ソフトウェア工学 122－2
（1999．3．18）

分散処理システムのアーキテクチャ・スタイル

森澤　好臣*,**、鳥居　宏次**
*：　日本ユニシス株式会社
〒135-8560 江東区豊洲 1-1-1
E-mail: Yoshitomi.Morisawa@unisys.co.jp
**：　奈良先端科学技術大学院大学情報科学研究科
〒630-0101 奈良県生駒市高山町 8916-5
E-mail: {yosito-m, torii}@is.aist-nara.ac.jp

あらまし　分散コンピューティング環境下でのアプリケーション・システム開発は、次のようなアーキテクチャに関する幾つかの質問に配慮しなければならない。どのようにコンピュータ資源を分散するのか。そしてどのようにコンピュータ資源間のコミュニケーションを実現するのか。これらの選択プロセスを支援するために、分散処理システムのアーキテクチャ・スタイルの体系を提案する。分散処理システムを、データの所在場所と、クライアントとサーバ間およびサーバとサーバ間の処理形態をベースにして９つのスタイルに分類する。そして、どのようにシステムに適用できるかをも示す。

キーワード　分散処理システム、クライアント／サーバ･システム、アーキテクチャ・スタイル、参照モデル

Architectural Styles for Distributed Processing Systems

Yoshitomi Morisawa *,** and Koji Torii **
*：Nihon Unisys, Ltd.
1-1-1 Toyosu, Koto-ku, Tokyo, 13-8560 Japan
E-mail: Yoshitomi.Morisawa@unisys.co.jp
**：Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma-shi, Nara 630-0101, Japan
E-mail: {yosito-m, torii}@is.aist-nara.ac.jp

Abstract　The development of an application system in a distributed computing environment must consider several architectural questions such as how and where computing resources are distributed and how computing resources should communicate with each other. To support this decision making process, we propose a framework for architectural styles of distributed processing systems. Distributed processing systems are classified into nine styles based on where the data storage is located and how client-server and server-server communication are done. We also show how this can be applied to several systems.

Key words　Distributed Processing System, Client/Server System, Architectural Style, Reference Model

1．はじめに

ビジネス活動を支援するために、コンピュータが使用されるようになって数十年経過した。コンピュータによる情報処理システムは、ビジネス活動を遂行するために不可欠のものになっている。そして、ビジネス活動を支援する情報処理システム（ビジネス・アプリケーション）の処理形態も時代とともに進歩している。

1990 年代以前は、銀行預金の出し入れや座席予約システム等に代表されるオンライン・リアルタイム・システムや、企業に於ける会計処理、給料処理、請求書発行等の事務処理を一括処理する、メインフレーム中心の世界であった。これらのために開発されたビジネス・アプリケーションは、ほとんどが手作りであった。

1990 年代に入って、Unix や PC の技術進歩により、クライアント／サーバ(C/S)システムが出現した。ユーザ・インタフェースとアプリケーション・ロジックがクライアント上で稼動し、サーバ上にあるデータベースを更新する 2 層モデルのビジネス・アプリケーションである。Unix と PC がその仕様を公開したオープンなプロダクトであったため、C/S システム構築用の多くの COTS (Commercial Off The Shelf)プロダクトが市場に登場している。そして、クライアント上で稼動するアプリケーションの肥大化を反省して、処理構造をプレゼンテーションとアプリケーション・ロジックとデータを論理的にも物理的にも分離する 3 層モデルの概念が打ち出された。

1990 年代も後半になると、情報技術の進歩により電子メール、グループウェア等のオフィス事務処理作業を支援するビジネス・アプリケーションが登場した。さらに、インターネットに代表される Web 技術がユーザ・インタフェースに追加され、コンポーネントウェア(ソフトウェアの部品化)が出現し始めた。データは、データベース・サーバ化の方向を目指している。ビジネス・アプリケーションの構成方法もプレゼンテーション、アプリケーション・ロジック、データの 3 層モデルが一般化し、更に Web サーバ等が付け加わり、多層化の傾向を示している。ビジネス・アプリケーションの適用範囲も拡大し、ビジネス・アプリケーションを構築する時に、プレゼンテーション、アプリケーション・ロジック、データ管理、セキュリティ、機能分散、コンポーネント化、等々と考慮すべきことが増大し、アプリケーションの処理構造が複雑になってきている。また、ビジネス・アプリケーションをどの様なツールを使用してで開発するかに眼を向けても、市場にはアプリケーション開発用の各種の COTS プロダクトが氾濫している。またアプリケーション部品も市場に流通し始め、ビジネス・アプリケーションの構築方法が COTS プロダクトをベースにした開発から組み立てへと移行している。オープンなプロダクトを使用してシステムを開発する時のガイドラインも報告されている。[1]

