2004－SE－145　(13)
2004／8／20

# 組織間連携ミドルウェアのための EDMA (Evolution Driven Middleware Architecture) の提案

坂田 祐司, 松田 栄之
（株）NTT データ　技術開発本部
e-mail: {sakatayu, matsudasg}@nttdata.co.jp

**あらまし** 本稿では，企業間取引におけるビジネスプロセス実行のためのミドルウェアアーキテクチャとして，EDMA(Evolution Driven Middleware Architecture)を提案している．EDMA は，Web サービスとワークフローエンジンを基盤の技術としており，企業間取引の進化に対して適応的であるという特徴がある．進化に対して適応的であるというアーキテクチャを検討するため, 企業間取引の進化モデルを特徴的な 5 つの段階に分類する. その分類を基に EDMA アーキテクチャを提案する．また，各段階への進化において，EDMA アーキテクチャでは，ソフトウェアの入れ替えを必要とせず，コンポーネントを追加すればよいという特徴を示す.

# A proposal for evolution driven middleware architecture for eBusiness process execution

Yuji Sakata, Shigeyuki Matsuda
NTT Data Corporation
e-mail: {sakatayu, matsudasg}@nttdata.co.jp

**Abstract.** This paper proposes the EDMA (Evolution Driven Middleware Architecture) for eBusiness process execution. EDMA is a middleware architecture which is adaptable for eBusiness evolution based on workflow management system and web services technologies. In order to build highly adaptable architecture, we pay attention to an evolutionary model of eBusiness. We show the characteristic phases from a eBusiness evolution and illustrate EDMA is so adaptable that the middleware implementing EDMA need not be replaced but be just added components to through eBusiness evolution. Besides, we introduce our implementation of a workflow engine and researches relate to EDMA.

## 1. はじめに

近年, インターネット技術を用いて企業間の取引を行う, 電子商取引が重要となってきた. 特に, 複雑で多様なビジネスプロセスを含む処理を電子的に行うことは, 処理の効率化の観点から, 企業に多大な利益をもたらすと考えられている. それゆえ, インターネット技術を用いてさまざまな取引相手と通信し, 複雑な処理行う技術が求められている.

現在, Web サービスは, インターネットを介してさまざまな取引相手と通信するための標準的な技術になっている． Web サービスは，ネットワークを介してシステム同士が通信することを目的として設計されたソフトウェアシステムである．具体的に言えば，システムが解釈可能な WSDL(Web Services Definition Language)[1]などの形式で表現されたインタフェースを持ち，他のシステムと SOAP[2]のメッセージ形式の送受信によって通信するシステムである[3]．通信には HTTP(Hypertext Transfer Protocol)が用いられることが一般的である．Web サービスは, 汎用的で標準化された技術によってシステム間の通信を実現しているため相互運用性が高い．よって，企業間取引を行うためのソフトウェアが, 個々の企業によりさまざまなフレームワーク(.Net や J2EE)やさまざまな実装言語 (C++, Java, Perl など)で実現されていることを考慮すると, Web サービスは, それらのシステム間で通信するための有力な方式であるといえる.

また，企業間取引において，発注処理や決済処理などビジネス上のまとまった意味を持つ処理を実現する場合, 企業間のシステムで単一の通信ではなく複数回の通信が発生する場合が多い．この通信の順序や通信内容の規定を，一般にビジネスプロセスと呼ぶ[4]. 現在，ビジネスプロセスの実行, 管理をシステム上で実現するためにさまざまな言語(以降，ビジネスプロセス言語と呼ぶ)が提案されている[5]．たとえば，BPEL4WS [6], WS-CDF [7] , BPML [8]などがあげられる. ビジネスプロセス言語は個々のモデル化の概念によってのみ比較されるべきものではなく, 実現する企業間取引の形態を考慮してその優劣が比較されるべきである．また，ビジネスプロセス言語を解釈し，ビジネスプロセスを実行するためのソフトウェアは, 言語独自の実装となることが通常である．すなわち，企業間取引形態が変化し, その形態に適したビジネスプロセス言語を利用しようとする場合, ビジネスプロセスを実行するためのソフトウェアを完全に入れ替えなければならない可能性がある．しかし，ソフトウェアの入れ替えは，迅速な企業間取引形態の改良を妨げる要因となり問題である．よって，企業間取引におけるビジネスプロセス実行のためのミドルウェア(以降，プロセス実行ミドルウェアと呼ぶ)は，入れ替えをする必要がないアーキテクチャが望まれる.

