連載　Web サービス
第 6 回

ビジネス環境と実装システムを繋ぐ BPM と SOA

日本ユニシス（株）ビジネスイノベーション本部
牧野 友紀　*tomonori.makino@unisys.co.jp*

編集：XML コンソーシアム

はじめに

目まぐるしく変化するビジネス環境に対してシステムの変更を俊敏に行えないことが，企業の大きな経営課題となっていることは改めて言うまでもありません．この課題に対して，BPM（ビジネスプロセス・マネージメント），SOA（サービス指向アーキテクチャ）という 2 つの抽象的な概念と Web サービスの技術を組み合わせ，ビジネス環境とシステム実装を緊密に連動させることで解決しようと考えられています．ビジネス環境の変化に対して，柔軟に対応するシステム環境を BPM と SOA がどのように実現するのか考えることが今回のテーマです．

BPM と SOA による柔軟なシステム環境とは

システム環境の変更が追いつかないビジネス環境の変化とは，企業合併，企業間の業務提携，事業の再編など業務をまたがる大きな変化です．これらの環境変化に対しては，業務内，業務間，事業間，企業間の業務の流れを変更します．これに対して，現状の企業内の多くのシステムは，特定の業務や事業をシステム化の対象範囲とした“一枚岩の閉じた構造”になっており，他システムとの相互運用性や，機能やデータの共有といったことにあまり考慮がされていません．システム内部のプログラムやコンポーネントは全体的に密に結合され，システム間連携は，内部データ形式そのままのファイルが転送されたり，個別に作成した通信プロトコルによりデータの送受信が行われたりしています．このため，システム上の処理フローが固定化し結果的に業務が硬直化してしまいます．このようなシステムの集合体では，ビジネス環境が変わるたびにシステムの再編成とシステム間の再統合が必要となり，環境変化への対応に膨大な時間がかかっています．

BPM と SOA の考え方を導入した柔軟なシステム環境では，各システムが内包する機能を業務の視点でモジュール化して切り出し，システムの境界や実装方式を意識せずに利用できるようにします．このモジュールを SOA ではサービスと呼びます．業務の変更に伴い，既存システムを極力修正せずに，切り出したサービスを組み替えることで，間接的にシステム全体の処理フローを変更できるようにします．これは，Web による抽象化の概念と同じです．Web により既存システムのユーザ・インタフェースが抽象化され，利用者の利便性に応じて異なる複数のシステムのユーザ・インタフェースが間接的に統合されるようになりました．

BPM と SOA による新たなシステム環境の柔軟性は，利便性の高いサービスをいかに品揃えできるかにかかっています．サービスの利便性を高めるためには，業務とサービスの明確な関連付けとサービスの粒度を適切にすることが重要となります．図 **-1** のように，BPM は関連付ける業務を抽象的なモデルに表し，SOA は適切な粒度の機能モジュール（サービス）を作るために既存システムの機能を抽象化します．

業務の抽象化

業務の抽象化とは，広範にわたり細かな作業が途切れなく続く現実の業務を把握するために，単純で構造化されたモデルを作成し実態ビジネスを写像することです．このモデルが，業務の視点でシステムの機能をモジュール化する基準となり，全体的な業務の中でシステムと関連付ける位置を明確にする地図の役割を果たします．

