SOAに連なる分散アーキテクチャの発展経緯

WebサービスからSOAへ

企業内部に存在するさまざまなシステムを相互に連携させ、さらに価値のあるサービスを実現したいというのは、大規模なシステムを運用している組織であれば誰もが抱く希望である。現在では、インターネットという地球規模の広がりを持つネットワークに接続されていることを前提としてシステムを設計できるため、インターネットを介して相互にやり取りしながら自動的／自律的に処理をこなしていくシステムを発想するのはごく当然といえる。しかし、現実にこうしたシステムを実現するには、さまざまな技術的要素が必要となり、現在でもまだすべてが完備したとはいえない状況だ。ここでは、SOAを取り巻く技術面での動向を紹介したい。

Feature Story

第2部　技術動向

システム・アーキテクチャの変化

企業内におけるコンピュータ・システムの利用は、大量の数値を扱うなど、もともとコンピュータが得意とする処理で、人手で処理したのでは時間やコストがかかりすぎる分野から順次段階的に始まった。人手で行なっていた作業を部分的にコンピュータ化する段階では、システム全体のアーキテクチャを変更する必要はなく、コンピュータ化された処理も、最終的には人手によって他の業務と統合される。具体的には、コンピュータ処理された情報が、最終的に帳票に打ち出され、従来の処理との違いは単に帳票が手書きなのかプリンタによる印字なのか、という違いだけ、という状況である。その帳票を元に次の処理を行なうのが人間なのであれば、帳票が手書きかプリンタ印字かという違いは何の影響も及ぼさないため、業務システム全体のアーキテクチャとしては、部分的に導入されたコンピュータのことを無視することが可能であった。

こうして構築された業務処理単位に特化したコンピュータ・システムは、いわゆる「部分最適」という形で洗練度を高めていった。当該処理のことだけを考えて最適化されたシステムは、実用上は低コストで効率的な運用を可能とする。

しかし、ITの発展に伴い、コンピュータ化はさまざまな業務に拡大され、現在では事実上ほぼすべての業務が何らかの形でコンピュータ化されているという企業が大半だろうと思われる状況になってきた。この状況では、人手による処理を前提に組み立てられた全体システムに手を加えることなく、特定部分だけを「部分最適化されたコンピュータ・システム」で置き換えていく、というアプローチには無駄が生じる。今となっては笑い話にもならないが、コンピュータ化が順次拡大していく局面では、あるコンピュータ化された処理で出力された帳票を、別のコンピュータ化された処理に引き渡すために、オペレータがプリンタ印字された帳票を見ながら再度データ入力していた、などという状況さえ生じていたのである。処理をコンピュータ化することで得られるメリットを最大限に引き出すためには、人手による処理を想定した全体アーキテクチャの中に他の部分に影響を与えない形で部分最適化されたシステムをはめ込んでいくという考え方から、最初からコンピュータ処理を前提とした全体アーキテクチャに切り替えていく必要がある。こうして生まれてきたのが、最新の企業システム・アーキテクチャであり、具体的には、EA（Enterprise Architecture）やSOA（Service Oriented Architecture）なのだ。

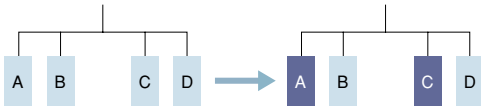

図1：業務別のコンピュータ化
全体の仕組みを変えずに特定要素だけをコンピュータ化するため、部分最適とならざるを得ない。

分散システムの相互接続

企業全体の活動のなかでは、さまざまな業務プロセスが実行されている。それをコンピュータ化していくうえで利用されるのが、さまざまな業務アプリケーションということになる。初期の業務アプリケーションは、当然だが他のシステムとの連携などはまったく意識せず、完結した存在として設計された。

つまり、この段階の業務アプリケーションは、単独で完結した「業務システム」だったのである。完結しているが故に独自の手法で最適化され、必要な処理要素はすべて自前で揃えている。

