SOAの新しい標準技術としてのSCAの登場

丸山不二夫（稚内北星学園大学）
中田秀基（産業技術総合研究所）

この1年の間に，SOAの分野でも，小さくはない変化が生まれています．連載の最後として，SOAの世界の新しい技術動向を紹介したいと思います．

まず，最初に紹介したいのは，それぞれが固有の機能を担う再利用可能なサービス・コンポーネントの結合としてアプリケーションを構成し，SOAに基づく開発を容易なものにしようという，SCA（Service Component Architecture）の動きです．

SCAは，SDO（Service Data Object）という技術とともに，2005年の11月に，IBM，BEA，Oracle，SAPら8社のベンダが中心となって，最初のドラフトが発表されました．翌2006年7月には，Sun，Red Hat，Interface21などもこの動きに合流して，最終的には18社が，SCA，SDOの推進を目的としてベンダ中立的なOpen SOAというコミュニティ（www.osoa.org）を形成して，SOAでのベンダ間の連携は深まります．今年2007年の3月には，このOpen SOAでの各社のコラボレーションは，SCA，SDOの最初の仕様をまとめるとともに標準化団体としてOASISを選ぶことを決定しました．

これを受けて，2007年4月，OASISは，SOAアプリケーション開発の簡易化を目指すSCA，SDOの標準化を進めるOpen Composite Services Architecture（Open CSA）という委員会（Member Section）を立ち上げました．こうして，SOAの新しい標準技術の一部として，SCAは登場しつつあります．

SCAの仕様の構成

OASISでの標準化が始まったOpen CSAを構成するのは，SCAとSDOという2つの技術なのですが，本稿ではSDOの紹介は割愛して，SCAの紹介に集中しようと思います．といっても，SCAの仕様は，**表-1**の9つの仕様書からなる，大きなものです．本稿では，残念ながら，SCAの概要の紹介にとどめて，詳細な説明は，別の機会に譲りたいと思います．

ここではまず，SCA仕様の構成から見ていくことにしましょう．まず，SCAの基本となるのが，先頭のAssembly Modelです．サービスをコンポーネント化し，それを組み合わせる（Assemble）ことによってアプリケーションを構成するというアプローチがSCAの最大の特徴です．

SCA Assembly Model V1.00
SCA Policy Framework V1.00
SCA Java Common Annotations and APIs V1.00
SCA Java Component Implementation V1.00
SCA Spring Component Implementation V1.00
SCA BPEL Client and Implementation V1.00
SCA C++ Client and Implementation V1.00
SCA Web Services Binding V1.00
SCA JMS Binding V1.00
SCA EJB Session Bean Binding V1.00

表-1　SCA Specifications Final Version 1.0 2007年3月21日

次のPolicy Frameworkは，セキュリティやメッセージングの信頼性の保証などの，SOAでのサービスの品質（QoS）をコントロールするという，SCAでの新しいメカニズムの導入に関するものです．残りの仕様書群が，1つを除いて基本的には，ImplementationとBindingに関するものだということに注目してください．SCAでは，サービスを，Java，C++，BPELやJavaScript，PHP，Rubyといった多様な言語で記述することが可能です．それぞれの個別の言語でのSCAの実装技術についての説明が，Implementationについての仕様書になります．また，SCAでは，SOAのサービスは，Webサービスによってだけでなく，JMSやEJB Session Beansといった多様なサービス提供技術・ネットワーク技術によってアクセスされます．Bindingとは，サービスへのアクセスの仕方を定めたものです．こうした，ImplementationとBindingの多様さは，SCAの大きな特徴の1つです．このことは，SCAのImplementation/Bindingの仕様は，具体的な技術ごとにさらに拡張されることがあることを示しています．こうした拡張性もSCAの特徴の1つです．

