Vol.2009-SE-166 No.15
2009/11/6

# UML モデルを活用した SOA の設計・実装プロセスの検証

齋藤 伸也†, 宗平 順己†, 大場 克哉†

†株式会社オージス総研

要旨:

SOA 開発にはトップダウンアプローチとボトムアップアプローチがあり，トップダウンアプローチはビジネスモデリングからシステムを設計する手法である．SOA はビジネスと IT システムの整合性を図るアーキテクチャであり，その実現のためには業務担当者とシステム担当者のコミュニケーションギャップを小さくする必要がある．

そこで筆者らは UML ベースの SOA 開発プロセスを提案した．この方法論の検証のために仮想企業による SOA 開発を実践した．本報告では，サービスの分析・設計にフォーカスし，具体的な手法を示すとともに，SOA 開発プロセスの検証結果を報告する．

# Development Process with UML Models for Service Oriented Architecture Design and Implementation

Shinya Saito†, Toshimi Munehira†, Katsuya Oba†

†Osaka Gas Information System Research Institute Co, Ltd

Abstract:

There are two approaches to implement Service Oriented Architecture (SOA); a top down approach and a bottom-up approach. A top down approach is a technique to design the IT systems starting with business modeling. Since SOA is an architecture that attempts the alignment of the business and the IT systems, it is necessary to reduce the communications gap between persons in charge of business affairs and persons in charge of systems to achieve their business objectives. The authors proposed the SOA development process using modeling with Unified Modeling Language. Also, the authors implemented some SOA scenario in a virtual company to verify the proposed methodology. In this report, the authors describe the service analysis and design phases with the outcome of the implementation.

## 1. はじめに

Service Oriented Architecture (SOA)は，ビジネスと IT システムの整合性を確保し，ビジネス環境と IT 環境の変化に柔軟に対応するための迅速性，適応性を実現するために，サービスを組み合わせてシステムを構築するアーキテクチャである．SOA を実現することで，情報システムの全体最適を実現し，変更のコストを削減することができる．高い価値を持つビジネス機能群を個別の再利用可能なサービスとして配置することで，サービス間の接続やオーケストレーションはより容易に実現できる．

SOA に基づくシステム開発の方法として，ビジネスモデルからビジネスに必要なサービスを定義するトップダウンアプローチと，現在のシステムからどのようなサービスを定義するのかを検討するボトムアップアプローチとがある．

本研究ではBPM+SOA[1]というトップダウンアプローチの開発手法の中の「サービス分析・設計」にフォーカスし，その具体的な手法について示すとともに，仮想事例を用いた検証結果について報告する．

## 2. 従来のシステム開発

従来のトップダウンアプローチの開発手法は業務担当者が作成したビジネスモデルから，システム担当者がシステム設計を行う．

ビジネスモデリングは，活動系アプローチ，相互作用アプローチ，目標指向アプローチの３つのモデリングアプローチに大別され，DFD，IDEF，UML，バランススコアカード(BSC)などが利用される．

一方，システムの設計の中では構造化アプローチ，データ中心アプローチ，オブジェクト指向アプローチなどがあり，DFD，ER 図，UML などが利用される．

SOA は，ビジネスと IT の整合性を目的とするため，その開発にはトップダウンアプローチを採用することが適していると考えられる．しかしながら，従来のトップダウンアプローチをそのまま適用するには，次のような課題がある．

1) ビジネスモデルらシステム設計を行う過程で少なからず担当者間のコミュニケーションギャップが発生する(図 1)．SOA は，ビジネスと IT の整合性の確保を目的とするため，このようなギャップが発生しないよう，開発プロセスを定義する必要がある．
2) SOA に基づいたシステム開発は１度きりで終わるものではなく情報システムの全体最適を目指して繰り返し行われる．繰り返し開発を行っていく中で情報システムからビジネスモデルへのフィードバックを行うことが重要になってくる．しかしながら，従来のトップダウンアプローチにはフィードバックの手法が明確化されていない．

この 2 つの課題を解決するために本研究では[1]で示されたビジネスプロセスモデルと情報システム設計モデルのシームレスな一貫性を実現するために統一モデリング言語である UML を活用した開発プロセスを適用した，

3 章からは参照論文では詳細に定義されていない「サービス分析・設計」および「サービス実装」の手法を明確に定義する．

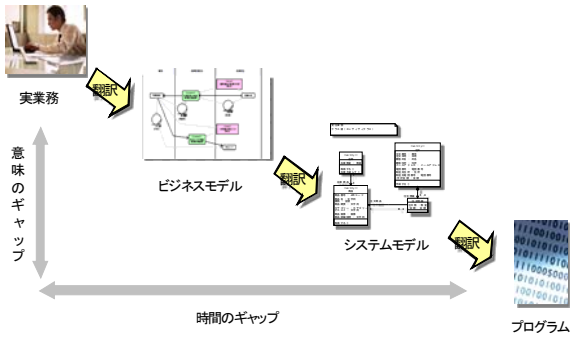


図 1　実業務とシステムのギャップ[2]