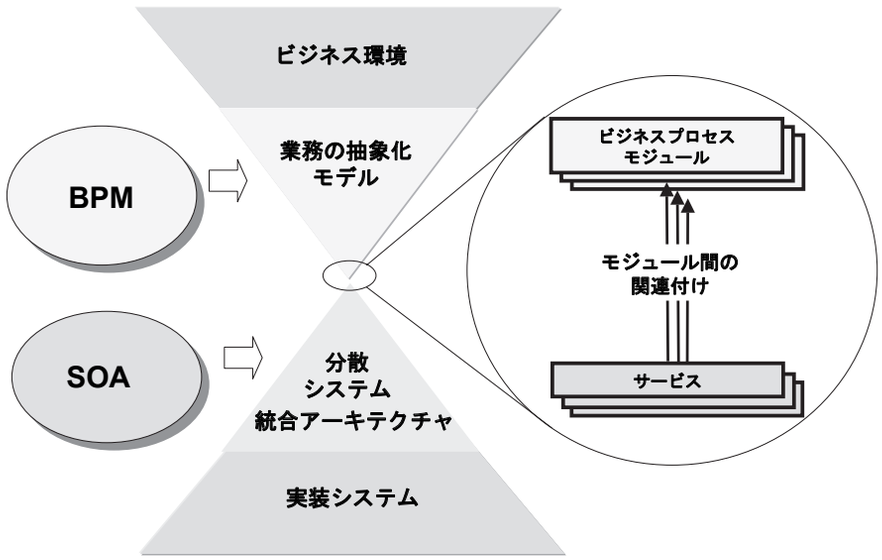

図 -1　BPM と SOA によるビジネス環境と実装システムの関連付け

BPM のモデル作成で中心的な作業は，ビジネスプロセス・フロー分析です．ビジネスプロセス・フロー分析では，事業レベルから担当者が行う作業レベルまで，階層的に業務の流れを単純化して表現します．同一階層では，相互関連性の高い業務処理をモジュール化し，モジュール間の関連付けを明確にします．階層化を進める中で，それ以上に分割すると業務的に独立性のない粒度のビジネスプロセス・モジュール（以後，文中でリーフ・プロセスと呼びます）が出現し階層化が終了します．このリーフ・プロセスがビジネスプロセス・フローを変更する最小単位になります．

モデル上のリーフ・プロセスに，モジュール化したシステムの機能を対応付けることで，ビジネスプロセス・フローに沿ったシステムの構成を，機能単位で考えることができるようになります．ビジネスプロセス・フローの変更は，ビジネスプロセス・モジュールを組み替えで理解することになり，システム環境の影響度合いや変更方法を容易に把握することが可能になります．

ビジネス環境の変化に強い柔軟なシステム環境を実現するためには，リーフ・プロセスのモジュール化において粒度と範囲が適正であることが重要です．モジュール分割する境界が不適切である場合やモジュール化する粒度が大きすぎる場合，ビジネスプロセス・フローの変更においてモデル上リーフ・プロセスの分割やリーフ・プロセスの再編成が発生します．リーフ・プロセスの変更は，対応するシステム機能の修正に繋がり，大幅なシステムの改修が必要となります．リーフ・プロセスのモジュール化は，業務に精通した人とシステムのアーキテクトが議論を重ね，モデリングすることが重要となります．

実装システムの抽象化

リーフ・プロセスへの分割が終わると，次はこれらのリーフ・プロセスを，情報システムで実装しなければなりません．リーフ・プロセスの実装は，さらに細かいシステムのコンポーネントから構成されます．実装システムの抽象化とは，システムの実装方式の差異を隠蔽し，システムを構成する細かいコンポーネントやプログラムを適切な粒度のモジュールにまとめることです．このモジュールがサービスです．現状，SOA を導入する場合，サービスのインタフェースとして Web サービスを用いることで，実装方式の差異を隠蔽します．

サービス化を考える上で最も重要なことは，サービスの粒度と業務との関連付けです．サービスの粒度は，ビジネス環境の変化に応じて適切に組み替えや追加が可能な単位に小さくし，業務視点でサービスが把握可能な単位に大きくする必要があります．また，サービスはさまざまなリーフ・プロセスや上位のプロセスで使われている可能性があるので，どこでサービスが使われているか明確になっている必要があります．

サービス化の考え方として 2 つの考え方があります．1 つは，ビジネスプロセス・モデルのリーフ・プロセスごとにサービスを対応付け，必要となる機能を集約する考え方です．たとえば受注登録プロセスにおける受注登録プロセス支援サービス．この種のサービスは，多くの場合，さらに細かいサービスの組合せ，すなわちコンポジット・サービスになります．もう1つは，1つまたは複数のコンポジット・サービスから呼び出される個別のサービスで，多くの場合業務管理対象ごとに関連する機能を集約する考え方になります．

