配置するセンターの障害時におけるリカバリー手段としての活用が考えられる。ただし、ミラーサイトへの接続切り替えを行うためには、リクエスタ側での対応が必要になる。システムがダウンするとサービスが継続できなくなるようなケースでの利用が想定され、UDDIのような動的バインディングを可能にするレジストリ情報サービスを提供するシステム等がこれにあたる。

ちなみに、必要なときにサービスを検索してすぐに利用できるようにするのが、Webサービスの大きな目的ではあるが、実際にUDDIを検索して動的に接続する仕組みを実現するには、事前に多くの準備が必要であり、利用できる相手も非常に限られたものになるというのが現実である。

例えば、リクエスタ側のプログラムが一時

的に接続相手を変えても継続的に処理を行うためには、プロバイダは業務的にもインタフェース的にも同一の機能を提供することが必須となるし、契約面においても、これまで取引のなかったプロバイダといきなり接続してサービスを受けるというのは、契約上問題がでてくるからである。従って障害対策の選択肢として、代替プロバイダを利用できるケースは限られたものになる。

（2）レスポンスタイム性能の確保

Webサービスは、従来のRPC接続と比べるとXML処理のオーバーヘッドなどにより性能面では劣ってしまう。実際にレスポンスを計測してみると、同一筐体や同一LAN上の接続であれば数10msecであったが、インターネット接続となると100msec以上であった。また、同一筐体や同一LANでの接続であってもその接続頻度が高ければ、全体のレスポンスに大きな影響を与えてしまうことになる。このため、レスポンスタイム要件が厳しいケース、サービスプロバイダへのアクセスが多くなってしまうケースでは方式面や設計面の工夫が必要になる。NRIでは解決策の一つとして、Microsoft社製品の.NETが持つオブジェクトプーリング機能及び、キャッシュ機能をレスポンス向上に役立てることができるかどうかの検証を実施したが、その効果は限定的となることが判明した。現時点では、純粋なWebサービスのリクエスタとプロバイダだけで、レスポンス向上を図れる仕組みを実装できる製品は、どのベンダーからも提供されていない。したがって、レスポンスをできるだけ早めたい場合には、リクエスタ側にテンポラリに情報を蓄えることのできるキャッシュDBを用意し、かつ、定期的にそのDBの情報を更新するエージェントサービス

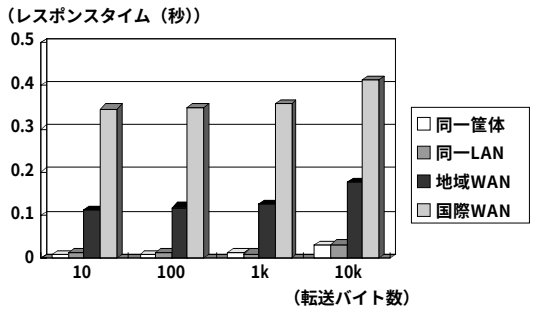

図９　レスポンスタイムの比較結果

オブジェクトプーリング

VS.NET で直接 SOAP プロキシのオブジェクト生成（Instance New）した場合と、VS.NET のオブジェクトプーリングを利用した場合でのレスポンスを単純なアプリケーションを作成して比較した。（環境：Pentium III 800MHz、Memory 256MB の PC1 台にリクエスタ、プロバイダの両方を実装。Windows2000 サーバ）

	スレッド数	リクエスト数/分	平均応答時間（ms）
通　常	20	475	2016
通　常	30	451	3477
オブジェクトプール	20	293	3605
オブジェクトプール	30	255	6617

結果を見ると、オブジェクトプーリングしているときの方が性能が悪くなっている。オブジェクトプーリングを実現するためには COM ＋への登録が必要になるが、COM ＋オブジェクトを呼び出すためのオーバーヘッドが生じ、かえって性能劣化をおこしてしまったと思われる。

キャッシング

Web サービスプロバイダのメソッドにキャッシュの設定を追加してその効果を測定した。キャッシュ時間 60s。（Webservice 属性に Duration を追加）（環境：Pentium III 800MHz、Memory 256MB の PC1 台にリクエスタ、プロバイダの両方を実装。Windows2000 サーバ）

