TECHNOLOGY REVIEW

UNISYS 技報

通巻 45

1995年5月発刊 Vol. 15 No. 1

特集：エンドユーザ・コンピューティング

日本ユニシス

ISSN 0914-9996

厳しい経済環境を背景として企業では利用部門によるEUCが関心事となっている．EUC実現には，基盤システム環境の実現，EUC支援要員やユーザの育成といった人的環境整備が必要であることから成果をあげられない場合も多い．松木規子は**EUC環境構築に向けて——MAPPER適用の経験から新たな手順を考える**の中で，EUC実現のための重要かつ現在欠けていると思われる手順を中心に環境，技術を明らかにしている．

ワープロ・表計算ソフトウェアなどのOAツールの普及と共に，OAツールで作成データの部門内での共有・有効利用，APシステムで蓄積データのOAツールへの取込み等の要求が高まっている．Camelは前者の，Latchは後者の要求に応える製品である．藤井昭彦・伊東春正の**OA支援ソフトウェアCamel/Latchの概要**は，Camel/Latchのソフトウェアと機能を概説し，次に特徴的な機能についてその要求の背景，狙い，実現方法を含め紹介している．

オフィスの中で個々の役割が決められているオフィス環境では，ワークフローは生産性の高いオフィス実現のためのツールである．オフィスでワークフローソフトウェアが使われるためには，定型的業務をこなすだけでなく，通常人間が行っている不規則な流れを制御しなければならない．戸叶孝一が解説している**スタッフウェア**は，そのような過酷な要求に耐えられるソフトウェアである．

門本光央・石倉哲善・林健一は**クライアント/サーバによるA社MP事業本部システムの再構築**の中で，ミートパッカー事業本部情報検索システム・窓口支援システムのA19・US120・AS1000/PCによるクライアント・サーバシステムへの再構築について，開発の経緯，Tippler・UNIshuttle・Linc採用の理由，これらを使用した開発のポイント等を述べている．

Windows*登場以来，PCはクライアント端末選択肢の最右翼の存在である．ビジュアル開発環境実現のツールとしてVisual Basic V2.0*（VB）が最もメジャな地位を占め，さらにExcel V5.0*とともにVBと同じ言語仕様のVBAが登場し，この二つの言語手段を用いたプログラム開発が多くなっている．翠秀幸は**Windows上でのアプリケーション開発の実際**の中で，Windows上の複数アプリケーション間の制御やデータ交換の方法としてDDEプロセス間通信を取り上げて解説し，VBとExcel間のデータ交換をDDEを使用したAPを例に紹介している．

* Windows, Visual Basic, Excelは米国Microsoft社の商標である．

病院の情報システムは医療オーダプロセシングシステムを核とした総合情報システム構築の時代に入った．診療行為情報は医師への診療支援機能として有効活用の期待が高い．ここで求められるのは，オーダ情報をベースに特定の病名・薬剤・処置・患者個人に着目しその推移を分析しうる情報である．日本海病院では医師が自由に検索・加工できる病歴検索システムを開発した．森良行の**病院におけるEUC——診療支援における病歴検索システム**は，システムの役割，データベース，機能と医師の利用を意識した利用者インタフェース，システムの今後について説明している．

最近エンドユーザ・コンピューティングを実現する環境として，分散データベースシステムが注目を集めている．山田和司・今泉正雄は**レプリケーション機能を用いた分散システム構築**の中で，分散データベースシステムを作成する場合のシステム構築方法として，レプリケーション機能を用いることを提案している．

90年代のエンドユーザ・コンピューティングの中心はワークステーションを用いたソフトウェアツール環境である．とくにデータベースとリポジトリに基づいて統合されたソフトウェアツール環境が必要になる．阪口喜好の**エンドユーザ・コンピューティング環境におけるリポジトリ**は，このような環境で果たす役割が極めて重要なリポジトリの概要と実装における日本ユニシスの考え方を紹介している．

特集 「エンドユーザ・コンピューティング」の発刊によせて

小林 允

End User Computing (EUC) という言葉はすっかり社会に定着しマスメディア等でもごく当たり前に使われるようになっている．しかしながら「EUCとは何か？」という素朴な質問に対して簡潔に答えるのは難しい．各々の識者，マスメディアの編集者などが，様々の切り口でその性質や属性，特徴をいろいろに述べ，その結果としてEUCというものが漠然とした形で理解されているというのが現状であるように思われる．

このような状況認識に基づき，EUC特集号を編集するにあたって以下のような姿勢で臨んだ．すなわち：

1) EUCという言葉をあらためて定義することをせずに使用し，具体例を並べることによってEUCがどのようなものであるかという説明に代える．
2) 世の中で一般的に理解されているEUCに近い領域における当社の商品，事例を見直し，技術的に整理して論文とすることによってその成果を公表し，似たような局面におられる方々への貢献とする．
3) 多くのお客様の業務システムの開発・運用・支援，および商品の開発・改良・保守などを通じて培ってきた当社の最大の財産である技術を再確認し，それらの技術をEUCを実現するための基盤技術という観点で整理する．
4) これらを論文集としてまとめることによって，EUCに対する当社の考え方を全体として理解していただく．

EUCを簡潔な言葉で定義することは難しいが，その特徴のいくつかを例示的に述べることはできるであろう．例えばEUCによって意思決定の方法，メカニズムがエンドユーザ(業務担当者)に与えられることになり，その結果として企業の組織/管理体系の変化をもたらすことになるかもしれない．また，企業間の競争が激しくなり業務担当者が情報システム部門から与えられる情報を利用するだけでなく，自らデータを加工し情報を作り出し利用することが必要となり，そのための手段をEUCと呼んでいると言っても良いかもしれない．そのためには従来情報システム部門がデータを加工して情報の形で業務担当者に提供していたのに対して，なまのデータとそのデータを加工するための手段を提供しなければならなくなる．言い換えれば，従来業務担当者が利用できないと考えていたデータが個人利用できるようになるということである．

一方，このような環境のもとで情報システム部門の役割は次のように考えられる．

まず第一に情報基盤の整備である．これは専門家でない人でも矛盾無く使えるように情報処理技術を組み立てると共に，そこで必要となるツール（情報処理機器，ソフトウェア）を調整し提供することである．二番目は現有の情報を活用することであり，関連する部門に現有の情

報を紹介し必要に応じてそれらの情報を配布することである．三番目は業務担当者に提供するためのデータベースの整備，再構築である．情報，データを経営資源の一つとして管理することが必要であり，データベース，リポジトリの技術が使われる．

この特集号を編集するに当たってEUCを実現するための基盤技術を，①情報共有・活用技術，②オープン技術，③分散データベース技術，④ネットワーク技術，⑤運用管理技術の5項目に整理した．

情報共有・活用技術としてはEUC環境の構築・実行で有効な当社の商品とその適用事例についてまとめた．松木の「EUC環境構築に向けて――MAPPER適用の経験から新たな手順を考える」では，EUCの実現ははあくまでもエンドユーザが主体の活動であることという主張を中心として議論を進める．藤井・伊東の「OA支援ソフトウェアCamel/Latchの概要」では，オフィス連携機能/データ連携機能について述べこれらの機能を実現するソフトウェア群を紹介する．戸叶の「スタッフウェア」では，オフィスの生産性を向上させるためのツールとしてのワークフローを紹介し，その機能を実現するスタッフウェアについて述べ，さらに外部オブジェクトとのデータの相互交換の機能についても述べる．門本・石倉・林の「クライアント/サーバによるA社MP事業本部システムの再構築」では，エンドユーザが一番使いやすい形で業務を支援し，また業務の充実をはかれる低コストのシステムを構築することを狙いとした事例を報告する．情報処理の基盤ソフトウェアとしてTIPPLER, UNIshuttle，ハードウェアとしてEWS，PCを採用し，データはメインフレーム(A 19)で一元管理するシステムとなっている．

オープン技術としては，種々のベンダの商品を組み合わせて実現されたEUC環境を紹介する．翠の「Windows*上でのアプリケーション開発の実際」では，Windows上の複数アプリケーション間で制御やデータを相互にやり取りする方法として，DDEプロセス間通信を取り上げ詳細に解説する．さらに，Microsoft Visual Basic*で作られたアプリケーションとExcel*で作られたアプリケーションとの間でDDEを使ってデータのやり取りを行い，動作する適用例を紹介する．森の「病院におけるEUC――診療支援における病歴検索システム」では，医療オーダリング・システムを核とした総合情報システムの構築をめざし，そのサブシステムとして病歴検索システムを開発した事例を紹介する．このシステムは意味データベースと汎用検索システムによって構築され，エンドユーザである医師はPCを使って自由にオーダ・データベースの検索・加工を行うことができる．

③～⑤は一般的な意味でも基盤技術であるが，本特集ではEUCを実現するためにどのような役割を果たしているか，EUC環境の中でどのように使われているかという視点でまとめた．

分散データベース技術ではエンドユーザにデータを公開するために必要な機能とそのための実装について述べる．山田・今泉の「レプリケーション機能を用いた分散システム構築」では，まず分散システムの構築という観点から2フェーズコミットメントとレプリケーション機能を検討する．さらにUNIX**上のいくつかのDBMSについてそれぞれのレプリケーション機能の比較を行い，最後にいくつかの事例を述べる．阪口の「エンドユーザ・コンピューティング環境におけるリポジトリ」は，EUC環境ではデータベースとリポジトリに基づいた統合されたソフトウェアツール環境が必要であることを主張し，そのためにリポジトリの果たす役割の重要性について述べる．

ネットワーク技術はデータの発生・蓄積・保守を行う部署とデータを自分の手もとに持ち込み，自分の環境で利用しようとするエンドユーザとの間の時間的・空間的隙間を埋める技術と

して捉えた．伊奈の「人事システム再構築における UNIshuttle の適用事例」は，利用者自身によるデータ活用を目的とし，UNIshuttle を用いてホスト，部門サーバ，利用者端末の 3 階層の構成で人事システムを再構築した事例を報告している．本田の「UNIshuttle の概要——分散処理システム構築向けの基盤ソフトウェア」は，UNIshuttle 開発の進め方，実現案，基本仕様，実現状況について述べ，オープンな分散処理システムの構築において UNIshuttle を用いることによって得られる利益——AP の可搬性，論理的なアドレスの採用による物理的な構成からの独立性，運用管理サービスなど——について言及する．佐藤・久米の「OLTP——基幹システムと EUC の融合」では，PC の構成するローカル・ネットワークとホストコンピュータ上の基幹業務を連動し，情報の加工処理を個人のニーズにあわせて自由に行えるシステムの必要性が高まっていることを課題として，当社の TransIT Open/OLTP を使用したメインフレーム上の情報系システムとエンドユーザの操作する UNIX または PC との連動について考察する．

運用管理技術は，数多くの独立した EUC 環境に対する資源の分散と管理のための技術であるが，ここでは特にネットワーク管理に着目し，水野の「ネットワーク管理」を収めた．この論文では，マルチベンダの多様な情報機器や通信機器がエンドユーザ主導で接続/使用される環境となって来ていること，そのような環境でネットワークを管理するためには，多くのベンダに支持された標準的な管理フレームに基づく統一的かつ拡張性のある手法を採用することが必要であることを主張する．さらにこのような手法として普及している「インタネット標準フレームワーク（SNMP）」とその実装例である「CNMS インタネット・マネージャ」を紹介する．

この様な構成でこの特集号を編集したが，冒頭に述べたように EUC の捉え方に必ずしも客観的な基準がなく，さらにいくつかの視点からの論文を収録するべきであったかもしれない．皆様のご批判を仰ぎたい．些かでも皆様のお役に立つことを願うとともに，忌憚の無いご意見をお寄せ頂ければ幸いである．

（システム企画部　主席部長）

---

* Microsoft は米国 Microsoft 社の登録商標，Windows, Visual Basic, Excel は同社の商標である．

** UNIX は，X/Open カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標である．

EUC環境構築に向けて
——MAPPER適用の経験から新たな手順を考える

Toward the Creation of the EUC Environment
——A New Methodology Based on Experience with the Use of MAPPER

松木規子

要約 厳しい経済環境を背景として，多くの企業では利用部門による情報活用（EUC）が大きな関心事となっている．しかし，EUC実現には，基盤となるシステム環境の構築や，EUC支援要員やユーザの育成といった人的環境整備が必要であることから，なかなか成果をあげるに至らない場合も多い．

ここでは，MAPPERによるEUC実現を目指した経験から，以下のことを明らかにしている．

- MAPPERを使ってEUC環境を効果的に構築することができる．
- EUCは実際に利用すればするほど様々な問題が発生する．
- 多くの場合，利用開始後の問題は利用部門の中に不満として蓄積されるだけで，従来のシステム構築の手順では把握できない．
- 発生する問題は，環境に関するもの，機能に関するもの，手順に関するものに分類でき，それぞれに解決策が存在する．

この中で最も重要であり，かつ現在欠けているものがEUC実現のための手順である．

EUC実現手順は，従来の情報システム構築手順と以下の点で異なるものである．

- あくまでもユーザが活動主体であること
- 何もかも一度に実現するのではなく，その時点で明らかになっている問題にだけ対処すること
- その業務が存在する限り繰り返し実施される日常業務に組み込まれるものであること

このような手順を確立することで，今後，より本質的なEUCの実現が可能になると思われる．

Abstract Reflecting Japan's lingering economic slump, end user computing (EUC) has been a great concern for plenty of businesses these days. In many cases, however, EUC implementation has not been successful because of the need to build infrastructure systems and to establish an effective human environment which involves the training and education of EUC support people and end users themselves.

The author's experience in pursuit of MAPPER-based EUC has revealed the following:

- MAPPER can help a great deal create the EUC environment.
- The more EUC is used, the more new problems there are.
- After the start of operation, problems, for the most part, are accumulated and buried in end users as their complaints, and they are not made clear to EUC developers if conventional system creation methods are adopted.
- Those problems are classified into three categories, each related to environment, functionality and methodology. There are solutions to each of them.

What is particularly important but not available right now is effective methodology for EUC. It differs, as follows, from the methods for building traditional information systems:

- End users are always central players.
- The wisest approach is to handle only the issues that are apparent at that time, instead of dealing with all of the requirements at a time.
- As long as EUC applications exist, they can be incorporated into daily routine data processing.

The establishment of this methodology would lead to a birth of more practical EUC systems in the future.

