静 岡 大 学　博 士 論 文

分散型アプリケーションシステムの構築に関する研究

1999 年 2 月

大学院電子科学研究科

電子応用工学専攻

坂下善彦

論文要旨

ネットワーク環境の発展と共に，多数のコンピュータシステムを有機的に結合する分散型システム環境が注目されている．分散型システムでは，高いパフォーマンスとシステム構築の柔軟性が期待される．システムが広範囲に渡り適用され，機能や制御が複雑になるに従って，システム全体の運用及び管理が困難になってくる．

分散型システム実行環境の上にアプリケーションシステムを構築する方式は，処理の要求とそれに応える構成，及び通信方式を意識した機構に焦点が当てられている．特に，通信手段によりアプリケーションが連携する機構や，アプリケーションがネットワーク環境のどこに存在するのかを特定する手段，あるいはコンピュータシステムやデータベースシステムとの連携関係の取り方，等システムの構成要素との関係に焦点を当てた構築方式が分散型システムの実行基盤となっている．

一方，ビジネスあるいは業務システム構成法は，クライアント・サーバ型の構築法を基盤とするものが多い．処理を要求するクライアント側の多数のユーザと，それを受けて処理を行うサーバがネットワークを介して構成される．ネットワークを介してユーザが遠隔地から利用できる利点があるが，サーバに処理が集中してしまう．サーバ機能の部分がネットワーク環境に分散化して，サービス負荷の分散や保全性の向上を図るという期待には，未だ十分には応えられていない．

情報処理システムが，対象とする業務や生産活動に依存して個別に構築されていたこれまでとは違って，ネットワーク環境での情報処理システムは，自身の業務や生産活動に閉じずに，営業，設計，製造，販売など，あるいは他の企業活動とも連携して，活動の枠組みが広がっている．このために，システム自身が扱う業務や連携する相手との関係も変化している．この変化に応じて，システム自身が柔軟に変化して発展していくことが求められ，それを担うシステム構築の手法が待たれている．

本研究では，分散処理実行環境上に構築され，サービス機能群がネットワーク環境に存在して互いに関連しながら業務サービスを提供するシステムを，分散型アプリケーションシステムと定義して，これらの要求に応えるために次の機構を実現することを目指している．

(1) 業務あるいは生産アプリケーションシステムが，対象業務や置かれている状況に応じて，その情報処理システムを再構築することなく，提供するサービスあるいは機能を柔軟に変化させる．

(2) 複数の異なるサービス機能を備えたアプリケーションシステムを，ネットワーク

を介して協調的に業務連携させる.

これらの目的を達成するために, 分散型アプリケーションシステムは, オペレータ・インタフェース部，協調型システム基盤部，そしてサービスプラグイン機構部から構成される，としてその構築手法の「分散型アプリケーションシステム構築アーキテクチャ」を提案した.

オペレータ・インタフェース部は，ユーザが直接に業務サービスを利用する環境となる．業務サービスを選択し，その実行を管理・制御するために，この部分は，業務が扱う情報を編集し，業務システムの状態を表示する．一方，システムに対するユーザの要求や意図を受付ける．このように情報処理システムとユーザの接点となる.

協調型システム基盤部は，オペレータ・インタフェース部からの操作イベントに基づき，業務作業の基本的な流れを規定するシナリオ解釈して業務の遂行を管理・制御し，業務を遂行する上で必要な役割をシナリオから抽出する.

サービスプラグイン機構部は，抽出された役割に基づき，プラグインの概念に基づく結合･連携機構により各種のサービスを結合して実行する. 抽出された役割を基に，ユーザの意図とシステムが置かれている状況を反映しその処理を実施するために必要な機能を，ネットワーク環境に存在する機能群から選択的に選んで起動実行する.

オペレータ・インタフェース部では，業務処理とシステムの内部状態を連携させるために，情報を表示する手段として，文書処理やグラフィックスで用いる表現手法により行い，その形態的情報をレイアウト構造により管理した．これにより，ユーザの操作による意図がトークンのような論理的な情報へと変換され処理部へと渡される．ユーザの操作は，予め定義されている業務の流れを現す業務フローを参照して，実行すべき業務が指定される．同様にして，システム内部の情報は，内部処理との関連を文書処理における論理構造に基づき処理と関連して定義する．処理結果は，この論理構造に対応するレイアウト構造に関連付けられ, 形態的表示情報となって表示される. これにより, 情報とオペレーションの送受を行うアプリケーション環境が構築される.

協調システム基盤部及びサービスプラグイン機構部では，業務フローに基づくシナリオを，業務実行の目的と手段を規定する役割に基づいて解析して，その実行機能を選択的に起動する. この役割を導入することにより, 業務システムの振舞いと機能は，独立に管理され，実行時に結合される．直接的に実行機能を司るモジュールを規定するのではなく，その機能を表す役割を指示することにより，その時のシステム環境やネットワーク環境により対応可能なあるいはその時のユーザにとって最適な実行モジ

ュールが選択されて結合される.

本研究の課題の一つであるアプリケーションの実行と制御の機構に関しては，オペレーションインタフェース部で示した形態的情報による表示と操作への対応と論理的情報構造による処理との関連付けにより，両者を独立に定義・管理し実行時に結合して制御する機構が確立した.

次の課題である動的再構成に関しては，協調システム基盤部で示した振舞いと機能を分離したことにより，特に機能部分をネットワーク環境上で動的に再構成できる構成となった．ビジネス系のシステムへの適用と制御システム系のシステムへの適用を行い，業務の仕様が変化することに対する情報システムの対応が旨く機能することを検証した.

また，サービス機能間の連携機構に関する課題は，役割を枠組みとした分散処理の場に基づく協調的実行機構により，自立的に振舞うモジュールが協調的に機能する基盤を構築した．同じ制御システム系による適用で，役割を共有して協調的に振舞うことを確認した.

本研究では，役割が備え持つ枠組みを利用して，目的と手段を分離して定義し管理し, 実行時に結合することで, システムの柔軟な構築を可能としている. 本方式は，ソフトウェア工学の領域で使用される，オブジェクト指向技術に基づくシステム分析と設計で広く採用されている方法論（OMT）の持つ概念と，極めて親和性が高いため，情報処理システムの設計段階から実装段階へと，一貫性良く設計の意図がスムースに渡ることが期待できる.

目次

第１章　序論

1.1　研究の背景 1
1.2　本研究の目的 7
1.3　研究の成果と特徴 10
1.4　論文の構成 14

第２章　分散型アプリケーションの構成

2.1　緒言 18
2.2　分散処理システム構築の基盤 19
2.3　分散型アプリケーションシステム構築アーキテクチャ 22
2.4　従来アーキテクチャとの比較 24
2.5　結言 30

第３章　構造化情報の表示と操作

3.1　緒言 31
3.2　情報構造とオペレーションインタフェース 34
3.3　情報の構造化 36
3.4　表示と操作 43
3.5　生産活動の成果物 44
3.6　結言 47

第４章　動的サービス結合方式

4.1　緒言 49
4.2　情報処理システムのモデル化 52
4.3　アーキテクチャ 54
4.4　役割部からサービス機能の協調的選択 58
4.5　適用例 63
4.6　評価と考察 65
4.7　結言 69

第５章　　自律型モジュール制御
5.1　緒言 70
5.2　サービスプラグイン機構 72
5.3　協調機能 73
5.4　協調のアーキテクチャ 74
5.5　共同作業における役割 77
5.6　自律協調機構 79
5.7　課題モデルへの適用と評価 83
5.7　監視制御システムへの適用 88
5.8　結言 90

第６章　結論 91

謝辞 95

参考文献 96

研究成果一覧 103

1. 序論

1.1 研究の背景

1990 年代は，分散というキーワードがシステムをもっとも特徴付けるといわれる．それまでの情報処理システムが抱えていた多くの課題を解決すると期待されたからであり，これまでに実現しなかったことへの期待が大きかったからである．そこには，大きく２つの要素が存在する．一つは，多数のコンピュータをどのように有機的に結合すべきかという課題，そして他は，多様な機能，多様なデータベースへアクセス，そして多様のコンピュータへのアクセスが要求され，しかも，ネットワークの存在を意識させずに，手元に全てがあるかのように利用できることへの，要求である．このような背景の中から，分散システムの研究が盛んになされてきた．

1.1.1 分散型アプリケーションが出現した背景

分散処理の研究は，Xerox 社の Alto の開発に代表されるように，各ユーザにできるだけ多くの要求に対応できる処理能力と通信機能を備えた作業環境の提供が目標であった．このユーザ環境を備えるワークステーションは，パーソナルコンピューティングに徹することで，１人のユーザがコンピュータ資源全体を利用でき，個人に向けたサービスの提供を目指した．他方，個人ユーザが独立に利用するのみではなく，企業活動においては，パーソナル環境を基盤として複数のユーザが共同してシステムを使用したり，共通の資源や情報を利用することになる．この故に，例えば他のコンピュータに存在するファイルやプリンタにアクセスできなければならなくなる，等の個と全体のバランスを持った環境が必要になる．これが，分散ファイルシステムが必要とされた理由である[Morris86]．

さまざまな種類の複数のコンピュータシステムを結合して利用することから，このようなシステムを構成する上で，並列処理，負荷分散，資源共有，ネットワーク利用，等などの技術課題があった．情報処理システムの共通的な構成方法が，汎用的に成立するために，システムを構築する際の共通的な機構を担うものが，分散オペレーティングシステム(分散 OS)である．そこでは，位置透過性，耐障害性，並列性，負荷分散，データ変換，既存システムとの接続性，そしてセキュリティ，等の機能が論じられてきた[Maekawa91]．

その後，高性能なマイクロコンピュータが開発されて，コンピュータの処理能力は飛躍的に向上した．これらのコンピュータを多数結合して，全体としては極めて性能の高いコンピュータシステムが誕生する．これと並行して，UNIX に代表されるオペレーティングシステムが誕生し，UNIX との互換性や接続性を備えた，Mach[Black92]，Amoeba[Mullender90]，V-system[Cheriton88]，Chrus[Rozier89]，等の数々のオペレーティングシステムが開発され，優れたネットワーク機能や処理機能等に関する分散システムの研究開発が一層と加速された．

コンピュータを接続する手段としてのネットワーク環境から，社会基盤とも言われる地域や都市あるいは地球規模のネットワーク環境へと，その変遷はすさまじい勢いでネットワーク化が進んだ．単なるコンピューティング環境あるいは関連する資源を利用するという受身的な利用形態から，メイルあるいは Web に代表されるようにユーザ自身が情報を発信する，あるいは情報資源を提供するという能動的な利用形態へと大きく変革している．

コンピュータの発展とネットワークの発展は，当然それを利用する産業活動としての業務の形態にも大きな影響を与えている．従来の情報処理システムは，それを適用する業務の機能や仕様が決まると，この機能や仕様に合わせたアプリケーションソフトが開発されて一定期間は変化せずに利用された．しかし，現在では，情報処理システムを構成する機器やアプリケーションソフトは，市販品と呼ばれる多くの種類の中から選ばれて，機器やアプリケーションに組込まれたり，それ自身で機能するように構築されるようになってきている．他方，特にネットワークの発展により，その情報処理システムを構成する要素を極めて柔軟に取捨選択でき，システムを構成することが可能となってきている．

従来は，業務に適用する情報処理システムは，前述のように特定の情報処理部門が製作したシステムを受取り利用する固定したものであったが，ユーザ自身が業務自身を変化させることも可能になり，機能や仕様をより自由に変化させたい，という要望が強くなってきている．

1.1.2 分散型アプリケーションの分類

一般に，業務を司る従来の情報処理システムを特徴付ける要素は，業務の遂行上基本的な要素となるデータと手続き処理の視点から整理することができる． 即ち，データの型と処理の型が固定であれば，業務処理の手順は全て予め決められた手続きで行われる．データの型が固定で処理が不定の場合は，データの値に依存して処理手順が

決められる．データの型が不定で処理が固定の場合は，手順は定められた順序で行われるがデータの処理の内容が型に依存する．両方が不定の場合は，ユーザの意思決定に依存して作業が進行する．

情報処理システムを構成する重要な要素は，以下と考えられる．

1)情報を利用目的に添った最適な表現にする

2)情報に依った各種のサービス（処理，行動，など）を提供する

3)情報の備え持つ意味を反映して，サービスの間を連携してユーザの期待する支援を行う．

このように，情報処理システムは，そのシステムが単独で全ての業務や制御などの処理を行うことは少ない．システムの状況を表示して作業者がシステムの状況を把握し，その状況に応じた指示をシステムに出す，など作業者等のユーザとの間で何らかのコミュニケーションが行われながら処理が進行していく．両者が扱う情報は，コミュニケーションを行うための極めて重要な媒体である．情報に基づいて，次に採るべき処理あるいは指示がユーザの意思決定によって定まり，その期待あるいは目的に添うべく，システムは機能しサービスを遂行しなければならない．

1.1.3 分散型アプリケーションが持つ課題

(1) プリケーションがネットワークを介してそのサービスあるいは機能を提供するためには，アプリケーションが特定のコンピュータ環境で実行されるが，ネットワークを介して多種類のコンピュータから利用できる環境を整えること，あるいはアプリケーションが特定のコンピュータ環境ではなく，複数の色々なコンピュータ環境で実行可能なことが求められる．

(2) サービスシステムが稼動・運用されている段階で，それを実行しているコンピュータシステムが更新されたり，新たな機能が追加あるいは削除されるこが，起こり得る．現実には，特定の時期にサービスシステムを停止して，新たなコンピュータシステムに合わせて，プログラムを変更して対応する．あるいは，新たな機能を追加修正するために，同じようにプログラムを変更して対応するのが，一般的である．

(3) 業務システムを担う情報処理システムは，利用目的や手段および使用環境の出現により，これらを採用して適用させるために，その都度システムを更新して来た．現実の企業における生産活動は，種々の環境で活動しており，活動の相手や目的も状況に依存して変わる．具体的には，その日の商品項目が変わったり，生産処理の形態や

手順が変わることは日常的に起こる．このような，業務処理形態や扱う商品項目の変動に柔軟に対応する，という要求にはこれまではほとんど不可能として，対処していない．

(4) 現在は，ユーザにとっての使いやすさ，コンピュータの運用と管理の仕方，そしてアプリケーション技術が課題になってきている．業務あるいはサービスの一般的な構成法として，クライアント・サーバ型を起点として発展してきた．しかし，これは縦関係の構成であり，システムは必ず依頼する側と依頼される側という２者の従属関係で構成されることになる．現在のような，ネットワークの時代に入ると，業務におけるサービスの実行は，コンピュータ間で依頼される側が，複数の機能やサービスの実行，類似のあるいは互いに補完するサービスの間での連携，あるいは協調的な動作が要求される．

1.1.4 従来の分散処理技術の研究

分散処理技術は，これまでに述べたように主として分散オペレーティングシステムの研究を中心にして，分散処理システムを構築する基本的な環境となる，分散処理実行環境 Distributed Computing Environments: DCE が発展した．マルチベンダーによるシステム環境を，エンドユーザやアプリケーション開発者に意識させずに開発し実行させる機構を提供している．これによって，分散システム構築のための基本的な枠組みが確立した．次に，このシステム基盤の上に分散型アプリケーション・システムを構築するために，アプリケーション・システムがシステムを構成する要素を管理すべき項目とその管理機構を提供するオブジェクト管理機構 Common Object Request Broker Architecture: CORBA が開発された．これにより，システム資源の管理の基本的な枠組みが確立した．

アプリケーション・システムの実行に当たっては，予め決められた作業手順に沿ってシステムが実行される定型的作業形態から，業務の状況や生産活動状況に依って，諸作業の手順あるいは仕事の手筈を定め，それに従って作業が進行する作業形態が求められた．このような諸作業の手順あるいは仕事の手筈を，ワークフローと呼ぶ定義手法により定めて，実行させる機構が開発された．

また，これらの作業は個人が単独で行うことはまれで，多くの場合，組織あるいはチームなどの単位で共同で作業が行われる．このために，作業者の間で連携を取りながら仕事を進めることを，情報処理システムが支援することが期待されている．これが Computer Supported Collaborative Work: CSCW，あるいはグループウェアと呼

ばれるものである．現在は，コンピュータによる積極的な支援を得るまでには至らず，共同で作業する環境や手段の提供といった側面的な支援に，留まっている．

次に，アプリケーション・システムの構築手法の変遷を概観する．ここでは，ネットワーク環境が出現してからアプリケーション・システムの構築に大きな影響を与えたシステムの構成方法に焦点を当てる．オペレーティング・システムとアプリケーション・システムにより構成された情報処理システムは，処理の要求とユーザインタフェースを担うクライアントと処理を実行するサーバを基本とした関係により構成され，ネットワークを介してこのクライアント部分とサーバ部分をどのように構成するかを中心に発展してきた．

1.1.5 従来のシステム構築手法

アプリケーション・システムのシステム構成法における階層構造の変化を図 1-1 に示す．

(1) 2 層構造

分散処理アプリケーションシステム構築手法の，起点となったクライアント・サーバ型の構造である．それまでのいわゆる集中型のシステムの操作端末と中央処理装置の関係を，クライアントとサーバシステムをネットワークを介して結合した形態である．依頼するものと依頼を受ける側との関係で構成される前述のワークステーションサーバ型が基であった．また，ハードウェアの構成もこの 2 要素のソフトウェア部分と一体化して対の関係で発展した．

この構成方式は，汎用機時代の集中型の従来方式を，ネットワークを介して置換えるというアプローチが採られることの利点もあり，飛躍的な発展をした．主に，パーソナルコンピュータをクライアント側に用いる場合が多かったが，規模の大きなシステムやユーザ数の大きな構成では，性能上の制約が次第に障害となってきた．

(2) 3 層構造

業務システムあるいはビジネス系のシステムでは，データベースの利用が必須であり，この領域での処理機能は業務形態に依存して類型化されることが多い．このために，クライアント・サーバ型のサーバ部分が，処理機能部分とデータベース部分に更に分化して，ネットワークにより 3 つの部分が結合される形態がとられるようになった．この形態が採用されることにより，従来の集中型の中央処理部分のコンピュータシステムと接続できるようになった．この手段により，規模への対応や，既存システムとの連続性が増大して，特に大型のシステムに採用されている．

(3)新３層構造

これまでの構成では，サーバは単一であったり，サーバの間で連携した処理を行うことはほとんど無かった．本来，分散処理が求めた情報や資源の共有，あるいは複数サービスの並行処理等の実行は少なかった．ネットワークの発展により，極めて広範囲に渡り資源の利用が可能となってきている現在では，サーバ部分はネットワーク上に並列的に多数存在するとして扱うことになる．先のデータベースもサーバ機能の一部として見ることができる．このような構成を新３層構造あるいはN層構造と呼んでいる．

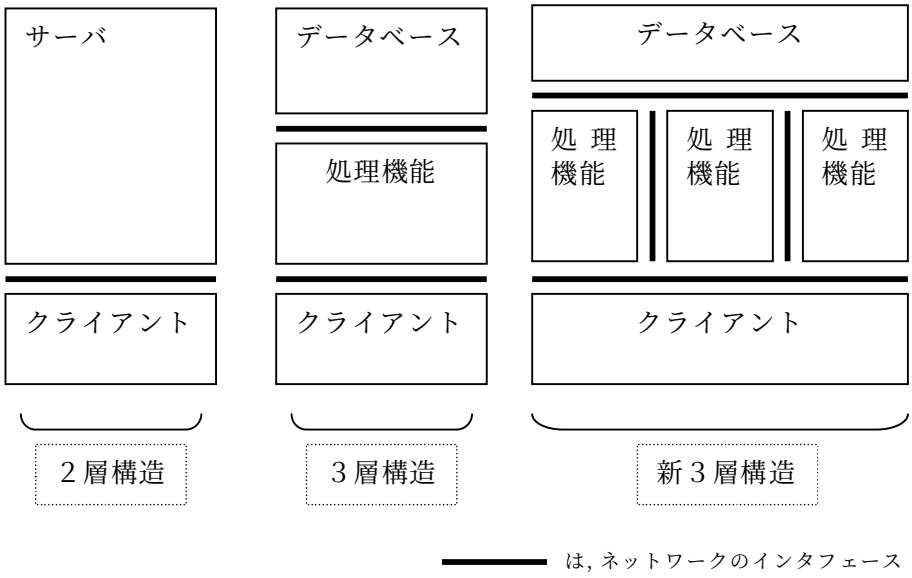

図 1-1 アプリケーション・システムの階層構造の変化

1.1.6 未解決の問題点と本研究で対象とする領域

分散処理のアーキテクチャ，実行基盤，あるいは資源管理等の分散処理システムの基盤環境は，盛んに研究されネットワーク技術と共に相補的に発展してきた．これらの技術は，業務向けのアプリケーション・システムを構築するための基盤環境であり，分散型のアプリケーション・システムそのものの構築手法とは，別枠のものである．他方，分散型のアプリケーション・システムの課題において述べたように，対象とする業務自身が，時間の経過と共に発展し変化している．これに関連してその実行を担う情報処理システムとしてのアプリケーション・システムが，柔軟に対応してシステム自身が進化するための，システムの構成方法に関してはほとんど未解決のままである．

情報システム構築の観点から，外部の変化に応じて自システムが自律的に進化して対応できるような柔軟な構築手法は，未だ経験も浅くそれを支える有効な方法論を見出していない[Suzuki97][Yurugi97][Harada97][Tokumoto97]．

業務の処理手順とそれを実施する機能との関連を保ちながら，処理を進行させるシステムの構築方法も十分ではなく，業務の処理手順を定義するワークフローに関する研究が進められているのみである[MfWC94]．更には，新３層構造でも触れたように，サービスを提供するサーバ側部分で，機能を連携させてより質の高いサービスを提供する，研究は今後の課題となっている[Sugawara94]．

事務処理システム，制御システム，あるいは産業システムにおけるそれぞれの業務システムを構築する場合に，ソフトウェア工学あるいはプログラム設計[Yonezawa86]の観点からの情報処理システム構築の方法論は多く研究されてきている[Aoyama90]．しかし，対象とする業務や生産活動を支えるサービスの形態や機能が時間の経過と共に進化していくことに応じて，それを担う情報処理システムを発展させていく方法に関する研究は少ない．

1.2 本研究の目的

情報処理システムは，対象とする業務や生産活動に依存して個別に構築されてきた．ネットワーク環境が発達した現在の情報処理システムは，特定の業務や生産活動領域に閉じずに，営業，設計，製造，販売などの部門を超えて，あるいは他の企業活動とも連携して，その活動の枠組みが広がっている．このために，扱う業務や連携する相手との関係に応じて，情報処理システム自身が柔軟に変化して発展していくことが求められている．

一方，クライアント・サーバ型の構築法を基盤としたビジネスあるいは業務システムの構成法は，ネットワークを介してユーザが遠隔地から利用できる利点があるが，サーバに処理が集中してしまった．サーバ機能の部分がネットワーク環境に分散化して，サービス負荷の分散や保全性の向上を図るという期待には，十分に応えられていない．

本研究では，これらの要求に応えるために，以下のような分散型アプリケーション・システムを構成する情報処理システム機構を実現することを目指している．

(1) 業務アプリケーションの対象業務や置かれている状況に応じて，そのアプリケーションを構築する情報処理システムを，再構築することなく，提供するサービスあ

るいは機能を柔軟に変化させる，新しい分散型アプリケーション・システムの構成方法.

(2) 複数の異なるサービス機能を備えたアプリケーション・システムを，ネットワークを介して協調的に業務連携させる，サービス連携機構.

1.2.1 新しい分散型アプリケーション構築方法の提案

分散型アプリケーションシステムは，サービス処理を要求するクライアントと，その要求を受けて処理を行うサーバの関係を基本とした型の構築手法を中心として発展してきた．その後，実際の業務処理では，情報を操作することが基本であることから，サーバの内部機能が，業務機能を実施する部分と，情報の格納と検索を行うデータベース部分に分化した．この両者の間はネットワークを介して連携する方式が主流となった．現在のネットワーク環境では，この依頼と実行の関係は，固定化せずに，業務の目的や流れに沿って必要な機能を用いて実行する関係へと変化している．

本研究では，新しい分散型アプリケーション・システムの構成する情報処理システムの枠組みを構築することを，目標とする．

現在の情報処理システムの構築手法に基づく，アプリケーション・システムは，業務の目的と，目的へ向けた作業の手順と，必要な機能を実行する機構が，明確に分類されてない．特に，手順と機能との関係を明確にすることにより，分離と結合の関係を構成できると考える．即ち，手順と機能の相互の関係を明確にすることにより，その業務を担う役割に基づき，基本的な業務処理機能を組合せるとこにより，業務サービスを提供することが可能になる．この業務を遂行するために，それを担う情報処理処理システムに対して，業務手順を外部から定義する手段を設けること，そして，この定義された手順に従って必要な業務処理機能が実行される機構とする．

1.2.2 分散型アプリケーション・アーキテクチャ

新しい分散型アプリケーション・システムの構成する情報処理システムが，備えるべき要素は以下である．

1) 業務情報と表現：情報処理システムが業務の担い手であるユーザと接する部分は，システムの内部状態や保有する情報を提示する機能と，ユーザがシステムに指示を与え必要な情報を与える機能を担うユーザインタフェース部である．情報の持つ意味を理解し易い形態で表現することと，システム内部の処理機能との関連を連携させること，及びユーザに操作の対象となる物を具象的に表現して，的確な指示を出

せる表示形態が要となる．また，これらの表現形態は，ユーザの部門あるいは個人により千差万別に異なるので，この要求にも柔軟に対応できることが求められる．

2) サービスの提供：ユーザからの要求に沿った業務処理サービスの提供，あるいは情報処理システムがユーザに要求する処理と，相互の間での交信により，業務処理やユーザの意思決定や情報提供（入力）などの業務手順が進行する．この進行は，必ずしも予め全てが決まってはおらず，それぞれの状況に依って異なるが，基本的にその目標は一定している．

3) アプリケーションの統合化：実行段階において，情報処理システムは，ここで注目する手順と機能が動的に統合化されて，全体として機能する．業務処理は，単一の業務遂行者（ユーザ）だけで行われることは無く，部門あるいは部所などの構成単位を基礎にして，複数のユーザにより実行される．このために，業務遂行の過程で，ユーザ間の一貫した判断や操作が行われる．

以下に，これらの要素に関する本研究の目的を述べる．

(1)情報の表現と操作

情報の持つスケルトンを文書処理における情報構造化の手段を利用して，内部処理が把握する意味体系を関連する処理の関係を保管し制御しながら，情報表現する．一方，情報要素をシンボル等の具象化手段を用いて操作対象を適切に表現し，ユーザの指示や入力操作において容易に識別判断ができる，情報表現を行う．これにより，業務サービスを選択しその実行を管理・制御する．

これらの情報表現におけるそれぞれの表現形態に対する，組織やユーザ個人の要望は千差万別であり，その要求に応えられる構成方法を確立する．

(2) サービス提供

ネットワークを介してシステムを構成することにより，情報処理システム提供，業務サービスの提供，業務に必要な情報の提供，等さまざまな形態の業務サービスを生み出している．業務の置かれている環境に依って，実施すべきサービス機能の最適なものにより実施することが，優れたサービスとして受入れられる．また，業務のサービス仕様が変化あるいは進化することに対しても，柔軟にそれらの変化に対応できる情報処理システムの構築手法を確立する．

更に，情報処理システムを再構築することなく，アプリケーション間を連携させながらそれぞれの機能を有機的に繋げて活用することにより，新しい業務サービスを実現する構築手法を確立する． アプリケーションをサービス機能単位として，それらを

組合せることにより新たなサービス機能を実現する機構である．新たな業務の手順や操作する対象データの処理機能を新たに定義する手段を設け，サービス機能単位を挿入して組合せることで，稼動する「プラグイン」方式の機構を確立する．

(3) 分散型アプリケーション・システムの統合化

業務の流れの中での関係，あるいは制御対象の状態に依存して，次に成すべき行為が規制される．このように，状況に適応して振舞う機構を基にして，サービス機能単位を連携させて新たなサービス機能を構築する．このために，前述の「プラグイン」を実行する際に，全てを記述的に定義して行うのではなく，業務における役割のような抽象的なレベルで成すべきことを定義して，実際の実行に当たっては，その状況に依存して最適なサービス機能を選択して，実行させることを目的とする．

1.3 研究の成果と特徴

1.3.1 新しいアーキテクチャ

制御と処理の両者の関係を分離して定義・管理し，実行段階で結合するために，新たに「役割」の概念を導入した「分散型アプリケーションシステム構築」の手法を構成した．

従来の情報処理システムに基づくアプリケーション・システムでは，業務制御の部分と，業務処理部分が混在して存在している．本研究では，この制御と処理の両者の関係を分離して定義・管理し，実行段階で結合するために，新たに「役割」の概念を導入した．この「役割」を媒体として，制御の部分と処理の部分を独立に定義して，実行時に再度関係付ける方式である．即ち，業務は役割に基づく活動単位により構成され，そしてこの役割に基づいて選択された最適なサービス機能が起動されて，業務が遂行される．このように，業務の処理手順は，目的の達成の為に，機能ではなく役割を単位要素として定義される．従来の分散型アプリケーション・システムは，サーバクライアント型を基本とする３層構造を基盤としているが，その業務機能部分に混在している制御処理の要素と機能処理の要素を，役割の観点から分離している．

この構造を備える分散型アプリケーション・システムを構築するアーキテクチャを提案した．本アーキテクチャは，ユーザが業務システム利用する環境となるオペレーションインタフェース部，役割により分離した業務制御と業務処理とを結合する協調

型システム基盤，そして実際の業務処理を実行するサービス機能をネットワークを介して結語するサービスプラグイン部から構成される．

1.3.2 情報の構造化による情報の表現と操作

オペレータによる処理の意図と，システムの振舞いの意図の接点としてオペレーションインタフェースを位置付けた．レイアウト構造による操作対象情報表現と，論理構造による情報の意味と内部処理の表現文書に２次元空間の概念：領域・フレーム・ブロックを導入し各種メディアを統合的に扱う手法によりレイアウト構造を定義表現と操作の間には処理の流れを制御する対話制御機能がある．

