SOA解説

ＳＯＡによる情報システム構築方法
プロセス統合編

2012.4
April

＜本書で使用する用語＞

AP	:	Application	MDB	:	Message Driven Bean
API	:	Application Program Interface	MF	:	MainFrame
BPEL	:	Business Process Execution Language	MPN	:	Multi Payment Network
BPM	:	Business Process Management	MVC	:	Model-View-Controller
BPMN	:	Business Process Modeling Notation	RPC	:	Remote Procedure Call
CAFIS	:	Credit And Finance Information System	SOA	:	Service Oriented Architecture
EA	:	Enterprise Architecture	SPP	:	Service Providing Program
EAR	:	Enterprise Archive	TP	:	Transaction Processing
ESB	:	Enterprise Service Bus	UML	:	Unified Modeling Language
GW	:	Gateway	WFA	:	Web-Flow Architecture
I/F	:	Interface	WSDL	:	Web Services Description Language
JSF	:	JavaServer Faces	XML	:	Extensible Markup Language
JSP	:	JavaServer Pages			

＜本書で紹介している日立製品＞

製品名		概要
COBOL2002	コボル 2002	COBOL アプリケーションの開発実行プラットフォーム
COBOL2002 for .NET Framework	コボル 2002 フォー ドットネット フレームワーク	Microsoft .NET Framework 上で稼動する COBOL 開発環境
JP1/AJS3*	ジェーピーワン エージェーエススリー	ジョブ管理
uCosminexus OpenTP1	ユーコズミネクサス オープンティーピーワン	オンライントランザクション処理
TP1/COBOL adapter for Cosminexus Version 2	ティーピーワン コボル アダプタ フォー コズミネクサス バージョン ツー	OpenTP1 の COBOL プログラムを Web アプリケーションから利用
uCosminexus Application Server	ユーコズミネクサス アプリケーションサーバ	Web アプリケーションサーバ
uCosminexus Developer	ユーコズミネクサス ディベロッパー	Web アプリケーションの開発環境
uCosminexus Service Architect	ユーコズミネクサス サービス アーキテクト	ビジネスプロセス定義、データ変換定義、サービスアダプタ定義等を行う開発環境
uCosminexus Service Architect – WorkCoordinator	ユーコズミネクサス サービス アーキテクト ワークコーディネーター	対話ワークフローの開発環境
uCosminexus Service Platform	ユーコズミネクサス サービス プラットフォーム	BPEL に準拠したビジネスプロセス制御機能とエンタープライズサービスバス機能
uCosminexus Service Platform – WorkCoordinator	ユーコズミネクサス サービス プラットフォーム ワークコーディネーター	人が介在する業務を統合する対話ワークフロー

*) JP1/AJS3： JP1/Automatic Job Management System 3 の略称

本ホワイトペーパーの想定読者

SOA による情報システムの構築方法についてまとめたものであり、情報システムの設計・開発に携わる情報システム部門の方々を想定しています。

概要

企業のビジネス活動に情報システムが果たす役割が増大する中で、情報システムに期待することは何でしょうか。単に必要な業務を遂行するだけではありません。ビジネスニーズの変化へ対応するために、すぐにシステムを変更できることや、企業間競争を勝ち抜くために業務を効率化できたり、ビジネス活動の改善点を見出すための情報を提供するといったことが、情報システムに求められているのです。

このような期待に応える情報システムを実現する手段が SOA(Service Oriented Architecture)です。SOA は、情報システムを構築する手法、およびアーキテクチャの考え方であり、その特長は「サービス」というソフトウェアコンポーネントの考え方を導入し、情報システムを「サービス」の組み合わせによって構築することです。

日立は、SOA を用いたシステム構築をトータルに支援するため、要件定義から実装に至るシステム開発で、SOA を用いたシステム開発の考え方と手順を定義した SOA システム構築手法と、各種のシステムパターンを実現できる SOA システム構築基盤を提供しています。日立の SOA 構築手法は、業務の流れからビジネスプロセスを階層化してサービスを抽出することによって、ビジネスプロセスとサービスの設計を支援することを特長としています。

SOA システム構築基盤は、豊富なシステム構築ノウハウを組み込んだ各種機能を提供しています。サービス間連携を実現するための業界標準の機能はもとより、サービスに求められる様々なシステムパターンを実現できる機能を提供します。

日立の SOA システム構築手法・システム構築基盤は、SOA の実践に真に必要な観点を的確に捉えたものであり、情報システムをさらにビジネスへ貢献させることができます。

- uCosminexus Service Platform、uCosminexus Service Architect、uCosminexus Application Server、uCosminexus Developer は、経済産業省が 2003 年度から 3 年間実施した「ビジネスグリッドコンピューティングプロジェクト」の技術開発の成果を含みます。
- 記載している仕様、デザインは予告なしに変更することがあります。あらかじめご了承下さい。

Contents

1. SOAとは

1.1. SOAの狙い

（1）情報システムの現状と SOA への期待

企業のビジネス活動へのITの活用が進展しています。情報システムなくして、企業活動は成り立たない時代になったと言ってよいでしょう。その一方で、消費者ニーズの多様化、市場のグローバル化、規制緩和、コンプライアンス意識の高まりなど、ビジネス環境の変化や競争は益々激しさを増しています。このため情報システムには、ビジネス活動の変化や迅速化を支えることが期待されています。

しかし、多くの情報システムは、この期待に応えるのが難しい状況にあるのではないでしょうか。例えば、情報システムがメインフレーム上でモノリシック(システム全体が一つに統合されている形態)に実装され、システムを構成する様々なプログラムの独立性や共通化が不十分なままで、プログラムの変更が繰り返される状況に陥っています。また、部門ごとの業務へのシステム最適化を遂行した結果、情報システムが複数の個別システムに分断されて、システムを構成するプログラム間の構成や関係が統制されないまま、システムごとに個別にプログラムの追加や修正を実施しているという状況もあります。このような状況から、次の課題が考えられます。

- ビジネスニーズに応じた変更や拡張をしようとしても、影響範囲の特定が難しく、改修が広範囲に及んでしまう
- 各システムの個別のユーザインタフェースを使って、複数システムを人手で連携させている場合が多く、ビジネス活動が非効率になっていたり、情報システム活用が属人的ノウハウに依存する事態になっている
- ビジネス活動の改善点を探るための的確な情報を入手することが困難

このような情報システムの課題を解決する手段として期待されるのが SOA です。SOA は、情報システムを構築する手法とアーキテクチャの考え方です。その特長は、「サービス」というソフトウェアコンポーネントの考え方を導入し、情報システムを「サービス」の組み合わせによって構築することです。

（2）SOA の狙いと実現の考え方

SOA の狙いとその実現の考え方を、図 1－1 に、a～c の三項目に分類して示します。

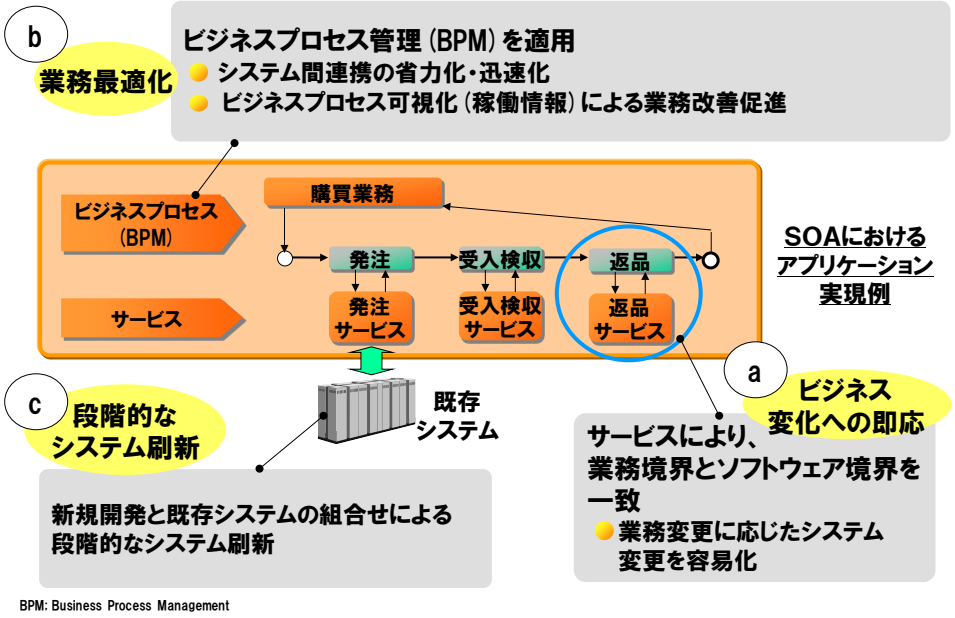


図 1－1　SOA の狙いと実現の考え方

(a) 情報システムをビジネス変化へ即応させる。

SOA の第一の狙いは、ビジネスニーズに応じてビジネス活動を変化させていくために、情報システムの変更を迅速にできるようにすることです。

このためにSOAでは、ビジネス活動(ビジネスプロセス)を構成する業務と一対一に対応するソフトウェアコンポーネントを「サービス」という情報システムの構築単位として使用します。これによって、次のように情報システムの課題を解決できます。

(b) 業務の境界とソフトウェアの境界を一致させられるため、ビジネス変化の発生個所と情報システムでの変更発生個所の対応を把握することが容易になり、システムの改修範囲が局所化され、ビジネスニーズに応じた変更や拡張が迅速にできるビジネスプロセス管理を用いて業務の効率化や最適化を行う。

SOA の第二の狙いは、情報システムをよりビジネス活動に役立てるために、システム間の連携をより効率化・迅速化したり、情報システムに内在するビジネス活動の状況などを表す情報を可視化したりすることです。

このために SOA では、一般にビジネスプロセス管理(BPM: Business Process Management)と呼ばれるシステムを構築・運用するための手法・基盤を適用します。BPM ではIT 化対象のビジネスプロセスを構成するサービスをビジネスプロセスの流れに従って組み合わせ、そのビジネスプロセスを実現するシステムを構築します。これによって、次のように情報システムの課題を解決できます。

- 人手で実現していたシステム間連携を、省力化・迅速化できる
- ビジネスプロセスの稼働情報を統一された形式で収集できることで、ビジネス活動の状況を可視化し、ビジネス活動の改善を促進できる

(c) 情報システムを段階的に刷新する。

SOA の第三の狙いは、段階的なシステム刷新を実現することです。(a)と(b)で説明したサービスと BPM の導入による情報システムの最適化を、より優先度の高いところから部分的に着手していけるようにすることです。

このため SOA では、BPM でのサービス仮想化を重視します。上位業務のビジネスプロセスにとっては、下位業務のサービスが新規システムか既存システムかの区別は意識されません。これによって、IT 投資計画の優先度や積極的に残したい IT 資産などの観点で、刷新する部分と既存のまま残す部分を選別し、段階的に情報システムの刷新ができます。

このように SOA は、「サービス」という考え方とビジネスプロセス管理を情報システムに導入することによって、本来の業務モデルに従った情報システムを全体最適化することを狙いとしています。一方で、全体最適化を一気に実施するのではなく、段階的に情報システムを刷新していく手段を併せて提供しています。

（3）システム開発からみた SOA の効果

SOA には、情報システムの開発や保守の効率を向上する効果があります。システム開発からみた SOA の効果を図 1－2 に三項目に分類して示します。

(a) サービスの再利用を促進できる。

ある業務のために開発したプログラムを他の業務へ再利用することで、情報システム開発コストを削減できますが、SOA を用いることで、このような再利用を促進できます。SOA を用いたシステム構築では、ビジネスプロセスに着目してシステム設計します。そのため、ここで新規に IT 化するビジネスプロセスに対して、既に IT 化済みのビジネスプロセスの中から、類似のものを見出すことで、既存ビジネスプロセスで使用されているサービスの再利用の検討を促進できます。

(b) システム変更の局所化により保守性を向上できる。

サービスの導入で、業務の境界とソフトウェアコンポーネントの境界を一致させることで、業務の変更に応じたソフトウェアの変更箇所の特定を効率的にできます。また、サービスを適切な粒度で設計することで、サービスの独立性を高めて、ソフトウェア変更をサービス内に局所化できます。これらによって、情報システムの保守性を向上できます。

(c) 既存システムの活用により開発コストを削減できる。

既存システムをサービスとして活用することで、新規開発部分を減らし、開発コストを削減できます。

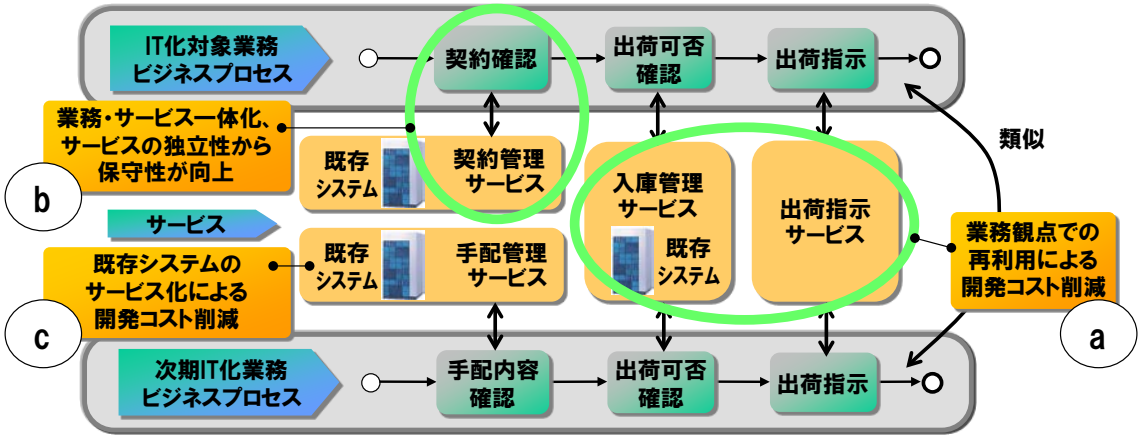


図 1－2　システム開発からみた SOA の効果

1.2.　SOAによる情報システム構築の考え方

SOA による情報システム構築の基本的な考え方を図 1－3 に示します。

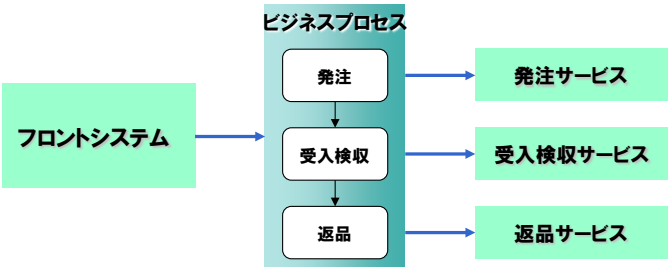


図 1－3　SOA による情報システム構築の基本的な考え方

サービスとは、「発注」、「受入検収」、および「返品」といった業務の境界とソフトウェアの境界を一致させたものであり、ビジネス活動を構成する業務を実現するために必要な機能とインタフェース(サービス I/F)を持つソフトウェアコンポーネントのことを指します。サービスは、サービス I/F を介してメッセージで入出力データを授受します。また、これ以降、ビジネス活動の中で BPM により IT 化される範囲のことを特にビジネスプロセスと呼びます。SOA におけるビジネスプロセスは、複数のサービスを組み合わせて連携させた一連の処理の流れを定めたものです。ビジネスプロセスは、サービス I/F を介してメッセージ授受をすることで、サービスと連携するとともに、サービス間のメッセージの流れを定義します。またビジネスプロセス自身も連携先のサービスを組み合わせた複合的なサービスとなり、サービス I/F を持ちます。

これらのビジネスプロセスやサービスは、システムの設計段階で、業務の流れからビジネスプロセスを段階的に階層化していき、その中で抽出したサービスをソフトウェアコンポーネントに対応させたものになります。また、サービスを利用するソフトウェアコンポーネントのことをフロントシステムと呼び、ビジネスプロセスを実行する役割を持ちます。

