金融機関統合におけるSOAアプローチの実践

加藤 洋　三谷 隆　杉山 真一

Implementing the SOA Approach in Bank Mergers

Hiroshi Katoh　Takashi Mitani　Shinichi Sugiyama

昨今, 金融業界において企業統合や再編の動きが活発になってきている. これまでの銀行統合では, 両行の勘定系システムをリレーコンピュータで接続する方式が一般的であった. しかし, 修正が広範囲にわたること, また発展的な利用形態への拡張には向かないことなど課題があった. これら課題の多くを解決する方法として, SOA（Service Oriented Architecture）アプローチを実践し, 有効性が確認できた. 本論文では, 銀行統合での実践を題材に, SOAにおけるコンポーネント定義のアプローチと, 実装テクノロジー選択のアプローチという2つについて, 実プロジェクトでの実践結果を含め述べる. このアプローチ手法については, 金融のみならず, グループ企業間の接続や他業態での統合などにも十分活用可能であると考える.

Recently, company mergers and industry realignments have blossomed in the finance industry. Previously in bank mergers, it was usual for system integration to be implemented by connecting the core banking systems of both banks via relay computers. However there were several issues with this approach. Modifications were far-ranging, and the approach was not suited for expanding to more constructive configurations. As a way of resolving many of these issues, we implemented a Service-Oriented Architecture (SOA) approach and verified its effectiveness.

In this paper, using the example of bank mergers, we discuss two approaches, namely, the SOA component definition approach and the implemented technology selection approach. Our discussion also covers the results from implementing the approaches in real projects. Beyond just the finance industry, these approaches could also be fully utilized in the integration of corporate group companies and mergers in other industries.

Key Words & Phrases：銀行統合, SOA, ESB, EA, リファレンス・アーキテクチャ
bank merger, SOA, ESB, EA, reference architecture

1. はじめに

昨今, 金融業界では企業統合や再編の動きが活発になってきている[1][2]. 企業が統合する場合に最も重要視されるのが, システム統合と言える. 特に銀行業界においてメガバンク同士のシステム統合での障害発生は, 社会問題として扱われるほど影響が大きいものである.

銀行システムは, ここ10年くらいの間に急激に複雑化してきている. 営業店テラー端末, ATM（現金自動預け払い機）だけでなく, インターネット, コンビニなどの複数チャネルに対して, さまざまなサービスを提供している. 銀行システムの巨大化に伴い, 銀行統合時には2つの銀行のシステムを並存させたまま合併日を迎えるのが主流である.

銀行におけるサービスは, さまざまなバックエンドシステムに実装されたサービスを, 営業店やインターネットなど複数のチャネルにより提供するといった形態である. 銀行統合においては, 2つの銀行の提供するサービスを組み合わせて, 両行のチャネルで提供するといったニーズがITへの要件となる. また, 発表から新銀行発足までの時間制約があるという経営からくる絶対条件や, 統合による障害の発生は大きなレピュテーションリスクであるといった点も重要な要件となる. これは, ソフトウェア再利用と柔軟なアプリケーション統合を目指すSOA（Service Oriented Architecture）の狙いと一致している. さらに, 時間的制約, 既存資産の維持や品質維持といったより厳しい条件が加えられていると言える.

これまでの銀行統合では, **図1**のように両行の勘定

提出日：2005年8月31日　　再提出日：2006年8月30日

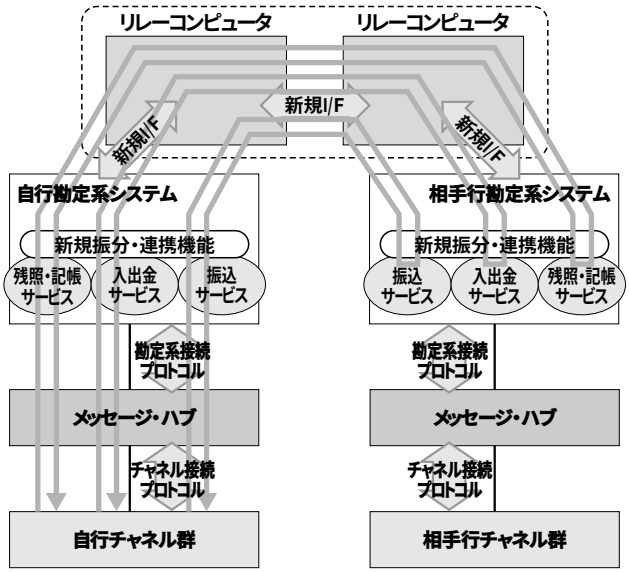

図1. リレーコンピュータ接続

系システムをリレーコンピュータで接続する方式が一般的であった. この方式では, リレーコンピュータの新規構築だけでなく, 両行既存プログラムへ銀行判別・振分ロジックなどの追加が必要であり, 修正が広範囲にわたる.