本稿では，プロセス実行ミドルウェアのアーキテクチャとして，EDMA(Evolution Driven Middleware Architecture)を提案する．上記の議論から，プロセス実行ミドルウェアは，企業間取引の進化に適応できることが求められている．EDMA は，企業間取引の進化に対して，構成するコンポーネントを追加することにより対応できるアーキテクチャである点が特徴である．本稿は，以降，次のように構成される．2 章では，企業間取引の進化を特徴的な 5 つの段階にモデル化する．また，個々の段階におけるプロセス実行ミドルウェアの要件を示す．3 章では，2 章で示した要件から望まれるアーキテクチャの特徴について議論する．4 章では，この議論を基に EDMA の全体像を示し，3 章で示したアーキテクチャの特徴を満たしていることを示す．5 章において関連論文と今後の展開を示し，最後に 6 章でまとめる．

## 2. 企業間取引のモデル化

本章では，購入処理という典型的な企業間取引例を用いて，企業間取引の進化の段階を示す．

### 2.1. 購入処理の例

企業 A は，コンピュータ製造業者である．多くの部品業者からコンピュータの部品を購入する．企業 A と部品業者の間では，あらかじめ購入に関する契約がなされているものとする．企業 X は，企業 A と購入に関する契約を締結している部品業者であり，また企業 A 以外の業者に対しても部品を販売している．

企業 A では，部品購入処理を支援するための複数の内部システムが存在している．たとえば，決裁処理，発注処理，支払い処理などのシステムである．さらに，企業 A と企業 X は専用線で接続され EDI を用いて取引が電子化されている．現在，企業 A における部品購入者は，企業 A において定められたビジネスプロセスに従い，企業 A の内部システムと独自の EDI アプリケーションを利用して購入処理行っている． 企業 A は，EDI を利用する費用が高いため，電子商取引の技術を用いて購入処理を効率化しようと検討している．

### 2.2. 企業間取引の進化の段階モデル

この節では，2.1で説明した企業 A の購入処理の進化を 5 つの特徴的な段階にモデル化して説明する．表 1に，その要約を示す．

#### 2.2.1. 段階 I: 特定取引相手との固定的な処理

「特定取引相手との固定的な処理」とは，固定されたビジネスプロセスを特定の相手と実行することである．あらかじめ，企業 A と企業 X が購入処理を行うために送受信すべきデータとその送受信の順番を定義する．そして，図１のように，その定義をプロセス実行ミドルウェアが自動的に実行，管理するという形態である．企業 A のプロセス実行ミドルウェアは，ビジネスプロセスを管理し，企業 X のシステムおよび企業 A の内部システムと通信する責務を持つ．

この段階への移行は，規則的な業務の効率性を高めるためになされる．ビジネスプロセスを自動化すると，システムごとに異なるデータを人手で入力する形態に対して安価であり，かつ正確に処理が遂行されるためである．

**表 1: 企業間取引の進化段階**

Time →

| | I | II | III | IV | V |
|---|---|---|---|---|---|
| **企業間取引の段階** | 特定取引相手との固定的な処理 | 取引企業間の固定的な処理 | 役割ベースの動的連携 | 機能ベースの動的連携 | 適応的連携 |
| **主要な動機** | 規則的な取引の効率化 | | 取引の最適化 | | |
| **取引相手との連携モデル** | 特定の連携相手 | あらかじめ定められた連携相手 | 動的に決定する連携相手 | | 適応的に決定する連携相手 |
| **実行時に決定する単位** | | | 役割 | 機能 | |

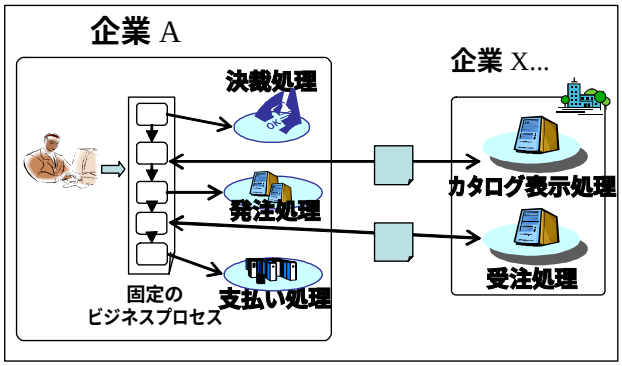


**図 1 特定取引相手との固定的な処理**

### 2.2.2. 段階 II: 取引企業間の固定的な処理

「取引企業間の固定的な処理」とは，コンピュータ製造業者と部品製造業者間における購入処理のビジネスプロセスを規定することにより，企業 A と企業 X といった特定の取引企業間だけでなく，図 2のように，複数の企業間の購入処理をプロセス実行ミドルウェアによって自動化する形態である．