## 3.　モデルベース SOA 開発プロセス

### 3.1　概要

ソフトウェアの開発プロセスには，ウォータフォール型，Rational Unified Process(RUP)に代表される繰返し型や，XP に代表されるようなアジャイル的な手法が存在する．モデルベース SOA 開発プロセスでは，各作業を繰り返し行っていく手法を取っている．これは BPM が PDCA(Plan, Do, Check, Act)のサイクルを回して業務改善を行っていくことに対応して，IT システムも繰り返しのサイクルで開発するためである．

モデルベース SOA 開発プロセスでは大きく４つの役割(ロール)を定義する．

1)　ビジネスモデラ
　業務を分析し，企業のあるべき姿をモデリングする．
2)　サービス分析者
　ビジネスモデリングから情報システムのサービスの分析を行う．
3)　サービス構築者(サービスプロバイダ)
　サービス分析モデルからサービスの設計，実装を行い，サービスを提供する．
4)　アプリケーション構築者(サービスコンシューマ)
　構築されたサービスを利用し，情報システムを構築する．

モデルベース SOA 開発プロセスは大きく 6 つの作業分野から構成される(図 2)．この作業を繰り返し行い SOA に基づくシステムを構築してゆく．

1)　ビジネスモデリング
　本報告ではビジネスモデリングについては概要だけを述べるにとどめ，詳細は文献[3]に譲ることとする．
　ビジネスモデリングでは，ビジネスの現状の姿(As-Is)とあるべき姿(To-Be)を UML によってモデル化し，そのギャップを明確にする．To-Be モデルのデザイン方法としては戦略マネージメントツールである BSC(Balanced Scorecard)を利用する．ビジネスモデルは，目的ビュー，プロセスビュー，組織ビュー，ネットワークビューの４つのビューから構成され，それぞれ異なった視点から，ビジネスを可視化する．
　ビジネスモデリングのアウトプットは以降の作業の重要なインプットとなる．特にプロセスビューのアウトプットとなる業務フローからはサービスの抽出を行うために重要である．また，サービスのメッセージを分析するには組織ビューのアウトプットとなる概念モデル[4]が有効なインプットとなる．

2)　サービス分析
　サービス分析では，ビジネスモデリングをもとにビジネスと整合性がとれ，かつ情報システムとして機能するサービスの論理構造を決定する．この論理構造はサービス分析モデルとして表現される．
　この作業は SOA 開発プロセスの中でも，ビジネスと IT の整合性を図る重要なものである．本報告の 3.3 節にて詳細に説明する．

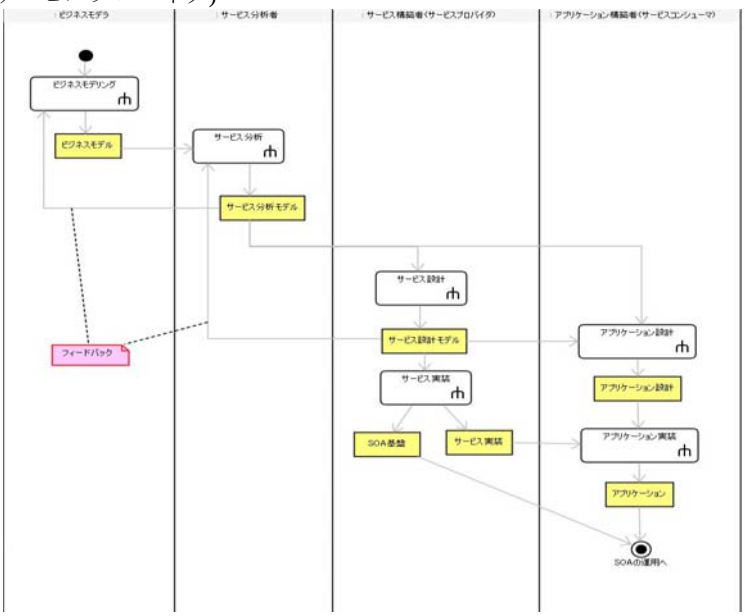


図 2　モデルベース SOA 開発プロセス

3) サービス設計

サービス設計では，サービス分析モデルを実装可能なレベルまで詳細化する．SOA の初期開発においては SOA 基盤の設計も含まれる．本報告の 3．4 節にて詳細に説明する．

4) サービス実装

サービス実装では，インタフェース定義されたサービスを Java や.NET，その他プログラミング言語を利用して実装する．サービスの実装技術は多岐にわたり，新しいツールや製品が次々とリリースされている．SOA のサービスを実装する技術としては，言語に依存しない Web サービス(SOAP or REST)の技術を用いることが一般的である．

5) アプリケーション設計

アプリケーション設計では，従来のアプリケーション開発とほぼ同様の作業を行う．従来と異なる点は，サービス間の接続やオーケストレーションを中心に設計してゆくところである．

6) アプリケーション実装

アプリケーション設計に基づいて実装を行う．ユーザインタフェースやサービス間の接続やオーケストレーションを実装する作業が中心になる．ワークフローエンジンや，BPMS などのツールが利用される．

### 3.2 初期開発と繰り返し開発

モデルベース SOA 開発プロセスにおいて初期開発と繰り返し開発は 3．1 節で示した作業を行うことに違いはない．しかし，繰り返し開発の場合，各作業のインプットに既存のサービス，SOA 基盤が加わることになる．

