連載　Web サービス

第 4 回 Web サービスの方法論

Fujitsu Laboratories of America, Inc.
長橋 賢児　*nagahashi@jp.fujitsu.com*

編集：XML コンソーシアム

◇ Web サービスの方法論とは

今回は提供されている Web サービスを使うユーザの立場から見たときに Web サービスがどういう考え方をしているか，というお話です．UDDI や BPEL4WS など Web サービスの技術仕様を細かく見るのではなく，背景にある考え方を探ってみたいと思います．

すでにご存知のように Web サービスのコンセプトには三種の神器があります．

- UDDI（Universal Description，Discovery，and Integration）
- WSDL（Web Services Description Languages）
- SOAP（Simple Object Access Protocol）

最近ではこれに

- BPEL4WS（Business Process Execution Language for Web Services）

を加えて SOA の要素とするようです．

このそれぞれに Web サービスの方法論，すなわち利用者から見て Web サービスを

- 見つける
- 組み合わせる

ことでシステムを構成するという考え方が反映されています．この中で特に「組み合わせる」を中心に見ていくことにします．それから筆者個人の考えではありますが，問題点や，将来どうなる（なって欲しい）のかということについても触れます．

◇ サービスを見つける

UDDI は希望の用途に合った Web サービスを発見し，その使い方に関する情報を得るための「電話帳」です．産業分類などをもとに検索すると，WSDL など Web サービスの仕様の情報が得られるということは多くの読者がご存知だと思います．

思えば UDDI が最初に発表されたときには「e- コマースのレジストリ」などとニュースに書かれ，UDDI を使えば自動的に取引相手を発見して簡単にビジネス関係を構築できる！というようなことまで言われました．UDDI に登録されているのはビジネスサービスである，と考えられていたわけです．しかし Web サービスが提供できるものと，現実の企業間通信に必要なものとの間にはかなり大きなギャップがありました．その結果 UDDI は当初想定された公開ビジネスサービスのレジストリとしてはあまり使われていません．最近大企業が提供し始めている「真剣な」Web サービスが UDDI に登録されたという話も聞きません．先日 UDDI の創始メンバー企業の運営する公開 UDDI サーバで Web サービスを検索してみましたが，登録されているサービスの多くはすでにアクセスできなくなっていました．

一方で UDDI は企業内のプライベートサービスディレクトリとしては次第に広く使われ始めているようです．つまり企業内に分散して存在する Web サービスの場所（URL）を具体的ハードコーディングしてしまわず，機能に基づいて実行時に検索して利用するために UDDI を使うわけです．これはビジネスサービスの検索というよりはソフトウェアのコンポーネントを探すという利用法で，結局 CORBA の Object Repository と同じ使い方に落ち着いたと言えるのではないでしょうか．Web サービスが当初盛んに言われていた企業間のビジネス通信手段というよりは，企業内のアプリケーション統合の手段として使われ始めたのと同じ時期だと思います．

◇ サービスを組み合わせる

発見されたサービスがWebサービスの方法論の出発点になり，次はそれを組み合わせて何か新しいアプリケーションなり，サービスなりを実現することになります．これに専用の言語が必要じゃないか，という考えが出てくるのは自然でした．

2001年にWSFLとXLANGが発表され，それに対抗するWSCI，そしてWSFLとXLANGを融合させたBPEL4WS（2002年）の登場，W3CのWS-Choreographyの発足といまやかなりホットな領域です．

実はXLANGとWSFLなどの言語はもともと実行用に設計されたものではありません．WSDLに付け加える形で複数のWebサービスの関係，つまりこのWebサービスはこういうふうに組み合わせて使うのだ，という振舞いを説明するために設計されたものでした．この2種類の記述は前者が「オーケストレーション」，後者が「コレオグラフィ」と呼ばれ区別されていますが，実際にはこの2つの記述の境目はとても曖昧で，振舞いを詳細に書こうとするとほとんど具体的な実行の記述になってしまいます．実際のところBPEL4WSに対しては多くの人が他のWebサービスを部品として使って新たなソフトウェアを実装するための新しいプログラミング言語，つまりオーケストレーション言語として期待しているのではないでしょうか．

