Department of Social Systems and Management

Discussion Paper Series

No.1255

流通ビジネスメッセージ標準の分析と SOA 環境での取引システム試作

by

山田泰介
佐藤亮

March 2010

UNIVERSITY OF TSUKUBA
Tsukuba, Ibaraki 305-8573
JAPAN

流通ビジネスメッセージ標準の分析と SOA 環境での取引システム試作

Analysis of Distribution Business Message Standard and Prototyping a System in an SOA Eivironment based on BMS

山田泰介 [1]　　佐藤亮 [2]

[1] 筑波大学　大学院博士前期課程　システム情報工学研究科
[2] 筑波大学　システム情報工学研究科
[1]Graduate School of Systems and Information Engineering, University of Tsukuba
[2]Department of Social Systems and Management, University of Tsukuba

概要

本論文では次世代の電子商取引(EDI)である流通ビジネスメッセージ標準（流通 BMS）の普及の妨げとなっている技術的要因を分析し、流通 BMS の XML スキーマである SecondGenEDI XML スキーマの構造に由来する問題と DBMS 実装環境に由来する問題があることを明らかにする。

さらに、複雑化する IT 環境へと対応するためのアーキテクチャであるサービス指向アーキテクチャ（SOA）を用いる際の、流通 BMS で想定される一般的な受発注プロセスを実装するための SOA 環境を確定する。流通 BMS のデータ形式を用いてサプライチェーン間の在庫変動を調べるプロセスを考え、それを SOA 環境で試作し評価を行い、流通 BMS を SOA 環境で用いるシステムの特徴の一端を明らかにした。

１．序論

企業の活動の大半は伝票を通じて行われる。企業は IT 技術の導入によって、企業情報システムにおける伝票の電子化を行い、1970 年代後半からチェーンストア界や銀行業界をはじめとして多くの企業が電子商取引(EDI)へと移行してきた。

電子商取引とは伝票を電子化したデータのフォーマットをやり取りすることで取引をする仕組みであり、コンピュータを使うことでコストを削減することができる。財団法人日本情報処理開発協会電子商取引推進センター(JIPDEC/ECPC)と EDI 推進協議会(JEDIC)「国内外の EDI 実態調査報告書-2006 年版-」によると EDI 導入済み企業は回答を行った 267 社中 194 社(72.6%)、うち大企業では 88%、中小企業では 63%が導入している[30]。

しかし、この電子商取引は 1 対 1 での取引が基本であり、取引先ごとにフォーマットが決められている。加えて、通信には専用回線、専用モデムが必要であるという問題もある。さらに専用回線によるデータの送受信が遅い、専用モデムメーカーのサポート停止といった問題が発生した。そこで経済産業省主導の下、2003 年度から「流通サプライチェーン全体最適化促進事業」を 3 年間行い、電子商取引の標準的な仕組みを検討した。そして、2006 年度に「流通システム標準化事業」を実施し、電子商取引の標準的な仕組みを「流通ビジネスメッセージ標準」(流通 BMS)

とすることを定めた[24]。 しかし、従来の EDI に置き換わるには至っていない。普及のための要因が不明である。

企業のシステムは企業の統廃合や事業再編、そして市場競争の激化といった目まぐるしく変わる市場環境に対して、柔軟に対応しなければならなくなってきている。しかし、今日の企業の情報システムとは必ずしもビジネスと整合性がとれているとはいえない。それはひとつの企業の中ですら異なったハードウェアやソフトウェアなどが乱立する複雑な IT 環境となってしまっているためで、さらに加えて相次ぐ企業合併・再編により、企業同士のシステムを統合しなければならないケースが増えてきているためである。これらのシステムは、企業全体のビジネスモデルから見ればもはや整合性が取れているとは言えず、ビジネス環境の変化に対応してすばやくシステムに変更を加えることは困難な状況である。システムの変更が必要な度にシステムを新規に作っていてはコストも時間も追いつかない。その中で、2000 年ごろから「複雑化する IT 環境をいかに統合するか」という課題に対する解決策として、注目されてきたのが SOA(サービス指向アーキテクチャ)という考え方である [39]。

SOA とは一言で言うと「サービス」を組み合わせることでアプリケーションを構成するシステム構築の考え方である。このような異機種分散環境における、ソフトウェアの統合技術の考え方は以前も存在したが、インターネットの普及やオープンスタンダードの技術が発達したことにより、再び SOA のような考え方が注目されるようになってきた。この SOA と流通 BMS は親和性が高いと考えられるが、具体的な分散システム構築の手順や技術的要因が公表されていないので細部は不明な状況である。

本研究では次世代の電子商取引標準として使われ始めている流通 BMS 対応のシステムを SOA を用いて構築する際に発生する問題を分析する。

- 流通 BMS により定義される XML スキーマを技術的、論理的側面から評価する。
- XML データベースを利用して、流通 BMS 対応システムを構築する際に起こる問題を技術的側面から分析する。
- SOA により流通 BMS 対応システムを構築する時にサービスをどの粒度にすればよいかを提案する。
- 流通 BMS で想定される一般的なプロセスを実装したシステムを試作し、さらに、サプライチェーン全体の在庫情報を共有できる機能を実現するシステムの設計を行う。
- 設計したシステムを試作し評価を行う。

これらの分析によって、本論文では流通 BMS の普及の妨げとなっている技術的な要因を明らかにする。さらに SOA というアーキテクチャによって分散システムとして流通 BMS を用いる情報システムを構築する際の基本的手順や要因を明らかにする。

2. 流通ビジネスメッセージ標準(流通 BMS)

2.1 現状の電子商取引の抱える問題

電子商取引(EDI)

電子商取引(以下、EDI)とは企業間の取引を行う際に、伝票を電子化したデータのフォーマットを使用する仕組みのことであり、現在はほとんどすべての企業間取引で用いられている。

EDI を利用することで得られる利点

EDI を使用することで次の利点があげられる。

・コンピュータによる業務の自動化により、人為的なミスを減らすことができる。

・データを電子化することで需要予測、売れ筋の分析などが簡単に行える。

・電子帳簿の保存が法的に認められるためペーパーレス化ができる。
・EDI データに印紙を取り付ける必要がないため印紙代が節約できる。

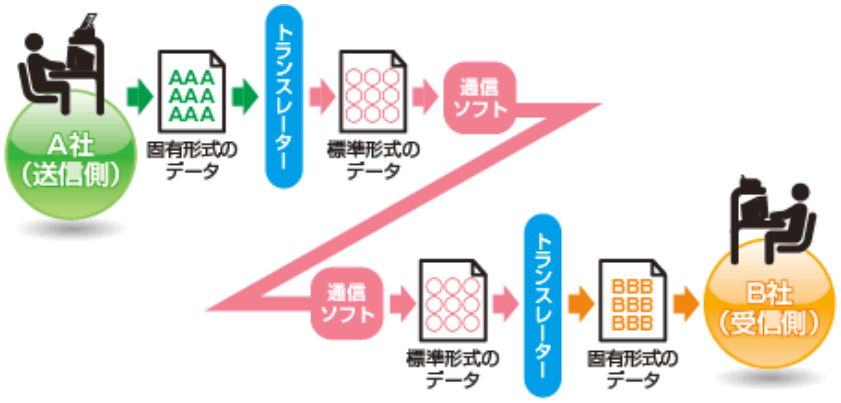

図 0-1　EDI 利用の仕組み
(JEDIC – EDI とは　http://jedic.ecom.jp/edi/about.html#h1 より引用)

EDI 利用手順

A 社(送信側)が B 社(受信側)と取引をする際(図 2-1)、

1. A 社は A 社固有の形式のデータをトランスレータを使うことで A-B 社間で取り決めた標準形式のデータにする。
2. A 社は B 社との間に敷いた専用回線を使い標準形式のデータを送信する。
3. B 社は A 社から受信した標準形式のデータを自社のトランスレータを用いて B 社固有形式のデータに変換して B 社の企業情報システムで扱う。

現状の EDI の抱える問題

しかし、企業間取引で EDI がより使われるようになり、従来の EDI を行うために様々な問題が起きている。(図 2-2)

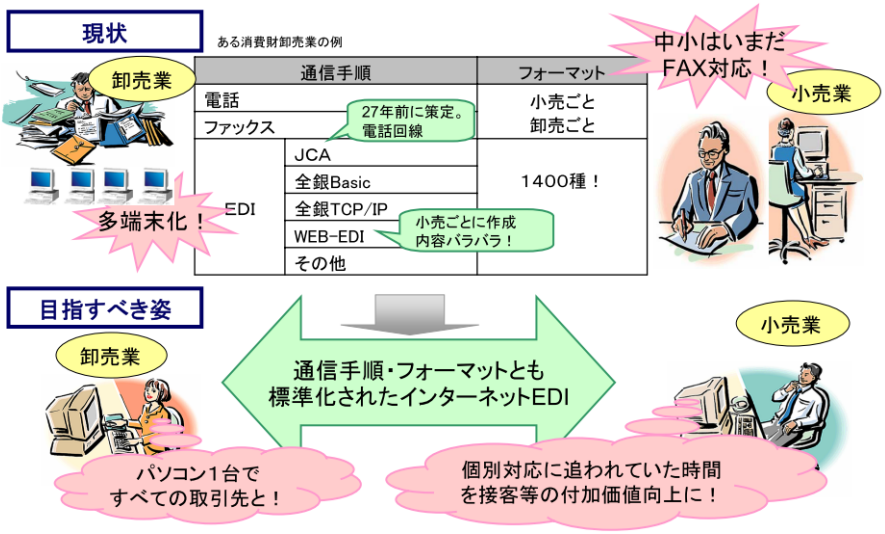

図 0-2　現行 EDI の抱える問題
(平成 19 年度流通システム標準化事業 導入ガイドライン p5 より引用)

・ EDI 導入のための開発コストが高い。

・ 従来の EDI 手順の機器の保守が高コストであり、さらに製造メーカーが販売を中止した。
・ 漢字・画像データの送受信ができない。
・ 大量のデータを送受信するには回線速度が遅すぎる。
・ 標準が大量にあるため、取引先ごとに標準を決めなければならない。
・ データ長が固定されており、新たな項目追加など拡張ができない。
・ 専用回線を使っているため安価な通信回線(インターネット回線など)が使用できない。
・ 統一フォーマットは受発注のみであり、それ以降の基本的な業務プロセスである請求・支払いなどは各社で仕様を作らなければならない。

したがって、現状の EDI が抱える問題を解決するために通信手順、フォーマットを標準化する範囲を広げる EDI を新たに策定することで、パソコン 1 台ですべての取引先と取引ができるようになる。よって、個別対応に追われていた時間を削減することでコスト削減を狙うことができる。このような背景から次節で説明する流通ビジネスメッセージ標準が策定された。

2.2 流通ビジネスメッセージ標準

流通ビジネスメッセージ標準（以下、流通 BMS）は、消費財流通業界で唯一の標準となることを目標に策定している、メッセージ（電子取引文書）と通信プロトコル/セキュリティに関する EDI 標準仕様のことである。本節では流通 BMS について説明する。

2.2.1 流通 BMS の策定経緯

流通 BMS は現状の EDI システムの問題を解決すべく経済産業省主導のもと策定された次世代の EDI 標準である。(表 2-1)

表 0-1 流通 BMS 策定経緯

平成 15～17 年度
事業名：流通サプライチェーン全体最適化促進事業(略称：流通 SCM 事業)
内容：目標として商品マスタデータの同期化と EDI の標準化を掲げ、基礎調査研究、標準モデルの作成、EDI の実証実験を実施。
平成 18～20 年度
事業名：流通システム標準化事業
内容：流通 SCM 事業の検討成果を着実に実運用に移行させるべく、商品マスタデータ同期化と次世代標準 EDI（流通 BMS）の仕様検討と共同実証を実施。その結果、双方とも 18 年度までに標準仕様を確定し、19 年度から実運用開始。
平成 21 年度以降
事業名：流通システム標準普及推進協議会
内容：さまざまな業界独自で流通 BMS の検討・策定が進められることで生じるメッセージの統一性を守るために流通 BMS の維持管理、および普及推進。

また、流通システム標準化事業によると、EDI 標準化の目的と検討経緯は以下の 5 点である[24]。

1. 個別仕様の発生を抑える
2. 現行業務の担保を図る(現行システムの担保ではない)
   ・現行の業務が回ることを念頭に置き、現行メッセージ項目を基にデータ項目を整理し、同一意味・機能のものを集めて項目を一本化している。
3. 将来の技術・業務に対応できる準備を盛り込む
   ・国際標準の EDI メッセージで使われている XML を使うことにより、利用の拡大と時代の変化に対応できる柔軟性、汎用性を持った標準 EDI メッセージを実現している。さらに、

国際取引の標準商品コード(GTIN)と企業／事業所コード(GLN)に対応させるため、発注の商品コードにGTINをEDI上の企業／事業所を表すコードにGLNを使用することも想定している。

4. インターネットを使用した通信を前提とする。

・専用回線ではなくインターネット回線を用いることでコスト削減、通信速度の向上を図っている。

5. 伝票レスモデルの確立

・メッセージ内容だけでなく、法制度面の条件をクリアするための要件も併せて検討したため、電子帳簿保存の対象となる取引データを保存し、発注時に小売が付番した発注No.をキーとして出荷・受領が行われ、最終的には受領データ上で、受注～出荷～受領の履歴(数量、金額、日付)が確認でき、各社社内会計帳簿との相互追跡ができるようにすることを前提としている

2.2.2 流通BMSの技術要素

流通BMSでは通信インフラの基盤としてインターネットを使用し､TCP/IPを用いることを定めている。また、通信手順としてはebXML/MS、AS2、JX手順などに準拠している。次に、EDIメッセージとしてはデータ表現形式としてXML形式を採用している。これは流通BMSがインターネット通信を前提としていることと、データを可変長とするためである(図2-3)。流通BMSではEDIメッセージをXML形式で定義している。これをSecondGenEDI XML Schemaといい、2.4節で説明する。また、コードとして将来に対応するために国際的な標準である、企業の識別コードとして共通企業識別コード(GTN)、商品の識別コードとして(GTIN)を採用している。特に、流通BMSでは、現行業務の担保を図り、個別仕様の発生を抑えるために、取引業務プロセスとデータ項目についての標準化がおこなわれている。データ項目は　取引業務プロセスは8業務、14メッセージが標準化されている。

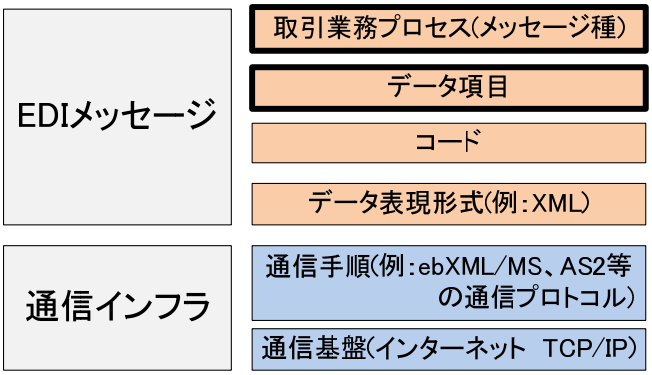

図 0-3　流通BMSの技術基盤

(平成19年度流通システム標準化事業導入ガイドライン p7を改変)

2.2.3 従来のEDIとの技術的比較

流通BMSによる通信では従来のEDIと比較して次の特長がある。(図2-4)

・　フォーマットの変換プログラムが1つで済む。

従来の通信手順では自社内で扱うデータを取引先ごとに定めていたフォーマット変換プログラムを使用していた。

・　標準フォーマットはXMLスキーマに基づいている。

データの形式も取引先ごとに決められていたが、XML 形式でデータを定義することでデータを取り扱いやすくなった。

・　専用のインターネット回線を使用する。

保守コストの高い専用モデムおよび、回線速度の遅い専用回線(公衆電話線程度)ではなく、専用のインターネット回線を使用することで保守コストの削減および回線速度の上昇が可能になった。

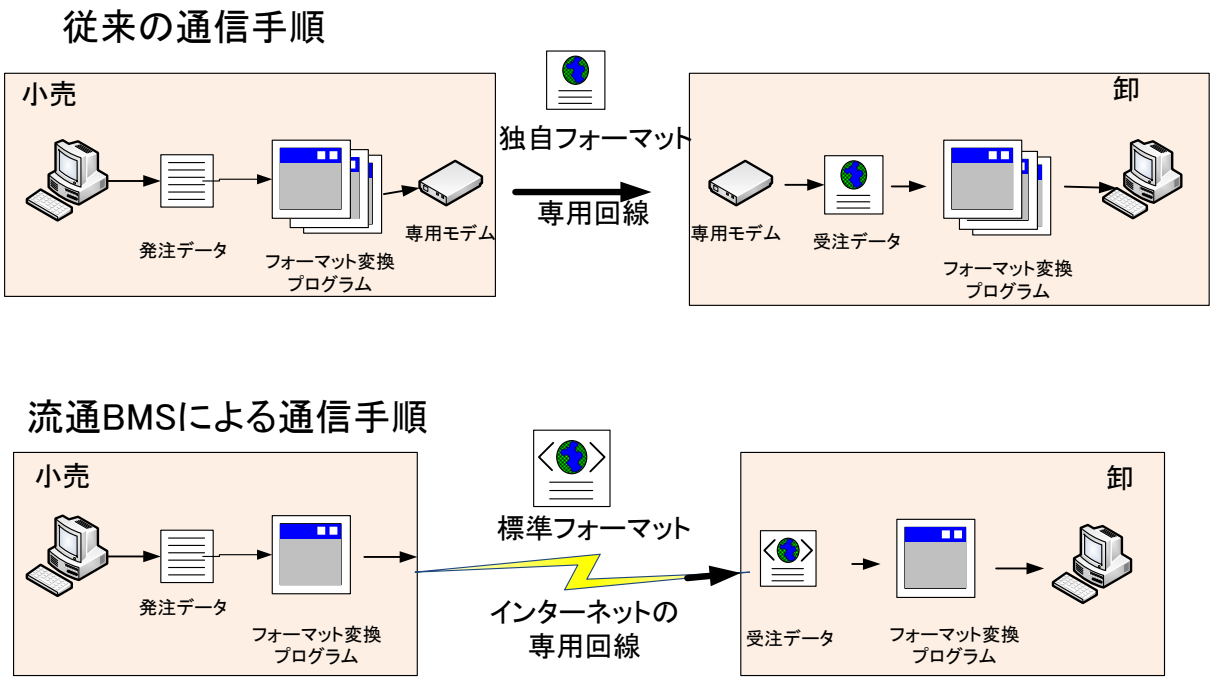

図 0-4　従来の通信手順との比較

表 2-0　流通 BMS と従来の EDI 標準との比較

	全銀/JCA	全銀TCP/IP	ebXML/MS	SOAP-RPC	AS2
通信環境	BSC point to point モデム販売停止	TCP/IP Point to Point	インターネット	インターネット	インターネット
通信速度	2400bps	9600bps～ 64kbps	10Mbps～ 100Mbps	10Mbps～ 100Mbps	10Mbps～ 100Mbps
伝送可能なデータ	キャラクタ(JCA: 漢字不可)	主にキャラクタ(バイナリ可)	全てのデータ可	全てのデータ可	全てのデータ可
Push/Pull	Push/Pull	Push/Pull	Push	Push/Pull	Push
レコード長	JCA:256byte 全銀:2000byte	32Kbyte	無制限	無制限	無制限
添付データ	不可能	不可能	可能	可能	可能
業務規程	JCA:流通 全銀:金融	全銀に順ずる	標準での取り決めは可能	なし	なし

流通BMS

表 2-2 は流通 BMS と従来の EDI 標準との比較をしたものであり、この表では太枠で囲まれた通信手順が流通 BMS による通信である。

2.3　流通 BMS が想定するビジネスプロセス

流通 BMS では基本業務プロセスとして「受発注」から「支払」までの基本的なプロセスを 8 業務 14 メッセージ(発注、出荷、出荷梱包(紐付けあり)、出荷梱包(紐付けなし)、受領、返品、請求、支払、値札、集計表作成データ、生鮮発注、生鮮出荷、生鮮受領、生鮮返品)で標準化している。

基本的な業務のデータの流れとしては

前提条件：小売と卸・メーカーはお互いに商品マスタを登録する。
　　小売は流通 BMS によって定められた値札メッセージを卸・メーカーに送り卸・メーカーは値札を作成する。