	スレッド数	リクエスト数 / 分	平均応答時間（ms）
キャッシュなし	20	475	2016
キャッシュなし	30	451	3477
キャッシュあり	20	506	1844
キャッシュあり	30	497	3123

結果を見ると、キャッシュのないときと比べ、若干ではあるが性能が向上している。ただし、キャッシュ期間中は完全に同じデータのみを返すため、同じデータのみを提供している Web サービス以外には利用できない。したがって、利用できるケースは非常に限られることになる。

を用意するといった方法での対処が現実解として考えられる。この場合、SOAPがXMLベースであることから、キャッシュDBとしてXML DBを利用すると実装が容易になる。

（3）拡張性/可用性の確保

Webサービスにおいても、接続数の多いシステムでは、利用量の増加に柔軟に対応できるように拡張性を考慮しておく必要がある。対応方法は、従来のWebシステムと同様であり、スケールアウト（ロードバランサー等を活用したハード的な多重化対応）と、スケールアップ（CPUやメモリ増設による処理能力アップ）の組合せによる対応が基本になる。このため、アプリケーション設計ではリクエスト受付の部分のインタフェース層と、

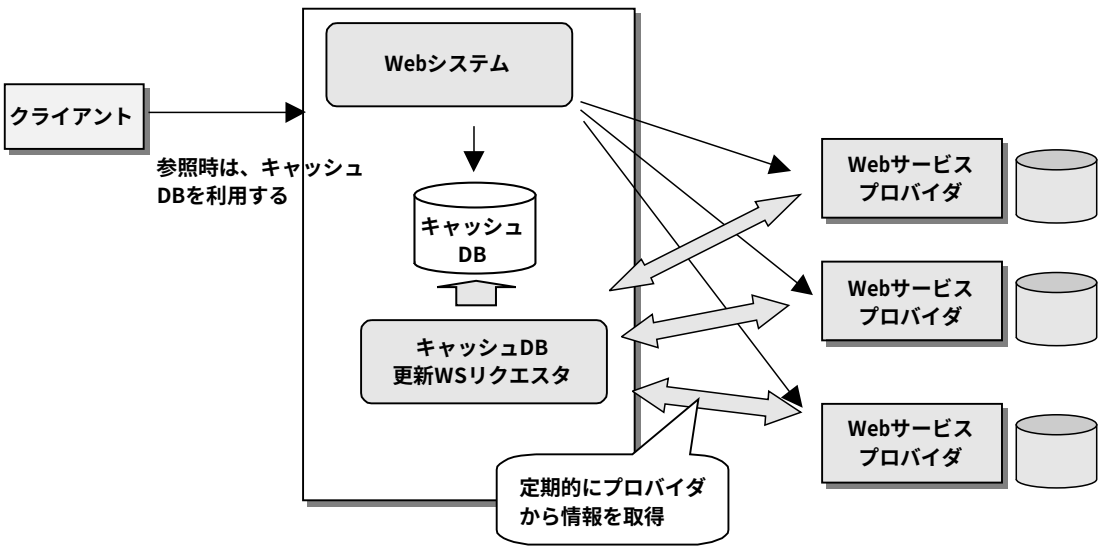

図10　キャッシュDBを利用した構成

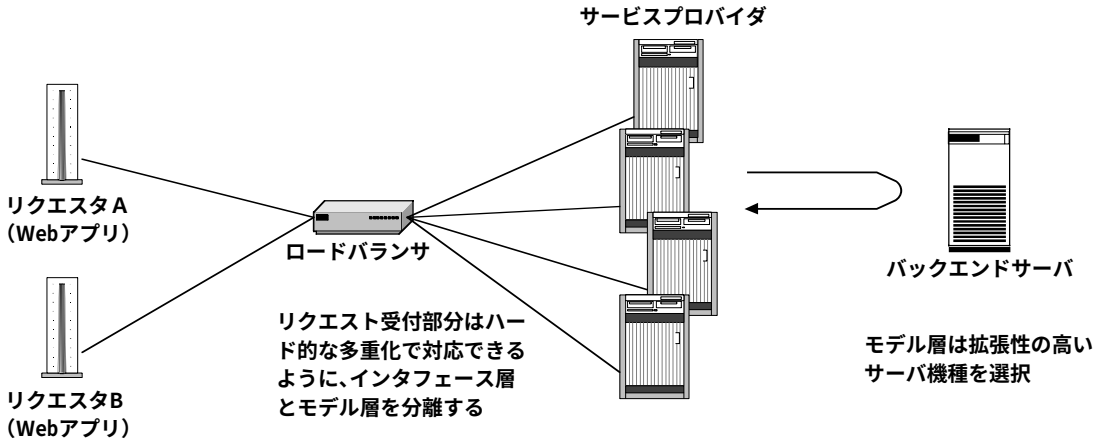

図11　Webサービスの拡張性

ビジネスロジック層を分離しておく必要がある。多重化が容易なインタフェース層はスケールアウトにより対応する。ビジネスロジック層は、アプリケーション的に考慮しないと多重化が難しいため、スケールアップでの対応を基本とする。可用性については、インタフェース層は多重化による冗長構成、ビジネスロジック層は、クラスタリングによる耐障害対策を検討する。