ユーザがビジネス・アプリケーションを構築するとき、どのような処理形態で、どこに資源を配置するか。どのような COTS プロダクトの組み合わせを使用して開発するか。これらのことは、常にシステム設計時に問題になり解決する必要がある課題である。ビジネス・アプリケーションの基本的なアーキテクチャ・スタイルを明確にし、ビジネス・アプリケーションをこれらのスタイルやスタイルの組み合わせで構築する。そのことによって、ビジネス・アプリケーションのシステム構造を単純化することが出来れば、アプリケーションの見通しが良くなり、構築ツールや部品の選択や組み合わせ等が支援され、トータルなアプリケーション開発の生産性向上や品質向上が期待できる。[2, 3]

本論文は、データの所在場所（集中保管と分散保管）とクライアントとサーバ間およびサーバ同士間の処理形態（同期と非同期）に焦点を当てて、分散処理形態のビジネス・アプリケーションを 9 つのカテゴリに分類した。アプリケーション・ロジックをどのように分割し分散配置するかの視点も重要であるが、今後の研究課題とした。

2．クライアント／サーバ・モデルの拡張

本論文で提案するアーキテクチャ・スタイルの説明に入る前に、スタイルのベースになった考え方を説明する。

本論文で対象とする情報処理システムは、主として企業におけるビジネス処理用のアプリケーションである。ビジネス処理を情報処理システムとして実現するために、ビジネス世界の商行為や商慣習をシステム化する。従って、「注文を出す」、「商品の在庫を調べる」、「商品を発送する」、「請求書を発行する」、「入金を確認する」等々のビジネスの手順やプロセスを抽象化することになる。

ビジネス行為をモデル化するために、まず、作業を依頼する「もの（物、者）」と作業を遂行する「もの（物、者）」、及びそれらの間の関連を抽象化する。[4] 作業を依頼し、依頼に依って遂行し、結果を知らせるという一連の行為は、コンピュータ処理のモデルとしては、古典的な C/S モデルがよく適合している。しかし、現実のビジネスの進み方は、依頼しそれに同期して作業を遂行する作業形態だけでなく、依頼と依頼された作業の遂行が非同期に為されることが多々ある。従って、複数の作業が協調しあって作業を遂行する分散協調処理をも抽象化のモデルとして含む必要がある。

図 1 に依頼する「もの」と作業する「もの」の関連を示す。作業する「もの」は作業を遂行するために他に依頼することもある。

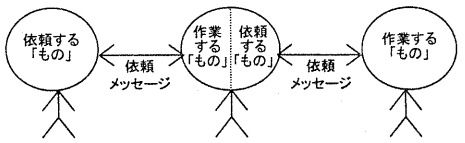

図1　依頼する「もの」と作業する「もの」

依頼する「もの」には、依頼者（クライアント、アクターやユーザ）、作業するもの（サーバやシステム）そして事象（イベント）がある。作業する「もの」には、作業するもの（サーバやシステム）がある。

一般に、作業の依頼メッセージには、結果を知る必要がある「作業依頼」や「問い合わせ」がある。これらは、依頼に対して同期的に作業を遂行し結果を報告することが要請されている。もう一つの依頼メッセージには、依頼した作業結果を知る必要がない、作業の「通達」がある。これらは、依頼に対して非同期的な作業実施が要請されている。従って、作業の依頼メッセージは、最小単位として作業依頼し結果を返すトランザクション型メッセージ、問い合わせに対して調査し応答する依頼応答型メッセージと依頼するだけの通達型メッセージに大別することが出来る。前者二つがクライアントとサーバが同期して作業する形態であり、後者が非同期で作業する形態である。