段階 I において，企業 A は，個々の部品製造業者ごとに，購入処理を実現するためのビジネスプロセスを定めることを想定している．しかし，多くの部品業者が企業 A と取引を行い，また企業 X も多くのコンピュータ製造業者と取引を行う場合を考慮すると，個々にビジネスプロセスを定めて，そのプロセスによって購入処理を実現するというモデルは，保守性やコストの観点から望ましくない．また，複数の企業間と取引を行うという環境においては，組織間の独立性，すなわち，企業の内部のシステムは，他の企業のシステムから直接的に制御されないことが望まれる[9]．それゆえ，段階 I から段階 II への変化は，取引相手のシステムとのビジネスプロセスを標準化することにより，規則的な処理の効率性をさらに高めようという動機による．

また，この段階では，取引の方式が標準化されているため，企業 A の調達者は，どの部品製造業者に購入処理を行うかを，取引開始時に人手により選択することが可能となる．

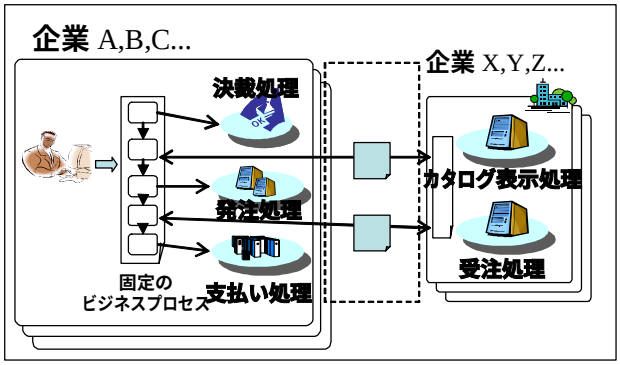


**図 2 取引企業間の固定的な処理**

### 2.2.3. 段階 III: 役割ベースの動的連携

「役割」とは，企業間取引において，期待される振る舞いが同じであるような企業の集合を表す．2.2節における例では，「コンピュータ製造業者」や「部品製造業者」などが相当する．「役割ベースの動的連携」とは， 図 3で示すように，企業 A がビジネスプロセスの実行を開始する時点で，ある役割の企業群から自動的に一つの企業を選択し，取引を実行するという形態である．

この段階へ進む契機は，規則的な取引の効率化ではなく，取引内容の最適化である．段階 I,II における企業 A の購入処理において，取引相手は固定的(段階 I)であるか，人手によって選択（段階 II）されていた．しかし，部品の値段や納期など，取引相手を選択するための情報が豊富になると，人手によって選択をすることは困難となる．さらにこれらの情報が動的に変化すると，固定的な取引相手と取引を行うことは，最適な取引であるとはいえない．よって，取引を開始する時点で，その時点の情報から自動的に取引相手を選択する仕組みが求められる．

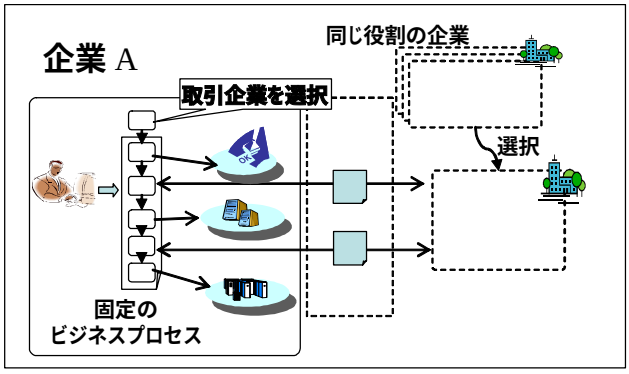


**図 3 役割ベースの動的連携**

### 2.2.4. 段階 IV: 機能ベースの動的連携

「機能」とは，企業間取引における処理の最小単位である．「機能ベースの動的連携」とは，取引の開始時に，取引の内容とその時点の環境情報を利用して，取引相手だけではなく，機能を組み合わせるためのビジネスプロセスも決定し，そのビジネスプロセスを実行する形態である．

この段階に進化する動機は，取引の開始時における情報に基づいて，ビジネスプロセスと取引相手を決定することによる取引内容の最適化である．今後，企業間取引が進化し，さまざまな企業が参加する開かれた取引場が形成されることが予想される．その場合，取引の種類ごとにビジネスプロセスを統一することが困難となる．つまり，同じ取引内容が，異なるビジネスプロセスによって実行される場合が考えられる．よって，固定のビジネスプロセスによって取引を実行することを前提とした段階 III までの段階では，企業 A は，最適な取引内容を実現するためには，その取引内容を実現する複数のビジネスプロセスを前もって用意しなければならず，保守性にかける．よって，取引が実行される際にビジネスプロセスを自動的に形成することが望まれる．