（4）Webサービスにおける認証モデル

リクエスタが単一のプロバイダに対してのみにアクセスする場合、認証制御はプロバイダの管理するリクエスタ情報によるアクセス制御で必要十分であり、既存のベーシック認証、クライアント認証の手段を用いて実現できる。データの暗号化についてもSSLで十分である。

一方、複数のプロバイダ、特に異なるサイトに存在するプロバイダにまたがっての認証を実現しようという場合には、認証サーバにトークン（チケット）を要求し、その情報を

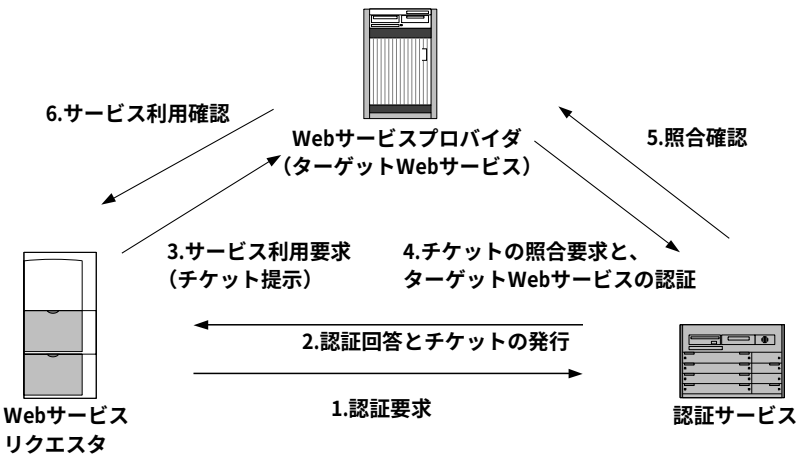

図12　認証プロトコル概略（フロー）

もとにプロバイダ側で認証する方式が適している。このとき、リクエスタは身元を保証するために証明書（鍵）技術を利用する。

SOAPはテキストベースのメッセージであり、誰でも容易にメッセージの中身を覗き見ることができる。従って重要なメッセージには暗号化を施しておく必要がある。また、SOAPはセッションレスの仕組みであるため、ログイン時に時間とともに変化するワンタイムパスワードを利用すると、より信頼性をあげることが可能になる。

2002年４月、Microsoft社、IBM社、VeriSign社からWS-SecurityというWebサービスのセキュリティ規格が発表され、同年７月に標準化団体であるOASISに引き継がれた。このWS-Securityは、メッセージの完全性、秘匿性を保証するSOAPメッセージの記述方法を提供し、セキュリティトークンとメッセージを関連付ける汎用のメカニズムを提供している。しかしながら、実装方法については何も規定していないため、今後もWebサービス導入時には、セキュリティについての十分な考慮が必要であることに変わりはない。