このとき，繰り返し開発の中で類似のサービスを構築しないようにサービスリポジトリによるサービス管理を行う．

モデルベース SOA 開発プロセスの繰り返し開発(図 3)は，以下のように述べることができる．

「リアルビジネスに変更があった場合，ビジネスモデルを変更し，その新しいビジネスプロセスにサービスの再割り当てを行う．既存のサービス群から割り当てができる場合には，直ちに新しいビジネスプロセスを動かすことができる．サービスが不足する場合には，そのサービスを実現するコンポーネントの開発のみを行う」[2]

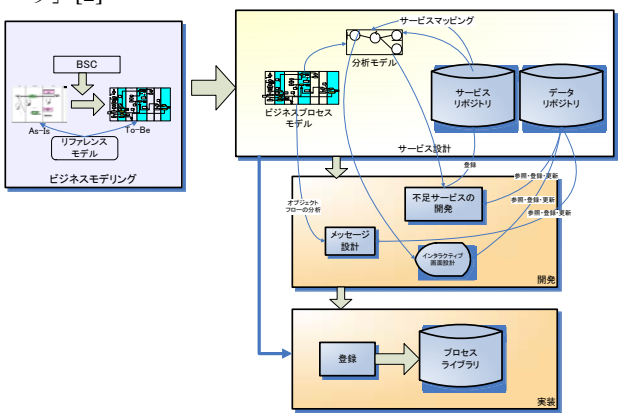


図 3 サービスリポジトリを利用したビジネスモデリング

### 3.3 サービス分析

まず始めに，サービス分析作業の大まかな流れと入力や出力となるモデルを概観する．図 4 に作業の詳細をアクティビティ図で示した．

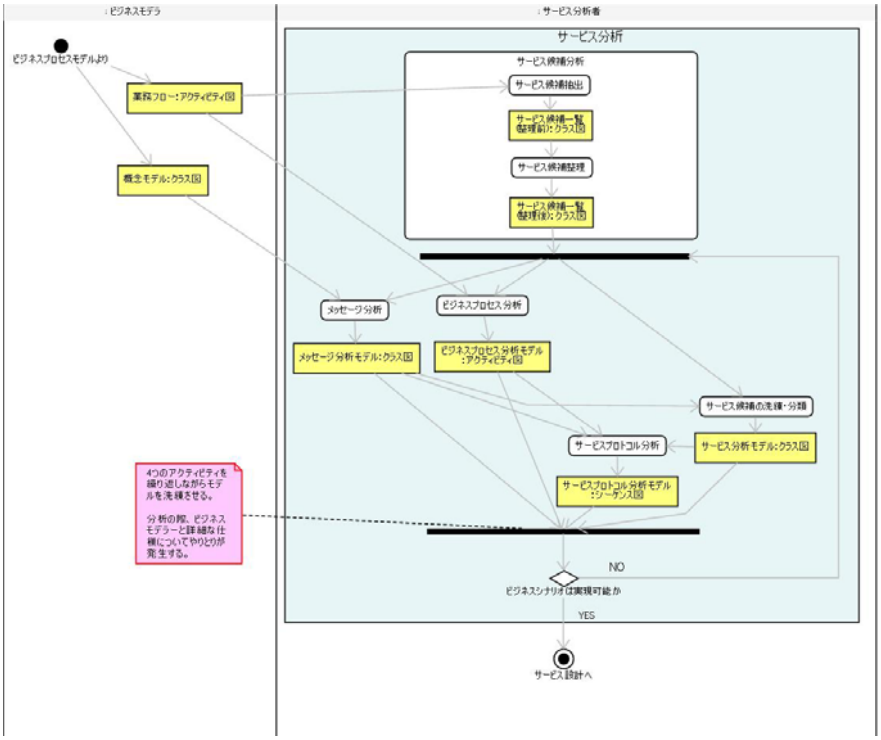


図 4 サービス分析作業

サービス分析は大きく２つの作業で構成される．サービス候補分析とサービス候補の洗練である．サービス候補の洗練の作業はさらに，サービス候補の洗練と分類，メッセージ分析，ビジネスプロセス分析，サービスプロトコル分析に分かれており，それぞれ異なった視点からサービス候補を分析するための作業である．これらの作業は互いに関連し合っており，繰り返し行うことでサービスを明確化していく．

### 3.3.1 サービス候補分析

サービス候補分析はアクティビティ図で表現されている業務フロー(図 5)をインプットにサービス候補を抽出していく作業である．抽出されたサービスは，そのまま「サービス」として定義可能なものと「サービスの操作」として定義されるものある．この２つの区別はアクティビティ図の階層構造や，アクティビティが関連しているビジネスエンティティから判断することができる．「サービス候補」，「サービス操作候補」，「ビジネスエンティティ」の関連をサービス候補一覧(図 5 左上)で表現する．