しかし、企業全体の活動は、こうした個々の業務システムの結果を統合することで動いているため、何らかの形で相互連携を実現し、取りまとめていく必要がある。前述の、出力帳票を見ながら再度データ入力する、という方法も、条件によっては現実的な解の1つではあるのだが、これではあまりに非効率的であることは明らかだ。そこで、各業務システム間を連携させ、必要な情報をやりとりする仕組みが必要となった。

初期の段階では、相互連携が必要なシステム同士を再設計し、独自に連携を実現するという手法が採られた。これは、考え方としては、従来独立したシステムであった業務Aと業務Bを、新たな統合業務である業務Cにまとめ、業務Cのためのシステムを新たに設計する、という手法だといえる。つまり、システムの粒度が上がり、内部に比較的独立性の高い複数の業務が存在すると捉えているのである。

従来の個々の業務システムを完結した全体と見る見方からは、新しいシステムは複数システムを連携させ、統合したものと見えるが、根本的な考え方としては、やはり独立して完結したシステムを構築するという手法のままだといえるだろう。

こうした発想で多数のシステムを統合していくと、システム間の連携をその都度個別対応で作り込んでいく、という結果になる。新たに連携を実現する必要のあるシステム間で連携のために必要なデータ形式やデータ交換のためのメカニズムを摺り合わせ、個別に実装していくわけだ。こうした手法は、既存のシステムに対する変更を最小限に留め、必要最小限の追加投資で連携を実現できる可能性がある。しかし、連携を必要とするシステムの数が少なければよいが、システムの数が増えてくると、接続が増えるにつれて作業量が爆発的に増加するという問題がある。そこで、次に試みられたのがハブ&スポーク・モデルによる相互接続である。ハブとは車軸のことであり、スポークは、車輪と車軸を繋ぐ放射状の支持構造である。自転車の車輪を思い浮かべて頂ければ、イメージが沸くだろう。

ハブ&スポーク・モデルでは、中央に相互連携とデータ交換のための専用サーバを設置し、各システムはこのサーバとのみ接続する、という形になる。システム間の連携は、このサーバが中継して実現するのである。この場合、システムを接続する際に新たに開発する必要があるのは中継サーバと接続するためのメカニズムだけで、連携相手となるシステムに個別に対応する必要はなくなる。そのため、接続すべきシステムが増加しても、開発負担は最小限で済む。新たに接続することになったシステムと中継サーバの間の接続さえ実現すれば、既存のシステムには何ら手を加える必要はない。

ハブ&スポーク・モデルに基づくシステム連携は、EAI(Enterprise Application Integration)として実装され、中継サーバはEAIツールとして販売されている。システム間の連携では、データの形式を変換したり、通信の開始/終了を制御したりといったさまざまな追加機能が必要になり、従来の個別接続のアプローチでは、接続するシステム間で毎回こうした機能を作り込んでいたが、EAIでは、こうした変換や調整の機能をEAIツールにまとめて組み込んでしまう。

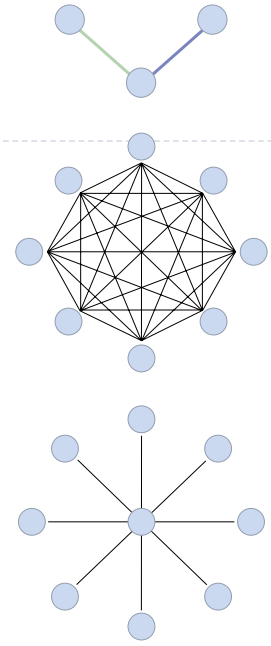

図2:ハブ&スポーク・モデルによる相互接続の簡素化
システムごとに専用のインタフェースを使って相互接続するのは、数が少ないうちはいいが、システム数が増えると接続も膨大な量になる。ハブ&スポーク・モデルでは、接続数を大幅に削減できる。