1. はじめに

1968年米国ローズビルの工場で誕生したMAPPERは，その後四半世紀にわたりその時々の技術動向やニーズの変化に対応し，進化し続けてきた．

そして，政治，経済のパラダイムシフトに伴い，情報処理のあり方も大きな転換点にさしかかっている現在，MAPPERはUNIX*，PCを中心にオープンMAPPERとして一段と機能強化されている．

もともとユーザ**指向の4GLといわれながら，システムを「作る人」と「使う人」がはっきり切り分けられていた時代には開発言語としての側面が強調され，業務システム開発のツールとして使用されていたが，さまざまな環境変化の中でEUCの時代を迎えようとしている今，MAPPERはプラットフォームを問わず，定型・非定型全般の情報活用業務を支援するツールとして，位置付けられる．

ここでは，MAPPERをEUC環境構築・実行ツールとして適用している事例を見ながら，このようなMAPPER適用に必要な手順，技術，環境を明らかにしていく．

2. EUC環境としてのMAPPER適用

EUC環境とは，ユーザ自身による自由な情報活用を実現するための基盤となるハードウェアおよびソフトウェア，ツールを意味する．

EUC環境を構成する機能要素には，データベース，データ検索・抽出機能，情報加工機能（定型，非定型），情報の表現・伝達機能などがある（図1参照）．広い意味で

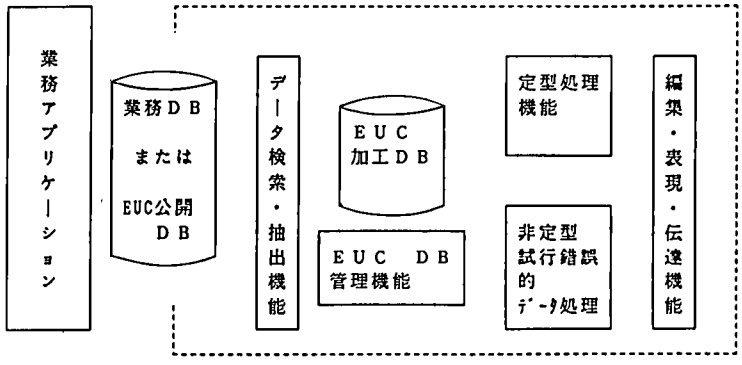


図1　EUC機能モデル

* UNIXはX/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標である．

** 本稿では，とくに断りのない限り利用部門のいわゆるエンドユーザを“ユーザ”とのみ記述する．

は，EUC で利用されるデータを作り出す基幹の業務システムも構成要素の一つといえる．EUC 実現環境の一般的な考え方，各機能の内容については，「技報」38 号（開発支援環境特集号）を参照されたい．

EUC 環境では，これらの機能要素が，最適のプラットフォーム（ハードウェア）に展開されることになる．

しかし，実際の適用には，これらの環境整備に加えて EUC 支援要員の育成やユーザの教育といった人的環境整備を欠くことはできない．

本章では，あるユーザにおける EUC 実現の経緯を，EUC 実利用開始までの手順を追って紹介する．

2.1 A 社人事部門における EUC への取り組み

A 社人事部門では，平成 3 年から 5 年にかけて，人事システム全般の再構築が実施された．

従来，人事部門では業務を遂行するための情報システムがホスト，PC に混在しており，手順が複雑化し作業効率を悪化させていた．また，従業員の増加や，これにともなう人材の育成・適正処遇実現のためには，従来にもまして人事業務の合理化・効率化が望まれた．さらに，従業員の福利厚生の推進や健康管理の徹底のため，人事情報のより有効な活用が必要となってきていた．このような問題を解決するため，在来業務の見直しも含め，採用から退職までのトータル・システムの再構築に踏み切ったわけであるが，このとき以下のような基本方針が設定された．

人事情報システム全体を，以下の二つの要素に分割して考える．

1) 基幹となる人事業務システム（採用，異動，昇格・昇進，給与計算，社会保険，住宅融資など）
2) 人事情報の有効活用（EUC）

人事業務システムの開発，EUC の基盤として MAPPER を利用する．

全体のスケジュール，予算の関係から開発生産性の高さが求められたこと，A 社人事部門では，従来より MAPPER による業務システムを利用しており，さらにこのデータをユーザ自身が会話機能で加工していたことから，既得の情報技術の継承，発展は妥当なものと考えられた．

まず，人事業務システムの設計・開発が開始され，その後 EUC 環境の実現方法が検討された．EUC 環境に関しては，日本ユニシス（以下，当社）より基本的なモデルを提示し，これを客先要件にすり合わせ実現することとなった．

2.2 EUC 環境概要

A 社人事部門向けに構築された EUC 環境の概要は以下の通りである（図 2 参照）．

まず，EUC で利用するデータであるが，これは MAPPER レポートとして構築されている．しかし，このレポートの主たる利用者は業務システムであり，業務システムに適した形に，逆にいえばユーザには使いづらい形に設計されている．したがって，これらのレポートを直接ユーザが使用することは禁止し，ユーザが利用しやすい形式に抽出・再編集するツールが提供された．このツールは，業務で使用するレポートの構造を知らなくても，画面から各種のパラメタを指定することで，必要な項目を収集し新たなレポートを作成するものである．

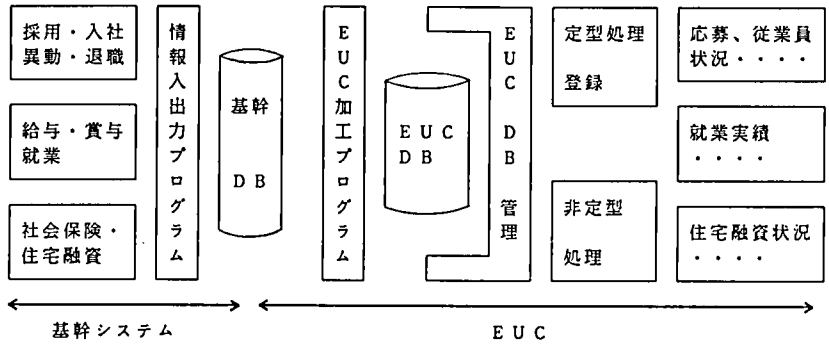


図 2　A社EUC環境概要

抽出・再編集された新たなレポートはユーザの管理下に置かれる．これらのEUC用レポートは，定常的に利用される共有レポートと，共有レポートでは不足する情報を必要に応じて抽出したものや共有レポートを加工した結果などの個別レポートの2種に分類される．共有レポートは，日次の業後処理の中で自動的に最新情報に置き換えられる．

MAPPERに展開されたEUCレポートは，複数の使用者による同時利用を可能とする．また，業務データベースから抽出された一次情報だけでなく，二次加工，三次加工の結果も他者と共有することができる．人事業務における情報活用は，単なる個人の情報利用ではなく，チームやグループといった組織によって遂行される業務の一部を形成するものである．したがって，一次情報のみならず，その加工結果も共有されることが必要になる．

すべてのEUC用レポートには使用期限が設定され（最長保存期間は1年），期限を過ぎたものは自動的に削除されるようになっている．これは，従来データをユーザに解放すると量的・質的管理（いつ，誰が作ったか，使っているのか，必要か不要か，といった判断）が困難になると危惧されていたことに対する一つの解決策である．

これらのレポートの加工は基本的にはMAPPERの会話機能を使用するが，会話機能だけでは実現困難な処理，あるいは操作が煩雑な処理を実現するため，いくつかのユーティリティ（部品ラン）が提供されている．

繰り返しのデータ処理に対応するためには，会話操作を自動的にプログラム化する自動ラン生成機能がある．自動生成されたランもデータ・レポートと同様に使用期限で管理される．また，自動生成されたランは，その利用状況をチェックできるようになっている．

これらの機能は総称してEUCツールと呼ばれる．

ハードウェア環境は従来から使用しているもので，EUCのための新たな投資は行っていない．

2.3 実 現 手 順

EUC環境というのは，ユーザが作業しやすいように機能を配置した枠組みのことである．ユーザの利用形態や，利用技術に応じて，枠組み自体柔軟に変化していくことが必要になる．この枠組みを構築するツールとしてMAPPERが適する理由は，その

機能の豊富さであり([付表1]参照)，さらに，従来評価されているMAPPERの開発言語としての側面は，EUCの基盤となる機能を迅速に構築し，速やかにEUC活動を立ちあげることに貢献する，といえる．

EUC環境の構築およびその利用のために，以下の手順で作業が進められた．

1) EUC環境構想の策定

① EUC環境モデルの提案

当社より情報システム部門へEUC環境モデルを提案し，了承を得る．説明の要点は以下の通りである．

・業務データベースとEUC用データベースの物理的分割の必要性
・EUC用データベースの性格と作成方法
・EUC環境における利用状況管理の考え方

② ユーザの承認

EUC環境モデルをユーザに説明し，了承を得る．説明の要点は以下の通りである．

・EUC環境の必要性
・EUC環境における具体的な作業例（画面遷移）
・提供される機能例

ユーザからは従来と何が変わるのか，どこが良くなるのか具体的な説明を求められた．

③ 詳細仕様の決定

個々の機能について詳細を検討し，仕様を決定した．A社システム運用規則との整合性をとるため，業後処理の組み込み方，自動実行処理の実施タイミング，ユーザ生成ランの実行登録方法などについて，詳細な検討が必要となった．

2) EUC環境の構築

ハードウェア環境は既存のものをそのまま利用したので，ここではEUC機能を実現するツールの開発がEUC環境の構築を意味した．

① 日本ユニシス開発

EUC補完ツール（詳細は［付図1］参照）は当社が開発を担当した．

② 客先システム部門開発

データ抽出・再編集機能は客先が開発を担当した．具体的には，当社のパッケージである「帳票作成支援ツール」をカスタマイズすることで実現した．

3) 初期EUCレポートの設計

EUCレポートというのは，本来ユーザ自身が自分のニーズに応じて作り出していくものである．しかし，多くの場合，何もない状態から新たに作り出すには困難が伴う．そのため，ユーザのニーズを引き出したり広げたりするための，いわばサンプルとしてのEUCレポートを初期設定することとした．

① 対象帳票の洗い出し

人事部門で使用している帳票は人事業務システムの設計時に洗い出されていたので，その中から新たな人事業務システムでは作成されない帳票をピックアップし，対象帳票とした．

② 対象帳票の分析

対象帳票の構成要素，元となるデータ項目，利用方法，利用頻度などを分析し，帳票のグループ化を行った．

③ EUC用レポートの設計

それぞれの帳票グループを構成する項目が，業務のデータベース上のどのレポートに存在するかを対応付け，データ抽出・再編集ツールに指定するパラメタを明らかにした．

④ EUC用レポート加工手順の設計

グループ化され設計されたEUC用レポートを使ってもとの帳票を作り出す手順を設計した．

加工手順そのものが今後のユーザの活動サンプルになることと，加工手順設計によってEUC用レポートの正当性を検証する意味を持つ作業といえる．

4) EUC環境の使用方法教育

① 教育内容の検討

情報システム部門のEUC支援担当者，およびユーザのキーマンを対象にEUC環境の機能と利用方法について教育を実施すことにし，内容を検討しカリキュラムを設定した（表1参照）．

表1　EUC教育カリキュラム一覧

教育内容	スケジュール
1．ＥＵＣとは 2．ツール構成 3．基幹ＤＢ概要　＆　ＥＵＣツール・デモ	1日目 9:00　～　10:15
4．ＥＵＣツール適用の手順 5．手順の詳細と留意点 5－1．対象ユーザの確定 5－2．基幹データベースの整理とデータ辞書 5－3．既存出力の分析と絞り込み 5－4．ＥＵＣデータベース初期設定プロセスの洗い出し	10:15　～　12:00
5－5．情報グループの設定 5－6．インデックス管理	13:00　～　17:00
6．ＥＵＣツール操作方法 7．汎用部品ランの操作方法 8．ＥＵＣツール利用 9．利用環境設定	2日目 9:00　～　14:30
10．教育 11．評価	未実施

② 教育用資料の作成

カリキュラムに合わせ，説明用資料，実習用資料，実習用データおよび環境を作成した．資料は合計48ページ，[付図2]として目次のみ添付したので参照されたい．データは人事業務システム開発で使用したデータをそのまま利用，環境は客先開発機上に設定した．

③ 教育の実施，フォロー

教育は，本番開始前に1回，カリキュラム通り2日間実施された．参加人員は13名（情報システム部門5名，ユーザ8名）であった．

人事部門の他のユーザに対しては，この教育を受けたキーマンが指導し，情報システム部門が質疑応答，電話対応などで支援することとした．

3. 利用状況と現状の問題

EUCの事例を紹介する場合，どのような環境あるいはシステムを構築したかということだけでは十分とはいえない．何度も繰り返すことになるが，EUCというのは，実際にユーザがその環境下でいかなる活動を行うかが最も重要であり，またその活動によって環境そのものに絶えず変化が求められるものだからである．

したがって，本章ではMAPPERにより構築されたEUC環境がどのように利用され，利用されることでどのような問題を明らかにしてきたかを見てみる．

3.1 利 用 状 況

いったん，ユーザによる利用が開始されると，その後の利用実態を外部から把握することは困難な場合が多い．今回は，ツールの利用に関しては管理資料が残る仕掛になっているため，作成されたレポートやランの数は明らかになった．

1) 利用開始時期

平成5年7月

2) 利用者数

12名

3) 利用頻度

利用回数	合計	平均(月)
レポート作成	181	18.1
ラン作成	42	2.3
ラン実行	465	25.8

4) 利用例

・統計管理：従業員の就業時間数の傾向把握のため，上期/下期，部署ごとの残業時間数をまとめる．
結果は，役員への報告，労働組合との折衝資料として利用．

・教育研修：中堅社員，新任課長/係長，海外研修等など，各所研修受講候補を洗い出し，対象者を選定する．

・その他　：各種名簿，出身校/学部別卒業者一覧などの外部依頼資料や異動候補者リストのような非定常資料の作成．

5) 使用所感

実際に利用することによって，様々な問題が明らかになってきた．また，人事要員の仕事の幅が広がったが，具体的な効果の把握は難しい．

利用面では以下の問題を感じている．

・レスポンスが遅い.
・2000行以上のレポートが保存できない.
・基幹の機能とした方がよい処理がある.
・操作性が悪い.
・EUCツール環境で使用できない命令，オプション等が不明確.
・罫線帳票定義保存の数が少ない.
・部品ランで誤操作した場合，操作前の状態に復帰できない.
・既存の抽出条件の変更時にもとの値がわからない.
・レポートの保存可能な数が少ない.

3.2 現状の問題

これらの情報は，A社における現在の利用状況を知る上で，決して十分なものとはいえないが，明確になっている問題を整理，分類してみると以下のようになる.

1) 環境・機能の問題

① 実行効率

複雑な構造の業務データベースを対象にしていること，様々なシステムが稼働するホスト上で集中的に実行されることといった要因から，データ抽出に予想外の時間を要することがある.(たとえば，5レポートに展開された千数百人分の給与情報12か月分を対象にした抽出で約60分かかったことがある.)

業務の効率化を物理的に阻害するのみならず，ユーザの「使おう」という意識を喪失させる恐れもある.

② ツールの制約

ツール設計時の考慮不足によるもの，ツールがもともと用意している可変の制約がユーザの想像を超えて厳しく設定されたものとがある.

③ 処理機能の不足

従来，利用できた機能がEUC環境で使用できなくなったことへの不満が最も大きい.

④ 操作性

機能面を充実させるため，さまざまなツールを提供した結果，操作の不統一，複雑化により，作業が煩雑になったという印象を与えた.

⑤ 説明不足，勘違い

ツール本来の機能範囲が正しく理解されていなかったり，誤った使い方に起因する問題が発生している.

EUC環境のPR不足や事前教育内容の不備，教育実施後のフォロー不足等に原因があると思われる.

これらの問題はいわばユーザの目に見えた問題である．これとは別に，ユーザが直接的に見たり感じたりしない問題も存在する.