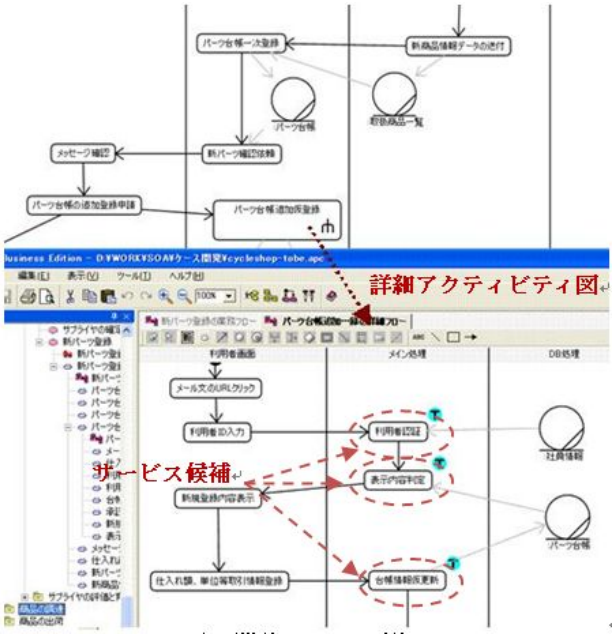


図 6　業務フローの例

次に抽出されたサービス候補を整理する．このときの観点はサービスの粒度を揃えることと，再利用性である．この作業を行う際の指標となるのが，ビジネスエンティティに対する CRUD(create, read, update, delete)操作である．例えば，複数のサービス操作候補から同じビジネスエンティティへ参照(Read)がある場合は，1 つのサービスにまとめる．サービス操作候補が複数のビジネスエンティティに操作を行っている場合，そのサービス操作候補は 1 つのサービスとし，複数のサービス操作に分割する．

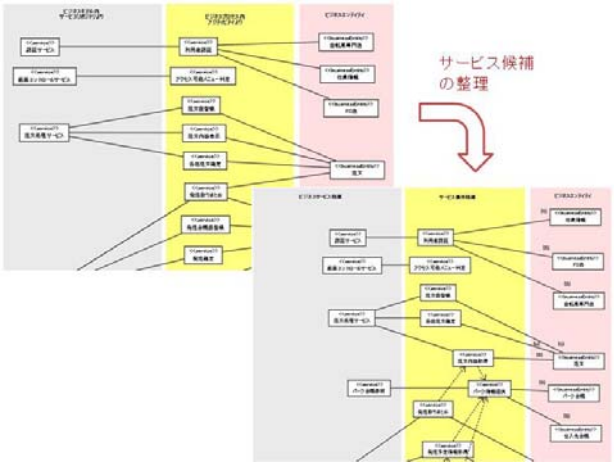


図 7　サービス候補一覧モデルの例

### 3.3.2 サービス候補の洗練・分類

サービス候補の洗練・分類はサービス候補一覧を元にサービスを分類する．SOAにおけるサービスは，その粒度や役割によって次のように分類できる[5].

1)　プロセスサービス
　業務フローそのものをシステムで実現するためのサービスである．業務フローの変更はこの層のサービスの変更になるため，再利用性は低い．下位のビジネスサービスのオーケストレーションによって実現され，その実装にはビジネスプロセス記述言語などが利用されることが多い．状態を持ったサービスになる．サービス利用者(サービスコンシューマ)によって実装される．
2)　ビジネスサービス
　業務フローを実現する個々の要素に対応するサービスである．プロセスサービスのような状態を持たず再利用性が高い．サービス提供者(サービスプロバイダ)によって実装される．
3)　基本サービス
　基本サービスはプロセスサービスまたはビジネスサービスに共通的に利用されるサービスである．再利用性が高い．基本サービスは単体ではビジネス価値を提供しない．

分類されたサービスはサービス分析モデル(図 7)として表現する．

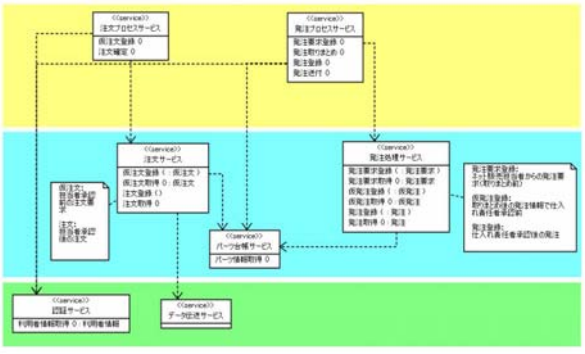

図 8　サービス分析モデルの例

### 3.3.3 メッセージ分析

メッセージ分析はサービス間でやりとりされる情報を明確化する．概念モデルをインプットにし，それらを情報システムで実現することを考慮してメッセージ分析モデル(図 8)を作成する．

ビジネスサービスと基本サービスは状態をもたない(ステートレス)なサービスとして定義されるため，それらのサービスで利用されるメッセージは状態を含めたデータの集合として定義する．

対象業務によっては，分析作業の中でメッセージ分析が最も重要な場合がある．例えばビジネスモデルの中では同じ顧客情報に対する操作として表現されていたが，メッセージ分析を行うと部署ごとに若干項目が異なる情報を取り扱っていることが判明したとする．その場合，再利用性の高いサービスにするために双方のデータ属性を持ったメッセージを定義する．