また今後のさらなるM&A（Mergers and Acquisitions）による統合やグループ企業間での接続といった発展的な利用形態への拡張はさらなる既存プログラムへの分岐ロジック追加となり, 修正のたびにリスクを伴うこととなる. そこで筆者は, 実際の銀行統合においてSOAアプローチを実践し, 結果として統合における多くの課題を解決することができることを実証した.

一方で, SOAについては総論については誰もが同意できる内容であるが, 利用技術や製品といった各論レベルとなると現行システムとの乖離（かいり）やトランジションプラン立案の困難さ, 投資に見合うメリットが得られるかなど課題が多い. 実際に統合の対象システムは最近構築されたものばかりではなく, レガシーや他社プラットフォームなど既存資産が莫大（ばくだい）にある. そこで, 現実的な実装を目指し, 既存システムへの修正範囲を抑えながら段階的に発展可能であるバスをデザインした. このようなデザインを行うことで, 異なるシステム間を連結し, 既存資産の有効活用や異機種間のデータ連携を実現するEAI（Enterprise Application Integration）をサービス・バスに発展させる方法を実践した.

本論文では, 銀行統合でのSOA構築の実践を題材に, SOAにおけるサービスを分割し, サービス・コンポーネント化する「コンポーネント定義」のアプローチと, 既存資産を考慮しSOAを実現する「実装テクノロジー選択」のアプローチという2つのSOAアプローチについて, 実プロジェクトでの実践結果を含め述べる. 本論文でのサービスとは, 両行チャネルに提供する入金, 出金, 振込のようなサービスを, サービス・コンポーネントとは, 適当な粒度に分割した結果である振込入金処理と振込出金処理のようなコンポーネントを示す. このアプローチ手法については, 金融のみならず, グループ企業間の接続や他業態での統合などにも十分活用可能であると考える.

2. 銀行システム統合における要件と課題

統合する両行のバックエンドシステムは, 各々別の基盤（ハードウェア, OS, ミドルウェア）, フレームワークを使用していることがほとんどで, また各チャネルとのインターフェース（使用文字コード, プロトコル, 取引電文フォーマット, 障害時リカバリー機能）についても両行間で異なった実装になっているケースがほとんどである. 銀行間の接続については, 統合ATMセンターや, 全銀センターなどの外部機関センター経由で接続されており, 他行のキャッシュカード取引のサービスについては, この外部機関センター経由のネットワークで実現されている. このような環境を踏まえて, 銀行統合における要件と課題を述べる.

2.1 統合における要件

銀行統合においては, 2つの銀行の提供するサービスを外部機関センター経由の接続ではなく, 両行間を個別に接続し, 組み合わせて, 両行のチャネルで提供するということが, ITへの最大の要件となる. 銀行サービスの統合のメインである銀行勘定系システム統合における重要な要件として以下があげられる.

- 外部機関のセンターを経由せずに, 2つの勘定系システムを接続し, お客様へサービスを提供できること
- 両行の各チャネルから2つの勘定系サービスを極力意識することなく利用できること
- 各行内で完結する既存サービスに極力影響を与えないこと
- 短期間でサービスインできること
- システムだけでなく事務面も含めたEND to ENDでのリカバリーができること
- エンタープライズ・アーキテクチャ［8］に準拠し, 新銀行に向けた全体最適が考慮されていること

2.2 統合における課題

これまでの銀行統合では, 両行の勘定系システムをリレーコンピュータで接続する方式が一般的であった. リレーコンピュータ方式では, 複数のチャネルからのサービス・リクエストを全て各々の勘定系システ

ムで受け付けて, 勘定系システムの中で振り分け処理を行う. 振り分けられたリクエストは, 両行勘定系システムを接続するリレーコンピュータ経由で送信され, 相手側の該当するサービスを提供する仕組みになる. しかしこの方式で統合サービスを実現することには, いくつかの課題がある.