WS-Securityの特徴

- **メッセージの完全性、秘匿性を保証するSOAPメッセージ記述方法を提供**
- **特定のセキュリティトークンを利用しない→実装に依存**
- **セキュリティトークンとメッセージを関連付ける汎用のメカニズムを提供**
- **メッセージの完全性（改ざんされずに送信されること）を保証するためにXML署名をトークンと組み合わせて利用することが可能**
- **SOAPメッセージの特定の部分や複数の項目の暗号化のためにXML暗号化をトークンとともに利用可能**
- **ポイントツーポイントの暗号化ではなく、エンドツーエンドの暗号化を実現する（中継者介在の考慮）**
- **鍵の交換や、鍵の派生、信頼の確立・決定、セキュリティコンテキストの確立（複数のセキュリティ情報交換、認証メカニズム）については考慮の対象外→実装で考えることが必要**

５．アプリケーションに関わる設計ポイント

アプリケーションを設計する際に検討すべき事項としては、

・セッション管理
・インタフェース定義
・Webサービスの粒度
・Webサービスの再利用

等があげられる。以下に、これらの内容について説明する。

（１）セッション管理

Webサービスは、従来のWebシステムと同様にステートレスな通信が前提となる。やむを得ずステートフルな接続を実現したい場合には、Webシステムにおけるブラウザ～Webアプリケーション間での場合と同様に、セッション管理が必要になる。同一サイト内での接続であれば、WebサーバのHTTPセッション機能を利用して管理することもできるが、複数のサイト間でWebサービスのセッション管理を行いたい場合は、リクエスタ側にセッション管理機能を実装する必要がでて

くる。これは、サーバー側でブラウザからのリクエストを紐付けてセッション管理を行っていた従来のWebシステムでの考え方とは逆の発想になるので注意が必要である。

（2）インタフェース定義

①SOAPのメッセージタイプ

SOAPには、サービスリクエスタとサービスプロバイダ間でやりとりするメッセージタイプとして、Document/literal型とRPC/encoded型の二種類のメッセージタイプを利用することができる。

どちらのタイプを利用するかは、接続するパターン、接続環境等によって設計の段階で決めていく必要がある。現時点では、接続するWebサービスのプラットフォームがばらばらな状態で、相互運用性を重視する必要があるのであれば、対応製品の多いRPC/encodedタイプを選択するのが無難である。ただし将来的にはデータ構造の自由度が高いDocument/literalタイプをサポートする製品も増えてくるため、こちらを採用するケースが増えていくものと考えられる。

表1　SOAPのメッセージタイプ

メッセージタイプ	概　　要
Document/ Literal	アプリデータ構造をXML化したものを「そのまま」転送する方式
RPC/ Encoded	SOAP encoding Styleに従い、アプリデータ構造を「変換」したデータを転送する方式

②エラー情報の定義

SOAPメッセージのもう一つの検討ポイントに「エラー情報のインタフェース」があげられる。SOAPではエラーメッセージを返す方法としてSOAP Faultを用いる方法と、通常のレスポンスと同様にSOAP Resultを用いる方法がある。仕様的にはSOAP Faultを利用するべきなのであるが、SOAP Faultは内部構造の詳細が、仕様として規定されていないのが現状で、相互接続性に問題が残っている。従って、エラー情報の扱いについてもリクエスタとプロバイダ間で事前によく確認し、どのような形式でエラー情報を返すのかを把握しておく必要がある。

（3）Webサービスの粒度

Webサービスの設計時におけるもう一つの重要な検討項目として、「サービスの粒度（大きさ）」があげられる。Webサービスをシステム部品として扱う場合には、Webサービスが自律システムとして動作できるような粒度を設定する必要がある。ある業務処理を自律した処理単位に分割しようとした場合、その解は一通りとはならない。従って、設計時における業務分析が重要となる。粒度が大きな場合、処理単位は「顧客管理」、「受注管理」、「請求管理」などの大きさになる。粒度が小さくなると、共通ファンクション的なものとなり「区分コード管理」、「時価情報管理」、「クレジット与信接続」、「住所コード管理」

表2　コンポーネントの粒度

粒度	例	特　　徴
大	顧客管理、受注管理等	○ビジネスプロセスの標準化につながる ○自律度が高く、業務処理単位での連携が可能 ×独自仕様にこだわる場合は再利用できない
小	共通ファンクション（コード管理、汎用情報管理）	○共通的な汎用ファンクションであり、カスタマイズ要件が少なく、再利用しやすい ×自律度が低くなり、Webサービス化するメリットが小さい ×アクセス頻度が高いと性能に悪影響を与える