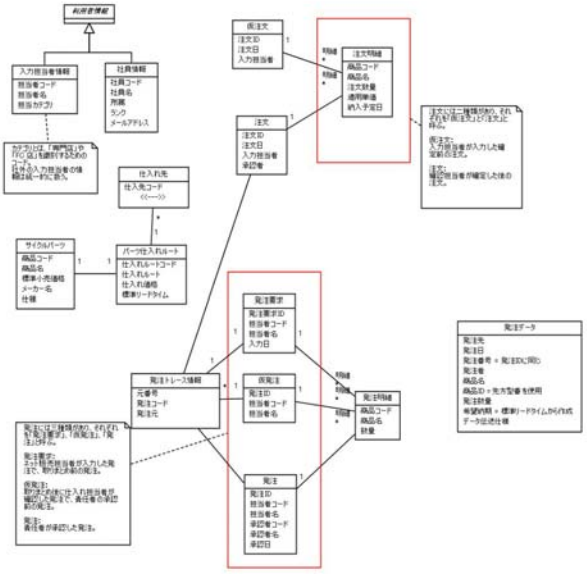

図 9　メッセージ分析モデルの例

### 3.3.4 ビジネスプロセス分析

ビジネスプロセス分析では，サービス候補の洗練・分類によって定義されたサービスを利用して業務フローが実現できるのかを検証する．成果物はビジネスプロセス分析モデル(アクティビティ図)である．ビジネスプロセス分析を行うことで，業務フローを実現するためのサービス及びサービス操作が不足なく定義されているかどうか検証できる．サービス及びサービス操作が不足していた場合，サービスの追加を検討する．

### 3.3.5 サービスプロトコル分析

ビジネスプロセス分析に対して，サービスプロトコル分析では各サービス間のメッセージのやりとりを中心にして，定義されたサービスを検証する(図 9)．この作業では特にプロセスサービスがどのように下位サービスを呼び出すのかを明確化することができる．サービスプロトコル分析の際にも，サービスの不足を発見することができる．

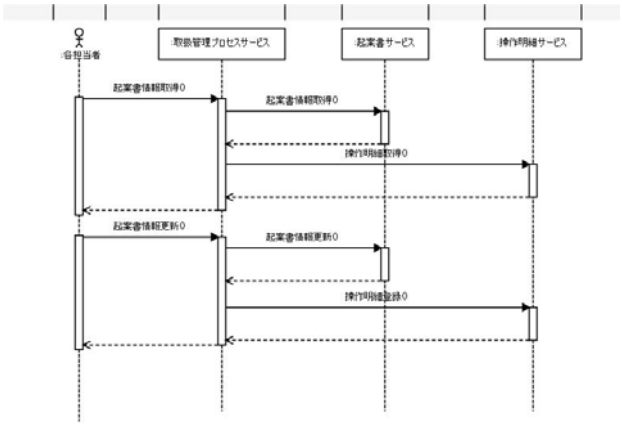


図 10　サービスプロトコル分析モデル

### 3.3.6 ビジネスモデルへフィードバック

以上で述べたサービス分析の作業を経て，作成されたサービスモデルはサービスリポジトリに格納する．同じサービスモデルを利用するとこでビジネスモデルとサービス分析モデルの整合性を図る．

## 3.4　サービス設計

サービス設計作業の大まかな流れと入力や出力となるモデルを概観する．図 10 に作業の詳細をアクティビティ図で示した．

サービス設計ではサービス分析の各モデルをインプットにそれらをシステムとして実現させるための設計作業を行う．サービス分析までは日本語によるサービス名を使っている場合でも．サービス設計で英字の表記に変更して，システムで実現可能なモデルにする．

初期の SOA 開発の場合，SOA の基盤の設計もサービス設計作業に含まれる．これは初期の段階で出てくるサービスには認証・認可など基盤として実現されるような共通的なサービスが含まれるため，サービス自体の設計と合わせて行わなければならないからである．

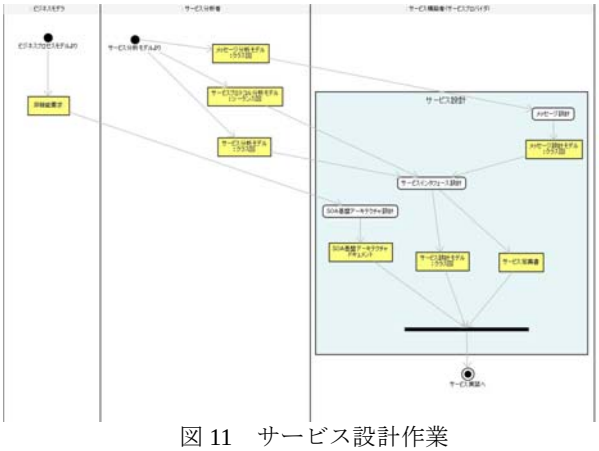

図 11　サービス設計作業

### 3.4.1 メッセージ設計

メッセージ設計ではメッセージ分析モデルをインプットにシステムとして実装可能なレベルまで詳細化し，メッセージ設計モデルを作成する．メッセージ設計ではシステムの非機能要件を考慮に入れて，メッセージを設計しなければならない．3.3.3 節で述べたように共

通性のあるメッセージモデルを維持すれば概念モデルに近い状態を保つことができるが，その代り特定の処理には関係のない無駄なデータがネットワーク上で送受信されることになる．

逆に，共通なメッセージを必要な情報に分割すると，分析モデルから乖離し，複雑になる一方，ネットワークリソースの消費を抑えることにつながる．このトレードオフは SOA 基盤上で解決する方法が存在する．

#### 3.4.2 サービスインタフェース設計

サービスインタフェースの設計では，サービス分析モデルとサービスプロトコル分析モデルをインプットにし具体的な設計を行う．サービスインタフェースの設計方針として重要なのが，手続き呼び出し型のサービスなのか，イベント型のサービスなのかである．