1. 小売は発注をするために伝票に相当する発注メッセージを卸・メーカーに送信する。
2. 卸・メーカーは出荷準備ができると出荷伝票に相当する出荷メッセージを小売に送信する。
3. 小売は受け取りサインにあたる受領メッセージを卸・メーカーに送信する。
4. 卸・メーカーは請求書にあたる請求メッセージを小売に送信する。
5. 小売は支払伝票に当たる支払いメッセージを卸・メーカーに送信する。

となっており、基本的な発注から支払いまでの業務プロセスに従っている。(図 2-5)
図 2-5 ではメッセージ部分が塗られたものが標準化されているメッセージである。このプロセスはアパレル業界対象のものであり、生鮮業界になると発注、出荷、受領、返品がそれぞれアパレルと異なったメッセージを使うことになっている。

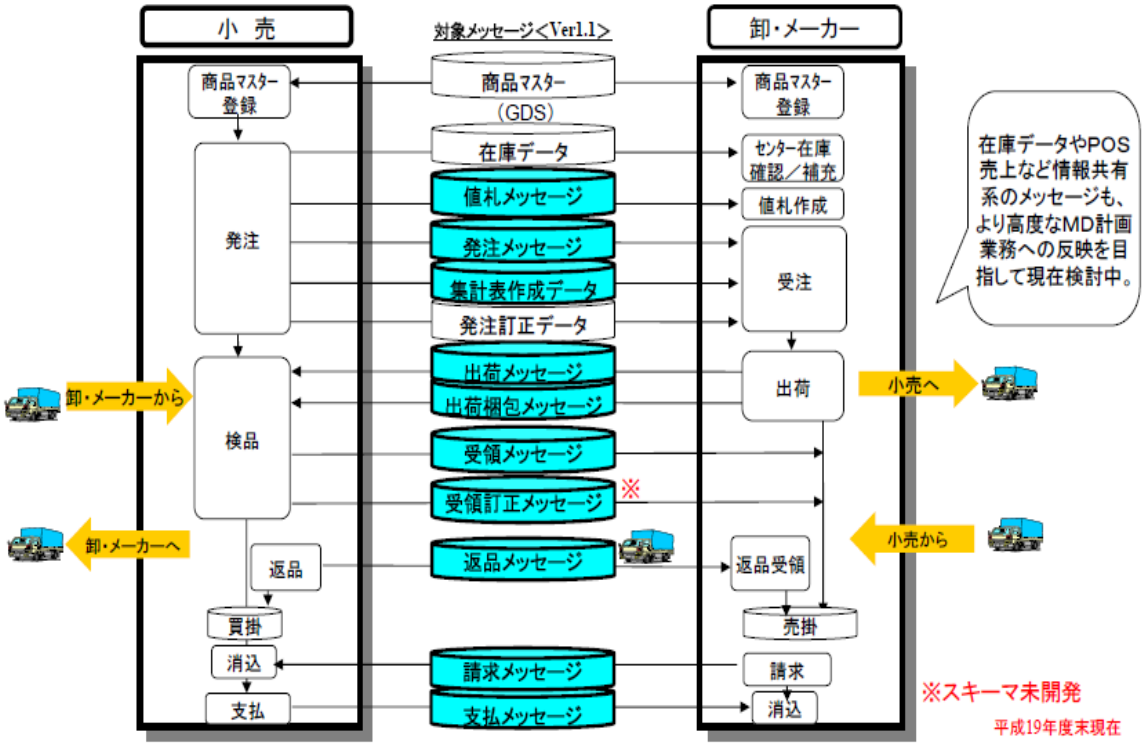

図 0-5　流通 BMS の基本業務プロセス

(平成 19 年度流通システム標準化事業 導入ガイドライン p12 から引用)

2.4　SecondGenEDI Schema

流通 BMS では XML によって定義した EDI メッセージを SecondGenEDI XML スキーマとい

う。

2.4.1 XML(Extensible Markup Language)

XML とは文書やデータの意味、構造を記述するマークアップ言語のひとつである。XML はタグと呼ばれる特定の文字列を使うことで文書に情報の意味、構造を埋め込むことができる。また、XML ではユーザーが独自にタグを指定することができる。この XML を使用することで独自の意味や構造をもつマークアップ言語を作成することにより、ソフトウェア間の通信・情報交換に使用されるデータ形式などに使われる。

XML を用いて文書を作成するには、文書内でのタグや属性の種類、順番など具体的な構造を定義する必要がある。この定義をするものをスキーマ言語といい代表的なものに DTD や XML Schema がある。このようなスキーマ言語に従って作成される XML 文書を XML インスタンスという。

2.4.2 流通 BMS に XML を用いる利点

流通 BMS の EDI メッセージを XML で定義する利点として次のものがあげられる。

- XML はインターネット上で配布・受信・処理出来ることを目的として設計されたためインターネットとの親和性が高い。
- XML は標準化されており、特定のベンダー・プラットフォームに依存しないため、独立性が高く、誰でもが利用できる。
- 独自にタグを定義することが可能なため、対象となるデータモデルに変更が生じた場合も対応がしやすい。また、データ構造を XML Schema を用いて定義することにより、やり取りされるデータが構造的に正しいかの妥当性検証を行うことができる。

表 0-2 メッセージの種類とフォルダ名

メッセージ名	フォルダ名	内容
発注	Order	発注伝票に相当
出荷伝票	Shipment Notification	発注形式に応じた出荷伝票に相当
出荷梱包 (紐付けあり)	Package Shipment Notification	出荷梱包に対応した出荷情報
出荷梱包 (紐付けなし)	Non- associatedPackageShipment	梱包と発注行と関連付けなし
受領	Receiving Notification	発注伝票に応じた受領情報 取引の証憑として保存
返品	Return Notification	受領後の訂正に使用
請求	Invoice	請求書に相当
支払	Payment	請求への回答および、請求のない 計上払いに使用する
値札	Price Tag	値札印字情報
生鮮発注	Fresh Order	発注メッセージを生鮮用に拡張
生鮮返品	Fresh Return Notification	返品メッセージを生鮮用に拡張
生鮮出荷	Fresh Shipment Notification	出荷伝票メッセージを生鮮用に拡張
生鮮受領	Fresh Receiveing Notification	受領メッセージを生鮮用に拡張
集計表作成	Picking List	発注情報のピッキングリスト情報

2.4.3 SecondGenEDI XML スキーマ

流通 BMS の業務メッセージ構造を XML Schema によって定義したものを SecondGenEDI XML スキーマという。この SecondGenEDI XML スキーマは経済産業省流通システム標準化事

業 Web サイト(http://www.dsri.jp/)から利用申請用紙を入手し、申請することで利用することができる。2009/12/15 日現在利用できるスキーマの種類は 5 種類(基本形 Ver1.1 アパレル対応、基本形 Ver1.1 生鮮対応、基本形 Ver1.2、生鮮版 Ver1.2、百貨店 Ver1.0)ある。本研究では基本形 Ver1.1 アパレル対応を使用する。SecondGenEDI XML スキーマの各メッセージとスキーマファイルが格納されているフォルダ名は表 2-3 のようになる。

2.4.3 メッセージ構造

SecondGenEDI XMLSchema において定義されるメッセージは SBDH、メッセージ層、業務データ層の 3 つに分かれている。そして、SBDH・メッセージ層が業務に依存しない部分としてすべてのメッセージで共通に使われ、業務に依存するデータは業務データ層に記述される。(図 2-6)

以下、SBDH、メッセージ層、業務データ層の説明は流通ビジネスメッセージ標準 SecondGenEDI XML スキーマ XML テクニカルガイド p6-p7 より引用

SBDH

SecondGenEDI XML メッセージでは、メッセージヘッダーに国際連合(UN)の機関である、Centre for Trade Facilitation and Electronic Business(CEFACT)が提供している Standard Business Document Header (SBDH)を採用している。これは、流通業界の国際標準である GS1 の Business Message Standards(BMS)でも、同様のものが使われており、将来のグローバル化を目指す現われである。SBDH はメッセージの送受信に必要な情報を記述する部分であり、実際のメッセージの送受信で用いられる伝送プロトコル(AS2、ebXML MS V2、 JX)レベルの情報を組立てるために必要な情報が記述される。XML 文書内では <sh:StandardBusinessDocumentHeader>というタグで示した部分がこれにあたる。

図 0-6 SecondGenEDI XMLSchema の構造

メッセージ層

SBDH の下位要素にある、<common:message>のタグで示した部分をメッセージ層とよぶ。Ver1.0 で提供する 8 つすべてのメッセージに共通で、業務に依存しないデータを保持する部分として「メッセージ層」を用意してある。ここには、業務データに含まれる取引の数のように、メッセージの整合性をチェックするためにシステムに与える情報を保持する。

業務データ層

業務に依存するデータは、メッセージ層の下位要素として保持され、8 つのメッセージでそれぞれ異なるタグを用いて記述される。

例えば、受発注業務で使用される発注メッセージの構造は図 2-7 のようになり、業務データ層は発注ヘッダーの中に取引ヘッダー、取引ヘッダーの中に取引明細という入れ子構造になっている。なお、他のメッセージについても業務データ層はヘッダーと明細の形になっている。また、一般的な伝票と発注メッセージとの関連は図 2-8 のようになる。

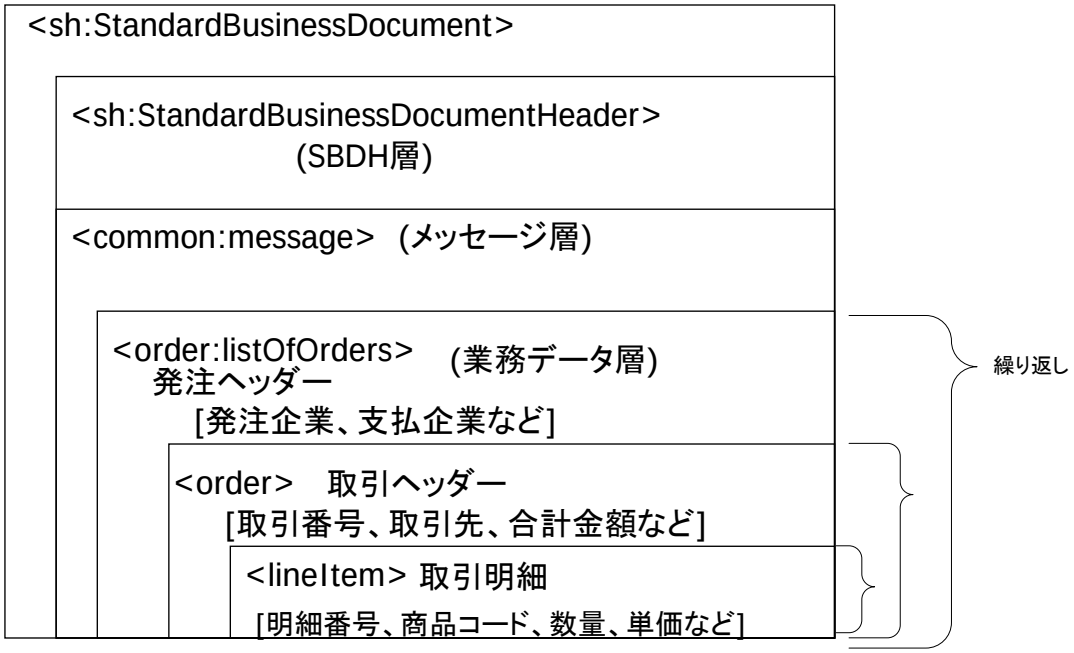

図 0-7　発注メッセージの構造

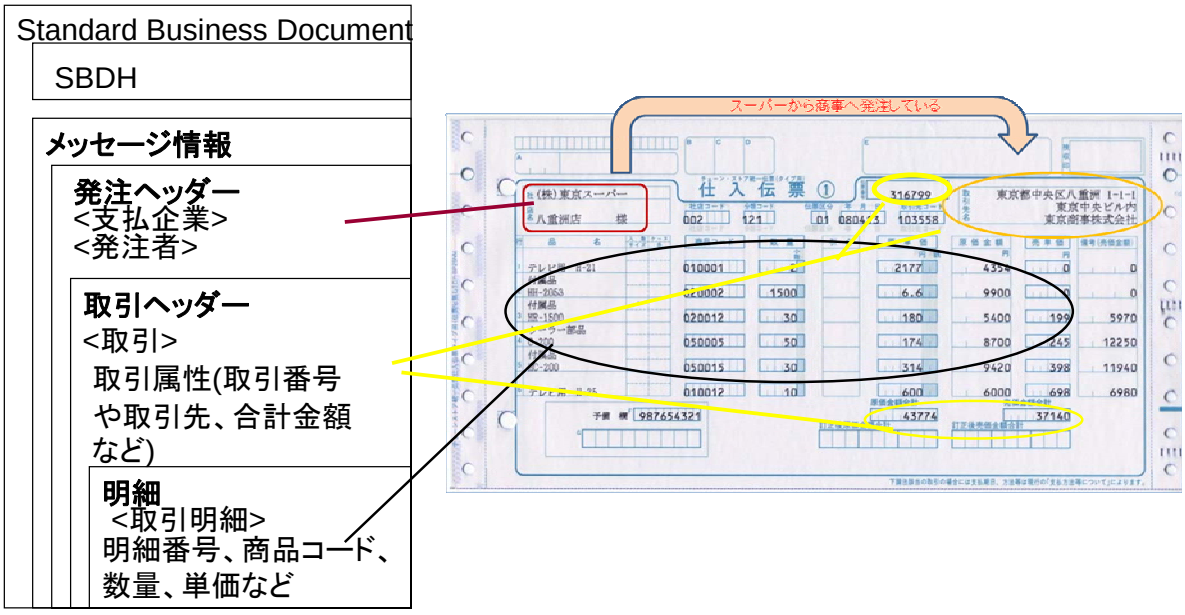

図 0-8　伝票との相関例

(伝票画像は http://www.gdx.co.jp/denpatu/type-samp-1.jpg 改変)

3.　流通 BMS システム実現に生じる技術的問題

流通 BMS は SOA と親和性が高いと考えられるが、具体的な分散システム構築の手順や技術的要因が公表されていないので細部は不明な状況である。

流通 BMS では、1 つのメッセージにつき、約 200 項目のデータがある。そのデータを定義するために約 70 個のスキーマファイルを扱うことになる。XML インスタンスを XML スキーマを用いることで検証をすることができる。しかし、SecondGenEDI XML スキーマをそのまま使用するだけでは妥当性検証などをすることができない。本章では SecondGenEDI XML スキーマを利用できるようにするために必要なことを述べる。

3.1　SecondGenEDI XML スキーマによる妥当性検証

3.1.2　XML スキーマを使った妥当性検証

XML インスタンスが定義にしたがっているかどうかをあらかじめ DBMS などに登録してある XML スキーマを用いることで検証することができる。このことを XML スキーマを使った妥当性検証という。このとき検証する際に使われるスキーマをルートスキーマという。また、検証にはルートスキーマとスキーマ内の最上位要素であるルート要素の 2 つを指定して行う。

妥当性検証を行うことで検証できる項目は

- データ構造：タグ名の正誤、必須要素、不要な要素、要素の出現数、要素の出現順序
- データ属性：入力可能データ文字、データのサイズ・桁数、データの範囲
- コードリスト：あらかじめ定義されたコードかどうか

であり、これらの検証項目の 1 つでも誤っていると妥当性検証でエラーが発生する。

発注スキーマ OrderType.xsdの一部

```
1  …
2  <xsd:complexType name="OrderTradeTotalQuantityType">
3     <xsd:sequence>
4       <xsd:element minOccurs="0" name="itemTotal" type="common:Quantity_6"/>
5       <xsd:element minOccurs="0" name="unitTotal" type="common:Quantity_6"/>
6       </xsd:sequence>
7   </xsd:complexType>
…
```

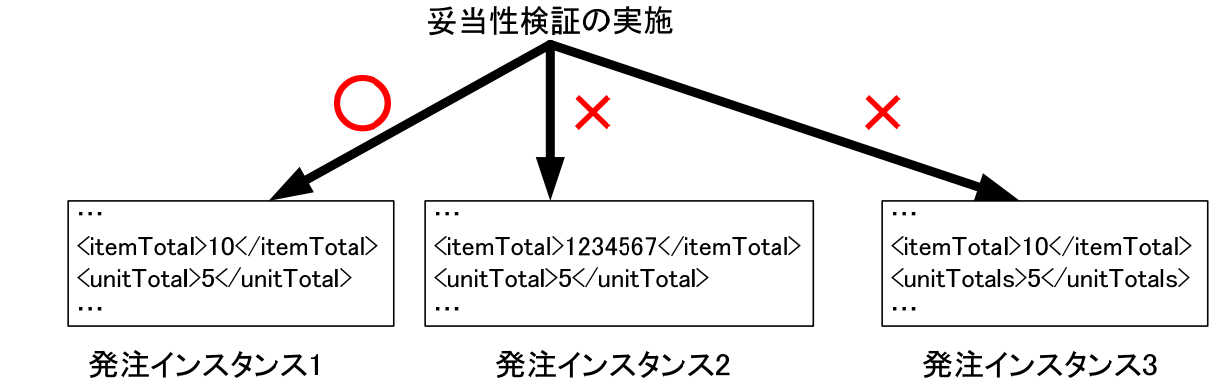

図 0-1　XML スキーマを使った妥当性検証

図 3-1 は OrderType.xsd の一部である、<itemTotal>と<unitTotal>の定義部分である。

2 行目以降では“OrderTradeTotalQuantityType”の要素、属性、出現回数、構造を定義している。要素名は”itemTotal”と”unitTotal”であり、最低出現回数は 0、桁数が他のスキーマで定義した最高桁数 6 の”common:Quantity_6”に従うことを定義している。
発注インスタンス 1、2、3 に妥当性検証を行うと発注インスタンス 1 は OK である。発注インスタンス 2 は<itemTotal>の桁数が 7 なので NG、発注インスタンス 3 は要素名が<itemTotals>と最後に”s”が入っているため要素名が違うため NG となる。

図 3-2 は実際に、IBM DB2 9.5 にて妥当性検証をした実行例である。

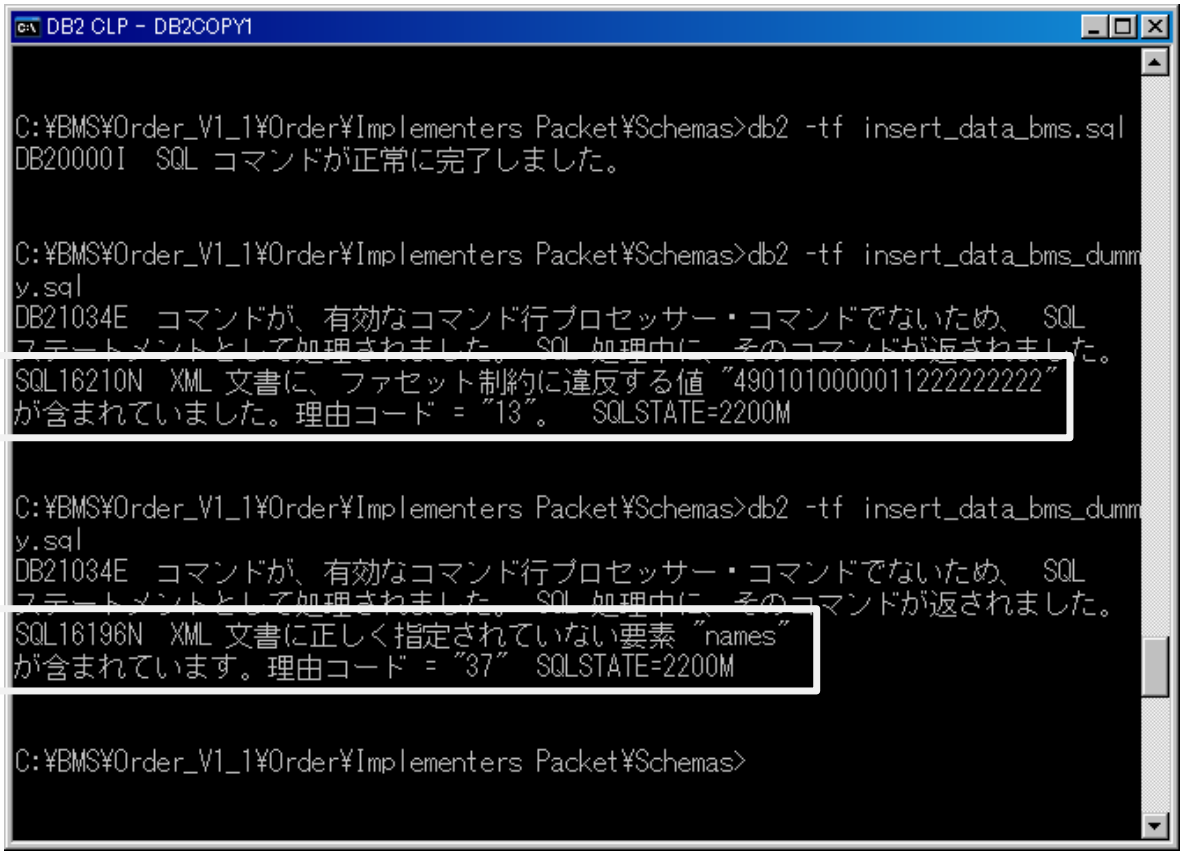

図 0-2　妥当性検証の実行例

3.1.2　2 つのルートスキーマからなる SecondGenEDI XML スキーマ

SecondGenEDI XML スキーマでは各メッセージには 2 つのルートスキーマが存在している。例えば、発注メッセージでは SBDH 層を定義する StandardBusinessDocumentHeader.xsd とメッセージ層、業務データ層を定義する ListOfOrders.xsd である。これら 2 つのスキーマファイルは参照関係がない。そのため、スキーマによる妥当性検証を行う際に、どちらをルートスキーマにすればよいか問題が生じる。(図 3-3)