といったレベルになる。粒度が小さいほど、そのサービスの再利用性を高めることができる反面、サービスの自律度は低くなってしまい、Webサービス化するメリットが薄れてくる。また、粒度が小さいと呼び出し回数が増えてしまい、性能に影響を与えることにもなるので注意が必要である。

（4）クラス構成

Webサービス内部のクラス構成を設計する場合、作成されたWebサービスの再利用性も考慮にいれて設計する必要がある。例えば、Webサービスの内部構造をインタフェース部、ロジック部、データ操作部の三階層にクラス分けして設計する。Webサービスを利用しないWebシステムについても同様のクラス構成にすることにより、インタフェース部以外のロジック部とデータ操作部は、WebシステムとWebサービスの間で再利用が可能になる。

（5）Webサービスの再利用

既存のWebサービスを利用する場合、そのサービスを全くそのままでは利用できず、何らかのカスタマイズが必要になる場合がある。このときのカスタマイズの方法は、そのサービスの提供形態によっても異なってくる。代表的と思われるパターンについて適用方法と留意点について説明する。

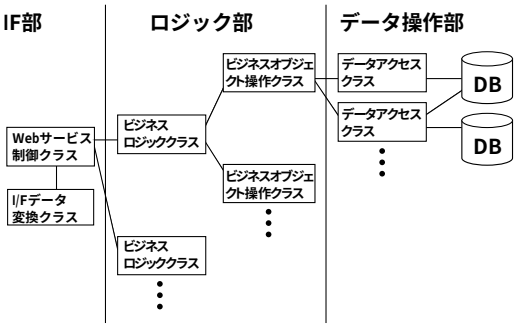

図13　Webサービスのクラス構成

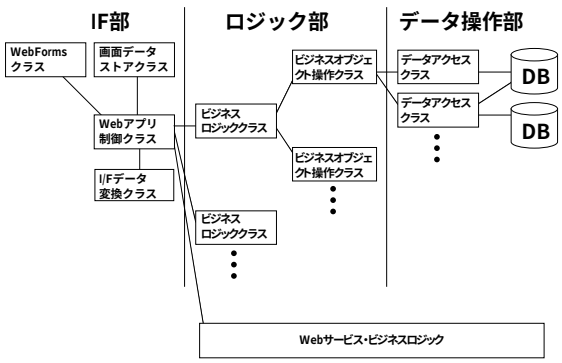

図14　Webアプリケーションのクラス構成

①サービス固定型

【適用方法】カスタマイズ不可の外部サービスを利用する形態。自らがカスタマイズ部分をWebサービスとして作成し、外部のサービスをラッピングして利用する。

【留意点】WebサービスからWebサービスを呼び出す形態になり、ネストすることになるので、性能やエラー処理に注意する必要がある。また、ループから呼び出さないようにするといった対策も必要である。