3．アーキテクチャ・スタイル

3.1 スタイルの分類

前章で記述した C/S モデルの拡張の考え方を適用して、分散処理システムを次の視点で分類する。

1) クライアントとサーバ間の処理形態が、クライアントの依頼に対してサーバが同期して処理する同期処理か、またはサーバが依頼とは非同期に処理する非同期処理か。
2) サーバ上にデータが、集中保管されているか、または分散保管されているか。
3) 複数のサーバにより処理される時、サーバ間の処理形態が同期処理であるか、または非同期処理であるか。
4) クライアントとサーバ間の処理形態が同期処理の時、依頼のメッセージ・タイプがトランザクション型であるか、または依頼応答型であるか。

一覧表にすると、分散処理システムのアーキテクチャ・スタイルは、表1に示す9つの「基本スタイル」になる。現実のシステムは、基本スタイルの1つ以上の組み合わせで構成されている。

この分類は、通常の C/S モデルに対してデータの分散形態と処理に時間の概念を導入したことになる。原則として、「プレゼンテーション（P）」は、クライアント側にあり、「データ（D）」は、サーバ側にあることを仮定している。

表1　アーキテクチャ・スタイル

データ形態 / C/S 間の処理形態（注-1 / 注-2）		集中	分散	
			同期処理	非同期処理
同期処理	トランザクション型	集中トランザクション スタイル	分散トランザクション スタイル	非同期トランザクション スタイル
	依頼応答型	集中依頼応答 スタイル	分散依頼応答 スタイル	非同期依頼応答 スタイル
非同期処理	通達型	集中通達 スタイル	分散通達 スタイル	非同期通達 スタイル

（注-1）：サーバ間の処理形態　（注-2）：メッセージ・タイプ

表1で使用する用語の意味は以下の通りである。

- 「**データ形態**」は、使用するデータの保管場所を示している。但し、個人用ファイル及び個人用データベースは対象外とする。
- 「**集中**」とは、データが一カ所に集中保管されている形態である。
- 「**分散**」とは、データが複数箇所に分散保管されている形態である。
- 「**処理形態**」は、クライアント／サーバ間およびサーバ間の処理の形態を示している。同期処理と非同期処理がある。
- 「**トランザクション型**」とは、ACID 特性を持ったトランザクション処理の形態である。
  原始性(Atomicity)、一貫性(Consistency)、
  独立性(Isolation)、耐久性(Durability)
- 「**依頼応答型**」とは、クライアント側の依頼に対しサーバ側で同期的に処理がなされ、応答が返される形態である。
- 「**通達型**」とは、クライアント側からの通達に対しサーバ側間で非同期的に処理がなされる形態である。

本章のモデル図で使用する記号を説明する。

- P　：プレゼンテーション
- AP　：アプリケーション・ロジック
- DM　：データ管理
- D　：データ
- ←→　：トランザクション型メッセージ
- ⟷　：依頼応答型メッセージ
- ⟋⟍　：通達型メッセージ

3.2 集中トランザクション・スタイル

単一のサーバ上に単一のデータベースがあり、クライアントとサーバ間の処理形態がトランザクション型の同期処理である。

典型的な集中トランザクション・スタイルを図２に示す。

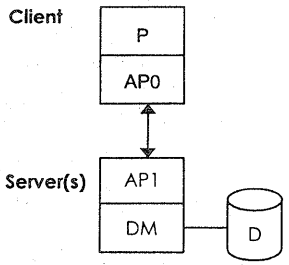

図２ 集中トランザクション・スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 よりトランザクション型メッセージが AP1 に送られる。トランザクションの処理結果で D の中のデータが DM により検索・更新・追加・削除される。そして処理結果がクライアントに返される。AP と DM が別のサーバ上に置かれることもある。