2) 手順の問題

① ユーザへのアプローチの問題

基幹となる人事システム開発とほぼ並行してEUC環境構築が実施されたた

め，従来のシステム開発手法が引き継がれた形となり，開発時にユーザが作業主体になりきれなかった．このため，事前にユーザが利用している機能範囲を確認することができず，処理機能の不足を招いたりした．

② EUC実現の具体的な目標，効果の測定方法の問題

何がEUCの実現といえるか，EUCによってどのような効果がどのくらいあったかを，把握するための手段，目標値等が事前に設定されていなかった．このため，EUC環境利用結果を評価することができない．

③ 利用環境改善手順の問題

従来の業務システムと同様の提供方式がとられたため，機能改善の継続性が断絶してしまった．EUC環境は利用実態に合わせて変化できなければ，すぐ陳腐化してしまう．

3.3 改善策の検討

本節では，上記の問題に対して，いくつかの改善の方向性を提示してみたい．

1) 環境への提案

① 分散環境

実行効率の問題や厳しい利用制限は，このEUC環境が複数のシステムが共存するホストで集中的に実行されることに起因している．

そこで，EUC（情報活用）部分をその他の処理から切り離し，部門サーバやPC上に展開することで，これらの問題を解決することができる．

MAPPERの場合，UNIXサーバ上で稼働するOpen MAPPER，PCのWindows* 3.1上で稼働するMFW（MAPPER For Windows）により，既存の環境をそのまま移行することが可能である．

② EUCデータベースの構築

A社の場合，EUC環境の中で特にデータ抽出の機能を複雑かつ効率悪化させているのは，複雑な構造の業務データベースを直接ユーザに意識させないよう，すべての負荷をツールが負っていることにある．

分散環境の中に，ユーザにとって意味のある項目だけを，できる限りフラットに展開した，わかりやすい構造のEUC向けのデータベースを構築すれば，この問題を解決することができる．

2) ツール，機能への提案

① EUCツールの見直し

現状のEUCツールで使用できないMAPPER機能を明確にし，必要に応じて利用できるよう改善する．また，GUI（グラフィカル・ユーザ・インタフェース）を取り入れる等して，操作性の向上をはかる．

② PC連携の強化

すべての処理をMAPPERだけで実現するのではなく，市販のPCソフトが得意とする分野（たとえば，罫線/カラー編集やグラフ機能など）は，その機能とスムーズに連動するようにする．OpenMAPPERとDW（デザイナ・

---

* Windowsは米国Microsoft社の商標である．

ワークベンチ)またはMFWの組み合わせにより，PC上のファイルのやりとりやPCアプリケーション（たとえば表計算ソフト）との連携等が可能になり，ユーザにとってシームレスなデータ利用環境が実現する．

3)　今後の手順への提案

①　効果と評価

どのような情報システムも，それが実際にどれだけの効果をもたらしたか検証できなければ，存在価値は認められない．

環境やツールの改善にあたっては，その結果の具体的な目標や効果をできる限り数値化し，あらかじめ，測定時期や測定方法を明確にしておく．

②　業務活動への組み込み

効果の測定や評価，これに基づく改善は一回限りのものではない．継続的に改善，発展させていくためには，この活動を継続的にスケジューリングし，日常の業務活動に組み込むことが必要である．

4. EUC実現における今後の課題

ここまで，A社におけるEUC実現の手順と現状を見てきた．我々は，ここからいくつかの汎用的な問題と，これに対する解答のヒントを得ることができる．

最も大きな問題は，EUC実現のためには従来のシステム開発型とは異なる，ユーザ自身が作業主体となる新たな作業手順が必要だということである．

本章では，今回実施された作業手順とその問題，実施されなかった作業とそのことによる問題を整理した後，EUC実現のためのモデル手順を明らかにする．

4.1 手順と問題点

まず，今回実施された作業手順は2.3節の実現手順の通りである．手順ごとにその作業主体と作業対象，問題点を明らかにしたものが表2である．

作業の流れ自体に大きな問題はないと思われる．しかし，ユーザによる情報活用の実現を目指しながら，その実現主体であるユーザが単なる情報提供者にとどまっていることに問題がある．たとえば，EUCツールの詳細機能検討時にユーザが主体として機能していれば，あらかじめユーザが使用する機能範囲を考慮した設計が可能であったし，後に以前できていたことが新たな環境でできないという不満を引き起こすことはなかった．少なくとも，ユーザが不満を持つ前に代替案を提示することができたであろう．EUC用レポートの初期設計に関しても，設計担当の情報システム部員は何度もユーザと接触し，情報を引き出し設計したが，ユーザからは，自分の提供した情報がどう分析され組み立てられ，EUCレポートが設計されたのかが見えなかった．しかし，実際にEUC環境で活動が始まればこの作業は常にユーザ自身が実施しなければならないため，使い方が良く解らなかった利用開始直後には次々とレポートが作成され，これが処理時間や容量制限などの問題となって現れた．

また，実施された作業はEUC環境を構築し，これを使えるようにすることを主目的にしており，使用開始以降に関しては何の作業手順も予定されなかった．手順としては，従来の情報システム開発型に限りなく近いといえる．その結果，“EUC＝魔法の玉手箱”式の，何でも簡単にできるだろうという期待ばかりがふくらみ，実際の利用時

表 2　実施された手順とその問題点

手順		作業主体	問題点
1 EUC構想策定環境	1 EUC環境モデルの提案	NUL→情報システム部門	詳細仕様検討時に利用部門と十分検討ができなかったため、ツールの機能不足や運用ルールへの不満を引き起こした。
	2 ユーザの承認	NUL→利用部門	
	3 詳細仕様の決定	NUL＋情報システム部門	
2 EUC環境の構築	1 EUCツール開発	NUL	
	2 データ抽出ツール開発	情報システム部門	
3 初期EUCレポートの設計	1 対象帳票の洗い出し	情報システム部門	予め展開するデータに対してユーザの意志が十分反映されなかった。 必要データの絞り込み方や共有の考え方をユーザに伝えるチャンスを逸した。
	2 対象帳票の分析	情報システム部門→利用部門（ヒアリング）	
	3 EUC用レポートの設計	情報システム部門	
4 EUC環境の使用方法教育	1 教育内容の検討	NUL	NULが主体で実施したため、情報システム部門内にも何をどう指導すれば良いかという技術が十分伝わらなかった。
	2 教育用資料の作成	NUL	
	3 教育の実施	NUL＋情報システム部門→利用部門、情報システム部門	

NUL：日本ユニシス(当社)

には予想以上のギャップを生むことになった．さらに，この種の誤解を解き，ギャップを削減するための活動を実施することもできなかった．

今回実施されなかった作業手順と，実施されなかったことによる問題を記述したものが表3である．EUC環境がある程度整った後，ユーザの利用実態が把握できなかったり効果の評価が行えないのは，これらのことを必要な作業手順として実施していないからに他ならない．

一般的には，情報システム部門は次々と開発要件を抱えているし，新たな業務システムとEUCと同時に実施した場合，業務システム側の問題に引きずられてEUC支援が手薄になってしまいがちである．したがって，効果把握・評価・改善の手順はかなり意識的・強制的に行われない限り，その実施は困難な場合が多い．

4.2 EUC実現手順

これらのことから，EUC実現に必要な手順をまとめたものが表4である．

目的や目標・担当・作業等を明らかにし（計画），計画に基づき実際の環境を整え（構築），その環境を業務に取り込み利用し（実施），効果を評価し改善策を立てる（評価）といった全体の流れは，すべてのシステム構築と何等変わるところはない．

表 3　実施されなかった手順とそのことによる問題

手順と作業内容		作業主体	未実施による問題点
効果予測	ＥＵＣ環境によって何が実現できれば良いのか、どの程度の効果が期待されるか予測する。	利用部門 （情報ｼｽﾃﾑ部門支援）	機能の充実度、ユーザの生産性向上度を判断することができない。
試用	構築された環境、実際のデータを試用して、効果予測の対象となった実務処理を実践してみる。	利用部門 （情報ｼｽﾃﾑ部門支援）	試行から本番までの時間が短かったた、効率問題、機能不足、思い違いなどが一度に顕在化した。
評価	使用結果の効果を測定し、評価する	利用部門 （情報ｼｽﾃﾑ部門支援）	ＥＵＣ環境改善の必然性、重要度、歯止め等の判断ができない。
改善	評価結果に従って、 ・ツールの機能改善 ・運用方法の改善 ・目標、スケジュールの改善 等を行う。	利用部門 情報ｼｽﾃﾑ部門	ＥＵＣ環境の不備、陳腐化などによってユーザの作業負荷が高くなる。 ＥＵＣ環境が利用されなくなる恐れもある。

表 4　EUC実現手順

	ＥＵＣ実現手順	作業内容	作業担当	備考
計画	ＥＵＣ推進計画立案	推進組織（機能、役割）の具体化 推進スケジュールの策定 ＥＵＣのスローガン、目的の明示	ＥＵＣ推進プロジェクト （情報ｼｽﾃﾑ部門 利用部門）	EUCへの取り組み方、範囲によって異なる
構築	ＥＵＣ環境構築準備	現状分析（ＤＢ，システム、業務） ツール調査 対象ユーザ、対象アプリケーション選定 効果予測、評価方法検討 教育準備	ＥＵＣ推進プロジェクト ＥＵＣ実行プロジェクト	ＥＵＣの対象となった部門のユーザ主体
	ＥＵＣ基盤整備	Ｈ／Ｗ，Ｓ／Ｗ整備 データベース整備 基盤アプリケーション整備 ＥＵＣ要員の教育	ＥＵＣ実行プロジェクト 情報ｼｽﾃﾑ部門	部分的に外部へ切り出すことも可能
実施	ＥＵＣプロトタイピング実施	データ利用の手順試行 業務への適用 問題発見、改善	ＥＵＣ実行プロジェクト	
評価	ＥＵＣ評価、改善	効果測定 評価実施 改善方法検討	ＥＵＣ実行プロジェクト 情報ｼｽﾃﾑ部門	

大きな違いは，以下の二点である．

・ほとんどの工程で作業主体がユーザ自身であること．

・構築→実施→評価の流れが短期的に繰り返し実施されることで真に役立つ環境を作り続けていくこと．

従来も，情報システムへのユーザの参画，ユーザ主体のシステム構築ということが行われてきたが，ユーザの主体性は一時的なものであることが多かった．

実際，システム構築時にはそれなりの体制が整えられ，主体となるユーザの中にも熱い想いがたぎっている．しかし，時とともに想いは薄れ，当初の担当はローテーション等でその担当をはずれ，後にはできあがった仕掛と，その仕掛を使うための手順書と，アウトプットだけが残されるのが常であった．

ここでいう“ユーザ主体，作業の繰り返しと継続”は，EUC 実現手順そのものが，ユーザの業務活動の一環であることを求めている．また，単なる情報技術の発展，継承だけでなく，業務やそれを実現する想いやノウハウといったものさえ，進化・継承されることをめざした手順である．

以下に各手順の概要を簡単に述べる．

1) EUC 推進計画立案

EUC の全体的な方針，全体スケジュールを明確化する．この作業は，EUC 推進プロジェクトによって実施される．

EUC 推進プロジェクトの構成員が誰かということは，各企業の EUC への取り組み方や既存の組織・役割との関係から一概には言えないが，EUC 活動全体を統括するという役割を果たすためには，情報システム部門が技術的中心となる必要があろう．

2) EUC 環境構築準備

EUC 実施対象の選定と EUC 実行プロジェクトの立ち上げが活動の中心となる．

企業の中で活動するものはすべて EUC の対象ということができるが，最初から全体を対象として EUC 実現を目指すのは現実的ではない．EUC を実現するためにはこの後，環境構築，教育，ユーザ支援等様々な活動が必要であり，一度に全体に対して行うには負荷が大きすぎるからである．

したがって，EUC 実現の範囲を絞り込み徐々に拡大していくことが重要である．対象が決まったら，対象部門を中心とする EUC 実行プロジェクトが実活動を行うことになる．まず行うことは，業務の現状を分析し，EUC 実現の効果を予測することである．

3) EUC 基盤整備

EUC の基盤として，ハードウェア，ソフトウェア，データベース，アプリケーション，要員がある．これらを整備するのがこの工程である．

EUC 実行プロジェクトが実際にこの基盤をすべて構築しなければならないわけではない．基盤の規模や必要となる技術要素にもよるが，「何が必要かを明らかにし，これを整備できる人材を手配しコーディネイトする役割を持つ」と考えることが現実的であろう．

また，ここで整備される基盤ははじめから完全なものである必要はない．

4) EUC プロトタイピング実施

EUC を考える上で最も重要なことは，「ユーザのニーズはユーザが活動を続ける限り，常に変化し成長し続ける，どこかで完成したり，固定的になることはない」ということである．この変化し続けるニーズに対応するためには，情報ニーズの発生源であるユーザ自身が問題を解決し続けること，これが EUC の本質であるといえる．

情報ニーズを満足するためには，現状情報システム部員が保有している情報技術のある部分がユーザに移植される必要がある．さらに，これらの技術は一過性でなく，継続してユーザの中で伝承され成長していかなければならない．さらに，自分が取得した技術を伝承する技術の獲得が必要となる．

これを可能にするための技術的基礎を習得する活動が，EUC プロトタイピングである．

5) EUC 評価・改善

あらかじめ予測された効果が，プロトタイピングを通じて実現されたかどうかを測定し，その結果を評価する．その後，この評価に基づき環境，ツール，手順等の改善を計画する．

3)，4)，5)の作業が短期的に繰り返されるためには，その時点で明らかになっている情報ニーズ，そのとき実施し効果を評価することができる情報ニーズのみを対象とすることが重要である．

また，この EUC 実現手順では，利用部門と情報システム部門の関係が従来とは異なるものとなる．特に，情報システム部門は EUC 実現のために，以下の役割を果たさなければならない．

・EUC の基盤技術の保証，基盤環境の維持・整備
・EUC 全体の各種調整
・EUC 実施の技術移転，技術適用支援

このような支援を継続することによって，ユーザ自身が情報ニーズを自ら解決することができるようになっていくのである．

この，EUC 実現手順のポイントをまとめると，以下の四点になる．

・EUC の活動全体を統括する推進プロジェクトと，EUC 実現の実働部隊としての実行プロジェクトによる活動の定義
・EUC 実現の全体計画の立案
・整備，実施，評価/改善の短期サイクル実施
・EUC プロトタイピングによるユーザへの技術移転

このように見てくると，EUC を実現するために必要な手順とは，何ら目新しいものではないといえる．EUC 実現に際し，新しい技術要素は必ずしも不可欠ではないのである．

むしろ，従来無意識のうちに行われてきたことを意識化しその結果を検証すること，従来の仕事の進め方に対し発想を転換することが求められている．

5. おわりに

今回MAPPERを，単なる開発言語でなく，EUCの環境基盤として有効活用している事例から，EUCとは何か，それはどのように実現されるかを考えてきた．

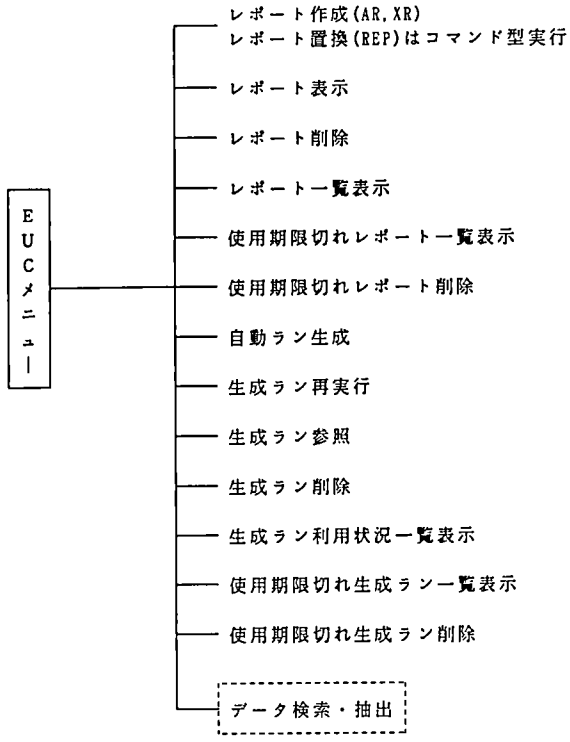
現在，EUC環境を実現するためのツールには様々なものがある．MAPPERは，有力ではあるが，この中の一つにすぎないともいえる．ユーザの真の情報ニーズに対し，最適なハードウェア，ソフトウェア環境を選択するための明確な判断基準が必要である．