- リレーコンピュータを新規構築するだけではなく, 両行既存プログラムへ銀行判別・振分ロジックなどの追加が必要であり, 修正が広範囲にわたるため, 時間的制約がある中で品質の確保が困難である
- リレーコンピュータに何らかの障害が発生した場合, 両行の既存サービスに影響がおよぶ可能性が高い
- また今後の更なるM&Aによる統合やグループ企業間での接続といった発展的な利用形態への拡張はさらなる既存プログラムへの分岐ロジック追加となり, 修正のたびにリスクを伴うこととなる

上記のような銀行システム統合の要件および, リレーコンピュータ方式における課題を考慮しデザインする必要がある.

3. 統合におけるSOAアプローチの考え方

SOAへの取り組み方については, 主にコンサルタントが行う企業戦略策定ステップ, ITアーキテクト主導で行うデザインステップ, ITスペシャリスト主体で行う実装ステップのそれぞれにより目的やアプローチ方法が異なる[4][5][6]. 統合においては, すでに存在する既存システムを組み合わせてサービスすることが要件であることから, デザインステップにおけるSOAアプローチの活用について述べる.

3.1 統合へのメソドロジー適用

SOAとは, システムの機能単位を「サービス」として定義し, 複数のサービスの組み合わせでシステムを構築しようという思想であり, IBMはサービス・モデリングのメソドロジーとしてSOMA（Service-Oriented & Modeling Architecture）を確立している. SOMAはすで に確立されていたソフトウェア・アーキテクチャの基本原則に基づき, サービスを統合してプロセスとして組み立てるサービスコレオグラフィーなどの視点を追加したものであり, SOA実現のための有効なメソドロジーである[7]. SOMAでは,「サービス仕様定義」,「サブシステム分析」,「コンポーネント使用定義」,「サービス実装の決断」の4つのステップから構成されているが, ここでは, 統合において提供するサービス範囲がユーザー要件により特定されること, またサービスを実装するコンテナーについては既存サブシステム主体であることより,「サービス仕様定義」,「サブシステム分析」については, 適用を見合わせた. よって「コンポーネント定義」と「サービス実装の決断」の2つのステップを適用することとした.

3.2 コンポーネント定義のアプローチ手法

当アプローチには, SOMAにおける「コンポーネント定義」のステップを利用した. 実際にサービス・コンポーネントを定義する手順について, ATMによる振込を例にとって記述する. 振込は一方の口座から出金し, もう一方の口座へ入金するというサービスであり, 口座から出金するというサービス・コンポーネントと, 口座に入金するというサービス・コンポーネントの組み合わせによるサービスととらえられる. さらに取引フローを詳細化すると口座確認や手数料計算といった, さらに細かいコンポーネントに分割される. ここで詳細なコンポーネントを定義する際には, オンライン取引で同期処理が必要なものと, 非同期で後処理として確実に実施されればよいものといった非機能要件面での分類や, 2つのシステム間で振込が発生した場合に, どちらのシステムで実施されるべきコンポーネントなのかを意識し, 適切なレベルまで分割する.

3.3 実装テクノロジー選択のアプローチ手法

当アプローチは, SOMAにおける「サービス実装の決断」のステップを取り入れた. 手順としては, コンポーネントが分割された段階で, 取引種類ごとにサービスとしてのフローを作成する. 振込の例では, 銀行1から銀行2へ振込する場合, その逆や他行宛の振込といったバリエーションごとに既存サブシステム上に配置したコンポーネントをフローでつなぎ, 最終的なサービスの形態と実現性をチェックする. フローが完成した時点で, フローのパターン化を行う. この際に取引量などの非機能要件からもフローをアセスし, 分類を行う必要がある. パターン分類後, 各パターンごとに必要な機能を洗い出し, 既存機能との対比により実装テクノロジーの選定を行う.

4. SOAアプローチの実践

3章で述べたアプローチを実践するにあたり,「コンポーネント定義のアプローチ」についてはサービス・コンポーネント化の1ステップで,「実装テクノロジー選択のアプローチ」についてはフローパターンの分類と実装テクノロジーの選択といった2つのステップに分けて実施した. 以下にこれら3ステップの作業について具体的に述べる.