このスタイルは、トランザクション管理プログラムを利用する単一データベースによる追加更新が主体のトランザクション処理に適する。トランザクション負荷が大きい場合は、アプリケーション・サーバとデータベース・サーバに分割されることがある。アプリケーションの例としては、受発注管理、商品在庫管理、株式注文処理、生産管理、小売 POS システム等がある。

3.3 分散トランザクション・スタイル

複数のサーバ上に複数のデータベースがあり、クライアントとサーバ間の処理形態がトランザクション型の同期処理であり、サーバ間の処理形態が同期処理である。

典型的な分散トランザクション・スタイルを図３に示す。

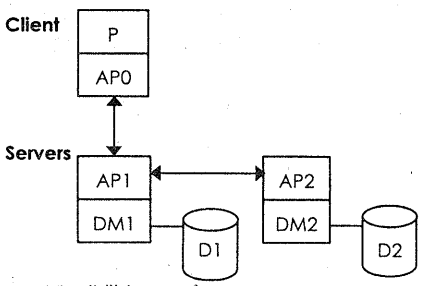

図３分散トランザクション・スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 より、トランザクション型メッセージが AP1 に送られる。そのトランザクションの一部の処理結果で D1 の中のデータが DM1 により検索・更新・追加・削除される。そして処理されなかった又は処理された結果のトランザクション型メッセージが AP2 に送られ、処理結果で D2 の中のデータが DM2 により検索・更新・追加・削除される。そして処理結果がクライアントに返される。必要に応じて、D1 と D2 の整合性を保つためのメカニズムが使用される。APx と DMx が別のサーバである場合がある。

このスタイルは、複数のデータベースへの更新や問い合わせが複雑にからむ基幹系トランザクション処理に適する。トランザクション負荷が大きい場合はアプリケーション・サーバとデータベース・サーバに分割されることがある。トランザクション管理プログラムのメッセージ・キューイングなどを利用する非同期処理を伴う場合は、非同期処理を含んだモデルに分類される。アプリケーションの例としては、銀行の勘定系システム、座席予約システム、工場生産管理システム等がある。

3.4 非同期トランザクション・スタイル

複数のサーバ上に複数のデータベースがあり、クライアントとサーバ間の処理がトランザクション型の同期処理であり、サーバ間の処理が非同期処理である。

典型的な非同期トランザクション・スタイルを図４に示す。

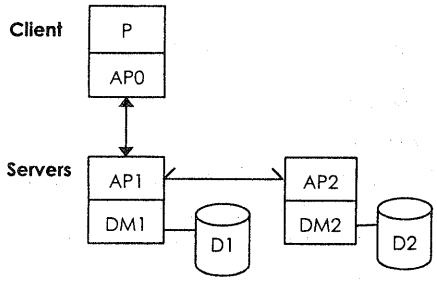

図４ 非同期トランザクション・スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 より、トランザクション型メッセージが AP1 に送られる。AP1 の処理結果により D1 の中のデータが DM1 により検索・更新・追加・削除される。そして AP1 の処理で作成された通達型メッセージが AP2 に送られる。そして処理結果が、AP0 のクライアントに返される。AP1 からのキック等、適当なタイミングで、別のサーバ上の AP2 より、既に送られている AP1 よりのメッセージに従って、D2 の中のデータが DM2 により検索・更新・追加・削除される。

典型的な利用方法としては、適当なタイミングで、上位のサーバ上の D2 の中のデータの一部（全部）を使用して、下位のサーバ上の D1 の中のデータを更新（ダウンロード）する。又は、その逆、D1 の中のデータを使用して D2 の中のデータを更新（アップロード）する、トランザクショ

ン処理型のアプリケーションである。

このスタイルは、データの非同期な共有による情報の管理と活用を促進する基幹系アプリケーションに適する。アプリケーションの例としては、支店や部門別に収集されたトランザクション・データの本社へのアップロード、部門サーバによって収集された受注データの本社サーバへのアップロードと本社サーバでの集中受注処理などである。