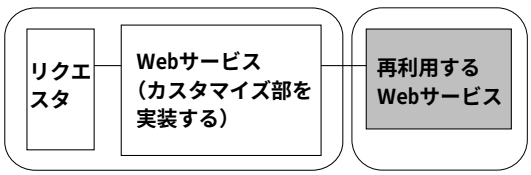

図15　サービス固定型

②レディメイド型

【適用方法】カスタマイズ部分をサービス提供者に依頼して実装してもらい、そのサービスを利用する。

【留意点】利用者から見るとシンプルな構成になるが、提供者側の管理が複雑になる可能性がある。

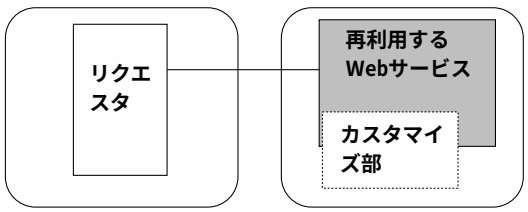

図16　レディメイド型

③テンプレート型

【適用方法】Webサービスを作成するテンプレートのソースコードを入手し、カスタマイズすることにより、Webサービスを作成する。

基本的にカスタマイズは、テンプレートを継承しておこなう。これにより、オリジナル側のバグフィックス等の情報の反映が容易になる。

【留意点】ソースコードを入手できるので、カスタマイズは自由に行えるが、似て非なるモジュールが増殖していくことになり、ソース管理が複雑になる。

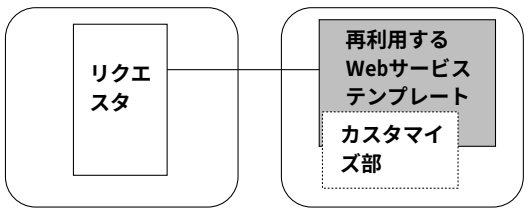

図17　テンプレート型

（6）Webサービスにおけるトランザクション管理モデル

Webサービスでのトランザクション処理を考える場合、リクエスタから見てどのようにWebサービスを利用するかによって対応が変わってくる。従来のWebシステムと同様に特定の一つのサイトにあるWebサービスのみにアクセスする場合には従来のWebシステムと同じ考え方でシステム構築が可能

である。

しかし、複数サイトに位置するWebサービスプロバイダに対してリクエストを発行する場合には、従来とは別の対応が必要になる。SOAPは疎結合の仕組みであるため、コネクション情報を保持することができないからである。このためメッセージの一貫性を保証するにはSOAPメッセージ中に識別子を入れ、その識別子を元にしてトランザクション管理を行う仕組みを作りこむ必要がある。また、インターネット上で確実にメッセージが届いたことを保証し、処理の状態を保証するためには、常にメッセージ受信者であるサービスプロバイダが処理ステータスを発行して管理する方式がモデル的には理解しやすい。

しかしWebサービスの場合、サービスプロバイダは複数のサービスリクエスタからの要求を受け付けることになるため、サービスプロバイダ側ですべてのトランザクションを管理するのは現実的ではない。したがって、トランザクション管理はリクエスタ側で処理するべきといえる。この場合のトランザクションモデルの一例を図18に示す。

2002年８月にWS-Transaction、WS-Cordination、BPEL４WSというトランザクション実装方法についての三種の仕様が発表されている。これらの仕様も参考にしながら、疎結合に適したトランザクションモデルを考

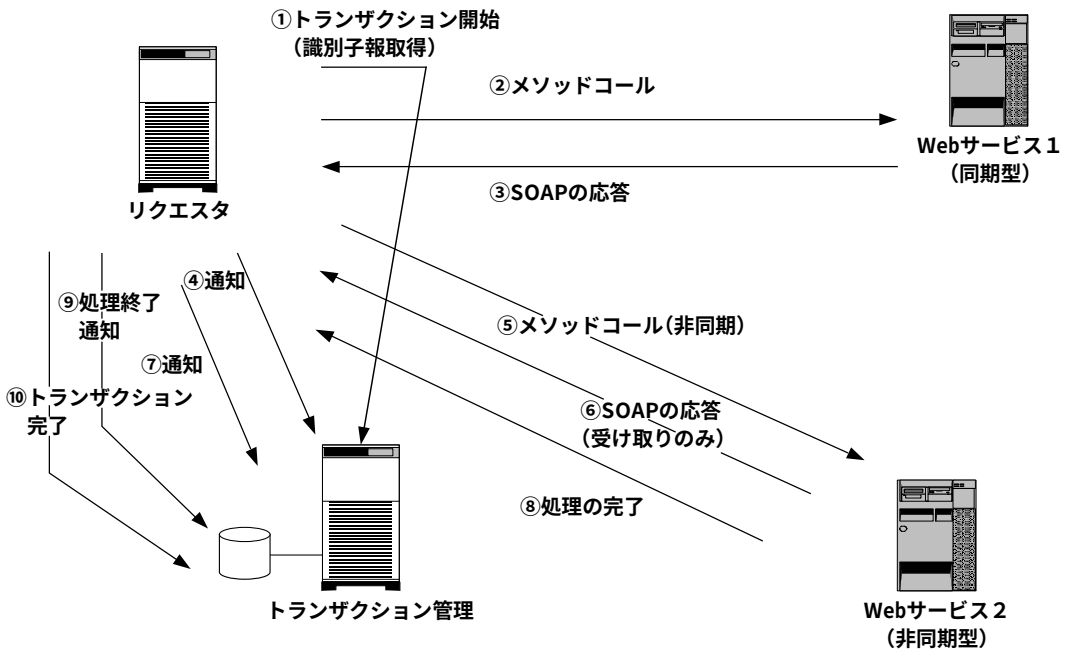

図18　トランザクション管理モデル

えていく必要がある。

６．Webサービスの今後

Webサービスの導入が広まることによって、今後システム構築モデルが大きく変わっていくことが考えられる。また、Webサービス提供者が増えることにより、新たなビジネスモデルも登場してくると想定される。そのいくつかの例を紹介する。