SOA を使用したシステム構築で、フロントシステムとサービスの実現方法のバリエーションを分類したものをSOAシステムパターンと呼びます。システムの要件に応じて、フロントシステムとサービスの最適なパターンを組み合わせてシステムを実現できます。SOA システムパターンの全体像を図 1－4、システムパターンとその内容を表 1－1 にまとめました。

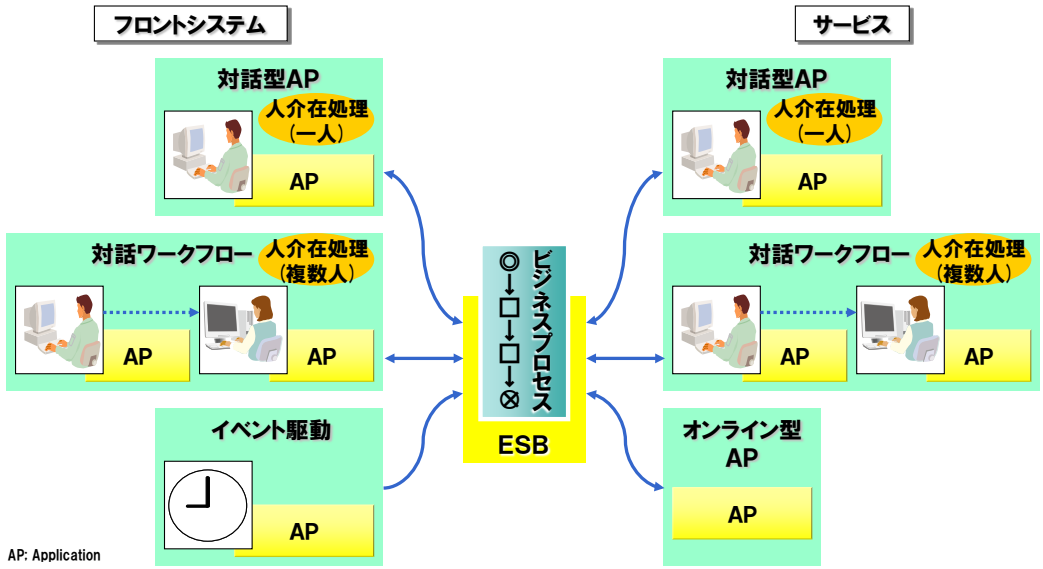


図 1－4　SOA システムパターン

表 1－1　SOA システムパターン

#	分類	パターン	内容
1	フロントシステム	対話型アプリケーション	一人の作業者がアプリケーションを用いて業務を遂行する構成であり、当該アプリケーションは画面インタフェースにより作業者との間で入出力を授受し、サービスとの連携を実施する。
2		対話ワークフロー	複数の作業者がそれぞれアプリケーションを用いて一連の業務を遂行する構成であり、複数の対話型アプリケーションを組み合わせた実現形態となる。
3		イベント駆動	時刻やファイル到着といったイベントに応じて、サービスを駆動する形態である。
4	サービス	ビジネスプロセス	ビジネスプロセスにより複数のサービスを組み合わせてサービスを実現する。サービス連携やメッセージ処理の流れは、SOA におけるビジネスプロセス定義言語の業界標準である BPEL（Business Process Execution Language)を用いて実現する。
5		オンライン型アプリケーション	アプリケーションが問合せ応答処理によりサービスを実現する形態である。呼出元からの要求に応じてプログラム処理を実行して応答を返す(応答がない形態も含む)。例えば Web サービスで実現するサービスは当パターンである。
6		対話型アプリケーション	#1 と同様に、一人の作業者がアプリケーションを用いて業務を遂行する構成である。当該アプリケーションは画面インタフェースにより、作業者との間で入出力を授受するとともに、ビジネスプロセスとの連携のためのサービスI/F を備える。
7		対話ワークフロー	#2 と同様に、複数の作業者がそれぞれアプリケーションを用いて一連の業務を遂行する構成である。複数の対話型アプリケーションを組み合わせた実現形態となるが、アプリケーションの一部がビジネスプロセスとの連携のためのサービス I/F を備える。

フロントシステムやサービスは、アプリケーションプログラム(以降、「アプリケーション」と呼びます)によって実現されます。サービスには、その実現形態として次のバリエーションがあります。

・プログラム処理によってサービスに必要な処理を実現するオンライン型アプリケーションパターン

・人が介在してサービスを実現するパターンとして一人が処理を行う対話型アプリケーションパターン

・複数人が連携して処理を行う対話ワークフローパターン

また、ビジネスプロセスによって複数のサービスを組み合わせるパターンがあります。同様にフロントシステムにも実現形態として、いくつかのバリエーションがあり、人が介在する対話型アプリケーションパターンや対話ワークフローパターン、時刻やファイル到着などのイベントに応じて実行するイベント駆動パターンがあります。

また SOA を使用したシステム構築では、フロントシステム、ビジネスプロセス、サービスの間の連携をESB(Enterprise Service Bus)を介して実現します。ESB は、呼出元とサービスを疎結合させるミドルウェアであり、サービスの実装や所在の変化に対して、呼出元への影響を極小化する役割を担います。

図 1－4 で示した SOA システムパターンを組み合わせたシステム構築の例を、次に挙げます。

（1）サービスオーダ管理業務への適用例

通信業でのサービスオーダ管理業務に SOA を適用した例を図 1－5 に示します。

この図は、フロントシステムに対話型アプリケーションパターン、サービスにオンライン型アプリケーションパターンを適用した例です。顧客管理サービスや工事管理サービスなど複数のオンライン型アプリケーションをビジネスプロセスを使用して組み合わせることによって、ワンストップサービスを提供できます。顧客が申し込み Web 画面から必要な情報を入力することで、顧客システムの設定から料金の設定まで一連の流れを自動化できます。

（2）受注出荷業務への適用例

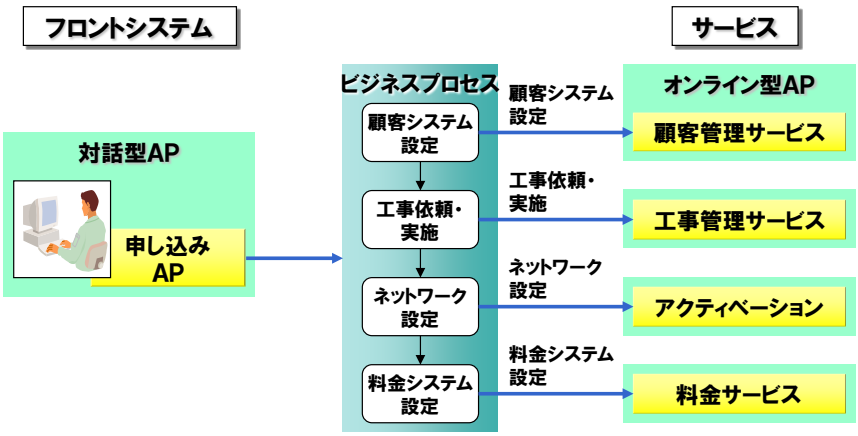


図 1－5　サービスオーダ管理業務

製造業での受注出荷業務に SOA を適用した例を図 1－6 に示します。

この図は、フロントシステムに対話ワークフローパターン、サービスにオンライン型アプリケーションパターンを適用した例です。フロントシステムとして、受注・出荷を行う作業者間で、各作業者が用いているアプリケーションを連携させます。次に、出庫管理サービスや配送管理サービスへのアクセスをビジネスプロセスで自動化します。これらによって、受注・出荷業務の効率化を図れます。

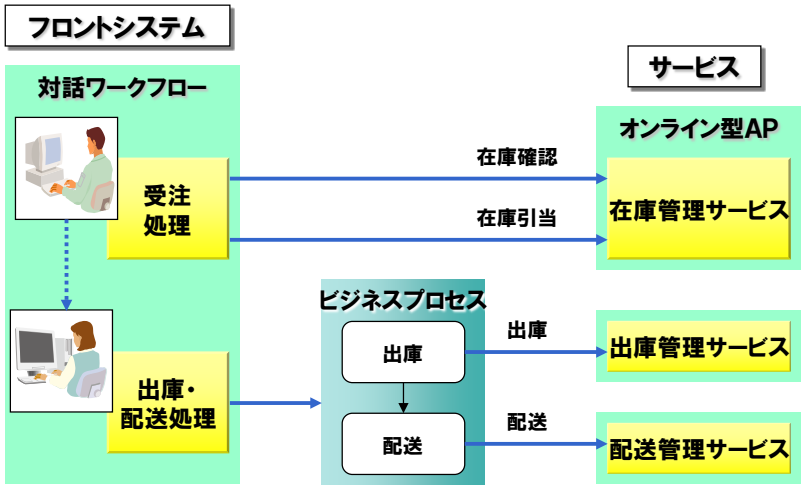


図 1－6　受注出荷業務

1.3.　SOAによる情報システム構築の流れ

SOA による情報システム構築の流れを図 1－7 に示します。構築を進めるにつれて、構築の流れは大きく二つに分かれます。

- ビジネスプロセス設計・開発:ビジネスプロセスの設計を行い、ビジネスプロセスパターンを実装します。設計時の記法として、業界標準の BPMN(Business Process Modeling Notation)を用いることができます。
- フロントシステム・サービス設計・開発:フロントシステムやサービスを実現するアプリケーションの設計を行い、対話型アプリケーションパターンやオンライン型アプリケーションパターン、対話ワークフローパターンを用いて実装します。設計時の記法として、業界標準の UML(Unified Modeling Language)を使用できます。

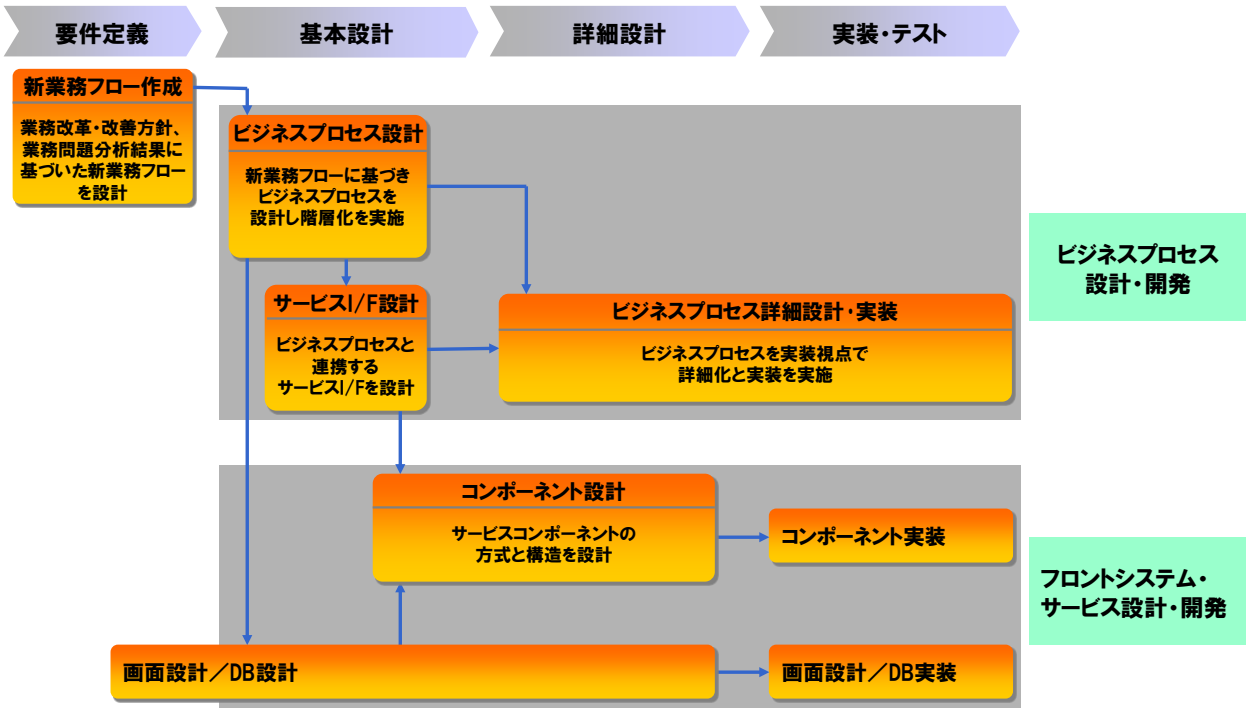


図 1－7　SOA による情報システム構築の流れ

1.4. 日立SOAソリューション

日立は、SOAによる情報システム構築を支えるために、SOAによるシステム構築の流れを実現するSOAシステム構築手法と、各種のSOAシステムパターンを実現するミドルウェアであるSOAシステム構築基盤を提供しています。

1.4.1 SOAシステム構築手法

日立のSOAシステム構築手法は、図 1－7に示したSOAによる情報システム構築の流れを実践するためのもので、SOAによるシステム設計の考え方や手順を定めています。日立は、従来から各種のシステム開発事例で実績のある情報システム構築手法があり、SOA システム構築手法はこの拡張として整備しています。当手法では、アプリケーション開発の従来からの構成要素である画面・帳票やデータベースの設計と整合性を確保できるようになっています。

1.4.2 SOAシステム構築基盤

日立は、SOAシステム構築基盤として、図 1－8 および表 1－2 に示すように、各SOAシステムパターンに応じたミドルウェア製品をそれぞれのための開発ツールと共に提供しています。各ミドルウェアや開発ツールについては3章から5章で説明します。

SOA システム構築基盤は、各種オープンソースや著名な上流設計ツールと組み合わせて、情報システム構築に適用できます。uCosminexus Service Architect や uCosminexus Developer を使用するときには、オープンソースの開発ツールとして広く開発者に支持されている Eclipse を基盤にできます。また、uCosminexus Application Serverでのアプリケーション開発では、Strutsなどのオープンソースの著名フレームワークを使用できます。

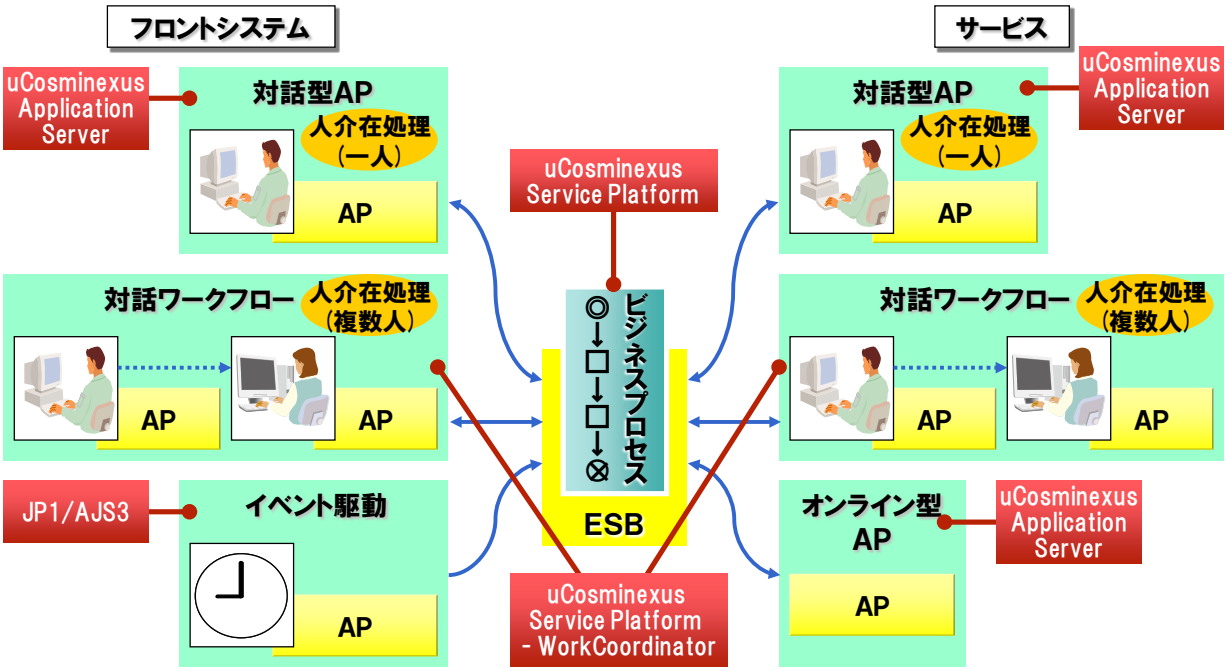


図 1－8 日立SOAシステム構築基盤

一方、基本設計工程においては、BPMN や UML を用いた設計に対して、著名な上流設計ツールを利用できます。

表 1－2　日立 SOA システム構築基盤

#	分類	パターン	製品名	概要
1	ビジネスプロセスの開発	ビジネスプロセス	uCosminexus Service Platform	サービス間を統合するビジネスプロセスを制御する機能と ESB 機能を提供する。開発ツールとして uCosminexus Service Architect を併せて提供する。
2	フロントシステム・サービスの開発	オンライン型アプリケーション	uCosminexus Application Server など	オンライン形態のプログラム処理を制御する。開発ツールとして uCosminexus Developer を併せて提供する。
3		対話型アプリケーション	uCosminexus Application Server など	作業者の画面操作と、それと連動するプログラム処理を制御する。開発ツールとして uCosminexus Developer を併せて提供する。
4		対話ワークフロー	uCosminexus Service Platform － WorkCoordinator	フロントシステム・サービス内の対話型アプリケーション間を連携させる対話ワークフローを制御する。開発ツールとして uCosminexus Service Architect － WorkCoordinator を併せて提供する。
5		イベント駆動	JP1/AJS3	時刻やファイル到着といったイベントによって駆動されるプログラム処理を制御する。