EAIによって、分散したシステムを相互に接続する基礎はできあがったのだが、長期的な発展を考えると、インタフェースが安定した標準的なものであることが求められる。つまり、各システムとEAIツールとの接続が、EAIツール独自のインタフェースで実装されると、EAIツールに囲い込まれ、変更が難しくなるという問題が生じるのである。

そこで、まずデータ自体をXMLで標準化することで個別のデータ変換プロセスを実装する必要を排除する方向に進んだ。続いて、インタフェースにも標準が確立された。それが、Webサービスということになる。XMLによるデータ表現とWebサービス・インタフェースによる相互通信を前提とすれば、実は接続自体はハブ&スポーク・モデルに拘る必要はなくなる。要は、すべてのシステムが相互接続のための標準インタフェースを備えており、それを使って接続できるのであれば、システム数が増えることによる負担の大半が排除できることになる。こうして、Webサービスに基づく相互接続によって分散したシステムを統合する動きが強まり、それがSOAへの注目を高めることになった。

FS Feature Story

第2部 技術動向

Webサービスの拡張

Webサービスは、XMLによるデータ表現と、SOAP（Simple Object Acess Protocol）に基づく通信を基本としたプログラム・モジュールの相互連携のメカニズムである。そもそもの発想は、インターネット上で提供されている多数のサービスを統合／連携させて付加価値を生み、新たなサービスを作り出すことにある。しかし、一足飛びにインターネット上でのサービス連携に進むのはやはり無理があり、現実には企業内でのシステム連携や、イントラネットを介した企業グループ内での特定システム間の連携に使われ始めたというレベルに留まっている。

当初はWebサービスの仕組みさえあれば、どのようなサービスの連携でも実現できるかのような語られ方もしたが、実際にはそのようなことはなく、連携させるサービスごとにさまざまな合意事項が必要である。そのため、各種業界団体を舞台に、Webサービスのための周辺規格とでもいうべきものが次々と標準化されつつある。ごく単純に言えば、Webサービスで交換されるデータはXML形式になるので、業界ごと、あるいはサービスごとに、XMLデータのためのスキーマを統一し、利用しやすいデータとなるように、あらかじめ準備を整えておくという作業が次々と進んでいるのだと表現してもよいかもしれない。また、通信プロトコルとして使われるSOAPはごく単純なメッセージ交換の機能だけを実装してあるため、企業間でのサービス呼び出しに必要となるセキュリティや認証、確実性を保証するための問い合わせ／確認など、さまざまな追加機能を付け加える必要がある。技術的には、XML+SOAPという基本的な要素を使えばXMLデータの交換は可能になるわけだが、ビジネスの現場で使うためには、より高度な機能も揃っていないと安心して使えるものにはならないということでもある。