また，EUC実現のためには，ハードウェア，ソフトウェアだけでなく，人的環境を含め，何をどう整備するか，何をどの順に実施するか，どのポイントで判断し軌道修正するか，といったことを明らかにする手順や方法が重要だということが解った．特に，この手順は特定のソフトウェアに拘束されるものではないと思われる．オープン・システム下におけるEUC実現の詳細手順の確立が急がれる．

私たちは長年MAPPERによってEUCの基盤となる環境を整備し，ユーザの情報利用を実現してきた．EUCの実現は，基本的には利用部門と情報システム部門の共同作業であるが，この実績と経験を活かし，さらに発展させることでユーザのより高度な情報活用の実現に今後とも貢献していきたいと考えている．

［付表1］　EUC環境を支えるMAPPERの機能

EUC環境の機能要件	MAPPERの機能
業務データベースから柔軟にデータを検索、抽出できる	MRI機能、ファイル・インタフェース機能、処理インタフェース機能、IDBKIT
利用者個人個人の視点を反映したデータの集合を保存し、複数の人が共同利用できる また、これらのデータを量的、質的に管理できる	MAPPERデータベース
情報活用の中の定常的な処理を個別のアプリケーションとして構築することができる	ラン機能、APT、DW
作業者の視点を反映したデータを見ながら、試行錯誤的にデータを加工し、最適の情報を創り出すことができる	会話機能、グラフ機能 ユーティリティ・ラン
試行錯誤や突発的な情報利用の中から生まれた新たな手順を簡単に保存・蓄積し、複数の人が再利用することができる	自動ラン生成機能
データから創り出された情報を様々な形で表現したり、効果的に他者に伝達することができる	グラフ機能、PCソフト連動機能 電子メール機能
上記の機能が個人レベルから全社レベルまで、必要に応じて最適のプラットフォームで稼働するとともに、それらが有機的に結合し機能することができる	3階層のMAPPER ネットワーク機能



[付図1]　EUC補完ツール

[付図2]　EUC教育用資料　目次（例）

執筆者紹介 松 木 規 子 (Noriko Matsuki)

1956年生. 80年早稲田大学第一文学部卒業. 同年日本ユニシス(株)入社. 81年10月よりMAPPERシステム適用サポートに従事. 現在システム技術本部応用ソフトウェア部に所属.

OA支援ソフトウェアCamel/Latchの概要

An Overview of OA Support Software—Camel/Latch

藤井昭彦, 伊東春正

要　約　ワープロ，表計算ソフトウェアなどのOAツールの普及が進むにつれて，次の要求が高まっている．

① OAツールで作成したデータを部門内で共有して有効に利用したい．

② アプリケーション・システムで蓄積したデータをOAツールに取り込んで活用したい．

Camelはサーバ/ワークステーションの2階層構成のもとでOAツールのデータを保管するキャビネット，ワークステーション間でデータを転送する電子メールなどの「オフィス連携」機能を提供することによって，①の要求に応える製品であり，Latchはホスト/サーバ/ワークステーションの3階層構成のもとでホストのデータベースからデータを抽出してOAツールに取り込む「データ連携」機能を提供することによって②の要求に応える製品である．

本稿では，まずCamel/Latchのソフトウェア構成と機能を概説し，次に特徴的な機能についてその要求の背景，狙い，実現方法を含めて紹介する．

Abstract　With the wider spread of OA tools, including powerful workstations, such as word processors and useful spreadsheets, the following requirements have been on the steady rise:

1) The sharing, within the same departments, of data created on OA tools for more effective use of them.

2) The incorporation by OA tools of data accumulated by application systems for better use of them.

Camel is a product that responds to item 1) above by serving as a data cabinet for OA tools in a two-layer configuration of servers and workstations, and by providing "office-linking" functionality like e-mail which transmits data between workstations. On the other hand, Latch satisfies the item 2) requirement by providing "data-linking" services which help select data out of a host database and feed them into OA tools in a three-layer configuration of a host, servers and workstations.

Besides giving an overview of the software structure and functionality of Camel/Latch, this paper discusses their remarkable features as well as the background of their need and implementation.

1. はじめに

近年ホワイトカラーの生産性向上が企業の課題となっており，ワープロ，表計算あるいはデータベースなどのパーソナル・データ処理用ソフトウェア（以降ではOAツールと呼ぶ）がワークステーションの低価格化とも呼応して急速に普及している．

当社では統合型のOA支援ソフトウェアとして，EXOS(Excellent Office System)を提供してきた．EXOSではワープロ，表計算，ペイント，ドローなどのOAツールのほとんどすべてを当社製品として提供している．また，OA手続きソフトウェア(EXMILD)によって，エンドユーザ自身が定型的な処理をシステム化することを可能

本稿に記載の会社名，商品名等は，一般に各社の商標または登録商標である．

ている．

一方，使いなれた市販のOAツールを利用して，パーソナル処理システムや部門処理システムを構築したいとの要求がある．このようなシステムの構築には，OAツールのデータやホスト/サーバに蓄積されたデータをネットワーク上で転送・変換・保管する支援ツール群が必要となる．

支援ツール群は走行するプラットフォーム，連携するOAツールの種類などによって多岐にわたるが，当社では次の2種類の体系のもとに提供している．

・オフィス連携

・データ連携

オフィス連携では主にキャビネット機能と電子メール機能を提供することにより，複数のワークステーション間でOAツールのデータ共用を実現する．また使用者管理機能によって，共用データの機密保護を強化する．

データ連携はホスト/サーバ/ワークステーションで構成された分散処理環境上で，データベースを選択し，条件を設定してデータを抽出し，ネットワーク上で転送し，コード変換/形式変換してOAツールに取り込むことを可能とする機能である．

Camel/Latchは各々オフィス連携機能，データ連携機能を提供する製品群の総称であり，本稿で紹介するバージョン（1R2B）ではホスト（シリーズ2200），サーバ（U6000シリーズ），ワークステーション（U6000/DTシリーズ）で走行する複数のソフトウェアによって構成されている．

2. Camelの概要

Camelはオフィス業務の効率化を支援するソフトウェアであり，OAツールのデータを共有するためのキャビネット機能，コミニケーションを円滑にするための電子メール機能およびCamelの使用者を管理する運用管理機能を提供している．

2.1 ソフトウェア構成

Camelは，図1で示すとおり，サーバで走行するCamel/Sとワークステーションで走行するCamel/DTで構成されている．

なお，電子メールでは，UNIX通信によるサーバ間のメール転送に加えて，分散データデリバリ（DstDDL）を介した転送も可能としている．

2.2 機 能 概 要

2.2.1 キャビネット

キャビネットは，UNIX*ファイル・システム上で階層化されたディレクトリとファイルを，オフィス内のキャビネットと保管書類の関係に当てはめて，データ・ファイルの保管を行う．

1) キャビネットの種類

使用目的により次の4種類のキャビネットが提供されている．

① 共有キャビネット

サーバに設置されたキャビネット．サーバに接続されたすべてのワークステーションから参照する共有データを格納する．

---

* UNIXはX/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標である．

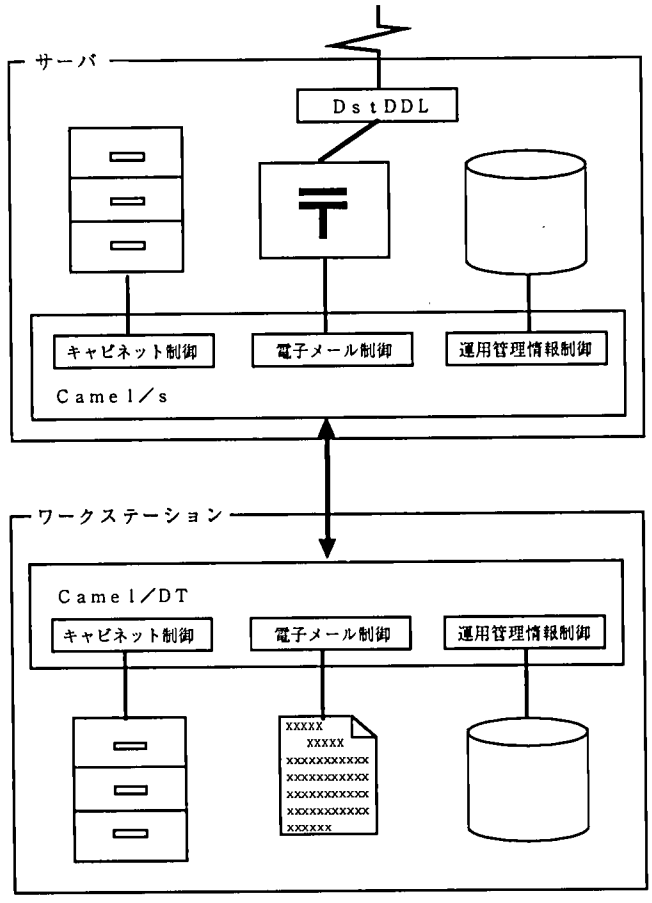


図 1　ソフトウェア構成

② ワークステーション・キャビネット

ワークステーションに設置されたキャビネット．個人データを格納する．

③ ディスケット・キャビネット

ディスケット上のキャビネットで，次のような目的で使用する．

・オブジェクトのバックアップをとる．

・オブジェクトを他のワークステーションで使用する．

・パソコンで作成したオブジェクトを取り込む．

④ 拡張ワークステーション・キャビネット

共有キャビネットと同様にサーバに設置されたキャビネットであるが，次のような目的で使用する個人用キャビネットである．

・ワークステーションのディスク空き容量が不足している．

・使用するワークステーションが特定できない．

2) 属性の付加

UNIX ファイル・システムが管理するディレクトリ名とファイル名とは別に，ファイル内容を表す日本語名称や機密保護のための所有者名や使用権限などを属性として付加する．属性を付加したディレクトリをフォルダ，ファイルをオブジ

エクトと呼んでいる．フォルダおよびオブジェクトには，属性により個人用，グループ用または共用の使用者範囲を設定することができる．

3) オブジェクト操作

Camel 起動後に表示される作業机から，フォルダおよびオブジェクトの一覧ウィンドウを表示し，一覧ウィンドウからマウス操作によりオブジェクトの作成や更新ができる．また，一覧ウィンドウ間でのドラッグ操作によりオブジェクトの複写や移動もできる．

図2は，作業机からワークステーション・キャビネットを開いたウィンドウ例である．

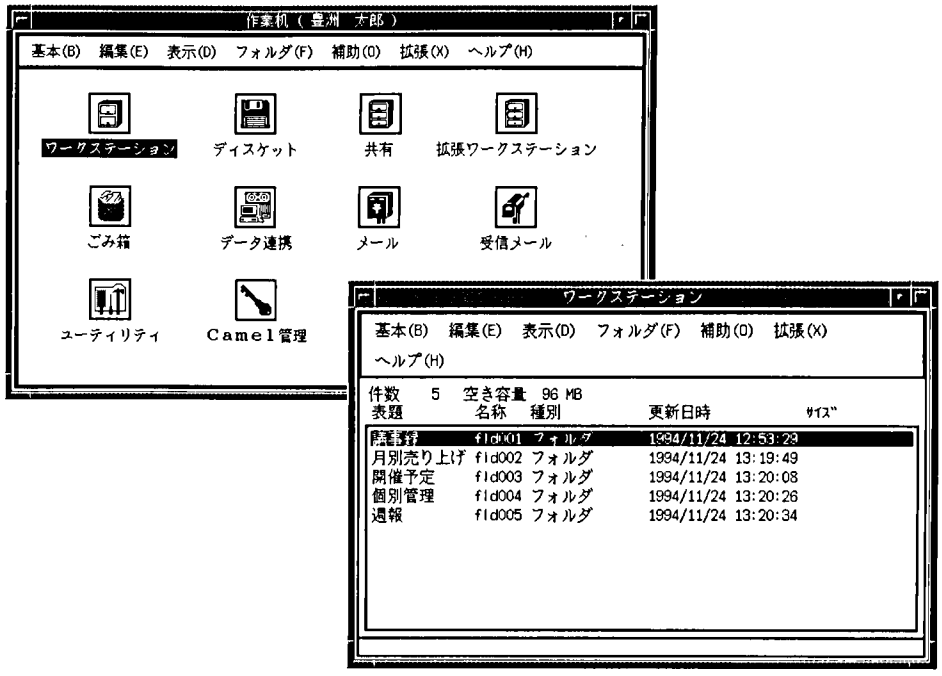


図 2 キャビネットのウィンドウ例

2.2.2 電子メール

Camel の電子メールは郵便局と私書箱で構成される．各サーバには郵便局が配置され，郵便局内にはメールを受信するための私書箱が置かれる．

1) 私書箱

私書箱には使用者および使用者グループを割り当てて使用する．

メールを発信する際には郵便局と私書箱を宛先としてメールする．私書箱に届いたメールは登録された使用者のみが開封できる．

私書箱には，複数の使用者および使用者グループを割り当てることができるため，一人の使用者が個人用，職位，担当分野などに応じて複数の私書箱を使い分けたり，複数の使用者が一つの私書箱を共同で使用するなど柔軟な運用ができる．

2) メール内容

メールでは，簡単なメッセージを含むメモにキャビネットで保管しているオブジェクトを同封して送付できる．また，メール内容をキャビネットに保管し繰り返して利用できるため，定型的なメールでの発信操作を簡略化することができる．図3は，電子メールの発信ウィンドウ例である．

