

SOAのテクノロジーと日本の環境

欧米IT文化の華、SOAを取り込むために必要なこと

SOA、サービス指向という言葉はまさにホットな言葉になっておりSOA化を目指す企業が増えています。SOAを実践するには、単なる製品の導入でなく、テクノロジーの理解そしてその背景にある物事への理解が必要ですが、より重要なのは今まで慣れ親しんだ仕組みや方法論を少しずつ変えなければいけないことです。テクノロジーだけの吸収ではSOAは実践できません。欧米企業のIT部門では、WebサービスやXMLは、Java™の利用と同様に常識化していますし、反復型やアジャイルな開発手法も日常です。聖書に次の言葉があります。「新しいぶどう酒を古い皮袋に入れはしない。もしそんなことをしたら、その皮袋は張り裂け、酒は流れ出るし、皮袋もだめになる。新しいぶどう酒は新しい皮袋に入れるべきである。」SOAはその意味では新しい酒です。入れ物には知恵が必要です。

Article 3

SOA Technologies and Japanese IT Environment

Nowadays, the expressions "SOA" and "Service Oriented" are very hot topics, and there is an increasing number of businesses striving to implement SOA. In order to implement SOA, rather than just buying and implementing a product, businesses need to understand technology and they need to understand the business environment surrounding SOA. However, more importantly, they must progressively change the mechanisms and methodologies with which they have previously been accustomed. SOA cannot be achieved just by consuming technology. IT departments at businesses in Europe and the U.S. now use XML and Web Services to the same degree as Java™. It is now also common for them to use iterative or agile software development methodologies. The Bible contains the following proverb: "No one pours new wine into old wineskins. Otherwise, the wine will burst the skins, and both the wine and the skins are ruined. Rather, new wine is poured into fresh wineskins." In this context, SOA is a new wine. Thus we need to be wise when it comes to the container.



日本アイ・ビー・エム株式会社
ソフトウェア事業
ディステイングイッシュト・エンジニア(技術理事)

清水 敏正 Toshimasa Shimizu

[プロフィール]

IBM入社後、ホスト、分散システム、LAN、OS/2などの技術サポートを経て米国テキサス州オースチンに5年間赴任。帰国後1999年からWebSphereの技術サポートに専念。多くのお客様プロジェクトを開発部門に結び付ける役割を果たす。2004年にDE就任。その後日本でのSOAの普及を目指して社内外での活動を実施中。

① リフォーム文化とSOA

SOA(Service Oriented Architecture : サービス指向アーキテクチャー)はリフォーム文化であるということと突拍子もないと思われるかもしれませんが、その辺りからSOAを解き進めていきたいと思います。現在の企業には、1970年代からのさまざまなシステムが作り込まれており、全体が網の目のようになっています。あたかも建て増しを繰り返してきた大きな家のようです。欧米には古いものを忌み嫌わず、使いこなしていく文化がありますが、IT(情報技術)文化にもリフォームとDIY(Do It Yourself)が色濃く出ているのが日本との違いでしょう。

米国東海岸には、築100年というような古い住宅が数多く建っています。持ち主が何回も変わっても基本は修理し改築して使い続ける文化です。30~40年ほど経っている住宅を自分で大きく改築する隣人の姿には驚きます。DIYという言葉の一般的な日本語訳は間違いです。つまり日曜大工のできる程度の作業



ではありません。日本のホームセンターとは比較にならないほど巨大なHome Depotでは、家一軒を建てる材料がすべてそろいます(写真参照)。

アマチュアだけでなくプロの職人も10%は利用しているというレベルです。屋根の高い倉庫のようなHome Depotで必要な材料を購入し、難しい施工はプロの職人を時間/日単位で契約します。つまり、改装の計画と管理、材料の仕入れはすべて自分でいき、かつ自身でもかなりの作業を行うのですが、専門性が必要な作業や量が多い作業などは、大工などを必要日数で契約してDIYします。

その辺りの考え方が、まさに米国のITプロジェクトのこなし方と日本との差にもつながるのです。つまりSOAはリフォーム型であり、かつ業務を責任持つ当事者が仕切って(DIY)やらなければいけないところに特徴があります。