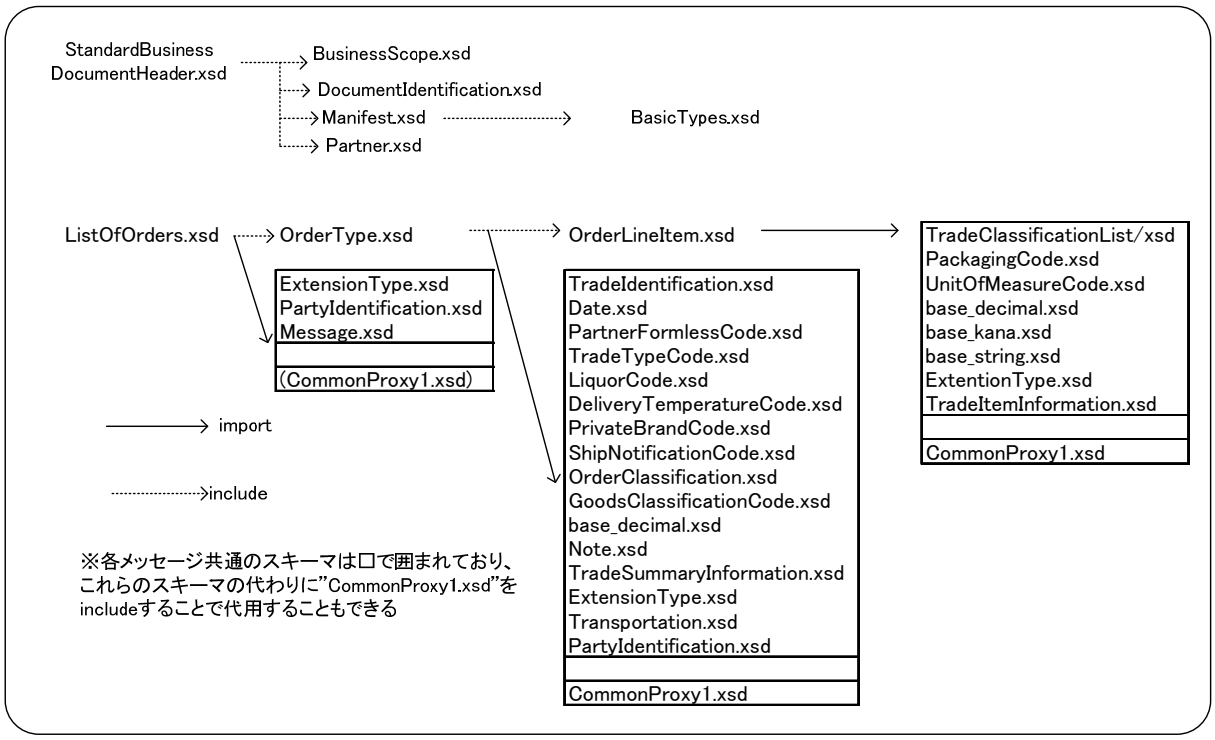

図 0-3　発注メッセージを構成するスキーマファイルのファイル構造

各メッセージ共通スキーマ内での参照関係を表したのが以下の図 3-4 である。例えば ListOfOrders.xsd が include している Message.xsd スキーマファイルは base_string.xsd、base_decimal.xsd、PartyIdentification.xsd スキーマファイルを include している。

解決案

・ 2 つの独立したスキーマを統合するようなスキーマを作成する。(図 3-5)

1. 2 つのルートスキーマを参照するようなあらたなルートスキーマを作成する。その際、問題となるのはその統合したスキーマに新たな名前空間や要素が必要となることである。この場合、SecondGenEDI XML スキーマで定められた標準と大きく異なってしまう恐れがある。
2. SBDH 層のルートスキーマである"StandardBusinessDocumentHeader.xsd"をルート要素とし、このスキーマファイルからメッセージ層と業務データ層のルートスキーマである"ListOfOrders.xsd"を import すればよい。この場合、"StandardBusinessDocumentHeader.xsd" にメッセージ層と業務データ層で使用される名前空間を参照させるなどの変更をする必要がある。
3. スキーマを 1 つずつ妥当性検証に使う

この方法では一旦、"StandardBusinessDocumentHeader.xsd"を使って妥当性検証を行った後に、"ListOfOrders.xsd"を使って妥当性検証を行う。この場合妥当性検証を行うための構文が 2 重になる。ただし、Oracle10g ではこの方法は使えない。

6 節のシステムの実装ではスキーマの統合は解決案 2 を使う。

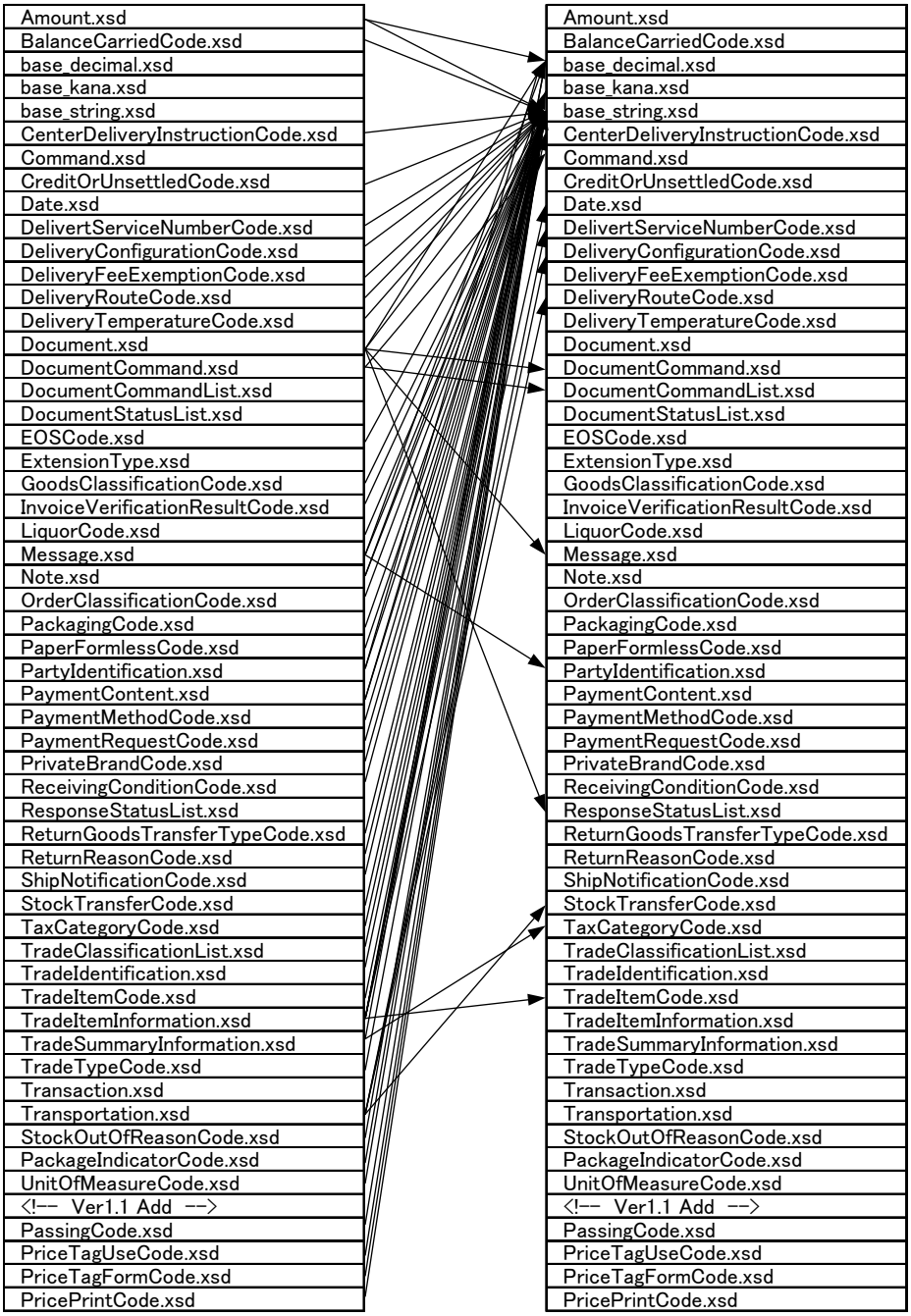

図 0-4　メッセージ共通スキーマ内の include 関係

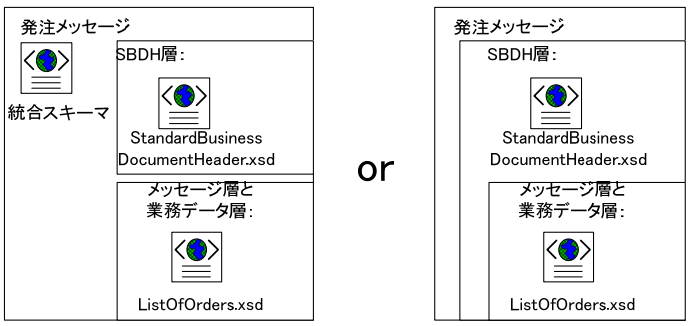

図 0-5　統合した発注用 XML スキーマの案

3.2 Schema ロケーションの問題

3.2.1 名前空間とスキーマロケーション

XML では名前空間(NameSpace)を利用することで、別の複数のスキーマを参照し 1 つのスキーマを作成することができる。なお、別のスキーマファイルを参照するには名前空間”http://www.w3.org/2001/XMLSchema”を参照しなければならない。

例:発注メッセージ用スキーマ(ListOfOrders.xsd)の場合

```
<?xml version="1.0" encoding="UTF-8"?>
  <xsd:schema       xmlns:common="urn:SecondGenEDI:common:Japan:1"
                    xmlns:order="urn:SecondGenEDI:order:Japan:1"
                    targetNamespace="urn:SecondGenEDI:order:Japan:1"
                    version="1.0"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    attributeFormDefault="unqualified"
                    elementFormDefault="unqualified">
  <xsd:import   namespace="urn:SecondGenEDI:common:Japan:1"
                  schemaLocation="../CommonProxy1.xsd"/>
  <xsd:include schemaLocation="OrderType.xsd"/>
                                                  .
                                                  .
                                                  .
</xsd:schema>
```

1 行目は XML のバージョン情報と文字のエンコードを指定している。

2 行目は”xmlns:common=”は” urn:SecondGenEDI:common:Japan:1”の名前空間で定義された全メッセージ共通の属性、構造をこのスキーマ内で使えるようにするものであり、”xmlns:order=”では発注メッセージ用の属性、構造定義をこのスキーマで使えるようにしている。”targetNamespace=”ではこのスキーマファイルが定義された内容はどの名前空間に属するようにするかを指定するものである。

9,10,11 行目では使用する名前空間を読み込む操作を表している。import とは異なる名前空間で定義された内容を参照する際に使用するものである。include は同一名前空間の内容を参照する際に使われる。図 3-6 では名前空間 A に属するスキーマ a1 はスキーマ a2,b1 で定義された属性や要素を使用したい場合、a2 と b1 の名前空間に注目する。a2 は a1 と同じ名前空間に属しているため include を使用する。一方、b1 は名前空間 B に属しているため、a1 が参照するには import によって参照しなければならない。

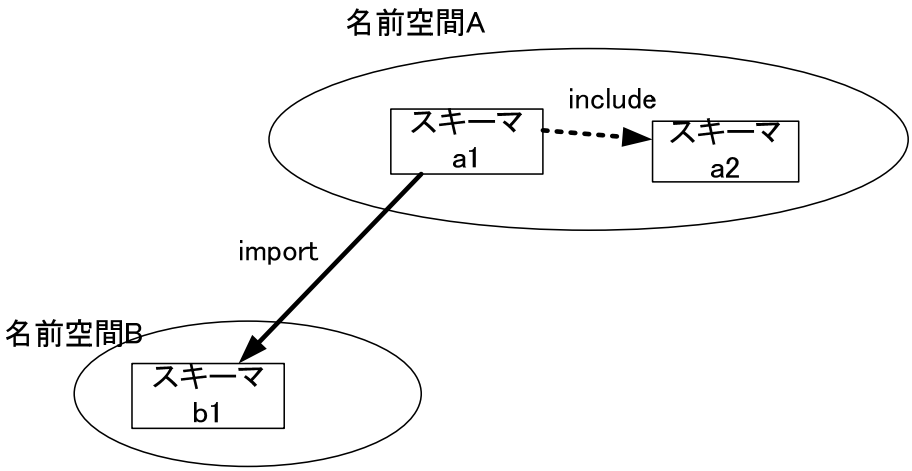

図 0-6　スキーマの import と include の違い

また、参照するスキーマファイルは”schemaLocation=”により場所を一意に指定しなければならない。(図 3-7) 通常スキーマファイルはあらかじめデータベースなどに登録しておく必要がある。shemaLocation の決め方はファイルのディスク上の位置に関係なく、各自で指定する。

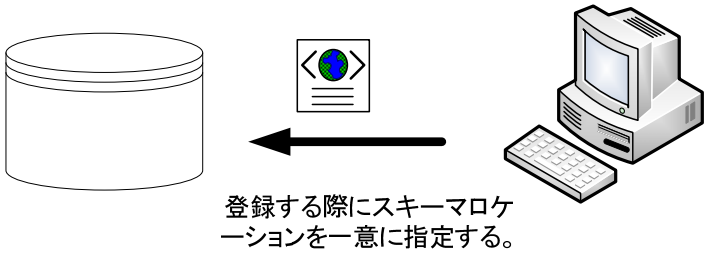

図 0-7　スキーマロケーションの指定

SecondGenEDI XML スキーマの共通スキーマと業務依存スキーマは、以下のルールに従って名前空間が決められている。

urn:SecondGenEDI:業務カテゴリ:|産業区分:|Japan:バージョン

業務カテゴリ：メッセージの業務カテゴリを示す名前であり、発注、返品などに使用される order、出荷・受領に使用される deliver、請求・支払いに使用される pay、各メッセージ共通の common、拡張を表す extend がある。

産業区分：現在の流通 BMS では省略されているが、将来的に対応される商材拡大用の領域。

バージョン：メッセージスキーマのメジャーバージョン。現行では 1 が入る。

たとえば発注メッセージ用の名前空間を使用する場合は以下のようになる。

urn:SecondGenEDI:order:Japan:1

3.2.2　スキーマロケーションの変更

SecondGenEDI XML スキーマではデフォルトのままスキーマファイルを取り扱うとエラーが発生する。それはスキーマロケーションが問題だからである。以下に、発注メッセージ用スキーマファイルのディレクトリ構造を示す。(図 3-8、表 3-1)

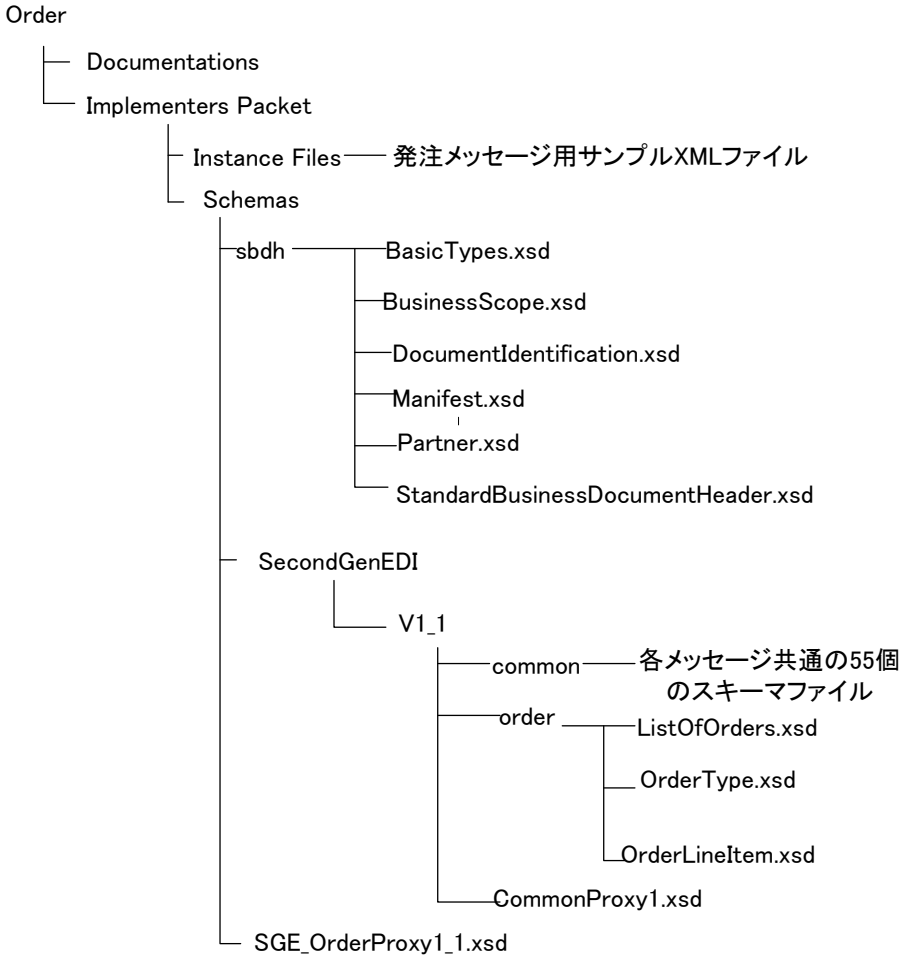

図 0-8　発注メッセージのディレクトリ構造

表 0-1　各ディレクトリとファイル説明

ディレクトリ名	説明
Schemas	各メッセージに使用するスキーマファイルが入っている
sbdh	StandardBusinessDocumentHeader情報に使用されるスキーマファイルが入ってい
SecondGenEDI	業務データ(伝票に相当)を定義するためのスキーマファイルある。
V1_1	バージョン情報
common	各メッセージ共通の55個のスキーマファイルがある。
order	発注特有の業務データ作成のためのスキーマファイルがある
ファイル名	説明
SGE_OrderProxy1_1.xsd	発注業務ではどのスキーマファイルがルートになるかを示すファイル。
CommonProxy1.xsd	全メッセージ共通で使われるスキーマファイルをすべてincludeしたもの

CommonProxy1.xsd ファイルは全メッセージ共通で使われる属性、要素の定義ファイルをすべ

て include したものである。つまり、common ディレクトリ以下にあるスキーマファイルを include しているため、この CommonProxy1.xsd ファイルを import することで各メッセージ共通の要素・属性定義が扱える。(図 3-9)