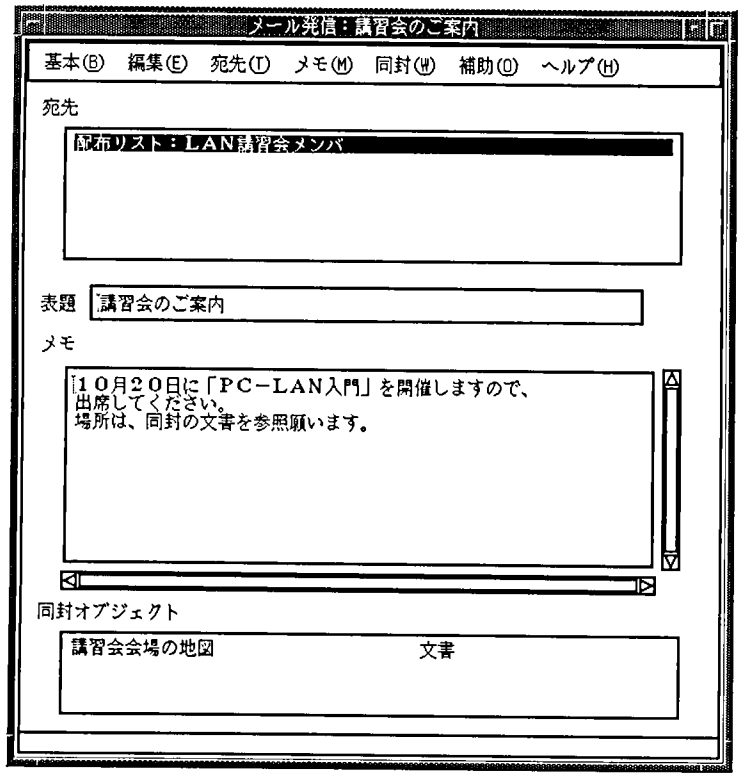


図 3　電子メールの発信ウィンドウ例

3)　受信通知

メールを受信した時，Camel を使用中であれば即時に受信通知ウィンドウを表示する．使用中でなければ使用開始時に受信通知ウィンドウを表示する．

2.2.3 運 用 管 理

Camel を運用するための次の機能を提供している．

1)　使用者管理

Camel を使用するには，使用者情報の登録が必要である．

使用者情報には，使用者 ID，パスワード，姓名，使用権限，管理権限などがあり，使用開始時に，使用者 ID とパスワードの入力を要求し正当性をチェックしている．

2)　グループ管理

複数の使用者をまとめて一つのグループとすることができる．

グループの登録により，キャビネットおよび私書箱の使用権限をグループ単位に設定できるため，これらの管理が容易となる．

3) 管理者機能

「Camel 使用者を登録する」「共有キャビネットにフォルダを作成する」「電子メールの私書箱を設置する」などの動作環境の設定は，管理者としての権限を持つ使用者（Camel 管理者）にのみ許されている．

Camel 管理者は各サーバごとに置くことを基本としているが，複数の部門が1台のサーバを共同利用するような場合，それぞれの部門ごとに管理者を置き独立してキャビネットや私書箱を管理することもできる．また，各サーバごとに管理者を置くのではなく，複数サーバをまとめて管理することができる遠隔サーバ管理機能も提供している．

2.3 Camel の特徴

2.3.1 オフィス連携用デスクトップの提供

ワークステーションの OS である UnixWare*では，デスクトップと呼ばれるウィンドウが提供されており，GUI 操作による OA ツールの起動やファイル操作が可能である．これらは，Camel のデスクトップ(作業机と呼ぶ)と類似しているが，UnixWare のデスクトップはファイル管理だけではなく，マウスボタンの機能割り当てや画面色の設定などのシステム管理用のメニューも提供している．このため，ワークステーションに詳しくないエンドユーザがデスクトップを操作することは困難である．

また，複数ファイルで一つのオブジェクトを構成するような場合，UnixWare のデスクトップでは，すべてのファイルが表示されるため，オブジェクトの複写や削除などの操作が煩わしくなる．

このように，Camel では OA ツールの使用者を対象として，OS の特徴やファイル構造を意識させない，より簡便なデスクトップを提供している．

2.3.2 キャビネットの機密保護

キャビネットでは，フォルダおよびオブジェクト単位に機密保護を行っている．

機密保護クラスは，Camel の使用開始時に入力する使用者 ID とフォルダおよびオブジェクトに設定されている保護属性で管理される．

- フォルダの保護属性　　：個人用，グループ用，共用の 3 段階
- オブジェクトの保護属性：個人用，共用の 2 段階

使用が許可されていないフォルダおよびオブジェクトは，一覧ウィンドウに表示されないため，それ自身の存在すら隠されることになる．

図 4 は，フォルダとフォルダに格納されているオブジェクトに対する保護属性の設定状態と，参照できる使用者の関連を示している．

2.3.3 OA ツールとの連携レベル

Camel では，OA ツールとの連携を三つのレベルに分けて提供している．

レベル 1 ：ファイル連携レベル

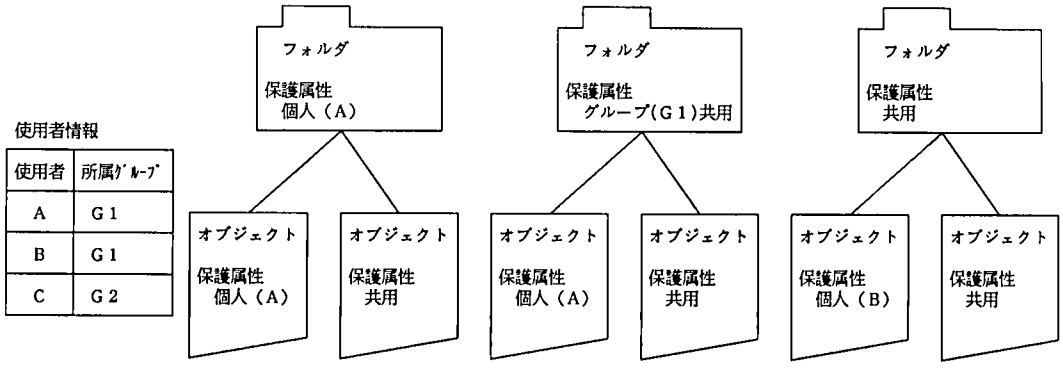
OA ツールが出力したファイルの管理のみを行う．

MS-DOS**の一太郎**や Lotus 1-2-3**などワークステーション上で実行できない OA ツールのファイルが対象である．

---

* UnixWare は米国ノベル社の商標である．

** MS-DOS は米国 Microsoft 社，一太郎は(株)ジャスト・システムの登録商標である．また Lotus 1-2-3 は，Lotus Development Corporation の商標である．



各使用者が参照可能なオブジェクト

使用者　A	○	○	○	○	×	○
使用者　B	×	×	×	○	○	○
使用者　C	×	×	×	×	×	○

○参照可
×参照不可

図 4　保護属性の設定例

このレベルで管理されたファイルに対しては，キャビネットでの保管およびメールによる転送が可能である．

レベル 2 ：プログラム連携レベル

Camel が OA ツールを起動するレベルである．

あらかじめファイルとプログラムとをリンク付けすることにより，ファイルに対応するプログラムを自動的に認識し起動する．その後のプログラムからのデータ保存時に，日本語名称や所有者などのオブジェクト属性が自動的に設定される．

レベル 3 ：メニュー連携レベル

OA ツールのメニューの中から，オブジェクトを選択したりメールを発信できるレベルである．

このレベルを実現するには，OA ツール側でメニューを組み込むためのマクロ機能が用意されていなければならない．Camel では WINGZ* に対してメニュー連携のためのスクリプトを提供している．

図 5 は WINGZ の Camel 連携メニューからオブジェクトを選択するウィンドウである．

2.3.4　API（Application Program Interface）の提供

アプリケーション・プログラムから Camel の機能を使用できるインタフェースを提供している．これによりユーザは，キャビネットおよび電子メールと連携するアプリケーション・システムを構築することができる．

次に Camel が提供している API の概略を述べる．

キャビネットでは，フォルダおよびオブジェクトの作成，複写，削除，取り出し，

* WINGZ は，米国 Informix Software, Inc. の登録商標である．

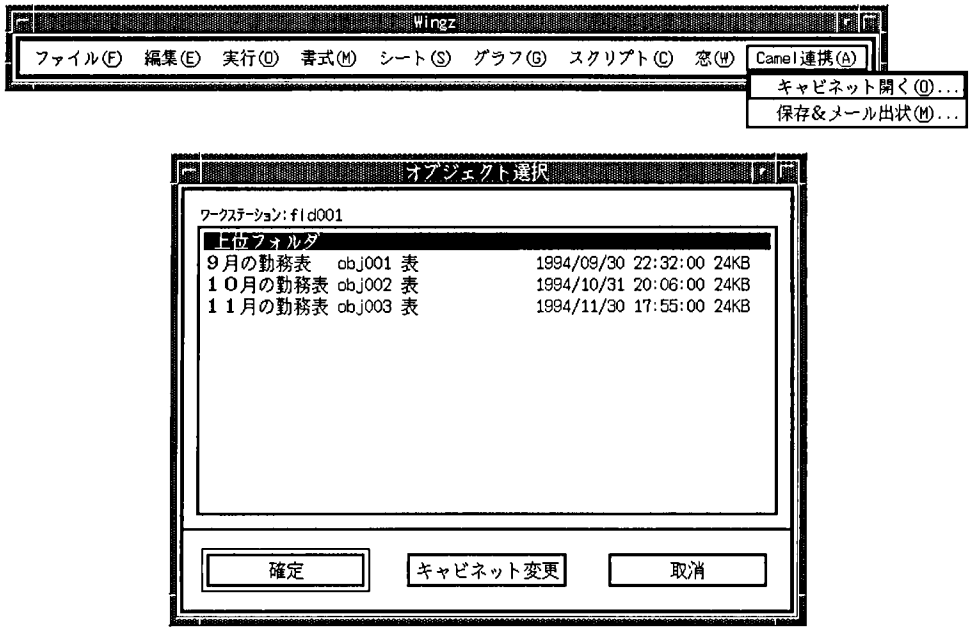


図 5 WINGZ の Camel 連携メニュー

検索などの API を提供している．これらを使用して，アプリケーション・プログラムで作成したデータをキャビネットに保管したり，キャビネットに保管中のオブジェクトを取り出すことが可能である．また，オブジェクト起動も可能であるため，アプリケーション・プログラムから直接 OA ツールを起動する考慮はいらない．

電子メールでは，メールの発信，送付状態の確認，受信メールの受け取り，返信，削除などの API を提供している．また，私書箱にアプリケーション・プログラムを登録しておくことにより，メールを受信すると受信したメール情報を付けて登録したアプリケーション・プログラムを起動する機能も提供している．これらを使用して，ユーザ独自のメール・システムや，データ集配信システムが構築できる．

運用管理では，使用者登録，使用者検索，Camel システムへのサインオン/サインオフ・インタフェースなどを提供している．これらを使用して，Camel 使用者の一括登録や，アプリケーション・プログラムのメニューから Camel システムにサインオンすることが可能である．

3. Latch の概要

Latch はデータ連携機能を提供するソフトウェアであり，レベル 1R2B ではホストの RDMS 1100 データベースからデータを抽出して，informix*データベース，Camel キャビネットおよび WINGZ へ格納する機能を提供している．

3.1 ソフトウェア構成

Latch は図 6 に示すとおり，ホスト，サーバおよびワークステーションで走行する 5 種類のソフトウェアで構成されている．

* informix は，米国 informix software 社の登録商標である．

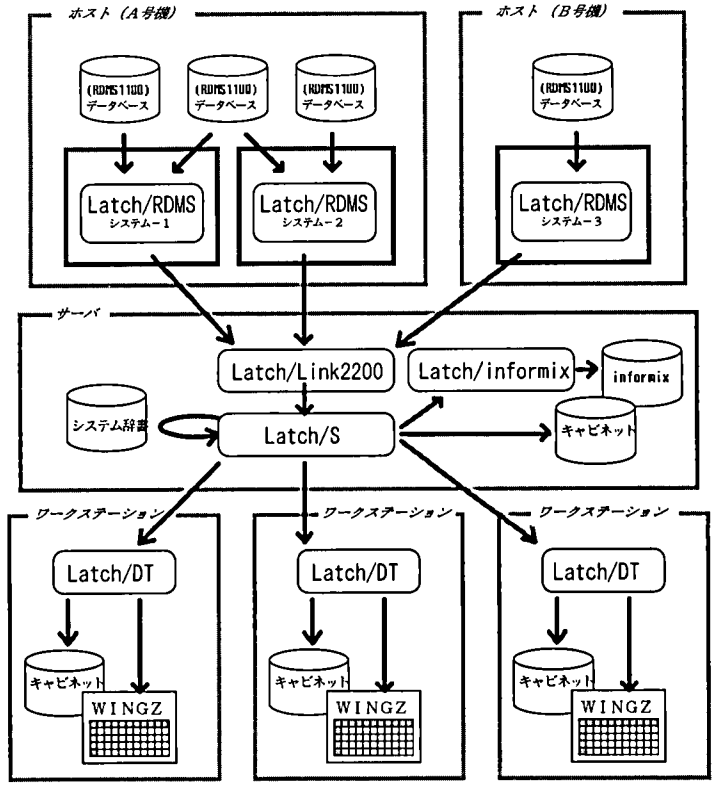


図 6　ソフトウェア構成

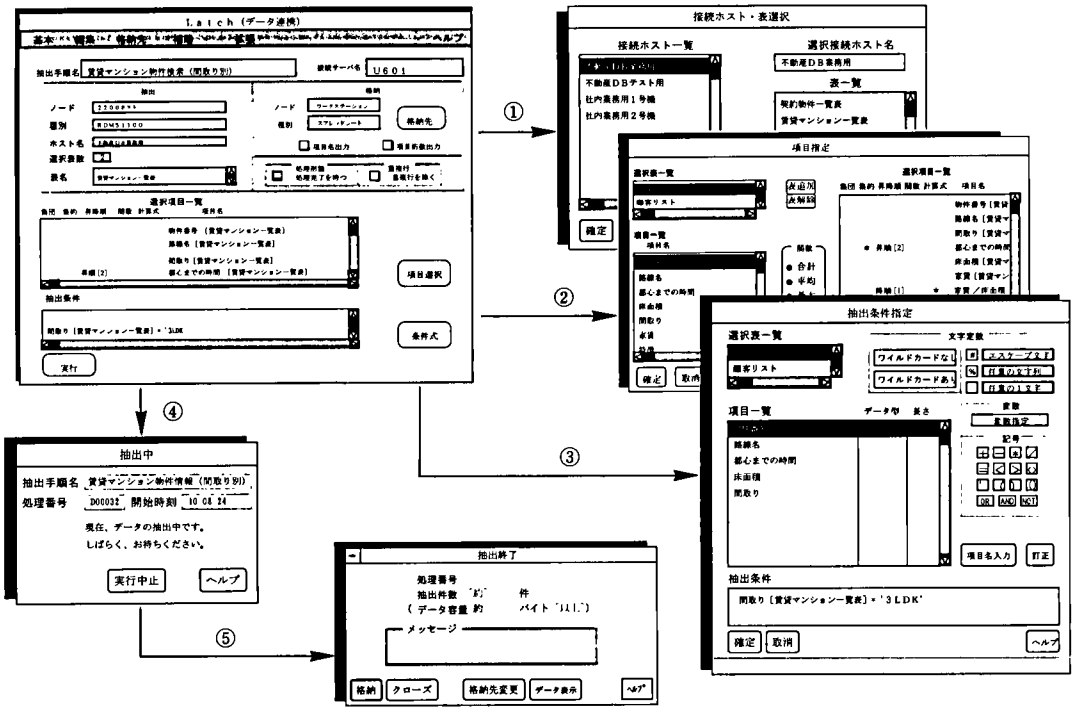
各ソフトウェアは次の機能を分担する．

- Latch/RDMS：サーバから送られた抽出指示文を解析して RDMS 1100 データベースからのデータ抽出を行う．
- Latch/Link 2200：サーバ上の通信ソフトウェア IS 6000 を介してホストに設置された Latch とプログラム間通信を行う．また，送受信データの文字コード変換も行う．
- Latch/S：データ連携のサーバ機能を持つソフトウェアであり，実行環境を定義したシステム辞書とユーザが保存を指示した抽出指示文を管理する．また，サーバ内の各ソフトウェアを制御して，データの抽出指示から格納に至る一連の処理を実行する．
- Latch/informix：抽出データを informix データベースへ格納する．
- Latch/DT：データ連携のクライアント機能を持つソフトウェアであり，抽出指示文の編集，実行/格納指示などを行うユーザインタフェースを提供する．また，表計算ソフトウェア WINGZ との連携機能を提供する．