アプリケーション・システムのオペレーションインタフェース部分は，情報処理システムとユーザが唯一接する部分であり、情報や指示の交信を行うために重要な部分である．このオペレーションインタフェース部の中核を成すのは，各種業務や生産活動におけるあらゆる情報を表現する部分である．このために各種アプリケーション・システムで用いる様々な形態の情報を統一的に表現する手段が必要になり，各種の形態の情報構造を表現する機能と，各種のメディアを扱い２次元空間に表示する機能が求められる．

本研究では，上の要件を満たす情報表現の手法とした文書処理モデルを定義した．特に，直感的な視覚に訴える要素の強いレイアウト部分では，領域・フレーム・ブロックという２次元空間の概念を導入して，各種メディアを統合的に扱うことを実現している．文書情報の意味的な構成を表す文書構造を木構造で管理し制御している．また，文書を目に見える形態で表現するために，レイアウトの構造も同様に，木構造により階層的に管理し制御している．このことは，同じ文書を文書の論理構造とレイアウト構造の２つの異なる視点から見ていることになる．この両者の間の関係を一貫性を保って管理するために，両構造を文書内容を介して連携する構成とした．なお，この考えは，国際標準化機構 ISO/ODA（開放型文書アーキテクチャ）のモデル[ODA86]に反映された．

オペレーション・インタフェースの視点からは，情報の配置を管理するために階層的レイアウト構造を備えたことにより，自由な平面位置に任意の形態の情報を設定することができる．且つ，個々の情報単位が論理的に構造化されているので，関連する処理機能を論理構造に関連付けて管理することにより，関連処理と連携させることが容易に実現できる．

このような文書構造に基づき，例えば，設計図面あるいは設計情報を事業計画書に組み入れること，あるいは，現場の状況を示す写真やビデオ情報を，報告書に組み入れることが可能になった．従来は，設計部門の情報と製造管理部門の情報は分離独立して管理され利用されていたが，この機構を用いることにより，複数部門間での情報の連携ができ，情報と業務の管理がより高い次元で，管理されることになる．また，アプリケーションが備え持つ情報構造はそれぞれに特定の構造をしている．このために，アプリケーションの間で情報を交換する場合は，一般には両者の間で変換ツールを用意して，それぞれに変換する．しかし，N種類のプログラムに対して，${}_{N}C_{2}$ の数の変換ツールが必要となる．ここでは，共通文書情報構造を設けることにより，直接にアプリケーション間で変換せずに，定めた共通部署情報構造を想定してこの構造との間で送受する．このように，共通的なデータ構造を用意することにより，ツールの数は，基本的には対象プログラムの種類の数だけ必要になる．業種や業務形態が同じようなシステムにおいては，このような共通情構造を備えて，連携させることにより大きな効果が期待できる．

1.3.3 分散型アプリケーションの構成方法

従来の３層構造を基本とする情報処理システムの業務機能を実施する部分と，業務作業の流れの制御部分を，「役割」の概念により整理し分解した．業務の流れと実行処理の関係を，シナリオ・役割・サービス機能により構成し，振舞いと機能を分離して役割により管理しサービス機能をソケット「役割」へプラグインする構成とした．

本研究では，情報処理システムでも業務の処理手順などの振舞いに係わる部分は同じ概念で構成できる，と仮定した．この手法に基づく分散型アプリケーションシステムの構築の新たな枠組みを提案した．本アーキテクチャは，先に述べた情報表現，サービス，そしてサービス間の連携を基本的な要素として構築している．この基盤環境の上に,コンピュータシステムの構成方法とサービスの提供方法を繋ぐ仕掛けとして,サービスプラグインという概念を導入した．この機構により，サービス機能をシステムへ自由に組込むことで新たな業務サービスを実現する．これによって，ユーザが望むサービスをコンピュータシステムとは独立に用意して，それらをシステムに当てはめることにより容易にサービスの体系が構築できる．

既に述べたように，業務そのものが時間の経過と共に変化することに，情報処理システムが如何にして適応していくかが，大きな課題である．本アーキテクチャに基づ

き，対象とした業務の処理手順の変化，そしてサービス機能の変化に対して，どのように対応できたかを評価する．

他方，分散処理環境が研究されて，今日では，分散コンピューティング環境 DCE そして分散オブジェクトい管理アーキテクチャ CORBA 等の，プラットフォームと呼ばれる基本的な実行基盤が確立している．このような基盤に基づく，アプリケーションシステムの構築の環境は，次第に整いつつある．本研究においても，これらの基盤を前提としながら，業務や制御を司るアプリケーションの構築の枠組みを構成した．

1.3.4 アプリケーション間の協調メカニズム

プラグインの機構を協調的連携の観点から構成した．分散処理における場の概念に基づきその特性を「役割」により規定し，参加メンバーであるサービス機能群が自律的に行動する自律的に振舞う主体が協調の場を介して相互連携する方式とした．

自律的な協調を行うことを特徴とした計算モデルや，オブジェクト指向に基づくエージェントモデルの研究が，分散処理における問題分割や最適配置に向けて行われている．いずれの場合にも，メンバーの間で行われる各種の調整作業に関する手続きが，宣言的にあるいは特定のメッセージとして記述されている．

本研究では，システムの状態に対して原則的な対応ができる基本的な対処方式を備えることにより，その状況になれば必ず対応できる機構を目指している． 原則的な対応に相当する行動に相当する概念を，前述の役割に基づいて規定した．即ち，役割により目的を持ち，その役割を実現するための手段を明確に獲得できるからである．アプリケーション及びサービスの構成要素が，自律的に振舞う中で，特定の条件や環境に置かれた場合に，メンバー自身が備え持つ役割に従って，置かれている状況に反応する形態で振舞い，結果として構成要素の間で協調的に振舞うことに焦点を当てている．

1.4 論文の構成

第 2 章では，本研究と関連する従来の研究動向を整理し，提案する構築のアーキテクチャの全体を提示し，分散型アプリケーション・システムを構成するために必要となるシステム構築の全体的な枠組みについて論ずる．

従来の分散処理環境に基づくアプリケーション・システムを構築するうえで，特に重要なことは,分散処理環境に分散して存在する個別アプリケーションを連携させて,新たなサービス機能をユーザに提供すること,更に適用する業務の形態や処理方法が,変更されることに対して,実行する情報処理システムが再構築せずに対応できること,である．

本研究では，業務を担うアプリケーション・システムは，業務を構成する活動の単位で処理手順が決められ,この活動単位は目的を達成するための役割により規定され,そして最後に，この役割を実行するサービス機能により実際の処理が成されるとしている． このように,従来はソフトウェア工学の手法に基づく業務の分析が行われていたが，その成果を直接的にシステム構築結びつける方法にまでは至っていなった．提案するアーキテクチャでは，ソフトウェア工学の代表的な手法であるオブジェクト指向技術で用いる役割を，システム構築の領域に取り入れている．

この枠組みを構成する機構を，分散型アプリケーションシステム構築アーキテクチャとして，定義した．業務の振舞いと機能の関係を，役割により結びつける階層的構成としている．即ち，業務を遂行する作業者と情報処理システムの間は，情報の表示や業務に関する操作を行うオペレータインタフェース部により対応する．この操作に関連して，協調型システム基盤部では，対応する役割に基づき関連する業務との間の連携を管理し，実行に必要なサービス機能を特定する．更に，サービスプラグイン機構では，特定したサービス機能をネットワーク上に分散する環境上で実行させる．

従来の手法は，プラグインという概念は存在したが，明示的に且つ人為的に組込ませる手法に留まっていた．また，それを実現する環境は，分散処理実行環境と呼ばれる情報処理システムの基本的な実行の枠組みの範囲であり,システムの振舞いの中で,自律的に実行対象が選択されて，システム自身が組込む機構は存在してない．

本研究では，前述のアーキテクチャにより，システム自身が自律的に構成していくことを可能とした．

第３章では，アプリケーション・システムにおける情報の表現と操作の手段として文書を位置付け，文書が備えるべき機構と情報の表現手法について論ずる．システムとユーザの接点となるユーザインタフェース部において，ユーザからの要求をシステムへ伝えるために，操作イベントに依ってシステム内部に持つ業務の流れの定義に基づき解析しながら進行させる機構を設けている．また，システムの状態情報をユーザに提示するために，システムの内部状態情報をこの機構を用いて行う方式を構成している．ここで扱う情報はシステムの振舞いの意図を効果的に伝えること，且つ操作に基づき情報の構成要素単位で制御することを，目的に木構造に基づく情報の構造化を実現している．

文書処理は，文書内容の作成・編集，レイアウト・表示形態，可視化処理の段階から構成される．文書の概念を整理する上で，論理構造及びレイアウト構造がスケルトンを表す概念として利用されてきた．文書に構造を備えることにより，構成要素の特質や構成要素間の関係を定義することが可能となる．画像や音声も含むマルチメディア情報を含むあらゆるデータがネットワーク上に分散された資源としてオブジェクト化される．これらのさまざまな情報を，業務や仕事の処理や操作目的に沿った形態で情報を読み出して操作する環境として，この文書を位置付けている．文書構造とデータ資源との連携に注目している．

文書構造の機構としては，表示形態の視点からは，領域・フレーム・ブロックと呼ぶ２次元空間要素を単位として，複合階層構成をなすレイアウト構造を導入した．意味形態の視点からは，章・節・項などの文章の意味構造を部分的に備え持つ論理構造を導入した．両構造は，文書内容を共通項として互いの関係が保たれる仕組みである．

第４章では，処理機能を結合させて起動・利用するシステムの構成方法とサービスの提供方法を連携する仕掛けとなるサービスプラグインの概念とその構成と機能を論ずる．ユーザの要求を受けて，分散ネットワーク環境に存在する各種のサービス機能を有機的に連携して，目的の機能を実行するために，役割の概念を基にして，業務の処理の流れを表すシナリオと，サービス機能を動的に連携させる機構を提案している．そこでは，役割に基づき選択されたサービス機能を，プラグイン機構によりシステムに結合し連携させて，機能させる新しい方式を構成している．複数の業務システムに適合して，その方式を検証している．

アプリケーション・システムは，シナリオと役割と機能から構成される情報処理システムのモデルとした．シナリオはワークフローとも言われ，業務処理の手順や作業

の流れに相当する．他方，従来のクライアント・サーバ型のネットワークシステムの構成においては，サーバ側に存在する業務ロジックと呼ばれる部分が，アプリケーションシステムの重要な部分であった．そこには，業務を構成する基本的な機能が存在し，業務処理を複合的に構成していた．前述のようにこの部分には業務の要が漠然として存在していた．

本研究では，この業務を構成すると言われている部分を，役割を基にして業務の手順に相当する部分と実行機能の部分を分離して管理する方式を基とするアーキテクチャを構成した．提案したアーキテクチャに基づき，事務処理系のシステムと産業システムに適用して構築した．対象業務の処理手順に関する変更，及び提供する機能の変更に対して，構築したアプリケーション・システムがどのように対応できたかを，影響の範囲とその両の観点から評価した．

第５章では，役割を行動モデルとする協調的なアプリケーション連携の実行基盤の機構について論ずる．サービス機能を結合し連携させる方式として，自律協調の機構により，対象とするサービス機能を選択し，更に複数のサービス機能同士で協調的に振舞う機構を検討し,自律的に行動する形態と協調的に行動する形態の特徴を整理し，分散処理における場の概念に基づく協調連携の機構を考察している．

本研究では，振舞いと役割の関係を，制御の分野での典型的な共同作業の課題モデルを対象にして，検討を進める．その課題モデルの１つである「神輿担ぎモデル」では，類似の機能と能力を備え持つメンバーが，共通の目的に向かった共同の作業を行う．他は,「ごみ集めモデル」であり，そこでは類似の機能と能力を備え持つメンバーが，同一の目的に向かって行動を起こすが，最終的には，何らかの仲介調整が行われて唯一のメンバーがその任務を担う．

これらのモデルに対して，シナリオ・役割・機能の枠組みを導入するものである．即ち，シナリオが全体的な振舞いの枠組みを与えることにより各段階での行動に副目標を与える，この副目標に沿って関係する役割を備えたメンバーが集まる，そこで各メンバーがどのように振舞うかが，副目標と役割の関係からどのような形態の共同作業を行うかが規定される，そしてこの実際の作業を実施するための機能サービスが適用される仕組みとなる．

協調的処理を行う機構を分散処理システムの「場」の中に組込んだ．作業に酸化するメンバーが置かれている環境を「場」として位置付け，この枠組の中で目指す目的に沿った管理と運営を場の管理者が司り，実行する機能や手段を規制する枠として役

割の概念を導入した．場の管理者が関連する役割を持つメンバーを募り，これに応えたメンバーが参加する候補となり，最終的には管理者がその都度与えられる判断規準によって選択してメンバーを決める．ここでは，同じ役割を持つメンバーが統合されて役割を共有する形態となる．本論では，適用対象の状況に応じた作業を行うことに注目している．予め決められた作業に添って作業するのではなく，その時々の置かれている状況と目標との関係で，とりうる作業の形態が動的に決まることにより一層と効率的に目的を達成できると考えている．換言すると，対象と自身の関係において，対象からのフィードバックを得ながら自身の行動の形態を動的に調整していくことである．

重要な概念は，メンバーが行う振舞いの結果が，制御対象物に影響を与えそれが存在する場にも反映されることである．即ち，メンバーと対象体との間には相互関係が存在し，対象体を介して他のメンバーへ影響を与える結果となる．これは，メンバー同士の直接的な連携ではなく対象対を通した関係であり,本論では制御の伝播と呼ぶ．ここでは，関連する役割の間での接続関係が重要であり，それぞれの役割が実施された結果が次の役割の入力条件となっている．このようにして，振舞いの流れが役割の列として存在する．

最後に，６章において本研究をまとめる．

情報処理システムを構築する際に，全ての仕様を事前に把握することは不可能に近い．ネットワーク環境の飛躍的な発展に伴い，システムを利用する環境やその方法は，時代や環境と共に大きく変化する．このような変化は，当然仕様の追加･削除・変化を要求することになる．

本論では，特定の条件が整えば自律的に行動が起こせる機構を情報処理システムに備えることにより，仕様の変化が発生してもそのプログラム，サービス機能，あるいはモジュールは機能することを目指してきた．このために，適用するアプリケーション・システムが備え持つべき特質を論じ，それを実現する分散型システム構築のアーキテクチャを提案した．

アプリケーション・システムは，様々な業務に利用される．このような広範な利用に対応できるために，システムとシステム，システムと人間，あるいは人間と人間，の間で，優れた協調関係が重要になってきている．この協調関係を情報処理システムが，どのように支援できるかは大きな課題である．自律的そして協調的に振舞うために，役割の概念に基づく振舞いの制御機構を設けた．

２章 分散型アプリケーションの構成

2.1 緒言

分散処理環境における情報処理システムの構築手法は，アプリケーションシステムが分散化されたネットワーク環境で振舞うための基本的な枠組みを提供することを中心として発展してきた．その一つは，コンピューティングシステムがネットワーク環境で構築されるために，共通的なシステムの枠組みを確立するものとして研究されてきた．他の一つは，このコンピューティングシステムの構成要素が，ネットワーク上に広い範囲にわたって存在するという特徴から，その論理的な位置付けと物理的な位置付けの関係を保ちながら，システムの実行時に，先の関係に基づき実行される機構を確立するものとして研究されてきた．

業務システムのようなアプリケーションシステムは,上の枠組みの上に構築される．その基本的な枠組みを与えているのが，クライアントとサーバの関係によるアプリケーションシステムの構成法である．このアーキテクチャは，依頼する側と依頼を引き受けて実行する側という極めて単純な関係に基づいている．

しかし，ネットワークシステムが発展するにつれ，それらを構成する要素の互いの関係が必ずしも，クライアント･サーバの関係では成立しなくなってきている．その大きな理由は，関連する構成要素は，他者が作成したものも含まれいわゆるオープンな環境になり，それらの要素を利用できる環境を整える必要が生じている． このように，システムの状況に依存した形態で実行が行われるという特徴を持ったアプリケーションシステムを構成するには，以上の基本的な環境基盤の上に，新たな構築手法としての枠組みを設けることが必要である．

本章では，本研究の基盤になる，業務サービスあるいは制御に必要な機能をシステムに差込んで利用するサービスプラグイン機構の構築アーキテクチャを示す．

2.2 分散処理システム構築の基盤

本節では，分散型情報処理システムの基礎ともなるシステム基盤に関して，これまでの分散処理研究の中で，特に高信頼度分散コンピューティングに対するプロセスアプローチ[Kenneth]に焦点を当てて概観する．

信頼性のある分散ソフトウェアの開発の困難さは，多くの対象に分散コンピューティングを適用する上で障害となっている．ISiS[IsiS92]の経験を基に，仮想同期プロセスグループに基づく構造化のアプローチは，分散システムの開発を容易にし，信頼性の高い自律的なシステムを構築することを可能にしている．

(1)プロセスグループ

Anonymous Group と Explicit Group の２つの型のグループの利用方法がある．Anonymous Group は，アプリケーションがトピックスに関する情報を公開したときにグループが構成される．そのグループには他のプロセスが参加することができる．アプリケーションに対して，自動的に信頼性の高い操作を行わせるように，次のような特性をこのグループは持つ．

1) グループアドレスを用いてグループに対してメッセージを送る
2) メッセージは厳密に１回だけ配送される
3) メッセージは順序正しく配送される
4) グループに出現したイベントのヒストリを管理しなければならない

Explicit Group は，プログラムの集合による仕組まれた行動で陽に協調する．例えば，フォールトトレラントなサービスでは，現在のプライマリに障害が発生した場合，何らかの行動をとるプライマリメンバーを持っておりバックアップ手段を備えており，メンバーシップの変更はグループメンバーによる行動を起動する．同じ変化が同じ順序で全てのメンバーに見えなければプライマリが存在しない，でなければ複数のプライマリが存在してしまうことになる．分散プロセスグループに基づくソフトウェアを開発する際には，以下の技術的課題に留意する必要がある．

1) アドレッシング，障害原子性，そしてメッセージ配信順序を含むグループ通信
2) 複数のグループメンバーを並列に扱うために，分散アルゴリズムの入力として，グループメンバーシップやメンバーシップの変更が利用できる
3) 大域的に正しい動作を得るために，アクションがとる順序を正しく同期させねばならない

(2)メッセージパッシングによる分散サービス構築

1) 従来のメッセージパッシング

先進的な OS が備える通信サービスのクラスは次の 3 点である.

a. 信頼性のないデータグラム．即ち，壊れたメッセージを自動的に無視する.

b. 遠隔手続き呼出．通信は結果を返す手続きのための起動として表現される.

c. 信頼性のあるデータストリーム．フロー制御による順序配送などの実現.

2) 従来技術に基づくグループ構築

a. グループアドレッシング

プロセスがグループに参加したり脱退したりするために動的に変化するアプリケーションにおいて，グループアドレスをメンバーシップリストに対応付ける問題に注目する．メンバーシップサービスは，グループ名からメンバーシップリストへの対応を管理する．プロセスは，ネームサービスに転送することによって，あるいはキャッシュされたメンバーシップ情報を検索することにより，入手しメッセージを送信することができる.

b. メッセージ配送順序

メッセージが配信される順序に注目する．一般にアプリケーションは，それぞれのメッセージ間での協調は無く一貫性無く動作する．実際，設計者は，起こり得るメッセージ順序の不一致を無くすようにしなければならない．また，順序の整合性が取れるまでサーバ内で処理を待たせるなど，明示的に回避する手段を講じなければならない.

c. 状態伝送

新しいメンバーにサービスの状態を転送する場合の課題に注目する．新しいメンバーがグループに参加した時より以前に，更新によって変更されるため，サーバの状態はサービスによって管理されているデータを反映したデータを持っている．即ち，メッセージはメンバーシップが変化するのと並行してサーバに送られて，直前の状態を反映した状態情報を得られない可能性がある.

d. フォールトトレラント

障害は多くの問題を発生する．良く知られているアルゴリズムは，Nonblocking 3 Phase Atomic Commitment Protocol に基づくものである．障害を管理する原理は以下である．最初に宛先は送り手の障害を検出する手段を持っているものと仮定する．障害が発生した場合に，第 1 ラウンド，あるいは第 2 ラウンドのメッセージを受信したプロセスはプロトコルを停止できる．障害が発生した場合に，その送り手が動作していたラウンド以上に動作を続ける宛先の集合のうちのいくつ

かのメンバーを確保することである．そのラウンドで既にメッセージを受信しているプロセスは重複を検出することが可能で，それらに対してオリジナルのメッセージの受信を応答することができる．しかし，実際には障害の検出は容易でない．

3) 仮想同期

ISiS は，プログラマに対して，分散実行の擬似同期スタイルを仮定することを推奨している．シングルイベントはグループの複数のメンバーによって観測されるとしている．

a. プロセスの実行はイベントの流れからなる．それらは，内部処理であっても，メッセージ転送，メッセージ送出，あるいはメンバーの変更であっても良い．
b. システムのグローバルな実行は，プロセス実行の集合である．グローバルなレベルでは，メッセージをプロセスグループに対するマルチキャストと考える．
c. 同じグローバルイベントを観測した２つのプロセスは，同じ順序で対応するローカルイベントを観測する．
d. プロセスグループに対するマルチキャストは，そｎグループ全員に配布され，システムによる配布がスケジュールされたときはその時点で解釈される．

擬似同期実行は分散アプリケーションの設計を簡単化するが，現実の環境に適用するにはコストがかかり過ぎる．多くの問題は，メッセージの送信と配布の関係ある．分散システムでは，高い性能は非同期な相互作用に依拠している．例えば，送り手からのメッセージが配布を待たずに連続的に送信されることが許される形態である．非同期のアプローチは，通信システムを有限のバッファのように扱っており，送り手がブロックされるのは，データの生成率が消費率を越えた時，あるいは送り手が応答や他の入力を待つ場合に限られる．この手法の利点は，送り手と宛先との間の遅延がデータ転送率に影響しないことである．即ち，システムはパイプライン化された方法で，送り手と宛先とが連続に動作することを可能としている．しかし，擬似同期実行は，このようなパイプライン化やプロセスの実行の遅延を排除している．

2.3 分散型アプリケーションシステム構築アーキテクチャ

分散型アプリケーションシステムを構築する方式は，ネットワークの環境の要素を意識した機構に焦点が当てられて提案されている．即ち，通信の手段によりアプリケーションが連携するための機構や，アプリケーション自身がネットワーク環境のどこに存在するのかを特定する手段，あるいはコンピュータシステムやデータベースシステムとの関係の取り方等，システムの構成要素との関係に焦点が当てられたアーキテクチャとなっている．

他方，ビジネスシステムあるいは業務システムと言われる，クライアント・サーバ構成に代表されるように，事務処理分野あるいはトランザクション処理の分野に特徴的な処理形態を反映して，ユーザ操作部，業務処理部，そしてデータベース部とそれぞれの処理機能の単位で構成される形態を採っている．

本論で提案するアーキテクチャは，汎用的なコンピュータシステム環境の提供よりも，顧客の要望する業務システムを提供する際に，業務システムの利用の仕方や目的，あるいは業務自身の形態が時代とともに変化していく中で，その業務システムを再構築することなく，利用環境にコンピュータシステムを適応させていくことを目指している．

この目的を達成するために，業務を遂行する枠組みに沿って必要な処理機能を結合させて起動・利用するために，コンピュータシステムの構成方法とサービスの提供方法を繋ぐ仕掛けとして，サービスプラグインと呼ぶ概念を導入して，業務連携を実現している．そしてこれらの処理機能の集合である業務処理部，そして分散システム環境部から構成される．なお，システム運用管理部は，業務システムを利用する組織とコンピュータシステムの各種資源との対応管理を司り，そして利用の許可権を管理制御する．

業務あるいは制御には作業あるいは行動の基本形が存在すると仮定し，この基本形を基軸にして適用対象に適合したサービスや機能を選択的に提供できる機構を実現する．同時に，個別の機能やサービスの間で協調的に連携することによりより効果的なサービスや機能を提供することを目的とする．

図２-１に，本研究において提案するシステム構築のアーキテクチャを示す．以下に，その構成要素に基づき説明する．

A．オペレーションインタフェース

ユーザ・オペレータがシステムの状態を見て，それに基づく処理を行う部分．システムの持つあらゆる情報を表示し，ユーザからの操作を受付ける．

業務や制御の基本的な筋書きとでもいう業務活動のシナリオを定義し，このシナリオに沿ってユーザはシステムの状態を見てそれに基づく処理を行う．

B．協調型システム構築基盤

与えられた役割の実行を担う機能群を協調的に選択管理する制御部分である．指定された役割に対応できる機能を分散型ネットワーク環境から資源として集めて，最終的に起動する制御を行う．また，シナリオ内部存在する振舞いに関わる要素において，同時並行の実行，順序実行，あるいは合同した実行などの実行管理を行う．

C.サービスプラグイン

シナリオを担う主体は役割である．役割を単位としてこのシナリオを理解して実行に移す．実行を担うのがサービスそのものである機能である．本アーキテクチャでは，役割をソケットとして，必要な機能がプラグインされる構造となる．これにより，シナリオ・役割・機能が連携して業務が処理される．

D．システム運用管理

分散処理システムにおいて基本的に必須となる，ユーザの管理，データ及び

業務アプリケーションの管理，等を制御する．更に，ここでは，業務あるいは制御に関連するシナリオの管理，そして機能群の管理がこれに加わる．

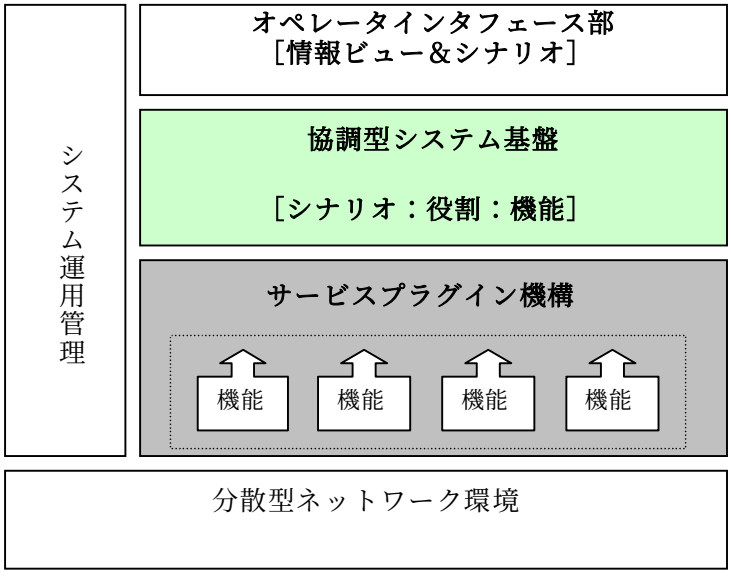

図 2-1．分散型アプリケーションシステム構築アーキテクチャ

2.4 従来アーキテクチャとの比較

ネットワーク環境に構築される分散型コンピュータシステム環境を，汎用的に構築するアーキテクチャとしては，OSF/DCE と OMG/CORBA が代表的である．

2.4.1 分散コンピューティング環境

OFS/DCE Open Foundation System/Distributed Computing Environments

UNIX を中心とするソフトウェアの標準化を推進する非営利団体 Open Foundation System:OSF が提供する分散コンピューティング環境におけるプラットフォームである．この環境では，利用者はネットワーク上に分散して存在している資源がどこにあるかを意識せずに，利用できる．構成する技術は，図 2-2 のように，オペレーティングシステムとアプリケーションの間に位置するソフトウェアであり，マルチベンダーのネットワークによる環境を，エンドユーザやアプリケーション開発者に意識させずに，アプリケーションプログラムを開発し，実行させることができる．このアーキテクチャの構成要素の概要を以下に示す．

(1) スレッド

分散環境下で並行処理を実現する」ために，アプリケーション内の情報の流れを制御する．スレッドサービスは，１プロセスの中に複数のスレッドを生成して並行処理を行う．各スレッドが共通にデータをアクセスする場合，データへのアクセスの同期を制御する．

(2) RPC: Remote Procedure Call

分散するプログラム間の通信機能を提供する．Interface Definition Language:IDL コンパイラ，ネームサービスとの統合，ネットワーク独立，信頼性の高い RPC，大量データ転送，スレッド機能などの特徴を持つ[Birrell84]．

(3) 分散時間管理

分散環境下の全てのクライアント，サーバの時間を同期化する．LAN 及び広域網で時刻が同期化されたり，RPC，ディレクトリ，セキュリティ等の他のサービスと統合される特徴がある．

(4) ディレクトリサービス

分散環境下にあるサーバ，ファイル，ディスク，プリンタ，等の各種資源をネットワーク上のどこに存在するかを意識させずに，分散環境全体の中で単一の名前で識別する．

(5) 分散ファイルサービス

ネットワークに分散しているファイルをローカルに在るファイルと同様に利用できる．

(6) ディスクレスＰＣサポート

ディスクを備えないワークステーションから，DCE の機能を利用できる．ブートストラップやファイル転送の機能を利用して行う．

(7) セキュリティサービス

分散環境下にあるデータの完全性と機密を保証し，分散環境への無許可でのアクセスを防止する．このために，認証サービス，許可，登録サービス，及び高信頼な RPC の機能を提供する．

(8) 管理サービス

分散システムを構成する資源を管理する．Distributed Management Environments: DME の機能により実施される．具体的には異化の管理項目がある．