② 請負型開発からの脱却そして反復型開発へ

DIY文化は個人レベルだけでなく、一般の企業にも存在します。米国企業のIT部門がいわゆるSI(System Integrator)会社に発注する形態も、依頼する仕事をはっきりと定義して支払いはT&M(Time and Material)契約をベースに行うということが普通に行われます。そこで重要なのは仕事を依頼する側は、成果のチェックや進捗管理は、自分の責任として行います。別の言葉で言うと、お金だけ用意してSI会社に丸投げするような依頼は米国の文化には見られません。SOAと話がつながるのはその点です。SOAは、既存・新規のアプリケーションをサービスという単位でくり、それらを組み合わせるだけでさまざまに再利用できるIT構造にするのを大きな目標とします。とはいっても「サービス化」は自社のIT部門では未知の分野であり、かつSI会社にも経験がなく、業務部門との密接な会話も必要になりますので、従来型のアプリケーション開発のように、すんなりと全体の作業量が見積もれるはずがありません。

DIYから学ぶ第2点ですが、スクラッチ&ビルドの

建て直しでなく、リフォームがSOAの基本であり少しずつの改修(スモールスタート)が自然である理由は、この「SOA化した場合の全体の難易度や開発量は、最初からは分からない」という点です。作業量は300万ステップで、2年間で500人月、XX億円です、というような見積もりはまったく不可能でしょう。古くからのウォーターフォール型を前提とした巨大な開発プロジェクトとSOAは、おそらくとても相性が悪いでしょう。

日本では建主は、ほとんどがハウスメーカーなり工務店なりに一括で契約し、材料の仕入れも職人の雇入れもすべて直接関与しません。会社を信用して任せるわけです。このため知名度の高い大企業ほど信用度があり、受注しやすい構造になっています。これと対比して米国の企業のIT部門のやり方を見ると、材料は自分で選び買い入れる(コンピューターのハードウェアやソフトウェアを部品として買う)そしてコンサルタントやITのエンジニアは自社内調達もしくは外部から時間で調達して、プロジェクトは自分で管理する、というスタイルが一般的です。SOAを成功させるためには、エンドユーザーである企業のIT部門が、業務部門と連携し合いながら少しずつ「サービス化」を試行することが必要です。従って、未知の分野で作業するプロジェクトは、材料(ベンダーのSOA対応ソフトウェア製品/技術)を購入し、専門家(SI会社)を出来高払いで契約する、といったスタイルが自然でしょう。

しかし、それを日本で実現するには確かに大きな障壁がさまざまな部分にあることは確かです。まず企業内部には、優秀な人材をIT部門内部に育てること、かつITアーキテクチャーだけでなくプロジェクト管理の専門家であるプロジェクトマネージャーも養成する必要があります。IT部門を別会社にするケースは米国でも普通にありますが、基本精神は同じです。

IT部門は、一般にいわれるアジャイルソフトウェア開発手法(開発期間の短縮やコスト削減を実現しながら、高機能なソフトウェアを迅速に開発する手法)またはIBMのRUP(Rational Unified Process®)を適用した開発方法などに慣れる必要があります。そして、サービス化の案件を数週間のプロジェクトとして定義し、自社開発が困難ならば最適と思われるSI

会社を選び発注することです。最初のサービス開発の経験は、次の反復に生かされることとなります。一方、受ける側のSI会社も、従来通りの成功体験であるウォーターフォール型の開発手法でなく、反復型などの手法による細切れ受注とT&Mの出来高払いに対応する必要があります。卵が先か鶏が先かという話ではなく、両者が同時に新手法に向かって動かないことにはSOAの手法に基づいたアプリケーション開発は不可能のように思えます。

つまりSOAには新しい開発技術としてのRUPなどが必須なのです。エンドユーザーである業務部門との試行錯誤があって初めてサービス定義の一つが完成するのが実情でしょう。ここまでをまとめますと、SOAを自社内で成功させるためには、RUPなどの反復型開発手法による少しずつの開発、業務部門との密接な協業関係による強力なプロジェクト管理、請負型でなくDIY、T&M契約などがキーワードでしょう。