1.4.3　PDCAサイクルを支える日立SOAソリューション

日立は、情報システムに SOA を適用するためのソリューションとして、SOA システム構築手法や SOA システム構築基盤だけでなく、IT に関する戦略策定から、戦略実装、戦略実行、そして戦略評価に至る PDCA サイクルを実施できる各種のサービスやミドルウェアを提供しています。日立 SOA ソリューションの全体像を図 1－9 に示します。

戦略策定フェーズは、ビジネスニーズを踏まえた情報システムに関する戦略策定などを行うフェーズです。業務分析や EA(Enterprise Architecture)の考え方を用いた IT 最適化などに関するコンサルティングサービスを提供します。

戦略実装フェーズ・戦略実行フェーズでは、情報システムの設計・実装から稼働・運用を行うフェーズです。これらのフェーズに対しては、SOA の考え方に基づきシステム構築を支援する各種コンサルティングサービスとともに、SOA システム構築基盤や IT を負荷・障害等の各種変動に即応させるための統合プラットフォームなどのミドルウェア製品を提供します。特に、サービス指向システム構築支援コンサルティングサービスでは、SOA システム構築手法を適用したシステム構築についてコンサルティングを行います。

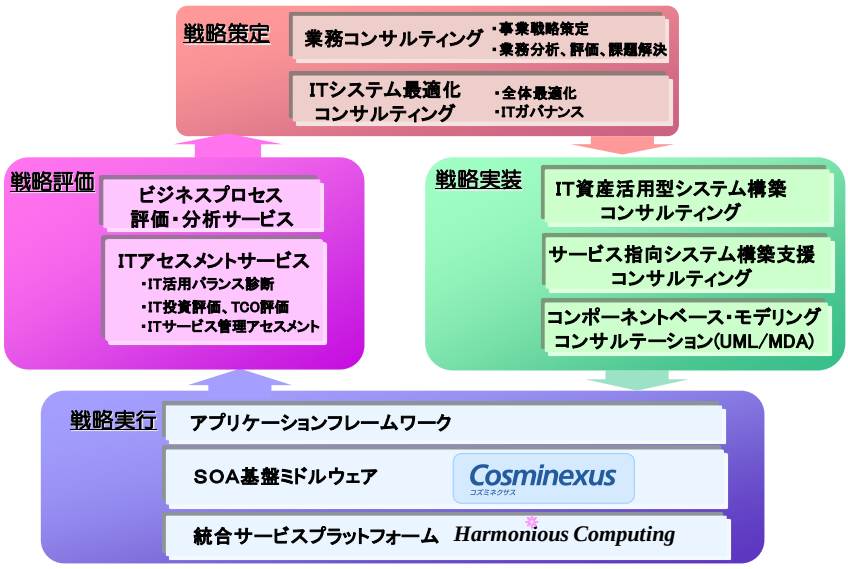


図 1－9　日立 SOA ソリューションの全体像

また、戦略評価フェーズ向けには、IT 投資評価や構築した情報システムを通じてビジネス活動の分析・評価を支援するコンサルティングサービスを提供しています。

2. SOAシステム構築手法

本章では、SOA システム構築手法における基本的な考え方を 2.1 節で説明し、2.2 節以降で開発の全体的な流れを紹介します。

2.1. 概要

ビジネスプロセスとは、ビジネス活動のIT化された範囲について、一連の業務の流れを定めたものです。上位の業務は、下位の業務の組み合わせによって業務内容を明確にできます。このことを業務の階層化と呼びます。階層化された下位の業務の組み合わせもビジネスプロセスで表現でき、ビジネスプロセスの階層化が可能となります。

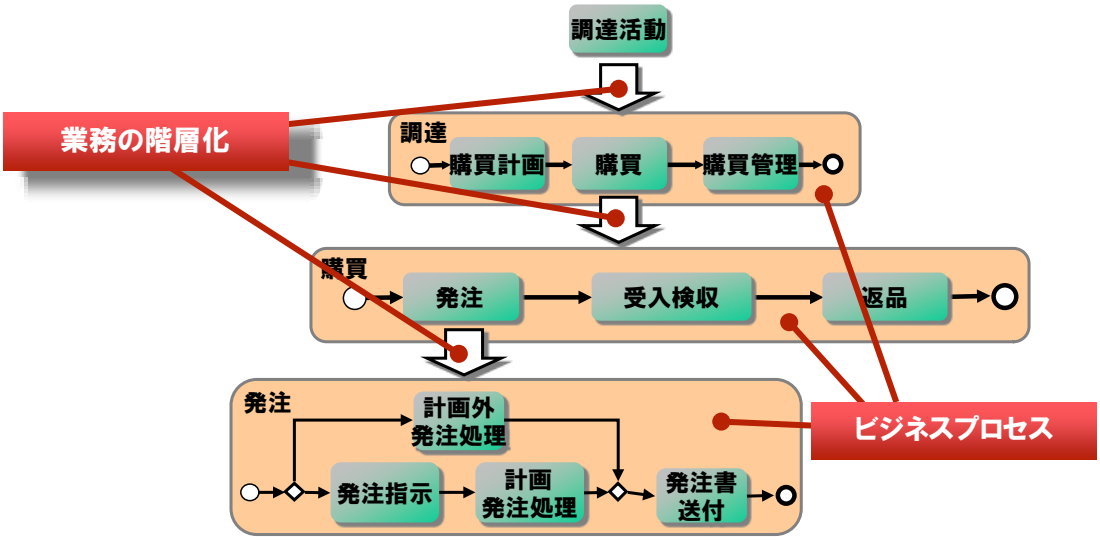


図 2－1　ビジネスプロセスと業務の階層化

サービスは、1.2 節で示すようにビジネス活動を構成する業務と一対一に対応するソフトウェアコンポーネントと定義されます。

図 2－2 は、一連の業務（アクティビティ）の流れを示すビジネスプロセスとサービスの関係を表現したものです。サービスにはアクティビティに機能を提供するためのサービス I/F があり、このサービス I/F を通じてビジネスプロセスと連携します。サービス I/F はサービスの顔であり、サービスが提供できる機能や必要な入出力データを定義する役割を持っています。

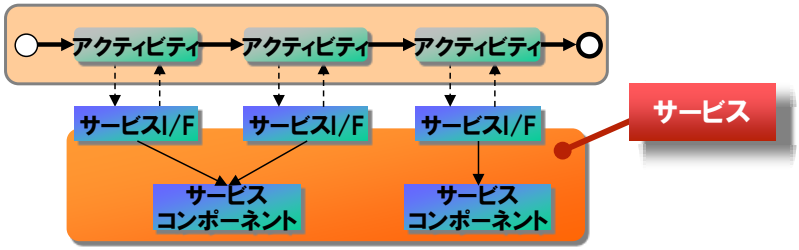


図 2－2　サービスの構造

サービスが提供する機能の実装は、サービスコンポーネントが担います。サービスコンポーネントを使用したサービスの実現形態には、1.2 節の SOA システムパターンで示したような、対話型アプリケーションパターンや対話ワークフローパターン、オンライン型アプリケーションパターンなどがあり、導入方法として新規開発や既存システムの再利用、パッケージ導入などの方法が利用できます。

アプリケーションには、ビジネスプロセスを起動する側のアプリケーションもあり、ビジネスプロセスを通じてサービスを利用することから、フロントシステムと呼ぶことがあります。またサービスコンポーネント内で、さらに別のビジネスプロセスを呼び出すような多階層の形態も考えられます。

2.2. システム構築手法の全体像

SOA システム構築手法全体の流れ図を図 2－3 に、全体の工程を表 2－1 に示します。

本手法では、ビジネスプロセスに着目してサービスの抽出をした上で、サービスを実現するコンポーネントの設計へ進める点に特長があります。

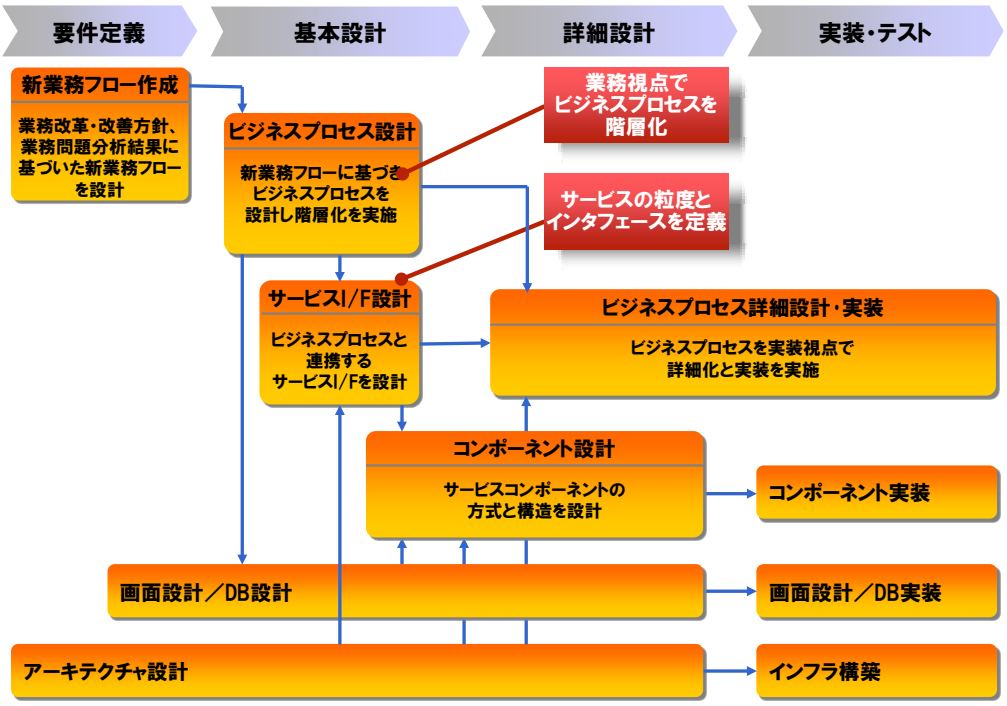


図 2－3　SOA システム構築手法の全体像

表 2－1　SOA システム構築手法を構成する設計フェーズの概略

#	区分	設計作業名	内容
1	共通	新業務フロー作成	システム化計画に基づく新業務フローを設計 ・サービス粒度を概ね決定 ・サービス開発方針の検討(既存システム利用、パッケージ利用など)
2		アーキテクチャ設計	非機能要件の洗い出し 非機能要件に基づくシステム構成の概略設計 システム方式の設計
3	ビジネスプロセス	ビジネスプロセス設計	業務フローからビジネスプロセスを設計 ・業務の観点によるビジネスプロセスの階層化を実施 ・基本フローと業務のバリエーションを明確化し、サービス粒度を見直す ・ユースケースシナリオの定義
4		サービス I/F 設計	サービス粒度の決定とサービス I/F の決定
5		ビジネスプロセス詳細設計・実装	BPEL、WSDL、XML といった業界標準を用いてビジネスプロセスを詳細化および実装を実施
6		ビジネスプロセスデバッグ	ビジネスプロセスデバッガを使用してビジネスプロセスのデバッグを実施
7	アプリケーション	画面設計	作業者の業務遂行に必要な画面、画面遷移を設計
8		DB 設計	必要なデータの分析、エンティティとその関連の抽出、データ論理仕様を設計
9		コンポーネント設計・実装	フロントシステム・サービスを実現するアプリケーションの設計と実装

次に、各作業について説明します。

（1）要件定義工程

要求内容を適切に定義する重要な工程です。

- **新業務フロー作成**：システム化計画に基づき、機能要件の洗い出しを行い、新業務フローを設計します。そのとき、システム化対象の業務の機能に対して、サービス候補を概ね決めておきます。粒度の精査は後工程で見直す方法をとります。また、この工程で各サービスの開発方針を決定します。新規開発、既存システムの再利用、パッケージソフトの導入等を方針として決定します。
- **アーキテクチャ設計**：対象システム全体のアーキテクチャを設計します。要件定義工程では、システム化計画に基づき、非機能要件の洗い出しを行い、アーキテクチャ概要を設計します。そのとき、新業務フロー作成でサービス候補として挙げられたサービスの開発方針を考慮し、既存システムとの連携方式、パッケージソフトとの連携方式を検討します。設計工程としては、図 2－3 に示すように、基本設計工程、詳細設計工程にまで及び、非機能要件に対応するシステム方式の設計を行います。

（2）基本設計工程

定義された要件を実現するビジネスプロセス、及びフロントシステム・サービスの機能を設計します。

【ビジネスプロセスの基本設計】

- **ビジネスプロセス設計**：ビジネスプロセスを段階的に階層化し、サービスの粒度を見直します。新業務フロー作成フェーズで定義された業務フローを元に、BPMN 記法に変換し初期のビジネスプロセス（基本フロー）を設計します。続いて、業務的な観点でビジネスプロセスの階層化を行います。ビジネスプロセスの階層化で洗い出された業務のバリエーションにより、サービスの粒度を見直します。さらにフロントシステムやサービスに求められる振る舞いを示すユースケースシナリオ（機能の順序を定義）を作成します。このユースケースシナリオは、後述するコンポーネント設計や画面設計・DB 設計、サービスの実現形態の選択に使用します。
- **サービス I/F 設計**：実装の観点でビジネスプロセスを見直し、メッセージとして授受するデータの明確化などを行い、サービス I/F を決定します。このサービス I/F 設計でサービスの粒度の最終決定をします。また、図 1－4 の SOA システムパターンで示した、対話ワークフローパターン、対話型アプリケーションパターン、またはオンライン型アプリケーションパターンに対応したサービス I/F の設計をします。