4.1 サービス・コンポーネント化

統合において提供するサービスを, 最適な粒度の処理単位に分割し, コンポーネントとして定義した. こ

の作業をサービス・コンポーネント化と呼ぶこととする. 現状は, 全ての自行内サービスを同期処理で行っているが, 実際該当のサービスを提供する中で同期処理が必須なのか, 非同期で確実に実施されればよいものなのか分類した. また現状, 各チャネルからの全てのサービス・リクエストは, 必ず自行勘定系システムにあがるようデザインされているが, 統合において相手行のサービスを提供するという観点で, 必須なのかどうかについても分類を行った. 単純に考えると相手行のサービスを提供するのであれば, チャネルからのサービス・リクエストは必ずしも自行勘定系システムにあげる必要ないと考えがちであるが, 実際のチャネル側での事務処理を考慮するとそうとは限らないのである. これらの分類は両行間の連携において, 2章で述べた統合における要件や課題を解消すべく, さらなる疎結合な形式での実装を実現するといった観点より重要な分類となる. 例として**表1**に, 自行営業店ATMから, 自行キャッシュカードを使用したサービス・コンポーネント化の結果を記載する.

表1. サービス・コンポーネント化の結果

サービス	コンポーネント	同期／非同期	サービス・リクエスト送信先システム	備考
入金	入金処理	同期	相手側勘定系	
	会計情報更新処理	非同期		
出金	出金処理	同期	相手側勘定系	
	会計情報更新処理	非同期		
残高照会	残高照会処理	同期	相手側勘定系	
通帳記帳	記帳処理	同期	相手側勘定系	
振込照会	受取人口座照会	同期	自行勘定系	振込エラー時のお客様対応で処理した店舗の情報が必要であるためサービスリクエストは自行勘定系にあげる必要あり
	依頼人口座照会	同期		
	手数料計算	同期		
振込処理	振込出金	同期	自行勘定系	振込エラー時のお客様対応で処理した店舗の情報が必要であるためサービスリクエストは自行勘定系にあげる必要あり
	振込入金	非同期		

4.2 フローパターン分類

4.1のサービス・コンポーネント分割の結果を既存サブシステム上に配置し, サービス・リクエストを送信するチャネルから, コンポーネントへのデータフローをサービスごとに記述(**図2**参照)し, チャネルとコンポーネント, およびコンポーネント間の通信についてパターン分析を実施した. 結果として, 両行勘定系システムを連携するデータフローは, 3つのフローパターンに集約できた.

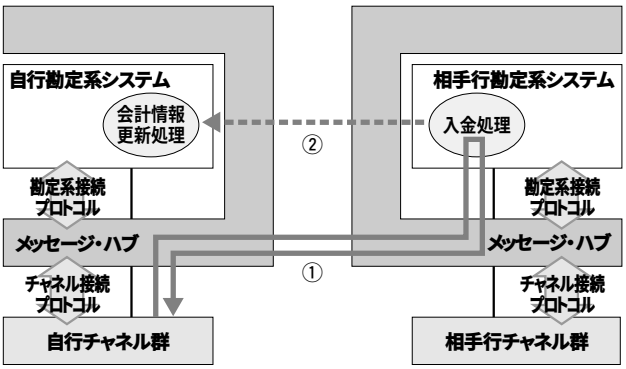

図2. 入金サービス・データフロー

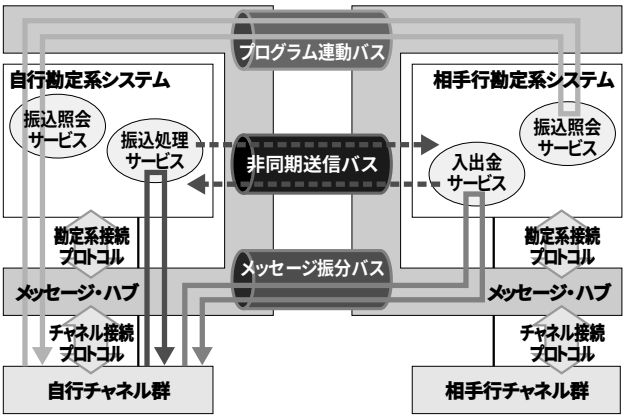

図3. 3種類のサービス・バス

- サービス・リクエストを直接相手勘定系に送信しサービスを提供できるパターン
- チャネルからのサービス・リクエストとは非同期に勘定系システム間で一方送信するパターン
- サービス・リクエストは自行勘定系システムに送信し, 両行のサービス・コンポーネント同士が同期連動しサービスを提供するパターン

この3つのフローパターンを組み合わせることにより, 全てのサービスの提供が可能となる. この3つのフローパターンを実現する機能をメッセージ振分バス, 非同期送信バス, プログラム連動バスという3つのサービス・バスとして定義(**図3**参照)した.