3 ビジネスモデリングとSOA

次にビジネスモデリングとSOAの関連性を見てみましょう。

IBMのSOAライフサイクルでは、業務プロセスの個々の処理内容やフローをモデリングし、「あるべき姿」を決めていき、それをベースにツールによってJava™のコードまで生成して実行環境に導入するという一連の流れを推奨しています。そして設定したビジネス目標が達成されたかどうかをモニターし、元のビジネスモデルに戻して必要な修正を施すことで、全体の流れを何回も回すわけです。ここで問題になるのが、ビジネスをモデリングできっちりと表現していくという行為です。

マクドナルドの店舗においてそれぞれの店員が行うべきマニュアルの存在は、昔から米国式ビジネスの典型として語られましたが、ほとんどの日本企業では、入社した社員が果たすべき役割をきっちりと定義した業務マニュアルを持たないでしょう。仕事のやり方は先輩が教えるというやり方が日本の長い文化ですが、最近の一時雇用の増加、グローバルリソースへの業務の移転などの要因は、業務そのものと業務

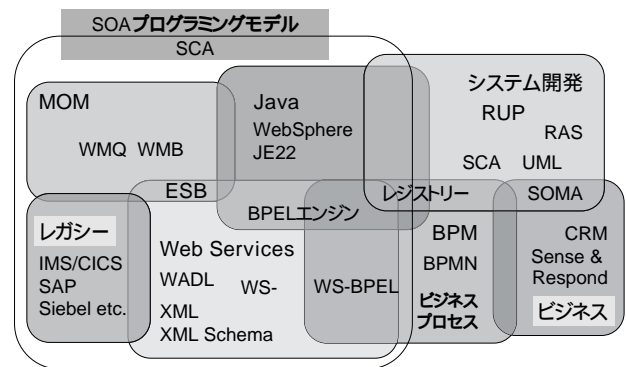


図1. SOAテクノロジーの全体図

の流れを明確に見える形にする「見える化」のニーズの高まりとともに、モデリングのテクノロジーが注目されてきた兆候があります。

図1に、筆者が最近使っているSOAテクノロジーの全体図を示します。

この図は、SOAのテクノロジーを理解する際の全体的な鳥瞰図^{ちようかんず}ともいえます。図中の「WS - 」は、Webサービス関連の多くの仕様をまとめて表現するときによく使われる表記です。WSアスターと呼ばれます。これらのテクノロジーはツールや実装環境を提供するベンダーが深く理解し使用するべきものです。限られた誌面でこの図で示すすべてのテクノロジーを解説することは困難ですが、幾つかの重要なものについて触れます。システム開発の領域で重要なのは2章で「SOAでは反復型開発が必要」と紹介したRUPです。

4 中間データの意味付けの重要性

図1のWeb Servicesの領域にある「XML Schema」ですが、日本の多くのIT企業におけるSOAへの取り組みでは、このXMLスキーマという地味な技術に対する意識に問題があるように思います。XMLそのものはほぼ広く理解されてきたと思われませんが、企業内部のアプリケーションプログラムが処理する入出力データのXMLスキーマを決めなければならない、という理解はまだ普及していないようです。

ここ10年近くの大規模ITプロジェクトを実現してきたのは、OO(Object Oriented: オブジェクト指向)ではなくDOA(Data Oriented Approach: データ

中心型アプローチ)です。SOAで目指すのは、プロセスインテグレーションです。つまりビジネスプロセスの基本部品を「サービス」という単位で表し、それらの組み合わせ・統合によりさまざまなアプリケーションを作ります。既に企業内にはさまざまなデータベースが現存し、データインテグレーションはプロセスインテグレーションよりもはるかに困難です。