◇手続き型オーケストレーション

これまで提案されてきた言語は，Webサービスの呼び出しの間の流れを記述しているという点ではどれも同じで，それをどういう方法で書くかに違いがあります．WSFLはこの流れを状態遷移図のようなフロー図で記述し，XLANGは手続き型プログラミング言語のような制御構文を使って記述する方法を提案しました．BPEL4WSにはXLANGの制御構文方式が多く取り入れられています．簡単に言うと伝統的な構造化手続き型言語の，サブルーチン呼び出しがWebサービスの呼び出しになったもの，というイメージですが，並行処理のメカニズムもあり，Webサービスに特化した新しいプログラミング言語です．

実はこの種の言語は，Webサービスで初めて作られたものではありません．たとえば，もともと承認プロセスなどの自動化のために設計されたワークフローシステムには作業のフローを記述するための言語（たいてい視覚的なもの）が用意されています．ワークフローシステムはその後アプリケーション統合の分野にも応用され，ちょうどBPEL4WSと同じ用途に使われています．WfMC（Workflow Management Coalition）の定めたXPDL（XML Process Definition Language）という標準もあります．

もっとBPEL4WSに近いのはBPMI（Business Process Management Initiative）のBPML（Business Process Modeling Language）でしょう．BPMLもBPEL4WSと同じ構造化制御構文を使っていますし，プロセスの表現力ではBPEL4WSより豊かなところもあります．

◇ コリレーション

サービスを提供している側が一方的に呼び出されるだけなら簡単なのですが，実際のアプリ連携，特に企業間の連携では，メッセージ送信とは別のセッションで返信が返ってくるというパターン（非同期レスポンスといいます）がよく使われます．この場合，BPEL4WSで書かれたプロセスは，サービスを呼び出すだけでなく，自分が呼び出される側にもなるわけです．この非同期レスポンスのことをBPEL4WSでは「コールバック」と呼んでいます．

このコールバックを正しく動作させるには，ある仕組みが必要です．というのは，BPEL4WSで書かれたプロセスは同じ形のもの（インスタンス）が複数同時に実行される可能性があるので，相手から呼び返された時に，どれに対して返事を返しているのかを判別する必要があるからです．ワークフローの場合，エンジンがこのインスタンスにプロセスIDを振り，返信メッセージにプロセスIDを指定する特別な場所を用意するでしょう．CORBAなら，インスタンスをCORBAオブジェクトに対応させるでしょう．

ところがWebサービスではサービスは自分とは違う人・組織がばらばらに提供しているものなので，何か統一されたフレームワークに従って作られていることを期待するのはよそう，というのがいわば信条第一号です．CORBAが広く普及しなかった理由の1つは，オブジェクトという概念を前提にしてしまったことにある，といいます．確かに既存のXMLメッセージを見ると，オーダ番号など「流れ」を特定するのに必要なキーは他のビジネスデータと区別なくメッセージの中に埋め込まれています．どこにもプロセスやオブジェクトといった概念は見当たりません．

そこでBPEL4WSではコリレーションというメカニズムが作られました．コリレーションは送受信メッセージの中のどの値をキーとしてプロセスと対応付けるかを記述する仕組みで，これによって特定のフレームワークを前提にする

ことなく，どんな既存サービスでもインスタンスの識別ができるようにしています．

◇ メッセージのパターン

しかしこの Web サービスの現在あるものを何でも受け入れようという考え方には限界も見えます．Web サービスの方法論は既存の個別サービスを出発点として，それを一種場当たり的に積み上げていく方式ですが，このアプローチではあまりに効率が悪い場合もあります．

Web サービスは企業間のビジネス情報交換（B2B）にも使えるものとして期待されていますが，企業間のビジネス情報交換は単に HTTP でデータを送ればよいというほど単純なものではありません．組織間の法的な責任がかかわってくるため確実な情報交換を保証するための仕組みが必要になります．