3.2 機 能 概 要

3.2.1 RDMS 1100 からのデータ抽出

GUI 操作によって抽出指示文を編集した後，実行指示によって RDMS 1100 データベースからのデータ抽出を行う．操作の流れを図 7 に示す．



①ホストと表を選択する． ②抽出する列を選択する． ③検索条件を指定する．
④抽出処理の実行を指示する． ⑤データ抽出の終了が表示され，了解指示でデータが格納される．

図 7 抽出指示の操作例

3.2.2 抽出データの格納

抽出指示文の作成時に，データの格納先を次の 3 種類の中から GUI 操作で選択して指示する．格納先はデータ抽出が終了した時点で変更することも可能である．

- informix データベース
- WINGZ のワークシート
- Camel キャビネット

informix データベースの選択操作を図 8 に例示する．

3.2.3 WINGZ との連携

UNIX ワークステーションで走行する表計算ソフトウェアとして，当社は WINGZ を提供している．Latch では WINGZ のマクロ言語で記述した拡張メニューを提供することによって，起動から抽出データの取込みに至るシームレスな操作を可能としている．WINGZ と Latch の連携操作例を図 9 に示す．

3.2.4 API

アプリケーション・システムの中に Latch を組み込んでシステムを構築するために，各種の API が用意されている．

1) Latch の起動

アプリケーション・プログラムから API によって Latch を起動することがで

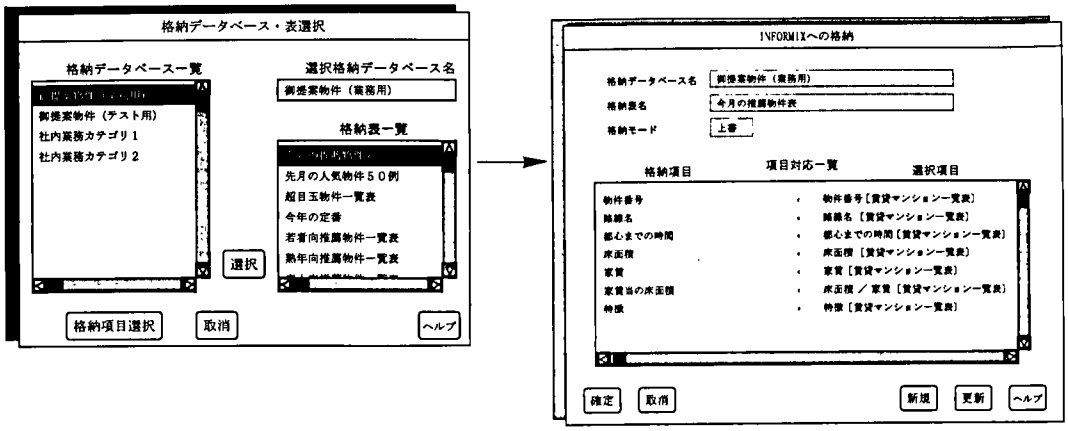
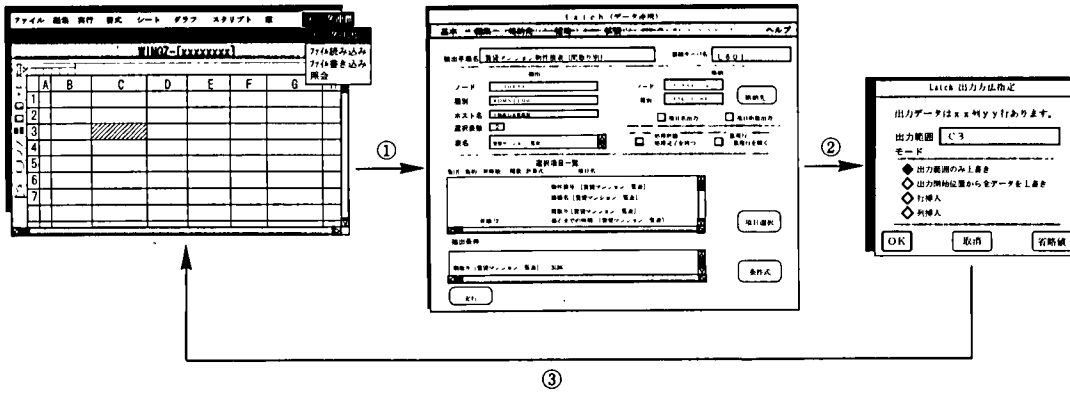


図 8　格納先選択の操作例



①データ抽出を指示する．　②Latchが起動される．　③シートへの格納方法を指示する．

図 9　WINGZ との連携操作例

きる．この場合，Latch は一切の画面表示を行わないで指定された抽出指示文の実行のみを行う．指示文の誤りなどの実行ステータスは指定されたファイルを経由してアプリケーション・プログラムへ引き渡す．

2)　抽出指示文の参照・更新

ユーザ・プログラムから保存済の抽出指示文を参照して，ユーザ・プログラムから API によって更新することができる．この機能は指示文中に変数（後述）を含んでいる場合に有効となる．

3.3　Latch の特徴

3.3.1　大規模データベースの考慮

クライアントの表計算ソフトウェアから UNIX サーバのデータベースへ接続して検索/更新を行うソフトウェアは数多く市販されているが，大半の製品では中規模以下のデータベースを想定してクライアント/サーバ方式を採用している．この方式では，SQL 文や検索結果を 1 行ごとに通信することとなり，伝送量の増大と処理時間の長大

化を招くこととなる．

Latch はシリーズ 2200 ホストの大規模データベースを想定して設計されており，次の機能を実装している．

1) 抽出データ容量の制限

サーバに設置したシステム辞書に抽出データ容量の制限値を設定することができる．制限値には警告値と限界値の 2 種類があり，限界値を越えた場合は抽出を許可しない．この機能によって，検索条件の誤指定などにより膨大なデータを抽出することを防止できる．抽出データの容量（概算値）は抽出終了時に利用者へ通知される．

2) データ抽出の中止

ホストに対する抽出指示文の送信を終了した後でも，抽出処理の中止を指示することができる．

3) 抽出データの確認

抽出データを格納する前に，結果を画面に表示して確認することができる．また，確認後に条件を変更して再度抽出を実行すること，格納先を変更することが可能となっている．

4) 検索時間の制限

抽出指示文の中に検索時間の最大値を指定することができる．制限時間を超過すると処理を中断して，その旨利用者へ通知する．

5) 非同期型のデータ抽出

データ抽出実行時にオプションとして「検索の終了を待たない」モードを指定することができる．このモードでは，ホストに対する抽出指示文の送信が完了すれば利用者は次の操作を行うことができる．抽出処理の実行状況は，照会機能を使用して確認できる．

3.3.2 抽出データの転送方法

抽出データの転送方式として一般に次の 2 種類のいずれかが採用されている．

- プログラム間通信
- ファイル転送

プログラム間通信はクライアント/サーバのプログラム間でデータを直接受け渡す方式であり，少量データに対して効率が良い方式である．

一方，ファイル転送は抽出結果をファイルに落として転送する方式であり，データが大量となった場合に効率が良い．

Latch は抽出データの先頭部（約 32 K バイト）をプログラム間通信，残りをファイル転送することによって，両方式の長所を取り入れた設計としている．

3.3.3 転送データの構成

一般に OA ツール間のデータの受け渡しには次の 2 種類の形式が用いられる．

- タブセパレート形式
- カンマセパレート形式

タブセパレート形式は，フィールド間をタブで区切った形式であり，数値と文字は形式上は区別されない．

カンマセパレート形式は，フィールド間をカンマで区切った形式であり，文字型フィールドはダブルコーテーション（"）によって囲まれる．

データベースがあらかじめ列ごとにデータの属性を定義するのに対して，表計算ではデータシートに入力された値または計算結果によって，セルごとに属性が決定されるソフトウェアであり，データの受け渡しにはタブセパレート形式の親和性が高いと考えられる．

Latch では市販の代表的な表計算ソフトウェアが提供しているタブセパレート形式をホスト/サーバ/ワークステーション間で転送する際の内部形式として採用している．

3.3.4 集 団 項 目

リレーショナル・データベースの基本的な考え方では，データを意味のある最少単位に分割して列に収納するが，実際のデータベース設計では複数の項目を連結して一つの列とすることもある．

Latch では，このように複数の項目で構成される列を特に集団項目と呼び，データ抽出の際に集団項目中の指定された項目のみを抽出の対象とする機能を提供している．

3.3.5 抽出指示文への変数指定

抽出条件式の定数の替りに変数を指定することができる．この機能を利用することによって，一度作成した抽出指示文の条件を変更しながら，繰り返し利用することが可能となる．

定数指定の例：沿線＝'小田急'AND 部屋数＝'3 LDK'

変数指定の例：沿線＝%路線名 AND 部屋数＝%間取り

例示した条件式を含む抽出指示文を実行すると次のダイアログ（図 10）が表示され，Latch は変数を入力値で置換して，抽出を行う．

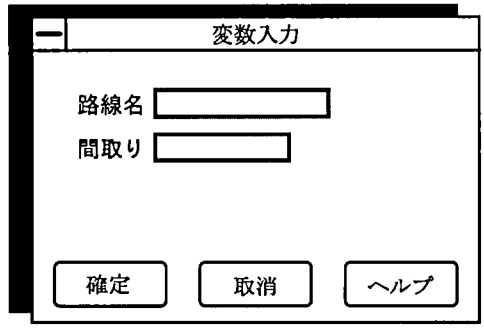


図 10　変数値入力ダイアログの例

4. お わ り に

EXOS の後継製品として開発してきた Camel/Latch も平成 7 年 4 月の 1R2B リリースをもって一応の完成をみた．また，EGWord*（ワープロ），WINGZ（表計算），メニュー・ソフトウェア GMenu の提供によって，UNIX を OS とするワークステーシ

* EGWord は，(株)エルゴソフトの登録商標である．

ョンであるU 6000/DTシリーズをOA分野に適用するための基本的なツールを整備することができた．

我々がCamel/Latchを開発した時期は，同時にパソコンの低価格化，Windows 3.1とそのもとで走行するOAツールの普及が加速した時期でもある．支援ツールに対する要求もさらに多様化していくものと予想している．

今後も利用者の声に耳を傾けながら，OAツールの利便性向上に有効な支援ツールの開発・提供を継続していくことが，我々の課題と考えている．

執筆者紹介 藤 井 昭 彦（Akihiko Fujii）

昭和20年生．44年大阪大学理学部物理学科卒業．45年日本ユニシス(株)入社．同社内システムの開発，ニューメディア関連ソフトウェアの開発を経て，OA関連ソフトウェアの開発・保守に従事．現在システム技術本部OAソフトウェア室長．

伊 東 春 正（Harumasa Itoh）

昭和30年生．48年福岡県立田川工業高等学校機械科卒業．同年日本ユニシス(株)入社．カスタマサービス部門を経て，OA関連ソフトウェアの開発・保守に従事．現在システム技術本部OAソフトウェア開発室に所属．

スタッフウェア

Staffware

戸叶孝一

要　約　ワークフローは欧米のビジネス環境では新しい概念ではない．オフィスの中での個々の役割がきちんと決められているオフィス環境では，生産性の高いオフィスを実現するための最高のツールとして使われている．現実のオフィスでワークフローソフトウェアが使われるためには，定型的な業務をこなすだけではなく，通常人間が行っているような不規則な流れを制御しなければならない．スタッフウェアはそのような過酷な要求に耐えられる唯一のソフトウェアである．

Abstract The concept of "work flow" is nothing new in the Western business world. In the office environment where an individual's role and function are clearly defined, work flow management has been adopted as the most powerful tool that helps pave the way toward an office of higher productivity. For the wide acceptance of work flow software in the actual office environment, it has to be able to control not only a predefined data flow but also the irregular flow that is usually taken care of by human beings. Staffware is the only software product available at present that meets such demanding requirements.

1. はじめに

UNIX* とPC，あるいはPCとPCをLANで接続したクライアント・サーバと分類される使用形態が使われ始めてかなりの時間が経った．電子メール，会議室予約，データベース・アクセスなど幅広い運用形態が存在する．このような形態のシステムにおいてワークフローという言葉が流行り出したのはそれ程新しいことではない．オフィスの生産性を向上させるためのツールとしてワークフローが取り上げられるが，その原点には二つの流れがある．一つの源流は光ディスクによるイメージ処理分野であり，もう一つはコンピュータによる事務処理分野である．いずれにも共通する点は源伝票の起票から始まり照査・承認が連続して行われる処理にある．光ディスクを中心とするワークフローでは源伝票がイメージであるか，あるいは伝票をチェックする上でイメージを参照することが必須であることが特徴である．コンピュータ処理から始まるワークフローではコンピュータ化から遅れた事務処理分野であること，および伝票の起票から処理終了までの期間が長いことが特徴とされる．通常の伝票処理は即時，あるいは数日の内にコンピュータ処理が行われる．しかし，会社設立のための審査，稟議システムなどのように書類の起票から承認が確定するまでの間に沢山のステップを通り長い日数がかかる．このようなアプリケーションはコンピュータ化からはほど遠いと思われていた．ワークフローの基本技術は電子メールである．電子メールは差出人から受取人へメールが送られるだけであるが，ワークフローはこれに一つの道筋をつけ，メールが次々と渡される流れを定型化し，メールの内容を調べメールを次の

* UNIXは，X/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標である．
本稿に記載の会社名，商品名は，一般に各社の商標または登録商標である．

誰に渡すかを判断できるようにした．これがワークフローの備えるべき最小の機能である．あとはコンピュータ技術者の工夫により，メールの追跡や，受け取ったメールを期限までに処理することを義務付けるなどの機能が追加されている．

ワークフローの本質はあらかじめ定められたルートの上をメールが渡されていくことにある．したがってあらゆるルートを考察し，フローの形に仕上げなければならない．しかし，すべての場合を網羅することは人間にとって非常に過酷な要求である．あらかじめ考えられていないルートにメールを送りたいという要求があるが，これを一度許してしまうと次から次へと要求が飛び出し，ワークフローにはならなくなってしまう．これならば電子メールのみの世界へ戻る方がよい．ワークフローを成功させるにはできる限りこのような要求を切り捨て，ルートを最短にするためのネゴを組織の中で行うべきである．

コンピュータから発生したワークフロー・システムは当初，自身が作る伝票形式のみを扱ってきたが，このようなシステムにも光ディスク・ファイルあるいは表計算やワープロのファイルなどの外部オブジェクトを電子メールに添付する機能が付加されるようになってきた．このような発展の元となるのはウィンドウズに寄るところが大きい．一つの画面の上に複数の窓を設けて，各々の窓に別々のソフトウェアが実行される．確かにプリエンプティなマルチ・タスクは使いにくいかもしれないが，複数のプログラムを手軽に使うことができるようにした功績は大きい．電子メールの限られた画面ではなく，使い慣れたアプリケーション・プログラムで情報が読めることは重要である．ワークフローがもて囃されるのはこれからである．単に外部オブジェクトを添付するだけでなく，外部オブジェクトとデータの相互交換が可能になり，ワークフローの分岐条件にこれらのデータを積極的に利用することができる．このようになるとワークフロー・ソフトウェアはレターヘッドになり，アプリケーションは外部のソフトウェアで行うことができる．数値を扱うデータはEXCEL* やLotus 1-2-3/Windows* などのような表計算で行い，稟議書や提案書などはMS-Word, AmiPro* のようなワープロを用いることができる．