SOAでは、それらのデータの違いを正すことを前提にはしません。企業内部の重要なデータは、OS(基本ソフトウェア)のファイル形式、リレーショナルデータベースの形式、Excelなどの形式、XML形式など、さまざまな形で存在するでしょうが、SOAではそれらをあえて一つに統一することはありません。同じ企業でもプログラムによっては、まったく違う表記を使うことがあります。データの意味付けを定義するデータのことを、メタデータといいますが、SOAでは中間データの意味付けが非常に重要です。XMLスキーマは、スキーマ言語によりXMLデータとして正しい表現、許される表現、意味のマッピングを実現します。

SOAの基本は、技術的には「サービス」が存在し、その「サービス」の意味がXMLの表現としてWSDL(Web Services Description Language)という標準に準拠したフォーマットで表現され、入力データと実行される機能、そして出力データが明確に定義されていることです。そこで出てくる入力データと出力データの基本はXML形式です。そのXMLデータに意味を与えるものがXMLスキーマなのです。

1970年代から急速な進歩を続けているITは、昔はバイナリーやパック方式、コード化(男性を1、女性を2とするなど)が基本でしたが、豊富なメモリー、非常に高速なCPUの時代には、メタデータをしっかりと定義し管理していくのが最優先となっています。

スキーマを表現・定義する言語は、非常にベーシックなDTD(Document Type Definition: 文書型定義)、RELAX(Regular Language Description for XML)、そしてWebサービスの世界で幅広く使われ事実上の世界標準になっているXMLスキーマがあります。企業の内外を往復するデータは、WebではHTTP(Hypertext Transfer Protocol)上のHTML(Hypertext Markup Language)ですが、Webサービ

スではXMLです。

最近Web 2.0の言葉とともに、AjaxというXMLを使用した手法がGoogleなどで大いに利用され、XMLに対するアレルギーも多少減ってきていると思われませんが、日本の企業でXMLを標準として普通に利用したシステムが動いている例は極めて少数だと思われまます。SOA化ではXML化のアレルギーをなくして、普通に使用することが必須です。XMLを扱う技術は、Java系およびマイクロソフト系で進んでいますが、IBMではホストのCICS®やIMS™のCOBOLの世界にまでXMLデータの扱いを拡張して提供しています。

作成されたXMLスキーマは、企業内で一つの倉庫に保管されるべきです。SOAレジストリーまたはリポジトリーという機能を持ったソフトウェアが必要になります。図1に示した「レジストリー」がそれです。レジストリーにはXMLスキーマの定義だけでなく、すべての「サービス」をそれぞれ表すWSDLも格納されます。SOAの実現に必要なWSDL、XMLスキーマの説明に加えて、個々の「サービス」の実行場所、プログラムの実態を利用するユーザーから隠す、あるいは仲介人を通してユーザーに提供する役割があります。それをわたしたちは、ESB(Enterprise Service Bus)と呼びます。ユーザーが期待する役割は、入力されたサービス呼び出しのリクエストを分析して、ログインしたり、送付先の経路を変えたり、プロキシのように自分がキャッシュした内容を返したり、稼働時間外には即その状況を返したりと、あえて言えば、クライアントとサーバー(サービスを提供する側)の関係をさらに緩やかにすることです。これを疎結合といえます。

⑤ WebアプリケーションのSOA的改造

もちろんWeb側の処理プログラムから目的のWSDLを直接呼び出す方が性能的には当然優れています。性能低下を補って余りあるメリットを備えているので、ESBをメッセージハブやネットワークの幹線のようにレイアウトします。