3.5 集中依頼応答スタイル

単一サーバ上に単一データベースがあり、クライアントとサーバ間の処理が依頼応答型の同期処理である。

典型的な集中依頼応答スタイルを図5に示す。

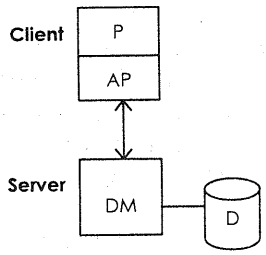

図5 集中依頼応答スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP より依頼応答型メッセージ（SQL 等）が DM に送られる。メッセージ形式で表された要求により D の中のデータが DM により検索・更新・追加・削除される。そして処理結果がクライアントに返される。AP 負荷が大きい場合の AP 機能や複数のクライアントに共通の AP 機能などはサーバに置かれる。

このスタイルは、意思決定支援システムに代表されるいわゆる EUC（エンドユーザ・コンピューティング）、および問い合わせ応答型の単純な即時処理に適する。EUC のアプリケーションの例には、予算計画、財務分析、市場調査・分析、売上分析、所要量計画、需要予測などの各種統計・分析・報告がある。問い合わせ応答処理のアプリケーションの例には、顧客サービス、営業支援、各種照会処理、情報提供サービスなどがある。

3.6 分散依頼応答スタイル

複数サーバ上に複数のデータベースがあり、クライアントとサーバ間の処理形態が依頼応答型の同期処理であり、サーバ間の処理が同期処理である。

典型的な分散依頼応答スタイルを図6に示す。

メッセージの流れと処理の流れを説明する。クライアント上の AP より、依頼応答型メッセージ（SQL 等）が DM1 に送られる。そのメッセージの（一部の）要求により D1 の

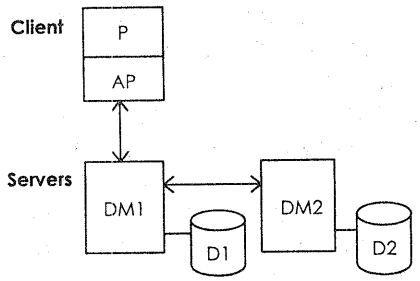

図6 分散依頼応答スタイル

中のデータが DM1 により検索・更新・追加・削除される。そして処理されなかったメッセージが、または処理された結果のメッセージが DM2 に送られ、D2 の中のデータが DM2 により検索・更新・追加・削除される。そして処理結果がクライアントに返される。必要に応じて、D1 と D2 の一貫性を保つためのメカニズムが使用される。

このスタイルは、複数のデータベースやファイルを同時にアクセスする EUC および問い合わせ中心の即時処理に適する。すでに存在するデータベースの共有による情報の有効活用ができる。集中依頼応答スタイルのアプリケーションはほとんど含まれるが、他に、全社売上統計、全社生産性データ分析などがある。

3.7 非同期依頼応答スタイル

複数のサーバ上に複数のデータベースがあり、クライアントとサーバ間の処理が依頼応答型の同期処理であり、サーバ間の処理が非同期処理である。

非同期依頼応答スタイルを図7に示す。

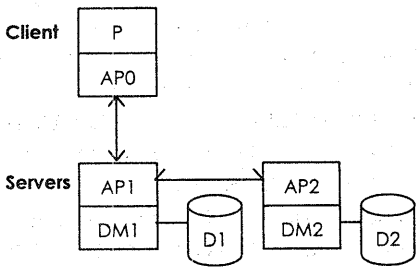

図7 非同期依頼応答スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 より、依頼応答型メッセージ（SQL 等）が AP1 に送られる。そのメッセージの要求により D1 の中のデータが DM1 により検索・更新・追加・削除される。そして AP1 の処理で作成された新しい通達型メッセージが AP2 に送られる。そして処理結果が、AP0 のクライアントに返される。AP1 からのキック等、適当なタイミングで、別のサーバ上

の AP2 より、既に送られている AP1 よりのメッセージに従って、D2 の中のデータが DM2 により検索・更新・追加・削除される。

典型的な利用方法としては、適当なタイミングで、上位のサーバ上の D2 の中のデータの一部（全部）を使用して、下位のサーバ上の D1 の中のデータを更新（ダウンロード）する、又は、その逆、D1 の中のデータを使用して D2 の中のデータを更新（アップロード）する、依頼応答処理型のアプリケーションである。