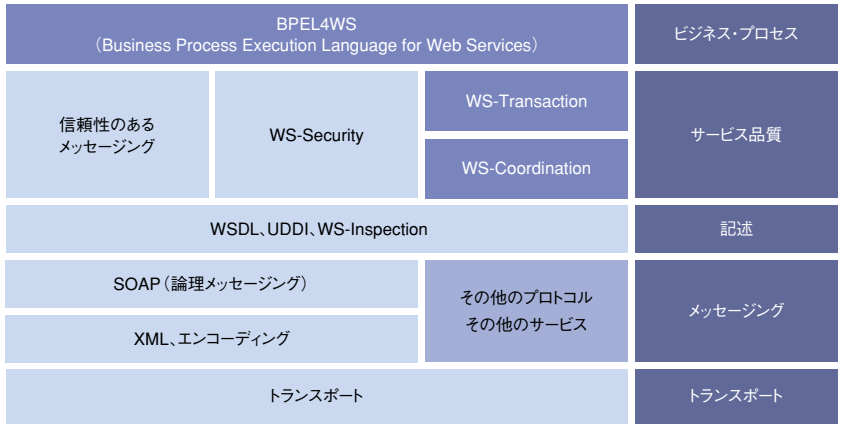

図3：Webサービス技術の概要

Webサービスは、現実には比較的低レベルのインタフェースであり、「サービスの連携」という高度な機能を実現するためには、不足している要素が多い。不足を補うための追加仕様の標準化では、OASIS（Organization for the Advancement of Structured Information Standards）などの業界団体が重要な役割を果たしている。OASISは、eビジネスの標準を開発／推進する非営利の業界団体である。ここでは、ビジネス利用のためのXMLの標準規格であるebXMLや、セキュリティのための標準規格SAML（Security Assertion Markup Language）といった各種の規格制定に精力的に取り組んでいる。

また、もう1つの重要な業界団体としてWS-IもWebサービスを現実的なビジネスの現場での要求に対応させるための活動を活発に行なっている。WS-I（Web Services Interoperability Organization）は、Webサービスの互換性を確保するための活動を中核として、標準プロファイル（Basic Profile）の策定を行なっている。

Webサービスの追加仕様

現在OASISでは、Webサービスの実用化に向けたさまざまな標準規格の策定に取り組んでいる。ここで、その内容をいくつか紹介しておこう。

現在Webサービスの分野でOASISに

設置されているTechnical Commitee（TC：技術委員会）は14ある（表1）。いずれも、Webサービスを実ビジネスに応用していく上では重要な機能や規定を扱っているのだが、特にSOAを考える上で重要と思われるものについて、以下に簡単に説明していこう。

まずは、ebSOA（OASIS Electronic Business Service Oriented Architecture）だ。ebSOA TCの活動は2004年4月に始まったばかりだが、TC結成以前に検討されたドキュメントや仕様のドラフトはすでに公開されており、OASISのWebサイトで参照することができる。今後の活動から、最終的にはebSOA技術仕様（eBusiness Service Oriented Architecture Technical Specification）が策定され、さらに、ebXMLとWebサービスを利用したSOAのベスト・プラクティスについて述べたドキュメントが作成される予定になっている。このTCが正式な成果を発表すれば、将来的にはSOAの標準的なアーキテクチャ・パターンとして活用される可能性が高い。

また、SOAを睨んだ技術要素としては、やはりビジネス・システムが必要とするさまざまな通信に関する機能が重要となる。

たとえば、ASAP（OASIS Asynchronous Service Access Protocol）は、非同期に、あるいは長時間かけて実行されるWebサービスを制御することを可能にしようとするものだ。

また、OASIS Translation Web Servicesは、Webサービスの翻訳やローカライズのための仕組みを考えていこうとするもので、現在米国中心に発展しているITを日本国内で展開しようとする場合に避けて通れない日本語化の問題にも大きく関わってくるだろう。

また、実ビジネスでWebサービスを利用していくなら、管理の問題は無視できないため、WSDM（OASIS Web Services Distributed Management）の取り組みは実際にサービスを展開する段階で重要になるはずだ。