(1) メッセージ振分バス

自行チャネルから, 相手行勘定系システムを照会もしくは更新するフローパターンである. サービス・リクエストは相手行勘定系システムにあげて, 直接相手行のサービスを提供する. これに該当するサービス・コンポーネントは, 残高照会処理, 入金処理, 出金処理, 記帳処理である.

(2) 非同期送信バス

自行勘定系システムから相手行勘定系システム宛に非同期送信するフローパターンである. このフローパ

ターンに該当するサービス・コンポーネントは, 振込入金処理, 会計情報更新処理である. いずれも他のフローパターンとの組み合わせで1つのサービスを提供することとなる.

(3) プログラム連動バス

自行勘定系システムから相手行勘定系システムに同期処理するフローパターンである. チャネルからのサービス・リクエストは自行勘定系システムにあがり, 相手行勘定系システムと同期更新処理が必要なサービスが該当する. またチャネル側での事務処理の関係でサービス・リクエストを必ず自行勘定系システムにあげる必要があるサービスについても対象となる. 振込照会処理が該当する.

4.3 実装テクノロジー選択

既存システムは, 両行とも勘定系システムとチャネル間は, 各々MQ (Message Queuing) ベースのインターフェースでデザインされたメッセージ・ハブで構成されていた. 今回のシステム統合においては, スケジュールがタイトであり, ESB (Enterprise Service Bus) [3] やWebServicesといった新技術の採用や新規インフラを構築することは既存システムへの影響も大きい. 安全に銀行統合を行い, 両行のサービスを提供するという観点で, この既存のメッセージ・ハブ同士を接続し拡張することにより, サービス・バスを実装することが最適な方法であると考えた.

実装にあたっては, 3つのバスそれぞれに必要な機能を, ガートナーにより定義されたバスの3層, 10種類の機能分類 [9] にマップした. この機能分類マップをベースに実装の検討を行った結果が**表2**である. 統合の対象システムは最近構築されたものばかりでなく, レガシーや他社プラットフォームなど, 既存資産が莫大にある. これらの既存資産を柔軟にアプリケーション統合しサービスを提供していく観点で, 既存のメッセージ・ハブが保有する機能との対比と実装方針の検討を機能分類マップにより実施した.

メッセージ振分バスについては, あて先振り分け機能が主要な新規機能であるが, 振り分けには店番や店名など業務データを意識する必要がある. そこで, あて先振り分け機能をチャネル側とハブ側各々で実装した場合の比較をし, チャネル側で業務データによるあて先判別とあて先ヘッダー情報埋め込みを実施し, ハブ側はヘッダー情報をベースに振り分けを行うという機能分担に決定した.

非同期送信バスについては, メッセージ蓄積が主要な新規機能であるが, バスの障害時にも取引継続を可能とすることを考慮すると, バックエンドにも蓄積機能が必要となる. バックエンドとバスの双方に蓄積機能を開発した場合, 業務リカバリーや送達確認など重複した機能開発になることや, バスでの蓄積にはシェアDBが必要となり単純な2系統化ができないといった理由からバックエンドにのみ機能を実装することとした.

プログラム連動バスについてはワークフロー関連が主な新規機能である. 今回, プログラム連動バスを利用する取引は振込のみであり, アプリケーションのコーディネータをバスに実装する場合には, 既存アプリケーションの機能の一部をバスに実装することとなる. また, 障害時の取引リカバリーを考慮すると, 既存業務についても修正だけでなく, 障害時には処理した内容を無効にする逆取引を開発し用意しておく必

表2. 機能分類マップと実装形態

	SOA実装機能	メッセージ振り分けバス	非同期送信バス	プログラム連動バス
フローコントロール	ステータス管理	バックエンドシステム・ステータス管理実施 ○	バックエンドシステム・ステータス管理実施 ○	バックエンドシステム・ステータス管理実施 ○
	プロセスフロー管理	不要 −	不要 −	バックエンドにて実施 ▲
	ワークフローアプリケーション	不要 −	不要 −	バックエンドにて実施 −
データサービス	あて先振り分け	チャネルシステムにて送信先キューを振り分け ▲	不要 −	バックエンドにて実施 ▲
	メッセージディクショナリ	−	−	−
	メッセージ変換	各種ヘッダー変換 ○	各種ヘッダー変換 ○	各種ヘッダー変換 ○
ネットワーク・コネクティビティ	メッセージ送受信	○	○	○
	メッセージ蓄積	不要 −	バックエンドにて実施 ▲	不要 −
	プロトコル変換	○	○	○
	コード変換	◎	◎	◎