このスタイルは、データの非同期な共有による情報の管理と活用を促進するアプリケーションに適する。アプリケーションの例としては、業務系データベースの一部のダウンロードによる情報活用、例えば、大福帳や多次元データベースなどによる DSS、EIS などである。

3.8 集中通達スタイル

単一サーバ上に単一データベースがあり、クライアントとサーバ間の処理が非同期処理である。

典型的な集中通達スタイルを図 8 に示す。

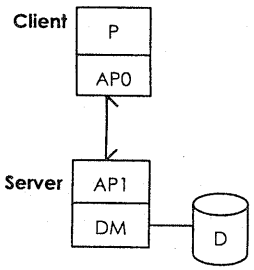

図 8 集中通達スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 より、通知型メッセージ（文章等）が AP2 に送られる。そのメッセージにより非同期に AP1 により D の中のデータが DM により検索・更新・追加・削除される。

このスタイルは、グループ内あるいは組織内での単純なワークフローの自動化に適する。アプリケーションの例としては、電子メールによる伝票・社内文書の配送と回付およびイベント通知、内部文書のファイリング、業務フローの実行・監視・報告などがある。

3.9 分散通達スタイル

複数のサーバ上に複数のデータベースがあり、クライアントとサーバ間の処理が非同期処理であり、サーバ間の処理が同期処理である。

典型的な分散通達スタイルを図 9 に示す。

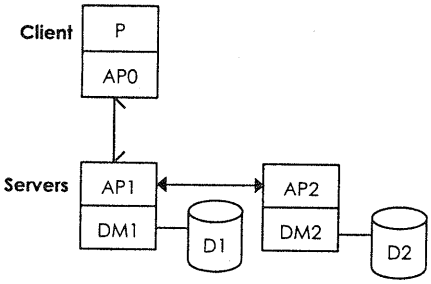

図 9 分散通達スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 より、通達型メッセージ（文書等）が AP1 に送られる。そのメッセージによりクライアントと非同期に AP1 により D1 の中のデータが DM1 により検索・更新・追加・削除される。そして AP1 の処理で作成された新しいトランザクション型（又は、依頼応答型）メッセージが AP2 に送られる。AP2 により D2 のデータが DM2 により検索・更新・追加・削除される。そして結果が AP1 に返される。

このスタイルは、モバイル・クライアントからサーバ上のエージェント・アプリケーションを用いた分散トランザクション処理や分散データ処理を行うのに適する。

3.10 非同期通達スタイル

複数のサーバ上に複数のデータベースがあり、クライアントとサーバ間の処理が非同期処理であり、サーバ間の処理も非同期処理である。

典型的な非同期通達スタイルを図 10 に示す。

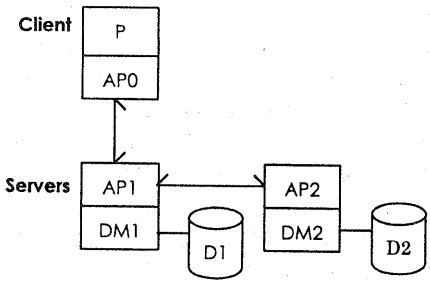

図 10 非同期通達スタイル

メッセージの流れと処理の流れを説明する。クライアント上の AP0 より、通達型メッセージ（文書等）が AP1 に送られる。そのメッセージにより非同期に AP1 により D1 の中のデータが DM1 により検索・更新・追加・削除される。そして AP1 の処理で作成された新しい通達型メッセージが AP2 に送られる。そのメッセージにより非同期に AP2 により D2 の中のデータが DM2 により検索・更新・追加・削

除される。

このスタイルは、独立した複数のアプリケーションやシステム同士の連携による疎な統合を行うのに適する。アプリケーションの例を次に示す。

1）グループ間組織間のワークフロー
- ・電子メールによる関連部門への伝票・社内文書の配布・回付およびイベント通知。
- ・社内稟議の回付と承認

2）企業内システム間のアプリケーション連携による統合
- ・メッセージ配送による、生産計画・在庫管理・工程管理などの統合
- ・受注・仕入・在庫・出荷・請求の各システムの統合。