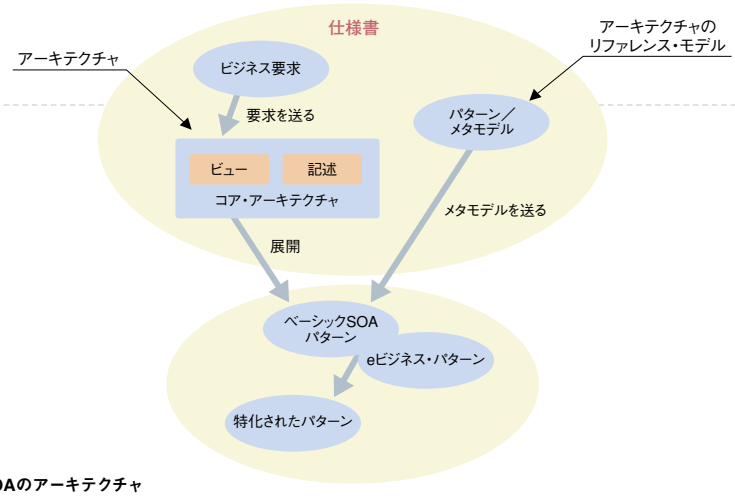

図4：ebSOAのアーキテクチャ

WSN（OASIS Web Services Notification）は、Webサービスが相互に情報交換するための仕組みの開発に取り組んでいる。サービスのステータスを確認したり、サービス停止などがあった場合に、それを確認する手段はSOAPのレベルでは実装されておらず、応答が返らなければ何かトラブルがあったと考える、という少々原始的な手法しかない。しかし、サービス品質を求めるならこれでは不十分であると考えられ、WSNの必要性が浮上する。

さらに、メッセージ交換の確実性を求めるなら、高信頼性プロトコルが必要になる。これに取り組んでいるのがWSRM（OASIS Web Services Reliable Messaging）である。重要なトランザクションを実行する際には、確実な処理を保証するために、2ウェイ・コミットメントなどの技法を使って処理の確実性を保証し、途中で中断された場合には確実に処理開始前の状態に復元するためのロールバック処理などが組み込まれる。これらは、従来から基幹のメッセージング・システムで利用されてきた技法だが、WebサービスやSOAなどのような、分散システムの相互連携ではメッセージ交換をベースにしていることから、こうした技術も必要となるのは明らかだろう。

ビジネス・プロセスの記述

さらに、Webサービスを使ってSOAにつなげていくために不可欠と考えられているのが、ビジネス・プロセス記述言語である。ごく単純に表現すると、Webサービスで提供される個々のサービスをどうまとめて必要な処理を行なうか、複数のWebサービスをどのような順序で実行し、その結果に応じて適切な処理に分岐していくといった制御をどう実現するか、ということである。こうした複数のサービスを一定のシナリオに当てはめて全体の処理の流れを定義するための記述言語

OASIS Asynchronous Service Access Protocol（ASAP）TC
OASIS Electronic Business Service Oriented Architecture（ebSOA）TC
OASIS Framework for Web Services Implementation（FWSI）TC
OASIS Open Building Information Exchange（oBIX）TC
OASIS Translation Web Services TC
OASIS UDDI Specification TC
OASIS Web Services Business Process Execution Language（WSBPEL）TC
OASIS Web Services Composite Application Framework（WS-CAF）TC
OASIS Web Services Distributed Management（WSDM）TC
OASIS Web Services for Remote Portlets（WSRP）TC
OASIS Web Services Notification（WSN）TC
OASIS Web Services Reliable Messaging（WSRM）TC
OASIS Web Services Resource Framework（WSRF）TC
OASIS Web Services Security（WSS）TC

表1：OASISに設置されたWebサービス関連のTC

が、ビジネス・プロセス記述言語である。標準候補にはいくつかの仕様案があるが、OASISで取り組んでいるのがWSBPEL(OASIS Web Services Business Process Execution Language)である。

サービスの粒度(細かさの度合い)にも依存するが、さまざまなサービスから呼び出される可能性の高い汎用的なサービスでは、逆にあまり特定の処理に特化した機能を含むことはできなくなる。そのため、こうしたサービスを組み込む場合、どうしても不足する要素が出てくるため、不足部分をカバーする別のサービスを用意して全体を構築する必要がある。

複数のWebサービスを統合して1つのビジネス・プロセスにまとめ上げる手法は、現時点ではまだ確立されたものはない。ソフトウェア・ベンダー各社は、プログラミングの知識を持たないビジネス担当者でも扱えるようなビジネス・プロセス構築ツールといったものの製品化に取り組みつつあるところだが、標準的なビジネス・プロセス記述言語がなければ、作成したビジネス・プロセスが特定のツールに依存してしまうことになる。