- 手続き呼び出し型
  同期的にサービスが呼び出される．この方式は比較的統合されているプロセスに適している．
- イベント呼び出し型
  非同期的にサービスが呼び出される．この方式はサービスプロバイダとサービスコンシューマの組織が異なり，それぞれの依存度が低い場合に有効である．

この２つの方式にはそれぞれメリット/デメリットがあり，非機能要求なども考慮に入れ決定していかなければならない．

サービス設計モデルはクラス図で記述されるが，非機能要件などの情報は UML では表現しきれない．そこで，サービスインタフェース設計ではサービス設計モデルをベースに非機能要件などの情報を盛り込んだサービス定義書(図 11)を作成する．

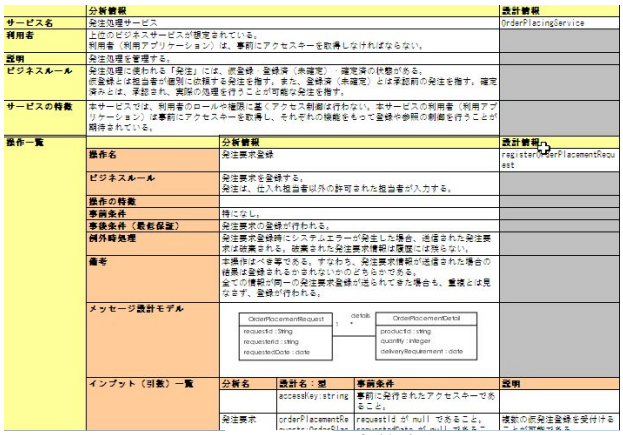

図 12　サービス定義書例

#### 3.4.3 SOA 基盤アーキテクチャ設計

SOA の価値は利用できるサービスが増えれば増えるほど上昇する．しかし，全てのサービスが同じ技術で実装されることは現実的ではない．異なる技術でサービスを連携する場合は，実装の違いを吸収するレイヤーが必要になる．それが SOA 基盤である．

SOA基盤に求められる要件としてはプロトコルなどの実装や非機能要件の実現をアプリケーションから切り離すこと，さらにはサービスの仕様変更の影響をアプリケーションに波及させないことである[6]．具体的には以下のような項目が挙げられる．

- サービスの認証・認可
- メッセージ変換
- メッセージルーティング
- フェイルオー場と負荷バランシング
- サービスの監視と SLA

これらの要件を満たすには ESB(Enterprise Service Bus)のようなミドルウェアをただ利用するだけではなく，サービス，アプリケーション，アプリケーションフレームワーク全体として基盤アーキテクチャを考えなければならない．

## 4.　仮想事例による検証

本章では，前章で説明したモデルベース SOA 開発プロセスを仮想企業の SOA システム構築プロジェクトに適用した事例を紹介する．

### 4.1 概要

検証プロジェクトは専門的なサイクルパーツを取り扱う卸小売を行う企業の商品の企画，商品の調達に関わる業務改善を行い，容易に業務移行が行えるシステム構築を行うプロジェクトである．

実施体制は，次の表 1 に示すように 3 地点で分散して開発を進めた．

表 1　実施体制

| 主な作業 | 地域 |
|---|---|
| ビジネスモデリング | 大阪 |
| サービス分析・設計 | 東京 |
| サービス・アプリケーション実装 | 上海 |

上海のオフショア開発者は以前に UML を利用したことがあり UML の読解には問題はなかった．また SOA 開発の経験はなく，利用する ESB や Web サービスなどの技術も初めて利用した．

### 4.2　繰り返し開発

図 2 に示す開発プロセスに従い，To-Be ビジネスモデルを設計後，サービス分析，サービス設計，サービス実装，アプリケーション設計，アプリケーション実装を進めた．この開発プロセスを次に示す通り 3 イテレーション繰り返した．各 Phase の開発対象はビジネスモデルの業務フロー単位である．

- Phase1 (11 月下旬~1 月下旬)
  「サイクルパーツ調達フロー」のシステム構築
- Phase2(1 月下旬~3 月下旬)
  「新パーツ登録のフロー」のシステム構築
  「サイクルパーツ調達のフロー」の改善
- Phase3(5 月下旬~8 月上旬)
  「サプライヤ候補の識別」及び
  「サプライヤの確定と承認」のシステム構築

### 4.3　ソフトウェアアーキテクチャ

本プロジェクトでは図 12 に示すソフトウェアを利用し，システムを構築した．

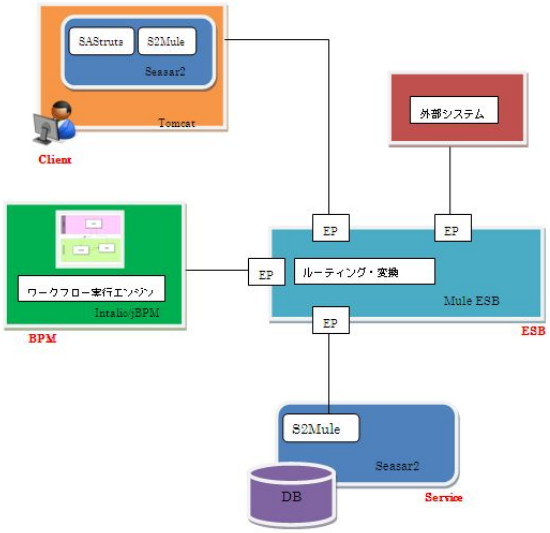


図 13　ソフトウェアアーキテクチャ

各コンポーネントの役割を次に示す.