もうすこし具体的に，SCAの特徴を見ていくことにしましょう．

SCAアセンブリ・モデル

SCAでは，サービスのコンポーネントを組み立てて，アプリケーションを構成していきます．次の**図-1**を見てください．回路の配線図のように見えますが，これが

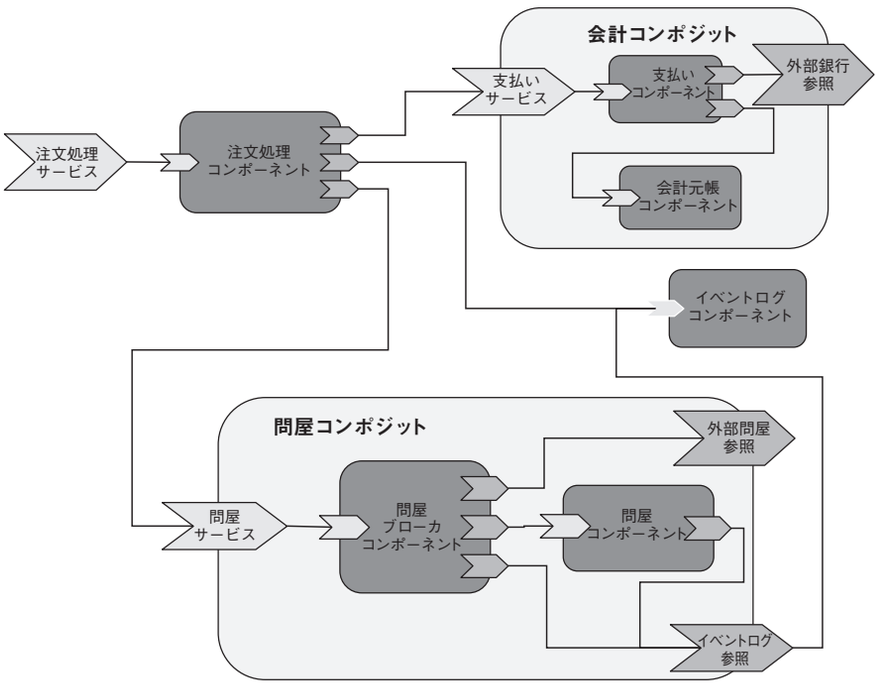

図-1　SCA アセンブリの例（OASIS Tutorial より引用）
http://www.osoa.org/download/attachments/250/SCA_OASIS_Tutorial_part1.ppt?version=1

SCA の「アセンブリ」の一例です．もちろん，こうした図を描けば，アプリケーションが動き出すわけではありません．この回路を構成している部品のうちの「コンポーネント」は，サービス提供の最小単位として，何らかの言語で実装されている必要があります．

図をよく見ると，いくつかのコンポーネントが集まって，「コンポジット」というかたまりを構成していることが分かります．コンポーネントと同様に，コンポーネントの複合である「コンポジット」もサービスを提供する 1 つの単位となっています．

コンポーネントもコンポジットも，他のサービスを参照して，自身のサービスを構成します．先の例では，「注文処理サービス」は，「注文処理コンポーネント」によって担われているのですが，このサービスは，「会計コンポジット」の担う「支払いサービス」と「イベントログ・コンポーネント」の提供するサービスと，「問屋コンポジット」の提供する「問屋サービス」の 3 つのサービスを「参照」して，自らのサービスを構成しています．同様に，問屋コンポジットの提供する「問屋サービス」は，「外部問屋」サービスと「イベントログ」サービスを「参照」しています．

ここで注意してほしいのは，コンポーネントとコンポジットとの違いです．どちらも他のサービスを参照して他にサービスを提供する点は同じですが，基本的には，コンポジットのサービスは Web サービスのようにネットワークを通じて提供されるのに対して，コンポーネントのサービスは同じ VM 内部のメソッド呼び出しのようにネットワークをまたぐことなく提供されます．これまでの SOA のサービスの提供・結合では，Web サービスなどネットワークを通じたサービスの疎結合が中心でした．SCA では，こうした特質をコンポジットでのサービスの提供・結合に残しながら，コンポジット内の，メソッド／関数呼び出しなど，コンポーネント間の密な結合をもサービスの提供・結合として捉えます．こうして，コンポーネント間の密な結合とコンポジット間の疎な結合を組み合わせることによって，両者の特徴を活かして，効率的にシステムを構成することが可能になりました．