SOAの本格的な普及に向けて、ビジネス・プロセス記述言語の標準化は重要な意味を持つことになる。

システム・アーキテクチャ

Webサービスは、SOAを考える上で基本となる技術要素だが、実装面と深く関わる存在であり、いわばシステムの最下層に位置するといえる。一方で、SOAはシステム全体を貫くアーキテクチャであり、実装技術の詳細とは離れ、システム全体を上から見下ろす視点だと表現できるだろう。

こうした、IT時代の企業システム・アーキテクチャを再構築する動きとして先行したのが、EA(Enterprise Architecture)である。EAは、特に大規模な組織を想定し、その内部の業務システムの標準化や組織の最適化を進めることで組織運営の効率化を実現しようという考え方で、特定のシステム・アーキテクチャを具体的に定義するものではない。しかし、EAの考え方に従って企業システムを見直すことで、長い時間をかけて構築されてきた、部分最適な業務システムを人手によって統合するような、いわば「古い革袋に新しい酒を注ぐ」ようなシステムを全体最適なシステムへと移行することが可能になるはずだ。

EAはジョン・ザックマンによって提唱された概念で、1987年にIBM Systems Journalに掲載された"A Framework for Information Systems Architecture"という論文が最初である。最近になって突然出てきたアイデアではないのである。ただし、この考え方が実際のシステム・アーキテクチャに反映されるには、それなりに時間も必要だったし、インターネットの普及やコンピュータ・システムの低価格化など、さまざまな周辺環境の整備を待つ必要があったということだ。

EAに関しては、日本政府の「電子政府構築計画」にも影響を与えたことで注目されており、今後の企業システムの再構築に当たって無視できない動向の1つとなっている。

では、EAに基づいて企業システムを構築するにはどうしたらよいだろうか。採りうる選択肢はさまざまあるはずだが、ここで設計の際の考え方の1つとして、SOAを位置づけることも可能だろう。つまり、EAを最上位のシステム・アーキテクチャに据え、その下位のより具体的なアーキテクチャとしてSOAがある、と考えるわけだ。そして、前述の通り、より下層の実装技術にはWebサービスとその関連仕様群が置かれる。

SOAとは何か

では、具体的にSOAの内容を見ていこう。SOAは、その名称からも明らかなように、システムをサービスの集合体とし

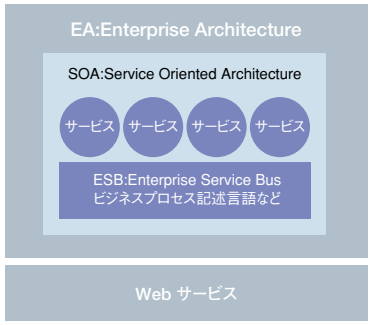

図5:EA、SOA、Webサービスの関係

て構築しようとする設計手法である。サービスとは、要求に応えるためのまとまった処理の単位であり、完結した一群の処理からなると考えられる。SOAでは、サービスを「ビジネス・プロセスの実現のために利用できるコンポーネント化されたアプリケーション機能」と捉える。従来型の完結した業務システムそれ自体をサービスとしてもよいし、もっと細かい共通機能をサービス化することもあるだろう。

プログラミング言語においては、OOP(Object Oriented Programming:オブジェクト指向プログラミング)という考え方があるが、SOAはこの考え方をシステム・アーキテクチャのレベルに展開したものと考えることも出来るかもしれない。OOPでは、プログラムをオブジェクトの集まりとして構築し、オブジェクト間のメッセージ交換によって処理を進めていく。オブジェクトは、データとデータに対する処理をひとまとめにした単位であり、オブジェクトにメッセージを送る側はオブジェクトの内部の詳細を知らなくても、公開されたインタフェースを使用してメッセージを送り、処理結果を返してもらうことができる。

SOAにおけるサービスの考え方は、OOPのオブジェクトとよく似ている。システムはサービスの組み合わせによって構成され、サービス同士のメッセージの交換で処理が進められる。個々のサービスの中身については知らなくてもよく、単にそのサービスが何を提供してくれるか、サービスを利用するためにはどうすればよいかを知っていればよいのである。