1)ソフトウェアの配布と実装サービスの管理

2)分散印刷管理

3)ネットワーク管理

4)分散資源管理

5)分散イベント管理

6)ＰＣ統合管理

7)管理情報格納

8)管理アプリケーションインタフェース

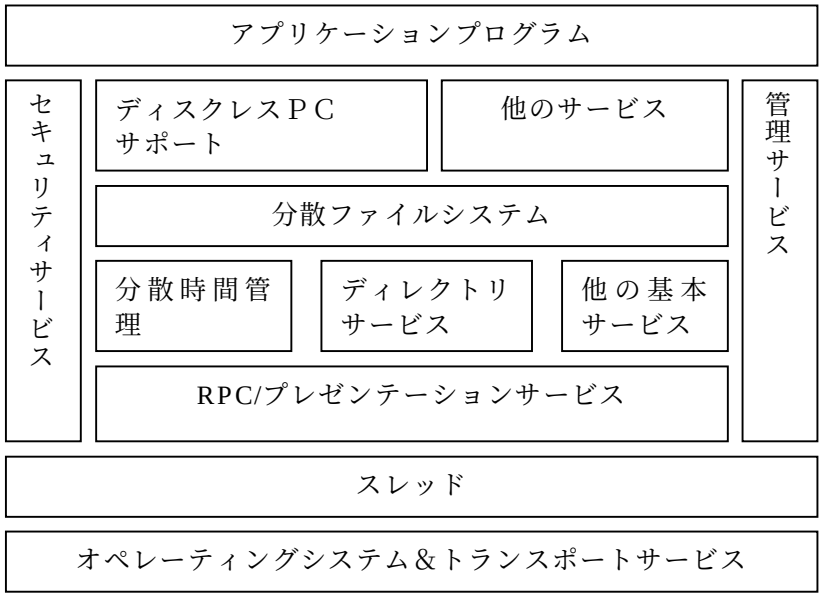

図 2-2 DCE アーキテクチャ

2.4.2 分散オブジェクト管理

OMG/CORBA: Object Management Group/Common Object Request Broker Architecture

ネットワークの普及によりさまざまなアーキテクチャの情報処理システムの間で情報交換を行う統一的な仕組みが必要になってきた．そして，分散コンピューティングによるソフトウェアの複雑さの増大に対する策としてオブジェクト指向技術が注目されてきた．このような背景を持って，提案されたモデルである．

その参照モデルは，図 2-3 に示すように，以下の５つのコンポーネントからなる．

ORB:オブジェクト・リクエスト・ブローカ

OS:オブジェクトサービス

CF:コンモンファシリティ

DI:ドメインインタフェース

AO:アプリケーション・オブジェクト

この参照モデルでは，ソフトウェアはオブジェクトとして存在する．オブジェクトは，ある時は他のオブジェクトへリクエストを投げかけるクライアントとして，ある

時は他のオブジェクトへサービスを提供するサーバとして振舞い，分散上に存在する複数のオブジェクトが互いに利用し協調しながら，処理を進める．

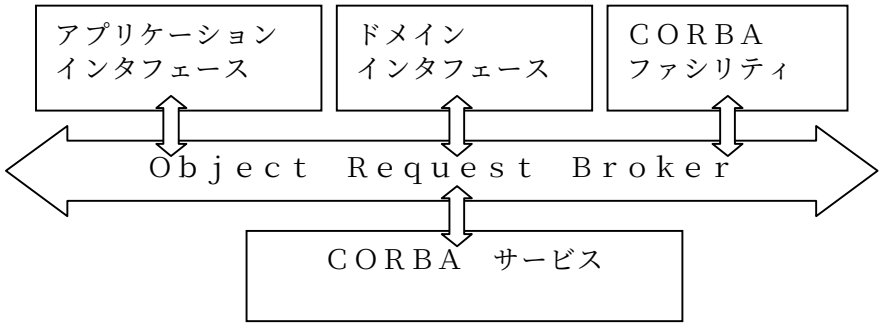

図 2-3　Object Management Architecture

2.4.3 クライアントサーバ

情報処理においては，通信し合うソフトウェア間で，要求を出すクライアントと，要求を処理するサーバという関係が現れる．この場合のアプリケーションプログラムの構成法は，次の利点を持つ．

(1) 多数のプログラムをクライアントに対応させて実行させる必要が無い．
(2) プロセス間通信を統一することによって，一貫したシステム構成が可能で，システムの開発の見通しを良くする．

このシステムの構成法での通信は，クライアントがサーバに要求を出しサーバが応答を返すのを待つ方式，及びサーバが複数のクライアントからの要求を受付けるという方式が基本となる．このようなクライアント・サーバ型通信は，システム内で共通に現れる基本的な操作であり，これを効率良く処理することはシステム全体の性能向上につながる．

図 2-4 は，クライアントの操作を同一のプログラムの手続きの呼出しと同じように指定できるようにした遠隔手続き呼出し(RPC)によりクライアント･サーバ型通信を実施する形態を示す．この構成法におけるサーバへの要求は以下である．

(1) 複数の同時に発生する要求をサーバ側でスケジュールして処理できる
(2) 要求された処理を実行し応答を返す順序は，要求の順序と独立である

(3) 要求を透過に他のサーバへ転送することができる

(4) サーバは，クライアントに対する応答で封鎖しない

(5) クライアントの機能停止がサーバに影響を与えない

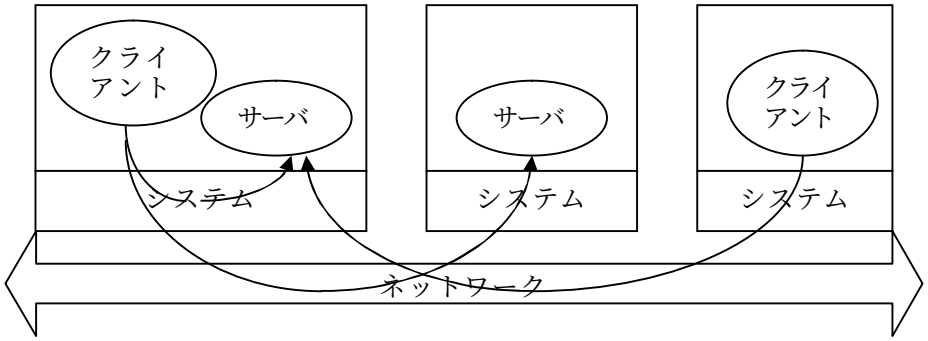

図 2-4. クライアント・サーバ型通信

本構成法では，入出力操作や帳票処理などのユーザ操作を主に司るアプリケーションプログラム部分をクライアント側に配置し，それ以外の実行部分をサーバに配置する例が多い.

2.4.4 3層構造

本構成法は，クライアントサーバモデルを基にしているが，業務あるいは制御システムの機能の分担により構成されている．特に，事務処理システムにおいては，扱うデータの処理が重要であり，その処理方法が適用対象業務により幾つかの基本的な業務処理機能に分類されて，機能部分あるいはビジネスロジックと称される．そして，データはデータベースから読み出され，先の機能部分で処理され再びデータベースに格納される. このような3つの構成部分からなるので3層構造と称されている. 図 2-5 に，3層構造の構成を示す.

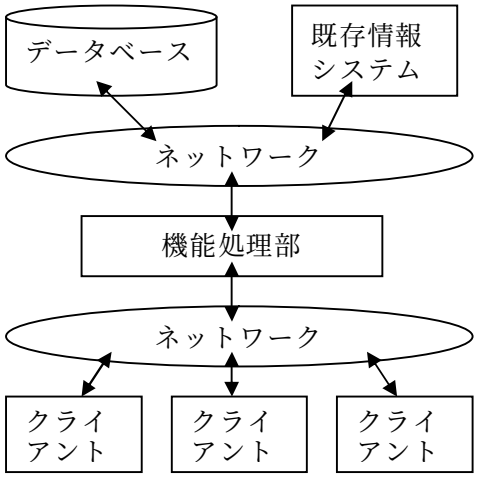

図 2-5. 3 層構造

本アーキテクチャの特徴は，各層間にネットワークが存在することである．機能部分とデータベース部分が分離独立したことにより，データベース部分が，旧来から存在する所謂汎用機で構築された既存情報システムにとって代わることができることとなった．これによって，従来システムのとの連携が可能となって，飛躍的に情報処理システムが新しい構成法と旧来のシステムとの連携が実現されたことにより，情報処理システムの新しいシステム構成法の発展に加速がかかった．

2.5 結言

本研究の基盤となる分散型アプリケーションシステム構築のためのアーキテクチャを述べた．ネットワーク環境が出現してからアプリケーション・システムの構築に大きな影響を与えたシステムの構成方法に焦点を当てた.オペレーティング･システムとアプリケーション・システムにより構成された情報処理システムは，処理の要求とユーザインタフェースを担うクライアントと処理を実行するサーバを基本とした関係により構成され，ネットワークを介してこのクライアント部分とサーバ部分をどのように構成するかを中心に発展してきた．その主な構成法が，２層構造，３層構造，そしてネットワークの特徴を反映した新３層構造である．

従来の構築アーキテクチャは，汎用的なシステム基盤を与えることが重要であり，その効果は大きく，今日のコンピュータシステムの発展に大きく寄与している．本章では，その中の極めて大きな存在価値をもつ，OSF/DCE と OMG/CORBA を概観した．従来のシステム構築手法では，設計時に存在する業務や制御対象に要求される機能やサービスを実現するものであり，これらの機能やサービスが変化する（本研究では，進化という）ことは，ほとんど考慮されてなかった．これらは業務システムや制御システムを構築するための基盤環境であり，アプリケーション・システムそのものの構築手法ではないことを指摘した．即ち，対象とする業務や制御が，時間の経過と供に，変化して進化していることに注目しており，それに情報処理システムが柔軟に対応できていないことを強調した.

ネットワークの一層の積極的な利用は，極めて広範囲のそれも自らの手で製作した物以外のソフトウェアを資源として用いる，などネットワーク上の利用できる資源を有効に使うことである.このような新しい形態のサービスを提供する状況を想定した，あるべきシステム構築手法を実現するアーキテクチャを開発した．このアーキテクチャは，後でも述べるように，ソフトウェア開発の方法論とも旨く整合する．このために，開発から実装までを一貫した思想で構築できる特徴を備えている.

３章　　構造化情報の表示と操作

3.1 緒言

オペレーションインタフェース部は，ユーザとシステムが接する唯一の部分で，システムの内部状態と行動モデルを表す表示情報と，ユーザの指示あるいは作業の形態を入力する入力情報を，それぞれに扱う部分である．

本章では，２章で示した構築アーキテクチャのオペレーションインタフェース部における情報表示と操作の位置付けをおこない，この為に必要な情報の構造化の概念を定義する．次に，情報処理システムにおける情報の表示と操作の方法を提案し検討する，更に，生産活動における成果物としての情報の意義について論する．

主な課題は以下である．
(1)情報を可視化する手段と情報の意味を統合的に保管すること
(2)情報の表示形態と情報の管理構造を独立に管理すること
(3)情報要素と関連する処理ソフトウェアを一貫性を保って管理すること
(4)情報要素はインポートあるいはエキスポートが外部と自由にできること

これらの課題の実現に向けて，情報の管理に関しては，文書処理のレイアウト構造と論理構造の概念を用いて実現する．また，情報と処理に関しては，4章で述べる予め与えられたシナリオを基に，処理の文脈の概念を用いてユーザからの要求を要素とした文脈のシンタクスを管理し，条件の成立した部分から処理を行う方式を採用し，これにより，業務の流れを制御する．

オペレーションインタフェース部において，唯一ユーザが可視化された情報を見たり操作する．即ち，システムの状況や内部状態，格納あるいは生成されたデータ，あるいはユーザが入力した情報や操作の状態を表す．ここでは，文字，図，データなどの情報から，操作に必要なスイッチ，ボタン，あるいは計器などのシンボル類も含めて統合して情報として扱う．

このように統合するためには，情報を構造化して管理し制御することが重要となる．このために，文書処理の構造に則して実現する．文書処理モデルを定義し，領域・フレーム・ブロックと呼ぶ空間の概念を導入して，文書の論理構造とレイアウト構造の連携を構成した[Sakashita83]．この考えは，ISO/ODA（開放型文書アーキテクチャ）のモデルに反映された[ODA86]．文書の概念を整理する上で，論理構造あるいはレイアウ

ト構造が一般的な意味でのスケルトンを表す概念として利用されてきた．現在，構造を定義する手法としては，このスケルトンを記述する SGML[SGML]等のマークアップ言語が多く存在している．また，同じ概念に基づく XML(Extended Markup Language)が情報構築に多く採用されている．

生産活動や社会・教育などのあらゆる環境の中で，それらを担う処理機能として文書情報が利用されるには，処理可能な形態(processable)にしておくことが必要条件となる．この故に，WYSIWYG(What You Get Is What You See)に代表される文書の形態となった状態で編集処理ができることや，見た目の形や大きさあるいは美しさを要求する以上に，他のアプリケーションから利用できる条件を整えることが重要な要素になってきた[Kinukawa86]．

ここでは，情報をシステムからユーザへ提供する視点を情報ビュー，そして反対にユーザからシステムへ情報を提供する視点を操作ビューとしている．システムが管理・制御するために構造化された情報から，現在のシステム全体の状況から，処理の焦点となる部分が抽出されて，ユーザへ提示される．ユーザは，スクリーン等に表示された操作要素の形態や関連するデータなどから，システムの状態を把握している．この表現形態に対して，操作を行うことにより，システムへの指示や情報の入力を行う．

生産活動の中では，ある単位の活動が実施されるとその成果物である情報が生成される．そこには，具象的な物自身を表すデータなどの情報も含む．身近な例では，手紙や報告書など文書である．これらは，ある活動の成果である．一般に，生産活動においては，これらの情報が活動単位の間で共用されたり，活動の元の情報であったりする．文書構造に対して，コンテンツ（文書内容ともいう）に相当する文書データの中に外部データを取入れたり提供する，いわゆるインポートとエクスポートを可能とするために，共通文書データ形式 CDFF(Common Document File Format)を導入している．従来は，CAD/CAE の分野で多く採用されたように，活動単位に相当する利用アプリケーションとの間で互いに意味構造が伝わるように対の関係で文書データ変換機構を導入していた．ここでは，文書処理を利用するアプリケーションが利用する標準環境として共通的なデータ構造を定義したことにより，文書処理以外のアプリケーションである CAD/CALS に代表されるエンジニアリングアプリケーションが文書データを取入れることが可能となった[Cowan95]．

これらの成果物は，一般にはデータベース化されて管理される．情報処理の分野での極めて基本的な要素であるデータベースは，扱う世界の情報のモデルが存在し，そ

れに従ったデータベースへ格納する構造の形態となるスケルトンあるいはスキーマが存在している．このスケルトンの概念に相当するものが前述の文書構造である[Nomiyama85][Sato87][Fujisawa86].

3.2 情報構造とオペレーションインタフェース

3.2.1 業務におけるオペレーションインタフェース

今日，ネットワークの発達による利用環境の変化により，SGML,HTML,XML,等により情報を構成して保持することにより,World-Wide-Web(WWW)環境から情報を自由に読み出せる[Chamberlin81]．個別のアプリケーションを超えて汎用的に情報を操作する情報処理の操作ビューが要求される[Chen76]． また，画像や音声も含むマルチメディア情報を含むあらゆるデータがネットワーク上に分散された資源としてオブジェクト化される[Price93][Koegel93]．

これらのさまざまな情報を，業務や仕事の処理や操作目的に沿った形態で情報を読み出して操作する環境として，このオペレーションインタフェースを位置付ける．情報を見るということは，ある処理に入力するデータ，処理した結果を見るデータ，あるは処理の途中あるいは結果の状態のデータ，などさまざまな形態でのデータを表示したり印刷して確認する．

図 3-1 は，一般的なビジネス処理系のシステムにおける帳票画面の例である．塗りつぶしてある矩形は，入力フィールドと一般に呼ばれるもので，オペレータによるデータを入力を行う領域である.他のフィールドは,表示フィールドと呼ばれる領域で,システム側から提示する情報が表示され，状態や指示の情報に相当する．また，入力フィールドは，入力に対するエコーバック表示を兼ねることが多い．

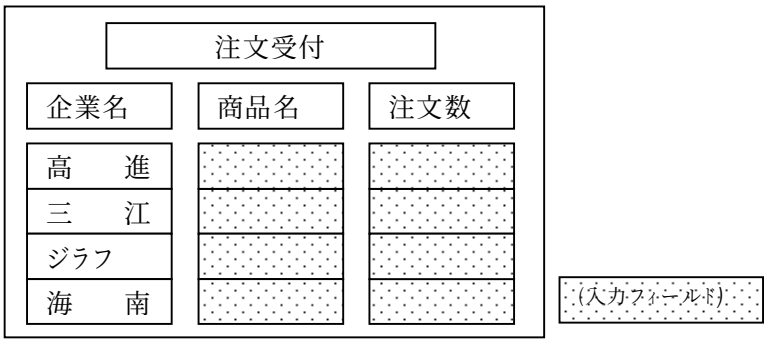

図 3-1. 帳票画面の例

図 3-2 は, 制御システム系における操作画面の例である. システムの状態を示すグラフ, メータ, あるいはスライダーの類や, 操作用のボタンやバルブなどを示すシンボル類や, オン･オフなどの操作用のスイッチなどが表示される. ここでも, システムの状態などを表示する要素と,

ユーザによる操作を受付ける要素とが存在する.

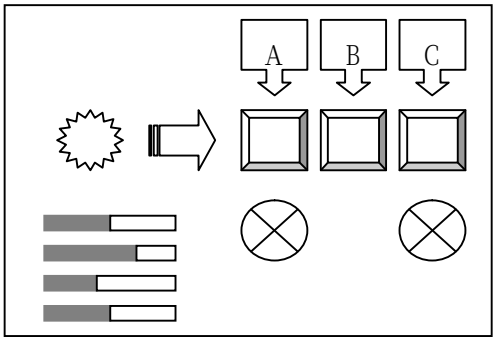

図 3-2. 操作画面の例

3.2.2 構成要素の表示方式

これらの構成要素を表示するための基本機構を図 3-3 にて示す. スクリーン等の表示媒体に対して, ＸＹ座標系を決め, 対象とする構成要素（図では, Field）の位置(Xp,Yp)とその大きさ(lx, ly)を定義する. 一般には, これらの構成要素は複数かつ階層的に構成されるので, 複合化した構成要素の概念を導入して, 同じようにその位置と大きさを定義して管理する.

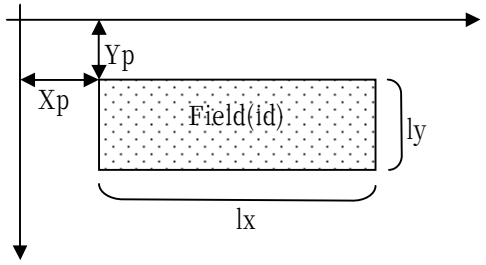

図 3-3. 表示と管理の方式

3.2.3 オペレーションと処理

帳票画面や操作画面の表示形態やレイアウトは, システム毎に違う. このために, 関連する処理プログラム側では, なるべくその影響を小さくすることが要件となっている. 本研究では, ユーザインタフェース管理システム[Miyazaki89]の経験に基づき, この間を論理的なトークンでコミュニケーションを行う方式を採用した.

その基本的な概念を図 3-4 に示す. 即ち, オペレーションインタフェース部では, 構成要素の形態的な定義情報をデータベース化して管理する. 構成要素は, 全てにその識別子(Id)を附して, 管理される. 外部からは, この識別子と操作のコマンドと関連するデータにより, 操作

される. 即ち, システムは協調型システム基盤部からこのデータセットにより, オペレーションインタフェース部へ要求を出したり, あるいはデータを受け取る.

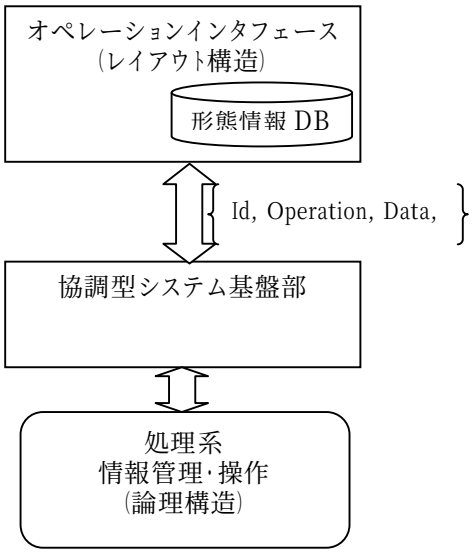

図 3-5. 情報構成要素と処理の関係

3.3 情報の構造化

情報を表すの代表的な文書を基にしてその基本的な概念を述べる．文書は，表現されている情報（内容）に関して，その意味を効率良く伝えることが，大きな使命の一つである．そのためには，文書の中で，注目する情報が置かれている位置や，隣接する他の情報との関係によって，その主旨を明確にすることが重要な要素となる．これまでは，表示あるいは印刷された状態（formatted form と呼ぶ）が，視覚に訴える要素が大きいので，編集機能や表示機能はこの面に焦点が当てられてきた．これは情報が紙に相当する２次元平面に展開された形態の,レイアウト中心のアプローチである. ワードプロセッサはここに焦点を当てて発展してきた.

他方，出版などの領域では，文章の意味的な流れに焦点を当てたアプローチも存在する．この場合には，文章の論理的な構造を持たせることにより，その意味的な構造を表すことを可能としている．このために通常，章・節・項のよる構成を用いるが，この方法が意味的な構造を表す手段の一つとして考えられる．意味の流れに焦点が当てられており，１次元的な形態といわれる.

troff[Ossana76]，TeX[Knuth79]，等の処理系は，章・節・項あるいは段落等の構造の枠を文章の中に，タグによりマーキングした論理構造表現を採用して，作業のフレームワークとして利用している．これは，意味構造を形態的な手段で行うものと言

える.

3.3.1 これまでの研究

文書が構造化されることにより，入れ物と内容とを独立に扱うことが可能となるために，その構成要素となる文書内容要素を単位として，文書操作に対するさまざまな作業形態を採ることが可能となる[Kimura84].

(1) 情報の操作

入れ物となる文書構造と内容となる文書内容とを独立に扱うことが可能とすることにより,入れ物に対する編集操作と内容に対する編集操作を区別して,処理できる.

ここで，ユーザインタフェースの観点から文書を見る．入れ物に相当する構造を，利用するアプリケーションに最適化した構成にすることにより，この構造化された文書を介して，アプリケーションを操作することが可能となる．アプリケーションのユーザは，そのアプリケーションが扱う情報をこの文書によって表すユーザインタフェースを介して，対話をすることになる．このようにして，構造化文書は，アプリケーションインタフェースの重要な手段となる.

(2) 情報の共有化

入れ物の概念を基にして，コンピュータシステムの上にアプリケーションに跨る情報ビューを備える．これにより，本研究では，グループウェアの方式になぞらえて複数ユーザが複数の文書内容を作成･編集することを可能とすることを目標にしている.例えば，文書内容に対する編集操作の視点から，筆者や編集者などに相当する複数ユーザが，共有している目的に向かって，それぞれの意見や主張を出し合いながら，共同作業を通して文書内容を作成・編集する[Brothers90][Burgress90].

この機構は，例えば，ソフトウェアの仕様書を作成する CASE(Computer Aided Software Environments) 環境に適用するする試みが多くなされ[Horikawa90][Yamashita90]. 仕様書は，その文書の同じ対象部分が，複数の異なる側面から定義される場合が多い．このため，それぞれの定義の間での完結性や無矛盾性のチェックが大きな意味を持つ．文書構造をベースとして，構成要素間での関連を持たせておくことにより，お互いの関係を管理制御して，保守することができる[Kimura84].

(3) 情報の利用

文書構造を基にして，構成要素間での関連を持たせる機構を更に発展させると，アプリケーションとアプリケーションの間で，互いに利用する情報を仲立ちとして，両

者を結合する手段となる．これは，従来とは異なる新しい形態のアプリケーションシステムの構築が可能となることを示唆している．

機能や処理のエンティティを核として，種々のサービスを実現するために，この文書情報を媒体としてこれらのエンティティが柔軟に有機的に結びつくことにより，より汎用的なサービス体系を構成することが可能となる．ネットワーク環境の上にこのようなサービス体系を構築する形態の新しい分散型アプリケーションシステムが生まれる．

3.3.2 構造化のアーキテクチャ

文書は，データを表現形態に基づき表現された情報媒体とみなす．ここで，データは文書内容であり，表現形態は，様式に基づく様式に従って表現される形である．文書内容は，文章，図，表，絵，図面，あるいは音や映像などである．様式は，手紙，報告書，論文，などの文書の型を示し，書式は，章・節・項・段落などの見出しを含む，主に文章の文字の大きさ，フォント，文字間隔，行間，インデント，などを規定する．ここでは，文書を作成・編集する場合の処理の観点から，文書処理のアーキテクチャを次のように定義する．

(1) 文書内容作成・編集部

文書様式に沿って，論理構造を定義する．

論理構造の内容部分の各種文書内容に相当するものを，作成あるいは編集する．

作成あるいは編集の手段やツールは規定しないが，内部のデータ型のみが定義される．このデータ型が次の書式作成・編集部へと渡す．

(2) 書式作成・編集部

書式を定義する．文書内容作成・編集部での出力を入力として，定義した書式に沿って文書内容を展開する．この書式化された文書内容をレイアウト作成・編集部へ渡す．

(3) レイアウト作成・編集部

レイアウトを定義する．書式化された文書内容を入力として，レイアウト定義に沿って２次元空間に配置する．２次元空間に配置された文書情報を，表示あるいは印刷の処理系へと渡す．この段階で，目に見える文書の形態になる．

図 3-6 に示す本アーキテクチャの特徴は，第一に，文書処理の基本的な処理要素を分離独立させていることにある．このことにより，文書処理以外のさまざまな処理系

と連携させるインターフェースを構成できる．従来の処理系は，これら各作成・編集部の一部を備えるのみ，あるいは一体化した形態のため，他の処理系とのインタフェースを構成することができなかった．第二に，各作成・編集部の出力形式は基本的にはオープンでなくとも，他の手段で作成したデータを入力として扱えることであり，論理構造とレイアウト構造の定義と，文書内容の作成・編集とは別手段としたことである．

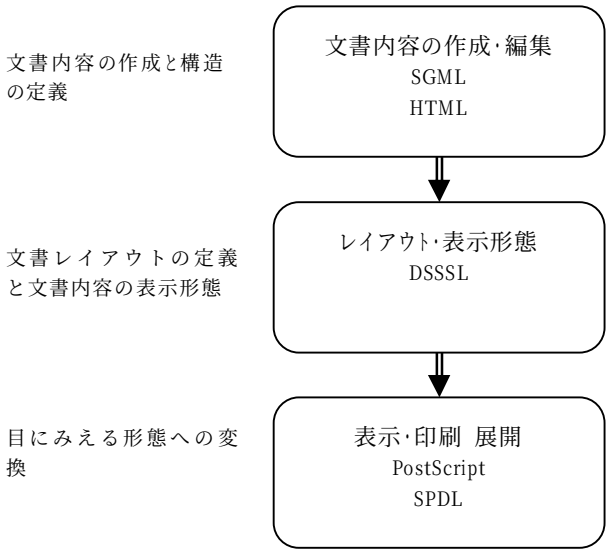

図 3-6．文書処理アーキテクチャ

3.3.3 文書における構造化アーキテクチャ

ワードプロセッサの多くが，WYSIWYG 形式が編集段階で直感的なユーザインタフェースを備えているという理由で，論理構造の処理とレイアウト構造の処理が分化されずに混在していた[Thacker79]． ここでは，従来の一体型の特徴は残しながら，両方の構造を分離して管理・制御する仕組みを導入した[Sakashita83][Sakashita84]．

(1) 自由多角形による文書二次元空間の管理

文書の論理構造とレイアウト構造を定義して，文書内容を介して両構造の間の関係を保つ方式を導入した．ここでは，論理構造は多くのシステムのように木構造により表す．レイアウト構造は，２次元平面に，エリア・フレーム・ブロックと呼ぶ自由閉矩形を，図 3-7 に示すように配置する．エリアは論理構造における文章部分，図・表部分，あるいはイメージ部分のように表現形態の形態の種類別に文書が表される空間

を示す. フレームは, エリアの内部に存在する文書内容が存在する 2 次元平面空間で, 論理構造の内容の一部あるいは全部が格納される．ブロックはフレームの内部に存在する文書内容を表す各種のメディア，即ち文字列・図・罫線・イメージなどのメディア単位が格納される 2 次元空間の集合である．

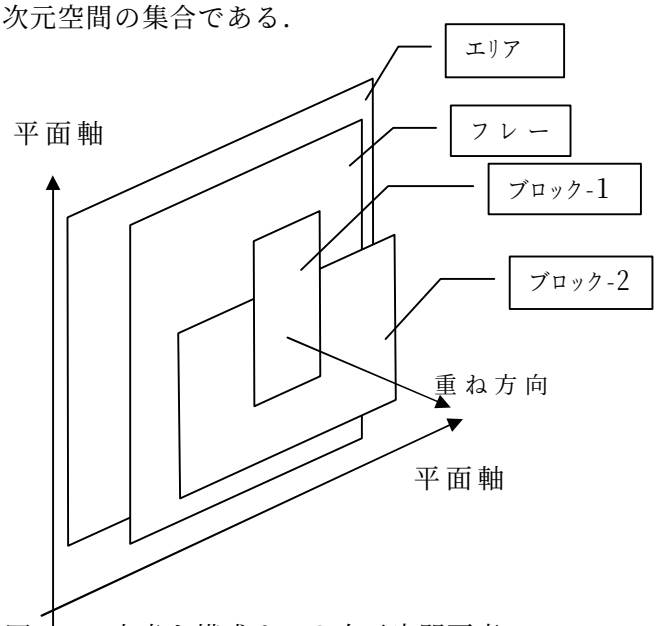

図 3-7．文書を構成する 2 次元空間要素

ここで，エリアはフレームを構成要素として持つ，但し，フレームは物理的には互いに重ならなずに 2 次元平面に排他的に配置される．文章の意味的な流れを規定するために，フレームとフレームはその結合関係を定義する．フレームは，ブロックを構成要素として持つ，但し，ブロックは物理的に互いに重なること，及び内部にブロックを階層的に構成することができる．ブロックには唯一のメディア単位のみを格納することができる．例えば，図 3-8 に示すように，文章を格納するブロック，イメージを格納するブロック，あるいは図形のみを格納するブロックである．これらの複数のブロックを互いに位置合わせして，統合した文書内容を構成する．