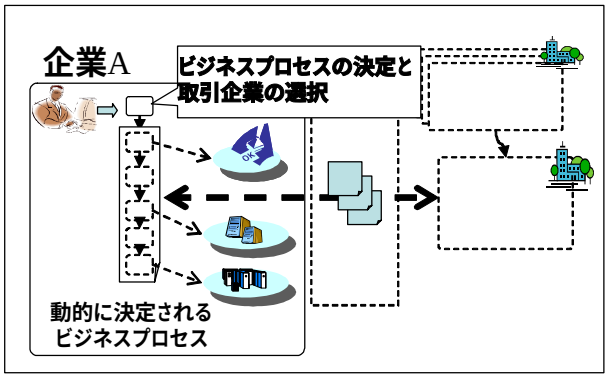


図 4 機能ベースの動的連携

#### 2.2.5. 段階 V: 適応的取引

「適応的取引」とは，取引の開始時にビジネスプロセスを確定する段階 IV と異なり，図 5のようにプロセス実行ミドルウェアが機能を実行するごとに次に行う機能を決定する形態である．

段階 IV では，取引の開始時点でビジネスプロセスを確定している．しかし，取引が長時間かかる場合，最適なビジネスプロセスは，取引開始時点と異なる可能性がある．また，開始時には最適なビジネスプロセスを完全に確定できず，部分的なビジネスプロセスのみを決定し，決定したプロセスを実行後に残りのビジネスプロセスを確定するという形態や，取引の実行中に何らかの障害が生じた際に，その時点での最適なビジネスプロセスを再計算し実行するという形態も，「適応的取引」の段階では可能となる．

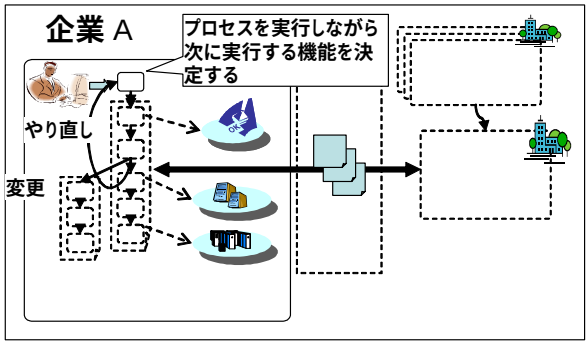


図 5 適応的取引

## 3. 企業間取引の進化モデル駆動のプロセス実行ミドルウェア

本章では, 2.2節で示した企業間取引の進化モデルを基に, この進化モデルに適応するプロセス実行ミドルウァアのアーキテクチャについて議論する．

### 3.1. プロセス実行ミドルウェアに対する要件と解決方法

最初に，各段階において考慮すべき要件と解決方法を抽出する．次にビジネスプロセス実行ミドルウェアを構成するコンポーネントを検討する．表 2に，要件と解決方法を要約している．

段階 I では，プロセス実行ミドルウェアは，さまざまな技術によって実装されている, 内部システムおよび取引企業のシステムと通信する機能と, 固定したビジネスプロセスを実行する機能が必要となる．よって，通信に使われるメッセージを標準化する SOAP[2]を利用することにより，実装に非依存である通信を実現し，ワークフローエンジン[11]によって，ビジネスプロセスを制御するアーキテクチャが必要である．また，段階 I では，一つのビジネスプロセスは, 1 つのワークフローエンジンによって実行するか，企業ごとに配置されたワークフローエンジンが協調して実行するか 2 通り考えることが出来る．保守性の観点から考慮すると，1 つのワークフローエンジンによって実行する形態が良い．段階 II では，企業間の独立性を確保する必要がある．そのため，企業の内部の振る舞いを取引企業から隠蔽する必要がある．よって，取引において取引相手とやり取りするための機能インタフェース(WSDL はこれに相当する) と，そのインタフェースを介して機能を実行する順序を明確に定義する必要がある (以降，この定義をコレオグラフィと呼ぶ)．また，取引において，お互いに隠蔽された内部処理を制御することは出来ないため，ワークフローエンジンは企業ごとに配置され，それらが協調して取引のビジネスプロセスを実行する形態でなければならない．よって，コレオグラフィから，個々の企業のワークフローを導出する機能が望まれる．