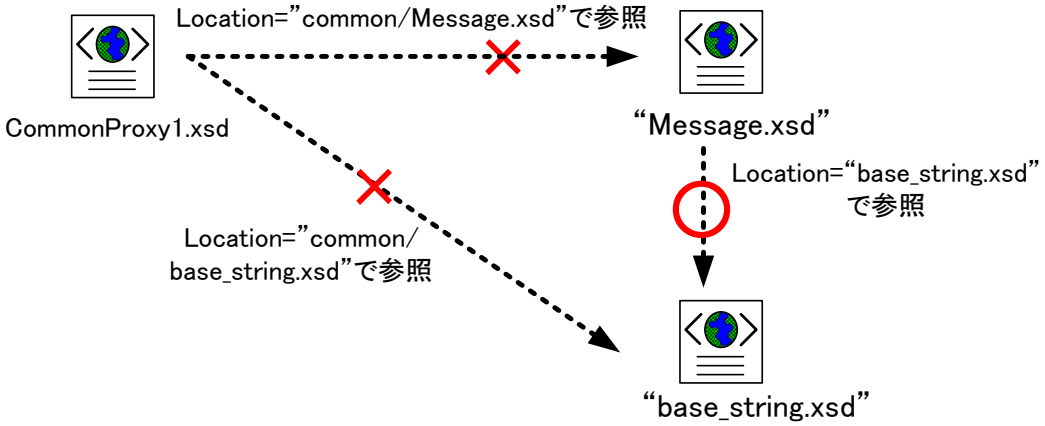

図 0-9　スキーマロケーションの問題例

流通 BMS ではデフォルトのファイルにおいてスキーマロケーションはディレクトリ構造に従って記述されている。そのため、CommonProxy1.xsd ファイル内では各メッセージ共通のスキーマファイルの SchemaLocation は" common/共通スキーマファイル" となっている。ところが、common フォルダ以下の各メッセージ共通のスキーマ内にも共通スキーマファイルを参照するスキーマがある。たとえば図 のように base_string.xsd はデータベースに登録する際のロケーションを"base_string.xsd"とした場合、Message.xsd からは参照できるが、CommonProxy1.xsd からは参照できない。この問題を解決するには以下の方法がある。

1. "common/base_string.xsd" でスキーマロケーションを指定するのではなく、"base_string.xsd"でスキーマロケーションを指定する。
   XML スキーマをデータベースに登録する際には、全メッセージ共通のスキーマはそのまま"base_string.xsd"のようにファイル名のみで登録する。そして CommonProxy1.xsd ファイル内のスキーマロケーションを自分で訂正することでこの問題は解決できる。
2. 同じ内容のスキーマファイルを 2 つ登録する
   スキーマロケーション Message.xsd、CommonProxy1.xsd 両ファイルから参照される base_string.xsd を "common/base_string.xsd" と"base_string.xsd"の両方のロケーションにスキーマファイルを配置する

6 節のシステムの実装ではスキーマロケーションの問題は解決案 1 の方法を使う。

3.3　XMLDB で実現する際に生じる問題

3.3.1　XML データベース

XML データベース(XMLDB)とは XML 文書をそのままの形式で格納することができるデータベースである。クエリー言語として XQuery、XML 文書の構造を別の形式に変換するための変換ルールを記述したものを XSLT、XML 文書の中の特定の要素を指し示す記述方法を定めた規格として XPath がある。つまり、XMLDB においてあるデータを参照するには XQuery 言語により XPath を指定することで参照することができる。(図 3-10)

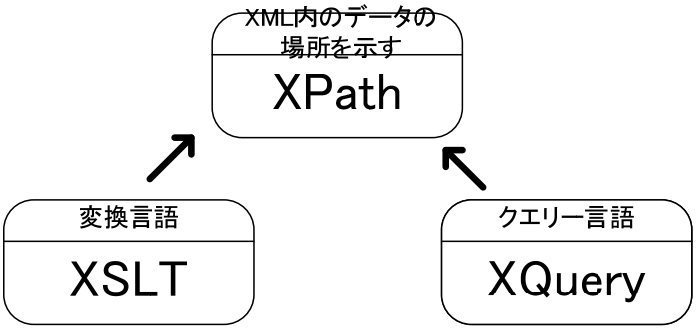

図 0-10　XMLDB における言語

XMLDB は格納形式を構造化記憶域を使う方法と非構造化記憶域を使う方法がある。構造化記憶域を使うと XML は、スペース情報は失ってしまうが、タグの構造や構造が同一であることが保証されるため、SQL 問い合わせのパフォーマンスの向上、要素の部分更新が高速になる。一方、非構造化記憶域を使う方法は XML 文書をそのまま単純なキャラクタ・ストリングとして保持するため、スペース情報などは失われないが、XPath 操作に際し、XML データをメモリー上にツリー構造で展開するため、構造の複雑な XML 文書を大量に処理する場合には向かない。

3.3.2　Oracle XMLDB におけるインスタンスサイズの問題

実装環境に Oracle10g を用いると XML 格納領域のサイズの問題が生じてしまう。Oracle では 1 つの XML テーブルにつき 4000 文字までしか格納できない。一方、SecondGenEDI XML スキーマで定義された業務インスタンスは通常、4000 文字以上あるため、業務メッセージをそのまま挿入することができない。

解決策

- 通常、XML インスタンスは XML Type 表に VARCHAR2 型で挿入されるため、それを CLOB 型で挿入するように変更する。ただし、CLOB 型にすると XMLDB の構造化領域を使った格納ができなくなり、SQL 問い合わせのパフォーマンスが低下してしまう。

4.　サービス指向アーキテクチャ(SOA)

本章では流通 BMS と親和性が高いと考えられる、複雑化する IT 環境に対応するシステム構築のための設計思想である SOA について説明する。

4.1　SOA とサービス

SOA とは Service Oriented Architecture の略であり、サービス指向アーキテクチャと訳される。SOA は環境の変化に柔軟に対応しながらビジネスと IT との整合性を持ちかつ、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。ここでいう「サービス」とは、ビジネスプロセスや業務の単位を考慮し、ある程度意味のある形でアプリケーションをコンポーネント化したである。また、これらのアプリケーションは外部から呼び出せるよう稼働している必要がある。[2][40]

4.1.1 SOA の特長

SOA を用いて構築されたシステムの特長として以下のものが上げられる。
(http://www.ibm.com/developerworks/jp/websphere/library/soa/soa_intro/1.html より引用)

- アプリケーションが業務処理などの単位でサービス化されていること
- オープンで標準的なインターフェースでサービスが定義され、呼び出すことが可能であること
- サービスを組み合わせてアプリケーションを構築すること

4.1.2 SOA とサービスの要件

SOA によるシステムのプロセスはサービスを順番に呼び出すことによって実現される。そのため、SOA とサービスには以下のような要件が求められる。

SOA は

- ビジネスの視点から理解可能なビューをシステムに与える。
- ビジネスプロセスの改善に有効なフィードバックを与える。
- 標準技術によって、社内外のサービスを組み合わせて実現することを可能にする。
- ビジネスプロセス実行の基盤を提供し、実行状況のモニタリングを可能にする。
- 企業に求められる内部統制を支援し、また内部統制に対応しなければならない。

サービスは

- ビジネスモデルと対応付けられている。
- 組み合わせて利用可能である。
- 再利用可能である。
- サービス同士は、高い疎結合性を持っている。
- 利用のための条件(インターフェースや契約)が明示的に定義されている。

4.1.3 SOA によるシステム開発のメリット

SOA による開発 の目的は IT システムをいかに柔軟に素早く対応させるかという点にある。SOA によって個々のアプリケーションを柔軟に組み合わせることが可能になるため、システムも市場の変化に迅速に対応するメリットがある。さらに、SOA ではサービスは特定のインフラに依存しないためベンダーやプラットホームを自由に選ぶことができる。また、IT 面から見た場合、各サービスの実装はサービス利用者にはわからない。そのため、利用者は各アプリケーションの内部構造を気にせずに自分たちの業務範囲にのみ集中して開発が可能となる。各アプリケーション開発側はサービスのインターフェースのみを考慮するだけでよい。また、各アプリケーションは他システムとは疎結合なので開発しやすい。さらにサービスの単位は従来のオブジェクト指向で作成されるコンポーネントよりも粒度が大きいため、サービスは再利用がしやすい。よって、時間も費用も削減できることから開発生産性の向上があげられる。

4.1.4 SOA を用いた開発プロセス

SOA に基づいてシステムを開発していくプロセスは次の 4 ステップに分けられる[42]。

1. モデリング

   モデリングではビジネスプロセスのモデル化を行う。現状のビジネスプロセス、および改善策として提案されたビジネスプロセスのフローをビジネスプロセス図でモデル化する。なおこのモデル化の際には BPMN(ビジネスプロセス表記法) を用いる。

2. プロセスデザイン

   プロセスデザインのステップでは BPEL プロセスを作成していく。この際、プロセスデザ

イナーと呼ばれる GUI ベースの開発ツールを使うことで、モデリング段階で最適化されたモデルを基にプロセスデザインを自動的に生成することができる。プロセスデザイナーによる開発は、パレットから必要なタイプのアクティビティを選択し、それを結合していくという作業になる。アクティビティをカスタマイズがなければ、ほとんどプログラミングをしなくてよい。プログラムのコードはプロセスデザイナーにより作成される。このステップで BPEL ソースが作成される。

3. デプロイ

デプロイのステップでは完成した BPEL プロセスをプロセスエンジンなどの BPEL 実行環境を持つコンポーネントにデプロイする。デプロイとは作成したプロセスを実行環境側でいつでも実行可能な状態にすることである。プロセスのインスタンスが実行されたら、それを定義済みのアクティビティシーケンスに渡すという処理を行いプロセスが実行される。プロセスが終了したらプロセスエンジンはどのような処理が行われていたのかを記録するためのイベントを生成する。

4. 管理・監視

管理ステップでは過去に実行されたインスタンスの管理を行うことでパフォーマンスを測定し、分析、改善していく。デプロイ、管理ができるツールとして、Oracle 社 の SOA Suite をはじめ、IBM 社の Websphere、BEA 社の SOA Resourse、VITRIA 社の BusinessWare など多数のベンダーが多様な製品を発表している。本研究では Oracle 社の Oracle SOA Suite を用いてプロセスのデプロイ、管理を行っていく。

4.2 SOA を構成する技術要素と三層構造

SOA を用いたシステム開発では実現するプロセスを「ビジネスプロセス層」、「サービス層」、「ソフトウェアコンポーネント層」の 3 層に分けて考える。「ビジネスプロセス層」「サービス層」のサービスが利用可能であるために外部からの要求を受け付けることができる部分である「サービスインターフェース」、このサービスをインターフェースを通して外部から呼出し可能にする要素の「サービスインターフェース定義(WSDL)」、ビジネスプロセスを複数のサービスを組立・構造化して業務フローを作成するときに使用される「ビジネスプロセス(BPEL)」がある。(図 4-1)

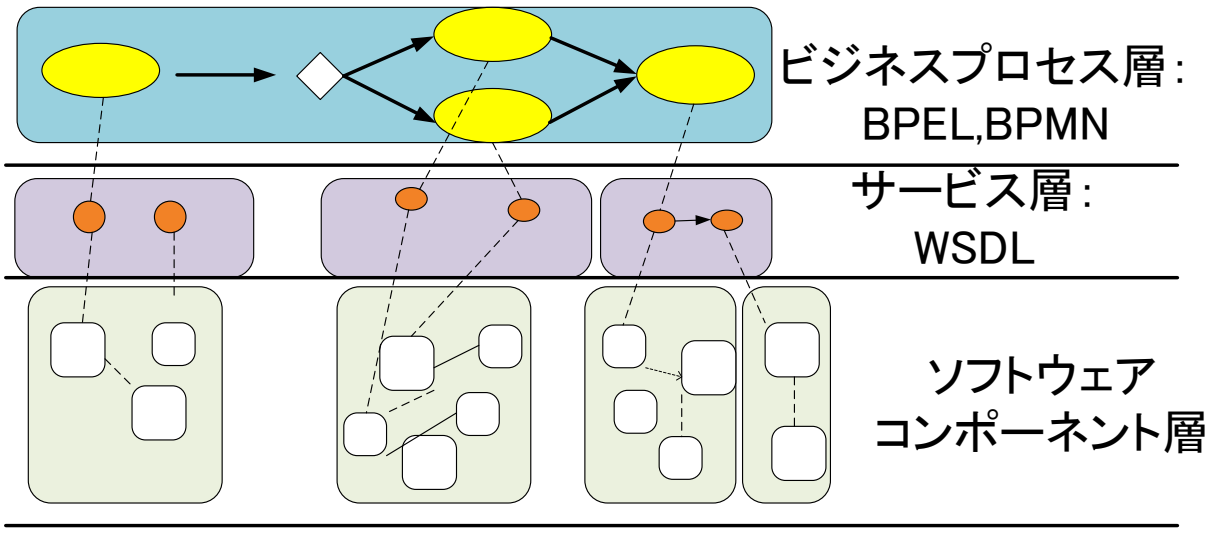

図 0-1 SOA の構造

4.2.1 ビジネスプロセス層

ビジネスプロセス層ではどのサービスをどのような順番で実行することでプロセスを実行するかを定義する層である。ビジネスプロセスを定義する図を表記する BPMN とビジネスプロセスを実現する BPEL を用いる。

4.2.2 サービス層

サービス層は SOA の肝となるサービスを実行するための順番とソフトウェアコンポーネントとのインターフェースを定義する層である。サービス層の実現には WSDL を用いる。

4.2.3 ソフトウェアコンポーネント層

ソフトウェアコンポーネント層ではデータ変換やデータベース挿入など実行単位毎にコンポーネント化されたソフトウェアなどを実行する。SOA によるシステム開発ではこの各コンポーネントとのインターフェースまで定義されるが、ソフトウェアコンポーネントの具体的な設計はしない。

4.3 Business Process Execution Language (BPEL)

BPEL とは Business Process Execution Language for Web Services の略であり、ビジネスプロセスを定義し、実行する XML ベースのプログラミング言語である。サービスとして記述された個々のビジネスロジックをビジネスプロセスとして統合する役割を果たしている。IBM、Microsoft、BEA によって誕生したが、現在は OASIS で標準化が行われている。BPEL は、IBM の WSFL と Microsoft の XLANG を基にしている[2]。

4.3.1 BPEL の特長

ビジネスプロセスを実行する BPEL の特長として以下のものが上げられる。

- ソフトウェアでコンポーネントを Web サービスとして受理することができる。また、自身も外部に Web サービスとして公開できる。そのため自身の BPEL プロセス内に外部の BPEL プロセスである Web サービスを組み込んでシステムを構築することができる。
- Web サービスを組み立て、構造化することができる言語なので、1 つのビジネスプロセスを単独のプロセスという単位で記述できる。
- XML により標準規格化されている。

BPEL はソフトウェアコンポーネント同士の構造を定義して、Web サービスを構造化させることで 1 つのシステムに組み立てることができる。BPEL 処理系はこのような BPEL による記述を実行して外部の Web サービスと呼出・応答するための処理系である。また、BPEL とその他のプロセス記述言語の違いは、操作、メッセージ、設定の全てが XML と XML スキーマ言語で記述できることである。XML による標準規格化が行われたことで、実装が容易になるだけでなく、ベンダー間での相互運用性が高くなる。

4.3.2 BPEL によるプロセス定義

BPEL 仕様は既存の Web サービス標準のビジネスプロセス拡張とされている。Web サービスは処理状態を把握しないような相互作用に限定されていたが、BPEL とその他のプロセスおよびコレオグラフィ言語は、Web サービス上で状態を管理する機能をもつ対話型のビジネスプロセスを実現する方法を示している。また、BPEL のプロセス定義は 2 種類のファイルからなり、

ソースコードは１つの XML 形式の BPEL ファイルと１つまたは複数の WSDL ファイルで構成される。

WSDL ファイル(.wsdl)

プロセスが必要とする、自らが実装するまたは呼び出す Web サービスのインターフェースを特定するファイル。インターフェースにはパートナーリンクタイプ、プロパティ、ポートタイプ、操作、メッセージ、パートが含まれる。

BPEL ファイル(.bpel)

ビジネスプロセス層のプロセスを実行するサービス層のサービスの順番や組み合わせを定義するファイル。メインアクティビティ、パートナーリンク、相関セット、変数、補償またはフォルトまたはイベント処理ハンドラなどが定義される。

4.3.3 サービス層を定義する BPEL のオブジェクト

BPEL はオブジェクトが組み合わされることで、WSDL とプロセス定義がビジネスプロセスの制御フローを形成し、Web サービスを通じて外部へのインターフェースとなる。フローを動かすプロセス内のメインステップは、サービスの接点である。このサービスの接点は receive、pick、invoke、reply アクティビティで表現される。また、BPEL のプログラムではプロセスは何らかのサービスとして呼び出されることで開始されなければならないという原則がある。つまり、receive または pick アクティビティで開始されなければならない。

BPEL のオブジェクトとして以下のものがある[2]。

・ プロセス(Process)
1 つまたは複数のオブジェクトを含むビジネスプロセスのことである。

・ 変数(Variable)
プロセスまたはスコープの中で使用される変数。変数の型は WSDL メッセージ、XSD 要素、XSD の基本型のどれかで定義される。

・ プロパティ(Property)、プロパティ別名(Property Alias)
プロパティは WSDL メッセージに含まれるデータのトークンであり、プロパティの別名はプロパティの値を取得するための XPath 式である。

・ 相関セット(CorrelationSet)
メッセージのデータをプロセスの対話状態と相関させるために使用される 1 つまたは複数のプロパティ。プロセスもしくはスコープアクティビティは 0 個もしくは 1 個の相関セットを保持することができる。

・ パートナーリンクタイプ(Partner Link Type)
プロセスがどのリンクをサポートし、各リンクに対し自らがどの役割をサポートし、リンク先にどの役割を期待するかをプロセスが宣言する。プロセスは複数のパートナーリンクを保持することができる。

・ パートナー(Partner)
パートナーリンクの集合。プロセスは 0 個以上保持することができる。

・ 補償ハンドラ(Compensation Handler)
キャンセルや巻き戻しロジックを含み、すでに実行されたスコープまたはプロセスを初期状態に戻す必要のある場合に行われるアクティビティ。プロセス、スコープで 0 個または 1 個保持することができる。

・ フォルトハンドラ(Fault Handler)、キャッチ(Catch)、キャッチオール(Catch All)

プロセスまたはスコープにおいてフォルトの型に基づいて例外処理を行うハンドラの集合。0 個または 1 個保持でき、ハンドラ内のキャッチの数に制限はない。

・ イベントハンドラ(EventHandler)、メッセージ受信時処理(onMessage)、警告受信時処理(onAlarm)

プロセスまたはスコープにおいて発生したイベントの処理をイベントの型に基づいて行うハンドラの集合。0 個または 1 個のイベントハンドラを保持できる。ハンドラ内のイベント検出子の数に制限はない。

・ アクティビティ(Activity)

BPEL アクティビティの基底型であり、プロセスまたはスコープは 1 個のアクティビティを保持できる。また、アクティビティは構造化アクティビティとして、その中に詳細要素を含むことができる。BPEL のアクティビティの種類は次のようになる。(表 4-1)

表 0-1　アクティビティの種類

アクティビティ名	説明
受信(Receive)	Web サービスのSOAP メッセージを受信する
呼出(Invoke)	パートナーのWeb サービスを同期または非同期で呼び出す
応答(Reply)	受信したWeb サービスに対して、同期で応答を返す
補償(Compensate)	スコープまたはプロセスの補償を引き起こす
スロー(Throw)	フォルトを生成し、プロセスまたはスコープのフォルトハンドラを起動させる
割り当て(Assign)	ある変数から別の変数へデータをコピーする
待機(Wait)	指定された時間、プロセスを停止させる
空(Empty)	何もしない
分岐(Switch)	XOR 構造で条件が真と評価されるアクティビティを実行
フロー(Flow)	有向グラフのフローに基づく、並列アクティビティの実行構造
直列(Sequence)	アクティビティを直列に実行する
ピック(Pick)	複数のイベントの中から1つだけを待機する。 最初に発生したイベントに従い、アクティビティを実行する
ループ(While)	XPath で定義される式が真である間、ループ内のアクティビティを実行する
スコープ(Scope)	ハンドラ、変数、相関セットを独自に保持する