○:既存機能拡張　◎:新規追加機能　▲:バス以外で実装機能　　文献[9]の分類に準拠

要があるなど全体として開発量が多くなる. そのため, 今回はサービス・バス上で実装することが必ずしも最適な配置ではないと考え, 既存と同様にバックエンドでの実装とした.

5. 考察

今回, SOAにおける「コンポーネント定義」と「実装テクノロジー選択」という2つのアプローチを適用することで, 銀行システム統合の持つ課題を解決し, 各チャネルとのインターフェースなどシステムが根本的に異なる2つのシステムを接続できた. これにより, お客様に相手行のサービスを提供することができた. また, サービスをコンポーネント化できたことにより今後の再利用性, 拡張性を確保できた.

5.1 銀行統合における課題解決について

SOAアプローチを適用することにより従来のリレーコンピュータの持つ現行システムへの影響の課題を解決し, 各チャネルとバックエンドシステムは相手行を意識することなく接続できた. これは, ハブによる接続を行うことでお互いのシステムの根本的な違いを吸収したからと言える. このことにより既存のチャネル, バックエンドシステムのリカバリー, 事務面への影響も最小限に抑えられた.

5.2 コンポーネント定義について

統合において, 現行モジュールを活用したサービス・コンポーネント化を追及したが, 簡単ではなかった. これは現行モジュールのコンポーネント化に依存することであるが, 銀行の勘定処理モジュールでは資金移動以外に画面や帳票制御, 勘定照合のための該当勘定カウンター更新, 手数料算出, 旧店番・店名の読み替え, 事務量把握のためのロギング, 金融機関コードによる処理の振り分けなど, さまざまな処理が混在しており, 2つの銀行の勘定系を組み合わせてサービスするには, これら処理を一部分離することがコンポーネント化には必要であった.

今回, サービスの特徴より3つのバスを構築することにした. メッセージ振分バスおよび非同期送信バスで提供されるサービスに関しては, コンポーネント化することができた. 例えば, ATMから出金処理を要求した場合, メッセージ振分バスで相手行の勘定系システムの出金処理コンポーネントにて出金をサービスする. そして, 両行の勘定精査を合わせるために非同期送信バスを通して会計情報更新処理コンポーネントにて勘定を合わせる処理をする.

しかし, 全ての処理がコンポーネント化できたわけではない. 為替処理などは単純に入金と出金サービスに分離できず, プログラム連動バスを通して, 相手行の為替処理と密連動することになった. これは, 事務手続きの観点からくる為替処理の仕向け店, 被仕向け店の考慮が必要であったことや, 手数料計算処理と振込処理の分離が困難であったためである.

今後, さらなるサービスのコンポーネント化をしていくためには2つの課題がある. 一つ目は, どのくらいの粒度までコンポーネント化していくのかである. どれくらい再利用されるのか, コンポーネント化にどれくらいワークロードがかかるのかなどを見極めていく必要がある. 二つ目は, 異なるシステム間を渡り歩き, サービスを提供する場合に, アプリケーションのコーディネータ機能をどこに配置すべきかである. 引き続きバックエンドにコーディネータの働きをさせるのか, ハブにコーディネータの働きをさせるのかを検討する必要がある.

ハブにコーディネータ機能を持たせた場合には, 複数のバックエンド間でのコミットタイミングが異なることとなる. 処理途中での障害が発生した場合, コミット完了している処理を取り消さなければならず, そのために逆取引などのリカバリー処理用の機能を開発する必要がある. これらの考慮はサービスごとに必要であり, 開発量が膨らむ可能性もあるので, コンポーネントの再利用頻度(さまざまなサービスで利用されるものか)を加味した検討が必要である.

5.3 実装テクノロジー選択について

実装テクノロジーについては, 統合の課題を解決し, かつ既存システムを有効利用できるテクノロジーを選択した. 両行のハブでのメッセージ連携時の標準として使われ, かつSOA技術の中心でもあるMQベースにした.