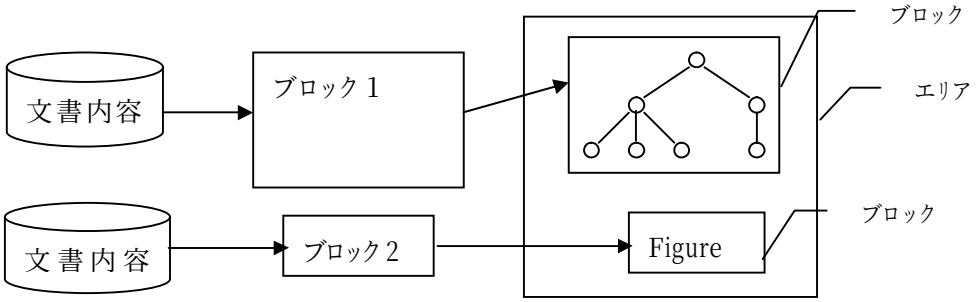

図 3-8．エリアとブロックと文書内容の関係

(2) 論理構造

文書の構成要素である論理オブジェクトの階層関係で表現した構造を，論理構造と定義する．図 3-9 に示すように，論理構造は木構造で表現され，各ノードが論理オブジェクトに相当する．論理オブジェクトは次の３種類に分類される．第一は，木構造の根の位置に存在する「論理オブジェクトルート」である．第二は，木構造の非終端のノードで表現される「複合論理オブジェクト」である．複合論理オブジェクトとしては，章・節・項・図・表・等を対象とした．第三は，木構造の終端のノードで表現される基本論理オブジェクトである．ここでは，文字列・図・イメージ・表など文書を構成する基本構成要素として，基本情報形態と呼んでいる．基本論理オブジェクトには，唯一の基本情報形態が対応する．

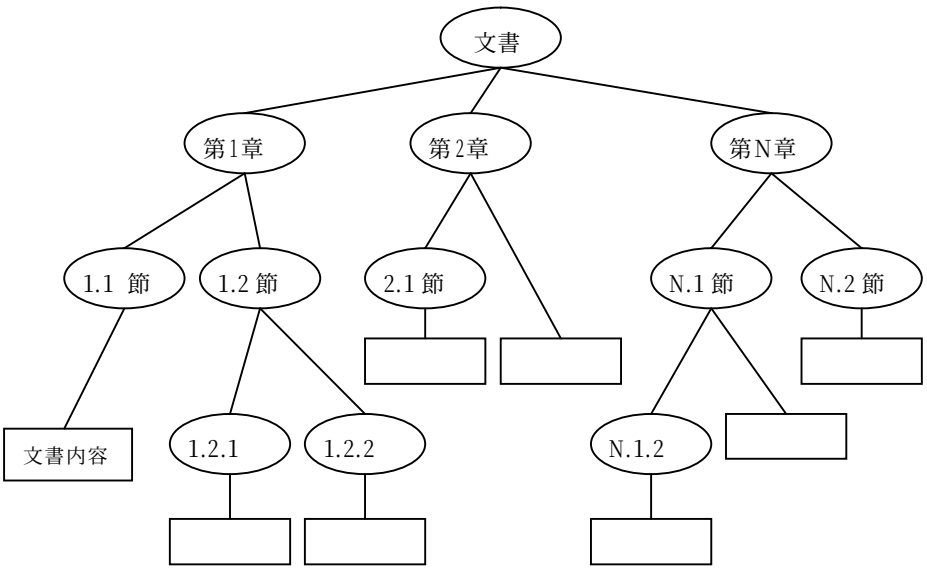

図 3-9．論理構造

(3) レイアウト構造

論理構造と同様にして，図 3-7 に示したように物理的な情報単位の配置を，図 3-10 に示すような木構造により管理する．即ち，文書は，ページ，エリヤ，フレーム，そしてブロックから構成される．

論理構造とレイアウト構造は，さまざまな編集や制御を行っても常に一貫性を保たなければならない，という新たな課題が生ずる．本方式では，図 3-11 に示すように，両方の構造から共有する文書内容に対して，ポインターなどの手段によりそれぞれの構成要素単位の域を管理している．

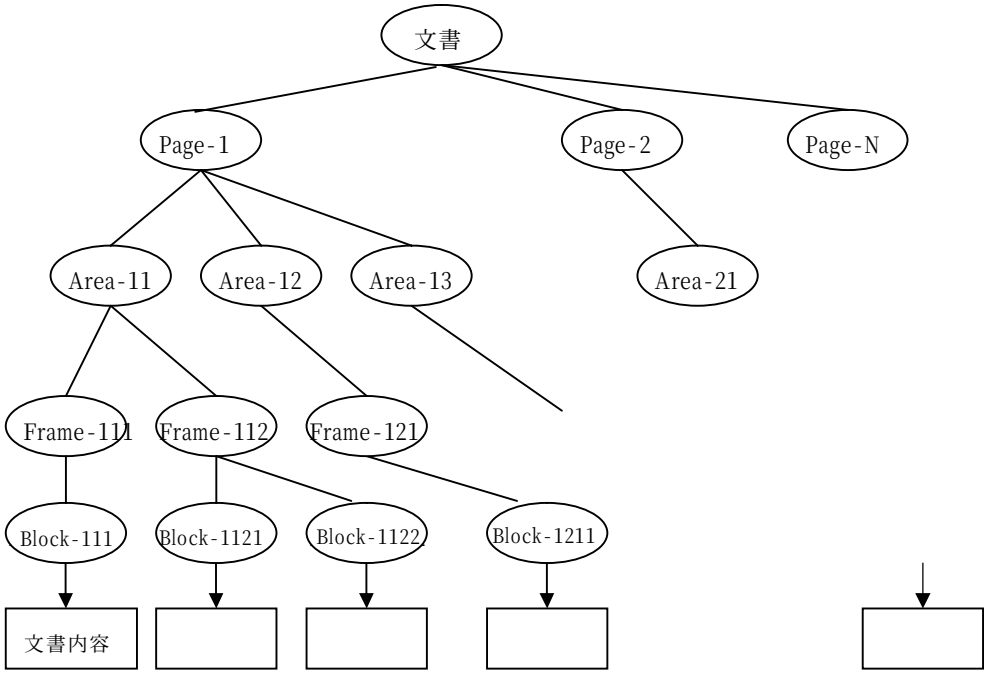

図３－11．レイアウト構造

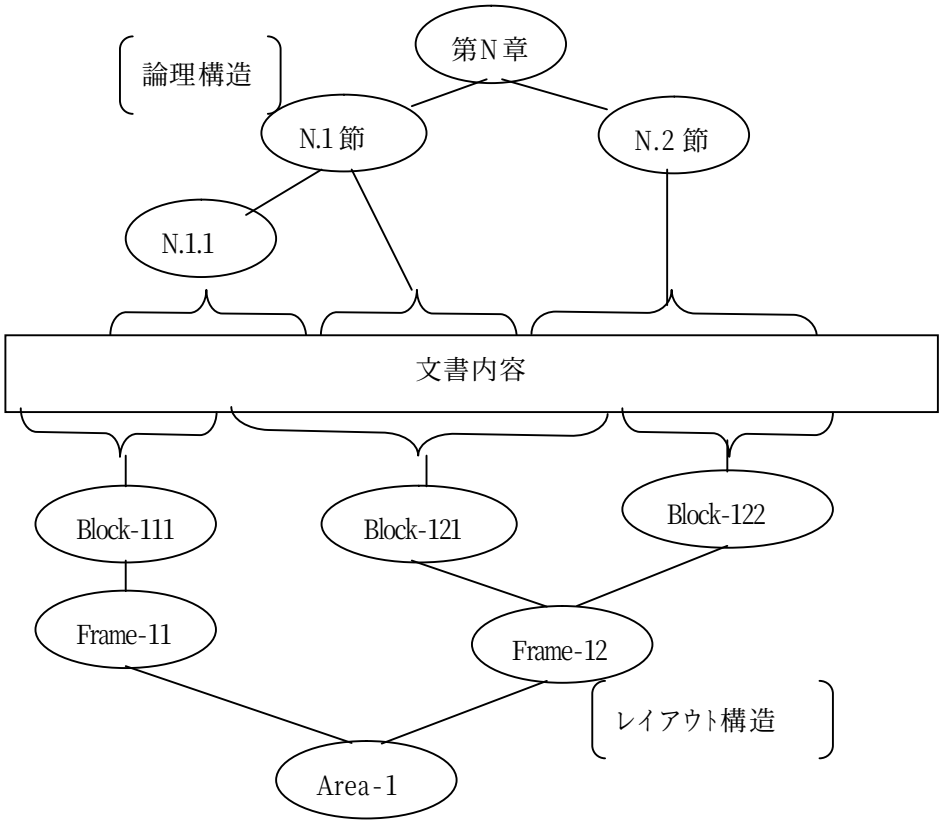

図 3-12．論理構造とレイアウト構造と文書内容を介した関係

3.4 表示と操作

オペレーションインタフェース部における情報の表示とそれに対する操作に対応して，構造化された情報との関係を論ずる．

一般には，ユーザからの操作をシステムはウィンドウシステムにおいて受けて，表示制御を行う部分において処理プログラムに渡す形式に変換し，対話制御において入力された情報がどの業務処理に関連し，処理プログラムが必要とするどの情報かを判断してから，実行処理プログラムへ渡されて起動される．この流れの関係を，図 3-13 に示す．

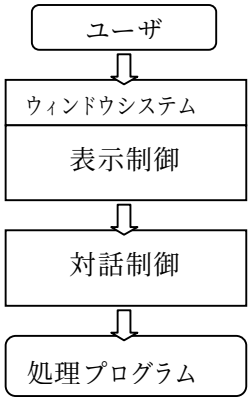

図 3-13．操作と処理

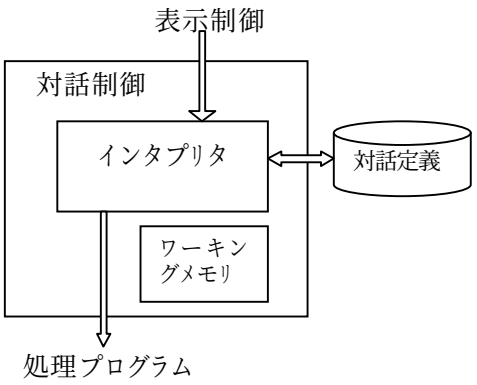

図 3-14．対話制御方式

ここで述べた「対話制御」は，図 3-14 に示すように，ワーキングメモリ，対話定義部，そしてインタプリタからなる．ワーキングメモリは，対話を進めるに当たって，その時々で認識しておくべき状態情報を保持する．即ち，アプリケーションの状態，ユーザからの入力，処理プログラムからの返値，そしてシステムの状態情報である．

対話定義部は，対話の流れを定義するルールの集合である．具体的には，ルール名，ルール選択の優先度，条件，そして実行部からなる．条件部の成立は，ワーキングメモリの内容との照合によって決まる．

インタプリタは，一般にはワーキングメモリの内容と各ルールの条件部との照合を行い，ある選択基準により1つを選択して実行する．本研究では，アプリケーションの状態に代わり次章で論ずる役割によりルールの選択範囲を絞って，優先度の高い順に照合し，最初に合致したものから実施している．

この処理を行うために，協調型システム基盤部で対応する機能を役割に基づき定め，そしてサービスプラグイン機構部で，実行するサービス機能が選択されて実行される．この実行結果を受けて，逆に，オペレーションインタフェース部に対して処理結果を伝えて表示される．

情報の表示は，論理構造により管理されている情報がアプリケーションにより制御されて，オペレーションインタフェース部における表示制御へ渡される．レイアウト構造に基づき管理されている表現形態の情報を参照しながら，表示制御機能が表示媒体に表示する．

以上述べたように，表示に関する処理はアプリケーションとの間は論理的なインタフェースにより通信が行われて，表示の様々な形態は表示制御機能部分に隠蔽されて処理される．

3.5 生産活動の成果物

生産現場においては，開発・設計・調達から保守・運用に至るあらゆる局面において，関連する全ての部門が円滑に且つインタラクティブに情報のやり取りを行い，情報を共有し活用することにより，開発・調達の期間の短縮，生産性の向上，製品のライフサイクルを通したコストの低減を図るための，新しい産業情報基盤環境を目指している．この考え方は，Product Data Management :PDM と呼ばれる[Krause93]．

3.5.1 生産プロセス管理

企業活動における生産活動の各段階で，アクティビティと呼ぶ活動単位に関する成果物あるいは操作対象がデータとして生成されて管理される．他の段階のアクティビティにおいて，この成果物データが参照されたり加工・編集される．具体的には，製品の営業，設計，生産，販売業務に関する電子化された技術，品質，資材などの各種情報の共有と流通を管理及び制御する．全社あるいは工場規模での生産活動の標準化を推進し，これらの情報を共有し流通することにより，各生産拠点での製品の品質管理，原価管理，工程管理を図り，生産性の向上を目指す．この概念を図 3-15 に示す．

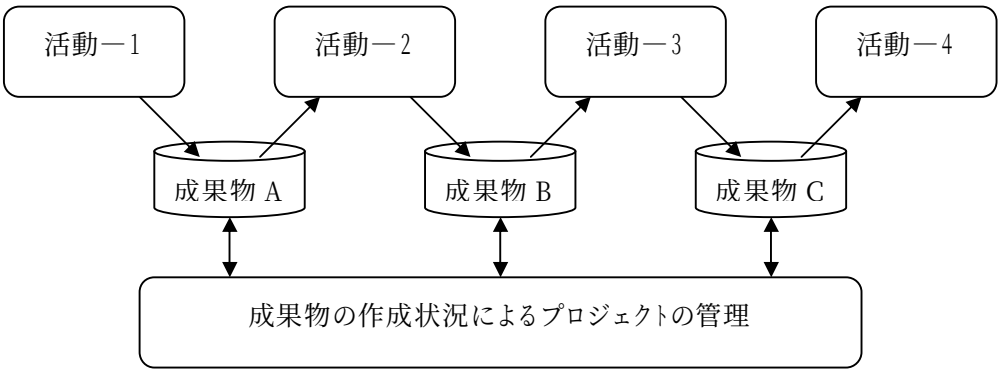

図 3-15. 生産プロセス管理の概念

この背景に在るのは，CAD/CAM の領域で開発されてきた手法で，製品の全ライフサイクルを通して，製品に関する全ての情報をモデル化したプロダクトモデルである．ここでは，出来上がった製品情報を記述することよりも，簡単であいまいな製品に対する要求から始まり，詳細な製品記述情報を生成していく過程を支援することが強く求められている．この製品情報を表す手段として，製品を構成する要素を，文書構造で述べた構成要素として捉えて階層的に管理する．

この情報処理システムの連携の代表的な例を次に挙げる．一つは，電子商取引 EC: Electric Commerce である. 将来の産業情報化社会における将来像の具体的なイメージとして挙げられる． Electric Data Interchange: EDI を拡張した形態であり, アメリカと欧州を中心にして広がっている. この定義には諸説があるが, ビジネス上の全てのプロセスの情報交換をオープンなネットワーク上で電子化して行う, と理解されている. 市場調査, 宣伝, といったマーケッティングから始まり, 最終的に請求および支払いといった金融決済に至るビジネスプロセスにおいて存在するさまざまな関連の主体の情報システムをネットワークで連携させて, 取引のプロセスを全て情報ネットワーク上で電子的に行う.

他の例は, Continues Acquisition and Lifecycle Support: CALS である. アメリカの国防総省 DOD が兵站支援のために考案した情報システムの概念 Computer Aided Logistics Support が元である. 国防産業から一般産業へ浸透した. 同時に, 概念も Logistics に基づく後方支援から, 製品の調達, ライフサイクルの支援へと広がった.

現在では, 概ね次のように定義されている.

(1) 部門間あるいは企業間で, 設計図等の技術情報や受発注等の取引情報を, 特定の機器, システムの制約を受けずに, 情報をデジタル化したままやり取りができる, ユーザ本位のデータ環境を形成する.

(2) 開発･設計･調達から保守･運用に至るあらゆる局面において, 関連する全ての部門･企業といった主体が円滑に且つインタラクティブに情報のやり取りを行い, 情報を共有し活用することにより, 開発･調達の期間の短縮, 生産性の向上, 製品のライフサイクルを通したコストの低減を図り, あたかも一つの企業（バーチャル企業）のように連携するための, 新しい産業情報インフラである.

3.5.2 業務と情報構造

生産活動の各段階で，活動単位に関する成果物あるいは操作対象がデータとして生成されて管理されるには，構造化された情報体系が必須となる．これにより，各活動段階で必要とする情報源を明示的に把握して，その処理に施すために必要な情報を取

り込むことが可能となる．同様に，処理結果である成果物を対象とする情報体系の部分へ配置することが可能となる．図 3-16 は，複数の活動単位に関連するアプリケーションの間で,成果物である情報をデータベースから取出したり,格納する様子を示す.

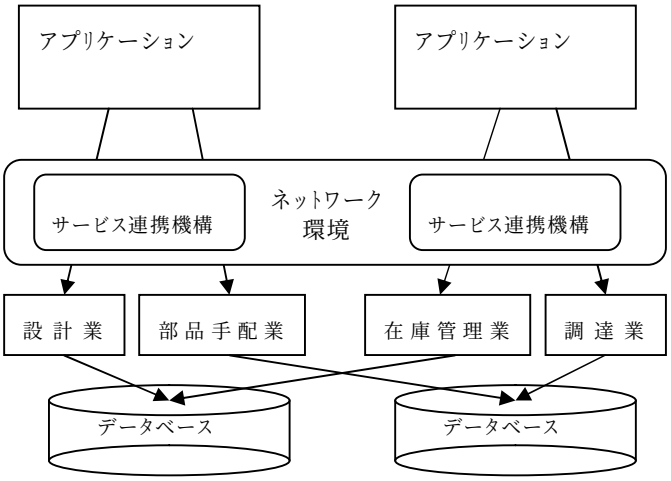

図 3-16．成果物の相互利用

一般には，各アプリケーションが生成する情報データの型はそれぞれに定義されていて，標準化されていない．この為に，アプリケーションの間で自由に，相互利用することは一般には難しい．このような成果物の相互利用の際には，適用業務の領域で用いるデータ形式が CAD/CAM の領域での STEP(Standard for the Exchange of Product model data)のように定まっていることが望ましい．しかし，実際にはこのように共通的に使用されるデータ形式は少ないので，そのシステムあるいは企業活動の組織の中で共通的なデータ構造を設けて，制御することになる．例えば，CDFF(Common Document File Format)のように，業界のコンソーシアムで決められたものがある.
この課題は，改めて４章で論ずる.

3.6 結言

分散型アプリケーションシステム構築アーキテクチャにおけるオペレーションインタフェース部を構成するための基本要件を挙げ，その実現のためのアプローチと実施内容を述べた．

情報を自由に且つ一貫性を保って操作するには，情報の構造化は必須の条件である．その構成要素である情報単位を可視化させる手段，そして逆に表現した情報をユーザが操作する手段を，表示制御機能が司る．ここでは，情報の表現形態と情報の管理構造を独立に管理することが重要な課題であった．特に，ユーザ毎，あるいは業務毎に，ことごとく異なる表現形態の定義と関連する処理とを分離するために，徹底的に論理的なインタフェースを用いて定義した．このことにより，関連する処理との関係は極めて統一化された連携インタフェースが構成され，情報を可視化する手段と情報の意味を統合的に扱うことができた．

ユーザからの処理要求に関しては，入力されたコマンド列を処理の文脈の概念を用いてユーザからの要求を要素とした文脈のシンタクスを管理し，条件の成立した部分から処理を行う方式を採用し，操作の流れを制御した．これは，ユーザインタフェース管理システムの基本的な枠組みに従うものである．この部分の処理は，次の章でのシナリオ及び振舞いに強く関連する．

また，情報の構造化には，文書に適用した論理構造とレイアウト構造の概念を，情報一般に拡大して実施した．このことは，生産活動における成果物である情報を操作し管理するために，その情報要素を外部と自由にインポートあるいはエキスポートを可能にした．特に，文書の構造化に関しては，マルチメディアを含むあらゆる形態の文書内容を対象として，文書の論理構造とレイアウト構造を対の関係で備える文書処理系のアーキテクチャを提案した．レイアウトの構成要素を矩形を基本とする２次元空間を単位として階層的に構成し，各種のメディアを直接的に操作できる構造とした．また，情報の意味の単位を要素としてその意味構成を文書のスケルトンとして表す木構造による文書の構造とした．この２つの構造は文書内容を挟んで互いの関係を保存する構成とした．

本アーキテクチャは，機能標準規格である ISO の機関で検討の材料として扱われ，検討と審議の結果，"Open distributed Document processing Architecture"として規格化された．その後，国内にて JIS 規格「開放型文書処理体系」として成立した．

文書情報の構造が全てのアプリケーション・システムが同じものを備える保証はない．アプリケーション・システム間で，情報のやり取りや，サービスのやり取りが，行われるためには，業務や制御領域で共通的な共通データ構造を用意して，この共通情報を媒体として情報の送受を行うことの，有効性を論じた．

ネットワークの発展により，アプリケーション・システム間での情報交換が盛んに行われる．これは，適用対象の業務や制御の領域の境界を越えて，ＰＤＭに代表される新しい企業の生産活動を促していることを述べた．重要なことは，それぞれが生成した情報を他の部門や産業が取り入れることにより，業務活動の枠が限りなく広がって，更には新しい形態のサービスを生んでいることである．このように，分散型アプリケーションシステムは,新しい形態のサービスを順次生み出していくと予想される．

第４章　　動的サービス結合方式

4.1 緒言

時間と共に変化あるいは進化する業務を構成するビジネスシステムを，より柔軟にかつ一貫性をもって構成することがシステムの構築上で重要な要素となってきている．

ネットワーク発展に伴い情報システムでは，その構成要素は固定的に限られた関係でのみシステムに組込まれるのでは無く，自由に自律したモジュールとして着脱が可能となっている．また同じ構成要素であっても対応するシステムに依って部分的に機能やサービスの内容が異なる．

このような変化に対応するために，ユーザ向け言語，あるいは作成ツールをベンダー側は用意して，業務に必要な業務サービスを実現させる手段を提供してきた．しかし，その大部分は新たに作るという作業になり，市販されている，あるいは世の中で多くのユーザに利用されているアプリケーションプログラムを取入れて構成する，ということは極めて難しい．

3層構造の分解：情報システムの構築手法には，ネットワーク出現により，ネットワークを介したユーザ側の情報処理システムをクライアント，そして業務の本体部分の機能あるいはサービスを実行してその結果を提供するサーバとに，役割分担した形態で構成することが大勢となった．しかし，従来の汎用機に代表される大規模な業務システムを担うことは，性能や規模の要因でその構築は困難であった．これに応えるべく，機能的に，ユーザ操作を主として司るブラウザー部分，業務アプリケーションの機能の実行を司る機能部分，そして業務システムの要でもあるデータベース部分，と3層からなる構成が採用され，それぞれの層の間にネットワークが絡む構成となった．

この3層構造での機能部分では，ユーザの要求の受取り，この要求に基づく処理やその結果の返答，そしてデータベース処理などのトランザクション処理が行われ，適用業務の仕組みをそのままに反映している．即ち，各種の業務の進行もこの部分において管理され運用処理されている．

情報システムの基本的の構成形態は変えずに，事業の目的や形態の変化，あるいは顧客に提供するサービスの種類や内容を柔軟に変更できる仕掛けが必要となる．このために，業務を実現するために，業務の基本的なやり方を行動形態を表す部分をシナリオとしてその振舞いを定義し，このシナリオに基づいてサービスを実現する手段としての機能に分化して，それぞれに制御する方式を，3層構造の機能部分に適用導入した．

役割の導入：分散型のビジネスシステムでは，前述のようにユーザ操作部分・ビジネス機能部分・データベース部からなる3層構造と呼ぶ構成が主となっている．ビジネス機能部分に注

目して，業務の基本的な行動の枠組みの部分と，そこで実行される機能部分に分離して制御する．

この分離する基盤として，役割の概念を導入し，シナリオと役割，そして役割とサービス機能の関係を導入して，シナリオ部分と機能部分とを論理的に関係付けることにより，実行時には連携して動作する．即ち，一般に，行動には目的が存在し，それを達成する使命あるいは役割を持って，実施する．この役割の概念を基盤として，行動としての振舞いと，実現手段としての機能あるいはサービスとを連携させる．

プラグイン方式：シナリオ・役割・機能の体系に対して，役割をソケットにしてサービス機能をプラグインする手法を導入する．これにより，役割という枠組みの中にサービス機能を自由に組込むことができる．従来から，カプセル化の手法は多く存在するが，どのような枠組みあるいは制約の基でカプセル化を行うかという汎用的な方法がなく，プログラミングの技法として「結合」させるレベルに留まっていた．本方式では，「役割」という明確な枠組みが存在しているので，利用レベルあるいはサービスレベルでの結合がなされる．

従来，ソフトウェア工学でのオブジェクト指向方法論の視点からは，業務の分析段階で役割に注目した業務の処理単位を抽出し，そこにはどのような機能が必要とされ，更に個々の処理単位の間の関係を分析し整理している．このように，業務システムにおいてもあるいは制御システムにおいても，役割を基軸に対象を整理することは良く行われており，この結果に基づいて，振舞いと役割と機能をそれぞれに構成することは，分析の段階から設計の段階に移行する時に，大きな構成概念の変化が無くなることになり，期待される効果は大である．分散情報処理システムは，ネットワーク環境の発展に伴い対象とする業務システムが部門を跨りあるいは企業を跨って連携して動作する傾向にある．そのため、システムが大規模化し，かつ企業活動に関わるあらゆる部分に対象が広がり，その処理内容も複雑化している． このようなシステムに対して、一般に要求分析や仕様の獲得作業を十分に行っても，対象システムの全ての仕様を完全には把握出来ない．更には，システム製作後に適用対象の業務形態が変化したり，対象業務のサービス内容が変化することがしばしば発生する．

このようなシステムの変更に対応するため、システム構築の段階において，機能部品を元にして組み立てて構築するコンポーネントウェア[Honiden94]の技術によりシステム構築が行われる．あるいは業務分野に依存はするが共通的な定式化が可能な場合，個別のシステム構築時に個々の機能オブジェクトを開発するフレームワークの技術によるシステム構築が始められている[Iijima94][Katoh94][Fujiwara97]．しかし，シ

ステム変更に伴う多彩で柔軟な業務サービスの提供には十分対応できていない.

業務システムを構成する情報処理システムは，機能の関連，操作，イベント，そして制約から構成される[Hill96]. 業務システムの構成要素であるサブシステムの分割に関して，業務システムは共通の目的を持ったサービス，シナリオ，役割，そして機能から構成されると我々は考えている[Maenaka97][Kanaegami97][Ohshima97]. 従来から、情報処理システムをシナリオとシナリオによって制御されるオブジェクトによってモデル化する方式にはいくつかの提案がある[VanList96][Lichter94]. 我々のモデルでは、オブジェクトが単に制御される対象にとどまらず、オブジェクトの構造やインタフェースの変更までを含めた柔軟性を持たせている点が特徴となる.

本章では, このモデルに基づき, 業務機能やアプリケーションを資源とみなしてこれらを協調的に制御するアーキテクチャを示す. そして, それを実現する制御機構を提案し, そしてこれらの機構に基づく実装例を示す. 以下, 情報処理システムにおける従来のシステムモデルの課題と, 本研究で提案する新しいシステムモデルを述べる. 提案するアーキテクチャとその実行機構を述べ, 本方式による産業システムへの実装例を説明し, 実装の評価について述べる.

4.2 情報処理システムのモデル化

4.2.1 従来のシステムモデルと課題

情報処理システムのモデル構築[Rumbaugh92][Heninger78][Robertson97][Monroe97][Fowler97]においては，典型的なシナリオを作成する．このシナリオは突発的に起こることすべてには対応することができないが，少なくとも共通の相互作用が見落とされないことを保証している．即ち，シナリオからイベントを抽出し，それぞれのイベントをその対象となるオブジェクトへ割り付けていく方法が有効とされて採用される．シナリオは，イベントに起因する作業の流れである．イベントは，システム内のオブジェクトとユーザ，センサー，他のタスクといった外部との間に情報が交換されるときに発生する．シナリオを基にするモデルでは，オブジェクトを定義し，そのオブジェクトとオブジェクトとの関連を階層的および複合的に生成して，対象とするシステムの振舞いを規定する[Fayad97][Sparks].

システムの振舞いを動的なモデルにより表すときは，システムおよびその中のオブジェクトの時間に依存した振舞いを表すことになる．この場合，一般に，特定のタイミングや同期を要求するが，相互作用の順序に注目している．

自律・分散・オープンという複雑系の特徴を備えるネットワークの環境では，システムの実行環境の変化，あるいはアプリケーションシステムそのものが変化するあるいは進化する状況が増大する．このような変化に対応するには，このシナリオとオブジェクトの連結関係が固定的に，あるいは直接的に定義されているために，前述のような変化への対応は困難である[Sakashita98]．即ち，動的なシステム再構成への対応が不十分である．

4.2.2 提案するシステムモデル

情報処理システムの基本構成を変えることなく、システム変更に柔軟にかつ動的に対応するために，我々は，図 4-１に示す制御構成を提案する．図１に示すように，情報処理システムのサービスはシナリオを規定し，シナリオは個々の役割に基づく振舞いを規定し，そして役割は個別機能により実現されると仮定する．

シナリオは、アプリケーションの目的や枠組み規定する．ここで、どのような役割がアプリケーションの中に含まれており、どのようなインタラクションを行う必要があるかを記述する．シナリオは、ビジネスアプリケーションにおけるワークフローと考えられる．役割は、業務システムの中における抽象的な機能単位を規定しており、

その機能単位を処理するために必要な条件や求められる結果を記述する．機能は、基本的な業務システムの実行単位であり、作業者によるデータの入出力やコンピュータ内部での処理の完了などのイベント処理が含まれる．

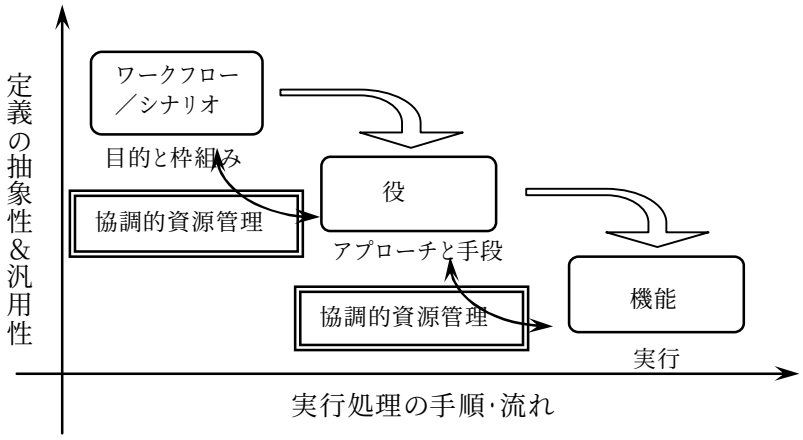

図 4-１．シナリオ・役割・機能による業務システムの制御構成の概念

図 4-１における協調的資源管理機能は、シナリオと役割、役割と機能を関連づける動作環境を示す．この協調的資源管理については、後で詳細に述べるが、シナリオと実際の実行主体とを動的に関連づける機構であり、かつ実行主体の動的な再構成の機構をも提供している．従って、従来の単なる機能単位の選択という固定的な枠組みより、更に柔軟性を増したモデルとなっている．

4.2.3 役割と機能

クライアント・サーバ型を基とする，業務システムの構築に多く用いられる従来の３層構造による情報処理システムの「業務ロジック」の部分には，業務の流れを司る制御の部分と実際の業務処理機能としての機能を実行する部分が，混在している．
本論文では, 業務の振舞いを,「役割」という事柄に注目して, 実世界の業務の物事について, オブジェクト指向技術に基づいて, オブジェクトとメッセージの関係で分析し, 再構成するものである.