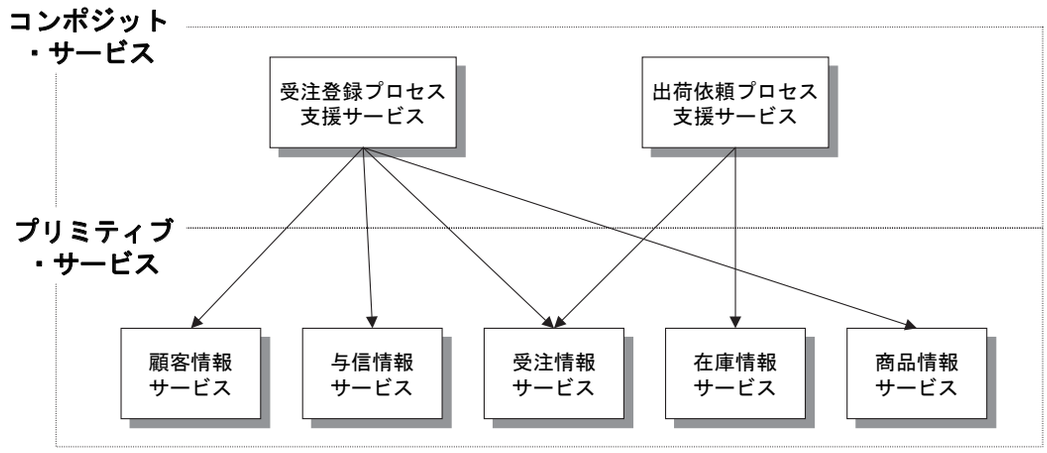

図 -2　プリミティブ・サービスとコンポジット・サービスの関係

たとえば顧客情報という管理対象に関する機能を集約した顧客情報サービスです．これをプリミティブ・サービスと呼びます．プリミティブ・サービスとコンポジット・サービスは図 -2 のように階層的に位置付けられます．

プリミティブ・サービスは，従来単一システムの中で利用していた機能を，ソフトウェア・コンポーネントとして切り出し公開することで，分散システム環境における開発生産性と品質を高めることを目的としています．プリミティブ・サービスの抽象化は，内部的に断片化したデータを，業務的に意味をなす単位のデータに併合し，トランザクション処理や DB アクセス等の技術的な制御処理を隠蔽し，複数の詳細な業務処理を集約します．プリミティブ・サービスは，製品情報の提供，在庫情報の提供など，正確な情報をリアルタイムに提供する機能や，注文書の受付処理など業務トランザクション処理のインタフェース機能として，他システムに提供されます．

コンポジット・サービスは，リーフ・プロセス単位に，システムの境界を越え複数のシステムが提供する機能を集約したアプリケーションを作成することを目的にしています．また，既存システムから独立した環境でプリミティブ・サービスを業務手順や業務規約に従い呼び出し，複数のシステムを統合的に操作する操縦室のような役割を提供します．定型的な業務に特化し，プリミティブ・サービスの汎用性を隠蔽し機能を単純化します．業務手順の変更や情報の追加などリーフ・プロセス内の変更に対するシステムの修正や，ビジネスプロセスの処理フローの変更など，多くの場合，特定コンポジット・サービスの修正やコンポジット・サービス間の関連付けの変更などで対応が可能で，既存システムを変更する回数が少なくなります．

SOA のアーキテクチャ構造

抽象化を実現する SOA のアーキテクチャ構造は，図 -3 のように，プレゼンテーション層―プロセス層―サービス層―ビジネスロジック層―データアクセス層の 5 層構造で表現することができます．この 5 層構造は，いくつかの大手ソフトウェア・ベンダやシステム・インテグレータの SOA の説明で類似した層構造で表現されるようになり一般的になりつつあります．従来の一般的な 3 層構造（プレゼンテーション層―ビジネスロジック層―データアクセス層）と比べると，ビジネスロジック層とプレゼンテーション層の間にサービス層とプロセス層が新たに追加されています．サービス層とプロセス層により，既存システムと SOA ベースの新規システムが混在する環境で，システムの境界を意識しない抽象化されたサービスを結合したり複合したりすることが可能になります．

サービス層は，ビジネスロジック層以下で実装される基幹の業務処理機能をモジュール化し抽象化します．プリミティブ・サービスはサービス層のモジュールとして位置付けられます．プリミティブ・サービスのインタフェースは Web サービスで実装することが一般的になります．Web サービスで実装することで，部門内，部門間，企業間で同じ方法でアクセス可能な位置透過のモジュールとして準備することが可能になります．

プロセス層は，プリミティブ・サービスを集約し，ビジネスプロセスで必要となる機能をモジュール化して，プレゼンテーション層に機能を提供します．また，プロセス層のモジュール間を関連付けることで，ビジネスプロセス・フローに即した処理連鎖を実現します．

コンポジット・サービスはプロセス層のモジュールとして位置付けられます．コンポジット・サービスでは，リーフ・プロセス内で利用される統合的な機能を実装します．また，あ