4.4　Web Services Description Language (WSDL)

WSDL とは Web Service Description Language の略であり、Web サービスを記述するための、XML をベースとした言語仕様。それぞれの Web サービスがどのような機能を持つのか、それを利用するためにはどのような要求をすればいいのか、その Web サービスはどこで稼動しているのかなどを記述する方法が定義されている。W3C(World Wide Web Consortium) で仕様化されており、現在は WSDL1.1 がもっとも使われている[2]。

4.4.1　インターフェースを記述する WSDL

WSDL によって定義される情報は以下のものである。(図 4-2)

・ サービスの名前、操作名

サービスの名前や操作名はサービスの管理、サービスが提供する機能を示すために使用する。

・ サービスに渡すべきメッセージの構造、サービスから返されるメッセージの構造

メッセージの構造については、サービスがシステムとして機能する際の入力と出力を定義する。

・ サービスの呼び出し方

サービスの呼び出し方とは、トランスポートである。SOA ではサービスは実装技術に依存しないように設計されるのでプロトコルを具体的に指定する項目が必要となる。通常、サービスの呼び出しには SOAP(Simple Object Access Protocol)が使われる。

・ サービスを特定する情報

サービスを特定する情報はエンドポイントとも呼ばれ URI 形式で表現される。同一のインターフェースを提供するサービスを複数の提供者がそれぞれ実装して公開している場合に、実行時にどのサービスを利用するのかを特定するのに必要である。

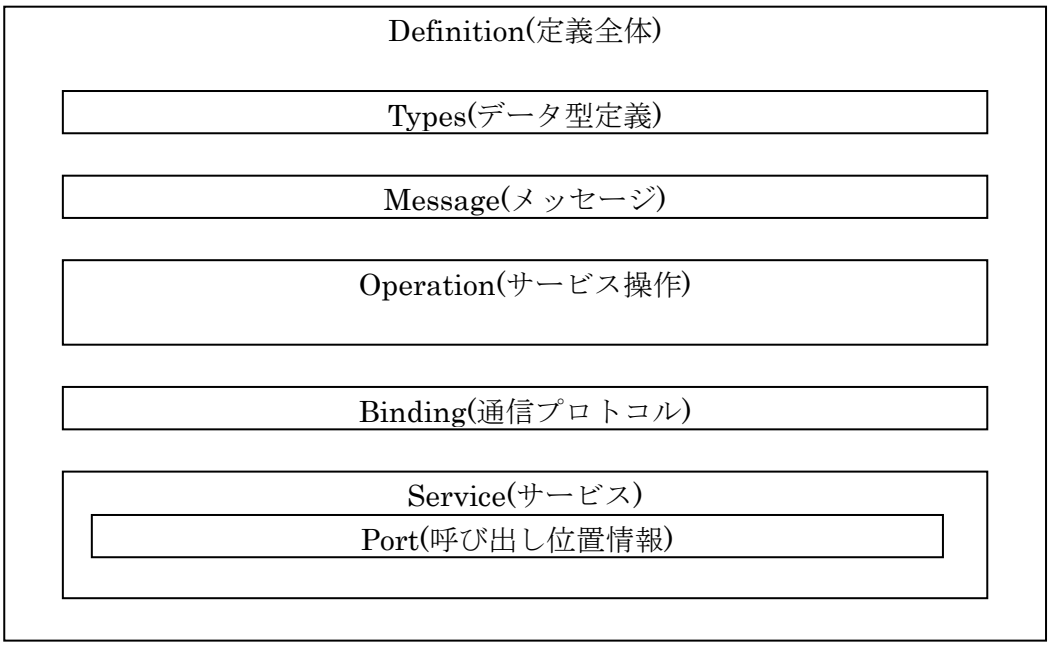

図 0-2　WSDL の構造

4.4.2　Business Process Modeling Notation (BPMN)

BPMN とは Business Process Modeling Notation の略であり、ビジネスプロセスモデリング表記法と訳される。BPMN は視覚的なフローチャート形式の言語であり、ビジネスアナリストや開発者が直感的な形式でビジネスプロセスを表現するビジネスプロセス図を作成できる。このビジネスプロセス図は BPEL が XML でコード化する内容を図で伝えることができる。

4.4.3　BPMN の例

BPMN ではフローオブジェクト、接続オブジェクト、スイムレーンがある[2]。(図 4-3)

- フローオブジェクト
  - イベントはビジネスプロセスを引き起こす出来事であり、プロセス内の段階(開始、中間、終了) で分類され、種類(基本、メッセージ、タイマー、ルール、例外、キャンセル、補償、リンク、複合、終了) によって分類される。開始イベントは細線、中間イベントは二重線、終了イベントは太線のそれぞれ丸で表記されそのなかに種類の表記がされる。
  - アクティビティはプロセス内で作業を実行するステップであり、タスク、サブプロセスの 2 種類がある。タスクは 1 つのアクティビティを実行するが、サブプロセスは階層的でさらに下にタスクまたはサブプロセスを持つことが可能である。- スプリットとジョインでは if-then,switch,all などのような制御構造を表している。排他 OR, 包含 OR, 複合、並列などがある。
- 接続オブジェクト
  - シーケンスフローはプロセス内の制御フローで、アクティビティ、イベント、ゲートウェイをソースからターゲットへと接続する矢印として表記される。

- メーセージフローは送受信を行う二人の別々のプロセス参加者間の、メッセージの流れを表すのに使用される。

・ スイムレーン
- 異なった機能や責任を明確にするために、アクティビティをはっきり区別できるカテゴリにグループ分けするのに使われる。
- プールは、ビジネスプロセス図中で物理的に分離されていて、プロセス中のひとつの参加エリアを表している。
- レーンはプールの中のサブパーテーションであり、アクティビティを編成し、分類するのに使われる。

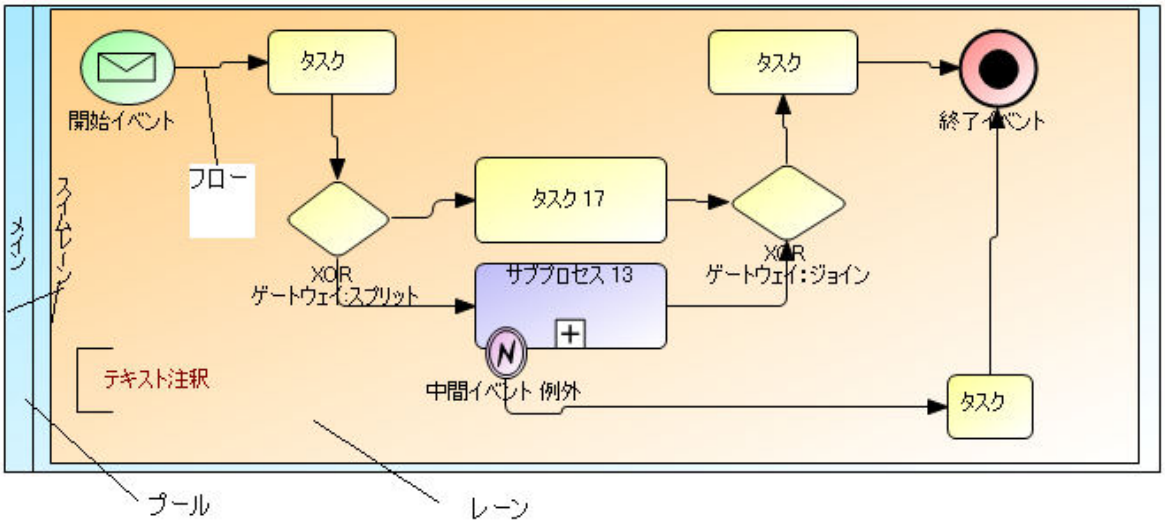

図 0-3　BPMN の例

5. システムの概要

本章では実際に流通 BMS 対応のシステムを SOA を用いて構築するための概要を示す。さらに、従来の流通 BMS 対応システムにはないサプライチェーン間の在庫変動を調べるプロセスをシステムに付加することで新しいシステムを提案する。

5.1　試作する流通 BMS 対応システムの概要

小売業者が取引先である、卸・メーカーとの間に流通 BMS 対応のシステムを SOA に基づいて構築する。構築するシステムは流通 BMS の「発注→出荷→受領→請求→支払」の基本業務プロセスシステムに加えて、小売と卸・メーカー合わせたサプライチェーン間の在庫変動を調べるシステムを実装することで、付加価値のあるシステムを構築する。

5.1.1　流通 BMS で想定される基本業務プロセスシステム

「発注→出荷→受領→請求→支払」の基本業務プロセスを実装する。伝票に相当するものがメッセージであるため、使用するメッセージ種は発注メッセージ、出荷メッセージ、受領メッセージ、請求メッセージ、支払メッセージの 5 種である。(図 5-1)

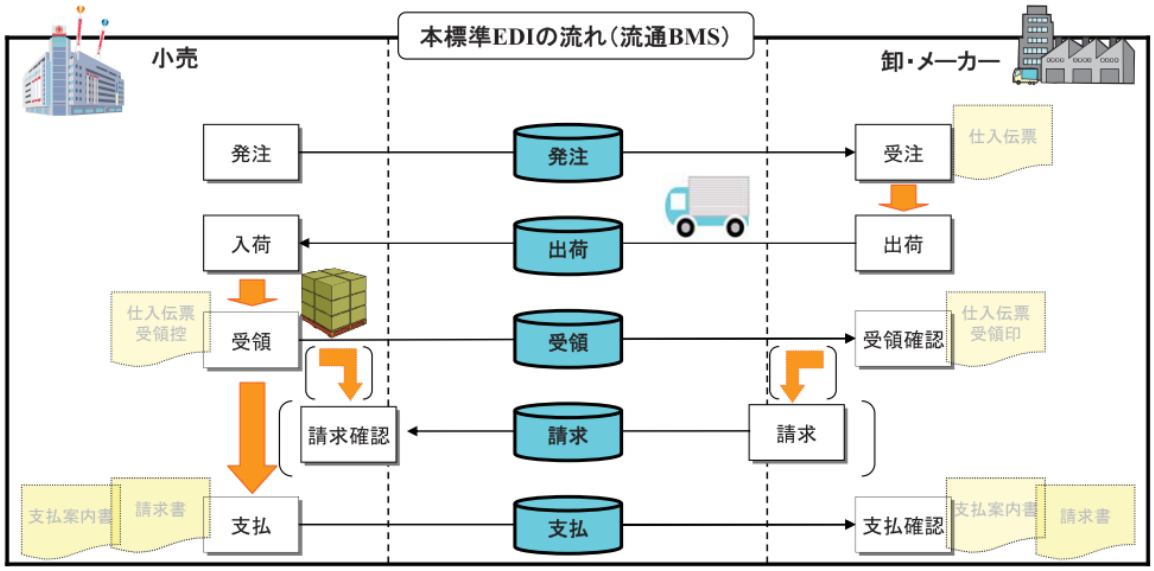

図 0-1　構築する基本業務プロセス

（流通 BMS 導入ガイドライン概要 p12 より引用）

5.1.2　サプライチェーン間の在庫変動を調べるシステム

小売、卸、メーカーそれぞれが倉庫を持っており流通網の在庫が見えづらい。そのため、SOA に基づいて流通網の在庫情報を共有するシステムを流通 BMS 対応システムに付加することでより付加価値のあるシステムとなる。システムの概要は小売は自社の在庫・製造情報、販売予定と卸の在庫・製造情報、さらにメーカーの在庫・製造情報を問い合わせるプロセスとなる。なお、前提としてサプライチェーン間での情報共有に関するセキュリティ面などの問題はクリアしているものとする。(図 5-2)

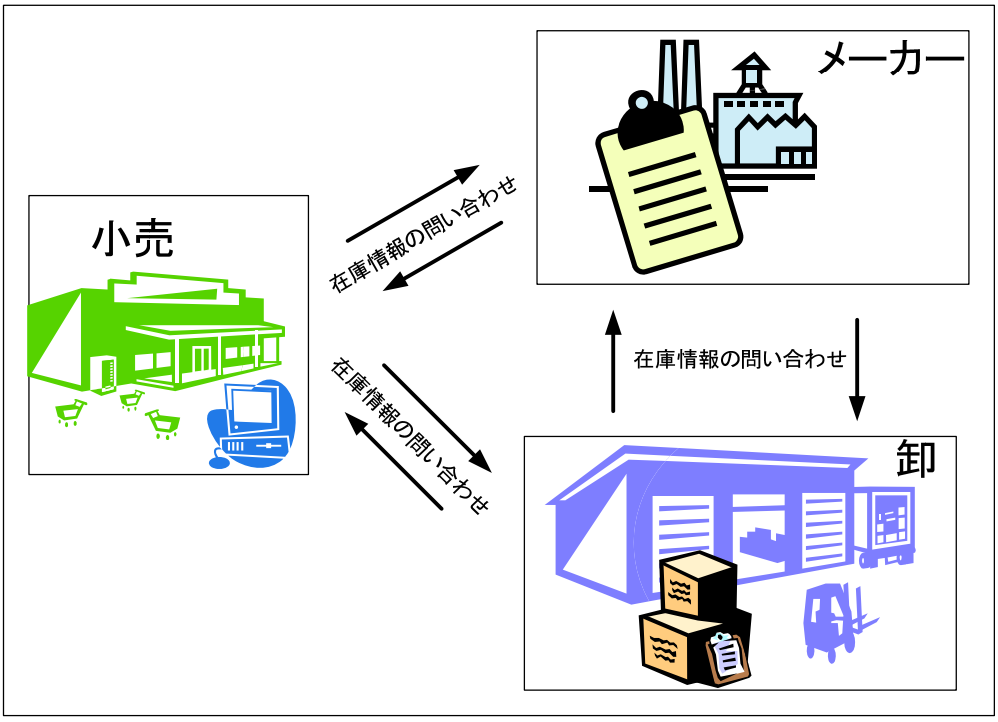

図 0-2　在庫変動を調べるシステムの概要

5.2　流通 BMS のデータモデル

本節では流通 BMS 対応システム構築のために、メッセージを格納するための論理データモデル設計を行う。使用するデータベースは RDB、XMLDB のどちらも想定するため、2 種類のデータモデルを作成する。

データモデルとは、現実の世界を、データを使って記述するための概念と表現方法のことである。具体的にはデータベースとして保存されるデータの集合にデータ間の関係を自然な形で定義し、これに対し検索や更新などの管理を効率的にできるような構造化の仕方などの方法の枠組みである。

5.2.1　Relational database におけるデータモデル

Relational database(RDB)におけるデータモデルではデータモデル化の概念として管理実体型、属性、ドメイン、参照関係、継承関係がある。管理実体型とはデータとして現実世界の存在を表現する場合、個々の存在を実体とよび、意味によって分けた実体の集まりを実体型である。属性とは同じ管理実体型の中に属する実体を区別するための表現である。また、管理実体型の中で、ある属性について同一の値を持つ実体が唯一のとき、その属性を主キー属性という。ドメインとは属性の値として取りうる、現実世界を記述するときに人間にとって意味を持つようなある集合のことである。参照関係および参照関係とは現実社会に存在する制約や条件を、複数の管理実体型からなる集合全体が満たすための一貫性条件のことである。流通 BMS の受発注業務におけるデータモデルを TH データモデルで表すと図 5-3 のようになる。

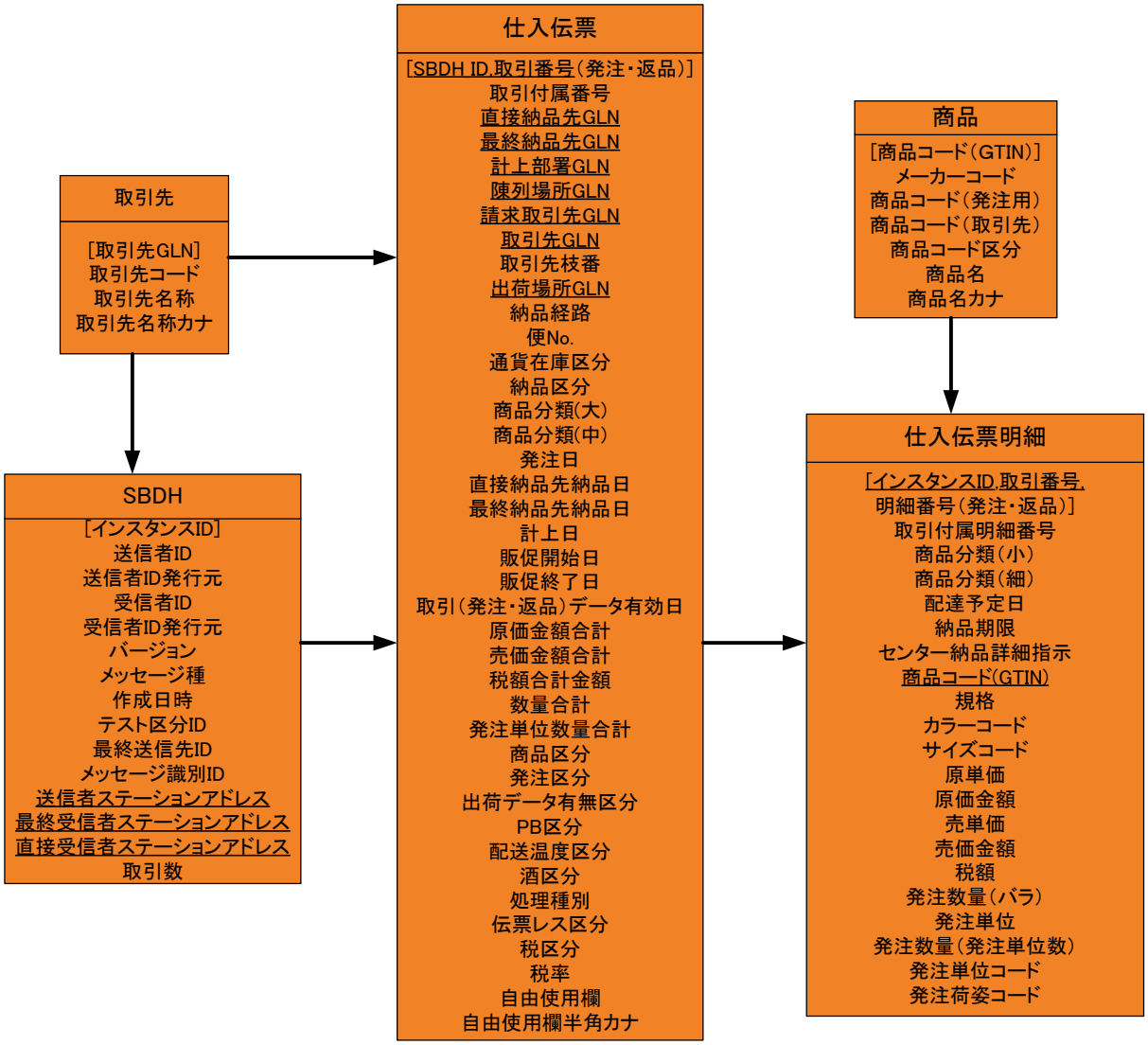

図 0-3　基本業務プロセスのデータモデル

図 5-3 のデータモデルではデータベース作成上、重要でないと考えられる 37 個の管理実体型は省略している。省略した管理実体型は表 5-1 のようになる。これらの省略した管理実体型は列としてコードとその内容しか持たないものがほとんどである。

表 0-1　データモデル表記では省略した管理実体型

EOS区分	請求区分
PB区分	税区分
カラー	センター納品詳細指示区分
計上部署	陳列場所
欠品区分	通過在庫区分
原価	訂正区分
サイズ	伝票レス区分
酒区分	納品区分
支払内容区分	納品経路
支払方法区分	売価
出荷データ有無区分	配送温度区分
出荷荷姿	配送料免除区分
照合結果区分	発注区分
商品移動区分	発注単位
商品規格	発注荷姿
商品区分	便No区分
商品コード区分	返品・値引き理由区分
処理種別区分	未払買掛区分
税額	

5.2.2　XML データモデル

XML データモデル(XDM)は XML スキーマに従って作成することができる。XML データモデルはXMLDBにXMLインスタンスがどのような構造で格納されるかを示すものであり、XMLDBに格納されている XML データの場所を示す XPath を使用する際に使うものである。XDM では RDB のデータモデルのように主キー属性や参照関係を定義することはできないため、XML の構造のみを定義したデータモデルである。(図 5-4)