SCAでのサービス実装の多様性

SCA の 1 つの特徴は，サービスの最小単位としてのコンポーネントのサービス実装に対して多様な言語を利用することができるということです．SCA の仕様のオープンソース実装の 1 つである，Apache の Tuscany プロジェクトでは，Java，C++，BPEL に加えて PHP，JavaScript, Ruby, Groovy といったスクリプト言語をサービスの実装に使うことができます．現在，C と COBOL への拡張も計画されています．

システムの設計者は，それぞれのコンポーネントが，どのように実装されているかということには関心を持たなくても，それぞれのコンポーネントがどのようなサービスを担っているかに関心を集中すれば，そうしたサービスの組合せとしてシステムを構想することができます．このことは，さまざまな言語で書かれた既存のソフトウェア資産を，それぞれのソフトウェアが固有のサービスを担っているという SOA の観点のもとで，SCA の枠組み

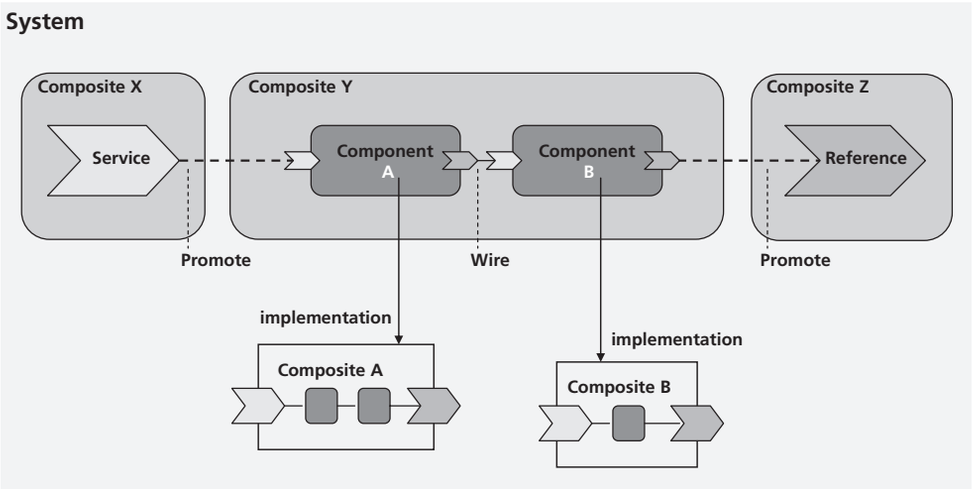

図 -2　コンポジットをコンポーネントの実装に利用する（SCA Assembly モデル仕様より引用）
http://www.osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf?version=1

で統一的に統合し得るということを意味しています．レガシー・システムの新しいシステムへのマイグレーションにとっても，SCA は，非常に柔軟な手法を提供します．

SCA では，コンポーネントのサービス実装を，プログラム言語だけでなく，Spring や EJB といったミドルウェアのフレームワークでも行うことができます．これは，システムが実際にどのようなサービスを提供しているのかが重要であって，それがどう実装・実現されているかは重要ではないという SOA の考え方の 1 つの帰結です．これは，同じ SOA の考え方に基づく，Grid における，Virtual Organization の考え方と共通なものです．

SCA では，こうした考えを推し進めて，あるコンポーネントの実装は，他のコンポジットによって行われるということも認めます．次の**図 -2** を見てください．コンポジット Y の中に，コンポーネント A とコンポーネント B が含まれているのですが，それぞれのコンポーネントの実装は，コンポジット A とコンポジット B によって与えられています．場合によっては，コンポジット A の中のコンポーネントが，さらに，他のコンポジットによって実装されるということがあるかもしれません．