3）EDI などによる企業間のシステム連携
- ・顧客システムと連携した受発注システム
- ・仕入れ先システムと連携した部品調達管理
- ・関連企業のシステムと連携したパック旅行予約

4．アーキテクチャ・スタイルの適用

前章で提案したアーキテクチャ・スタイルの適用性や記述性を検証するために、実稼働中のアプリケーション・システムのモデル化を試みた。ビジネス・アプリケーションのアーキテクチャ・スタイルを決定するために次の手順を採用している。[5]

1) データとデータを管理する組織を明確にする。
2) 必要なビジネス・プロセスを列挙する。
3) データを集中配置する場合とデータを分散配置する場合を記述する。これは、コンピュータ・システムの専門家でないユーザ部門よりの参加者に、提案するソリューションを理解してもらう必要があるからである。
   3-1) データを集中配置した時の各ビジネス・プロセスのアーキテクチャ・スタイルを明確にする。
   3-2) データを分散配置した時の各ビジネス・プロセスのアーキテクチャ・スタイルを明確にする。
4) 管理面、セキュリティ面そしてビジネス要件より集中配置か分散配置を決定する。
5) 実装可能性と安定性を考慮して単純なモデルを選択する。

論文サイズの制約より、一部試行の概要のみを記述する。

4.1 P 株式会社　チケット予約システム

全国 620 店舗（800 端末）から利用されるチケット予約システムである。大阪地域、北海道地域、名古屋地域、そして東京地域におかれた地域サーバと本社におかれた本社サーバによる分散トランザクション処理システムである。

図 11 で示すチケット予約システムでは、分散トランザクション・スタイルが適用可能である。

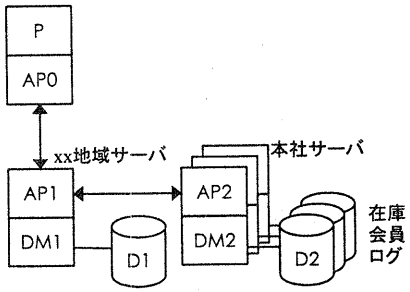

図 11　チケット予約システム

4.2 K 製薬株式会社　会計エントリ業務システム

10 拠点のサーバとホスト・システムで構成される会計データの入力システムである。各拠点に集められた会計データが適当なタイミングでホストに集める。

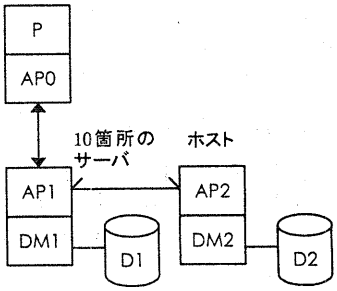

図 12　会計エントリ業務システム

図 12 で示す会計エントリ業務システムでは、非同期トランザクション・スタイルが適用可能である。

4.3 K 製薬株式会社　学術情報検索システム

営業支援システム用 PC から Web ブラウザによる学術情報の検索を可能にするシステムである。本社内と支店内の Web 用データベースは、一日一回、データベースの差分が転送されて、データベースの整合性が保たれている。

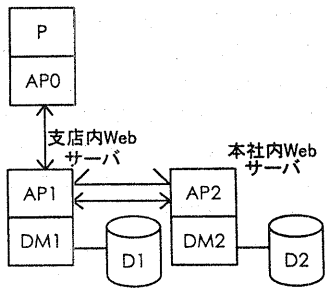

図 13　学術情報検索システム

図 13 で示す学術情報検索システムでは、分散依頼応答スタイルと非同期依頼応答スタイルの組み合わせが適用される。

4.4 C株式会社　フィールド・エンジニア支援システム

販売した商品の修理を担うフィールド・エンジニア(FE)の事務処理を支援するシステムである。修理依頼を受けた各地のフロント担当者は、受付データを入力し、サーバからFEのハンディ端末に訪問先を指示する。FEは、修理指示データをもとに現場に直行し作業を行い、修理結果を現場でハンディ端末に入力し作業完了書も出力する。作業完了データは、当日あるいは翌日にフロントサーバに送信される。フロントサーバに送信されたデータは、東京本社本部サーバに蓄積される。