段階 III では，望む役割を持つ取引相手から最適な取引相手を自動的に選択する機能が必要となる．どのような情報によって, 取引相手を選択するかは取引内容に依存しいている．よって，取引の条件(値段，納期，過去の実績など)を形式的に表現するメタデータの形式のみを定義する必要がある．プロセス実行ミドルウェアは，その形式的に表現された取引条件を解釈し, 同じ形式によって表現された取引相手の Web サービスの情報をデータベースから発見するという機能が必要となる．段階 IV では，適切な取引相手を選択するだけでなく, 取引相手の持つ機能から最適なビジネスプロセスを合成する必要がある．よって，システムの機能の動的な側面の意味定義とその定義に基づいたビジネスプロセスの合成を行う機能が必要である．

段階 V において，プロセス実行ミドルウェアは，取引の実行中に, 時間の経過や実行による状況の変化に対応して, ビジネスプロセスを変更する機能が必要となる．この機能を実現するため，個々の取引だけではなく，その取引の実行を必要とするアプリケーションに対する振る舞いを規定する必要がある．よって，ポリシー，すなわち「あるオブジェクトの振る舞いに影響を与える情報」[13]を基として，プロセスを実行できる仕組みが求められる．

表 2 各段階におけるビジネスプロセス実行ミドルウェアに対する要件と技術的解決手法

Time

| 企業間取引の段階 | I | II | III | IV | V |
|---|---|---|---|---|---|
| 主要な要件 | 1. ビジネスプロセス実行の自動化<br>2.システム実装に対する非依存性 | 企業の独立性 | 最適な取引相手の発見 | 最適なビジネスプロセスの決定 | 適応性 |
| 技術的解決方法 | 1. ワークフローエンジン<br>2. 標準メッセージ形式(SOAP) | 1.コレオグラフィ<br>2.システム解釈可能なインタフェース記述(WSDL)<br>3. コレオグラフィからのワークフロー導出 | 1. システムの静的な側面の語彙の標準化<br>2. レジストリ(UDDI) | 1. システムの動的な側面の語彙の標準化<br>2. プロセスの合成技術 | ポリシー |
| プロセス実行ミドルウェアのモデル | 静的連携 | | 動的連携 | | |
| | 単一 | 分散 | | | |
| | | | 動的結合 | プロセスの動的な決定と動的結合 | プロセスの適応的な実行と動的結合 |

3.2. EDMA

3.1節での議論をもとに，われわれは，(Evolution Driven Middleware Architecture)を提案する．このアーキテクチャは，アーキテクチャを構成するコンポーネントを追加することによって，企業間取引の進化に対応できることを特徴としている．アーキテクチャの全体像を図 6に示す．取引処理を実行するアプリケーションは，このアーキテクチャの実装を介して処理が行われる．個々のコンポーネントの概要を示す．

ワークフローエンジン(Workflow Engine)

ワークフローエンジンは，与えられたワークフロー定義を実行するコンポーネントである．

ワークフローレポジトリ(Workflow Repository)

ワークフローレポジトリは，取引に利用されるワークフロー定義を蓄積する．

Webサービスハンドラ(Web Services Handler)

Web サービスハンドラは，Web サービスを用いて，内部および取引相手のシステムと通信を行う．また，システムの実装依存のデータ形式を SOAP のメッセージに変換する責務を持つ．

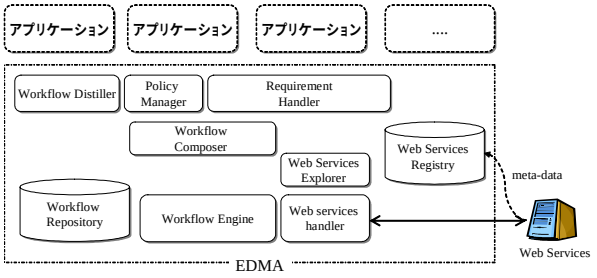


**図** 6 EDMA (Evolution Driven Middleware Architecture)

ワークフロー導出コンポーネント(Workflow Distiller)

ワークフロー導出コンポーネントは，与えられたコレオグラフィから，このプロセス実行ミドルウェアが実行するワークフローのテンプレートを導出し，ワークフローレポジトリに保存する．

Webサービスレジストリ(Web Services Registry)

Web サービスレジストリは，利用可能な Web サービスのメタデータを保持し，Web サービス発見コンポーネントとワークフロー合成コンポーネントに，与えられた条件を満足する Web サービス発見する機能を提供する．UDDI(Universal Description, Discovery and Integration) [14]は，この機能を実現するための標準であるといえる．

Web サービス発見コンポーネント(Web Services Explorer)

Web サービス発見コンポーネントは，Web サービスレジストリを検索して，Web サービスを発見する機能を持つ．

要件ハンドラ(Requirement Handler)