図 4-2 に示すように, 業務ロジックの部分を，「役割」という概念に基づいて分解して整理した．即ち，業務の活動単位を役割を媒体として，機能と関連付ける．これにより，この２つの要素を独立に扱うことが可能となり，機能部を交換したり，追加す

ることができる.

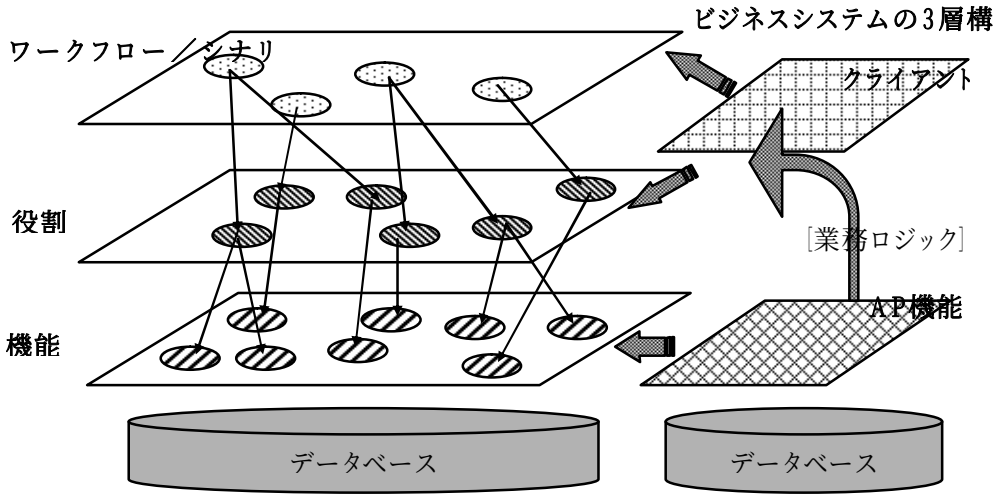

図 4-2．アプリケーション機能部の分化

4.3 アーキテクチャ

本章では、上記のシステムモデルを使って情報処理システムを構築するための実行アーキテクチャについて述べる.

4.3.1 基本機構

提案するシステムは、ワークフロー／シナリオ記述、役割記述、そして機能記述からなるシステム記述とそれらを解釈する実行系から構成される．実行系は、それぞれのシステム記述を解釈するワークフロー／シナリオ層、役割層、そして機能層に分かれており、それぞれ独立に動作する．図 4-3 に，本論で提示するアーキテクチャの基本的な枠組みを示す.

以下に各層の機能を述べる.

(1) ワークフロー／シナリオ層は，シナリオに規定された一連の業務を遂行するために必要となる業務単位の流れを管理する．シナリオに指定された役割の論理的集まりであるアクティビティを起動し、アクティビティが処理を終了すると次のシナリオに制御を移す.

(2) 役割層では，シナリオ層からの要請を受けて、アクティビティに含まれる役割を起動し、業務を実行する．このとき、役割に対応する機能や引数に関する条件を、

管理テーブルから検索して適切な機能を起動する．また、シナリオ層から新しい役割や役割の性能向上に関する起動要請があった場合、協調的資源管理機構により、基本機能の組み替えや実行マシンの追加などの動的な対応をとる．

(3) 機能層は，役割層によって起動要請があったときに、ネットワークに接続されている適切なコンピュータシステムに存在する適切なアプリケーションを実行する．

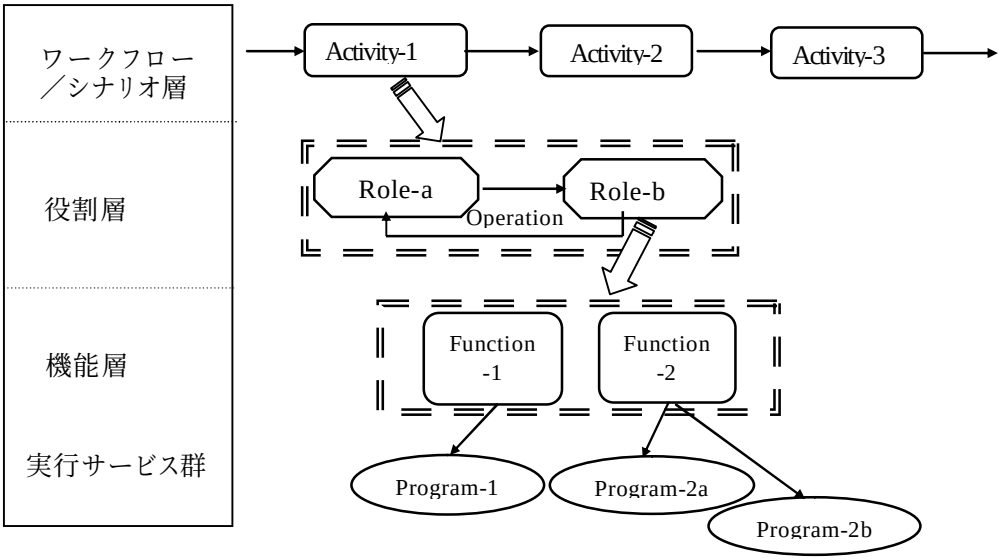

図 4-3. アーキテクチャ

4.3.2 シナリオと役割

シナリオの各作業ステップは，アクティビティと言われるように，役割を持つ行動主体が作業全体を遂行する上での副目標を構成しながら振舞う．これらを役割毎に，その内部構造や機能分担を整理して表現する．即ち，与えられている情況の中で何をすべきかがここでの焦点である．図 4-4(1)に，シナリオ部分から役割部分を使用する定義例を示す．

4.3.3 役割と機能

役割に従い，なすべき事を実現する段階である．ここでは内部構造や機能分担を整理して形式的に持たせることにより，表現された役割を実際の実行するサービス部分へ関係付けている．これにより，シナリオと機能は，直接に連携するのではなく，役

割を介して関係付けられることになり，システムの機能とサービスを分離することが可能となる．図 4-4(2)に，役割部分から機能部分を使用する定義例を示す.

4.3.4 機能とサービス群

シナリオ部分から機能部分までは，論理的あるいは形式的に定義される．機能が特定された後に，実際に実行処理を行うプログラム群が選択され起動される．図 4-4(3)は，この部分の定義例である．実際のプログラム群は，ネットワーク介して特定のコンピュータシステムに存在している同じあるいは類似の機能や版が異なるソフトウェアの集合である.これらのプログラム群は一般に資源管理システムにより管理される.この資源管理システムを介して，機能部分から特定されたサービスへの起動が行われる.

```
Scenario Definition ::=
      ( Scenario_name, Input_event_from_GUI, Output_event_to_Role) ;
Scenario_name ::= string ;
Input_event_from_GUI ::= null | Input_event | Input_event, Input_event_from_GUI ;
Output_event_to_Role ::= null | Output_event | Output_event, Output_event_to_Role ;
Input_event ::= ( Event_name, <Input_relation>, <Output_relation> ) ;
Output_event ::= (Role_name, Input_relation, Output_relation ) ;
Event_name ::= string ;
Input_relation ::= null | Attribute_name | Attribute_name, Input_relation ;
Output_relation ::= null | Attribute_name | Attribute_name, Output_relation ;
Attribute_name ::= string ;

Calling Scenario Interface from GUI ::= (Scenario_name, Input_event) ;
Scenario_name ::= string ;
Input_event ::= ( Event_name, Input_relation, Output_relation) ;
Input_relation ::= null | Attribute_name | Attribute_name, Input_relation ;
Output_relation ::= null |  Attribute_name | Attribute_name, Output_relation> ;
Attribute_name ::= string;
Event_name ::= string;
```

(1)シナリオ部分から役割部分を使用する定義例

```
Role Definition ::=
(Role_name, Operation_type, Services, Input_relation, Output_relation,
Evaluation_function) ;
Role_name ::= string ;
Operation_type ::= "CONCURENT" | "RECOVERY" ;
```

```
Services ::= null | Service_name | Service_name, Services ;
Service_name ::= string ;
Input_relation ::= null | Attribute_name | Attribute_name, Input_relation ;
Attribute_name ::= string ;
Output_relation ::= null | Attribute_name | Attribute_name, Output_relation ;
Evaluation_function ::= max( Service_property ) | min(Service_property) |
                    equal(Service_attribute, Input_evaluation_value);
Service_property ::= Service_property_name;
Service_property_name ::= Attribute_name;
Service_attribute ::= string | integer ;
Input_evaluation_value ::= string | integer ;

Calling Role Interface ::=
  (Role_name,           Input_relation_value,           Input_evaluation_value,
Output_relation_value)
Role_name ::= string ;
Input_relation_value ::= null | Input_value | Input_value, Input_relation_value ;
Input_value ::= string | integer ;
Input_evaluation_value ::= string | integer ;
Output_relation_value    ::=    null    |    Output_value    |    Output_value,
Output_relation_value ;
Output_value ::= string | integer;
```

(2) 役割部分から機能部分を使用する定義例

```
Service Definition ::=
  (Service_name,          Service_evaluation,          Service_input_relation,
Service_output_relation);
Service_name ::= string;
Service_evaluation ::= null | Evaluation | Evaluation, Service_evaluation ;
Evaluation ::= (Service_attribute ,Evaluation_value);
Service_attribute ::= (Service_attribute_name, Type);
Service_attribute_name ::= Attribute_name;
Evaluation_value ::= string | integer ;
Service_input_relation ::= null | Service_attribute | Service_attribute,
 Service_input_relation ;
Service_output_relation ::= null | Service_attribute | Service_attribute,
Service_output_relation;
Service_attribute ::=  (Attribute_name ,Type);
Attribute name ::=  string;
Type ::=  "string" | "integer";

Calling Service Interface ::=
```

```
(Service_name, Service_input_relation, Service_output_relation);
Service_name ::= string;
Service_input_relation ::= null | Attribute_value | Attribute_value,
Service_input_relation>;
Service_output_relation ::= null | Attribute_value, | Attribute_value,
Service_output_relation>;
Attribute_value ::= string | integer;
```

(3)機能部分からの起動の定義例

図 4-4. シナリオ部・役割部・機能部の定義例

4.4 役割部からサービス機能の協調的選択

4.4.1 協調的選択の機構

従来は，対象オブジェクトの指定とその利用は，一意に決定して行っている．最近の分散処理環境における資源オブジェクトの利用では，OMG の Trader 機能のように利用する側が利用するインタフェースを含む環境条件を指定して，利用する機構が存在する．本研究で提案する方式は，図 4-5 に示すように，利用する資源オブジェクトを明示的に一意に指定するのではなく，これまでに述べた役割のレベルで指定する．

これにより，アプリケーションシステム自身ではなく，情報処理システム基盤が管理するネットワーク環境を含めた状況や条件を含めて,選択を行うことが可能となる．

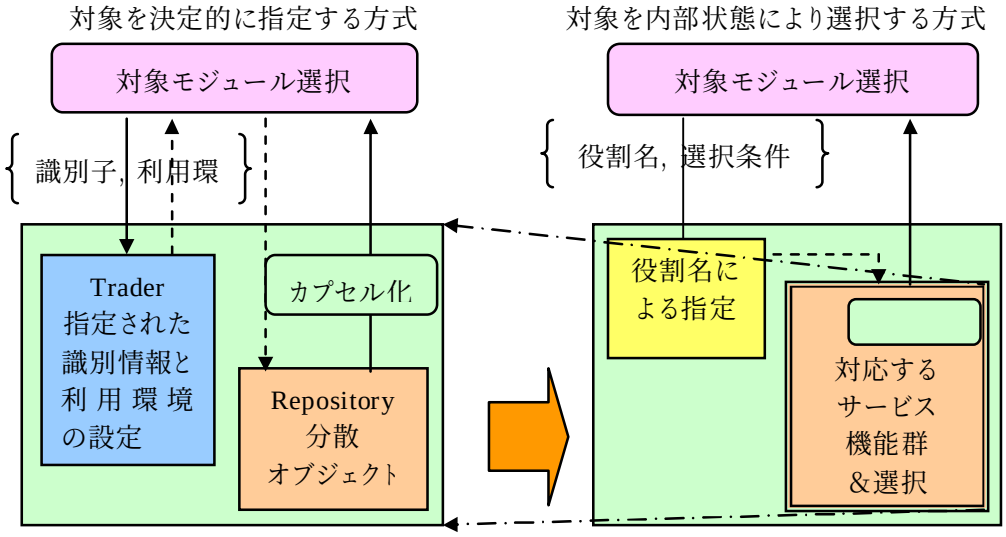

図 4-5 協調的選択の機構

4.4.2 役割部からサービス機能の選択

指定された役割が，関連するサービスを選択する枠を決める．これにより，調整の対象となる関係者・メンバーの候補が挙げられることになる．次に，この候補からの選択は，調整作業の指針となる基準を与えることにより，候補者自らではなく候補を挙げた側が行う．

具体的には，サービス機能の選択は，役割に基づき，これに対応する機能を契約ネットの手段により応募する．これに対して，求められる機能と要求に対応できるサービス機能が応札する,この場合複数の応えが在る場合には,評価基準により判定する．前節で述べた実行の流れの中で，役割定義から実際のサービスが起動される場面での機構を図 4-6 に示す．個々のサービス機能が選択される様子を図に沿って以下に述べる．

指定された役割名”role1”から，実行エンジン(EE:Execution Engine)は，”Operation-type”の属性を調べてサービスの選択基準を判断する．この例では，”CONCURRENT”なので， Operation-type は，並列処理となる． EE は，サービス群の中から適切なサービスをどのように選択するかを，評価基準情報 ”Evaluation function”により調べる．

役割とサービスの関係を表す情報 “A table complying with Role and service” から， EE はサービス名のリストを得る．

EE は，それぞれのサービス定義情報 “Service definition”から，”Evaluation function” に必要な評価値を得る. ここでは”delivering cost”である．EE は, service1 から service3 に定義されている情報を得る．

値を評価した後， EE は要求されているサービスを選択する. ここでは，”Evaluation function”はコストが小さい方の２つを選択することが示されているので， service1 と service3 が選ばれる．

4.4.3 カプセル化機構

新しい方式の提案は，一方で既存のプログラムの活用を前提とすることも大きな課題である．そして，基本的にはプログラムとプログラムとのやり取りは，交信するメッセージの型とその引数が合致すれば可能となる．しかし，一般的には，メッセージの型の整合，引数の過不足，そして初期値と取り得る値の範囲に関する課題がある [GDiD93].

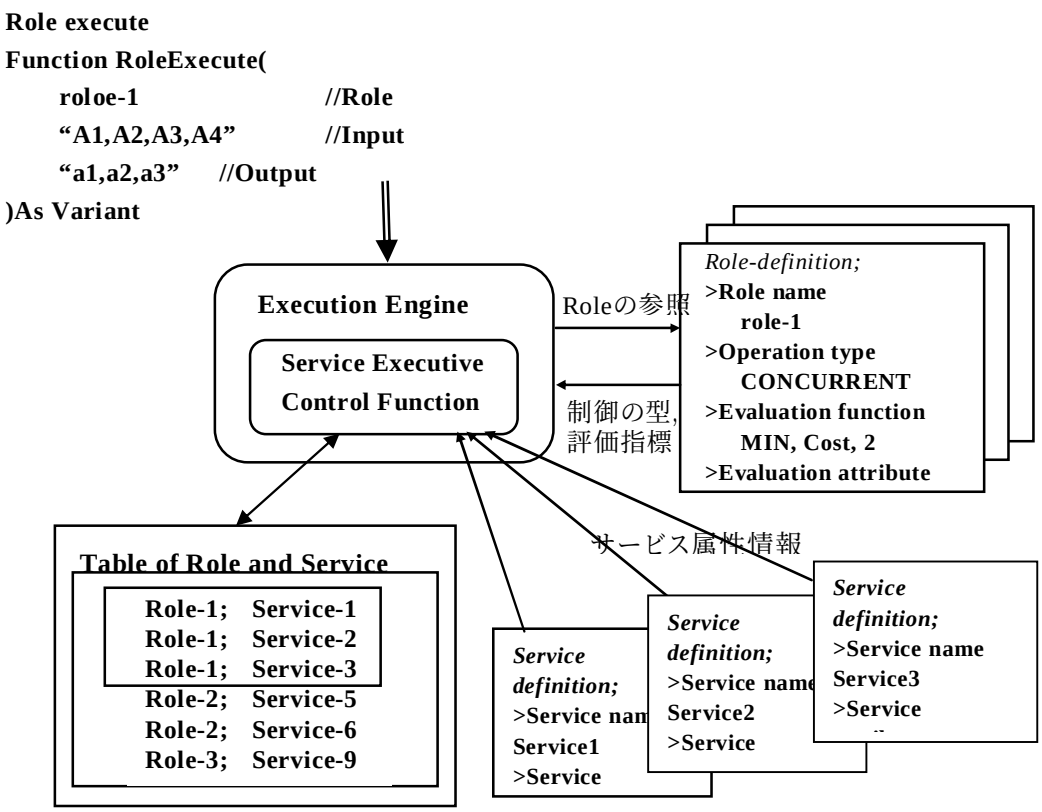

図 4-6.　サービスの協調的選択

このために，対象の業務分野で利用される共通的なデータ項目，データ構造，そしてデータの値の初期値と取り得る範囲，を前提条件として存在させる．この手法は，システムの相互運用性を確保するために，その対象領域でコンセンサスを得たプロファイルを設計する方法と同じである．それぞれのモジュール（ここでは，役割や機能）に関する実装の規約を定め，そこに，引数が必須か否か，引数の初期値，および引数の取り得る範囲をプロファイル情報として備えることが，一般に相互運用性を確立するために行う手段である．渡されたメッセージの引数が未定義の場合は，上記の情報を基にして受け取った側で埋め込んで，処理を実行する．前述の役割と機能の選択における場面での，交信メッセージに必要な引数に関する調整の様子を示す．以下に図 4-7 に沿って説明する．

役割が備えるメッセージに関する情報は，Role-definition にあり，他方機能が備えるメッセージに関する情報は，Service-definition にある．この場合，それぞれの引数に相当する情報の項目数は同じであるが，扱うデータの型が異なっている．

型が異なる場合は，いずれか一方の側の型に合わせるべく変換する．機能側の内部

処理の内容から，特定のデータの型を推測することも可能である．例えば，水の流量の集計を実施している場合の合計値の型が，明示的に定義してない場合などは，この基となる情報源から，流量として扱える数値の型を推定することが可能である．

それぞれの引数の初期値とその取り得る値の範囲に関する課題がある．この課題は，実装の段階において，それぞれのモジュール（ここでは，役割や機能）に関する実装の規約を定め，そこに，引数が必須か否か，引数の初期値，および引数の取り得る範囲をプロファイル情報として備えることが，一般に相互運用性を確立するために行う手段である．渡されたメッセージの引数が未定義の場合は，上記の情報を基にして受け取った側で埋め込んで，処理を実行する．

以上の手順を踏んでもなお，起動ができない場合は中止する．本例の場合は，Operation type が”RECOVERY”なので，他の起動できる機能を探すことになる．型が異なる場合は，いずれか一方の側の型に合わせるべく変換する．機能側の内部処理の内容から，特定のデータの型を推測することも可能であるが，一般的に適用はできない．それぞれの引数の初期値とその取り得る値の範囲に関しては，プロファイルにより定める．

4.4.4 共通データ構造

一般にアプリケーション・システムが，他のアプリケーションを利用する場合，例えば複数のデータベースやファイルシステムを操作する際に，互いの資源を利用するインタフェースが必要になる．文書処理や CAD/CAM の領域では，共通データ構造を定めてそれを共有する．本研究では，より汎用性を持つ抽象データオブジェクトと呼ぶ中間的な情報セットを用意して，この情報を介してアプリケーション・システムはアクセスする方式を用いている．

情報処理系のシステムでは，既存データベースシステム，あるいは特定のファイルシステムを利用する場合が多く，且つ複数のシステムを同一の業務システムが制御する．このために，それぞれのデータ構造の異なる情報資源を操作する仕組である．

処理を依頼する側からの基本的な手順は以下のようになる．

1) コンテキストとデータプロパティの定義を行う
2) 処理の目標と範囲を定めて，データ型を決める
3) パラメータとプロパティを設定する
4) 実行処理

図 4-8 のような実行環境を構成する．本図の場合は，データベースを操作して，情報を得る場合の例である．

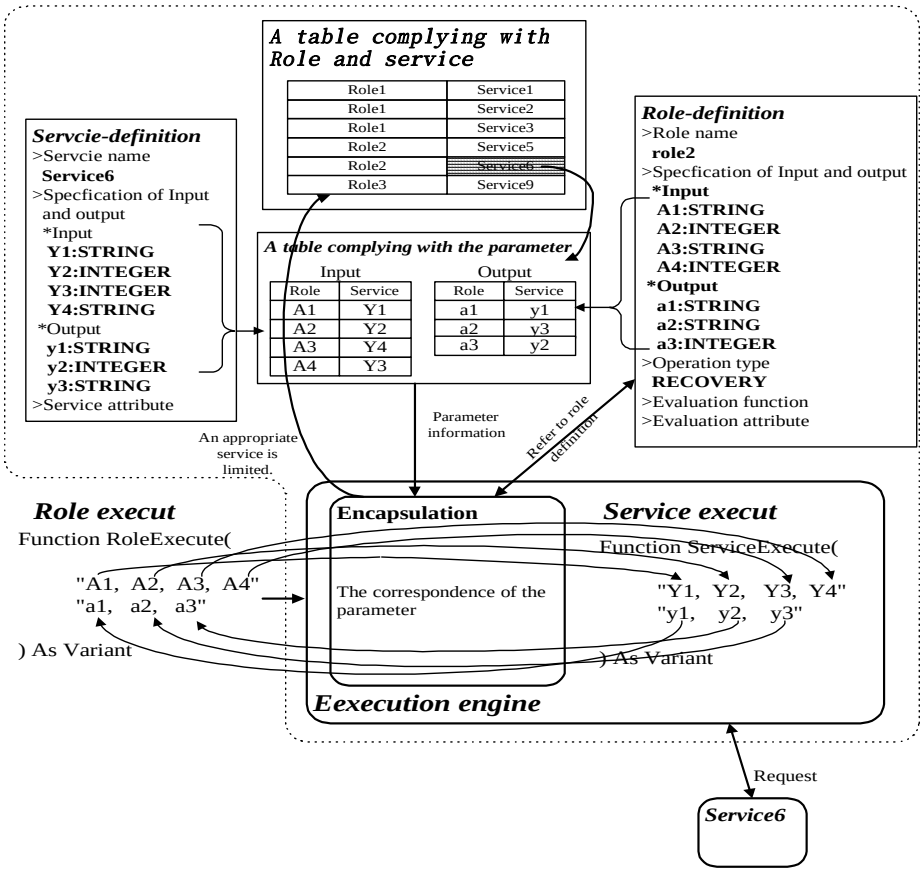

図 4-7. カプセル化

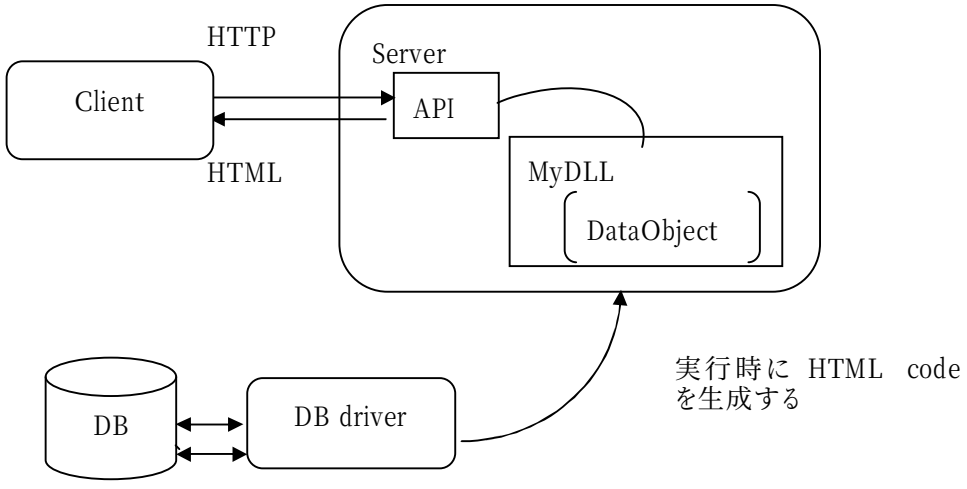

図 4-8. 抽象データ構造の利用形態

4.5 適用例

発電や精製プラントあるいは道路システム等における監視制御システム，及び事務処理系の営業支援システムへの適応を例にして実装の内容を以下に述べる．

4.5.1 監視制御システムへの適用

監視制御システムに適用した例を図 4-9 に示す．監視制御システムとは，遠隔地に存在する監視対象をセンサー機器などにより運転状況を把握し，定常あるいは異常状態や変化の兆候などを観測する．状況に応じて対象システムに対して直接に制御したり，あるいは操作員に操作指示を出して運転する．図中，矩形は処理の単位を示し，矢印は層の内部では処理の流れあるいは手順を示し，層をまたがる矢印は，上位の層から起動される処理を示し，楕円は情報の存在を示す．

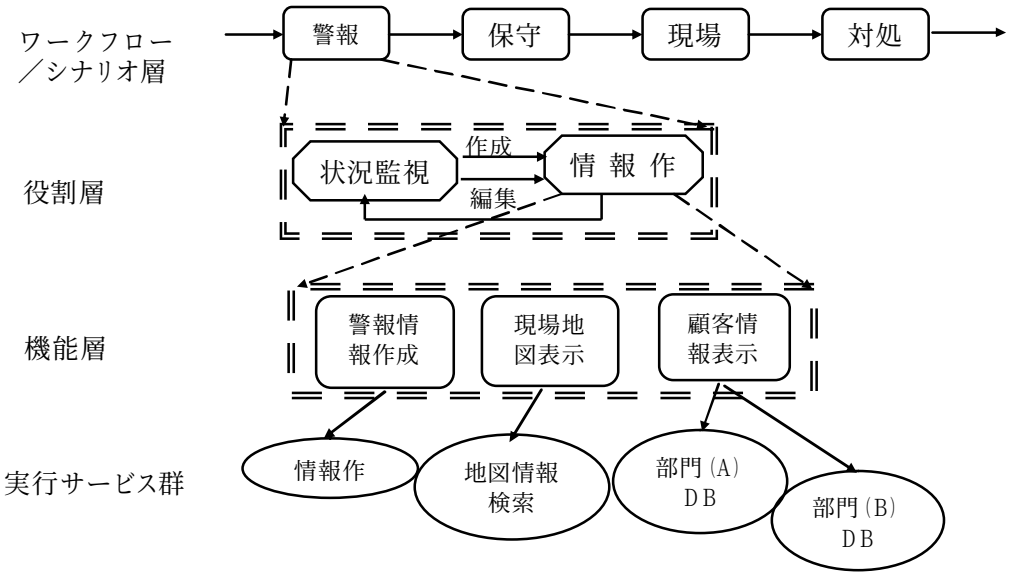

図 4-9．監視制御システムへの適用例

シナリオ層では，その基本的な作業処理の流れを備えている．「警報」を行うための部分で，役割層では複数のビューを備えており，それらを用いて操作員であるユーザが情報を見たり入力する．それぞれのビューに対して処すべき役割が複合化されて定義されている．これらの役割に対する実際の処理が，機能層において実行されて，それらの結果が上位の層へと上げられて伝わる．

4.5.2 営業支援システムへの適用

営業支援システム「化粧品販売」は，週毎あるいは月毎に，顧客毎に関連データベースを検索して納品項目とその価格の一覧を作成し，顧客毎請求金額リストを作成する．あるいは，顧客の要望する品目が在庫としてあるか否か，無い場合には発注すると何時入手出来るか，等在庫データベースを検索したり，関連企業へ問い合せを行って情報を得るなど，営業活動を支援するシステムである．図 4-10 は，化粧品販売支援システムへ適用した例の構成全体を示す．

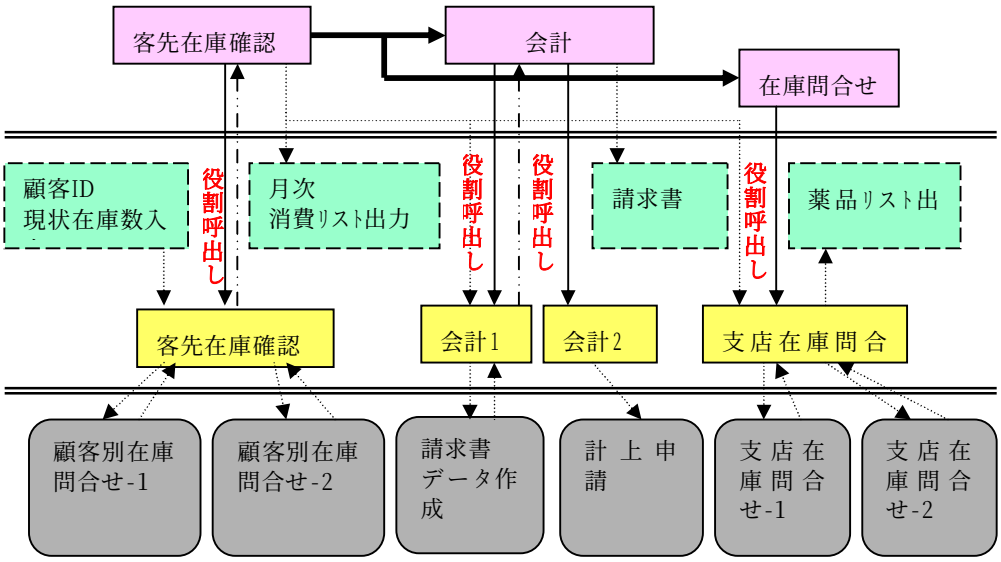

図 4-10.化粧品販売支援システム

更に，図 4-11 は，本適用におけるシナリオ部分の一部である．図に示すように，成すべき事柄の基本要素のみを定義しておけば，順次それを実現するために，役割部分が決まり，そして実際に機能するプログラムが適宜選択されて起動されていく．このように，システムを構築する段階においても，極めて役割要素的なレベルでの設計が進められれことになり，4-３節で述べたオブジェクト指向技術に基づく開発手法とも旨く適合することになる．

```
Sub AccountingService(ByRef Indata, ByVal outdata)
                                    'In:顧客名リスト, Out:請求金額リスト
Dim csvClientList                   '顧客IDリスト
Dim ClientID                        '顧客ID
Dim csvUsedDrugList                 '消費薬品リスト
```

```
Dim Charge                          ‘請求金額
Dim cvChargeList                    ‘顧客別請求金額リスト

csvClientList = TranslateClientList( InData )    ‘顧客名リストから顧客IDリストを作成,
                                    ‘顧客リストの最後までループし,
                                      ‘顧客別請求金額リストを作成する
Do
        ClientID = ParseCsvList( csvClientList )  ‘顧客名リストから顧客IDを取出す
        csvUsedDrugList = BacklogDBQuery( ClientID )
                                    ‘在庫ＤＢから消費薬品リストを検索
        Charge = GetCharge( csvUsedDrugList )  ‘顧客別請求書金額を求める
        ChargeList = CreateCsvList( ChargeList, CreateCsv( ClientID, Charge ))
                                      ‘請求書金額リストを作成
Loop While ClientID <>
End   Sub
```

図 4-11. 営業支援システムのシナリオ例

4.6 評価と考察

4.6.1 実装の状況

表 4-１に，監視制御システムと営業支援システムへの２種類の実装の状況を示す．使用した言語は VB が中心で，GUI 部分は HTML を用い，関連したデータ定義や結合部分でマクロ機能を使用して定義及び記述している．

(1)業務の流れを担う部分であるワークフローとシナリオ部は 10%弱と小さい．機能部にばらつきが大きい，これは営業支援はビジネス系でデータの処理が中心で操作画面でのユーザ操作を反映して複雑な処理量も大きなデータの処理がファイルあるいはデータベースに対して行われる．他方，監視制御システムでは，産業・制御系で機器等の操作処理が中心で操作画面の種類は多くなく，表示されている機器を表すシンボルが具体的な機器に対応しており，その制御操作は比較的簡単である，という業務の特質を反映している．

(2)役割部と機能部の切り分けはなされているが，操作インターフェースに関わる GUI（グラフィカルユーザユーザインタフェース）部分を，シナリオ部分内に取り入れるべきか，あるいは役割部分において定義すべきかが揺れたので，本表では GUI 相当部分を別に表した．この GUI 部分は，一般的に言われている比率に比較すると，小さいと言える．

表 4-1. 実装

	言語･定義手段	監視制御システム 6.6 Klines	営業支援システム 10.0 Klines
ワークフロー･シナリオ部	VB, データ部	7.6%	10.0%
GUI部	HTML, VBCC	15.2%	20.0%
役割部	VB, データ定義	60.6%	36.0%
機能部	VB	15.6%	34.0%

4.6.2 業務の進化への対応

(1) 進化に対する変更の実施

本研究の主な課題は, 進化への対応である. 表2は, 営業支援システムを対象に, 手順と機能の変化に対して, 本提案手法に基づき変更すべき部分の数を評価したものである.