【フロントシステム・サービスの基本設計】

- **コンポーネント設計**：フロントシステム・サービスを実現するアプリケーションを設計します。プレゼンテーション層（画面関係）、ファンクション層（業務機能）、データ層の三層ごとに、内部的なコンポーネントを設計します。基本設計工程では、これら三層間の関係がユースケースシナリオに沿った制御の流れ、データの流れになるように設計します。
- **画面設計**：フロントシステムやサービスで、作業者の業務遂行に必要な画面、および画面遷移の設計をします。画面設計は、開発工程全体にわたって設計を継続します。要件定義工程では、画面のラフレイアウトで、エンドユーザの要件洗い出しをします。また、基本設計工程では、作成されたユースケースシナリオを元に、対話アプリケーションにおける画面設計、画面遷移設計をします。さらに詳細設計工程では、HTML、JSP などを対象とした実装のための設計をします。
- **DB 設計**：フロントシステムやサービスで扱われるデータの分析から、DB テーブルの設計までをします。DB 設計は、画面設計と同様に、開発工程全体にわたって設計を継続します。要件定義工程では、データの分析、キー項目の抽出によって、対象システム全体で扱うデータ関連の把握をします。また、基本設計工程では、作成されたユースケースシナリオを元に、必要なデータの分析、エンティティとその関連を抽出し、データ論理設計をします。さらに、詳細設計工程では、物理テーブルを対象としたデータ物理設計を行います。

（3）詳細設計・実装工程

ビジネスプロセス、フロントシステム・サービスについて、実装のための詳細な仕様を設計し、実装します。

- **ビジネスプロセス詳細設計・実装**：ビジネスプロセス、サービス I/F の各種基本設計仕様を元に、実装のためのビジネスプロセス定義、サービスが授受するメッセージの詳細定義などの設計を行います。設計・実装の言語としてはビジネスプロセスについてはBPEL、サービスI/F およびメッセージについては、それぞれ、WSDL(Web Services Description Language)、XML(Extensible Markup Language)を使用します。
- **コンポーネント設計・実装**：基本設計工程のコンポーネント設計で設計されたプレゼンテーション層、ファンクション層、データ層の三層ごとに詳細な設計をします。Struts、JSF(JavaServer Faces)など、業界標準の各種フレームワークの使用を前提とした設計ができます。

詳細設計工程および実装工程では、表 1－2 で示したミドルウェア製品を用いて、ビジネスプロセス設計・開発、フロントシステム・サービス設計・開発ができます。

2.3. サービス抽出の考え方

サービス抽出とは、サービスが提供する機能の洗い出しと、サービスの粒度（提供機能が及ぼす範囲）を決定することをいいます。SOA システム構築手法では、サービスの粒度を各設計工程で段階的に設計する点を特長としています。

- 要件定義：サービス初期抽出をして、サービスの候補として位置づけます。
- ビジネスプロセス設計：ビジネスプロセスの階層化によって、サービス粒度の見直しをします。
- サービス I/F 設計：サービスの粒度の決定とサービス I/F の定義をします。

例えば、図 2－4 で示すように、要件定義工程の段階では、作成した業務フローに基づき、業務フローを構成する各業務をサービスの候補として位置づけます。これは、あくまで候補であって、以降の工程においてより適切な粒度に見直しをします。

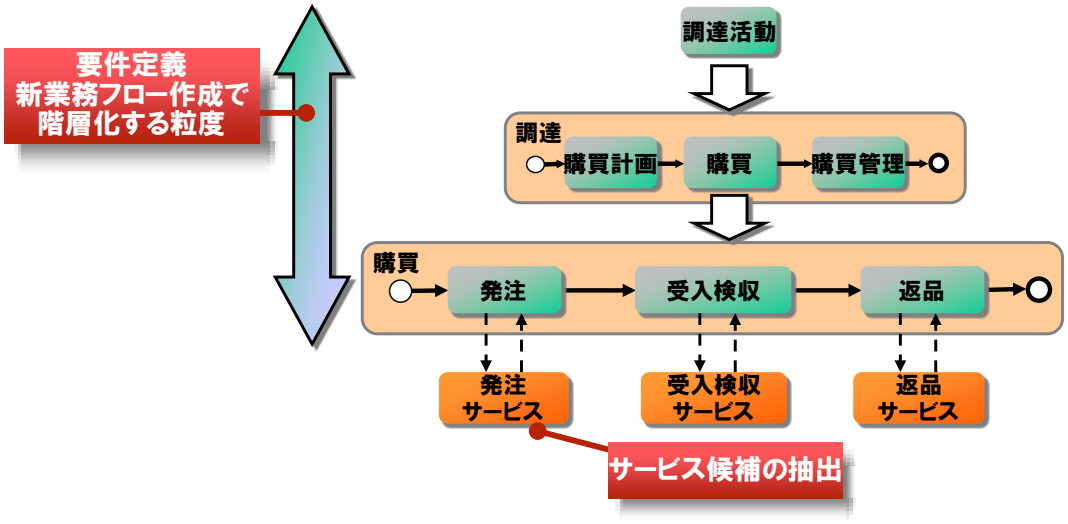


図 2－4　要件定義工程でのサービス候補抽出

サービス粒度の見直しは、主に業務の実施方法のバリエーションごとに、サービスを区分するかどうかを判断することで実施します。図 2－5 にサービス粒度の見直しの例を示します。「発注」業務をさらにビジネスプロセスで表現して階層化することで、業務のバリエーションが洗い出されます。ここでは、発注業務のバリエーションとして、計画発注業務、計画外発注業務というバリエーションが抽出されています。これらの業務のバリエーションに対応した粒度でのサービスは、業務の変化を局所化する単位として、より適切な場合があり、この粒度をサービスとするかどうかを決定します。

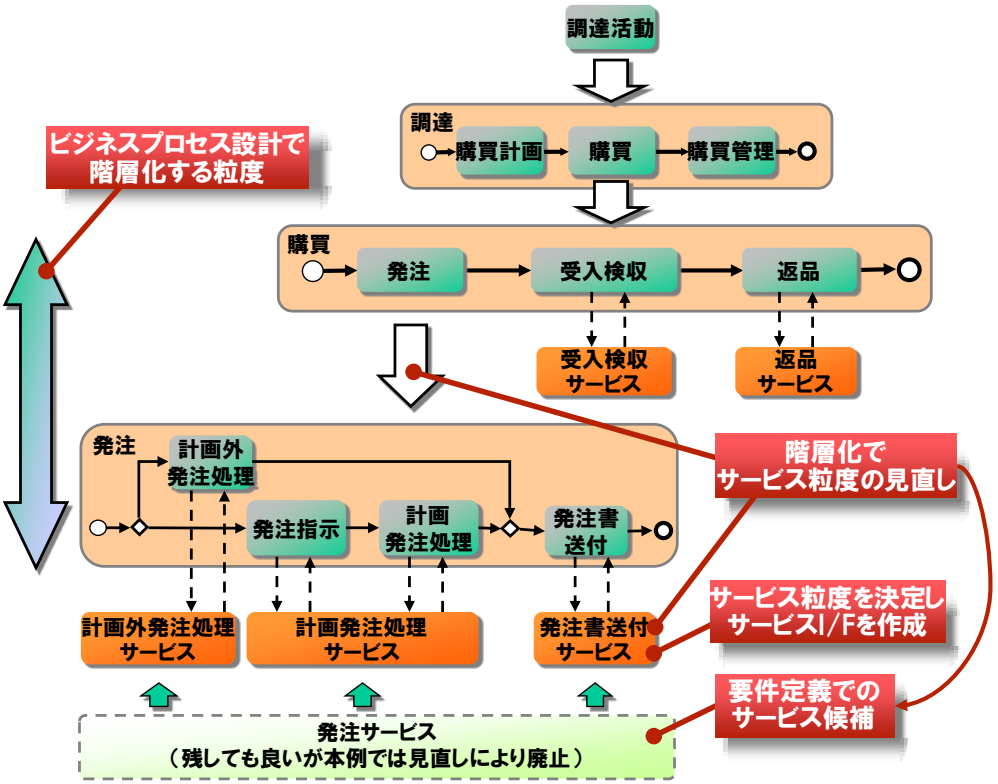


図 2－5　階層化とサービスの決定

要件定義工程の段階で抽出したサービス候補をそのままサービスとして使用する場合もあります。要件定義工程で作成する業務フローでの業務は、多くが「サブシステム」に対応するものであり、この単位で変化を局所化することが適当な情報システムもあるからです。サブシステムとは、データベースを共有するアプリケーションの集合で構成されるシステムのことを指します。

サービスの粒度の選定に関する考え方を図 2－6 に示します。業務機能の階層構造では、上位の階層は変化が少なく、ほかから独立している基本的な業務を示し、下位の階層になるほど変化が多く、ほかとの依存関係が複雑になる傾向を示します。

サービスの適切な粒度は、図 2－6 では中位階層の業務に対応したものです。これは、業務の変化が発生する単位に近く、サービスの独立性が高い粒度です。業務の変化に対するプログラムの変更をサービスという実装単位に局所化できます。これによって、システム変更を当該業務と対応したサービスを組み替えることができます。サービスの組み替えとは、サービスを新たな実装のサービスと入れ替えることです。なお、この中位階層のサービス粒度として、図 2－6 に示したように、前述した業務のバリエーションに対応したものが考えられます。

これに対して、最上位の階層で示すサービス粒度では、業務の変化に対して実装の単位が大きすぎる場合があります。プログラムの変更はサービス内で、クローズします。しかし、本来は入れ替えなくてよい部分まで入れ替えなければならないため、開発や運用のコストがより大きくなってしまいます。ただし、業務によっては、変

化を局所化する実装単位として、この最上位の粒度が適切な場合もあります。このような場合は、前述したようにビジネスプロセス階層化の中での設計が必要です。また、既存システムのオープン化やパッケージ製品の導入といったシステム計画レベルの変化に対しては、この最上位の粒度のサービスを組み替えの単位とするのが適切と考えられます。

最下位の階層のサービス粒度では、業務の変化に対して、実装の単位が小さすぎます。サービス間の依存関係が強く、プログラムの変更が複数のサービスにわたってしまい、システム変更が一つのサービスに局所化されないことになります。

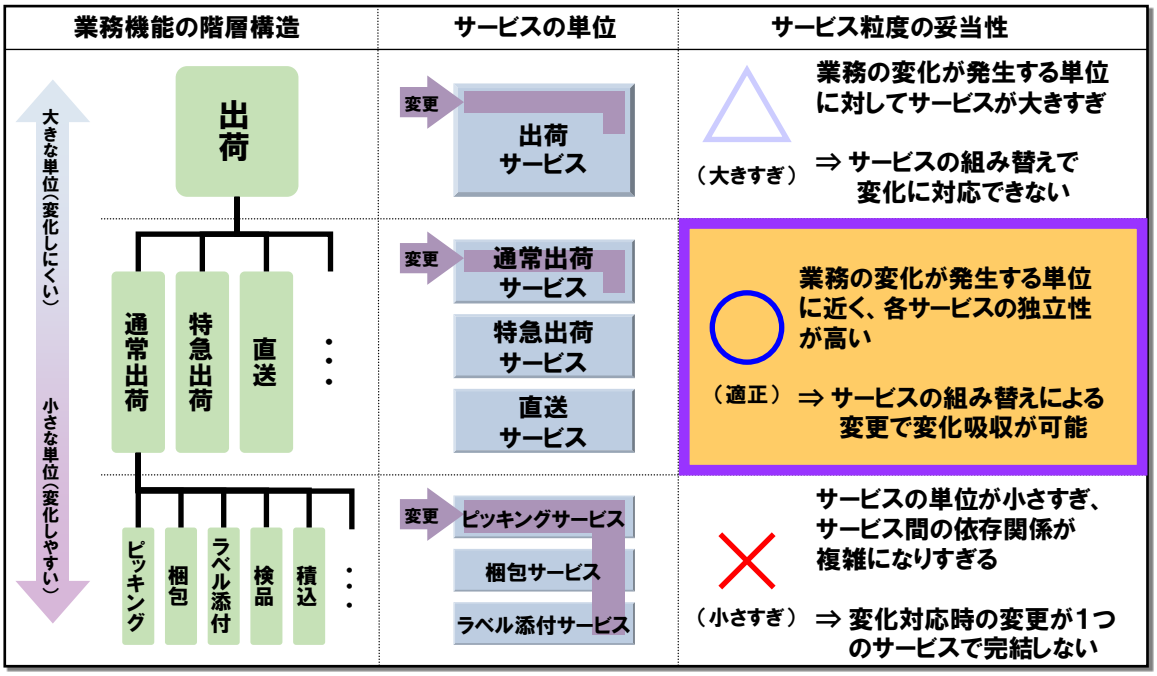


図 2－6　サービスの粒度

2.4. ビジネスプロセスの階層化とサービスI/Fの設計

ここでは、2.3 節で述べたサービス抽出の考え方を踏まえ、要件定義工程からビジネスプロセス設計、サービス I/F 設計に至る流れを説明します。

要件定義工程では、システム化計画に基づき、機能要件の洗い出しと新業務フローを設計します。業務フローは、図 2－7 に示すように、業務に関わる組織・担当者ごとに、業務の流れと授受される情報を定義していきます。業務フローの記法としては、一般に経済産業省がEA(エンタープライズアーキテクチャ)策定向けに提案しているＷＦＡ(Work-Flow Architecture)があります。

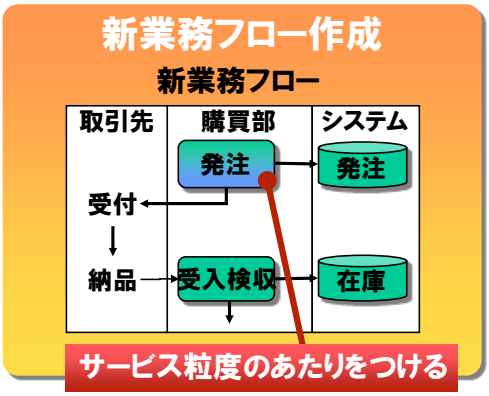


図 2－7　新業務フロー作成の成果物例

基本設計工程におけるビジネスプロセス設計では、新業務フローを BPMN 記法に変換後、BPMN 図による階層化を行います。新業務フローと同等の BPMN 図を基本フローとし、さらにアクティビティごとにその業務の流れを BPMN 図で記載します。階層化については、業務の実施方法のバリエーションを考慮して、下位のビジネスプロセスにおけるアクティビティを設計します。

例えば、図 2－8 で示すビジネスプロセス図では、発注アクティビティを一段階階層化する様子を示しています。発注業務に内在するバリエーションである計画発注処理と、計画外の発注処理についてのフローを設計し、ビジネスプロセスを階層化します。

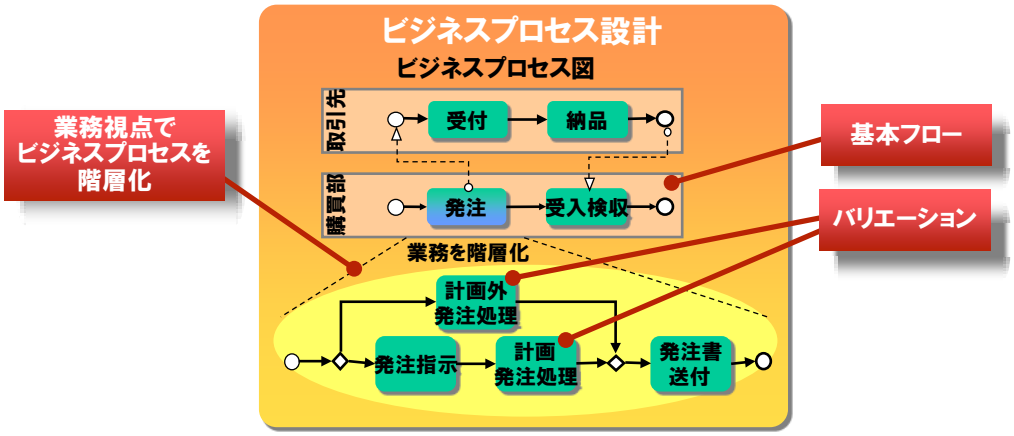


図 2－8　ビジネスプロセス設計の成果物例

このようなビジネスプロセスの階層化で、2.3 節で述べたようなサービス抽出を実施することによって、適切な粒度のサービスを決定します。図 2－9 は、図 2－8 で階層化したビジネスプロセスに対応したサービスの例を示しています。要件定義工程では、「発注」という業務全体をサービス候補としていましたが、この段階でサービスの粒度を見直し、階層化された下位のビジネスプロセスを構成する業務の「計画外発注処理」、「計画発注処理」などをサービス候補として定めています。