要件ハンドラは，アプリケーションの要求を解釈するモジュールである．

ワークフロー合成コンポーネント(Workflow Composer)

ワークフロー合成コンポーネントは，Web サービスレジストリに登録されている Web サービスを利用して，要件ハンドラから取得する要件を満足するワークフローを合成するコンポーネントである．

ポリシーマネージャ(Policy Manager)

ポリシーマネージャは，実行されている取引処理が定義されたポリシーを守るように制御する．

### 3.3. EDMA の適応性

本節では，EDMA の適応性，すなわち EDMA を基としたプロセス実行ミドルウェアは，企業間取引の進化に治してミドルウェアを取り替える必要が無く，コンポーネントを追加すればよいということを示す．

図 7に，1 つの取引処理を実行する場合の各段階におけるコンポーネント間のやり取りを示す． 図 7において，色の濃い長方形がその段階において必要なコンポーネントを表しており，コンポーネントは，英語表記の頭文字によって示している．また，矢印は，コンポーネントやアプリケーション間の相互作用を表している．また，矢印に付加されている番号は相互作用の順序を表している．

図 7で示されているように，企業間取引の段階があがると，コンポーネントが追加されていく．段階 I では，アプリケーションは，事前に定義されたワークフローをワークフローレポジトリから読み込み，そのワークフローを実行する．ワークフローエンジンは，Web サービスハンドラを利用することによって Web サービスと通信を行う．段階 II において，ワークフロー定義は，取引相手との合意されたコレオグラフィから，ワークフロー導出コンポーネントによって導出される．段階 III では，アプリケーションはワークフロー定義における「取引相手」を「役割」で抽象化した定義をワークフローエンジンに与える．同時に要求ハンドラに，希望する取引に関する情報を与える．ワークフローエンジンは，要求ハンドラの情報を基にして，Web サービス発見コンポーネントを用い，Web サービスレジストリを検索して取引相手を選択する．段階 IV の場合は，要求ハンドラに，望んでいる取引を表す情報を与え，ワークフロー合成コンポーネントが，その情報を満足するワークフロー定義を Web サービスレポジトリの情報を用いて合成する．段階 V では，段階 IV と比べると，取引の最初の段階だけでなく，ワークフロー定義で記述されている「機能」が実行されるたびに，ポリシーとその時点における環境の情報を基にワークフローの合成が行われる．

## 4. EDMA の実装と関連研究

本章では， EDMA に用いることが出来る開発したプロセス実行ミドルウェアと，このアーキテクチャを実現するために行っている研究について紹介する．

### 4.1. ΣServ

ΣServ は，EDMA の段階 I を満たすことの出来るフレームワーク製品である．つまり，ワークフローエンジンとワークフローレポジトリ，そして Web サービスハンドラから構成されている．さらに商業利用を想定した追加機能を持っている．また Java で実装されており，以下の 4 つのコンポーネントから構成される．