- Client
  ユーザと対話を行う画面を提供する.
  Seasar2 及び SAStruts を利用して構築した.
- BPM
  ビジネスプロセスモデルによって定義されたプロセスの制御を行う．3．3．2 節で示したプロセスサービスに該当する処理を行う.
- ESB
  各コンポーネントを連携する．各コンポーネントに合わせたメッセージ変換を行う.
  Mule ESB を利用して構築した.
- Service
  3．3．2 節で示したビジネスサービス及び基本サービスに該当する処理を行う.
  Seasar2 及び S2Mule を利用して構築した.

## 4.4　適用のまとめ

### 4.4.1 サービスの再利用

表 3 に検証プロジェクトで作成したサービス実装のソースコード行数を示す．Phase 毎によりソースコード行数が多いシステムとなっていることがわかる．しかし，追加開発の行数に注目すると，Phase2 で追加開発したソースコードの行数(6825 行)は Phase1 で開発した(8413 行)より少ないことがわかる.

再利用されたサービスの全ての行数が有効であったとは考えがたいが，単純なソースコードの再利用行数は Phase2:3549 行，Phase3:6842 行とサービスのソースコード全体の 30%が再利用されたソースコードであると言える.

また，表から再利用率が高いサービスには以下の傾向があることが読み取れる.

- 基本サービスは再利用率が高い
- ビジネスサービスでは業務の中心となる概念を取り扱っているサービス(パーツ台帳サービス，注文サービス，仕入先台帳サービス)の再利用率が高い.

### 4.4.2 コミュニケーションギャップの解消

オフショア開発を実施する場合，発注側はプロジェクト期間中にオフショア先に複数回出張し，オフショア開発者と打ち合わせを行うのが一般的である．しかし，本プロジェクトでは Phase1 が始まる直前に打ち合わせを行った以外はすべて大阪，東京，上海と離れた地域で作業を行った．オフショアではない開発に比べ，オフショア開発は国や文化の違いからコミュニケーションギャップが生まれ，要求とシステムの乖離が起こりやすい傾向にある(表 2).

このような状況下でビジネスモデルの要求に沿ったシステムを構築できたのは UML の力が大きい．サービス実装+設計を担当した上海のオフショアの開発者からも KPT 分析で，以下のようなコメントがあがっている.

- 設計段階に UML 技術をよく使います．打ち合わせと開発理解の方に大変役に立ちました.

表 2　各サービスのソースコード行数

| | | | コード行数* | | | | | |
|---|---|---|---|---|---|---|---|---|
| 分析名 | 実装名 | 種類 | | Phase1 | Phase2 | Phase3 | 追加開発(Phase2) | 追加開発(Phase3) |
| 認証プロセスサービス | AuthenticateProcessService | プロセスサービス | | 519 | 519 | 624 | | 105 |
| 注文プロセスサービス | OrderProcessService | プロセスサービス | | 1193 | | | | |
| 発注プロセスサービス | OrderPlacingProcessService | プロセスサービス | | 1782 | | | | |
| 新パーツ登録プロセスサービス | CyclePartsProcessService | プロセスサービス | | | 1857 | | 1857 | |
| 取引依頼プロセスサービス | PartsSellerRequestProcessService | プロセスサービス | | | | 1783 | | 1783 |
| 問い合わせプロセスサービス | QuaereProcessService | プロセスサービス | | | | 805 | | 805 |
| 仕入管理プロセスサービス | TradeOperationProcessService | プロセスサービス | | | | 1669 | | 1669 |
| パーツ台帳サービス | CyclePartsLedgerService | ビジネスサービス | | 611 | 2817 | 2817 | 2206 | 0 |
| 発注処理サービス | OrderPlacingService | ビジネスサービス | | 1889 | | | | |
| 注文サービス | OrderService | ビジネスサービス | | 1193 | 1675 | | 482 | |
| 仕入先台帳サービス | SellerLedgerService | ビジネスサービス | | | 716 | 1252 | 716 | 536 |
| 操作明細サービス | OperationInfoServide | ビジネスサービス | | | | 1227 | | 1227 |
| 取引依頼サービス | PartsSellerRequestService | ビジネスサービス | | | | 1116 | | 1116 |
| 問い合わせサービス | QuaereService | ビジネスサービス | | | | 1219 | | 1219 |
| 起案書サービス | SuggestDocumentService | ビジネスサービス | | | | 1424 | | 1424 |
| データ伝送サービス | DataTransmissionService | 基本サービス | | 687 | 687 | 1256 | | 569 |
| 認証サービス | UserService | 基本サービス | | 539 | 539 | 887 | | 348 |
| メッセージ送付サービス | MessageSendService | 基本サービス | | | 1564 | 1772 | 1564 | 208 |
| | | | 合計 | 8413 | 10374 | 17851 | 6825 | 11009 |