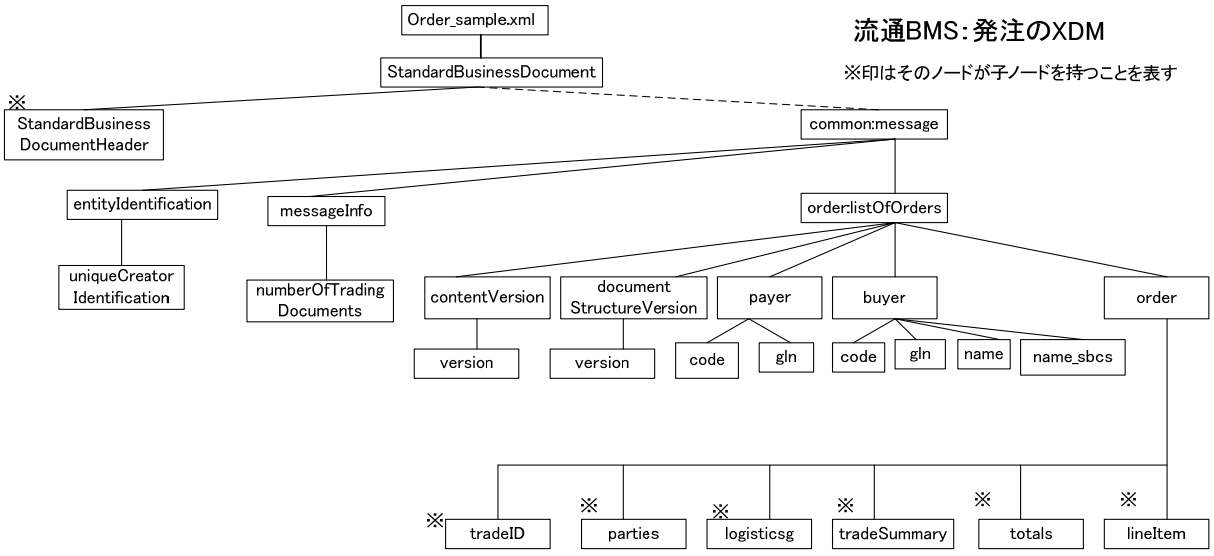

図 0-4　発注メッセージの XDM

図 5-4 ではさらに”StandardBusinessDocumentHeader”、”tradeID”、”parties”、”logisticsg”、"tradeSummary”、”totals”、”lineItem”それぞれがさらに展開することができ、100 種類以上値を格納することになる。”StandardBusinessDocument”と”common:message”が点線なのは”StandardBusinessDocumentHeader”型は直接”common:message”型を参照していないためである。

5.3　流通 BMS の基本業務プロセスの SOA を用いた設計

流通 BMS の基本業務プロセスを SOA を用いて設計するための概要を説明する。

5.3.1　流通 BMS 基本業務プロセスにおけるサービス

試作する流通 BMS 基本業務プロセスは

step1：小売は発注をし、卸・メーカーは受注をする。このプロセスは発注メッセージを使って行われる。
step2：卸・メーカーは出荷をし、小売は入荷をする。このプロセスは出荷メッセージを使って行われる。
　step2-1：在庫が無い場合は卸はメーカーに発注をし、卸とメーカーとの間で基本業務プロセスを行う。
　step2-2：メーカーは卸から商品が出荷されると小売に対し、出荷プロセスを行う。
step3：小売は受領をし、卸・メーカーは受領確認をする。このプロセスは受領メッセージを使って行われる。
step4：卸・メーカーは請求をし、小売は請求確認をする。このプロセスは請求メッセージを使って行われる。
step5：小売は支払をし、卸・メーカーは支払確認をする。このプロセスは支払メッセージを使って行われる。

となっており、一般業務プロセスは発注、出荷、受領、請求、支払の 5 プロセスで構成されている。よって、SOA によるシステム構築ではこれらの発注、出荷、受領、請求、支払がビジネスプロセス層になる。サービス層ではビジネスプロセス層の各プロセスを実現するサービスを定義する。つまり、プロセス”発注”では小売りが発注をし、卸が受注をする流れになる。よって、これを”受発注サービス”とすることができる。(図 5-5)

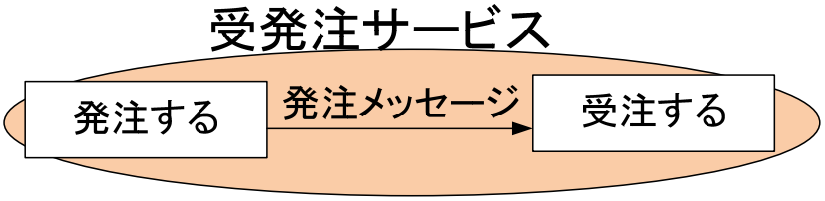

図 0-5　受発注サービス

同様にプロセス”出荷”は”入出荷サービス”、受領は”受領サービス”、プロセス”請求”は”請求サービス”、プロセス”支払”は”支払サービス”をそれぞれ呼び出すように定義する。(表 5-2)

表 0-2　各プロセスとサービス層の定義

ビジネスプロセス	サービス層	使用される流通BMSメッセージ
発注	受発注サービス	発注メッセージ
出荷	入出荷サービス	出荷メッセージ
受領	受領サービス	受領メッセージ
請求	請求サービス	請求メッセージ
支払	支払サービス	支払メッセージ

サービス層をまとめると図 5-6 のようになる。

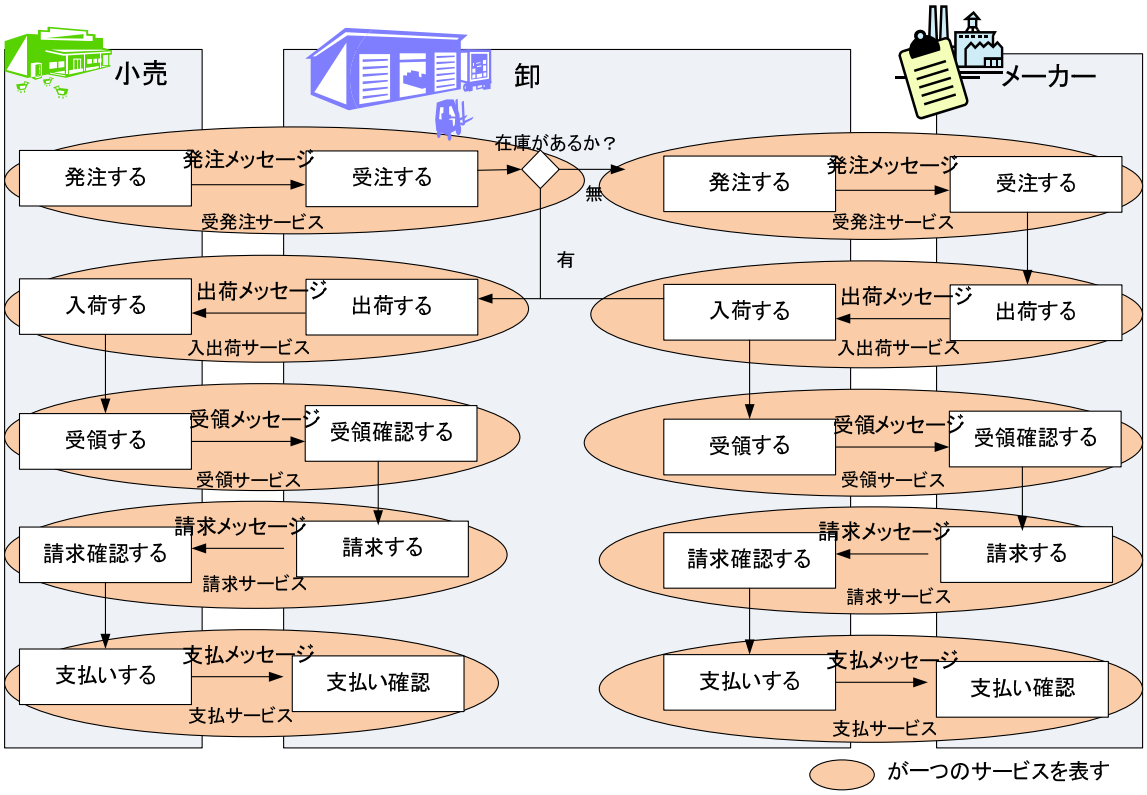

図 0-6　サービス層の設計

ビジネスプロセス層とサービス層を定義したことにより、流通 BMS の基本業務プロセスを SOA によって設計すると図 5-7 のようになる。ソフトウェアコンポーネント層はたとえば受発注サービスの”発注する”というプロセスではコンポーネントは「発注データ入力」→「流通 BMS メッセージに変換」→「データベース挿入」→「メッセージ送信」といった流れになると考えられる。SOA を用いた設計の要はビジネスプロセス層とサービス層の設計となるため、ソフトウェアコンポーネント層の説明・設計はここでは行わない。

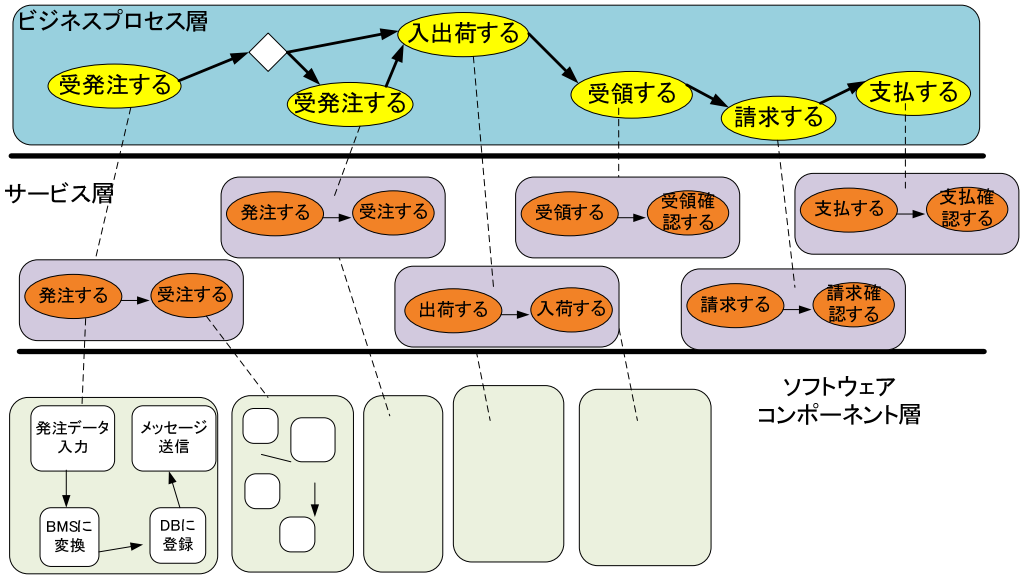

図 0-7　基本業務プロセスの 3 層構造

5.4　サプライチェーン間の在庫変動参照プロセス

サプライチェーン間の在庫変動参照プロセスでは小売は卸とメーカーそれぞれの在庫管理システムをサービスとして呼び出し、自社の在庫管理システムと合わせることでサプライチェーン間の在庫変動を調べることができるとする。ここでビジネスプロセス層には各社への在庫変動の問い合わせが入り、サービス層に各社の在庫管理システムが定義される。(図 5-8)

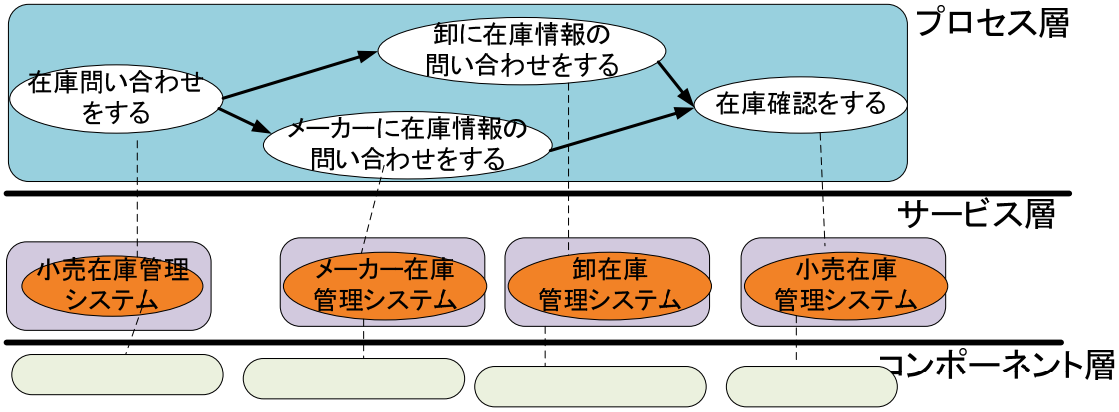

図 0-8　在庫変動参照プロセス

このような在庫変動参照プロセスを SOA を用いて設計する目的は、将来取引先が増えた場合などに対応しやすくなるためである。

現状では在庫問い合わせだけであるが、在庫予測を立てるなどのプロセスを将来的に拡張することができ、流通 BMS の基本業務プロセスと合わせて試作することでより価値のあるシステムを構築することができる。

6.　システムの実装

本節では 5 節で設計した SOA を用いた流通 BMS 対応システムを試作する。

6.1　システムの実装環境

システムの実装には無償で入手できるソフトウェアを用いた。(表 6-1)

表 0-1　システムの実装環境

Oracle SOA Suite10g (10.1.3.1.0)	SOAアプリケーションの設計、開発、デプロイ、監視
Oracle10g(10.1.2.0)	データベース
Oracle Jdeveloper 10g (10.1.3.1.0) Studio Edition	統合開発環境
Oracle Application Server10g(10.1.3.1.0)	サービスの実行
ActiveModeler Avantage1.3.30976	BPMNの作成
Microsoft Windows XP Home Edition	OS

SOA によるシステム実装をするためにサービスインフラストラクチャコンポーネントのセットである、SOA Suite 10g を使用する。

DBMS は以下の制約があるため、Oracle10g を使用し、データは XMLDB 形式ではなく RDB 形式としてデータを格納する。

・　データベースとの接続サービスを提供するデータベース・アダプタは XMLDB に対して選択

操作しかできない。

・ サードパーティの DBMS と接続するには別途、有償のアダプタが必要になる。

・ Oracle SOA Suite のバージョン以上の Oracle データベースや JDeveloper は使用できない。

開発環境として JDeveloper 10g Studio Edition を使用し、グラフィカルなツールを使用することで出来る限りコードを書く手間を省いている。

SOA のサービスを実行する基盤として Oracle Application Server10g を使用する。

BPMN を表記するツールとして KAISHA-Tec の ActiveModeler Avantage を使用する。

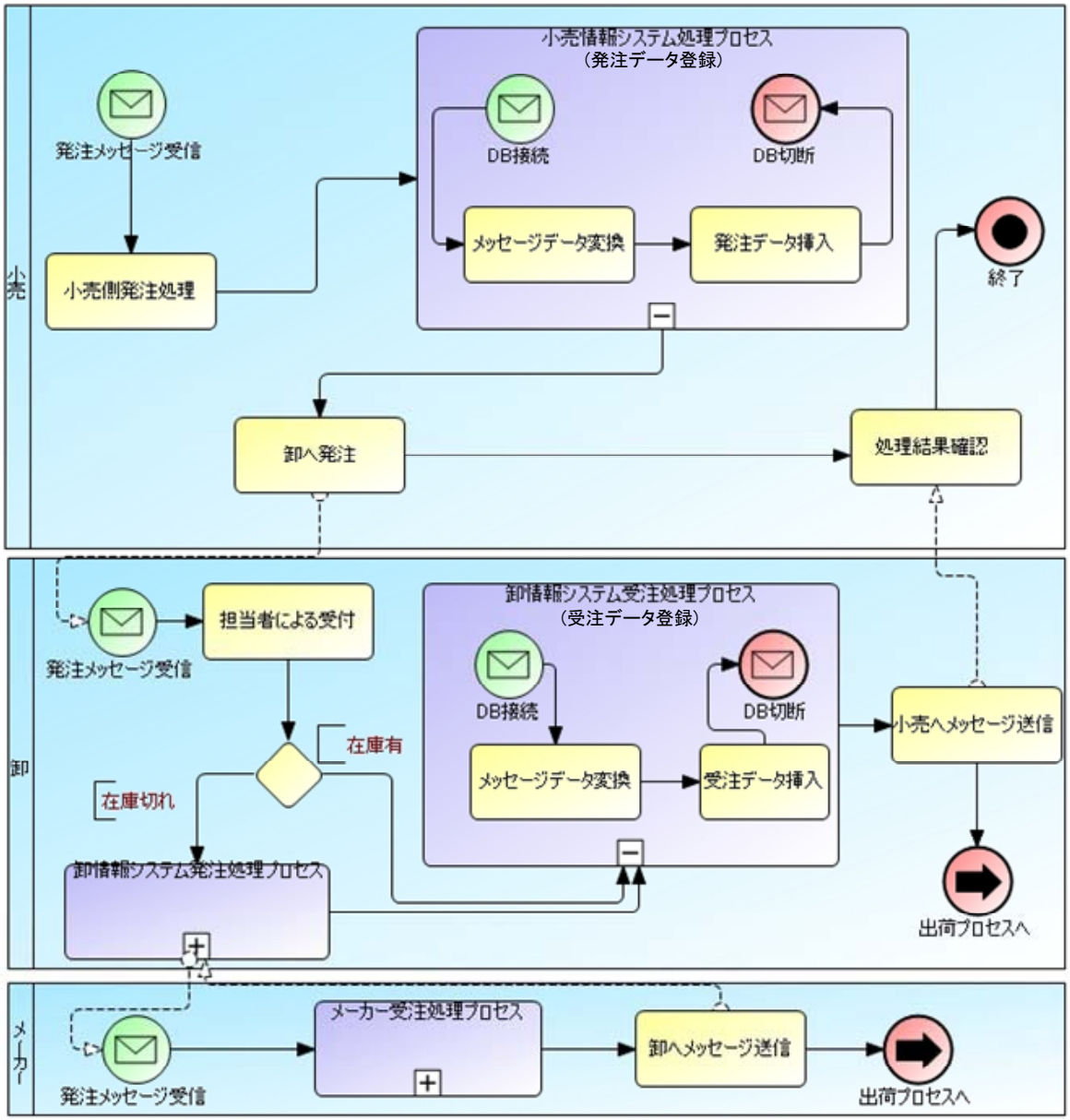

図 0-1　受発注業務プロセスの BPMN

6.2 受発注業務プロセス

6.2.1　受発注業務プロセスの BPMN

受発注業務プロセスを BPMN にすると図 6-1 のようになる。スイムレーンとして小売、卸、

メーカーがある。小売は SecondGenEDI メッセージに準拠したメッセージを入力とし小売情報システムサービスと卸受注処理サービスを呼び出し発注処理をする。卸の担当者は受注した商品の在庫状況を判断する。在庫がない場合は卸発注処理サービスを呼び出しメーカーと受発注業務プロセスを行う。メーカー受注処理サービスは卸からの注文を受注する。最後に卸は処理メッセージを小売に送信しプロセスは終了する。

6.2.2 実装

実装するプロセスでは BPMN で表記されたプロセス図からサブプロセスをサービスとして呼び出すプロセスとする。また、受発注業務は小売りと卸、卸とメーカーそれぞれ別に実装する。(表 6-2)

また、プロセスの入力変数は XML スキーマを使用することで定義することもできる。しかし、指定できるルートスキーマファイルは 1 つであるため、本研究では 3.1 節で説明した解決案 2 である、SecondGenEDI XML スキーマの 2 つあるルートスキーマファイルを SBDH をルートとする 1 つに統合したスキーマを使用した。

表 0-2　実装する受発注業務プロセス

ビジネスプロセス	小売・卸受発注業務プロセス 卸・メーカー受発注業務プロセス
サービス	小売発注処理サービス 卸受注処理サービス 卸発注処理サービス メーカー受注処理サービス

受発注処理サービス

はじめに、小売、卸、メーカーの受発注処理のサービスを実装する。そのサービスのプロセスは図 6-2 のようになり、クライアントからの入力を JDeveloper 付随のデータベース・アダプタにより Oracle10g と接続しテーブルにデータを挿入するサービスである。このプロセスを行うサービスを小売発注処理サービス、卸受注処理サービス、卸発注処理サービス、メーカー受注処理サービスと 4 つ作る。このサービスは図 6-1 の BPMN では情報システムの処理プロセスにあたる。