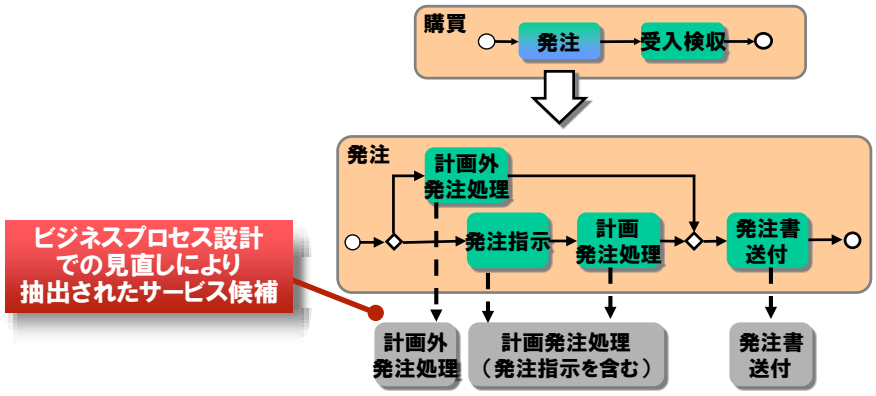


図 2－9　サービス粒度の見直し結果の例

さらに、ビジネスプロセス設計では、最下層のビジネスプロセスに記載しているアクティビティについてユースケースシナリオを作成します。ユースケースシナリオは、アクティビティごとの業務処理詳細を処理の順序に沿って文章で記載した成果物です。ユースケースシナリオでは、人がする作業と、それに関連したシステム側の機能を明確にします。このユースケースシナリオは、アクティビティを実現するサービスの実現形態の決定やサービスを実現するアプリケーションの設計をするときに活用します。

基本設計工程のサービス I/F 設計では、階層化されたビジネスプロセスを元に、メッセージの構造や例外などを明確化し、サービス I/F の仕様を設計します。なお、分岐条件の詳細仕様などの BPMN 図で表現できない仕様については、BPMN 図とは別に作成します。またビジネスプロセス設計で仮決定していたサービスを決定し、サービス I/F が所属するサービスを確定します。

サービス I/F には、ビジネスプロセスが呼び出すサービスのサービス I/F と、ビジネスプロセス自身のサービス I/F があります。図 2－10 は、図 2－9 で設計したビジネスプロセス・サービスに対して設計したサービス I/F の例を示しています。

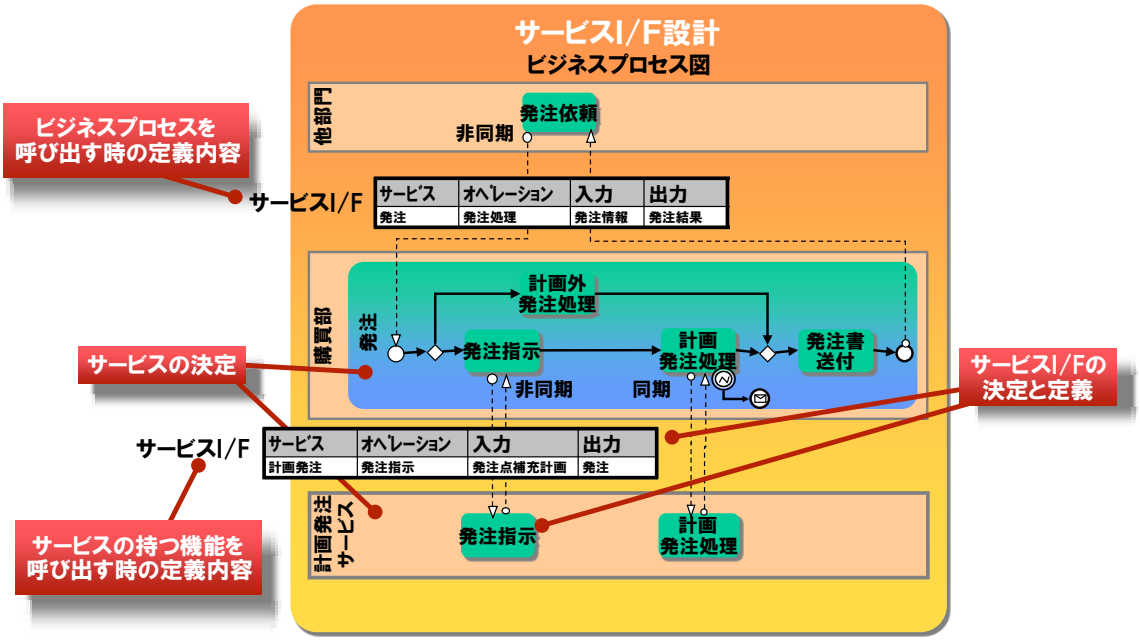


図 2－10　サービス I/F 設計の成果物例

2.5.　フロントシステム・サービスの設計・開発

2.4 節に示したように、抽出したサービスに対して、図 1－4 の SOA システムパターンに応じたサービスの実現形態を決定します。また、ビジネスプロセスを呼び出すフロントシステムについても、実現形態を決定します。フロントシステム・サービスの設計・開発では、選択された SOA システムパターンに応じて詳細設計・実装をします。

図 2－11 は、サービスに対して SOA システムパターンを選択する様子を示しています。単一の作業者が業

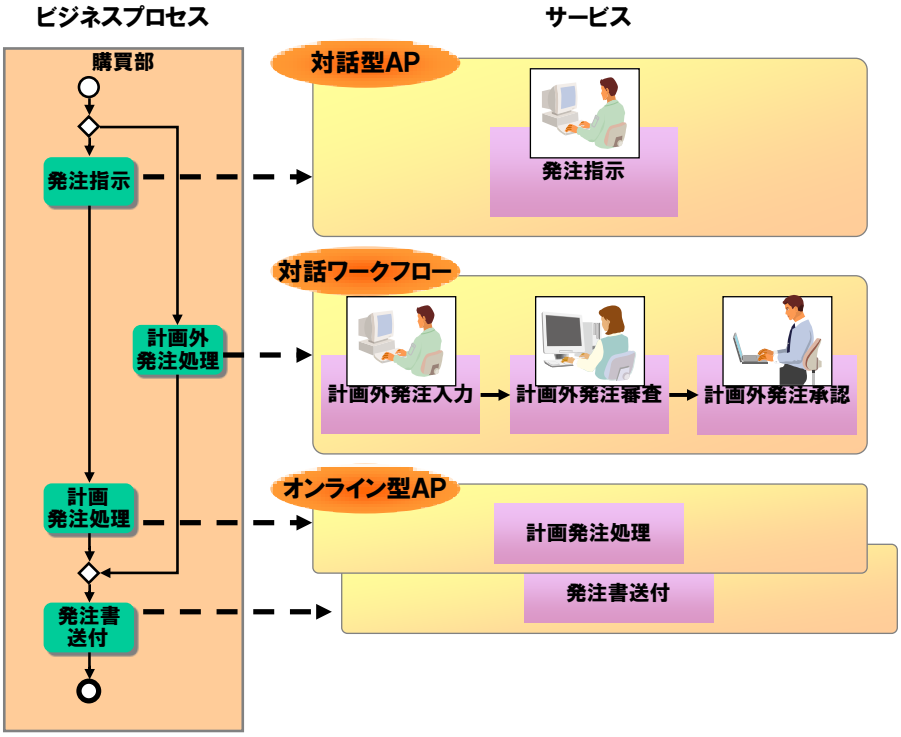


図 2－11　SOA システムパターンによるサービス実現形態

務を遂行する形態のサービスには、対話型アプリケーションパターンを選択します。複数の作業者が業務を遂行する形態のサービスには、対話ワークフローパターンを選択します。さらに、作業者が介在しなくプログラムで処理を実施する形態のサービスには、オンライン型アプリケーションパターンを選択します。

このように選択した SOA システムパターンに応じて、フロントシステムとサービスを各種形態のアプリケーションで実現するための設計・開発をします。ここでは、従来からアプリケーション開発方法として実績のあるコンポーネント開発手法を推奨しています。コンポーネント開発手法では、アプリケーションをプレゼンテーション層（画面・画面遷移）、ファンクション層（業務機能）、およびデータ層の三層で構成し、それぞれの特長に基づき設計をします。

サービスの実現形態としては、前述したように、対話型アプリケーションパターン、オンライン型アプリケーションパターン、または対話ワークフローパターンを選択できますが、これらのアプリケーションでは、ビジネスプロセスと連携するために、サービス I/F をファンクション層に設けます。

対話型アプリケーションパターンでは、図 2－12 に示す Web アプリケーションのように、プレゼンテーション層として画面 I/F を持つ一方で、ファンクション層でビジネスプロセスとの連携のためのサービス I/F を設けます。また、問合せ応答処理を実装するオンライン型アプリケーションでは、図 2－12 で示すプレゼンテーション層を持たない構成となります。対話ワークフローパターンでは、図 2－12 のサービスコンポーネントを複数個、組み合わせる構成となります。

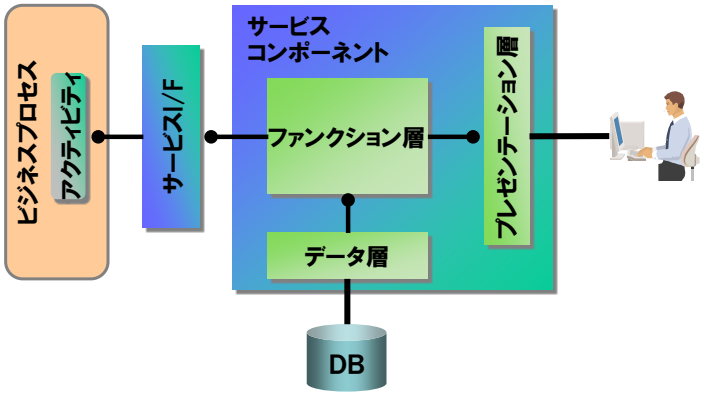


図 2－12　サービスコンポーネントの内部構成

一方、フロントシステムでは、図 1－4 で示した対話型アプリケーションパターン、対話ワークフローパターン、またはイベント駆動パターンを選択できます。図 2－13 に示すように、フロントシステムの対話型アプリケーションパターンでは、ファンクション層からビジネスプロセスを呼び出す処理を設定できます。これは、ビジネスプロセスもサービスコンポーネントと同様にサービス I/F を持つことができるため、このサービス I/F を介したビジネスプロセス呼び出し処理が可能となります。フロントシステムの対話ワークフローパターンでは、図 2－13 のサービスコンポーネントを複数個、組み合わせる構成となります。なお、イベント駆動パターンの構成については、4 章を参照してください。

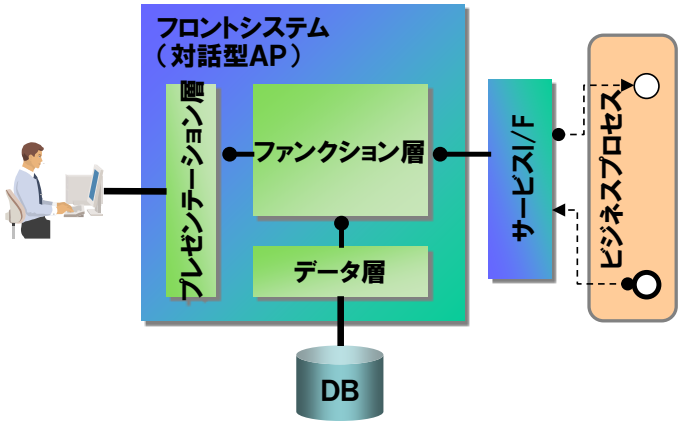


図 2－13　フロントシステムの内部構成

3. ビジネスプロセスの開発

本章では、ビジネスプロセスの開発について説明します。ビジネスプロセスは、複数のサービスを組み合わせた処理の流れと、サービス I/F を介したサービスとのメッセージ連携処理を定めるものです。Cosminexus のサービスプラットフォームを用いることで、ビジネスプロセスの開発、実行、および運用を実現できます。

（1）基本的な考え方

ビジネスプロセスは、複数のサービスを組み合わせて、サービス間の処理フローを制御する役割を持ちます。サービスの呼び出し方法は、要求となるメッセージを送信し、サービスを実行後、応答となるメッセージを受信する形となります。また、サービス間の処理フローとして、メッセージの変換やメッセージの内容による分岐処理などを行います。

（2）基本的な開発の進め方

uCosminexus Service Architect で、ビジネスプロセスを設計・開発します。ビジネスプロセスの開発モデルを図 3－1 に示します。

1. 多様な形態で実現されたサービスへの呼び出しを制御するためのアダプタ定義。サービス I/F の情報から要求・応答になるメッセージを作成します。
2. 複数サービス間のビジネスプロセスを制御するためのビジネスプロセス定義。サービス間の処理フローを作成します。
3. メッセージ間の変換をするためのメッセージ変換定義。処理フローの 1 ステップとして、次のサービスを呼び出すために必要なメッセージの作成などをします。

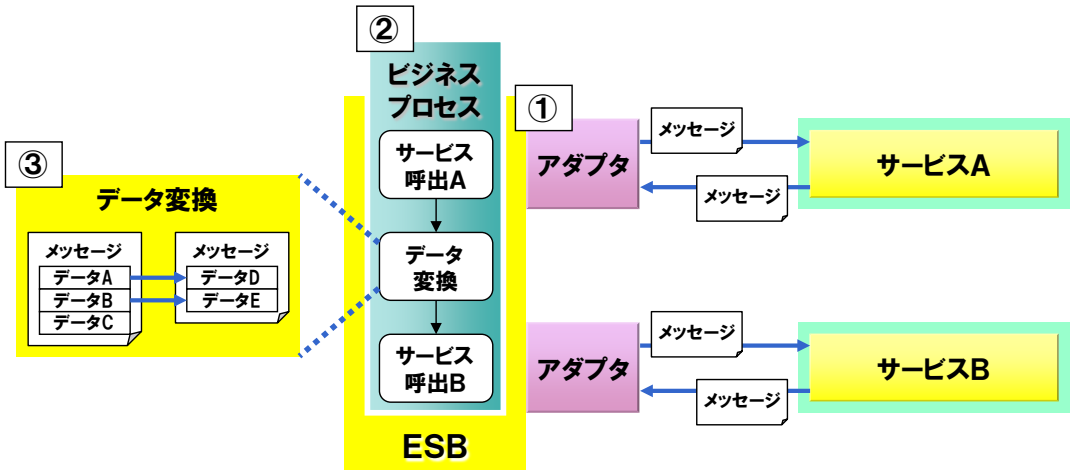


図 3－1　ビジネスプロセスの開発モデル

画面イメージを図 3－2 に示します。業界標準の BPMN の処理モデルに従って、サービス間の処理フローをビジュアルに定義できます。

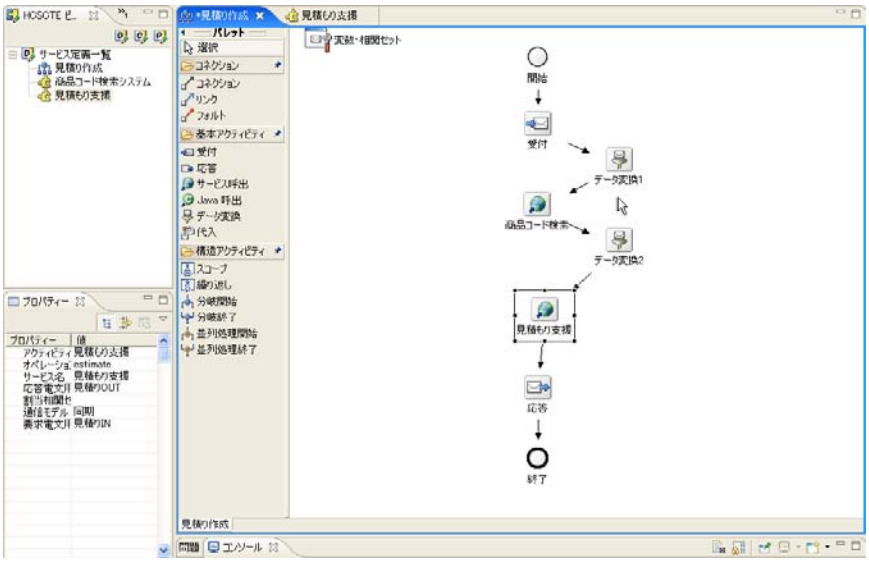


図 3－2 ビジネスプロセス定義画面イメージ

サービスを Java や COBOL で開発する場合、uCosminexus Service Platform に含まれるアダプタで接続できます。さらに、メインフレームや TP モニタなどの上で動く既存システムをサービスとして利用する場合、次の二つの方法があります。