Webサービスに基づいてSOAを実現するのであれば、サービスはWebサービスそのものであり、交換されるメッセージはXMLデータということになる。さらに、Webサービス間を適切に連結するのは、ビジネス・プロセス記述言語で記述されたビジネス・プロセスとなる。

逆の見方をすると、2001年頃から急速な盛り上がりを見せたWebサービスは、結局のところ単独ではビジネス・システムを根本的に変革するだけの実用性を持ち得なかったといえるだろう。Webサービスが実現したサービス間の通信メカニズムを実際に企業システムに組み込むためには、現在整備が進みつつあるさまざまな周辺規格と、その全体をまとめるシステム・アーキテクチャとしてのSOAが必要だったと考えることもできる。歴史的には、Webサービスを置き換えるような形でSOAが注目されるようになってきているが、それはSOAがWebサービスを実ビジネスで使えるようにするための努力から生まれたものだからだといっても過言ではないだろう。

SOAの構成要素

SOAは、サービスの集合体として構成される。そのため、まずはサービスがもっとも基本的な構成要素となる。サービスの実装にはWebサービスの技術が利用され、データやメッセージはXMLで表現される。

さらに、サービス間でのメッセージ交換を実現する機能が必要となる。Webサービスは、サービス間での基本的なメッセージ交換をSOAPを使って行なうが、これでは単にあるサービスが別のサービスを呼び出すことができるというだけである。前述の通り、ビジネス・プロセスをサービスの集合として構成するためには、ビジネス・プロセス自体をビジネス・プロセス記述言語を用いて記述し、そこで規定されたシナリオに従って必要なサービスを次々呼び出す必要がある。

こうした、SOAを実現するための通信インフラを、ESB(Enterprise Service Bus:エンタープライズ・サービス・バス)と呼ぶ。ESBを基盤として、その上に各種のサービスを載せて相互連携を実現する、という考え方だ。ESBはSOAを実現するためのミドルウェアとして提供され、そこでビジネス・プロセス記述言語で記述されたビジネス・プロセスが実行されることになる。つまり、概念的にはESBはハブ&スポーク・モデルで実現されたEAIシステムのハブ部分に置かれるということになる。

なお、ESBには、標準的なSOAPに加え、さまざまなメッセージング・サービスを実装する必要もあるだろう。OASISで検討中のASAPやWSN、WSRMなどはもちろんだが、このほかにもビジネスシステムで実際に使用されているメッセージング・サービスを一通り用意しなくては、既存システムの機能をSOAに組み込むことができなくなる。

こうしたことから、SOAの実現に関しては、既存のエンタープライズ・ソフトウェア/ミドルウェアの要素が多数統合されることになるだろう。EAIツールの多くは、わずかな機能拡張でSOAの中核的な存在に成長できる可能性が高い。また、MQなどに代表されるメッセージング・サービスのためのミドルウェアの機能も、SOAに組み込まれていくことになるだろう。さらには、ビジネス・プロセス記述言語を使用するための簡便なユーザー・インタフェースとしてビジネス・プロセス開発ツールといった製品の投入を表明しているソフトウェアベンダーも複数ある。さらに、複数のサービスの状態を把握するためのモニタリング・ツールや、サービス品質(SLA:Service Level Agreement)を実現するためのパフォーマンス管理ツールなども必要だろう。さらに、企業間で相互にサービスの利用を行なうようになれば、サービスの利用状況を精密に測定し、課金を行なうためのシステムも必要になると思われる。

現時点では、SOAの実現に向けて業界が動き出した段階であり、足りない要素を数え上げていくと膨大な量に上ってしまう。そのため、SOAの概念を取り入れてシステム設計を行なうことは有効であっても、現時点で理論通りのSOAシステムがすぐに運用可能だとは考えない方がよいだろう。長期的な視点で、企業システムの再構築を行なうつもりで腰を据えて取り組むことが重要だと思われる。