この種の仕組みの一例は送達確認です．B2B では受信したメッセージに対して受領確認メッセージを返すことでメッセージが確かに正しく受け取られたことを確認するのが普通です．既存の B2B のプロトコルではこの受領確認メッセージをシグナルと呼び，ビジネスメッセージの種類によらず使われる共通メッセージとして定義しています．さらにビジネスメッセージとそれに対応するシグナルメッセージのパターンはパッケージ化されているため，ユーザはシグナルを意識する必要はありません．しかし Web サービスでこれを単純に実現しようとすると WSDL でこれをすべて operation として記述して明示的に扱うことになり，とても不便です．

もうひとつ例を挙げましょう．最近 AIAG（Automotive Industry Action Group）の主催する IVI（Inventory Visibility & Interoperability）という実証実験プロジェクトがあったのですが，この中に Web サービスを使って企業間で在庫情報を共有する実験シナリオがありました．これを実現するために実験チームは OAGi の決めたビジネスプロトコルを Web サービス上でどう実装するかを設計する必要に迫られました．これは ebMS など他の B2B メッセージ送信プロトコル（トランスポート）ではすでに標準仕様化されているものですが，それを Web サービス用に再発明したわけです．これと同じことをあちこちでやると互換性のないフレームワークが複数できてしまうので，そのようなことにならないように標準仕様化されていく必要があります．

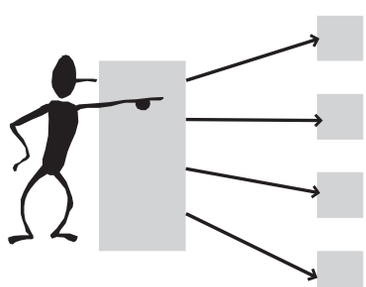
つまり Web サービスを使って実際の複雑な機能を実現していくには，1 つ 1 つの operation を直接扱っているのでは効率が悪く，一定のプロトコルパターンを設計してパッケージ化し，そこはブラックボックス化して全体を簡単にする，という階層的なアプローチが必要なはずなのですが，現在の Web サービスの方法論ではそれが難しいだけでなく，Web サービスらしくない，として拒絶している感すらあります．しかし人間は物事をパターン化して整理しないといずれ複雑さについていけなくなってしまいますし，これまでの歴史を見ても Web サービスを B2B に応用する場合にはこのアプローチが必要であることは間違いありません．議論は ebXML などですでに行われており，Web サービスが独自に再発明するのではなくそうした技術と融合していくことが望まれます．

上の送達確認の例では，今まさに WS-Reliability 仕様が OASIS で標準化されようとしているところです．この仕様では SOAP を使って受領確認メッセージを返すところは WS-Reliability のユーザからは見えないところで自動的に行われます．つまりユーザからは単なる 1 回の SOAP メッセージングに見えても，実は中で複数の SOAP メッセージがやりとりされている，ということが起こるわけです．Web サービスでプロトコルのパッケージ化を行うにはおそらくこの方法が中心になるはずなのですが，プロトコルモジュールが複数組み合わされたときにその間にどういう干渉が起こるか，など考えるべきことがまだまだあります．少なくとも今後 Web サービスは思ったよりずっと複雑なものになっていくでしょう．

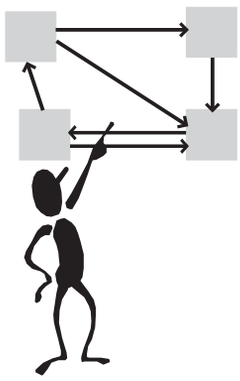
◇ 違うアプローチ

実は Web サービスの組合せを記述する方法は BPEL4WS 以外にもあり得ます．

BPEL4WS によるビジネスプロセスの記述では，BPEL4WS（の実行エンジン）が既存の Web サービスの利用者となり，既存のサービスは「使われる」立場になります．この形は BPEL4WS が主体となって他のサービスを呼び出す形なので，私は「一人称」と呼んでいます．