- 既存システムにラッピング技術を適用し、Web サービス化して ESB に接続する。
- アダプタで既存システムを ESB に接続する。

サービスの開発を 4 章、既存システムの活用を 5 章で説明します。

また、ビジネスプロセスのデバッグには、ビジネスプロセスデバッガを使用します。ビジネスプロセスのデバッガ画面イメージを図 3－3 に示します。

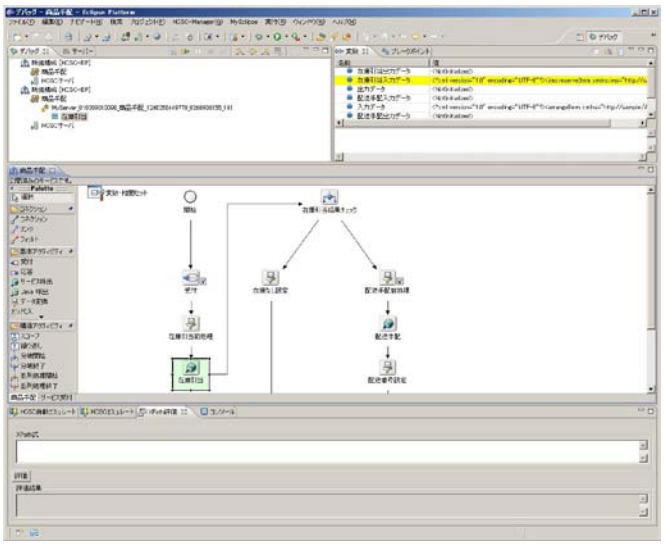


図 3－3 ビジネスプロセスデバッガ画面イメージ

テスト環境上で、実際のサービスを呼び出さないでテストを可能にするためにサービスエミュレータを使用します。サービスエミュレータでできることは、次の二つです。

- デバッグ中にサービス呼出があった場合、サービスをエミュレーションできる。

ビジネスプロセスデバッガ上で、応答内容を記述した XML ファイルをサービスの応答として定義することで、あたかもサービスを呼び出したかのように処理を進めます。(図 3－4)

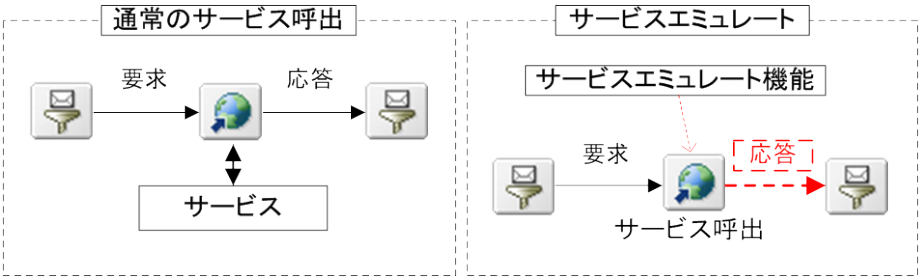


図 3－4 サービスエミュレータイメージ

- デバッグ中にサービス呼出があった場合、入力内容によって、応答メッセージを変えられる。

ビジネスプロセスデバッガ上で、サービスの出力条件をあらかじめ定義した場合、サービスのエミュレーションがされた時点で入力内容を評価し、条件に合致する応答メッセージを返します。例えば、図 3－5 のようになります。

①サービスエミュレータ上で、出力条件を定義します。
(発注者がAの場合はAに対する発注結果を、発注者がBの場合はBに対する発注結果を出力するよう定義する。)

②入力の発注情報中の発注者が A である場合、サービスエミュレータは、出力として発注者 A に対する発注結果を返します。

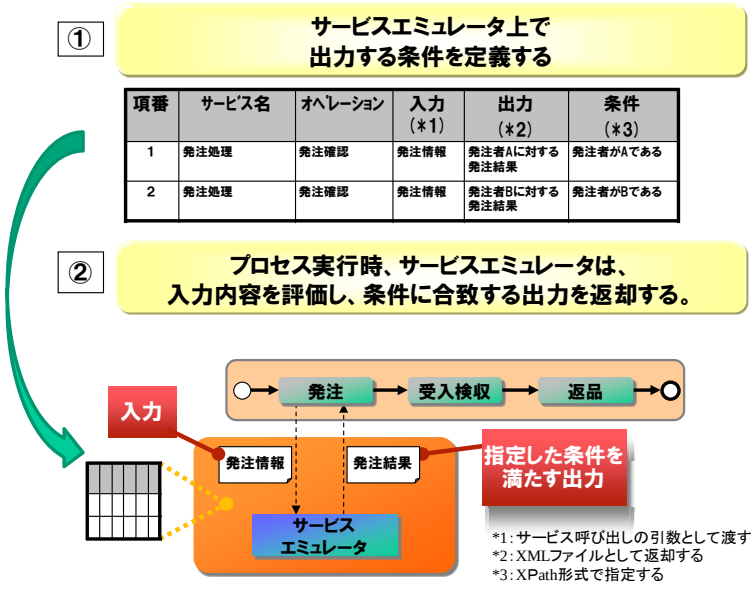
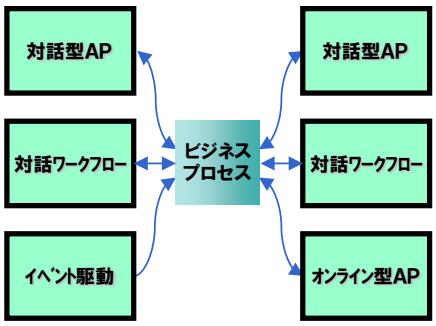


図 3－5 サービスエミュレータ出力条件定義イメージ

（3）サービスの拡張・改修の進め方

アダプタ定義やビジネスプロセス定義などのサービス群は、uCosminexus Service Architect 内で管理されます。サービス I/F やサービスの各種定義情報が管理されるため、開発者は登録されたサービスの一覧をブラウズすることで、サービスの拡張や改修を効率よくできます。

4. フロントシステム・サービスの開発



本章では、フロントシステムおよびサービスの開発について説明します。フロントシステム・サービスは、1章で述べた SOA システムパターンに従ったアプリケーションとして開発します。ここでは、パターンごとに開発方法を説明していきます。

4.1. オンライン型アプリケーションパターンの開発

プログラムによって処理を実行するオンライン型アプリケーションの開発について説明します。オンライン型アプリケーションの開発方法として大きく二つの方法があります。

1. Java による開発
2. COBOL による開発

4.1.1 Javaによるオンライン型アプリケーション

uCosminexus Developer で、uCosminexus Application Server 上で動作するオンライン型アプリケーションを Java 言語で開発できます。サービスの形態は、同期処理と非同期処理の二種類に分類できます。同期処理は Web サービスとして実現し、非同期処理は MDB(Message Driven Bean)として実現します。実行イメージを図 4－1 に示します。

Java 言語での開発では Eclipse 上で WSDL の作成、オンライン型アプリケーションを uCosminexus Application Server に配備する配備定義ファイル(Deployment Descriptor)や EAR(Enterprise ARchive)ファイルの作成などができます。

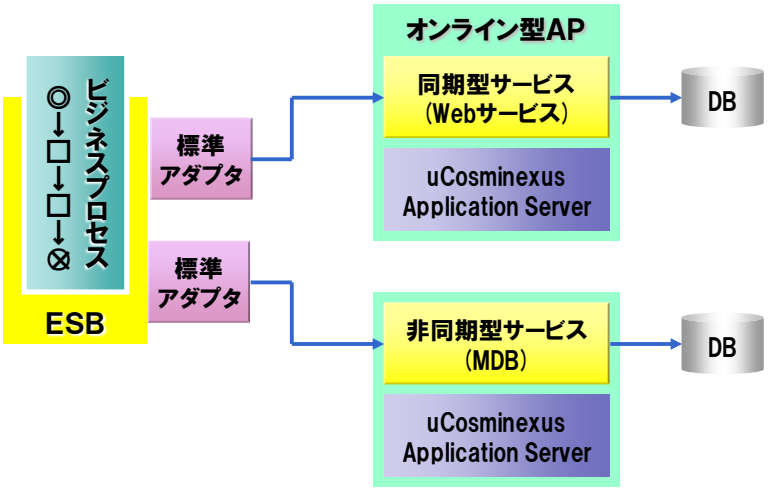


図 4－1 Java によるオンライン型アプリケーション

Web サービスの開発方法として、基本設計によるサービス I/F 設計を基に WSDL ファイルを作成するトップダウン開発と、Java のインタフェースクラスを作成して WSDL ファイルを自動生成するボトムアップ開発の二種類の開発方法があります。開発方法の内容について、表 4－1 にまとめました。

また、既存の Java アプリケーションがある場合、Web サービスの形態になっていればそのまま活用できます。その他の場合、上記の開発ツールを活用して Web サービスにラッピングすることで活用できます。

表 4－1　Java による Web サービスの開発方法

#	開発方法	ユースケース	概要
1	トップダウン開発 (WSDL→Java)	ビジネスプロセス詳細設計・実装を先に進めて、WSDL の形式で Web サービスの I/F を決める場合	基本設計によるサービス I/F 設計をもとに WSDL ファイルを作成します。次に WSDL2Java コマンドで Java ソースコードを自動生成し、Java プログラムを実装します。
2	ボトムアップ開発 (Java→WSDL)	コンポーネント設計・実装を先に進めて、Java インタフェースクラスによって Web サービスの I/F を決める場合	基本設計によるサービス I/F 設計を基に Java のインタフェースクラスを作成します。次に Java2WSDL コマンドで WSDL ファイルを自動生成します。

4.1.2　COBOLによるオンライン型アプリケーション

COBOL によるオンライン型アプリケーションの開発方法として、次の二つの方法があります。

（1）uCosminexus OpenTP1(SPP)によるアプリケーション開発

SPP(Service Providing Program)形態として COBOL 言語で開発した uCosminexus OpenTP1 上のアプリケーションを uCosminexus Service Platform から呼び出せます。この場合、TP1 アダプタから RPC(Remote Procedure Call)通信によって接続します。uCosminexus OpenTP1 アプリケーションの実行イメージを図 4－2 に示します。

また、既存の uCosminexus OpenTP1(SPP)上のアプリケーションがある場合、TP1 アダプタを使用してそのまま接続できます。

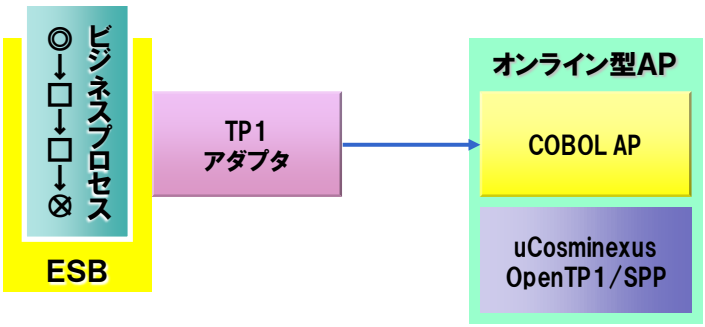


図 4－2　uCosminexus OpenTP1 によるオンライン型アプリケーション

（2）COBOL2002 Cosminexus 連携機能によるアプリケーション開発

COBOL2002 の Cosminexus 連携機能や TP1/COBOL adapter for Cosminexus Version 2 を使用して COBOL アクセス Bean を開発することによって、COBOL アプリケーションを uCosminexus Application Server 上の Web サービスとして実行できます。

COBOL アプリケーションの実行イメージを図 4－3 に示します。Web サービスラッパー、および COBOL アクセス Bean を介する形で COBOL アプリケーションの呼び出しができます。

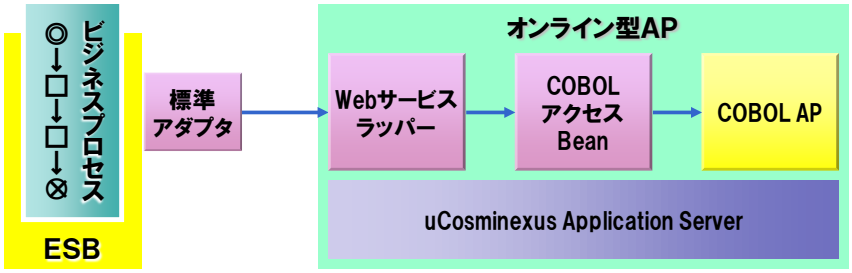
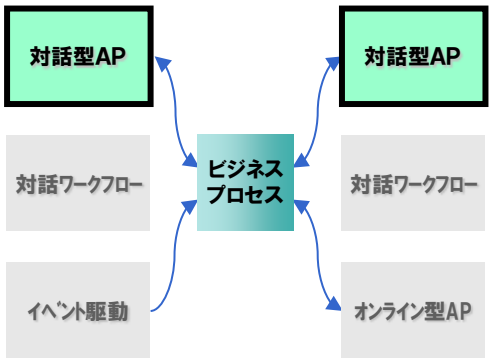


図 4－3　COBOL 2002 Cosminexus 連携機能による
オンライン型アプリケーション

4.2. 対話型アプリケーションパターンの開発



人が介在し、作業者の画面操作によってイベントが発生する対話型アプリケーションの開発について説明します。人が介在する画面操作が主になるため、画面の表示形式が重要になります。

対話型アプリケーションはフロントシステム側、サービス側の双方にありますが、図 4－4 に示すように呼び出しモデルが異なります。

- フロントシステムの場合：対話型アプリケーションからビジネスプロセスを呼び出します。
- サービスの場合：ビジネスプロセスからリクエストを受け付けるオンライン型アプリケーションの形式になります。ビジネスプロセスからのリクエスト受付後、リクエストを DB に永続化します。次に、対話型アプリケーションとして人が作業をして、作業終了後、ビジネスプロセスを呼び出す形で応答を返します。