1. Σ Serv FlowManager: Σ Serv FlowManager は，eCo-Flow [15]を基としたワークフローエンジンとワ

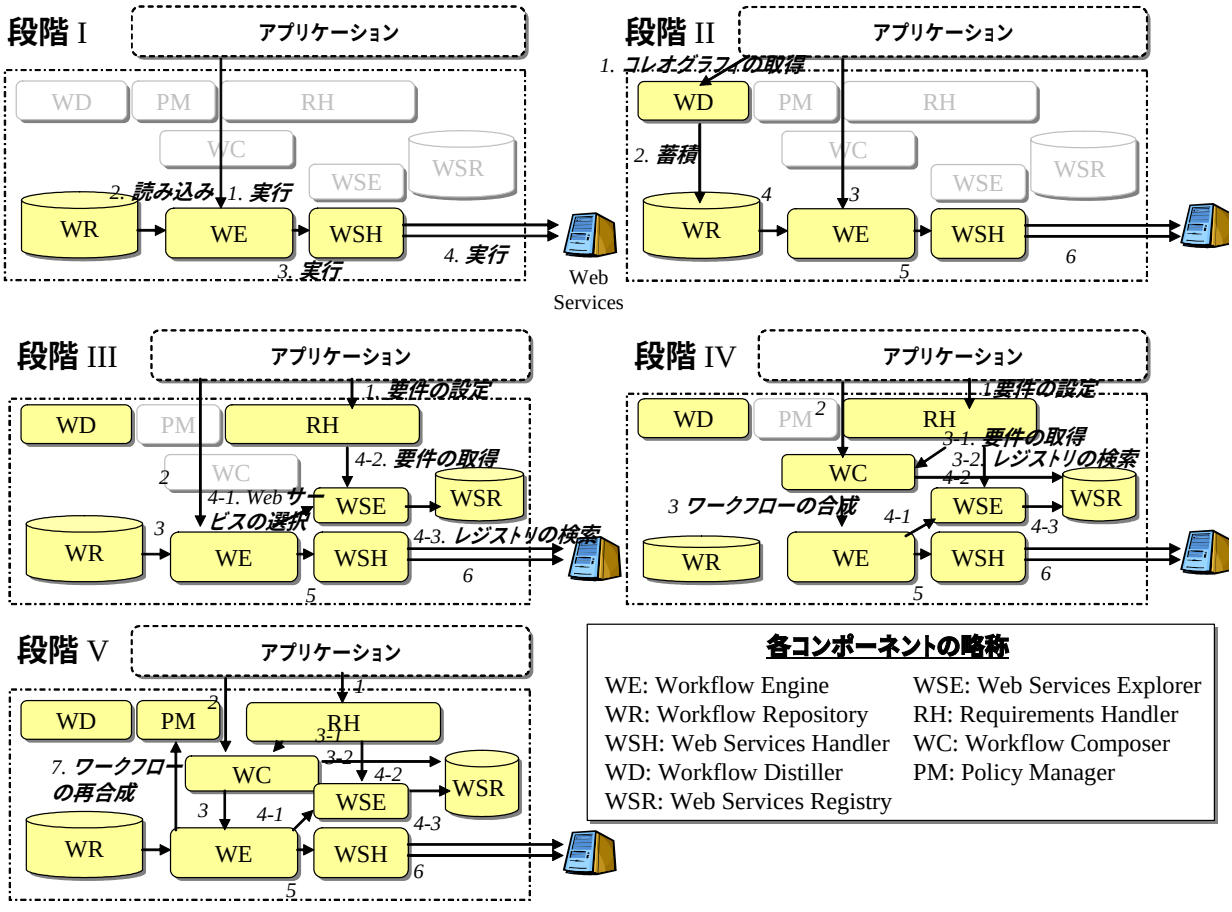


**図 7 各段階における処理概要**

ークフローレポジトリである．

2. ΣServ TrustMessage: ΣServ TrustMessageは，Webサービスハンドラであり，さらに，送達確認が可能な高信頼SOAPメッセージの通信をebXMLのメッセージサービス仕様[16]を満たすことにより実現している．
3. ΣServ TransInfo: ΣServ TransInfoは，データの変換やデータ形式の変換を行うコンポーネントである．
4. システムアダプタ: Webサービス以外の方式で通信を行うために用意されているアダプタである．

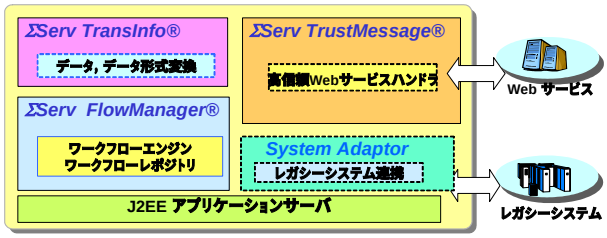


**図 8 ΣServ プラットフォーム**

### 4.2. EDMA に関する研究内容

ここでサービス合成コンポーネントに関する現在の取り組みを簡単に紹介する．

われわれは，Web サービス実行処理が，その Web サービスを利用するアプリケーションの状態を変化させることに注目し，ワークフロー定義を合成する方式を提案している[17]．この方式において，アプリケーションがどのような状態である場合に Web サービスは実行可能であり，また，実行した場合にアプリケーションの状態はどのように変化するかという点を各 Web サービスごとに形式化する(以降，この形式化された情報を機能定義と呼ぶ)．そして，この機能定義を Web サービスに対するメタ情報としてサービスレジストリに保存する．そして，アプリケーションが，ある取引を実現しようとする場合，現在の状態と取引の終了後に望まれる状態を要件定義として既定する．なお，機能定義と要件定義は Situation Calculus**[10]**によって形式化している．Situation Calculus を利用することにより，ワークフロー合成コンポーネントは，どの機能定義を持つ Web サービスを組み合わせれば，要件定義で記述されている取引終了後の状態に到達するかを算出でき，実行すべきワークフロー定義が導出できる．

## 5. 関連研究と今後の展開

現在，学術界や産業界において，プロセス実行ミドルウェアの研究は多くなされている[18][19][20][21]．特に，[21]において提案されているフレームワークは，ワークフロー定義や Web サービス定義を，RDF を基とした言語によってモデル化し，そのモデルを用いることにより，ワークフローの合成を実現している．すなわち段階 IV を実現するためのフレームワークであるといえる．しかし，本提案のように企業間取引の進化モデルに適応的であるかどうかという議論はなされていない．