その他の注目すべきSOAの技術動向 −RESTful Webサービス

駆け足で，SCA の紹介をしてきましたが，SOA の分野で筆者が注目している技術を紹介します．

1 つは，Grid での WS-RF（Web Services Resource Framework）から WS-RT（Web Services Resource Transfer）への移行にあたっても，大きな影響を与えたと思われる REST 系の技術です．Java での Web サービスの実装は，JAX-WS にまとめられているのですが，最近，そこから，RESTful な Web サービスが分岐して，JAX-RS というプロジェクトが立ち上がっています．JSR-311[1] のプロジェクトです．すでに jersey[2] という名の参照実装がオープンな形で提供されています．

REST では，すべての「リソース」が URI で一意に指定され，リソースはアプリケーション上で「表現」を持ちます（Web ブラウザ上での HTML のレンダリングをイメージしてください）．また，すべてのリソースに対して共通の少数の操作が定義されていて（HTTP の PUT，GET，POST，DELETE です），表現はリソースへの URI を含むことができます（HTML なら HyperLink です）．こうした REST の考え方は，単純ですが強力なものです．

ただし，これまでの REST は，Web サービスと比べたときに，WSDL のようなサービスの記述言語を持ちませんでした．この点では，新しい JAX-RS は，WADL（Web Application Description Language）というサービス記述言語を持っていることは注目に値します．すでに，Apache AXIS[3] での WSDL2Java/Java2WSDL と同じように，WADL2Java[4] というツールが作成されています．サービス記述言語 WADL とこうしたツールの提供は，RESTful な Web サービスの普及を加速させるでしょう．また，JAX-RS が，主要なリソースとして，インターネット上のコンテンツを格納する XML の統一的なフォーマットである ATOM を想定していることも重要です．

グリッドとWebサービスの動向

ここで，Web サービスをベースとしたグリッド技術のこの 1 年の動向についても触れておきましょう．インターネットの世界は（1 年間で 7 年分の出来事が起きる）ドッグイヤーだとよくいいますが，グリッド，Web サービスの世界も例外ではなく，この 1 年でいろいろなことが起きました．

なかでも最も驚かされたのは，WS-RF の有力なサポータであった IBM が WS-RF へのサポートを取り止め，

WS-RT をサポートすると表明したことでした．これは，WS-RF が，Web サービス業界の有力なプレイヤの 1 つである Microsoft からのサポートを受けられなかったためであるとされています．元々 Web サービス規格は相互運用のための規格ですから，業界のすべての有力なプレイヤから受け入れられていなければ意味がありません．WS-RF はこの条件を満たすことができなかったため，IBM，Microsoft，HP の 3 社で新たに WS-RT を定義し，これをスタンダードとする動きがあるのです．

これを受けて，グリッド標準化団体である OGF（Open Grid Forum）の旗艦アーキテクチャである OGSA（Open Grid Service Architecture）でも，下位レイヤとしての WS-RF を捨て，WS-RT へ移行することを検討しています．OGSA が下位レイヤの乗り換えを余儀なくされるのはこれが初めてではありません．OGSA は，元々 OGSI（Open Grid Service Infrastructure）を下位レイヤとして想定していたのですが，OGSI がやはり業界にスタンダードとして受け入れられなかったため，2004 年 1 月にあらたに WS-RF を定義し，そちらに乗り換えたという経緯があるのです．幸か不幸か，このように一度下位レイヤの乗り換えを経験しているため，OGSA は下位レイヤから分離された抽象的な形で定義され，各プロファイルで下位レイヤへのバインディングを行うことになっているため，この件によるコア部分への影響はないようです．しかし，ようやくいくつかのコンポーネントの定義がそろい，これから本格的に普及を図ろうという時期に下位レイヤの乗り換えを行うことは，残念ながら OGSA の普及の大きな妨げになってしまうでしょう．

Grid のミドルウェアとして広く用いられている Globus Toolkit も現行のバージョン 4 では，WS-RF を用いています．Globus Toolkit を管理している Globus Alliance も，WS-RT の定義が固まった時点で，WS-RT への対応（実質的には実装しなおし）を検討しているようです．