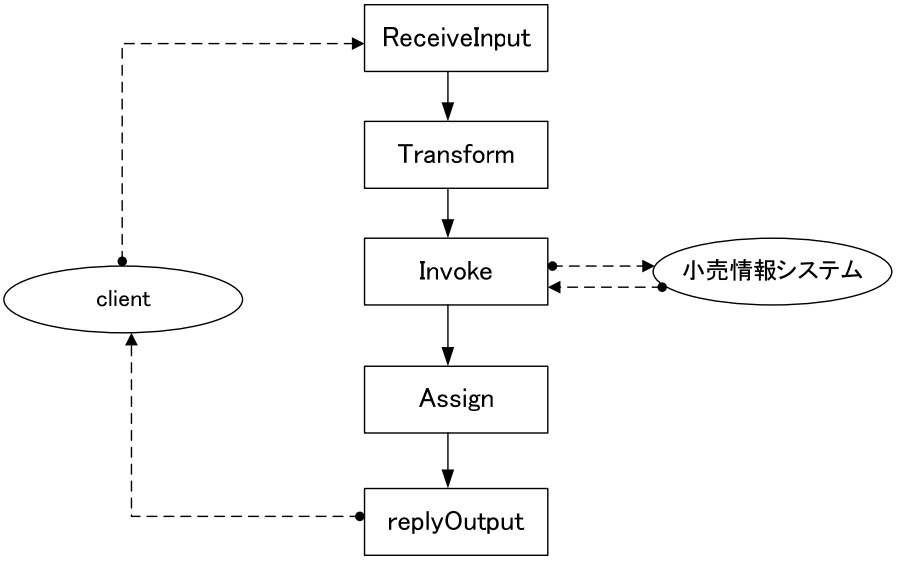

図 0-2　小売発注サービス

プロセスの流れ

1. ReceiveInput
   プロセスの入力変数としてあらかじめ SecondGenEDI XML スキーマを登録しておくことで流通 BMS のメッセージに対応した項目を入力とする。
2. Transform
   小売のデータベースに登録するために、プロセスの入力変数(発注データ)の各項目をデータベースのテーブルと列名に対応させるアクティビティ。サービスを起動する Invoke アクティビティの入力変数に割り当てる。
3. Invoke
   小売情報システムサービスを呼び出すためアクティビティ。入力変数と実行結果である出力変数をもつ。
4. 小売情報システム
   発注データを小売のデータベースに登録するためのサービスでデータベース・アダプタによって定義される。
5. Assign
   小売情報システムサービスの結果をプロセスの出力変数に割り当てるアクティビティ
6. replyOutput
   プロセスの出力変数をクライアントに返すアクティビティ

小売と卸の受発注業務プロセス

このプロセスは小売りと卸との間で行われる受発注業務プロセスである。図 6-1 の BPMN ではスイムレーンの小売りと卸との間のプロセスにあたる。小売りは SecondGenEDI XML スキーマに則った発注メッセージ(発注伝票)を卸に送り、卸は手動で商品の在庫の有無を受注確認とともに小売に送る。なお、メッセージは小売り・卸ともに自社内のデータベースに挿入される。(図 6-3)

1. ReceiveInput
   SecondGenEDI XML スキーマの発注メッセージをプロセスの入力変数にする。
2. Assign_1
   プロセスの入力変数を小売発注サービスを呼び出す Invoke_1 アクティビティの入力変数に割り当てる。
3. Invoke_1
   小売発注サービスを起動する。
4. 小売発注サービス
   別途、作成した小売発注サービス
5. HumanTaskFlow
   HumanTaskFlow では HumanTaskService を呼び出すアクティビティとその結果を受け取るアクティビティがある。
6. HumanTaskService
   タスクを割り当てられたユーザーが手動で在庫のありなしを決定する。
7. Switch
   HumanTaskService での結果によってプロセスは XOR 分岐され、正常にヒューマンタスクが行われた場合は、その結果をプロセスの出力変数に割り当てる。ヒューマンタスクに何らかの異常がある場合はプロセスは強制終了される。

8. Assign_6

卸の受注サービスを呼び出す Invoke_3 アクティビティの入力変数にプロセスの入力変数を割り当てる。

9. Invoke_3

卸受注サービスを起動するアクティビティ。

10. 卸受注サービス

別途、作成した卸受注サービス

11. replyOutput

プロセスの出力変数を出力するアクティビティ。

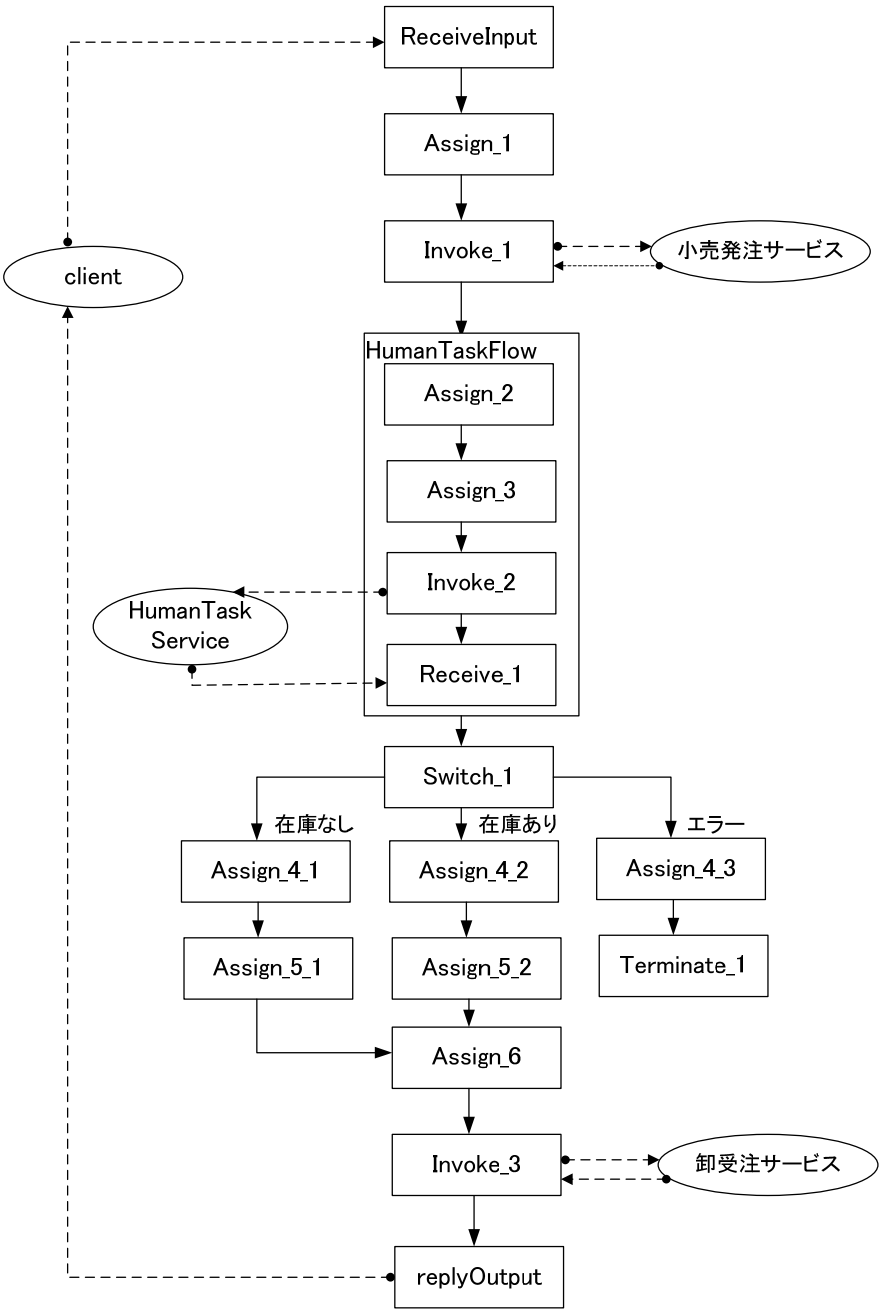

図 0-3　小売と卸の受発注業務プロセス

卸とメーカーの受発注業務プロセス

このプロセスは小売りと卸の受発注業務プロセスから HumanTask を除いたものである。図 6-1 の BPMN ではスイムレーン卸の在庫なしのプロセスで行われる卸発注プロセスからメーカー側のプロセスまでにあたる。小売りからの発注に対し在庫がない場合はこのプロセスを卸は実行する。このプロセスも SecondGenEDI XML スキーマの発注メッセージに則ったメッセージで実行される。(図 6-4)

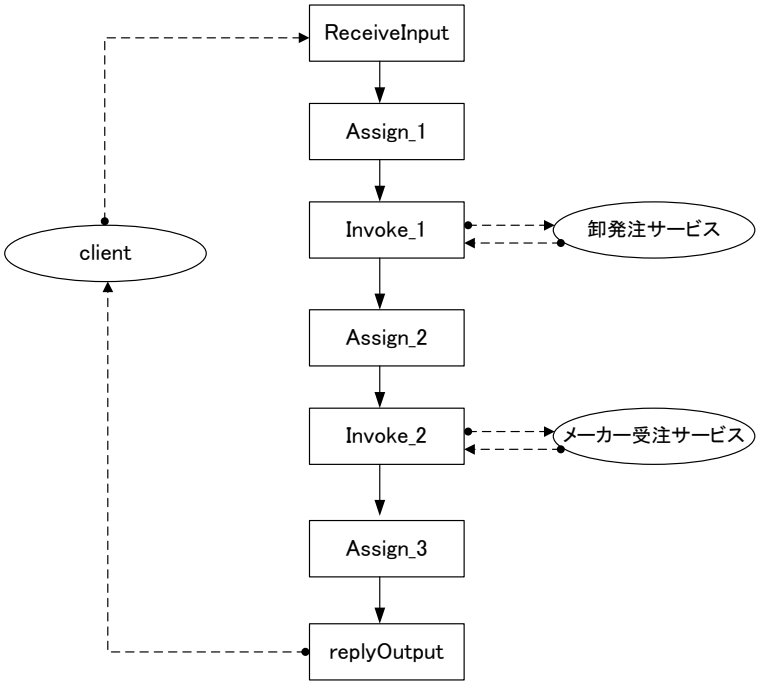

図 0-4　卸とメーカーの受発注業務プロセス

1. ReceiveInput
SecondGenEDI XML スキーマの発注メッセージに基づいた入力を入力変数とする。
2. Assign_1
卸発注サービスを起動する Invoke アクティビティの入力変数にプロセスの入力変数を割り当てる。
3. Invoke_1
卸発注サービスを起動するアクティビティ。サービスの入力変数と出力変数をもつ。
4. 卸発注サービス
データベース・アダプタにより定義された卸の発注用データベースにデータを挿入するサービス。
5. Assign_2
メーカー受注サービスを起動する Invoke アクティビティの入力変数にプロセスの入力変数を割り当てる。
6. Invoke_2
メーカー受注サービスを起動するアクティビティ。サービスの入力変数と出力変数をもつ。
7. メーカー受注サービス
データベース・アダプタにより定義されたメーカーの受注用データベースにデータを挿入するサービス。

8. Assign_3
サービスの実行結果である出力変数をプロセスの出力変数に割り当てる。
9. replyOutput
プロセスの出力変数を出力するアクティビティ。

6.2.2 実行結果

試作したシステムの実行は BPEL コンソールを使用する。SOA Suite に付随する BPEL コンソールはブラウザ上でプロセスを実行し、実行結果の履歴をビジュアル表現できる。

図 6-5 は BPEL コンソールの画面であり、実行した BPEL プロセスが各アクティビティでどのようなメッセージをやり取りしていたかが分かる。左に出ているポップアップがアクティビティでのメッセージのやり取りを表す。このポップアップはアクティビティ監査証跡といい、アクティビティの実行結果を表す。

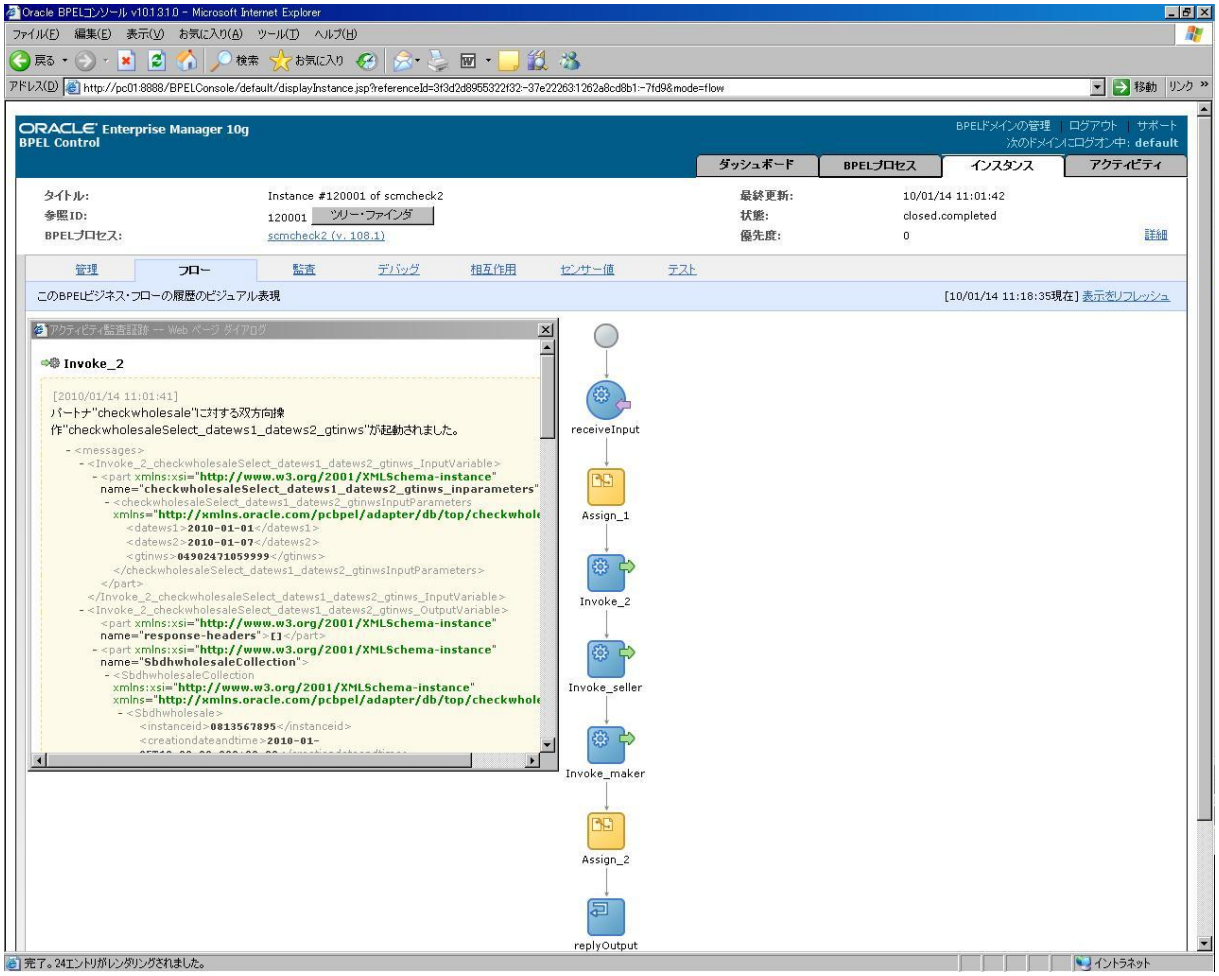

図 0-5 BPEL コンソール画面

図 6-6 は受発注プロセスの receiveInput アクティビティの監査証跡である。図 6-1 の BPMN では小売りレーンの発注メッセージ受信にあたる。図 6-7 は Invoke アクティビティであり、小売りの発注プロセスをサービスとして呼び出す際の入力変数とサービスの実行結果の出力変数

を受け取っているアクティビティである。図 6-1 の BPMN では小売り情報システム発注プロセスを起動するものである。ここではインスタンス ID が 111167895 の発注データを取り扱っている。図 6-8 はヒューマンタスクアクティビティの実行結果であり、卸側の担当者が在庫ありとしたため図 6-9 のようにプロセスの実行結果は”在庫あり”になった。ヒューマンタスクアクティビティは図 6-1 の BPMN では卸レーンの担当者による受付プロセスにあたり、図 6-9 のプロセスの実行結果は処理結果確認プロセスにあたる。

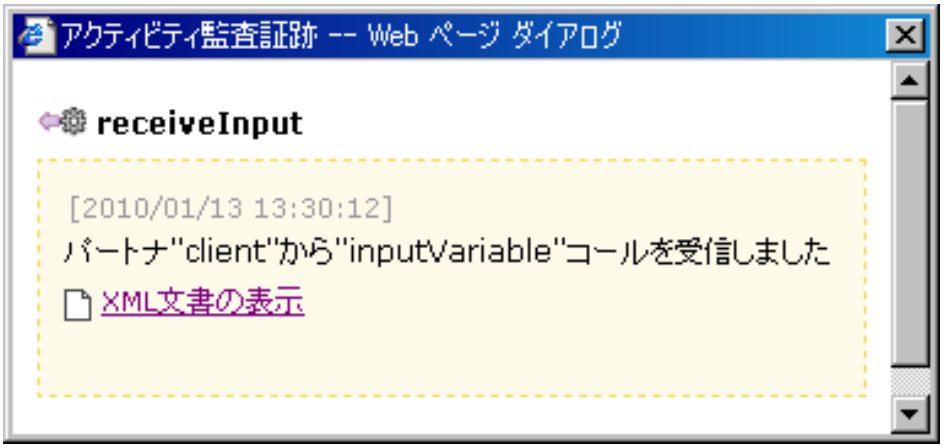

図 0-6　receiveInput アクティビティの監査証跡

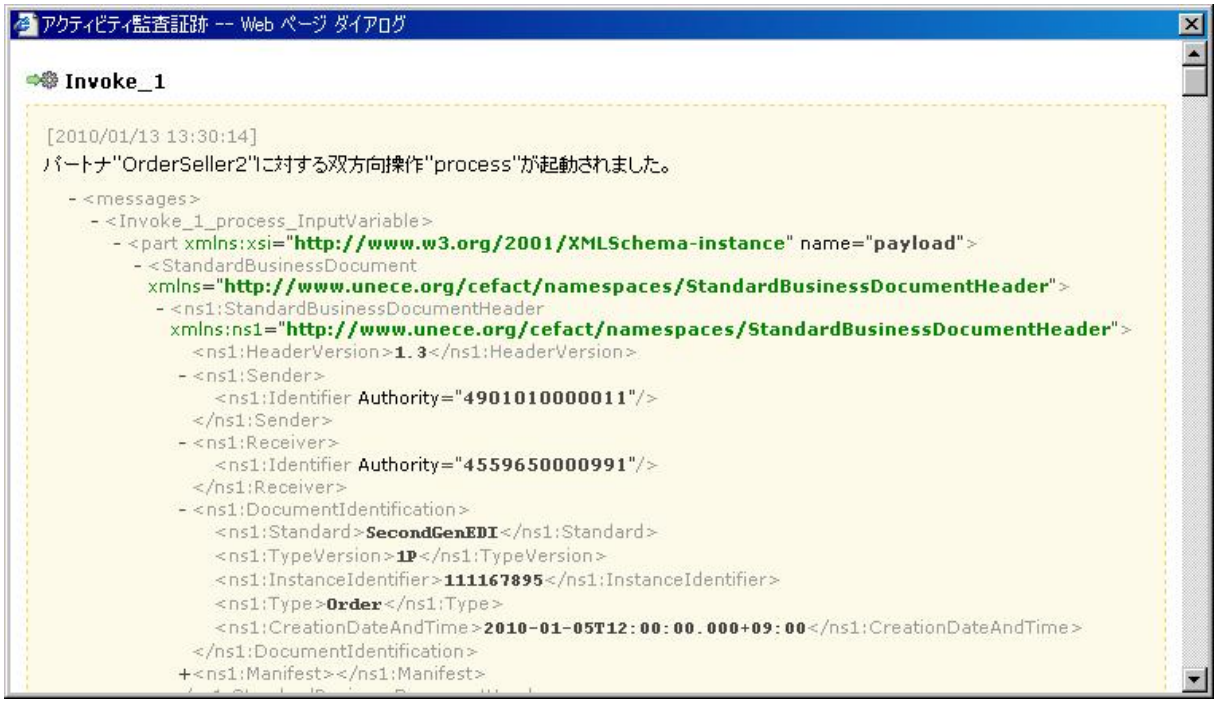

図 0-7　Invoke アクティビティの監査証跡

図 0-8　HumanTask アクティビティ(担当者による受付)の監査証跡

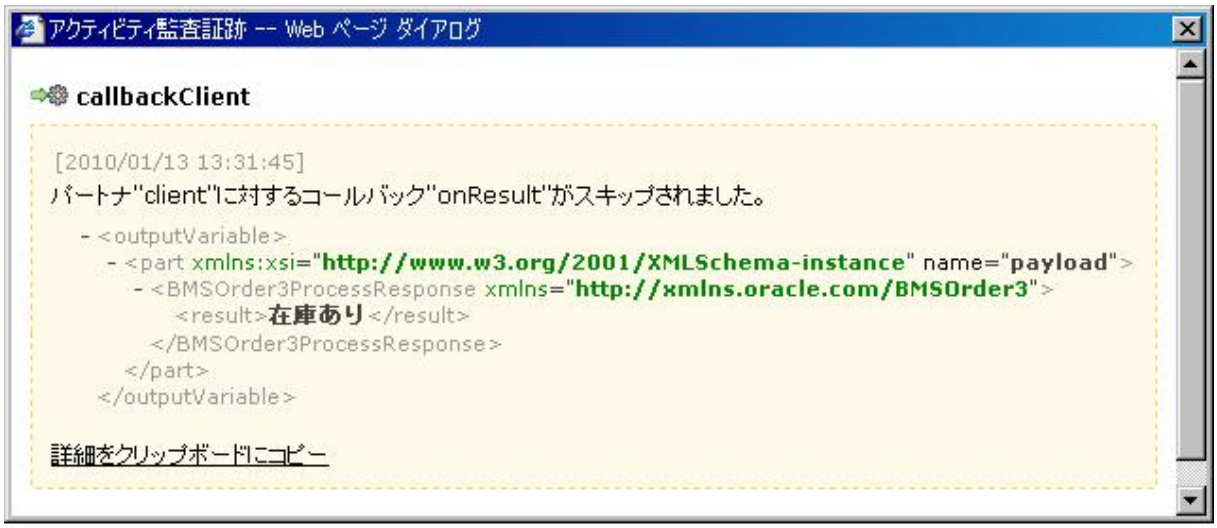

図 0-9　実行結果を示すアクティビティ監査証跡

6.3 サプライチェーン間の在庫変動確認プロセス

6.3.1. プロセスの BPMN 図

サプライチェーン間の在庫変動を調べるプロセスの BPMN 図は図 6-10 のようになる。本研究で試作するシステムではある期間にある商品が小売(自社)がどれだけ発注をし、卸・メーカーはそれぞれどれだけ受注されたかを調べるプロセスとする。