（1）Webサービスアグリゲータ

Webサービスを提供するサービスプロバイダが増えてくると、これらをとりまとめる役割をもったサービス提供者がでてくると考えられる。これがWebサービスアグリゲータである。Webサービスアグリゲータは、複数のWebサービスを束ね、より付加価値の高い高度なサービスを提供する。例えば、サービス検索エージェント、サービス・マッチメーキングといったようなサービスが考えられる。また、アグリゲータが取引相手の信用保証を行ったり、取引履歴を活用した分析情報の提供や課金代行、トランザクションの一時的なキュー保管サービスを行うようなサービスも考えられる。このようなWebサービスアグリゲータの担い手としては、eマーケットプレイス運営業者等が候補になる。

（2）セレクティブアウトソーシング

企業はIT戦略として、全ての業務システムを自らが構築し資産化するのではなく、外部の提供サービスを活用することにより、限られた経営資源をコアコンピタンス業務に集中投資するべきであるとよく言われている。Webサービスを活用したシステムモデルは、この戦略の実現に大いに貢献するものと考えられる。従来のASPモデルと違い、企業内に保有するシステムと外部からサービス提供を受ける（＝アウトソースする）部分のシステム連携が容易なため、部分的な業務のアウトソーシングが実現しやすくなるからである。

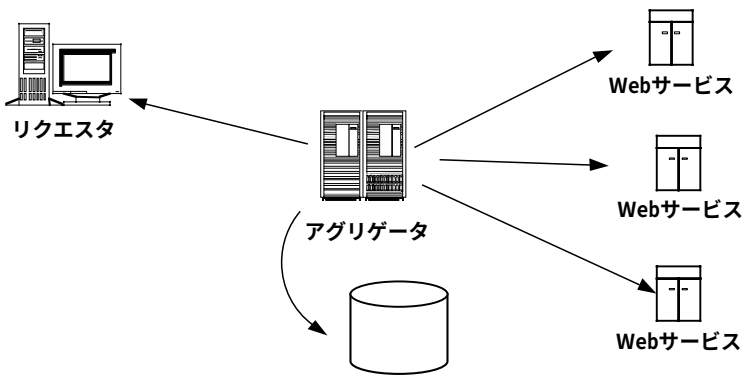

図19　Webサービスアグリゲータ

7．まとめ

現時点では、本論で説明してきたとおり、セキュリティやトランザクション関連といった仕様として未整備のものや、UDDIから検索するサービスのサービス保証をどうするかといった課題等が残っており、当面は利用できる範囲が限られる状況である。従って、Webサービスによるセレクティブアウトソーシングの実現までには、もう少し時間がかかる。しかしながら、本論で紹介したように、Webサービス設計時における検討ポイントも着実に見えてきている。また、W3CやOASYSなどの標準化団体の尽力により仕様の標準化も着実にすすんできており、課題が解決する日もそう遠くない。各企業や団体においては、情報提供サービスや社内システムにおける利用のような形で、Webサービスの活用をまずは考えていくべきである。

参考文献

[1] 嶋本正他「Webサービス完全構築ガイド」日経BP社、2001

[2] Steve Graham他 著、嶋本正他 訳「JavaによるWebサービス構築」、ソフトバンクパブリッシング、2002

[3] 福原信貴「Webサービスシステムの構築」オブジェクト指向シンポジウム2002

[4] Web Services Activity, W3C
http://www.w3.org/2002/ws/

[5] 日本IBM、developerWorks
http://www-6.ibm.com/jp/developerworks/webservices/

[6] マイクロソフト、MSDN
http://www.microsoft.com/japan/msdn/webservices/