筆者は個人的に今回の変更を歓迎できません．というのは，WS-RT への移行が技術的な判断に基づいたものではないからです．OGSI から WS-RF への移行は技術的に考えても妥当性がありました．OGSI は分散オブジェクトという概念を，半ば無理矢理 Web サービスにマップしたもので，サービス自体に状態を持たせてしまうため，他の Web サービスとの親和性に問題がありました．WS-RF はこの問題点を解決するために，「状態」をリソースという形で分離することで，この問題を解決したものでした．一方，WS-RT と WS-RF の差はどちらかというとマイナーで，サービスを作る上で，本質的なものではありません．また，WS-RT はリソースへのアクセス方法を規定するだけのもので，WS-RF の直接的な代替にはなりません．WS-RT の周辺になんらかのインタフェースを追加しなければなりません．さらに，WS-RT はリソースに対して汎用のアクセスオペレーションを定義しますが，これは，Web サービスの持つ美点の 1 つであるインタフェース定義の明解さを損なうともいえます．

さらに，SOAP をベースとした狭義の Web サービスをベースとしたグリッドのあり方に対する疑念も生じつつあります．5 月にリオデジャネイロで行われた CCGrid2007[5] で，インディアナ大学の Geoffrey Fox 教授が，「Grids Challenged by Web 2.0 and Multicore Sandwich」と題して面白いキーノートを行っていました[☆1]．Fox 教授は，グリッドの標準化団体である OGF/GGF でも活発に活動してきたグリッド研究者です．

Fox 教授は，Web サービスをベースにした「狭義のグリッド」が，Web 2.0 とマルチコアという 2 つのテクノロジの狭間で埋没しかけているのではないかと指摘しています．ここでいう Web 2.0 は，Web サービスにとらわれず独自の REST 的な API を用いたサービス群のことです．例として，Google Map や，Amazon の計算・ストレージサービスを示しています．狭義のグリッドの目指した，複数のサービスをユーザが自由に組み合わせて利用するという世界は，Web 2.0 というまったく異なる，はるかに（ある意味で）安直なテクノロジによって，実現されてしまいました．もちろん，グリッドが目指したのはより B2B の世界であり，現在 Web2.0 が主に利用されているのは B2C ですので，これをもって，狭義のグリッドが無用になったわけではないのですが，がっかりさせられる事実ではあります．

一方でマルチコアが興隆しつつあります．現在は 1 ノード 8 コアぐらいですが，Cell Broadband Engine のようなヘテロ構成をとれば，もっとたくさんのコアを押し込むこともできますし，ムーアの法則によれば，1.5 年で倍のコアを入れられるようになるはずなので，数年後には 1 つのチップ上に 100 近い数のコアが入る，というのは妥当なところでしょう．このマルチコアを使いこなすには，並列分散計算の技術が必要になりますが，Fox 教授は，そこで使われるであろう技術は，グリッド由来のものでなく，もっと昔の並列計算由来のものになるであろう，と指摘しています．

要するに，Fox 教授の指摘は，分散サービス技術とし

☆1 筆者は参加できなかったのですが，スライドの pdf ファイルが下記 Web サイトからダウンロード可能です．http://ccgrid07.lncc.br/keynotes.php

ての地位を Web2.0 に追われ，並列技術としての意味をマルチコアに奪われることで，いまやっている Web サービスを基盤としたグリッド技術は，行き場がなくなるのではないか，ということです．

実際，特定のアプリケーションを指向したグリッドでは，Web サービス技術を用いない標準も提案されつつあります．OGC（Open Geospatial Consortium）[6] は，地理情報の共有と処理に関する標準規格を定めようとする団体ですが，ここで定義されているサービスは，SOAP ベースの狭義の Web サービスではなく，REST 的なインタフェースです．SOAP をベースとした技術によるオーバヘッドと，記述時の煩雑さを嫌ってのことでしょう．