小売の担当者が期間と商品コード(GTIN)を入力するとその条件に合う発注データを自社情報システムから、またサプライチェーン間でどれだけ受注されているかを卸の情報システム、メーカーの情報システムをサービスとして呼び出すことで取得する。

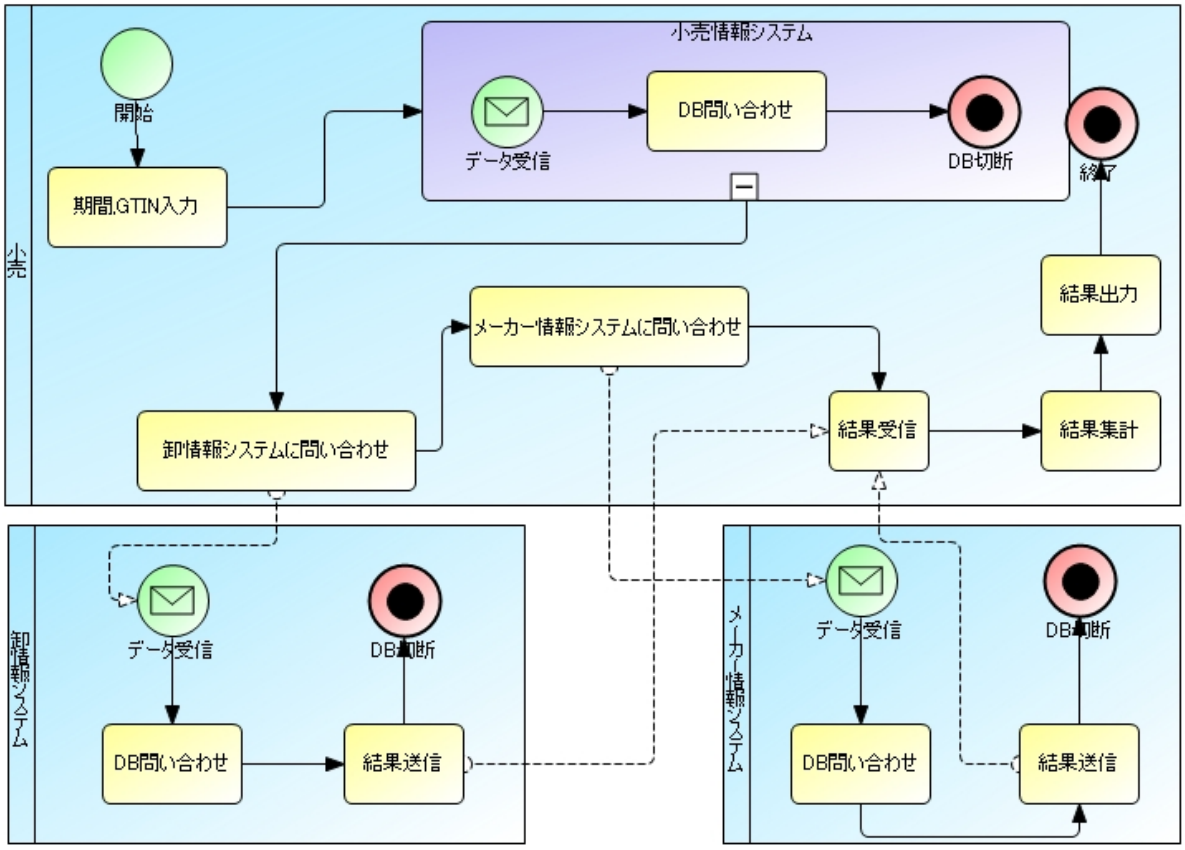

図 0-10　サプライチェーン間の在庫変動確認プロセスの BPMN

6.3.2 プロセスの実装

このプロセスでは小売りが集計開始期間と集計終了期間、商品コード(GTIN)の 3 変数を入力することで、集計期間に小売はどれだけその商品を発注したか、卸はどれだけ受注したか、メーカーはどれだけ受注したかを調べるものである。図 6-10 の BPMN であらわされた全てのプロセスにあたる。自社だけでなく、卸、メーカーの受注量を調べることで市場の変化などに対応できる。(図 6-11)

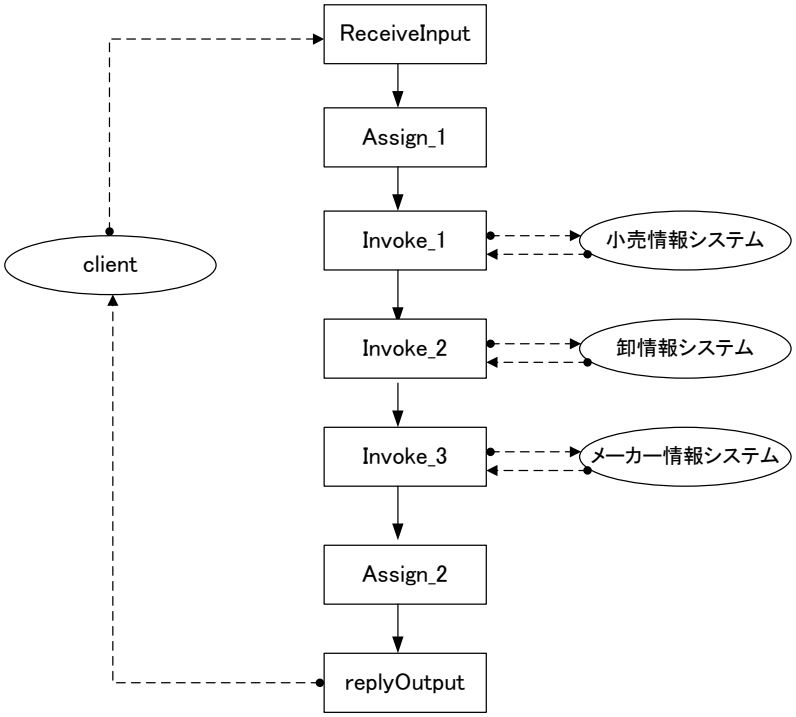

図 0-11　サプライチェーン間の在庫変動を調べるプロセス

1. receiveInput
プロセスの入力変数として集計期間(開始日と終了日)とアイテムコード(GTIN)をもつ。
2. Assign_1
Input 変数を小売・卸・メーカーの情報システムに接続するためのサービスを起動するためのそれぞれの Invoke アクティビティの入力変数に Input 変数を割り当てる。
3. Invoke_1
卸の情報システムにアクセスするサービスを起動するためのアクティビティ。集計期間と GTIN を条件としデータベース・アダプタを使用することで条件に合う受注データを出力変数として受け取る。
4. Invoke_2
小売の情報システムにアクセスするサービスを起動するためのアクティビティ。データベース・アダプタによりサービスは定義され条件に合う受注データを出力変数で受け取る。
5. Invoke_3
メーカーの情報システムにアクセスするサービスを起動するためのアクティビティ。データベース・アダプタによりサービスは定義され条件に合う受注データを出力変数で受け取る。
6. Assign_2
3 つの Invoke アクティビティの出力変数をプロセスの出力変数に割り当てるアクティビティ。また、データベース・アダプタでは SQL 集計関数を使えないためこの Assign アクティビティにおいて BPEL Editor で用意される組み込み関数により集計する。
7. replyOutput
6 の Assign アクティビティにより集計された結果を出力する。

6.3.3　実行結果

2010 年 1 月 1 日から 2010 年 1 月 7 日までの GTIN=”04902471059999”の受発注量を調べた。(図 6-12)

図 0-12　プロセスの入力変数

図 6-13 はプロセスの Invoke アクティビティで呼び出されたサービスにより該当するデータを抜き出している。図 6-10 では小売り情報システムを起動するものである。

図 0-13　Invoke アクティビティの出力変数

プロセスを実行した結果、2010年1月1日から2010年1月7日までのGTIN=”04902471059999”の小売りの発注量は36、卸の受注量は36、メーカーの受注量は6となった。(図6-14)

図 0-14　プロセスの出力変数

6.4　試作したシステムの評価

本研究で試作したシステムの評価として

- 流通 BMS の受発注業務プロセスでは今後、取引先が増えても同様の卸受注サービスを定義することで簡単に対応することができる。例えばヒューマンタスクフローの前にSwitchアクティビティを作成し、GLN(企業の独自コード)で場合分けすればよい。
- 本研究では流通 BMS の基本業務プロセスである、受発注→入出荷→受領→請求→支払の 5 プロセスの中で受発注業務のみの実装となった。受発注業務から入出荷など次の業務プロセスに移る場合は受発注業務プロセスの変数を受け継ぐプロセスを作成することになる。
- サプライチェーン間の在庫変動を調べるプロセスではデータベース・アダプタで用意されている機能を使って DB 選択操作をし、受発注データを XML 形式で取得し、自社でデータから該当項目を抽出し集計している。これでは自社側に負担がかかり SOA による分散処理をうまく生かせない。そのため、他社データベースへのアクセスにストアドプロシージャなどを予め定義することで他社情報システム側に集計処理もさせるとよい。
- 流通 BMS の基本業務プロセスだけでなく、サプライチェーン間の在庫変動を調べることができるプロセスをシステムに追加することで付加価値のあるシステムを試作することができた。

7. 結論

本研究では流通 BMS 対応のシステムを SOA を用いて構築する際に発生する問題を分析した。

2 節では限界を迎えている EDI の問題を解決するために経済産業省主導で策定された流通 BMS について説明をした。

3 節では流通 BMS の普及の妨げとなる要因について分析をした。本研究で明らかになった流通 BMS の普及を妨げる技術的な要因は 2 つあることを示した。

1. SecondGenEDI XML スキーマの構造に由来する要因

・ SecondGenEDI XML スキーマをそのまま使用するだけでは妥当性検証をすることができない問題があることを見つけた。妥当性検証に使用するルートスキーマが 2 つ存在するからである。この問題を解決するために、2 つのルートスキーマを統合するスキーマを作成する方法と 1 つのルートスキーマがもう 1 つのルートスキーマを参照する形にする方法の 2 種類の解決法を提案した。

2. DBMS での実装に由来する要因

・ SecondGenEDI XML スキーマはデフォルトのままスキーマファイルを取り扱うとスキーマの参照ができない問題がある。それはスキーマロケーションに問題があるからである。この問題を解決するためにローカルフォルダベースで記述されているスキーマロケーションを DBMS 実装環境ベースに修正する方法を提案した。

・ DBMS 実装環境において流通 BMS でやりとりされる業務メッセージのサイズが DBMS の通常、格納できるサイズ以上になるため、業務メッセージをそのまま挿入することができない。

4 節では複雑化する IT 環境に対応するためのアーキテクチャとして SOA を説明をした。5 節では実際に流通 BMS 対応システムを SOA を用いて構築する際、受注から支払いまでの一般的な業務プロセスを実装する際には、流通 BMS で定義された各業務メッセージをサービスとすればよいことを示した。さらに、流通 BMS 基本業務プロセスに加えて、サプライチェーン間の在庫変動を参照するシステムを SOA を用いて構築する方法も示すことでより付加価値のある情報システムを構築するための設計段階までを示した。

6 節では 5 節での設計をもとに実際にシステムを試作した。5 節の概要をもとに流通 BMS のデータ形式を用いてサプライチェーン間の在庫変動を調べるプロセスを考えて BPMN を用いてシステムを表記し、それを SOA 環境で試作し評価を行い、流通 BMS を SOA 環境で用いるシステムの特徴の一端を明らかにした。

今後の展望

流通 BMS 対応のシステムを SOA を用いて構築することによってサービス単位でコンポーネント化できる。新たにシステム構築をする際に有用であるため、SOA によるシステム構築がおこなわれていくと考えられる。また、流通 BMS 対応システムはコストをかけられない企業向けにクラウドコンピューティング型でサービスが提供されることになれば中小企業などにも普及が広がるかもしれない。

参考文献

[1] Dan Woods 著 木下哲也 訳 「SAP エンタープライズサービスアーキテクチャ」2004 オライリージャパン
[2] Michael Havey 「詳説 ビジネスプロセスモデリング SOA ベストプラクティス」2006 オライリージャパン
[3] 飯塚岳郎「P2P によるワークフローシステムの開発と応用」2006 年筑波大学大学院システム情報工学研究科修士論文
[4] 糸魚川茂夫「XML データベース入門以前」2006 毎日コミュニケーションズ
[5] 大場克哉,橋本誠,藤倉成太,宗平順己「サービス指向アーキテクチャに基づくアプリケーション設計の現状と課題」2005 巻 75 号 情報処理学会研究報告
[6] 大場克哉 左川聡 橋本誠 藤倉成太 明神知 「実践！SOA モデリング」 2007 翔泳社
[7] 岡田嘉昭「流通 BMS 導入企業のメリット」 2009 No.244 月刊流通ネットワーキング
[8] 小ノ島尚博「流通業界の新 EDI「流通 BMS」が普及に向け動き出す」2009 No.244 月刊流通ネットワーキング
[9] 栗田和則「企業間情報交換の現状と標準化動向について」2005 インテック http://www.intec.co.jp/itj/ITJ5/Contents/3.pdf （2009/06/16）
[10] 佐藤亮 「ビジネスプロセス工学序説」 2008 http://hdl.handle.net/2241/100764
[11] 佐藤亮 「ビジネスの e ビジネス化」 2009 年度経営情報システム第六回配布資料
[12] 佐藤亮 「データモデル、データモデリング機能」 2006 年度経営情報システム第二回資料
[13] 菅原香代子 米持幸寿「XQuery+XML データベース入門」 2006 日経 BP 社
[14] 高橋規生 星佳史 牛山克彦 「複数サービスの連携システム開発における SOA デザインパターン技術」 2007 48 巻 5 号 情報処理
[15] 牧野友紀,栗山勝宏,羽田昭裕,福地修一,妻木俊彦「SOA に基づくアプリケーションシステムの課題と提案」2005 巻 119 号 情報処理学会研究報告
[16] 日本 IBM 株式会社「プロジェクトの流れで理解する XMLDB デザイン徹底解説」 2008 日経 BP 社
[17] 日本オラクル株式会社 「Oracle SOA Suite 10g SOA 実践開発ガイド」 2007 ナレッジオンデマンド
[18] 流通システム開発センター「流通ビジネスメッセージ標準(流通 BMS)の動向」 2009 (財)流通システム開発センター
[19] 流通システム標準化事業「XML テクニカルガイド 」2008 経済産業省
[20] 流通システム標準化事業「共通ライブラリ解説書 H19_1_0 」2008 経済産業省
[21] 流通システム標準化事業「発注メッセージ解説書 H19_1_0 」2008 経済産業省
[22] 流通システム標準化事業「ビジネスメッセージ解説書 ?発注 Ver1.1-」 2008 経済産業省
[23] 流通システム標準化事業「流通 BMS 運用ガイドライン(システム編) 」2008 経済産業省
[24] 流通システム標準化事業「流通 BMS 運用ガイドライン(概要編) 」2008 経済産業省
[25] 流通システム標準化事業「流通 BMS 運用ガイドライン(業界編) 」2008 経済産業省
[26] (財)流通システム開発センター「流通ビジネスメッセージ標準(流通 BMS)の動向」2009 http://www.jipdec.or.jp/ov/n_edi/EDI003.pdf(2009/11/16)
[27] 経済産業省流通システム標準化事業 http://www.dsri.jp/ (2008/10/10)

[28] IT 用語辞典 e-words http://e-words.jp(2009/12/16)
[29] JEDIC - EDI とは http://jedic.ecom.jp/edi/about.html#h1(2009/11/1)
[30] JEDIC 「国内外の EDI 実態調査報告書-2006 年版-」
http://jedic.ecom.jp/activity/report/jittai2006.pdf(2009/12/1)
[31] Oracle XML DB ファースト・ステップ
http://otndnld.oracle.co.jp/tech/xml/xmldb/pdf/XMLDB_1st_step.pdf (2009/03/18 ダウンロード)
[32] Oracle XML DB 開発者ガイド 10g リリース 2（10.2）
http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc_cd/appdev.102/B19255-01/toc.htm(2009/06/10)
[33] SOA Manifesto Working Group SOA Manifesto
http://www.soa-manifesto.org/SOA_Manifesto.pdf(2009/12/6)
[34] XMLDB.JP「RDB と XMLDB 規格」
http://xmldb.moover.jp/db_contents/xpriori/original/rdb_xmldb.html(2009/12/16)
[35]たのしい XML http://www6.airnet.ne.jp/manyo/xml/index.html(2009/2/16)
[36]流通 BMS.com http://www.mj-bms.com/ (2009/9/16)
[37]概説 流通 SCM 次世代の流通情報システム標準化
http://www.dsri.jp/scmpjt/public_info/pdf/scm/scm.pdf(2009/6/15)
[38] IBM WSDD -SOA 入門- 第 1 回 Japan
http://www-06.ibm.com/jp/software/websphere/developer/soa/1.html(2010/01/18)
[39]経営改革を支援する SOA http://jp.abeam.com/result/pdf/dl.html?id=SL046 (2010/1/18)
[40]SOA 入門：第 1 回 SOA って何？
http://www.ibm.com/developerworks/jp/websphere/library/soa/soa_intro/1.html(2010/1/15)
[41]逆引き SQL リファレンス
http://orakumiko.hp.infoseek.co.jp/sql/r_refer.html#hikaku(2010/1/3)
[42]BPM 製品のトレンドと導入
http://www.computerworld.jp/news/sw/43863-2.html(2010/1/15)