表 4-2. 変更に対する対応個所の比較

変更内容	本論文の方式での影響部分	対応内容 変更箇所数[n]	従来(3層構造)方式での影響部分	対応内容 変更箇所数[n]
業務処理手順の変化(例:手順A/Bの順序の入替え)	シナリオ部分の変更	[2]シナリオの修正	クライアント部の変更	[3]順序及びGUI制御個所の変更
			GUI部の変更	[2]順序制御との起動関係を変更
業務サービス･機能の変化(例:サービス項目数の変化)	役割部分の変更	[0]役割の中に閉じるために不要	業務ロジック部の変更	[1]項目数制御部分の変更
	機能部分の変更	[0]機能の変化ではないので不要		
業務サービス･機能の変化(例:機能の改定)	役割部分の変更	[0]役割は変化しないので不要	業務ロジック部の変更	[1]機能ライブラリ相当部分の交換･変更
	機能部分の変更	[1]機能ライブラリの交換･変更		

この表から, 以下の結果が得られた.

1)手順の変更に対しては, 本方式ではシナリオ部分のみの変更に限定され, 従来方法の 2/5 ほどの修正個所で済む.

2) サービスレパートリーの変化に相当するサービス項目数の変更に関しては, 提案手法では, 役割の中に閉じるために変更の必要は無い. 他方, 従来方法ではこの項目数の制御

部分の変更が必要となる.

3)機能の改定あるいは変更に相当する機能の改定に関しては, 本方式では, 機能部分にのみ限定される. 従来方法においても同様に相当部分の入替え変更が必要であるが, 業務ロジック部分全体に関係する.

(2)サービス機能の変化への対応

機能層では，ネットワークに分散して存在するデータベースやサービスであるアプリケーションプログラムに対する起動のためのインタフェースを，抽象データオブジェクトおよび CORBA の管理方式に基づき定義した．このように，既に存在する，あるいはこれから本枠組みにより作成されるアプリケーションとの中間に機能層を設けたことにより，図 4-4 に示した各層間のインタフェースで定義出来る範囲では，複数の異なる版や機能のサービスを上位層に提供させている．

これにより同様のサービスがネットワークに複数散在する場合に，サービスの物理的位置関係を中間の層が管理することで，上位の層に対する位置の透過性を実現する事が可能となった.

(3) 操作手順や表示形態の変化への対応

扱う対象システムの形態の違いや監視する範囲によって，オペレータや作業者が情報を観察し処理する情報表示の形態が異なる.このように,同類のサービスに対して,適用毎にユーザインタフェースの形態が異なる場合には，ユーザインターフェースの扱うデータの識別肢は変えずに，形状に関する部分のみを対象システム毎に定義することで，情報システムを構築することができる.

これは，ユーザインターフェースの研究における，アプリケーションの本体部分と GUI 部分を分離する UIMS(User Interface Management System)[Pfaff86]の概念で論じられたように，業務を司るダイアローグ部分と表示形態に係わるユーザインタフェース部分において，適用毎あるいはユーザ毎に表示形態を適用させて他は共有するという方式にも旨く則っており，本実装でも旨く機能している.

4.6.3 カプセル化

インターネットの利用が盛んになるにつれて，プログラムとプログラムとの交信する場合が増大し，プログラム間で同じインタフェースを備えない限り互いに連携させることは難しい. CAD の分野あるいは文書処理の分野ではこの類の課題が多く存在しており，明示的に定義してない場合などはプログラムの振舞いの意味的な要素を追求する手法もあるが，必ずしも完全に推測できるとは限らない．我々の実装でも, アプリ

ケーション間で共有できる共通的な情報形式を前提として，既存プログラムを使用する個所では予めツール類などを用いて共通的な情報形式を利用することで，カプセル化により既存プログラムとの共存を可能にしている．

プログラムの間で，交換するメッセージの間で違いがある場合，プログラム処理の分析を行って分析して決定する手法も提案されているが，未だ完全ではない．しかし，本提案で示した共通データ構造と，取り得る値を定めたプロファイルを用意することにより，4.4 節で述べたように交信を可能としている．データマッピングを行うプログラム・ツールの数は，基本的には対象となるアプリケーションの種類の数Nと同じ，N数が必要になる．他方，共通データ構造を備えない場合は，べき乗に近い ${}_{N}C_{2}$ 数のデータマッピング用のプログラム・ツールが必要となる．このように，備えるべきツールの種類を極めて少なくする効果がある．

4.7 結言

業務自身が進化することに，それを担う情報処理システムが柔軟に対応できるためのアプリケーション・システムの構築手法を論じた．システムの拡張や変更に対しても，変更や作成のし直し部分をできるだけ少なくするための機構を提案し検証することであった．

目的に沿った作業を実施するプロセスを，業務遂行の手順・シナリオ・役割・機能の構成要素を導入して，その作業の実行を支援する方式を提案した．ここで，重要な概念は役割である．この役割を媒体として，システムの振舞い部分と機能部分を分離して構成し，実行時にはこの役割を媒体として連携させることで，両者を独立に定義して製作できることを検証した．具体的には手順の変化や機能の変更に対して，従来方式の１／３ほどの修正変更で，進化した業務に対応できること，あるいは全く修正をしなくとも，役割の範囲で吸収してしまい，進化に追随できることもわかった．

ここでも，分散型文書処理で述べた共通データ構造が寄与している．アプリケーション間の情報交換が必須であり，分野や対象領域で共有するプロファイルに基づき，この共通データ構造を用いて情報の交換を行っている．

役割に基づきサービス機能を選択する機構は，次の章で論する協調機構にも適用され，目的とアプローチの関係を機能させる手段として，選択基準を与えることで旨く機能している．

また，このシステム構築の方法は，ソフトウェア工学の領域で行うオブジェクト指向技術を基とする設計方法論での，対象業務や制御の振舞いと機能を分析する手法と旨く整合することが，大いに期待できることが見えてきた．

提案した機構の特徴は，構成要素は変更あるいは拡張に関する自由度が生まれ，自律性と柔軟性に優れ，システム全体の堅牢性を増すことになる．

実装と検証の試みを通して，このような特徴はシステムの保守や運用の段階においても，システムとは独立に交換したり追加することで，優れた効果が期待できることがわかった．

第５章 自律型モジュール制御

5.1 緒言

4章の動的サービス結合における「役割部分からサービスまでの協調的選択」に関しては，実行エンジンの類の制御機構が存在して，支配的に制御した．具体的には，コントラクトネット（契約プロトコル）による手法により役割に基づき，対応できる機能を選択する，あるいは役割に対応する機能を適用する，方式であった．

一般に，コントラクトネットによる手法は，次の3段階を経て実行される．

(1)タスク分割と割当て

(2)部分タスクの実行

(3)部分タスクの解の統合

しかし，最後の段階の統合については，段階(1)の主旨を反映してどのような機構により統合するのかという有効な方式は未だない．

このような，制御機構がトップダウン式に制御する機構は，本論で目指す適用対象業務の進化に対して，柔軟に追随できないと考えている．むしろ，分散環境に存在する多種のサービス機能を少ない制約で利用すること，そしてこれらのサービス機能を有機的に結合することにより，優れた機能を得ることあるいは新たな機能を得ることが可能になる，と予想する．

本章では，アプリケーションシステムにおけるサービスと役割の関係に注目して，サービスプラグインの機構を論ずる．

先ず，分散型アプリケーションシステム構築のアーキテクチャに基づき，サービスプラグインの機構の位置付けを行い，存在する技術的課題を挙げる．

次に，サービスの協調的選択の枠組みにおいて，統括的に処理されるのではなく，自律的にサービス機能が参加する形態の方式を検討する．この課題は，同一の目的を持って自律的にメンバーが参加するという視点から，共同作業あるいは協調的作業の形態と見ることができる．従来，自律協調制御に関しては，制御システム系の分野で，協調モデルやエージェントモデルに関する計算モデルなど，多くの研究が成されている．そこで，研究されてきた概念や成果を参照して，分散情報処理システムへの適用を検討する．

本論では，制御システムの分野で多く研究されてきた自律と協調の概念を基にして，制御の分野での典型的な共同作業のモデルを対象にして，検討を進める．即ち，「神輿担ぎモデル」では，類似の機能と能力を備え持つメンバーが，同一の目的に向かった共同の作業を行う．他は，「ごみ集めモデル」であり，そこでは同様に類似の機能と能力を備え持つメンバーが，同一の目的に向かって行動を起こす

が，最終的には，何らかの仲介調整が行われて唯一のメンバーがその任務を担う．

このモデルに対して，シナリオ·役割·機能の枠組みを導入するものである．即ち，シナリオが全体的な振舞いの枠組みを与えることにより各段階での行動に副目標を与える，この副目標に沿って関係する役割を備えたメンバーが集まる，そこで各メンバーがどのように振舞うかが，副目標と役割の関係からどのような形態の共同作業を行うかが規定される，そしてこの実際の作業を実施するための機能サービスが適用される仕組みとなる．

適用対象の状況に応じた作業を行うことに注目している．予め決められた作業形態をなぞるのではなく，その時々の置かれている状況と目標との関係で，とりうる作業の形態が動的になることにより一層と効率的に目的を達成できると考える．換言すると，対象と自身の関係において，対象からのフィードバックを得ながら自身の行動の形態を動的に調整していくことである．

更に，分散処理における場の概念を導入して，関係構成要素間での，コミュニケーションを行うことにより，関係構成要素の間での調整あるいは仲介の要素機能を実現する機構を，論ずる．

5.2 サービスプラグイン機構

5.2.1 サービスプラグインの概念

プラグインとは，一般には，着脱可能なものを指しこんで結合させる機構を指す．本論では，基盤となる情報処理システムに対して，サービス機能を随時結合してそのサービス機能を享受する意味で，サービスプラグインと称する．図 5-1 にその基本概念を示す．最近の情報処理の領域では，Plug and Play，Cartridge,等の言葉が同じような意味で使われている．

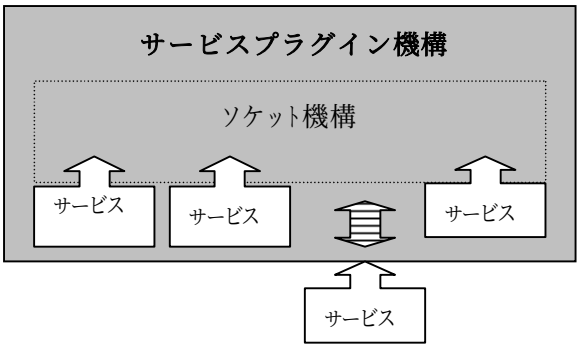

図 5-1.サービスプラグインの概念

これらの用語との大きな違いは，人為的あるいは宣言的に結合するのではなく，自律的あるいは協調的に連結することを目指しているところにある．

本論では，4 章で述べたように，シナリオ･役割･機能の体系に対して，役割をソケットにしてサービス機能をプラグインする手法の確立を目指している．これにより，役割という枠組みの中にサービス機能を自由に組込むことができる，と考える．プログラミング手法の領域では従来から，カプセル化の手法は多く存在するが，どのような枠組みあるいは制約の下でカプセル化を行うかという汎用的な方法がなく，プログラミングの技法として「結合」させるレベルに留まっていた．本方式では，「役割」という明確な制約としての枠組みが存在しているので，利用レベルあるいはサービスレベルでの結合がなされる．

サービスプラグイン機構は，協調的共同作業環境の視点からは，サブゴールに対して参加できるメンバーが集まる環境，と見ることができる．役割はある目的を持っている，この目的を実現するアプローチと手段を備える環境を構成することは，参加メンバーが目的に向かってそれぞれの作業を遂行する形態に類似しているからである．

そこでの課題は，先ず，どのようなメンバーをどのようにして集めるかにある．次の段階で，メンバーが互いにどのように振舞えば目的を達成できるかである．

5.2.2 分散型アプリケーションシステム構築との関係

一般に，システムを構築する際に，全ての仕様を事前に把握することは不可能に近い．また，システムを利用する環境やその方法は，時間や環境と共に変化する．このような変化は，当然仕様の追加･削除･変化を要求することになる．4章で見てきたように，このような変化に対応できることが，その情報処理システムのライフサイクルタイムを長くすることにつながる．システム構築の観点からは，このような，変化に如何に，柔軟に対応できるかが課題である．

ここでは，特に機能の変化に対応することに焦点を当てている．そこでは，当該モジュールの交換で対応できる形態と，複数のモジュールを組合せて対応できる形態とが考えられる．
前者の形態に関しては，特定の条件が整えば，代わり得る候補の中から最適なものを選択して対応させる．これは，候補の列挙と選択の課題，そして交換可能な結合インタフェースの課題となる．他方，後者は，必要なモジュールの抽出と，その組合せあるいは順序制御，そして結合インタフェースの課題が，主なものとなる．

システムを構築する上で重要なことは，様々な形態をとるサービス機能を統一的な手法で且つ汎用的に組み入れることができるかにある．図 5-1 に示したソケットの機構はこの課題を抱えている．単なる組込みであれば，従来のカプセル化の手法で実現することができる．本論では自律的あるいは協調的に連結することが課題であり，そのための機構を備えることが必要となる．

5.3 協調機能

一般に協調においては，何と何がどのように協調するかが主要な課題である．「何と何」に関しては，4章で論じた役割の枠組みから該当する制御主体であるメンバーが選択される，と考える．

本論では，サービス機能などの制御主体がそれぞれに，制御対象の状態を能動的に観測して，自身の役割を遂行できる条件が揃った時点で，自らは行動を起こす方式とした．即ち，起動条件の判定は他に任せるのではなく，自らの判断により行う．実行の基本的機構としては，制御対象に関する情報と，制御主体に与えられた役割の情報を，場に置き，制御主体は与えられた場の状況に依存して行動する．

しかし，依然として制御主体の選定を如何にして行うかは明確ではない．この故に，本論では，起動条件が整えば自らが能動的に起動する方式により[Miyazaki89]，役割の範疇の条件を満たし，この起動条件が整えば選択されたメンバーとして振舞う機構とした．ここで，起動条件とは，3章のオペレーションインタフェース部からのトークン情報を見ていて，どこかの文脈に

合致するトークン情報が届き，起動のためのシンタクスが完結することを示す．

「どのように」に関しては，一般的な協調形態を論ずるほどに，この研究分野においては，課題，技術あるいは方法論は整理されてないので，制御系システムの分野で研究されている焦点を起点として検討する[Sakane95]．　協調とは何らかの仲介が行われて，賛成あるいは反対の態度のような両極を含めたある解を得る要素を持っていることである，とした[Yamashita90]．

基本的な方法とその意義は以下である．

a．複数候補から最適なものを1つ選ぶ方法

同一目的を実行するため唯一に絞る「仲介」が必要

b．n個のタスクに，同じ目的の仕事を同時に協力して行う方法

同じ機能が同時に「共同」して実行を担当する

c．m個のタスクを順次目的に向かって連携させる方法

異なる機能が「連携」して実行を担当する

具体的には，項目 a の課題に関しては「神輿モデル」が相当し，項目 b の課題に関しては「ごみ集めモデル」が相当する．しかし，項目 c に関しては相当するモデルが無いが，この2つのモデルの組合せとも言える．

制御システムの領域では，比較的独立した処理が並行してあるいは連続して実行されるため，この2つのモデルでカバーできる形態が多い，と予想される．他方，ビジネス系では，定形型と非定形型において述べたように，データの値に依存して次の行動が規制されるように，コンビネーションに関する連携の形態が多いと，予想される．従って，同時進行型のモデルではなく，順序連携型のモデルが必要になると，予想される．

5.4　協調のアーキテクチャ

5.4.1 協調モデルやエージェントモデルに関する計算モデルの研究

本節では，自律と協調に関する代表的で，且つ本論に関連の深い研究を概観する[Jiritsu94][Kishimoto91]．

(1) 協調の場

分散処理における場の概念の導入と，その制御方式において重要な役割を果たした研究の特徴を以下にまとめる．

1) Kamui88[Harada90]

オブジェクト指向に基づく計算モデルである．オブジェクトのグルーピング，メッセージを受け渡す範囲や順序を指定するための「場」を導入．この場によりオブジェクトの型付け，そしてブロ

ードキャスト通信や優先度制御を可能にしている.

2) Kemari[Yano92]

自律的な協調を行うことを特徴とした計算モデルである. 協調のためのプロトコルが, 計算主体の内部に保持されている. その特徴は, オブジェクト指向の方法では, オブジェクトに送られたメッセージは, そのオブジェクトが受信する以外に選択の余地が無いが, Kemari では受信するか否かは計算主体によって決定される. Kemari の計算主体の中には, タプルスペースが存在し, 他からのメッセージは全てこのタプルスペースに投入される. このタプルスペースに対するタプルの入出力は計算主体が決定する.

3) Cellula[Yoshida90]

実行主体のプロセスと通信媒体としての場を一体化したCellを中心とした計算モデルである. 通信機構としての場の使い方は Kemari と同じだが, 協調プロトコルなどの複雑な機構を内部に持たない. 問題を分割して最適に配分することを柔軟に行うモデルである. 簡潔なモデルだが直接通信, 間接通信, ブロードキャスト, そしてブロック・ノンブロック型の通信が可能である.

4) Linda [Ahuja86]

計算主体であるプロセスと, 通信媒体であるタプルスペースから構成される計算モデルである. プロセスは, タプルスペースに対する非常に簡単なオペレーションのみで, ブロードキャストの通信を行い, 協調作業を進める. Linda におけるタプルスペースはプロセス間の共有メモリと考えることもできる. 非常に有効な手段だが, プロセスによってアクセスされるので, 識別子の衝突など多くの問題を抱えたままである.

(2)協調のプロトコル

協調の処理に必須のコミュニケーションを担うプロトコルに注目し, その代表的な手法をまとめる.

1) コントラクトネットプロトコル[Smith80]

a. 処理ノード間の通信と制御を表現するための問題解決プロトコル

b. 構成要素は, プロトコルに従って通信する処理ノードの集合と問題全体に対応するタスクと部分タスク群.

c. ネゴシエーションの対象となる部分を「コントラクト」と呼ぶ

d. コントラクトを他のノードに提示するノードを「マネージャー」と呼び, コントラクタを請負うノードが「コントラクタ」と呼ぶ.

e. マネージャがコントラクトを部分コントラクトに分解し, 相当するタスクをコントラクタに委託する. 以降, 分解と委託を続け他のノードへの委託無しにコントラクト履行可能になるまで

続ける.

f. 問題は3つのフェーズで解かれる.

- タスク分割と割り当て
- 部分タスクの実行
- 部分タスクの解の統合

g. 入札仕様や資格仕様までは規定してない.

2)マルチステージネゴシエーション[Conry86][Conry89]

a. 分散ネットワークのための資源割当のプロトコルとして提案された.

b. 部分タスク間の相互作用の処理において, 大域的な制約が存在する問題を複数回のネゴシエーションによって解決する所に特徴がある.

3) 階層プロトコル[Kuwabara89]

a. 処理ノードが複数の抽象レベルでの動作情報の交換して, 効率良く協調動作するプロトコル.

b. 6次元空間上に写像されている動作情報を元に, 動作の重複・競合や類似性を評価して, 将来の相互作用を予測する.

5.4.2 アプリケーション制御

アプリケーション・システムを構築して制御する仕掛けは, 基盤となる分散処理環境との連携とも関係して重要な役割を果たす. ISiS[ISiS92]と呼ぶ分散コンピューティングシステム構築環境の上に構成された Meta[Keith91]の特徴をまとめる. Meta では, システムの状態をセンサーと呼ぶ機構を用いて監視し, その状態に応じてAPマネージャがアクチュエータと呼ぶ機構を用いて AP の動作を制御する.

1) Sensor; 監視されるAPの状態の一部を表現する. それぞれのセンサーは監視するAPの構成要素, 監視する値の種類, 監視する構成要素の実態によって識別される. 環境から直接得られる情報に対応する内臓センサーと, AP のプロパティに直接対応するユーザ定義センサーとがある. 例えば, 次のような情報を獲得する.

- CPU 使用量
- アプリケーション要素の負荷
- アプリケーション全体のスループット
- 構成要素の存在

2) Actuator; 与えられた引数リストを用いてプロセスを起動するものと, 大域変数の修正を行うものがある. 例えば, 次のような制御を行う.

- プロセスの優先順位の変更
- ライトウェイトスレッドの優先順位の変更
- 他のマシンへのプロセスの移動
- 失敗したプロセスの再起動

Meta の場合, センサーをAP自体に組込み, エクスポートする必要がある.

5.5 共同作業における役割

メッセージの交信による非同期型のシステムが注目を集め, オープンな環境で利用できる電子メールをその基盤としたシステムが多く研究され, スケジュールの管理や業務ワークフローに関係する非同期型のグループウェア製品あるいはアプリケーションの開発が行われてきた[Tarumi97][Makabe96]. その後は, 会議システムに代表される実時間処理系の型のシステムも, 共同作業の典型的な適用例として良く挙げられ, そこでは意思決定に係わるプロセスを支援する為のさまざまなアイディアが開発された[Ishii93].
同期あるいは非同期を問わず, 支援を期待する側の活動が組織化されモデル化されることにより, 初めて支援が受けられる条件が存在する. そこでは, 単なる共同の作業では無く, 仲介としての協調がなされて初めて支援の有効性が発揮される.

共同して作業する場合には, 何故に作業を共同するかという目的と手段に対する理由が存在する. 即ち, 参加するメンバーに要求される期待には, 作業を分担することが前提にある. 問題はどのようにメンバーを選択するかという課題と, どのように互いにメンバー間で調整しながら分担するか, 即ち, 協調の形態に関する課題である. ここで, 現在共通的に且つ基本的な課題を以下に整理して列記する.

1. 役割:目的を同じくする参加者・メンバーが役割を分担あるいは共有して共同で作業を行う環境を提供する上で, その役割をどのように分担しどのような形態で共有するかを一貫した手法で確立するまでには至っていない[Uta88].
2. 協調の形態:協調あるいは連携の仕方に関する制御および情報伝送は, 予め定められた役割機能を指示して起動するために宣言的にその振る舞いが定められる.
3. イベントへの反応:複雑な文書情報を扱う事務作業は, 情報をフォームに基

づき形式的に表現し，ユーザの判断により操作される．他方業務手続は，if then elseの形態のrule形式により構造化し，その選択条件が揃うことで実行される．これは柔軟性があるように見えるが，基本的には事前に発生しうる状況が分かっている範囲に限定される[Shu82]．到着するイベントに対して受動的に処理を行い，更にシステムを構成するあらゆる構成要素との間にフォームとのインタフェースを司る部分が必須となり柔軟性に欠ける．

4. マネージャ：分散処理における分配問題は，適用対象に依存してその分配の際には既に解決手法が決まっている場合が多い[Humminen86]．いずれの場合も，全体の状況を把握してその解決のための制御を司るマネージャ機能が必要となる．
5. 経験と演繹：制御システムへの適用でニューロを利用した複数関節を持つ機器が特定経路を潜り抜ける制御の例では，隣合う関節の間での一種の協調作業といえる．経路が変化した場合，あるいはそれまでに学習していない経路が現れた場合，即ち状況の変化に対して学習や試行をせずに対応する事が課題となる[Lesser88]．
6. エージェント：エージェント間の調整による制御で行動スケジュールの候補を示す研究がある．関連する項目間で有効な条件を提示して，最も要求条件に近い解をもとめることになる．この範囲ではいわゆるスケジューリングの手法との違いが明確でなく[Tsutsumi87]，第2あるいは第3の候補を期待するためには，そのエージェントによる効果が明確ではない[VanHilst96] ．

本論では，以上述べたさまざまな課題の中でも，重要な適用対象である制御システムに関わる以下の課題について論ずるものである．

1. 状況に応じた動的な連携や協調：連携や協調の制御を主体的に行う機構をメンバーとなるアプリケーション自身が持たない構成により，システムの状況に応じた動的な連携や協調を参加するメンバーが直接に行う．
2. 役割と分担：各種機能を連携させて補完することにより，効果的に目的を達成するたに，役割をメンバーがどのように分担しどのような形態で共有するかを目的と役割の備える枠組みに従った手法で定義する．
3. 同一機能による協調の形態：システムの置かれている状況に依存して，構成要素が自律的に振舞って対応することで同じ機能を備えたメンバーによる協

調がなされる.

4. 異なる機能による協調の形態：システムの置かれている状況に依存して，構成要素が自律的に振舞って対応することで異なる機能を備えたメンバーによる協調がなされる.

5.6 自律協調機構

アプリケーションを含むメンバーが自律的に振舞える環境条件を備えて参加できる機構と，メンバー間でそれぞれの役割に基づき為すべき事柄をメンバーが置かれている状況を把握し，役割に基づき協調的に動作する基盤システム機構について述べる.

5.6.1 基本方針

一般にシステムを構成する情報処理システムは，機能の関連，操作，イベント，そして制約から構成される[Rich96][Satoh95]. 業務システムの構成要素であるサブシステムの分割に関して，業務システムは共通の目的を持ったシナリオ，役割，そして機能から構成されると考え，論文[Kozuka93]においてその基本的な概念を示した. そこでは，業務あるいは制御システムはシナリオを規定し，シナリオは個々の役割に基づく振舞いを規定し，そして役割は個別機能により実現されると仮定している. 即ち，一連の作業の流れをシナリオ，業務の中で分担する役目を役割，そして基本的な処理を行う機能に基づいて業務システムは制御される，と対応させる. 作業の流れの中に位置づけられる必要な役割を見出し，その役割を実行する機能を確定し遂行させることにより，システムの振舞い部分と機能部分とを分離することができると考える.

他方， 業務あるいは制御の流れは，ある目的を備えた役割を目標にしてこの役割を担う構成要素がメンバーとして参加して，役割を分担あるいは共有して互いに調整を行いながら先の目的を達成する環境を構成する. 我々は，先に自律協調機構を備えた分散処理基盤アーキテクチャ Noah を提案した [Kozuka94] [Satoh94] [Miyazaki94]. そこでは，メンバーが自律的に振舞って置かれている状況に依存して対応する機構を備えている.