パーソナル・コンピュータが出現してから簡易言語と呼ばれる分野のソフトウェアが出現した．当初の目的は数表を縦横斜め自由自在に操る表計算ソフトウェアと呼ばれていた．同じようなアイデアが次々と発表され，競合になると機能はそれにつれて増え，次第にエンドユーザがとても使えないソフトウェアになってしまう．複雑なものを簡単に表現できるのに，進めていくと益々複雑な問題を解こうとする．しかし当初あまりにも簡単にしてしまった機能を組み合わせてさらに複雑なことをしようと思うと，トリッキーな表現が増えるか，複雑な機能が増えてしまう．現在の表計算の機能の中で本当のエンドユーザが使用する機能は何割あるだろうか．ワークフローにも同様なことが言える．当初はメールの内容を調べて宛先を決めるだけでも満足できたが，今ではメールを条件によってサスペンドする，さらにこれをリジュームする，外部から強制的にデータを修正するなどの改良が加えられてきた．操作する人に十分な情報を与えるため外部のデータベースをアクセスする機能も備えられてきた．機能が

* EXCEL, Windows, Visual BASIC は米国 Microsoft 社の商標，Lotus 1-2-3，AmiPro は米国 Lotus Development 社の登録商標である．

増えるに従ってソフトウェアの大きさが大きくなり，複雑さを増す．さらにトラブルも増える傾向にある．

アプリケーションを開発するにあたって色々な機能をフルに使うよりもアプリケーションをできるだけ簡略化して基本的な機能でアプリケーションを開発する心構えが必要ではなかろうか．

次にスタッフウェアの概要を説明する．

2. スタッフウェア

2.1 処 理 手 続

スタッフウェアはコンピュータ処理から発展したワークフローの典型的なソフトウェアである．オフィスの処理をフローチャートで分析する．現在の作業をそのまま電子化するよりも，現在の作業を徹底して分析する必要がある．この分析を通しておのずから冗長作業が発見される．冗長作業を切り捨てると言うよりは作業に必要な情報は何か，作業員に課せられる仕事は何か，入力した結果はどのように判断して次の作業を決めるのか，などを検討すれば必要最小限の作業が残る筈である．スタッフウェアはこのようにして分析された作業をステップと呼ぶ(図1)．ステップには誰がこのステップを実行するのか(処理ユーザ)，作業員に与える情報と，入力フィールドを含む画面（フォーム)，次の作業ステップを決定する（アクション）が基本となる．このステップを連ねたもの，これを処理手続という．これがアプリケーションとなる．伝票をケースと言う．

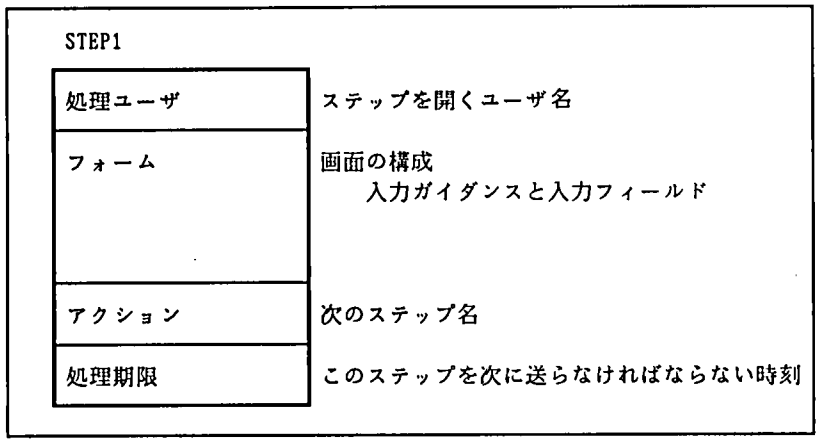


図 1　ステップの定義

フォームには作業員に情報を与えるだけの表示属性，必ず入力しなければケースを次のステップに送ることができない必須入力属性，入力を必ずしも必要としないオプション属性がある．この仕組みによって作業員は自分がしなければならない仕事を知ることができる．処理手続の開発者は作業員が何が必要かを調べ，表示属性のフィールドに情報を用意しなければならない．この開発作業はアプリケーションを良く知っているものか，あるいはエンドユーザが簡単にできるものであることが望ましい．

2.2 アクション

アクションは処理手続で定義されているフィールドの値を用いて条件文を書く．ケースの宛先はユーザ名を指定するのではなくステップ名で指定する．誰がそのステップを開くかは次に説明する処理ユーザが決める．スタッフウェアのフローの流れは人と人のつながりではなく，ステップ，すなわち作業のつながりで定義される(図2)．

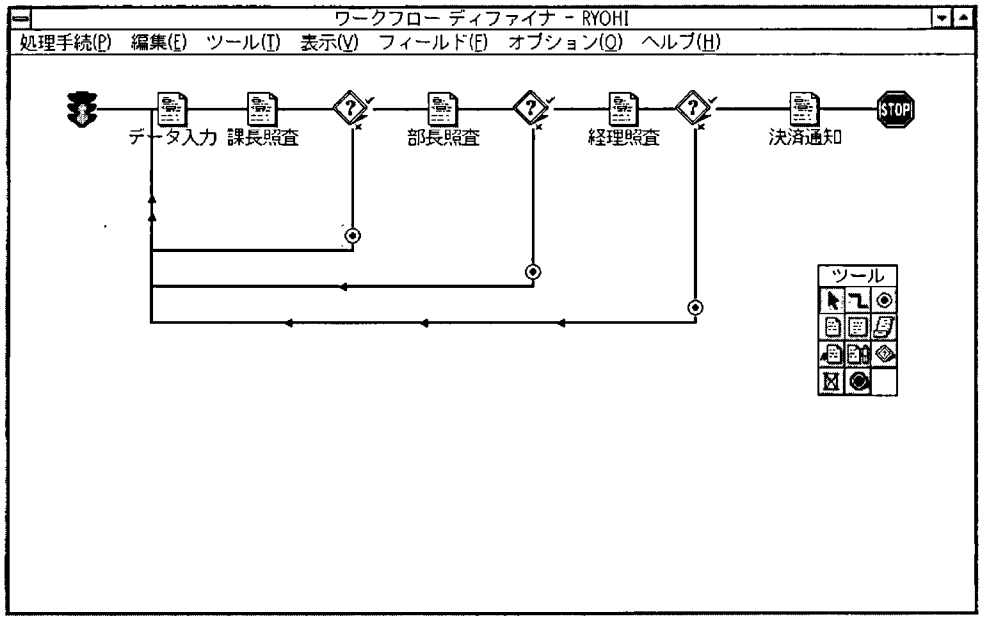


図 2 ステップの流れ

2.3 処理ユーザ

処理ユーザは3通りの方法で指定する．単純な方法はUNIXに登録されたログイン名をそのまま利用する方法である．しかし担当者が休日を取った場合，その担当者のステップで作業が止まってしまう．また同じ処理手続を別の部署で利用することもできない．2番目の方法は処理ユーザをフィールドの内容で決める方法である．フォームの中で作業員が次のユーザを選択してもよいし，入力されたデータをもとに計算で求めてもよい．3番目の方法はユーザごとに定義される役割名あるいはシステムに登録されるユーザ属性を用いる方法である．役割名とは組織の中でユニークな名前を付け，その名前にUNIXのユーザ名を登録する．ユーザ属性とはユーザの直属上司，部長などにユーザ名を登録する．これらの方法は組織が変わった場合，あるいは担当者の休日などの場合に，システム登録情報を変更するだけで処理ユーザを変更することができる．

2.4 グループ

事務処理を考えた場合，窓口で受け付けられた書類が次の部署に送られるが，窓口では簡単な書類の不備をチェックするだけで大切なチェックは次の部署で行われることがある．このような場合，窓口の人数よりも多くの人員を当てなければならないし，送られてきた書類を誰か一人が処理すればよい．スタッフウェアではこのような部署

をグループとして処理ユーザに指定することができる．ケースはグループのすべての人にキューされるが，誰か1人がキューを開くと別の人はケースを開けなくなる．

2.5 ステップ

ステップにはいくつかの種類がある．データを表示し入力するステップ，複雑なプログラムを記述するために使われるスクリプト・ステップ，処理手続のサスペンドの用いられるイベント・ステップ，処理手続で流されているケースのサマリを表示する管理レポート・ステップなどがある．

2.6 フォーム

ユーザの開く画面をフォームというが，フォームはフィールドとガイダンスで構成される．フィールドには，文字，数値，日付，時間，メモ，および添付という六つの型がある．メモ・フィールドは長さを気にしないでコメントなどを入力するために使われる．添付フィールドはケースに外部のファイルを添付するために用いられる．フィールドにはその利用目的に合わせて，必須入力，表示，オプション，非表示などの入力属性がある．

フィールド間の計算，フィールドに決まった値を入力をするための既定文字リスト，すべてのユーザで共有するリストあるいは簡易データベースなどの機能がある．フィールドから特定の外部プログラムあるいはスタッフウェアの演算を実行させるためのコマンドも定義できる．外部プログラムについては後で説明する．

これらの機能を利用すれば，おそらく事務処理の半分は表計算並みの易しさで処理手続を定義できる．フォームの定義は現在UNIX端末で行われるが近々ウィンドウズで行われる予定である．

2.7 処理期限

伝票（ケース）を受け取ったあと，このケースをいつまでに入力を終えなければならないかを設定することができる．これを処理期限という．処理期限は絶対時間，日付かケースを受け取った時刻からの相対的な時間で設定される．この時刻に達したときにまだケースをリリースしていないと警告メッセージをそのユーザあるいはシステムの管理者に送ることができる．そして必要ならケースをその担当者から取り上げて次のステップに送ってしまうことができる．

2.8 同報と同期

フロー制御について直線的に流れる，条件によって分岐する，処理の終了に期限をつけることまで説明した．次に同じ伝票（ケース）を同時に複数のステップに配布することができる．しかし，複数に分かれたステップはどこかで同期をとる必要がある（図3）．

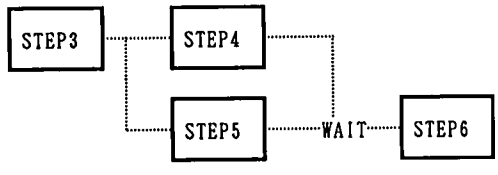


図 3　同報と同期

上の例ではSTEP 4およびSTEP 5の処理が終わらないとSTEP 6の処理は始められない．書類の審査を同時に行うが，このとき注意しなければならないのは複数の人に同じフィールドに入力させてはならないことである．1人は承認，別の人は却下した場合，あとから決済した人の判定に従って処理されてしまう．別々に決裁フィールドを用意して次のステップの人が判断するか，多数決にするかなどをアクションとして定義する．

2.9 外部プログラムの実行

フォームのフィールドから外部プログラムが実行できるほか，このステップが開かれるとき，次のステップへ送られるとき，およびケースの送信を保留し自分のワークキューに戻すときにそれぞれ異なるプログラムを実行することができる．

外部プログラムはウィンドウズあるいはUNIXどちらのプログラムでもよい．UNIXプログラムの場合フィールドの値を直接プログラムに渡し，かつプログラムの処理結果をフィールドに直接渡せるためデータベースの参照などに利用されることが多い．ウィンドウズ・プログラムの場合プログラムの実行に伴うパラメタはコマンドの一部として渡せるが，結果をフィールドに渡せないのでプログラムを実行した後DDE機能を用いてデータ交換を行う．DDEにはクライアントとサーバ機能があり，スタッフウェアはクライアントなのでデータ交換するソフトウェアはサーバ機能を有する必要がある．ウィンドウズの場合，すでに画面にソフトウェアが存在するかをチェックしないと複数の同じソフトウェアが画面に開かれてしまうことがある．このようなチェック機構を備えている．これらのコマンドは特別のステップ，スクリプト・ステップに書かれる．

DDEを用いてEXCELとデータ交換するスクリプトの例を次に示す．

```
If winexist（“Microsoft Excel”）          存在をチェック
  winactivate（“Microsoft Excel”）        あればアクティベートする
  winrestore()                            画面に表示する
Else                                      なければ
  winrun（“c：¥excel5¥excel. exe”, 1）     EXCELを実行
EndIf
ddeinitiate（chan, “EXCEL”, ” ”）         dde開始
ddepoke（chan, “R 1 C 1”, FieldA, 1）     EXCELへデータを渡す
dderequest（chan, “R2C2”, FieldB, 1, 5）  EXCELからデータを受け取る
ddeterminate（chan）                      dde終了
```

2.10 処理手続のデバッグ

このようにして定義された処理手続は当初開発モードでデバッグされる．デバッグは開発担当者の端末がすべてのユーザのワークキューをシミュレートするので実際の運用環境でも行うことができる．デバッグが終わった後処理手続を一般ユーザにオープンする．

2.11 ケースの発行

処理手続は通常誰でも使うことができるように作られるが，ケースをスタートする

ユーザを制約することもできる．ユーザがスタッフウェアのウィンドウズ版を起動するとツールが表示される．この中から処理手続開始をクリックするとユーザに利用が許されている処理手続が表示される．ケースには1件ごとにケース識別名を入力することができる．ケース識別名に何を入力するかはスタッフウェアでは制限していない．ワークキューで自分に送られたケースを見るときにソートする対象となるので分類に用いると便利である．

一つの処理手続をクリックすると処理手続の最初のステップのフォームが表示される．フォームの中にあるフィールドの枠が色で区別され，赤色は必須入力，青色はオプション，色のないのは表示フィールドである．必須入力フィールドにすべて入力が終わると画面の右上の白い紙が封筒に入れられケースが送ることができる状態になったことをユーザに知らせる．この封筒をダブルクリックするとケースが次のユーザに送られる．ケースにはケースごとにケース番号が付けられる．

2.12 ワークキュー

ツールの中にワークキューというアイコンがある．または画面の下に白い紙が重なったアイコンが表示されていることもある．これをクリックすると自分にキューされているケースの一覧表が見られる．白い紙の間に赤い紙が挿入されているときにはまだ開いていない新たなケースが到着していることを意味する．ケースの一覧表から必要な行をダブルクリックするとケースが取り出されフォームに表示される．白い紙に時計のマークが表示されているときには処理期限を持ったケースがキューされていることを表す(図4)．

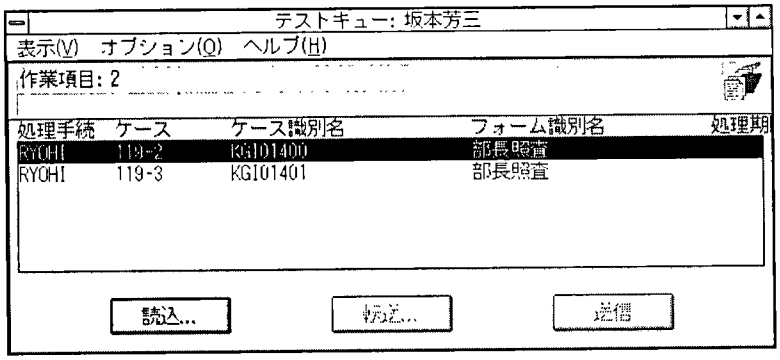


図 4　ワークキュー表示

2.13 処 理 追 跡

発信したケースがいまどこで処理されているかを処理追跡で見ることができる．このときケース番号が必要である．処理追跡情報はケースが発行されてから，誰にケースがキューされ，いつリリースしたか日付と時間が表示される．どのステップで誰がケースをリリースしないでいるかが分かる．処理追跡機能は管理者向けに用意されている(図5)．