このように，Web サービスをベースとしたグリッド技術の将来は，現在のところ順風満帆というわけにはいっていません．しかし，Web サービスのオーバヘッドは計算機とネットワークの性能向上である程度は解決すると思われますし，記述の煩雑さも補助ツール群の発展によって自然に解消されていくものと思われます．Web サービスの持つ，明示的なインタフェース定義というメリットは，B2B アプリケーションにとって本質的なものです．WS-RT で基盤技術が安定し，上部構造である OGSA の策定が進めば，徐々に浸透していくことが期待できるのではないでしょうか．

連載を終えるにあたって

最初に述べたように，グリッドでは，基本の技術はこの 1 年の間に，WS-RF から WS-RT に変わりました．WS-RT は，リソースの指定に WS- アドレッシングを用いる RESTful な Web サービスだと考えることができます．WS-* の標準技術に REST が影響を与えていることは，SOA 技術の現在を考える上で興味深いことです．

WS-RF は，OGSI がグリッドと Web サービスを乖離させると批判したのですが，残念ながら，エンタープライズ側での Web サービスは，グリッドとの統合の道を進むというより，グリッドとは独自の，「サービスの統合」の道を歩んでいるように見えます．SOA は，エンタープライズ側では，初期の Web サービスによるサービスの実現の段階から，Web サービスの統合技術としての BPEL を経て，多様なサービスの実装を認めてその統合をはかる SCA に進んでいます．

基本的には，サービス概念の拡大とサービスの統合への志向が，エンタープライズ側の SOA のドライビング・フォースになっています．この点では，人間の働きもサービスを提供していると考えて，人間の提供するサービスとコンピュータ・システムの提供するサービスを統合しようという BPEL4People の動きに筆者は注目しています．こうしたアプローチは，SOA 的な上流工程の分析手法に影響を与えてゆくでしょう．一般的には，実装技術の変化は，上流の方法論にも影響を与えることがあると筆者は考えています．

グリッドと共通部分の多い広域分散処理技術の近年の展開について連載では，ほとんど取り上げることができませんでした．ここでは Google が，技術的にも理論的にも，めざましい活動をしています．Google File System や Big Table，MapReduce といった，Google の技術は，高速のネットワーク技術とハードウェアのコストパフォーマンスの劇的な向上の産物です．

Globus 流のグリッドは広域のネットワークにそれぞれ異なる管理主体を想定し，Google のインフラは単一の管理主体を必要とするという大きな違いがあります．そうした違いにもかかわらず，両者は，ユーザに対してサービスの実装の細部を見せずに，Virtual Organization あるいは Cloud を通じてユーザにサービスを提供するという点で，深い共通性があります．

連載は，ここでひとまず終わりますが，SOA は，引き続きホットな話題を我々に提供してくれるだろうと，筆者は考えています．

参考文献

1）JSR-311 : http://jcp.org/en/jsr/detail?id=311
2）Jersey : https://jersey.dev.java.net/
3）AXIS : http://ws.apache.org/axis/
4）WADL2Java : https://wadl.dev.java.net/
5）CCGrid Web Site : http://ccgrid07.lncc.br/
6）Open Geospatial Consortium : http://www.opengeospatial.org/

（平成 19 年 7 月 26 日受付）

丸山不二夫（正会員） maruyama@wakhok.ac.jp

東大教育学部卒業．一橋大学大学院社会学研究科博士課程修了．「最北端・最先端」をモットーに，稚内で新しいスタイルとコンテンツの情報教育を展開．「新しい時代の新しい大学」を目指して，社会人 IT 技術者をターゲットとしたサテライト校を秋葉原に設置．アジアでの IT 教育も熱心に展開している．現在，稚内北星学園大学学長．

中田　秀基（正会員） hide-nakada@aist.go.jp

昭和 42 年生．平成 2 年東京大学工学部精密機械工学科卒業．平成 7 年同大学院工学系研究科情報工学専攻博士課程修了．同年電子技術総合研究所研究官．平成 13 年（独）産業技術総合研究所に改組．現在，同所主任研究官．平成 13 ～ 17 年まで，東京工業大学 学術国際情報センター研究・教育基盤部門客員助教授を兼務．博士（工学）．並列プログラミング言語，オブジェクト指向言語，グローバルコンピューティングに関する研究に従事．