提案する機構は，協調連携する領域を分散処理におけるフィールド（場）に設け，協調戦略を場のマネージャが司る方式とする[Akaishi98]. この場は，情報や交信の局所性を持たせるために階層的に構成し，且つ関連する階層の間で場を包括的にグループ化する形態とする.

5.6.2 基盤の要素

(1) シナリオ･役割･機能

実際の適用対象システムに対して, 振る舞いの汎用的定義方法が必要である. システム機能の定義は幾多の手法が試みられているが, その実行基盤に計算処理可能な形態の動作モデルを持つものは少ない[Abe92].

本論文では, ワークフローにおける業務遂行のための振る舞いを構成する活動要素を基にして考える. ワークフローにおいては, 活動単位となる Activity を単位に業務作業の流れを規定する. この Activity には複数の役割が存在し構成している. また, OMT 法に代表されるオブジェクト指向技術に基づくシステム分析や設計においては, 役割に注目してシステムのモデル化を行う. 更に, グループウェア研究に論理的な基盤を与えた AMIGO/Activity-Model においても, この枠組みを利用して役割を定義している.

業務･制御の基本的な流れであるシナリオ, それぞれの構成要素が持つ役割, そしてその役割を実現する手段としての機能より, 構成されるとする. この関係を図 5-2 に示す.

業務あるいは制御の基本的な流れは, 役割を単位とする作業あるいは制御要素から構成される. この役割部分を交換可能あるいは選択的に構成する事により, 同じシナリオで異なる内容の役割による業務の遂行が可能となる.

同様に, 役割はそれを実現する手段であるアプリケーション機能あるいはサービスに連携する. この機能あるいはサービス部分を交換可能あるいは選択的に構成する事により, 同じ役割に対して異なる機能あるいはサービスの提供を受ける事が可能となる.

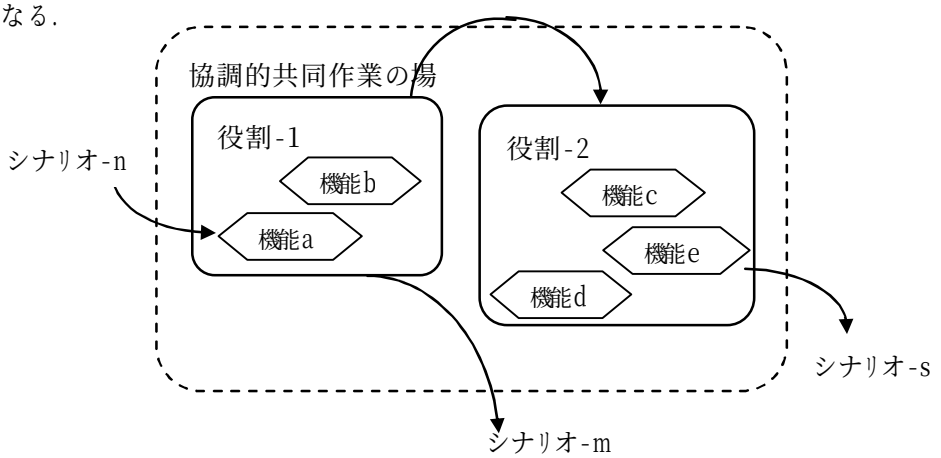

図 5-2. 協調的共同作業とシナリオ･役割･機能

システムの状況, 外部からのイベント, あるいはユーザ要求が状態情報として提案する協調処理の「場」に置かれる. この「場」の情報に基づき, その振る舞いの基本的な枠組みを与えるシナリオに沿って, 個別役割に対応する働きを実施するために, 何をすべきかを回答・提示して, その判断結果に基づき, 指示された機能を実行する.

(2) 分散処理のフィールド概念

分散型ネットワークシステムの基本的な基盤では, 組織やコンピュータシステムが提示する情報を取捨選択しながら参照あるいは取得し自身の情報に反映し, 更には逆に提示する処理が必要となる. 即ち, 計算主体が局所的な情報を持ち, その部分あるいは全体の情報を共有する「場」となるのが情報フィールドの概念である. この類の代表的な計算モデルに論理的なパターン照合の手法を用いて互いに通信を行うLinda[Ahuja86]がある. このモデルは均質な環境への適用が可能であるが, 情報や関係の局所性あるいは大規模なシステムへの適用には, 不必要な通信が生起され通信の無駄が生じたり, 特定の相手への伝達手段の機能において難がある. それを改善する階層化や局所化の研究も多い[Nakashima91][Mori93][Kondo92][Yoshida93].

これらの試みを基にそれぞれの関連する階層関係にある場の間に包含関係を導入して, 前述の役割を分担あるいは共有する制御を場を介した情報の伝達により行う機構とする. このアーキテクチャを図 5-3 に示す.

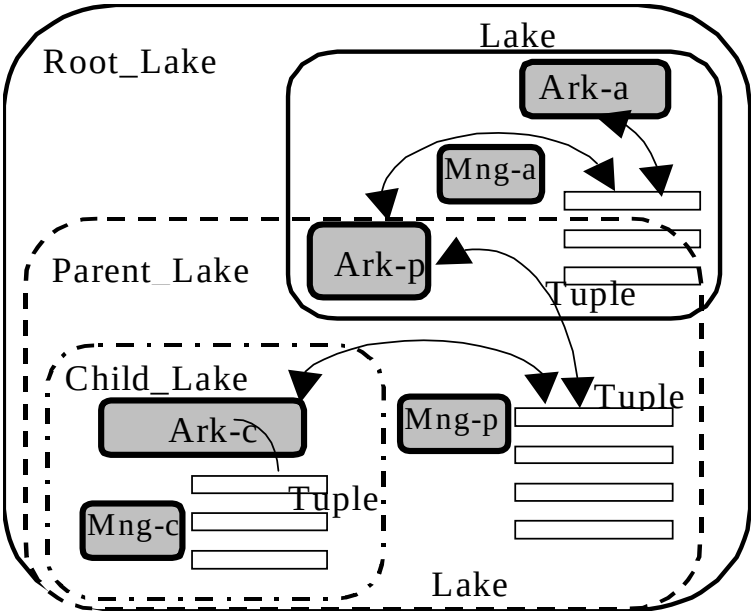

図 5-3. 階層・包括構造を備えた自律協調アーキテクチャ

このモデルは, 協調するアプリケーションが”協調の場”を観ながら自律的に参加する形態を採用している. 図中 *Lake*は場に相当した動作環境であり, 複数の *Ark*, 及び *Lake* 毎に存在する *Mng* と *Tuple* から構成される. *Lake* は *Parent_Lake* と *Child_Lake* のように階層構造を成して環境の継承関係を構成する. *Ark* は, アプリケーションの機能処理単位であり, 場の状態変化を検知して, 変化に応じた処理をメッセージの交換のためのメッセージ掲示用リスト *Tuple* を介して行う. *Mng* は, その都度場に与えられた役割を保持し, 場が制御する対象の状態情報を *Tuple* を介して得て, 更に関連するメンバーからの対応メッセージを取り込み, 与えられている役割に基づき選択及び判断を行って制御を行う. 即ち, *Lake* に存在する資源を用いて, 役割に沿った制御を行い, 且つ場自身を管理する管理機構である. また, *Ark-p* のように複数の場に包含関係を備えて所属し, 場の状態情報を他の場の状態情報として伝達する機構を備える. 即ち, 局所化された場の状態情報が他の関連する場へ伝わるので, 関係のグループ化が可能となる.

(3) 場のプロトコル

情報フィールドを用いて互いに通信を行って協調的な調整を行う ContractNet 方式[Smith80]に, 基づいている. 仕事(タスク)を幾つかのサブタスクに分割して応札させる, 複数の応札に対して選択のポリシーに基づき優先度評価して, 最も適当なものを選択して実行させ, その結果を集約する.

前述の役割と機能に関する情報を場の *Mng* に与え, この役割に従って, 場の *Mng* は自身の場で自律的に判断して管理する. 参加するメンバーは関連する情報を場から選択的に入手し, 自らの役割と機能に応じた可能な対応をフィールドに返す. 図 5-4 に, 場の中に関連情報が存在するタプル *Tuple* の操作機能を示す. この結果, その場を管理するマネージャが与えられた優先条件に従って解を決める.

ys_in(field_id, format, tuple, time);	input the information
ys_out(field_id, format, args[,args,.,.]);	output the information
ys_del(field_id, format);	delete the field data
ys_init(func_p);	initialize the field
ys_new(path, fieldname, module);	create a new field

ys_search(path); retrieve the information

ys_state(field_id, field_num, tuple_num, action_num, path);

read the field status

図 5-4. タプルの操作

5.6.3 制御状態

一般に, サブシステム内で制御状態を生成し, サブシステム内でその実行が可能か否かを判定し, サブシステム内部で完結するのであれば, 制御状態はサブシステム内に隠蔽され, 自律的に振舞う. 否であれば, 他のサブシステムへの機能分担が発生し, 協調制御が必要であれば, 他に伝える情報が制御状態になる[Iwahashi98].

5.7 課題モデルへの適用と評価

協調動作には大別すると次の2形態がある. メンバーが共同で仕事を分担して行う形態と, 他は競合するメンバーの中から仕事を担当するメンバーが仲介の結果決まって行う形態である. 本章では, 共同作業の適用対象として, 提案する役割型自律協調方式を, 制御システムにおける自律と協調の問題[Itou93][Kitamori93][Suda93][Baba94]における典型的な適用モデルとなっている前者の形態の神輿担ぎと, 後者の形態のゴミ集めに適用し, その評価を行う.

5.7.1 適用

(1)神輿担ぎ

神輿担ぎは, 同一対象である神輿の状態が保持されている場の情報を, 同じ役割のメンバーが制御対象の状況として把握し, それぞれの能力に応じて担ぐ共同作業を行う例である[Sakane95]. 神輿担ぎの基本的な構成を図 5-5 に示す. 多くのメンバーが参加するが神輿の4支点にそれぞれグループ化している. グループとしての担ぐ力をそれぞれの支点での総力として扱い, 更に各支点の力を総合して対象とする神輿の高さが制御される. この例では, 神輿が所定の高さに持ち上がるまで, メンバーの背丈と神輿の高さの差 g(i)を見て各々のメンバーの能力に応じた力 power(i)を出して持ち上げ, そして定常の高さに達した後は, 各メンバーの能力と神輿の高さに応じた力で持ち上げる, という簡単なモデルに依っている.

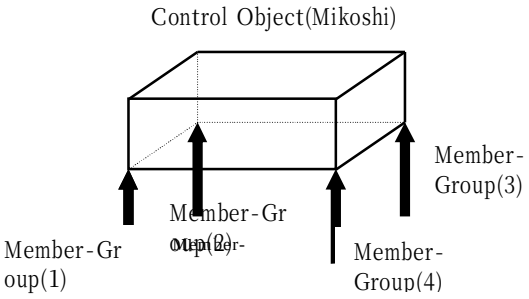

図 5-5. 神輿担ぎ (Mikoshi Carrying)

グループに局所化した役割分担と局所間の調整の例を図 5-6 に示す. 神輿全体の制御管理部分のマネージャは, 前述のように4つの支点にグループ分けして制御している. グループ1において, メンバーが減少してしまいその支点を支えきれない事を支点1の部分のマネージャは検知すると, 上位のマネージャに他のグループからの支援を要請する. これに応じて各グループへメンバーの供出の可能性を打診し, 対応が出来るグループ(図の場合, グループ3)が応じてその旨を伝える, そしてその解を要請したグループ1へ伝えて, メンバーを移動する. これは, 導入した場の階層関係と包括関係を持ち込む事により, サブシステム内で制御状態を生成し, その実行がサブシステム内で実行が可能か否かを判定し, サブシステム内部で完結するのであれば, 制御状態はサブシステム内に隠蔽される, 否であれば, 他のサブシステムへの機能分担が発生する例である.

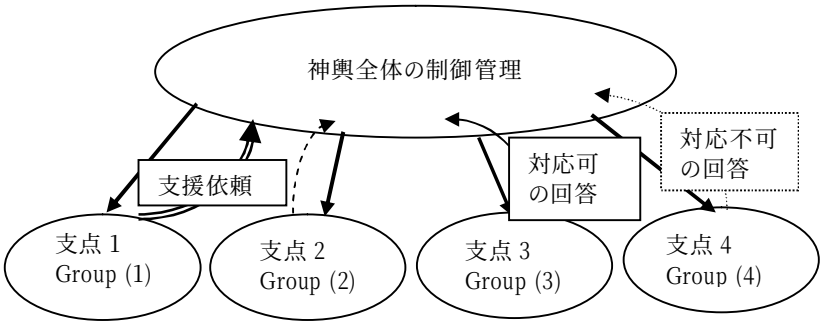

図 5-6. 局所的役割分担と局所間調整

(2) ゴミ集め

ゴミ集めは, 共同作業メンバーの間で同一のゴミを拾うという作業の衝突が起こり, 協議により任された唯一のメンバーが実施する形態の共同作業の例である[Sakane95]. その基本的な構成を図 5-7 に示す. 対象物であるゴミとメンバーとの間の距離が一定の間隔以内になると, 同じゴミの廻りの他のメンバーの存在を確認する. この時に, 図 5-8 に示すように, 場のマネージャは, 当該ゴミに関するメンバーを関係者とする調整の場を生成して, その新たな場のマネージャにその調整を委ねる. 調整の結果, 指定されたメンバーがゴミを拾うとその場の役目は終了し解消される. この結果, 他のメンバーが既にその領域に存在する場合には, 互いに協議する, あるいは先着優先とする等の調整規則のより仲介作業が行われたことになる. 図中, *競合 1*の後で*競合 2*が発生しているが, これは調整が完了するまでに更に他のメンバーが当該領域に進入した場合は, 更に調整するメンバーが増えて再調整が行われていることを示す.

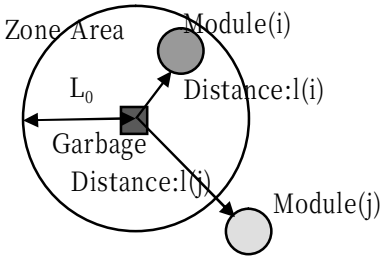

図 5-7. ゴミ集め

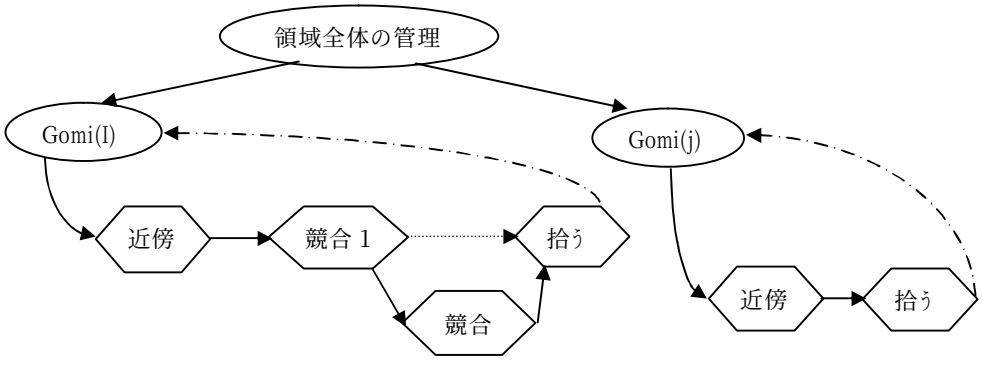

図 5-8. 調整の場の生成

図 5-9 は，32x32 の矩形区画の中にゴミをランダムに 4 個置き，ゴミを集める 4 台のロボットが縁にランダムに置かれそれぞれのゴミを目標に作業を行った場合の実施結果を示す．この図では，全ロボットがゴミの位置を確認するために行う通信の総和を *T-messg*，特定領域内での通信の総和を *S-messg*，特定領域に辿り着いた結果既にたのロボットがそのゴミを標的にしていることにより目的にしていたゴミを放棄する回数を *Cancel*，そして全ロボットが移動した距離を *Length* により，横軸に示す特定領域の距離ごとに示す．縦軸の数値は 32 回の試行の総和を示す．

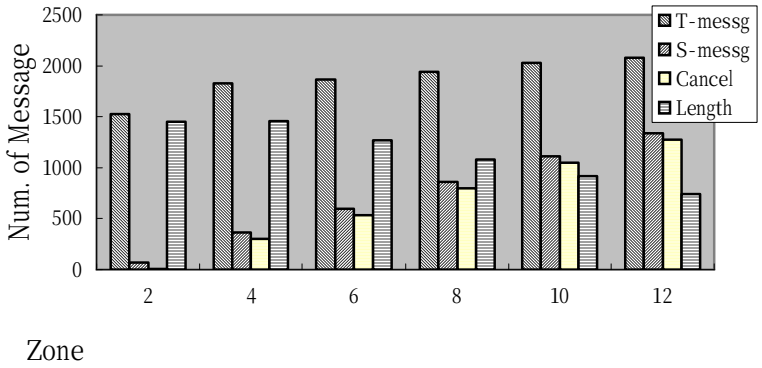

図 5-9. ゴミ集めにおける協調の効果

5.7.2 評価

制御対象にしているシステムの状況に応じた動的な協調的対応が出来る機構を目指し，そのシステムの状況を全体あるいは部分的に反映している環境として分散処理システムにおける場の概念を用いた．即ち，システムの振舞いの中で注目している情報や課題を備える協調作業用の情報空間として場を捉えた．これらのシステムの状態を反映する情報はタプルとして揃え，それらの情報とのパターン照合により検索操作する機構により実現した．

このような協調作業空間を用いて，制御系システムへの適応と情報処理系システムへの適用を行った．

(1) 同一機能による協調の形態

制御システムを対象にした代表的なモデルを対象に検証した．

1) 神輿担ぎ

対象メンバーの局所化により, 全メンバーが直接の制御対象を観測する必要はなく, 間接的に中間に存在する支点の動きに対応することで, 目的である神輿を担ぐという役割を協調的に担う事ができる. 同時にこの事は, ネットワークを利用する場合に, 広範囲に渡り情報を交信せずとも特定の領域に閉じて行えば良いことを示し, ネットワーク系の全体の通信量を削減する事になる.

関係メンバーに包括関係を持たせる事により, メンバーの交代や増減がある場合にも, 包括関係にある部分での対応で処理されるので, 直接に全メンバーが対応する必要はない.

2) ゴミ集め

協調対象の局所化により, 固定的に局所化する領域や時間を設けるのではなく, 特定の事態, この例の場合は, 特定のゴミに対して複数のロボットが集める行動を起こした状況になってから調整の場が設けられ, その調整が完了すると消滅する.

局所化する領域は, 本適用例のようにある種のトレードオフが存在し, 全体の通信量を抑えるが, 余分な動きをしてしまう結果となる. 従って, 何に焦点を当てるかによりこの領域の大きさを設定する事になり, 特定領域の内部においてのみ仲介作業が行われる. 図 5-9 は, 領域の大きさに依存して, 仲介のための情報交換の通信量は増大するが, 全体としてのロボットの総移動量が減少することになり, 協調作業の結果, 処理時間が短くなることを表している.

(2) 役割に基づく協調の振舞い

役割を基盤として各種機能の選択を行った. 従来は, *Roles* から *MessageObjects* への関係,そして *Roles* から *Functions* の関係を, 経験や知識等のある規則に基づけて明示的に行なわなければならない. このような規則の存在が前提となっており, システムの状態に依存した動的な関係を得るための規則の生成手段はない, という欠点がある.

本章では, 役割という行動の目標となる枠組みを設ける事により, この目標を達成出来る能力を備えた機能群が関連付けられ, 当該システムが置かれている現在の状況と要求との間に期待される最適な機能が協調的に選択されて実行される.

この特徴は, 役割という目標範囲に相当する中から, その時点でシステムに存在する実行可能な機能群が選択され動的に起動されるという, 柔軟な構成を備

えられる事にある．これは，システムが運用されている段階においても，機能群に対応するアプリケーションを拡張あるいは改善した版をその交換のみで対応出来る，というシステムの進化に対する対応が容易になることである．

5.8 監視制御システムへの適用

検証を目的とした制御システムの課題モデルへの適用と評価を踏まえて，監視制御システムの「水力水門監視制御システム」へ適用した．

本システムは，特定地域の降雨量，河川水位，及び沼湖の水位の観測，沼湖の水門の制御，並びに道路運行管理を，行うシステムである．それぞれに，観測と通知，開閉の制御，そして安全運行の役割を備えている．適用の様子を図 5-10 に示す．

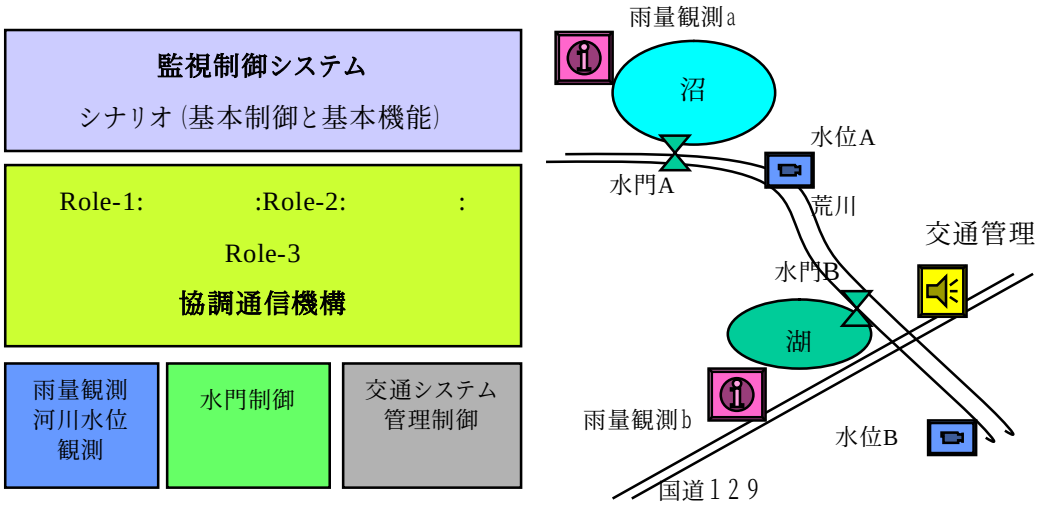

図 5-10．水力水門監視制御システム

山側から平地に向けて川が流れている．途中に沼と湖があり，湖の側を国道が走っている．沼と川の間に水門がある，同様に湖と川の間にも水門がある．近辺では，降水量を測定している．また，川の水位も沼と湖の下方で測定している．道路は，運行管理する部門がある．

以下のように制御される．雨のために，山側で降水量が増大し沼の水位が上昇し，放水の必要性生じた．どちらの水門も閉じている状況を確認して，川の水位を観測して，放水による影響が安全の範囲であることを確認して，放水するために，水門Aを開ける．

更に，降水量が増えて，湖の水位も上昇．既に水門Aが開いている状況を把握して，水門

Bを開けるために, 影響の予測をする. まだ, 余裕があることを判断して, 水門Bを開ける.

更に, 降水量が増えて川の水位が上昇してしまう. 既に, 両方の水門が空けられていることを, 踏まえて道路の利用が危険であることを予測する. 運行管理は通行止めの手段を講じる. 更には, 周辺地域へ, 避難を勧告する

このように，自律型モジュールは，場にある制御対象の状態を見ながら，自身が自律的に行動する．この行動は，場の情報に規定されるので全体として，この場に参加しているモジュールが協調的に連携することを実現している．

5.9 結言

役割を行動モデルとする協調的なアプリケーション連携の実行基盤を論じた. 共同作業の枠組みに, 分散処理における場の概念を導入した自律協調機構を提案した. この機構に基づき, 制御システムにおける自律と協調の代表的な課題モデルに適用し, その振舞いを検証し, その有効性を述べた.

これまでの共同作業に関する研究は, 情報処理システムを利用するユーザが共同作業する場合に, コンピュータシステムにより支援を受ける環境の域に留まっていたが, 役割に基づく行動モデルを, 関係するソフトウェアシステムや制御機械システムなどのメンバー自身が利用して. 能動的にそして互いに協調して行動できる見通しを得た.

この際, メンバーは直接にやり取りを行うのではなく, 役割に基づく行動を行う環境としての場を介して, 行うところに特徴がある. 即ち, 場が対象システムの状態を反映しており, それに対してメンバーが役割に基づき対応する. これは, メンバーが場に与えた結果が, 他のメンバーが受ける機構となり, 作業の効果が互いに伝播して行く. また, 対象が複雑になっても, 役割に基づく行動モデルは不変のものとなり, 安定した振舞いを行うことになり, 信頼性と保全性に寄与する.

本章で提案した場の階層関係と包括関係を備える事により, サブシステム内で制御状態を生成し, その実行がサブシステム内で実行が可能か否かを判定し, サブシステム内部で完結するのであれば, 制御状態はサブシステム内に隠蔽される, 否であれば, 他のサブシステムへの機能分担が発生する. この場合に, 協調制御が必要であれば, 他に伝える情報が制御状態になる.

これまでは, 人間の行動を支援することに焦点を当てた研究開発が盛んに行われたが, 一方で本論文で示した適用例のようにサービスの主体が協調的に振る舞う側面は多く存在している. この事により, コンピュータシステム内部に振る舞いの主体が存在し, それらが互いに協調することにより, さらに革新的な共同作業が実現できると考える.

第 6 章 結論

情報処理システムを構築する際に, システムの利用の仕方や, 目的あるいは業務の形態やサービスの内容が, 時間と共に進化していく中で, その情報処理システムを再構築することなく, 適用環境に情報処理システムを適用させていくための分散型アプリケーション構築アーキテクチャを提案した. このために, 次の3点に注目した.

(1) 情報処理システムにおけるユーザインタフェース部が担う多くは情報の表示と操作である. 扱う処理対象の情報を, 文書という形態の情報表示という視点から捉えて構造化し, 情報が備え持つ意味, 及びシステムの内部状態を示する機構とする.

(2) 業務の形態やサービス仕様が変化しても, システムを再構築しなくとも, 外部定義の変更や必要な機能を動的に結合して連携させることにより, その変化に対応できる機構を設ける. このために, 役割の概念に基づき, 情報処理システムの目的とそれを実現する手段との間を関係付ける.

(3) 目的と手段との連携において, 目的に対する振舞いのモデルを前述の役割の枠組みにより設定し, この枠組みの中で, 自由に且つ柔軟に行動できる仕組みとする.

本研究と関連する分散処理システムの基盤となる従来の研究を整理し, 上記3つの視点に基づいて提案するアーキテクチャの全体を構成した. あおの基本構成要素は, オペレーションインタフェース部, 協調型シナリオ基盤部, そしてサービスプラグイン部である.

システムとユーザの接点となるオペレーションインタフェース部において, ユーザからの要求をシステムへ伝えるために, 操作イベントに依ってシステム内部に持つ業務の流れの定義に基づき解析しながら進行させる機構を設けている. また, システムの状態情報をユーザに提示するために, システムの内部状態情報をこの機構を用いて行う方式を構成している. ここで扱う情報はシステムの振舞いの意図を効果的に伝えること, 且つ操作に基づき情報の構成要素単位で制御することを, 目的に木構造に基づく情報の構造化を行った.

オペレーションインタフェース部では, ユーザとシステムの間で互いの意図を正確に理解しやすく表現することが基本的な要素となる. このために, 情報の意味的な関係を簡素に表現するために, あらゆる情報を文書の概念に基づき表現することとした. そして, 情報の意味関係を分かりやすく表現する手段として, そのスケルトンともなる論理構造を導入した. これは, 日常においても比較的一般的に利用される構造である. この論理構造が, 対処としている業務や制御対象を旨く表現することが鍵となる.