サンプルとして図2で、Webアプリケーションの改造によるSOA化のアプローチを紹介します。現在多く

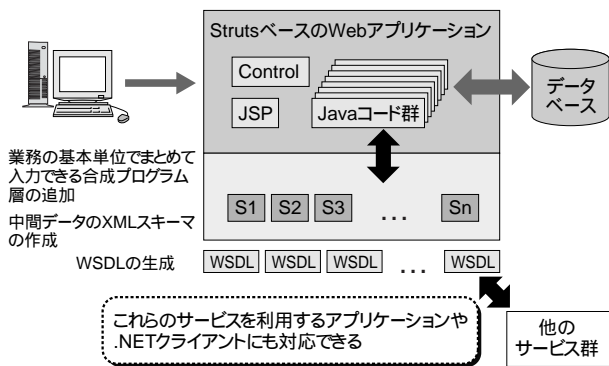


図2. WebアプリケーションのSOA的改造

のWebサイトのアプリケーションが作られている方式(サーブレット、Strutsなど)を発想転換して拡張するデザイン例を示します。

図2に示すように、Webアプリケーションの作りをブラウザとの対話型処理一辺倒でなく、業務的に意味のある単位でバッチ的に入力できるように図中のS1、S2などを追加開発します。例えば受注処理なら、個々の商品・製品の名前・数量・希望納期を20個まではまとめて行う機能がS1です。Webアプリケーションの作りは、コールセンターの担当者が電話で注文を受けるような会話型です。ブラウザ画面におけるあらゆるクリックに対応し、流れを形成し、最終的な受注処理を行います。これに対してSOAの「サービス化」は、ちょうど注文処理をFaxで受けるようなものです。一枚の用紙にすべての必要な項目が書かれてFaxされてきます。受注先の名前、代金支払いの方法、幾つかの商品名と数量、希望納期などがすべて指定されているので、受注処理は1回のFaxの交換で済みます。そのためのバッチから対話型に変換するプログラム群(S1-Sn)を作りWSDLを用意すれば、SOA化の第一歩が完成です。こうして作った「受注サービス」などはVisual BasicやMicrosoft®.NETクライアントからWeb Servicesとして呼び出す使い方もできますし、企業内部のほかの用途でも呼び出すことができます。

SOAを理解し実践するためには、何を置いても第一歩を踏み出すことが必要です。ESBの構築を考える前に、既存アプリケーションのサービス化を考えましょう。画面の作りは最終ユーザーの理解を得る必要があるというのは常識になっていますが、「サービ

ス」も同じです。業務部門とIT部門が討議を重ね、将来にわたって意味のありそうな「サービス」を数個・数十個・100個と作っていくとESBの構築やレジストリーやリポジトリーの必要性も出てきます。

6 SCAによる複合アプリケーションの実現

図1に示したSOAのさまざまな技術で、最近特に重要なものは、SOA用の新しいプログラミング・モデルであるSCA(Service Component Architecture)です。WebサービスやJavaの技術は、新しく標準化されようとしているものが山のようにあり、非常に難解です。

特にSOAでは現存する多数のプロトコルと言語、近い将来のインターネットで結ばれた複合アプリケーションの実現を目標にしているのが、技術や言語の違いを乗り越えたシンプルなComposite(複合)のための仕組みが望まれています。それがSCAです。2005年11月に発表されたSCAは、2006年9月時点でBEA、Oracle、SAP、IBMを中心に全部で15社が集まって仕様の開発とオープンソースでの実装に取り組んでいます[1]。

SCAの目的は、プログラムからのサービス呼び出しを簡略化し、XMLで記述することで自由なプログラムの組み合わせ・合成を可能にすることです。Javaに限らずC言語やPHP(Hypertext Preprocessor)など、言語を混ぜた複合アプリケーションの作成を目標にしています。2007年の早い時期には仕様は完成すると予想されます。SCAは既にIBMの製品では初期実装されており、ESBの内部やWPS(WebSphere® Portal Server)の内部で動いています。SCAの最大の特徴は、テクノロジーへの依存性を断ち切るところ