本研究では，企業間取引において利用する取引相手の Web サービスやその利用順序といった振る舞い的側面のみを考慮している．しかし，セキュリティや性能などの非機能的側面も考慮する必要がある．特に，本提案における段階 III 以降においては，取引を開始する際に，実行時に決定する取引相手と本当に取引をして良いのかという信用を考慮して決定する必要がある．

## 6. まとめ

本稿ではプロセス実行ミドルウェアのためのアーキテクチャである EDMA を提案した．EDMA に対する要件は，企業間取引形態の進化に適応してミドルウェアを更新出来る点である．そのアーキテクチャを検討するため，企業間取引形態の進化を特徴的な 5 つの段階に整理し，その整理に基づき，構成するコンポーネントを追加することにより進化の段階に対応できるアーキテクチャを導いた．

## 7. 参考文献


[1] Erik Christensen et al., “Web Services Description Language 1.1”, http://www.w3.org/TR/wsdl, March 2001.

[2] Martin Gudgin et al, "SOAP Version 1.2 Messaging Framework", http://www.w3.org/TR/soap12-part1/, June 2003.

[3] David Booth et al, “Web Services Architecture”, http://www.w3.org/TR/2003/WD-ws-arch-20030808, August 2003.

[4] Workflow Management Coalition, “Terminology & Glossary”, WFMC-TC-1011, Feb. 1999, accessible from http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.

[5] David Hollingsworth, "The workflow reference model: 10 years on", The Workflow Handbook 2004, Future Strategies Inc., 2004, accessible from http://www.wfmc.org/standards/docs/Ref_Model_10_years_on_Hollingsworth.pdf.

[6] Tony Andrews et al, "Business Process Execution Language for Web Services (BPEL4WS) Version 1.1", http://www-106.ibm.com/developerworks/library/ws-bpel/, May 2003.

[7] Nickolaos Kavantzas et al, "Web Services Choreography Description Language (WS-CDF) Version 1.0", http://www.w3.org/TR/ws-cdl-10/, April 2004.

[8] Assaf Arkin, "Business Process Modeling Language(BPML)", http://www.bpmi.org/bpml_prop.esp, November 2002.

[9] Jari Veijalainen et al, "Research Issues in Workflow Systems", Proceedings of 8th ERCIM Database Research Group Workshop, October 1995, accessible from http://www.ercim.org/publication/ws-proceedings/8th-EDRG/veijal.ps.

[10] Raymond Reiter, “Knowledge in Action”, ISBN: 0262182181, MIT Press, 2001.

[11] Workflow Management Coalition, "Workflow Management Coalition Terminology & Glossary", WFMC-TC-1011 in WfMC public documents, Feb. 1999, accessible from http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.

[12] Roberto Chinnici et al., "Web Services Description Language Version 2.0 Part 1: Core Language", http://www.w3.org/TR/wsdl20/, March 2004.
[13] M. Sloman et al, "An Architecture for Managing Distributed Systems", Proc. 4th IEEE Workshop on Future Trends of Distributed Computing Systems, 40--46, 1999.
[14] Tom Bellwood et al, "UDDI Version 3.0", UDDI.org, July 2002.
[15] T. Hatashima et al, "Web Services Processing Platform - eCo-Flow", SAINT 2002, Workshop, Jan. 2002, pp. 186-195.
[16] Ian Jones et al, “ebXML Message Service Specification Version 2.0”, OASIS ebXML Messaging Services TC, Feb. 2002.
[17] Yuji Sakata et al, "A Method for Composing Process of Non-deterministic Web Services", Proceedings of 2004 IEEE International Conference on Web Services, July 2004.
[18] Keith Levi and Ali Arsanjani, “A Goal-driven Approach to ENTERPRISE COMPONENT IDENTIFICATION AND SPECIFICATION”, COMMUNICATIONS OF THE ACM Vol. 45 No. 10 p45, Oct. 2002.
[19] Aniruddha Gokhale et al., “Applying Model-Integrated Computing to COMPONENT MIDDLEWARE AND ENTERPRISE APPLICATIONS”, COMMUNICATIONS OF THE ACM Vol. 45 No. 10 p 45, Oct. 2002.
[20] 勝間田 仁, 世古 将洋, 藤井 義真, 速水 治夫, "組織間ワークフローのモデリング支援機構の開発と検証", 情報処理学会論文誌 Vol.43 No.11 pp.3328-3342.
[21] Lerina Aversano et al., “Introducing eServices in Business Process Models”, the fourteenth International Conference on Software Engineering and Knowledge Engineering, July 2002.