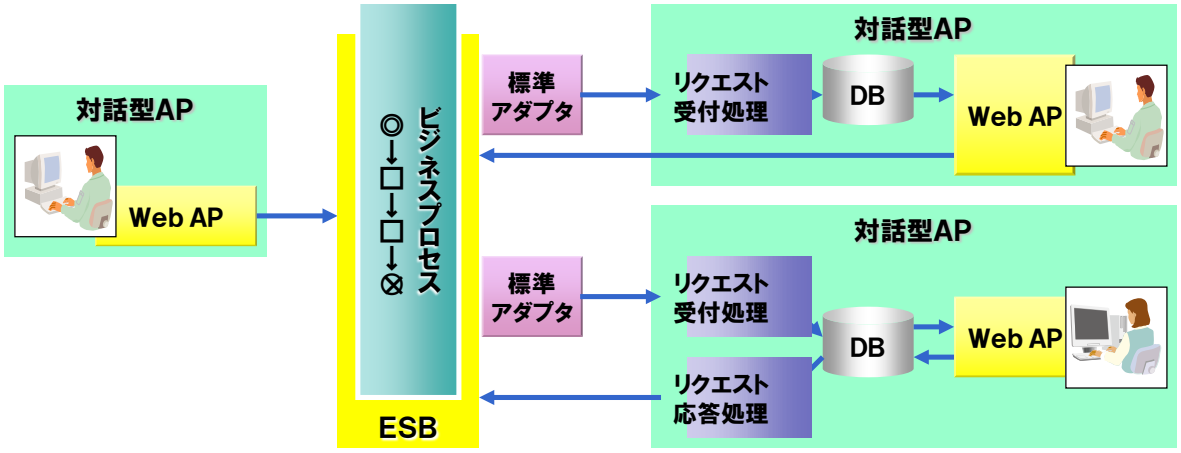


図 4－4　対話型アプリケーションの呼び出しモデル

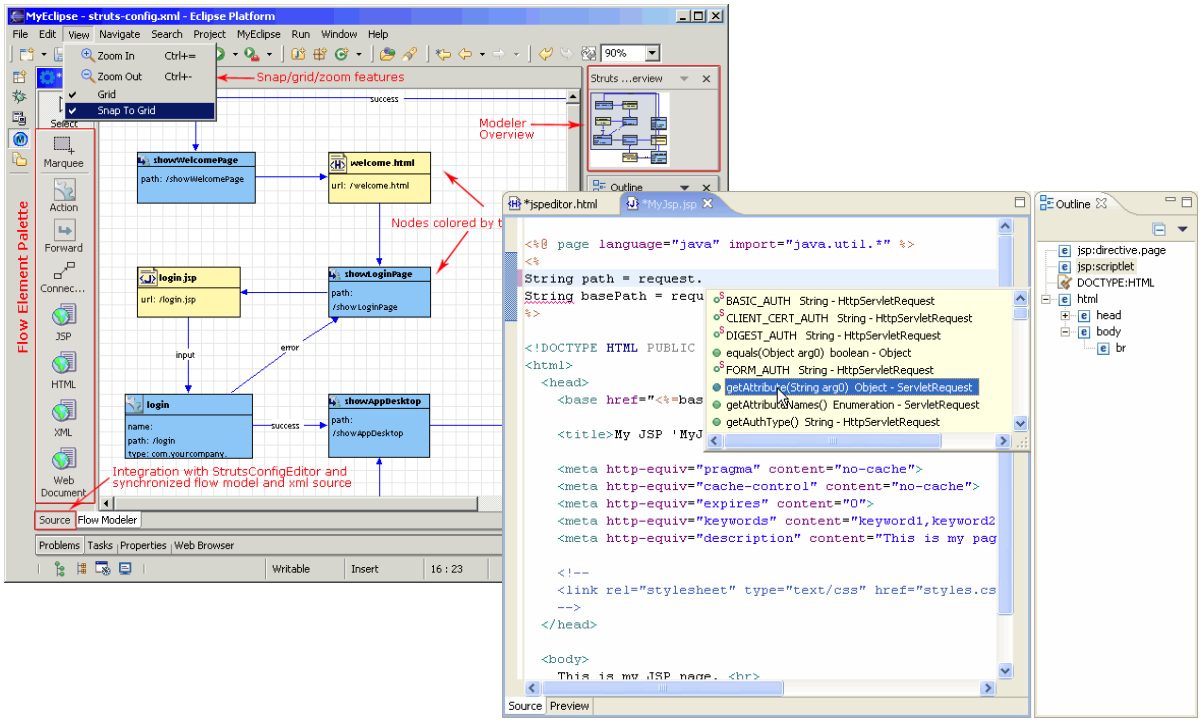
また、既存の対話型アプリケーションがある場合、ビジネスプロセスの呼出部分を改修したり、リクエスト受付処理を追加したりすることで活用できます。ビジネスプロセスの呼出部分は、DB を参照してビジネスプロセスを呼び出すリクエスト応答処理を外付けで追加することで改修部分を少なくすることもできます。

対話型アプリケーションの開発方法には、次の三つの方法があります。

1. Java による開発
2. .NET による開発
3. COBOL による開発

4.2.1 Javaによる対話型アプリケーション

uCosminexus Developer および Eclipse で、MVC(Model-View-Controller)モデルに基づいた Web アプリケーションの画面や画面遷移を開発できます。



4.2.2 .NETによる対話型アプリケーション

Microsoft 社の Visual Basic .NET や C#で、.NET Framework 上の対話型アプリケーションを開発できます。

4.2.3 COBOLによる対話型アプリケーション

COBOL2002 で、対話型アプリケーションを開発できます。COBOL アプリケーションの実行イメージを図 4－5 に示します。

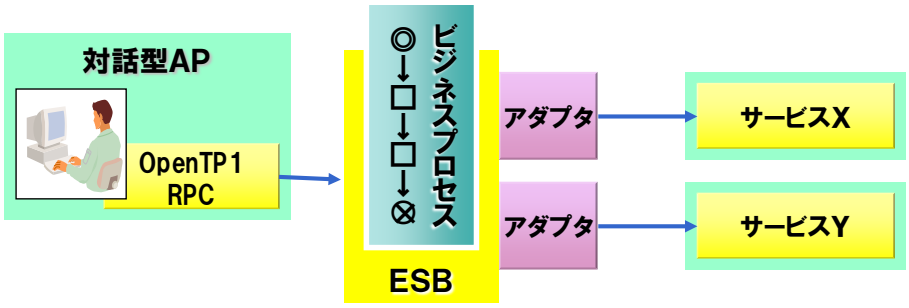
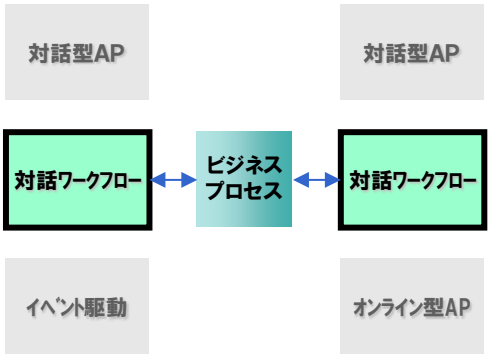


図 4－5　COBOL による対話型アプリケーション

4.3. 対話ワークフローパターンの開発



複数の作業者間の連携によって一連の業務を遂行するようなフロントシステム・サービスについては、対話ワークフローパターンとして実現します。対話ワークフローパターンの実現には、作業者間の作業の流れを管理し、各作業者が用いる対話型アプリケーションの統合ができる uCosminexus Service Platform – WorkCoordinator(以下、uCSP-WCO と略します)の使用がが有効です。

（1）基本的な考え方

3 章でビジネスプロセス開発の基盤として Cosminexus のサービスプラットフォームについて説明しましたが、uCSP-WCO もビジネスプロセス管理のためのシステム構築基盤の一種です。Cosminexus のサービスプラットフォームは、複数のサービス間のメッセージ連携を実現するビジネスプロセス(メッセージフロー)を管理するのに対し、uCSP-WCO は複数の作業者間を連携させるためのビジネスプロセス(対話ワークフロー)を管理します。

（2）基本的な開発の進め方

対話ワークフローの開発モデルを図 4－6 に示します。

対話ワークフローは、対話ワークフロー基盤と対話ワークフローアプリケーション(INBOX アプリケーション、業務アプリケーション)から構成されます。uCSP-WCO は、対話ワークフローを実行するための対話ワークフロー基盤、対話ワークフローアプリケーションを開発するためのワークフローAPI の双方を提供しています。

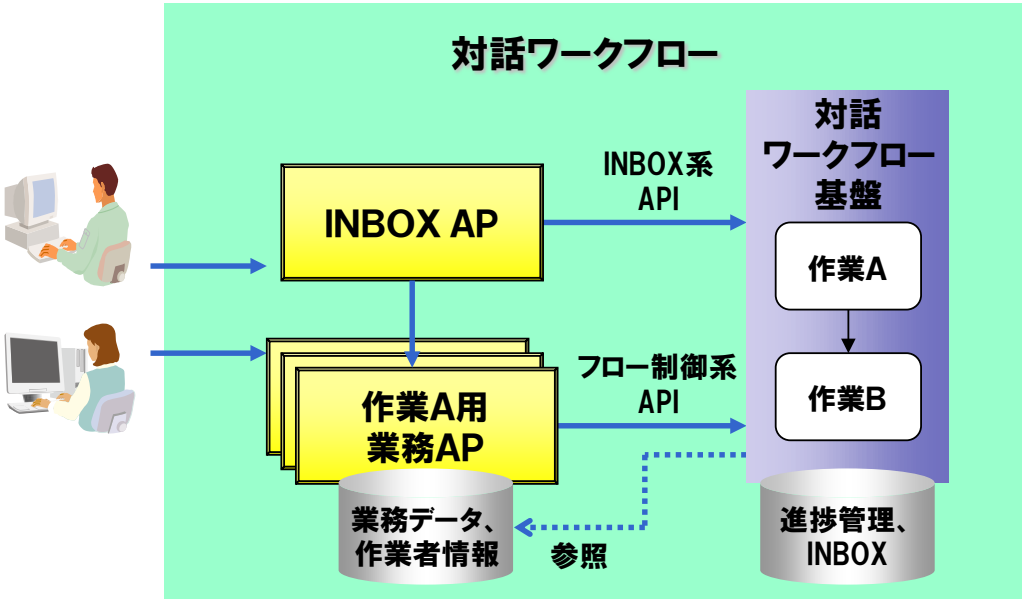


図 4－6 対話ワークフローの開発モデル

- 対話ワークフロー基盤:対話ワークフロー定義実行機能、INBOX 系 API、フロー制御系 API、ワークフローの監視機能があります。作業者がする業務の流れを定義したものを対話ワークフロー定義と呼びます。ワークフローの状態を管理し、対話ワークフローアプリケーションから呼び出されることによって、対話ワークフロー定義に従い、作業者の割当およびフローの進捗を制御します。
- 対話ワークフローアプリケーション:大きく分けて INBOX アプリケーション、業務アプリケーションの二つがあります。
-

1. INBOX アプリケーション

作業者ごとに受信トレー画面や送信ログ画面などを表示するためのアプリケーションであり、作業者共通に実装されるのが一般的です。uCSP-WCO は、作業の一覧情報を取得する INBOX 系 API を提供しています。

2. 業務アプリケーション

受信トレー画面に表示される特定の業務を処理するためのアプリケーションです。業務処理の実行後、uCSP-WCO が提供するフロー制御系 API を用いてフローの進捗を指示します。

対話ワークフローの定義には、uCosminexus Service Architect - WorkCoordinator を使用します。画面イメージを図 4－7 に示します。作業者間の処理フローをビジュアルに定義できます。作業者の割当ルール（振り分けルール）や分岐条件には、作業者情報や業務データを検索するための SQL を指定します。

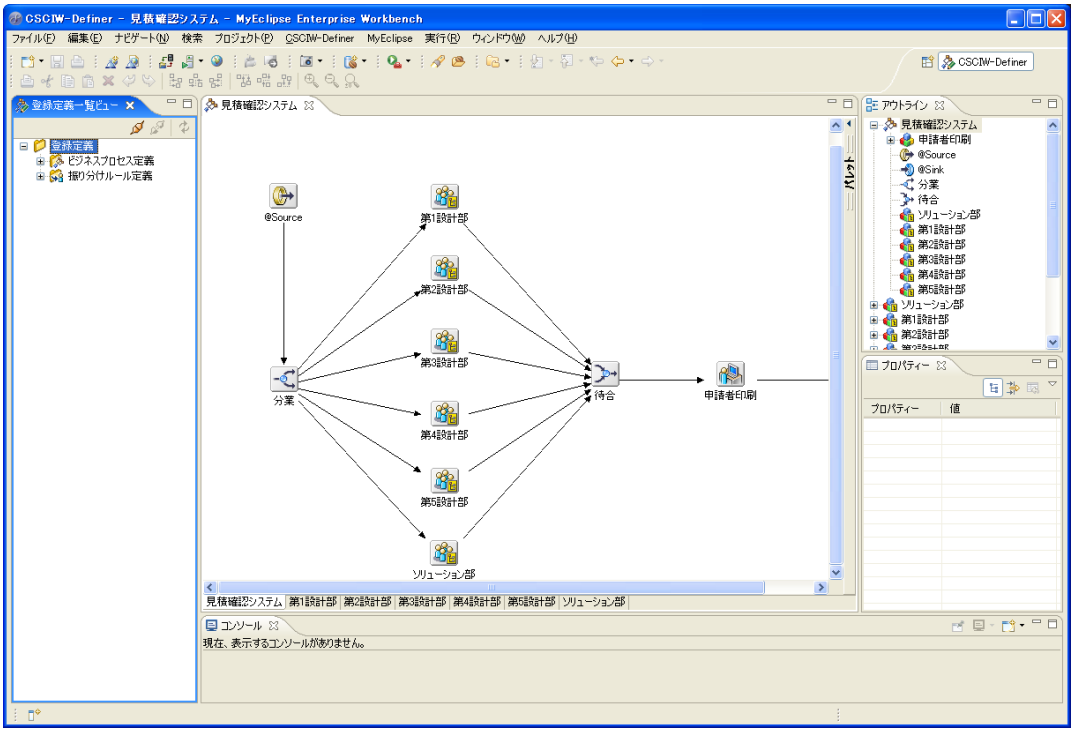


図 4－7　対話ワークフロー定義画面イメージ

（3） ビジネスプロセスと対話ワークフローの連携

ビジネスプロセスと対話ワークフローを連携させることによって、より複雑な業務に対応できます。連携方法として、大きく二つの方法があります。

（a） 対話ワークフローからビジネスプロセスの呼び出し

実行イメージを図 4－8 に示します。フロントシステムに複数人による対話業務があり、特定の業務でバックエンドシステムと連携しなければならない場合が該当します。対話ワークフロー内の Web アプリケーションからビジネスプロセスを Web サービスとして呼び出すことで連携できます。

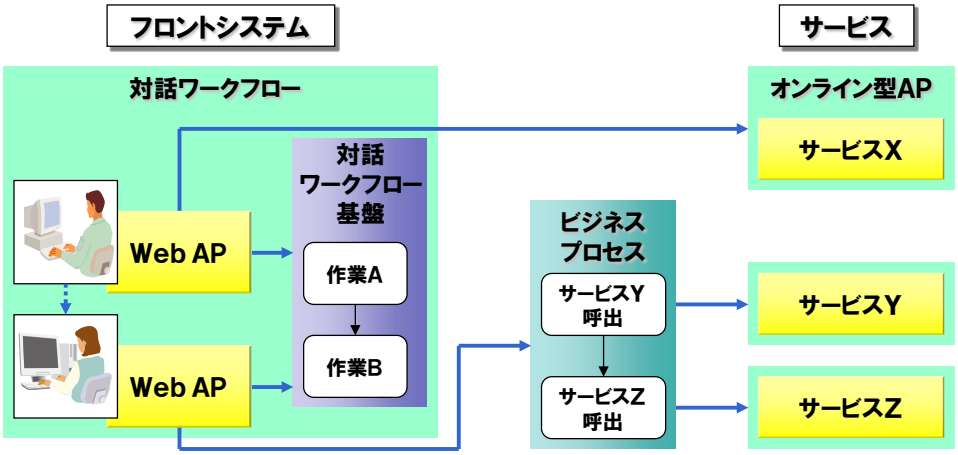


図 4－8　対話ワークフローからビジネスプロセスの呼び出し

（b） ビジネスプロセスから対話ワークフローの呼び出し

実行イメージを図 4－9 に示します。ビジネスプロセスでワンストップサービスを提供し、特定のサービス内で複数人による対話業務がある場合に該当します。ビジネスプロセスから対話ワークフローを Web サービスとして呼び出すことで連携できます。