※Java コードのみ，設定ファイル(XML)は除く．コメント，空行は除く

表 3　オフショア開発における課題と深刻度 – 発注側[7]

| | | 極めて深刻 | かなり深刻 | 深刻 | やや深刻 | 全く深刻ではない | 深刻度が高い割合(%) |
|---|---|---|---|---|---|---|---|
| 1 | 言語や文化の差が原因で誤解が生じる | 4 | 19 | 14 | 13 | 10 | 38 |
| 2 | オフショア企業側に伝達内容を正確に伝えるのに時間がかかりすぎる | 1 | 15 | 16 | 18 | 10 | 27 |
| 3 | 仕様書の曖昧さが原因で誤解が生じる | 1 | 11 | 14 | 22 | 13 | 20 |
| 4 | 仕様書とオフショア企業側から納品されたソースコードに乖離が生じる | 4 | 17 | 21 | 14 | 4 | 35 |
| 5 | 仕様書に漏れ・抜けが起こる | 2 | 13 | 19 | 16 | 10 | 25 |
| 6 | オフショア企業からくる大量の質問への対応で業務が圧迫される | 6 | 19 | 19 | 13 | 2 | 42 |
| 7 | テスト段階以降になって品質の問題が多く露呈する | 3 | 14 | 11 | 26 | 6 | 28 |
| 8 | 仕様書を詳細に記述するための負担が大きい | 3 | 14 | 22 | 14 | 8 | 28 |
| 9 | オフショア企業側でのテスト内容に不備、不足がある | 5 | 13 | 14 | 22 | 6 | 30 |
| 10 | オフショア企業の担当者が離職することで、業務の継続に支障がある | 12 | 22 | 13 | 9 | 3 | 58 |
| 11 | オフショア企業が保守・メンテナンスを担当する場合、バグの改修におけるリードタイムが大きい | 11 | 23 | 16 | 7 | 2 | 58 |
| 12 | オフショア企業と同じイメージで効果的な情報共有ができない | 7 | 27 | 17 | 6 | 2 | 58 |

ビジネスモデル，サービス分析モデル，サービス設計モデルの間でそれぞれ整合性が取れ，UML で記述されているからこそ実装担当者であるオフショア開発者が開発するものの背景(ビジネスモデル)を理解することができたと考えられる．

オフショア開発者がビジネスモデルを理解することで表 3のオフショア開発の課題を解決するだけではなく，上海側からサービス分析・設計モデルに対しモデルの矛盾の指摘や，サービスインタフェースの提案など有益なフィードバックが得られるようになった．その結果，表 4 に示す通り Phase 毎に上海の役割の範囲が広がり分析・設計の作業を行うことが可能になった．

表 4　役割の拡大

| Phase | 役割 |
|---|---|
| 1 | サービス・アプリケーション実装 |
| 2 | Phase1 に加えサービス設計 |
| 3 | Phase2 に加えサービス分析の一部 |

## 5. まとめおよび今後の課題

本報告ではビジネスモデルとシステムモデルのシームレスな一貫性を実現した SOA の開発プロセスを提案し，仮想企業を想定した SOA プロジェクトに適用した．有効性および課題をまとめる．

### 5.1 有効性

- 提案した SOA 開発プロセスはオフショア開発に非常に適していることがわかった．
- 提案した SOA 開発プロセスは再利用性の高いサービスを効率よく設計・開発することを示した．
- ビジネスモデリングとサービス分析モデル，サービス設計モデルの整合性が保たれ，各担当者間で共通したイメージを持って開発を行うことができる．

### 5.2 課題

- 仮想事例では，構築したシステムの本格的な運用については検証することができなかった．SOA の運用からのフィードバックを含めた SOA 環境下でのソフトウェアライフサイクルの検証を行う必要がある．
- 仮想事例ではサービス設計モデルからサービスの実装はモデルを参照しながら，プログラミングを行った．設計モデルから実装をシームレスに行うためのツールのサポートの検討を進めている．
- サービス分析，サービス設計の手法の標準化及びガイドラインの作成．
- 仮想事例では，非機能要件の要求が厳しいものではなかった．パフォーマンスが求められるシステムのサービスの分析・設計の方針を考える必要がある．

今後は，提案した開発プロセスを実プロジェクトへ適用，評価することにより改善していく予定である．

## 参考文献

[1] 宗平順己，「BPM+SOA，ビジネスサイドからの情報システム設計の新しいアプローチ」，
日本情報経営学会誌 Vol.28，No.2(20071220) pp. 88-96
[2]宗平順己著，島田達巳監修「情報システムの分析と調達」，日科技連出版社，2008 年，図 8-3
[3] 宗平順己，「SOA 設計のためのビジネスモデリング手法の検討」経営情報学会 2008 年度春季全国研究発表大会
[4] Eric Evans，「Domain-Driven Design」p75
[5]株式会社オージス総研「仕事の流れで理解する 実践!SOA モデリング」
[6]橋本誠，大場克哉，藤倉成太，宗平順己
「サービス指向アーキテクチャ(SOA)に基づくアプリケーション設計の現状と課題 」情報処理学会ソフトウェア工学研究会　2005 年
[7]特定非営利活動法人 UML モデリング推進協議会 オフショアソフトウェア開発部会「オフショア開発向け UML 適用ガイドライン Ver. 2.0」http://www.umtp-japan.org/