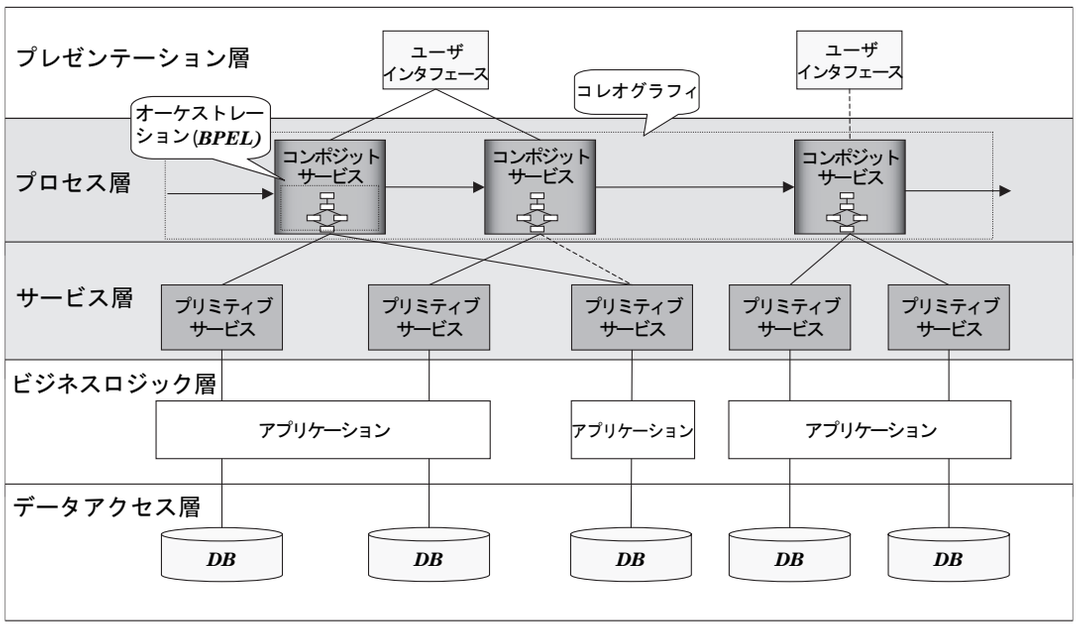

図-3 SOAの5層構造

わせてリーフ・プロセス内の業務手順を制御するワークフローを実装します．ワークフローは，注文伝票の起票，照査のように人的な業務手順の制御や，プリミティブ・サービスを複合する自動的な処理などから構成されます．コンポジット・サービス内のワークフローを柔軟に構成するためには，BPEL（Business Process Execution Language）などで提供されるプロセスのオーケストレーション機能が適切です．

プロセス層のコンポジット・サービス間をWebサービスにより疎結合で結合することで，柔軟にビジネスプロセス・フローの変更に対応することが可能となります．コンポジット・サービスを柔軟に統合する基盤には，すべての動作を管理するオーケストレーションの概念よりも，イベント駆動型のコレオグラフィの概念の方がより適しています．

5層構造の段階的普及

SOAの5層構造は，段階的にシステム環境に浸透していくでしょう．その浸透過程を予想してみましょう．最初の段階では，システム統合などを機に，既存システムの機能をプリミティブ・サービスとして公開することが進みます．利用するシステムでは，ビジネスロジック層でプリミティブ・サービスを呼び出すことが多いでしょう．第2段階は，プリミティブ・サービスの数が増え，Webサービスが社内のシステム開発で根付いた段階です．新規システムの開発では，5層構造のシステムが増え，コンポジット・サービスの実装が行われるようになります．第3段階は，レガシー・システムが段階的に刷新され5層構造のシステム開発が普及した段階です．プリミティブ・サービスはワークフロー・エンジンで統合されることが一般的になり，コンポジット・サービス間の連携もイベント駆動型の統合エンジンで行われるようになっているでしょう．

おわりに

BPMとSOAは，ビジネス環境の変化に伴い変更されるビジネスプロセス・フローに合わせ，サービスを組み替えることで，システムを柔軟に再編成できる分散システム統合環境を実現するための概念です．BPMの普及とWebサービスの普及で，業務の抽象化と，実装システムの抽象化が容易に行えるようになりました．ビジネスプロセス・モデルにサービスを対応付けることで，ビジネス環境の変化を分散システム統合環境で間接的に吸収し，実装システム環境の変更が必要とならないようにすることが，柔軟なシステム環境の本質と考えます．

参考文献
1) The SOA-BPMS Bridge To Agility, http://www.ebizq.net/topics/soa/features/3646.html?related
2) SAP エンタープライズサービスアーキテクチャ Dan Woods, オライリー・ジャパン．
3) SAP コンポジットアプリケーション Dan Woods, オライリー・ジャパン．
4) Application Architecture for .NET: Designing Applications and Services, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp
5) Application Conceptual View, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnea/html/eaappconland.asp
6) Service-oriented Modeling and Architecture, http://www-106.ibm.com/developerworks/library/ws-soa-design1/

（平成16年12月6日受付）