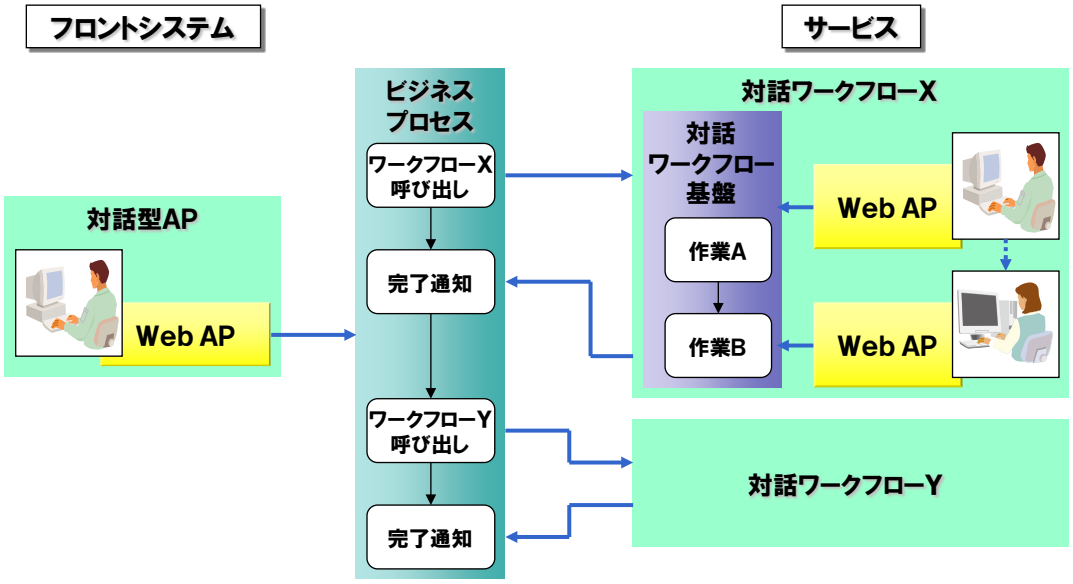
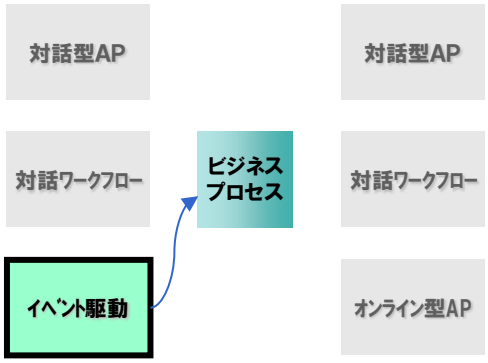


図 4－9　ビジネスプロセスから対話ワークフローの呼び出し

4.4. イベント駆動パターンの開発



ファイル到着といったイベントや日時や時刻といったスケジュールに応じてビジネスプロセスを駆動する処理の開発について説明します。イベント駆動には、ジョブスケジューラ(JP1/AJS3)の利用が有効です。

JP1/AJS3 によるイベント駆動の基本的な開発モデルを図 4－10 に示します。

1. ジョブスケジューラを起動させるイベントまたはスケジュールを定義。ファイル到着や日付、時刻など
2. ビジネスプロセスの呼び出し方法を定義

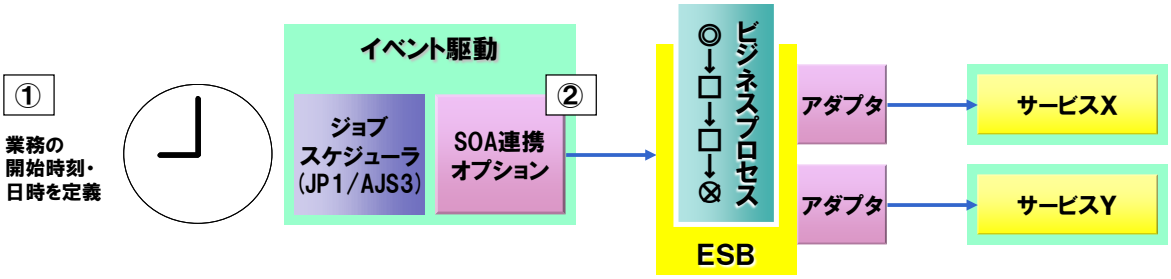
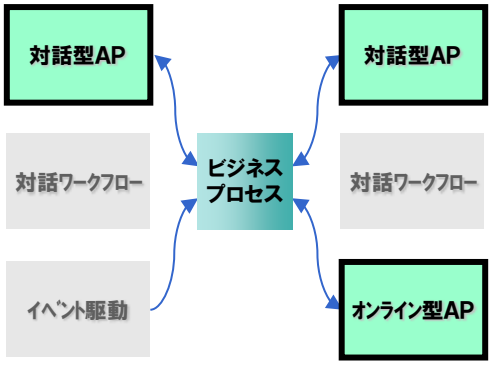


図 4－10　JP1/AJS3 によるイベント駆動

ビジネスプロセスを呼び出す場合、SOA 連携オプション(JP1/AJS3 － SOA Option)を利用します。

また JP1/AJS3 は複数ジョブの呼び出しを制御できます。標準ジョブとしてバッチファイルなどの非常駐プログラムを呼び出せるため、それらのジョブとビジネスプロセスを組み合わせることができます。

5. 既存システムの活用パターン



既存システムをオンライン型アプリケーション、または対話型アプリケーションの実現手段として活用できます。既存システムを活用する利点は、次のとおりです。

- 新規開発を最小限に抑えられます。
- 既存システムに含まれたお客様独自のノウハウや強みを継承できます。
- ESB を使用することで、既存システムと新規開発のシステムを使い分け、段階的にシステムを刷新できます。

5.1. オンライン型アプリケーションの活用

既存のオンライン型アプリケーションとの連携方法には、いくつかのバリエーションがあり、図 5－1 にパターンをまとめました。

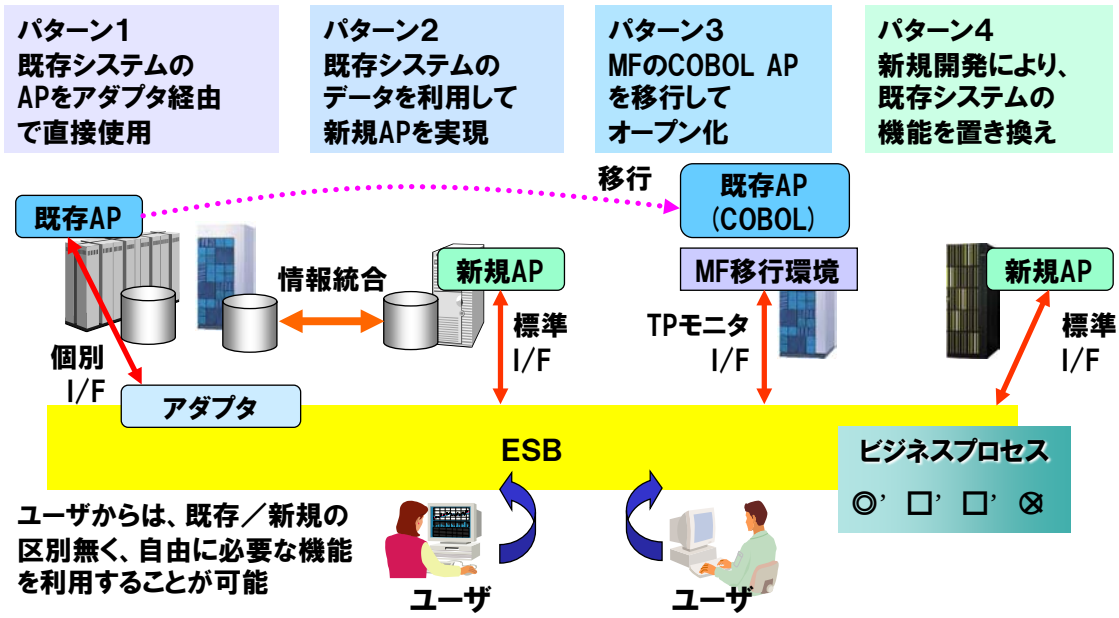


図 5－1　既存オンライン型アプリケーション連携パターン

- **パターン1：** 既存システムをESB に接続したい場合に適用します。既存システムとして、メインフレーム、TP モニタといった各種基盤で構築されたものなどがあります。
- **パターン2：** 既存システムのデータを利用して新規にオンライン型アプリケーションを実現する場合に適用します。既存システムとのデータ連携は情報統合技術を適用して実現を図ります。
- **パターン 3：** メインフレームの COBOL アプリケーションに特定されますが、メインフレーム上のアプリケーションをオープン環境にそのまま移行して、新システムを構築する場合(ストレートコンバージョン)に適用します。新システムの構築基盤として TP モニタなどがあります。
- **パターン4：** 新規開発で、再構築する場合に適用します。

既存システムとして、TP1 アダプタを使用して、TP1/RPC プログラム、または TP1/RPC によるゲートウェイプログラムを介して日立ホスト／他社ホスト、対外ネットワーク、全銀手順、CAFIS(Credit And Finance Information System)、MPN(Multi Payment Network)などに接続できます。また、データの形式を既存システムに合わせなけ

ればならない場合、uCosminexus Service Platform の持つデータ変換機能を利用できます。

ここで、既存システムの境界をどのように業務の境界と合わせ、サービス化するかが課題となります。既存システムのサービス化の接続バリエーションを図 5－2 に示します。

(a) 既存システムそのもの

既存システムがすでに業務の境界と一致している場合に適用します。

(b) ビジネスプロセスによるサービス化

既存システムが業務の境界と一致してなく、認証処理、前処理、後処理など、複数の処理を呼び出して一つの業務が完結する場合。ビジネスプロセスをサービス化するための制御フローとして使用することで、業務の境界とサービスの境界を一致させます。単純に処理を組み合わせれば良いケースに適用します。

(c) Web サービスラッパーによるサービス化

既存システムが業務の境界と一致していない場合、Web サービスでラッピングすることで、業務とサービスの境界を一致させます。(b)に比べて複雑なロジックが必要なケースに適用します。

(b)や(c)の場合、独立性を高めるため、既存システムの機能を段階的に切り出していくことで、ビジネス変化へ迅速に対応できる部分を増やしていくことができます。

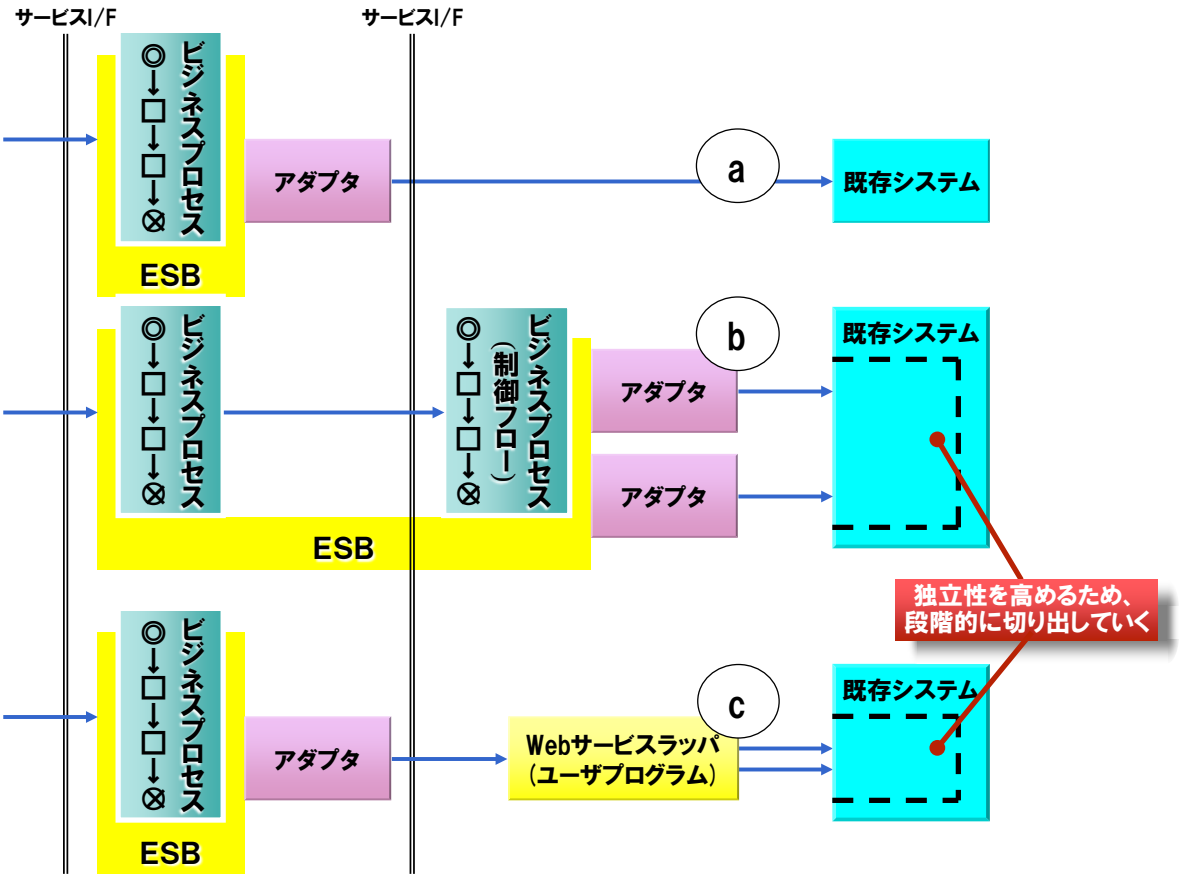


図 5－2　既存システムのサービス化の接続バリエーション

5.2. 対話型アプリケーションの活用

人が介在する対話型アプリケーションとして、既存の TP1 クライアントや専用端末などが残っている場合、残った対話アプリケーションをビジネスプロセスで連携できます。必要に応じて、段階的に Web ブラウザによる対話型アプリケーションへの移行ができます。

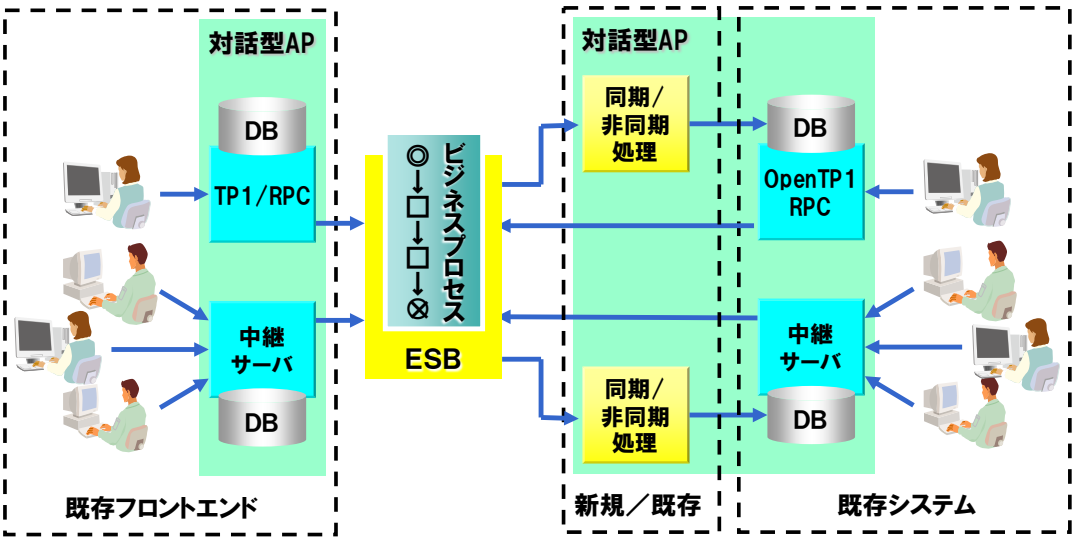


図 5－3　既存対話型アプリケーションの活用

Cosminexus

2012年4月第5版発行
お問い合わせ先：株式会社 日立製作所　情報・通信システム社
ITプラットフォーム事業本部　システム基盤ソリューション部
cosminexus-s@itg.hitachi.co.jp

インターネットで製品情報をご覧いただけます。
http://www.hitachi.co.jp/cosminexus/
http://www.cosminexus.com/