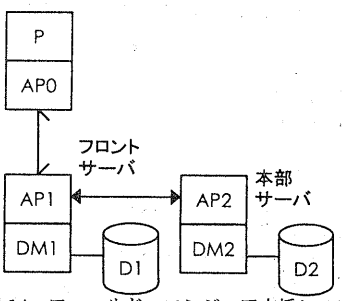

図 14　フィールド・エンジニア支援システム

図 14 で示すフィールド・エンジニア支援システムでは、分散通達スタイルが適用可能である。

4.5 C電力株式会社　お客様申込み工事支援システム

工事の申込みを支援するためのシステムである。システムの概要は次の通りである。

お客さまからの工事申込みの受付データはホスト・コンピュータの申込み内容データベースに蓄積され、一定時間ごとに営業所サーバへ自動的にダウンロードされる。営業所では、サーバ上のデータベースを使用して、当日分の施行予定データを区域別に自動的に振り分けて、各サービス・エンジニア分（携帯端末機器用）のデータを作成する。このデータは各携帯端末機へダウンロードされ、サービス・エンジニアはそのデータを基に施行する。工事終了後、サービス・エンジニアは施行結果を携帯端末機に入力する。帰社後サービス・エンジニアは、携帯端末に入力された施工結果を営業所のサーバにアップロードする。データの正当性が確認された工事竣工情報は、ホスト・コンピュータへ取り込まれる。

図 15 で示すお客様申し込み工事支援システムでは。受付業務は、集中トランザクション・スタイルそして工事支援システムは非同期通達スタイルが適用可能である。

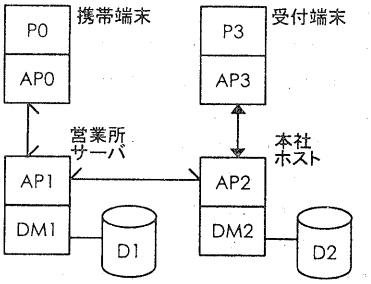

図 15　お客様申込み工事支援システム

5．適用の評価

現在本番稼働している複数のビジネス・アプリケーションのモデル化を実施した。実際のアプリケーション・システムでは、一つのスタイルのみに分類されるシステムもあれば、複数のスタイルが適用されている場合もある。複数のスタイルが適用されているビジネス・アプリケーションを評価するとシステムとして機能がてんこ盛りに近い状態になっている。しかし、サブシステム単位にみると単一のスタイルに分解することができる。またスタイル単位に分解することによってアプリケーション・システムの見通しが良くなる。

6．おわりに

データの所在場所とクライアントとサーバ間およびサーバ同士間の処理形態に焦点を当てて、分散処理システムを9つのアーキテクチャ・スタイルに分類した。この分類は、1996年に森澤が発表した分散処理システムの処理モデルを情報技術のその後の発展を考慮して見直した結果である。アーキテクチャ・スタイルの適用性と記述性を検証するために、本番稼働しているビジネス・アプリケーションのモデル化を試みている。また、企業の情報処理システムのインフラストラクチャを作成するための参照モデルとしての使用をも試みる予定である。

参考文献

1) Dargan, P.A.: Manager's Guide to Open Systems, The MITRE Corporation (1997)
2) 森澤好臣、岩田裕道、外山晴夫：分散処理システムの処理モデルの一提案、情報処理学会ソフトウェア工学研究報告、pp.17-24、96-SE-109-3 (1996)
3) 森澤好臣、岩田裕道、外山晴夫：クライアント／サーバ・システム構築のためのオープン・ソリューション・フレームワーク、日本ユニシス技報、Vol.16, No.2、pp.15-33 (1996)
4) 所真理雄、松岡聡、垂水浩幸（共編）：オブジェクト指向コンピューティング、岩波書店（1993）
5) Morisawa, Y., Okada, H., Iwata, H and Toyama, H.: A Computing Model for Distributed Processing Systems and Its Application, Proceeding of 1998 Asia Pacific Software Engineering Conference, pp.314-321 (1998)