今後, 業界で言われているESBやWebServicesといったテクノロジーを実装していくことが望ましいが, 安易に適用することには疑問が残る. 銀行においてSOAへ期待されることは, 新商品・新サービスの提供が主であり, 1つのサービスとしてとらえる単位が振込サービス, 投信購入サービスといったように小さく, オンライン処理でミリ秒単位の粒度となる. よって, サービス・コンポーネントも小さくなり, サービス全体のうち通信のオーバーヘッドが多くを占めることとなりかねない. 既存のATM, テラー端末など大量・高速で既存のハブを求めるチャネルと複雑処理向きのESBを求めるチャネルでバスを分離していくなどの考慮も必要だと考える. コンポーネント粒度と非機能要件を十分に加味し, 現実的なテクノロジーを選択することが重要である.

6. おわりに

本論文では, 銀行統合の実践において,「コンポーネント定義」と「実装テクノロジー選択」という2つのSOAアプローチを取り入れ3つのサービス・バスをデザインすることで, 既存サービスへの影響を抑えつつ, 拡張性および再利用性を確保できることを実証した. 今回は銀行統合での実践を題材に論じたが, このアプローチ手法については, 金融のみならず, グループ企業間の接続や他業態での統合などにも十分活用可能であると考える.

参考文献

[1] 加速する金融界の再編(全国銀行協会), http://www.morebank.gr.jp/category_a/02100_genjo10.html (2006.8.19).
[2] Gary A. Dymski:銀行合併の波, 日本経済評論社, ISBN 4-8188-1547-0 (2004).
[3] David A. Chappell:*Enterprise Service Bus*, Sonic Software, ISBN 4-87311-220-6 (2005).
[4] Dirk Krafzig et al., : *Enterprise SOA: Service Oriented Architecture Best Practices*, PrenticeHall Ptr, ISBN 0-13-146575-9 (2004).
[5] 米持 幸寿 :基礎からわかるSOA, 日経BP社, ISBN 4-8222-8230-9 (2005).
[6] 日本BEAシステムズ:サービス指向アーキテクチャ, 翔泳社, ISBN 4-7981-0848-0 (2005).
[7] SOMA Technique Paper, http://d06dbs51.ibm.com/services/emea/nordic/3nrgs.nsf/pages/soma (2006.8.19).
[8] IBMビジネスコンサルティングサービスIT戦略グループ :エンタープライズ・アーキテクチャ, 日経BP社, ISBN 4-8222-1873-2 (2004).
[9] ガートナーレポート:メッセージ・ブローカ:アプリケーション統合への集約的アプローチ, ガートナー, SAR-96-11(1996).

日本アイ・ビー・エム株式会社
銀行第一AS.銀行第四サービス部
部長,ICP-シニアITアーキテクト
加藤 洋 Hiroshi Katoh

[プロフィール]
1990年日本アイ・ビー・エムに入社.銀行担当のSEとして,分散系,Web系システムを中心に担当.
某都市銀行様の勘定系システムの営業店サーバを設計・構築した新営業店プロジェクト,融資稟議/個人ローンプロジェクトなど大規模なプロジェエクトに参画.
現在は,SIサービス部隊のマネージャー兼ITアーキテクトとして,メガバンク様のシステム統合プロジェクトを実施.

日本アイ・ビー・エム株式会社
金融クライアントIT推進
ICP-アドバイザリーITアーキテクト
三谷 隆 Takashi Mitani

[プロフィール]
1991年に日本アイ・ビー・エム入社.製造業・流通業・金融機関など多彩なお客様の担当SEを経験.1999年より銀行様担当SEとして分散系,Web系を中心とした各種プロジェクトに携わり,現在はクライアントITアーキテクト(CITA)としてアーキテクチャ構築全般に従事.

日本アイ・ビー・エム株式会社
銀行第一AS.銀行第四サービス部
ICP-アドバイザリーITアーキテクト
杉山 真一 Shinichi Sugiyama

[プロフィール]
1991年日本アイ・ビー・エムに入社.金融のお客様担当SEとして都市銀行様を担当.
主に都市銀行様の勘定系システムを核とした,他システムとの連携を中心に各種プロジェクトに携わる.メッセージ・ブローカーを構築した新営業店プロジェクト,融資稟議/個人ローンのBPRプロジェクト(イントラネットWeb)など大規模なプロジェエクトに参画.
現在,メガバンクのお客様担当のITA(ITアーキテクト)として,お客様のシステム統合サポートを行っている.