同時に，ユーザは情報の内容を表されている物理的な形態的な表現から，直感的にその表す意味を捉えている．この形態的な情報構造をなるべく情報処理システムの内部に保存することが良いと考えられる．この手段として，レイアウト構造を採用した．なお，このレイアウト構造では，あらゆる形態のメディアを用いて情報を2次元空間に表すために，エリア，フレーム，そしてブロックという空間要素を導入して表現する機構を開発した．この形態的手段は，極めて簡素な手法でも在り，この考えは，国際機能標準を定める ISO の規格 ODA にも利用された．

協調型シナリオ基盤部に関しては，ユーザの要求を受けて，分散ネットワーク環境に存在する各種のサービス機能を有機的に連携して，目的の機能を実行するために，役割の概念を基にして，業務の処理の流れを表すシナリオと，サービス機能を動的に連携させる機構を提案した．そこでは，役割に基づき選択されたサービス機能を，プラグイン機構によりシステムに結合し連携させて，機能させる新しい方式を構成している．複数の業務システムに適合して，その方式を検証した．

一般に，処理を実行する機構は，モジュール化，あるいは分散環境でのオブジェクト化という視点から検討されてきた．それは，ソフトウェアシステムを組み上げる，という視点からのアプローチであった．本研究では，処理の実行は，目的と実現手段の関係に在る，との見方から，目的と手段を論理的に分離することで，普遍的な目的と，それを実現するその時々の状況に応じた手段との，関係と捉えた．この場合の問題は，この両者をどのように一般的方法によりに関連付けるかである．本研究では，ソフトウェアシステムの分析と設計において利用される，オブジェクト指向技術に基づく OMT 法の枠組にある役割の概念を採用した．即ち，業務の振舞いを表すシナリオを実行の基盤となる役割に基づき解釈して，その実行機能を選択的に起動するサービス構築体系を開発した．

この役割の概念を導入することにより，業務システムの振舞いと機能の関係が明確になり，対の関係にすることにより分離できた．これにより，業務システムあるいは制御システムは，その振舞いを汎用的一般的な手法により，その基本的な動きを定義しておく．定義された業務を，実装された情報処理システムは，与えられている環境の中から，可能な機能を見出して実現を図る．

機能ソフトウェアを動的に関連づけて実行させるので，この機構を，サービスインテグレーションと呼んだ．分散コンピューティング環境 DCE あるいは，分散オブジェクト機構 OMG が提供する分散ネットワーク環境で動作する．このことにより，ネットワーク環境でのサービスの連携は，役割をソケットのように構成することで，振舞い部分と機能部分との間の関係を構成でき，ネットワークを介した連携の制御を行う上で極めて有効に作用した．

ソフトウェア工学の代表的な手法であるオブジェクト指向技術における，役割に基づく分析

手法とも，旨く整合しており，業務の分析結果から，本アーキテクチャへの移行が容易と期待される．即ち，対象とする業務を理解して情報処理システムにより処理可能な形態に整理する．このために，人の記憶と想起の機構に基づいて，対処とする業務を分析して整理する．このオブジェクト指向技術に基づく分析と設計の手法(OMT)の領域で，この分析の過程で，実世界の業務について，「役割」に注目して分析し整理している．このことは，「役割」の視点から物事を見ることにより，業務作業の振舞いに相当する部分と，それに関連する機能を実行する部分とに，分別できる可能性が高いことを示している．

サービス機能を結合し連携させる方式として，自律協調の機構により，対象とするサービス機能を選択し，更に複数のサービス機能同士で協調的に振舞う機構を検討し，自律的に行動する形態と協調的に行動する形態の特徴を整理し，分散処理における場の概念に基づく協調連携の機構を考察した．

置かれている状況に対して自らが担う役割に基づき応える機構とした．この役割は，前述の役割と同じ物であり，いわば実行主体側が積極的に，自らが置かれいる環境・状況に応じて，自らの役割の範囲で行動する機能を備えている特徴がある．それぞれの機能モジュールが置かれる環境を場と定義した．その場には，システムが現在置かれている状況が反映されていて，その関連の情報が場に置かれる．更に，その場には，その場の目的が与えられる．これにより，どのような振舞いをすべきかが確定し，その結果，その実現に必要な役割が明確になる．この役割が明確になることで，その実行に必要な機能モジュールが，順次決定されていく機構を開発した．

この機構の検証には，制御システムの領域の典型的な課題モデルに適用し，機能モジュールの協調的な振舞いの検証を行った．即ち，各モジュールが同じような機能と能力を備えており，互いに協力して同じ目的の制御対象を制御する形態の協調的制御と，同じ目的を備えたメンバーが互いにメンバー間で調整を取って，実際の行動は唯一のメンバーが実施する協調的制御である．役割がメンバーの行動モデルとして機能するこで，メンバー間の協調作業を実現した．

このように役割に基づく行動は，複雑で予測が困難な状況に遭遇する場合には，対応できる基本機能が確実に機能することにより，確実な対応が可能になるという特徴がある．複雑系のシステムに対する有効な対処の一つともいえる．

また，協調機構における参加メンバーの間での連携を支援する場合にも，メンバーが実行した結果が何らかの状態情報として反映されれば，その情報を他のメンバーが見て，対応することが可能となり，連携した共同の作業が協調して行われる．人に限らず，ソフトウェア・システムや産業システムにおけるロボット機器のように．独立した機能を備えて自律的に機能するも

のの間でも，同じような現象が存在する．例えば，車の部品を複数の溶接機器が共同作業のごとく行う，あるいは穴開け機器による作業の次にねじ止めを行う機器が作業を行う，等である．

このように，対象の状態情報が把握できれば，人も含めて，ソフトウェアや機器類が互いに協調的連携することが可能となる，と見られる．最近では，人口知能や知識処理の領域からのアプローチであるエージェントに代表される「何らかの意味で人間の活動を支援するソフトウェアモジュール」という認識が多い．

生産活動の中で，活動単位毎に操作あるいは加工対象となっている製品に関する状態情報などの成果物が生成され管理される．この成果物は，各段階の前後で参照され加工編集されて，次の段階へと進む．即ち，製品の営業，設計，生産，販売，そして保守等に関する電子化された各種情報が存在し，それぞれの生産段階の担当する部門が，これらの情報を生成し次の段階でそれらを参照する．あるいは，従来は事務処理系で使用されていた文書情報と，エンジニアリング系で使用されていた設計情報が，互いに利用し合うことにより，複数の業務あるいは業種の間で，情報を媒体とした新しい形態の事業や情報サービスが生まれてくる．また，アプリケーション連携の仕組みは，ネットワークに存在する各種の情報源やサービス機能を用いて，新たな形態あるいは仕組みのサービス産業が生まれてくる可能性を示している．

以上述べたように，分散型アプリケーションシステム構築アーキテクチャを提案した．役割の概念を軸にして，システムの振舞いと機能を関連付けることができた．これにより，従来は，宣言的に定義していたシステムの振舞いとその機能の関係を，分離することが可能となり，機能部分を柔軟に選択的に結合して実行することで，システムの置かれている状況に，適した処理を行うことが可能となった．

謝辞

静岡大学社会人博士課程におけます本研究に関しまして，ご指導ならびに有益なご意見を頂きました静岡大学情報学部教授水野忠則博士，そして助教授佐藤文明博士に，深く感謝いたします．更に，本研究に関する助言を頂き，また研究報告の査読や指導を何度も丁寧にご指導頂きました工学部システム工学科教授浅井秀樹博士，そして助教授前田恭伸博士に感謝いたします．また，ゼミや論文のご指導を頂きました工学部電気･電子工学科教授下平美文博士，そして情報学部助教授伊東幸宏博士に併せて感謝いたします．

学会及び研究会等の場で，助言やコメントを頂きました東北大学工学部教授白鳥則郎博士，東京電気大学理工学部教授滝沢誠博士，同･教授小泉寿男博士，更に愛知県立大学教授井手口哲夫博士には，心から感謝いたします．

三菱電機㈱顧問大野栄一博士，開発本部長野間口有博士，情報技術総合研究所所長片木孝至博士，そして同･副所長市川照久殿には，本研究の場を与えてくれましたことを深く感謝いたします．また，日常業務ともどもご指導を頂きました同･開発戦略グループリーダ立木武彦殿，及び同情報処理部門統括岩瀬正殿には，さまざまなご支援と心配りを頂きました．技術研修所の春原猛殿からは，ソフトウェア工学の分野の大先輩として色々ご指導を頂きました．

更に，同･情報処理部門の萩原正敏殿，金枝上敦殿，鈴木由美子殿，上野浩一郎殿，渡部修介殿，そして大島利浩殿，には本研究の中でさまざまな議論や検討に加わっていただき貴重な意見を頂きました．特に渡部修介殿には，実装や評価におけるさまざまな作業で，快く協力していた頂きました．また，同･情報処理部門の宮崎一哉殿には，ユーザインタフェースに関わる領域やグループウェアの領域からの，貴重な意見を頂きました，深く感謝いたします．

最後に，このような勉学コースの入った人生を与えてくれた亡き父と，研究の進捗状況を日々気にしてくれました母に，心からお礼を申し上げます．

突然に大学での研究を始めることになり，家族との触れ合いがなされるべき時間を削ったにも関わらず，作業場所や時間を工面してくれた妻，娘，息子，等家族の皆に，改めて感謝いたします．

参考文献

[Abe92] 阿部，土田，坂下："非同期分散処理型議論支援インタフェース:ArgView "，情報処理学会グループウェア研究会報告 1-4,1992

[Ahuja86] Sudhir Ahuja, Nicholas Carriero, David Gelernter, "Linda and Friends," IEEE Computer, Vol.19, No.8, pp.26-34, 1986

[Akaishi98] 明石，村上，天野，奥乃：" MueLinda モデルと自己記述による実装"，コンピュータソフトウェア, Vol.16, No.1, 1998

[Aoyama90] M.Aoyama, "Distributed Concuurent Development of Software Systems", An Object-Oritented Process Model, Proc, IEEE COMPSAC'90, 1990

[Baba94] 馬場，小島，小澤："ニューラルネットの基礎と応用"，共立出版，1994

[Birrell84] A.Birrell, and B.Nelson: "Implementing Remote Procedure Calls", ACM Transaction on Computer System, Vol.2, No.1, 1984

[Black92] D.L.Black, et al.,"Microkernel Operating System Architecture", J.Inf.Process, Vol.14, No.4,1992

[Brothers90] L.Brothers,et al.,"ICICLE: Groupware For Code Inspection", CSCW'90 Proceedings, ACM, pp169-181, 1990

[Burgess90] K.C.Burgess Yakemoviec and E.Jeffrey Conklin, "Report on a Development Project Use of an Issue-Based Inforamtion System", CSCW'90, Proceedings, ACM, pp105-118, 1990

[Chamberlin81] D.C.Chamberlin, L.C.King, D.R.Slutz,m S.J.P.Todd, and B.W.Wade: "JANUS: An interactive system for document composition", Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices, Vol.16, 1981

[Chen76] P.Pin-Shan Chen: "The Entity-Relationship Model-Toward a Unified View of Data", ACM Transaction on Database Systems, Vol.1, No.1, 1976

[Cheriton88] D.R.Cheriton: "The V Distributed Operationg System", Communication ACM, Vol.31, No.3, 1988

[Conry86]S. E. Conry and V. R. Lesser, "Multistage negotiation in distributed planning," Technical Report COINS Tech. Report 86-67, Univ. of Massachusetts, 1986

[Conry89]S. E. Conry, R. A. Meyer, and R. P. Pope, "Reasoning about nonlocal impact of local decisions in distributed planning," M. N. Huhns and L. Gasser, Eds., Distributed Artificial Intelligence, Vol.2, Morgan Kaufmann, 1989

[Cowan95] D.D.Cowan, and C J.P.Lucena: "Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse", IEEE Transaction on Software Engineering, Vol.21, No.3, 1995

[Fayad97] Mohamed Fayad, and Douglas C. Schmidt, “Object-Oriented Application Frameworks”, Communication of the ACM, Vol.40, No.10,1997

[Fowler97] Fowler, M., “Analysis Patterns : Reusable Object Models”, Addison-Wsley Longman, Inc., Menlo Park, CA, 1997

[Fujisawa86] 藤澤，他："高度ファイリングの理念と要素技術―文書管理と知的ファイリングー"，情報処理学会研究会資料，86-JDP-7,1986

[Fujiwara97] Fujiwara, K., Saito, H., and Chusho, T., “Experiment in Application Framework for Distributed Systems on Multi-Organizational Office Network”, IPSJ, 1997

[GDiD93] ISO/IEC JTC1/SC18/WG5 Working Draft, “Information Processing - Text and Office Systems - Generic Content Notation and Transforms: Generic Data Interchange/Interface for Document (GDiD)”, 1993

[Harada90] 原田，浜田，渡辺，宮本："分散型環境 Kamui について"，情報処理学会記号処理研究会 57-5,1990

[Harada97] 原田，萬木，北畠，上原："ビジネス系システム開発におけるオブジェクト指向開発技法(3)―再利用性向上のための実装設計―"，情報処理学会第 54 回全国大会,1-415,1997

[Heninger78] Heninger, K.L., et al., “Software Requirements for the A-7E Aircraft”, NRL Memorandum Report 3876, Naval Research Laboratory, Washington D.C.,1978

[Hill96] D.R.C. Hill, “Object-Oriented Analysis and Simulation”,Addson-Wesley Publishing, 1996

[Honiden94] 本位田，山城："オブジェクト指向分析設計"，情報処理学会，Vol.35,No.5, 1994

[Horikawa90] 堀川，伊藤，高野："ソフトウェア文書のためのエディタ - Spec"，情報処理学会論文誌，Vol.31, No.2, 1990

[Humminen86] H.Humminen and R.Solonen, "OAGES: Intelligent Forms, Intelligent Mail and Distribution", Proc. of the IFIP WG8.4 Working Conference, 1986

[Iijima94]飯島, 永田："実世界と形式的記述の接点としてのオブジェクト指向モデル", 情報処理学会誌，Vol.35, No.5, pp412-422,1994

[Ishii93]石井："グループウェアの実現に向けて―リアルタイムグループウェアのデザ

イン", 情報処理, Vol.34,No.8,1993

[ISiS92] Isis Distributed Toolkit Version 3.0 "User Guide and Reference Manual", Isis Distributed Systems Inc., 1992

[Itou93]伊藤："自律分散システムの研究の課題と将来", 計測と制御, 第３２巻, 第１０号, 1993

[Iwahashi98]岩橋："オブジェクト指向で制御システムを実現する手法", Interface, pp.93-148, July, 1998 CQ 出版社

[Jiritsu94] 特集:”自律分散の新たなる展開", 計測自動制御学会誌, Vol.32, No.10, 1994

[Kanaegami97] 金枝上, 中島, 原田, 前中, 大島："分散サービスの柔軟な連携を実現するサービスプラグインシステム(2)―アプリケーション記述方式―", 情報処理学会第 54 回全国大会,1-355,1997

[Katoh94] 加藤："オブジェクト指向分析・設計と従来の方法論との比較", 情報処理学会誌, Vol.35, No.5, pp423-431,1994

[Keith91]Keith Marzullo, Robert Cooper, Mark D. Wood, and Kenneth P.Nirman, "Tools for Distributed Application Management," IEEE Computer, Vol.24, No.8, 1991

[Kenneth] Kenneth P. Berman: "The Process Group Approach to Reliable Distributed Computing" TR91-1216, Cornell University.

[Kimura84] G.D.Kimura, and A.C.Shaw: "The Structure of Abstarct Document Objects", ACM, SIGPLAN, 1984

[Kinukawa86] 絹川："表階層モデルに基づく自然言語インタフェース処理方式", 情報処理学会論文誌, Vol.27, No.5, 1986

[Kishimoto91] 岸本, ”分散協調マルチエージェント型知的通信モデル”, 電子情報通信学会論文集, Vol.J74-B-I, No.11, 1991

[Kitamori93] 北森："自律分散システムと自己組織化―事例からの考察", 計測と制御, 第３２巻, 第１０号, 1993

[Knuth79] D.E.Knuth: "Mathematical typography", TeX and METAFONT: New Directions in Typesetting, Digital Press and the American Mathematical Society, 1979

[Koegel 93] J. F. Koegel, L. W. Rutledge, et al., "HyOctane: A HyTime Engine for an MMIS(Hypermedia)," ACM Multimedia 93, 1993

[Kondo92] T.Kondo, M.Inoue, K.Nakai, K.Doi and Y.Suzuki: "Application of Autonomous Decentralized Systems to the Steel Production Computer Control", 3rd IEEE workshop on Future Trends of Distributed Computing Systems, 1992

[Kozuka93] 小塚, 佐藤, 宮崎, 福岡: "分散協調環境 Noah(Network oriented applications harmony)の提案", 情報処理学会研究会, マルチメディア通信と分散処理 59-11, 1993

[Kozuka94] 小塚, 佐藤, 宮崎, 福岡: "自律協調分散システム Noah におけるエージェントの制御機構", 情報処理学会第４８回全国大会, 6F-6, 1994

[Krause93] F. L. Krause, F Kimura, T. Kjellberg and S. C.-Y. Lu: "Product Modeling", Annals of CIRP, Vol.42, No.2, 1993

[Kuwabara89] K. Kuwabara and V. R. Lesser, "Extended protocol for multistage negotiation," Proc. 9th Workshop on Distributed Artificial Intelligence, 1989

[Lesser88] Lesser,V. and Corkill,D.: "Distributed Problem Solving", Encyclopedia of Artificial Intelligence, S.C. ed., pp.245-251,John Wiley & Sons ,1988

[Lichter94] H. Lichter, et. Al., “Prototyping in Industrial Software Projects - Bridging the Gap Between Theory and Practice”, IEEE Transaction on Software Engineering, Vol.20, No.11, 1994

[Maekawa91] 前川, 所, 清水　編："分散オペレーティングシステム　UNIX の次にくるもの", 共立出版, 1991

[Maenaka97] 前中, 中島, 金枝上, 大島, 原田："分散サービスの柔軟な連携を実現するサービスプラグインシステム(1)－全体構成－", 情報処理学会第 54 回全国大会,1-353,1997

[Makabe96] 真壁, 多田, 樋口, 藤井, ：分散システムにおける因果関係を保存するメッセージ配信プロトコル, 情報処理学会研究会報告 96-DPS-74-38,1996

[MfWC94] Workflow Management Coaliation, "Reference Model," WFMC-TC-1003, 1994

[Miyazaki89] 宮崎, 福岡, 辻, 坂下:"プロダクションシステムに基づいたユーザインタフェース管理システム", 情報処理学会研究会報告, 文書処理とヒューマンインタフェース 23-1, 1989

[Miyazaki94] 宮崎, 佐藤, 小塚, 福岡: 自律協調分散システム Noah における協調機構, 情報処理学会第４８回全国大会, 6F-8, 1994

[Monroe97] Robert T. Monroe, Andrew Kompanek, Ralph Melton, and David Garlan, “Architectural Styles, Design Pattrens, and Objects”, IEEE Software, Vol.14, No.1, 1997

[Mori93] K.Mori: "Autonomous Decentralized Systems: Concept, Data, Field Architecture and Trends", Proc. of 3rd International Symposium on Autonomous Decentralized Systems (ISAD93), pp28-29, 1993

[Morris86] J.H.Morris, M.Satyanarayanan, M.H.Conner, J.H.Howard, D.S.H.Rosenthal, and F.D.Smith: "Andrew: A Distributed personal computing environment", Communications

ACM, 1986

[Mullender90] S.J.Mullender, and G. van Rossum: "Amoeba: A Distributed Operating System for the 1990s", IEEE Computer, Vol.23, No.5, 1990

[Nakashima91]中島，他："協調アーキテクチャ関連研究の現状"，電子技術総合研究所調査報告，第 221 号，1991

[Nomiyama86] 野見山，他："日本語文書の構造的作成支援環境構築の試み"，情報処理学会，日本語文書の入力と編集シンポジューム，セクション２，1985

[ODA86] ISO 8613 Information Processing - Text and Office Systems - Office Document Architecture(ODA) and Interchange Format, 1986

[Ohshima97] 大島，前中，中島，金枝上，原田："分散サービスの柔軟な連携を実現するサービスプラグインシステム(3)―アプリケーション制御方式―"，情報処理学会第 54 回全国大会,1-357,1997

[Ossanna76] J.F.Ossanna: "NROFF/TROFF user's manual", Computer Science Technocal Report 54, Bell Laboratories, 1976

[Pfaff86] G.E.Pfaff, “User Interface Management System”, Springer-Verlag, 1986

[Price93] R. Price, "MHEG: An Introduction to the Future International Standard for Hypermedia Object Interchange(Hypermedia)," ACM Multimedia 93, 1993

[Rich96] C.Rich, C.L.Sidner,: ”Adding a Collaborative Agent to Graphical User Interfaces”, *9th Annual Symposuim On User Interaface Software and Technology(UIST’96)*,1996.

[Robertson97] Paule Robertson, “Integrating Legacy Systems with Modern Corporate Applications”, COMMUNICATION of the ACM, Vol.40, No.5, 1997

[Rozier89] M.Rozier, V.Abrossimov, F.Armand, I.Boule, M.Gien, M.Guillent, F.Herrman, C.Kaiser, S.Langlois, P.Leonard and W.Neuhauser: "CHORUS Distributed Operating System", Chorus Systems Technical Report CS/TR-88-7.8, 1989

[Rumbaugh92]羽生田栄一監訳, "オブジェクト指向方法論 OMT", 凸版, 1992

[Sakane95] 坂根茂幸，"分散協調ロボットシステム"，情報処理学会誌，Vol.36,No.10, 1995

[Sakashita83] Y. Sakashita, H. Ito, O. Watanabe, E. Yamazaki: "Document Processing System(MIDOP)," ICTP'83, 1983

[Sakahsita84] 坂下，土田："文書処理と編集処理とに関する検討"，情報処理学会日本語文書処理研究会 3-3，1984

[Sakashita93] 坂下："グループｳｪｱにおけるグループ活動モデルの概要"，情報処理学会

誌，Vol.34, No.8, pp.1037-1045, 1993

[Sakashita98] Y. Sakashita, K. Ohta, F. Sato, T. Mizuno, "On the Notion of Adaptability to the environments in Autonomous Mechanism, ICOIN12, " pp.158-161, 1998

[Satoh87] 佐藤, 絹川, 大町："オフィス文書の標準化と文書データベースの研究動向", 情報処理学会誌，Vol.28, No.6, 1987

[Satoh94] 佐藤, 小塚, 宮崎, 福岡: "自律協調分散システム Noah の通信機構の実現", 情報処理学会第４８回全国大会，6F-7, 1994

[Satoh95] F.Sato, H.Kozuka, K.Miyazaki and H.Fukuoka: "Noah: An Environment for Autonomous Systems," *Proc. of 3rd International Symposium on Autonomous Decentralized Systems(ISAD'95)*, 1995.

[SGML] ISO 8879 Information Processing - Text and Office Systems - Standard Generalized Markup Language(SGML)

[Shu82] N.C.Shu, V.Y.Lum, F.C.Tung, and C.L.Chang, "Specification of Forms Proceeding and Business for Office Automation", IEEE Trans. On Software Engineering, Vo.SE-8,No.5,1982

[Smith80] R.G.Smith: "The contract net protocol: high level communication and control in a distributed problem solver" ,IEEE Trans. on Computer, Vol.C-29,No.12, pp1104-1113, 1980

[Sparks] S. Sparks, K.Benner, and C. Faris, “Managing object-oriented framework reuse”, IEEE Computer, Vol.29, No.9

[Suda93]須田："従来型工学システムと自律分散システム"，計測と制御，第３２巻，第１０号，1993

[Sugawara94] K.Sugawara, T.Kinoshita, and N.Shiratori: "A Consideration on Flexible Systems", Technical Report of IEICE, AI93-84, 1994

[Suzuki97] 鈴木，原田，北畠，上原，萩原："ビジネス系システム開発におけるオブジェクト指向開発技法(1)―分析から論理３階層設計への流れ―"，情報処理学会第 54 回全国大会,1-411,1997

[Tarumi97] 垂水，喜田，柳生，吉府："ワークウェブシステム：ワークフロー動的再計画の方式"，情報処理学会研究会報告 97-GW-21-17,1997

[Thacker79] C.P.Thacker, E.M.McCreight, B.W.Lampson, R.F.Sproull, and D.R.Boggs: "Alto: a peronal computer", Technical report, CSL-79-11, Xerox Palo Alto Research Center, 1979

[Tokumoto97] 徳本，原田，鈴木，萩原："ビジネス系システム開発におけるオブジェ

クト指向開発技法(4)―適用例による再利用性の評価―", 情報処理学会第 54 回全国大会,1-417,1997

[Tsutsumi87] K.Tsutsumi, and H.Matsumoto,"Neural Computation and Learning Strategy for Manipulator Position Control", *Proc. of IEEE Int. Conf. on Neural Networks(ICNN-87)*, Vol.IV, 1987

[Uta88] Uta Pankoke-Babatz: "Computer Based Group Communication: The AMIGO Activity Model", John Wiley & Sons ,1988

[VanHilst96] VanHilst, M., and Notkin, D., "Using Role Components to Implement Collaboration Based Design", Proceedings of OOPSLA'96, ACM SIGPLAN Notices, Vol.31, No.10, ACM SIGPLAN, 1996

[Yamashita90] 山下, 沢田, 鯵坂, 松本:"CASE 環境における協調活動支援ツールの試作", 日本ソフトウェア科学会, 第7回研究会資料, 分散環境におけるシステムと応用一般, 1990

[Yano92] 矢野, 武宮, 布川, 野口："自律的な協調処理を行う分散型計算モデル Kemari", 情報処理学会論文誌, Vol.33,No.12,1992.

[Yonezawa86] 米澤："オブジェクト指向型プログラミングについて", コンピュータソフトウェア, Vol.1, No.1, 1986

[Yoshida90] 吉田, 楢崎, "場と一体化したプロセスの概念に基づく並列協調処理モデル Cellula", 情報処理学会論文誌, Vol.31, No.7, 1990

[Yoshida93] N.Yoshida: "A Modeling Framework for Group Behaviors and Its Application to Cooperative Problem Solving", Proc. of 3rd International Symposium on Autonomous Decentralized Systems (ISAD93), pp70-76, 1993

[Yurugi97] 萬木, 鈴木, 原田, 北畠, 上原："ビジネス系システム開発におけるオブジェクト指向開発技法(2)― 3 階層設計におけるＤＢ処理パターンの利用―", 情報処理学会第 54 回全国大会,1-413,1997

研究成果一覧

(1) 坂下，亀山，伊藤，渡辺："文節内文法解析における文節の語構成"，情報処理学会第２４回全国大会，1G-2, pp.975-976, 1982

(2) 坂下, 渡辺："高速描画におけるメモリ制御シミュレーション"，情報処理学会第25回全国大会, 2J-3, pp.167-168, 1982

(3) Y. Sakashita, H. Ito, O. Watanabe, E. Yamazaki: "Document Processing System(MIDOP), ICTP'83, " pp.331-336, 1983

(4) 清水, 福岡, 伊藤, 坂下："マルチウィンドウを用いた文書処理システム(I)ーマルチウィンドウ表示方式ー"，情報処理学会第26回全国大会, 4H-9, pp.1309-1310,1983

(5) 伊藤, 清水, 福岡, 坂下："マルチウィンドウを用いた文書処理システム(II)ーマルチウィンドウの管理方式ー"，情報処理学会第26回全国大会, 4H-10, pp.1311-1312, 1983

(6) 坂下，宮崎，渡辺："カナ文字盤の文字配列"，情報処理学会研究会報告，日本文入力方式 18-2，1984

(7) 坂下，土田："文書処理と編集処理とに関する検討"，情報処理学会研究会報告，日本語文書処理 3-3，1984

(8) 渡辺，水野，坂下，風間，伊藤："高機能ワークステーション"，三菱電機技報，Vol.59, No.12, pp.841-845, 1985

(9) 坂下，土田，田中："マルチメディア文書処理システム"，情報処理学会第３２回全国大会，2K-1, pp.1809-1810, 1985

(10) 坂下，堀川，土田，田中："統合化ソフトウェア仕様書エディタ(2)"，情報処理学会第３４回全国大会，4T-2, pp.1127-1128, 1986

(11) 坂下，福岡，宮崎，辻："ユーザインタフェースシステム"，情報処理学会第３５回全国大会，4W-6, pp.937-938, 1987

(12) 高野, 水野, 坂下, 北畠："分散型開発システムで仕様書作成からプログラム生成までを支援する日経エレクトロニクス", No.425, pp.179-194, 1987

(13) 春原, 高野, 渡辺, 水野, 坂下, 藤掛："分散型ソフトウェア開発システム(1)ー基本思想"，情報処理学会第 35 回全国大会， pp.1107-1108, 1987

(14) 福岡, 宮崎, 辻, 坂下："ユーザインタフェース設計ツールー画面設計支援ー"，情報処理学会研究会報告, 文書処理とヒューマンインタフェース, 17-3, 1988

(15) 坂下："ワークステーションにおける文書処理"，情報処理学会，マルチメディア

通信と分散処理シンポジューム，pp.91-100, 1988

(16) 宮崎, 福岡, 辻, 坂下："プロダクション・システムに基づいたユーザインタフェース管理システム"，情報処理学会研究会報告, 文書処理とヒューマンインタフェース, 23-1, 1989

(17) 坂下, 宮崎："ハイパーメディアの情報処理システム", 電気学会誌, Vol.111, No.10, pp.817-819, 1991

(18) 坂下："文書スタイルとマルチメディアへの拡張　－　マルチメディア／ハイパーメディアへの拡張　－　ＯＤＡ拡張"，　画像電子学会誌，Vol.20, No.6, pp586-594, 1991

(19) 阿部，土田，坂下："非同期分散処理型議論支援インタフェース:ArgView "，情報処理学会グループウェア研究会報告 1-4,1992

(20) 坂下："分散開発環境の事例と今後の展望"，情報処理学会誌，Vol.33, No.1, 1992

(21) 坂下："グループウェアにおけるグループ活動モデルの概要"，情報処理学会誌, Vol.34, No.8, pp.32-39, 1993

(22) 坂下，井手口，滝沢，水野："分散システムシリーズ　分散システム入門"，近代科学社，1993

(23) 清水, 福岡, 坂下："マルチウィンドウ表示方式の評価(I) "，情報処理学会第２８回全国大会，６D-5, pp.247-248, 1994

(24) 福岡, 清水, 坂下："マルチウィンドウ表示方式の評価(II) "，情報処理学会第２８回全国大会，６D-6, pp.249-250, 1994

(25) 坂下, 太田, 水野："予約アクセスによる時間同期処理の試み"，情報処理学会研究会報告, マルチメディア通信と分散処理 70-22, 1995

(26) K.Ohta, Y.Sakashita, T.Mizuno:" A Proposal of Network Protocol with Performance for Multimedia Communication System", Second Joint Workshop on Multimedia Communication, pp.411-418, 1995

(27) 坂下, 山上, 大高, 冨樫, 藤本, 岡村:”WindowsNT 3.5 によるクライアント／サーバ・システム構築手法”，(株)ソフト・リサーチ・センター, 1995

(28) 坂下，太田，佐藤，水野：制御システムの自律協調機構，静岡大学大学院電子科学研究科研究報告第 18 号, pp135-140, 1996

(29) K.Ohta, Y.Sakashita, T.Mizuno:" Time Critical Oriented Network Protocol", Proceedings of the 10th International Conference on Information Networking (ICOIN-10), pp.268-274, 1996

(30) 坂下, 池田, 太田, 水野："共同作業と協調作業への考察"，情報処理学会，マルチメデ

ィア通信と分散処理ワークショップ, pp.373-378, 1997

(31) Y.SAKASHITA, K.OHTA, F.SATO, and T.MIZUNO, ”On the Notion of Propagation of Control in Collaborative and Autonomous Mechanism” *Proc. of ISCOM97*, pp.158-161, 1997

(32) 坂下，金枝上，佐藤，水野："フレームワークに基づく協調的アプリケーション・システムの構築", 静岡大学大学院電子科学研究科研究報告第 19 号, pp57 - 63, 1998

(33) Y. Sakashita, K. Ohta, F. Sato, T. Mizuno, "On the Notion of Adaptability to the environments in Autonomous Mechanism," ICOIN12, pp.705-708, 1998

(34) 坂下，金枝上，鈴木，上野： 分散システムにおける協調的業務とサービス， 情報処理学会，マルチメディア，分散，協調とモーバイル(DICOMO) シンポジューム，pp.227-233, 1998

(35) 坂下，井手口，佐藤，水野："役割に基づくアプリケーションの振舞い制御"，電子情報通信学会論文誌, Vol.J-B-I, 1999