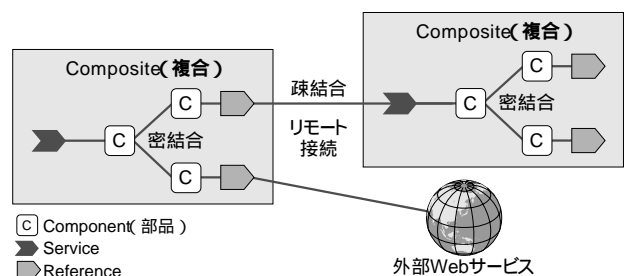


図3. サービスの複合で作るSOAアプリケーション

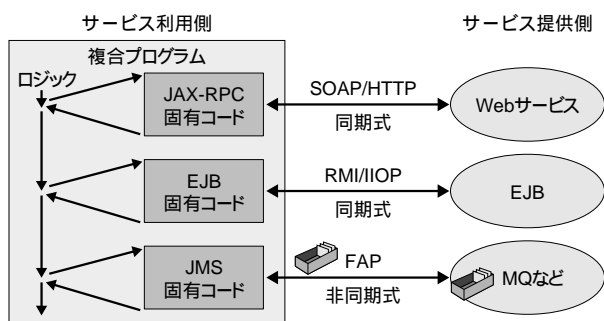


図4. 下位の技術への依存度が高い現在の方式

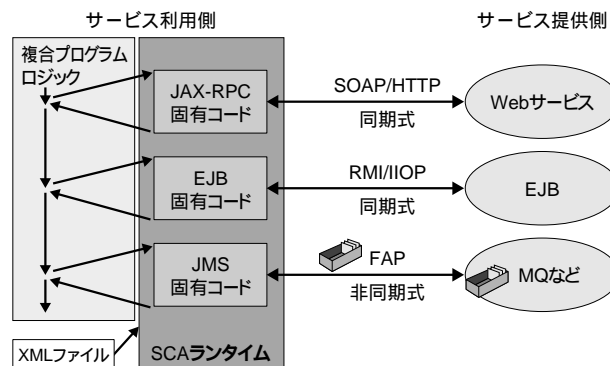


図5. 下位技術への依存度を絶つSCA

です。

図3は、SOAアプリケーションはサービスコンポーネントの複合・組み合わせにより作られるという概念を表現しています。一つの複合コンポーネント(図ではComposite)は複数のサービスコンポーネントをローカルで密結合したものと考えられます。コンポーネント(図中のC)同士の結合はワイヤリングと呼びます。SCAでは、XMLでコンポーネント同士の関係や実装を定義します。リモート接続が必要な場合もリファレンスからサービスにワイヤリングするだけですが、疎結合なので頻繁に呼び出すような設計は性能上、推奨されません。いわゆる粒度の大きなサービスコンポーネントを呼び出すケースが想定されています。

このような複合によるアプリケーションの作成を考えると、従来の技術には利便性に問題があることに気がつきます。図4に、現在のJava技術だけで3種類の異なるプログラムを呼び出す場合の構造を示します。

この図ではプログラム開発者は呼び出すサービスが稼動する条件にあったクライアント側のお作法に合致する方法でコードを書き込む必要があります。プロトコルの違いだけでなく、同期/非同期呼び出しによって、プログラム設計も異なってきます。多くのSI会社は、これを自前のフレームワークで吸収しようとしてきましたが、本来的にはミドルウェアの開発ベンダーが考慮すべきです。図4をSCAで置き換えるとどうなるのかを図5に示します。

SCAの仕様作成と標準化の組織であるOpen SOAというコラボレーションチームには、現在15社が参加しています。SOAの夢として、それらのベンダーのアプリケーションをサービス単位で利用して自由に組み

立てることができる時代もすぐそこです。最近急速に増えているといわれる企業内のPHPアプリケーションも、将来は、SCAで言語を問わずにアプリケーション部品を利用できれば、再利用性の観点から歓迎されるでしょう。

7 SOAに取り組むためには

SOAのテクノロジーは幅が広く、かつ幾つかの技術は相互に関連します。その実践には、今までの技術への取り組みとは違うアプローチが必要です。BPEL (Business Process Execution Language)やESBといった個別の技術・製品を勉強し、評価してもそれだけでは企業内でのSOA化は進みません。これまでに述べた企業文化への考察と変革が必要です。

今までと少し違う発想でのプロジェクトの発案、管理・発注形態の変革、スキルの養成、業務部門とのさらなる協業体制、RUPなどの反復型開発手法の実践、中間データのスキーマの整理・統一など、多くの課題を吸収し乗り越えてこそ、欧米の企業に勝てる日本型のSOA化が見えてくるのではないのでしょうか。

[参考文献]

[1] SCA開発の現状 , Open SOAコラボレーション , <http://www.osoa.org/display/Main/Home>