これに対して特に誰を中心ともせず，全体を俯瞰する立場から見た情報交換の様子を記述するアプローチがあります．これはいわば「三人称」です．三人称の方式の代表格は ebXML の BPSS(Business Process Specification Schema) です．



BPEL4WS では相手から呼び返されるパターンを「コールバック」と呼んでいましたが，これは BPEL4WS が一人称の記述だからです．企業間のビジネスプロセスにはこうしたサーバ・クライアント的な概念はなく，パートナが対等に双方向にメッセージを交換しあうと考えるのが普通です．そのためコレオグラフィの用途には三人称の書き方のほうが簡単で自然な記述になります．しかし一方で三人称方式は既存のサービスを使うというより，サービスの設計書になっているのではオーケストレーションには使えません．逆に BPEL4WS でコレオグラフィを記述できるという主張もありますが，一人称方式はある一人の視点でしかプロセスを記述できないため，全体を記述するのが苦手で，たとえば自分以外の参加者の間の通信を書くことができないなどの弱点があります．BPSS も Web サービスの取り込みを進めていますし，今後 BPSS を設計書として BPEP4WS のテンプレートを出力する，などの手法も出てくるのではないかと思います．

また，BPEL4WS でサービスを提供する側のプロセスを記述してみると，ある共通のパターンが見えてきます．

1. 最初の SOAP メッセージで BPEL4WS のフローインスタンスが作成され，初期化処理を行う．
2. 次のメッセージを待つ
3. 受信したメッセージによって分岐して処理
4. 2. に戻る

この処理パターンをどこかで見た覚えはないでしょうか．そう，GUI アプリケーションなどイベント駆動型のアプリケーションの形です．GUI アプリケーションではこのコードはほとんど開発者のおまじないになっていて，Java では書く必要すらなくなっています．Web サービスもそういうふうにはならないのでしょうか？

ビジネスプロセスにはこのようにすでによく知られたパターンがあるのですが，今回お話ししたように，Web サービスの方法論はまだこれらを取り入れていません．むしろそういうパターンやフレームワークを規定しないのが Web サービスらしい，とする様子もあります．しかしこのままの Web サービスを使って今後複雑なビジネスプロセスを記述したり，実装したりということをしていけるかというと，どうもそうは思えないのです．

これを反映してか，最近現れたのが EDA (Event Driven Architecture) というコンセプトです．2003 年に EDA という言葉を提唱した Gartner Group は，企業内アプリケーションの統合が高度になるにつれて，イベント駆動型の統合方法が必要になるだろう，と言っています．EDA の具体的な形がどんなものか，EDA と SOA がどういう関係にあるのか，まだこれからの議論ですが，イベントとそれに対するアクションのセットでアプリケーション統合を宣言的に実現するというアプローチは，現在の何でもありの Web サービスに比較すれば何がしかの「形」をユーザに与えてくれます．ユーザは EDA をモデルとして使い，中身は Web サービスを通信プロトコルとして動いている，という形になるような気がします．Web サービスや SOA と言う言葉は今度どうなっていくのでしょうか．

◇ 全部これでいいかも

Web サービスの方法論は「すべてが Web サービスになったとしたら世界は簡単になる」という発想をもとに作られていて，多くの点で技術的には既存のものを SOAP に焼きなおしただけかもしれません．それでもこれまでさまざまな API，データ形式，通信プロトコルを受け入れるかたちで複雑に発展してきた情報通信技術の世界で「すべてが Web サービスになったら」という発想がたいへんなインパクトを持っていることは確かです．Web サービスが登場して，もしかしたらすべてを SOAP と XML にしてもいいかも，と皆が思い始めています．実はこれこそ Web サービスの最大の方法論ではないでしょうか？　現在の Web サービスはデータの部分にあまり触れていませんが，「データは XML でいいかも」というところは今後とても大きなインパクトを持ってくると思います．

（平成 16 年 10 月 12 日受付）