2.14 ケースの転送

あらかじめ定義されたワークフローの流れは基本的に変えることはできないが，ス

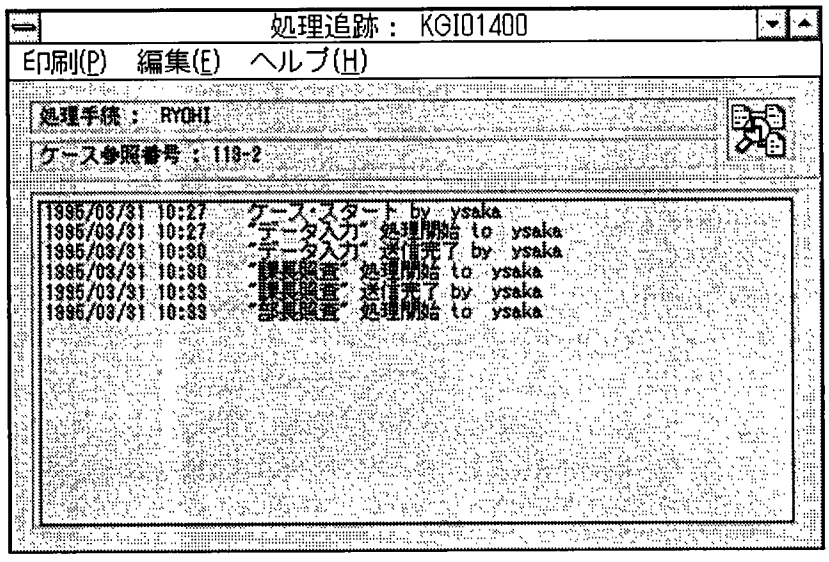


図 5 処理追跡表示

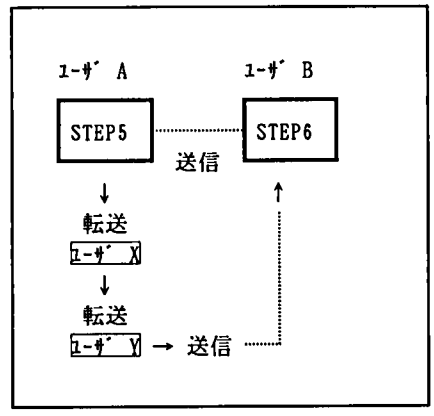


図 6 ケースの転送

テップごとに転送を許可することを指定すれば一時的にケースを他のユーザに転送することができる．現在表示されている画面そのものを別のユーザに渡してしまう．転送されたユーザがケースを送信すればあらかじめ決まっているステップにケースがキューされる．さらに別のユーザに転送することも可能である(図6)．

2.15 添付ファイル

添付ファイルはケースの移動とともにウィンドウズのアプリケーション・ファイルを次のユーザに送る機能である．ファイルを添付する最初のユーザは添付フィールドに対象とするファイルの名前をフルパス名で指定する．次のステップに送られるときにファイルはスタッフウェアのユーザ共通ディレクトリにコピーされる．スタッフウェアの構成ファイルに定義しておけば，このファイルをアクセスできるユーザはケースが送られたユーザのみとすることができる．

添付ファイルとフィールド・コマンドを組み合わせて送られてきたファイルをソフトウェアを実行して読み出すことができる．このソフトウェアには制限がなく，添付されるファイルの内容をスタッフウェアは感知しないので表計算，ワープロ，あるいはイメージを扱うことができる．添付ファイルは複数扱うことができる．

2.16 ユーザ・カスタマイズ

ソフトウェアが使われ始めると，ユーザからの要求によって機能が徐々に追加されるようになる．便利であればあるほどさらに機能追加が要求される．しかし，すべてのユーザの要求に合ったソフトウェアはかなり重たいソフトウェアになってしまうことが多い．ユーザの要求に応えるソフトウェアに機能を追加するためのインタフェースを提供することも一つの方法である．

スタッフウェアに寄せられる機能追加は本来スタッフウェアが有すべき機能であることも多い．しかし，スタッフウェアでは次の方法によってユーザによるカスタマイズを可能にしている．

2.17 オープン・フォーム

スタッフウェアが提供するフォームでは飽き足らないユーザに対してはオープン・フォーム機能を提供する．オープン・フォームはウィンドウズの API の一部として外部ソフトウェアとデータを交換する機能とウィンドウズ・プログラム同志がデータを交換する DDE 機能，ステップのオープン時に外部ソフトウェアを実行する機能を組み合わせて使われる．スタッフウェアのフォームを外部のソフトウェアが作る画面に置き換えてしまう．

ステップが開かれるときスタッフウェアはケース画面を表示しないで外部のプログラムを起動する．DDE 機能を用いて必要なデータを外部プログラムに渡す．あとは外部プログラムがスタッフウェアのデータをいかようにも処理できる．最後に外部プログラムはスタッフウェアにデータを返さなければならないが，このとき DDE サーバはクライアントに DDE を用いてデータを送ることはできない．外部プログラムがウィンドウズの API を用いてスタッフウェアのデータファイルに直接データを書き込みにいく．

現在動作が確認されている外部ソフトウェアは Visual BASIC および ACCESS である．

2.18 SWUTIL

SWUTIL は二つある．一つは UNIX コンソールあるいは端末から実行されるものであり，もう一つは Windows のコマンドラインから実行される．

UNIX の SWUTIL はスタッフウェアの管理者向けのユティリティである．いくつかのサブコマンドがある．EXTCD というコマンドではスタッフウェアのケースデータをすべて覗くことができる．したがって処理手続を開発するときにケースがどこに送られたか，ケースを起動したユーザは誰かなどの情報をフィールド情報として処理手続の中に組み込めば，ケースがどのステップまで処理されたか，承認の途中結果などを見ることができる．さらに工夫すれば今日の日付とケースの中の納期のフィールドの値を較べ，納期までの期間が 30 日を割っているケースを取り出して詳細を表示するようなプログラムを開発することができる．

SWUTIL の UNIX および Windows に共通した機能として，実行中のケースの流れをコントロールしたりデータを強制変更する機能がある．

2.19 外部イベント

ワークフローの流れを外部から変える仕組み，あるいはすでに入力されているフィールドの値を強制的に変える仕組みも必要になる．例えば為替レートのように日々一定の時刻に決められる値がある(図7(a))．あるステップまで処理されたケースがサスペンドされ，今日の為替レートで計算され次のステップの送るような処理手続を作ることができる．このため，ステップをサスペンドする機能，外部からフィールドの値を変える機能，サスペンドを解く機能が提供される．ケースをサスペンドし，時刻に達したらサスペンドを解く機能はイベント・ステップに処理期限を組み合わせたステップを定義する．為替レートが午前10時に決まるとすればこのサスペンドを解く時刻を10時20分とし，為替レートが決まりデータを変更する作業をこの時間までに終わらせればよい．外部からデータを変更するにはスタッフウェア・ウィンドウズ版に別の SWUTIL コマンドが提供される(図7(b))．

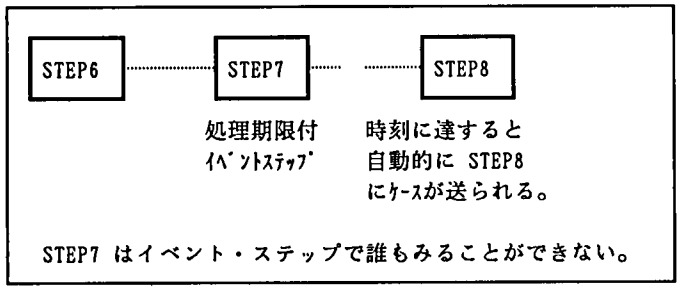


図 7(a)　送信期日指定

図 7(b)　データの強制的な変更

イベント・ステップを処理期限なしに定義すると，外部からの起動がないといつまでもケースをサスペンドすることができる．外部のプログラムの実行に合わせてケースの流れを制御するために用いられる（図8）．

2.20 複数の処理

一つのケースで同時に複数の流れを作ることができる．処理手続を開発するときに複数のエントリ・ポイントを定義する(図9)．

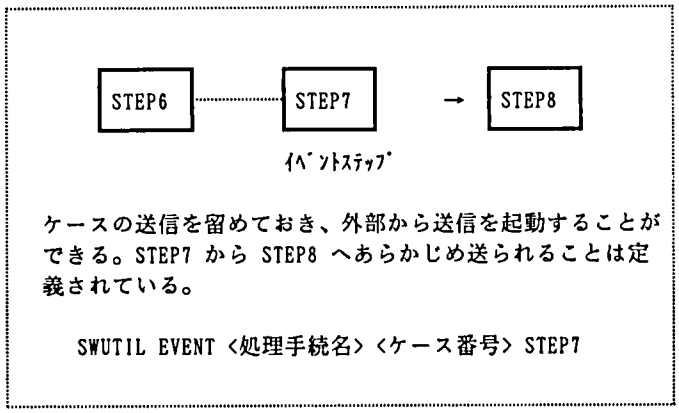


図 8　ケースのサスペンドとリジューム

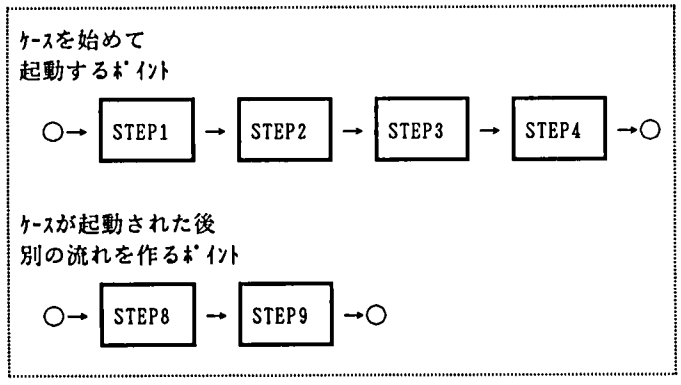


図 9　一つのケースで複数の流れ

処理手続のケースを始めて起動する場合，最初のステップは一つに限られる．処理手続の中に別のエントリ・ポイントを定義することができるが，ここからの起動はケースが起動された後，外部プログラムから SWUTIL コマンドを実行して起動する．このとき実行中のどのケースに別の流れを発生させるかをケース番号で指定しなければならない．両方の流れは別々のフィールドにデータを入力するが，互いに別の流れで入力された値を参照することができる．同じフィールドを両方の流れで入力すると常に最後に入力した値がファイルに残され，別の流れで入力した値の上に書き込まれてしまう．両方の流れは同期機能を用いて一緒にまとまってもよいし，そのまま別の流れとして終わってもよい．

3. システム構成

現在スタッフウェアを利用するためのシステム構成は次の通りである．

サーバ	ユニシス	U 6000/100, 300, 500, 65, 35	SVR 4　V 1.1.1 以上
	ユニシス	US シリーズ	日本語 Solaris 2.3 以上
	HP	HP 9000 700/800	HP-UX　9.0 以上
クライアント		DOS/V 互換機 PC 9801（限定機種） Windows 3.1, VGA が稼働できる環境	
ネットワーク		TCP/IP, PC-NFS	

ハードディスク装置およびメモリ容量については，同時に使用するソフトウェアと関係するので問い合わせして頂きたい．

4. ユーザ・アプリケーションの考え方

これまでスタッフウェアの機能について説明してきた．次にスタッフウェアを適用するアプリケーションについて考察する．

ワークフローのソフトウェアは決してオンライン・トランザクションの世界に向いてはいない．すなわち秒を争うシステムではない．書類が起票されてから処理終了まで数日かかる．そして書類が滞らないように監視が必要なアプリケーションに向いていると言える．現在 BPR という言葉がもて囃されているが，ワークフロー・ソフトウェアを導入することがすぐに効率向上に結びつくのではない．あくまで現在の書類の流れの分析と徹底した冗長作業の排除があって成功するものと確信する．事務処理を見直すきっかけとしてワークフロー・システムを導入することは極めて自然である．現在行われている事務作業の分析は，システム部門が行うのかあるいはアプリケーションに精通したエンドユーザが行うのかは，導入する部門の性格によってどちらとも言えない．どちらが危機感を持っているかが問題なのであろう．

システムの構築面では，アプリケーションのすべてを一つのソフトウェアに頼ってしまう傾向がある．とくにワークフローのようなソフトウェアは，ビジネスの基本的な機能が網羅されているのであらゆる面で使われることが多い．しかし，秒を争うようなトランザクション・システムやフィールド間の計算ができるとしてもスタッフウェアの機能でウィンドウズで扱われる専門の表計算と同じことを処理させるのは難しい．簡易データベースがあると言っても膨大な選択まではできない．文字を入力できてもワープロとしては使えない．これらの機能を外部のウィンドウズ・プログラムを添付ファイルとして積極的に取り入れることが大切である．すでに述べたが，ワークフローはレターヘッドとなってしまってもよいのである．

5. お　わ　り　に

スタッフウェアの機能を説明してきた．ワークフローと言われる分野はまだ市場を席巻するメーカが存在しない．またソフトウェアの目的とする機能，別な面から言えばその発生源の違いから色々なソフトウェアがある．我々がこれを取り上げた原点はビジネスの流れがフローチャートとして表され見やすいこと，および基本機能を取り上げたとき誰にでもアプリケーションを開発できる簡易さであった．すでに説明したように高度のコマンドを利用すれば，さらに複雑なことも実現できる可能性も十分備えている．現在のエンドユーザではマイクロソフトの EXCEL あるいはロータス 1-2-3 のマクロを書く実力をすでに持っている方々は，スタッフウェアのスクリプトを十

分に書けるものと思う．

しかしいくつか注意すべき点を挙げるとすれば，クライアント・サーバシステムはUNIXを基本としているためUNIXの知識を必要とすること，および扱うツールの性格，能力を見極めた上，アプリケーションに適用していくことであろう．

執筆者紹介　戸　叶　孝　一（Koichi Tokano）

昭和37年東京都立大学電気工学科卒業．44年沖電気を経て日本ユニシス(株)入社．ミニコンピュータ・ソフトウェアの開発，国外パーソナル・コンピュータ・ソフトウェアの日本語および商品化に従事．主な導入商品は，マイクロレポ，スーパーレポ，TK! Solver, PDS-Adept(BTOS)．現在，オープン技術本部 オープンソリューション部所属．

クライアント/サーバによるA社MP事業本部システムの再構築

The Reconstruction of a Client/Server System for Company A's MP Business Operations

門本光央, 石倉哲善, 林　健一

要　約　A社でのメインフレーム (A 19), UNIXサーバ (US 120), EWS (AS 1000), PCのクライアント/サーバによるシステム再構築の事例を紹介する.

システム再構築のポイントは, 既存のシステム資源と新しいシステム資源をうまく調和させるためにどのようなソフトウェアを使用すべきかということであった.

新システムは, エンドユーザにとって使い勝手が良く, 物理的にスペースを取らないEWSとPCを使用し, 開発ツールとしてTIPPLERを用いることとした. また, メインフレームではLINCを用いてアプリケーション開発を行い, データの一元管理を図るように工夫した. そして, これらを統合するミドルウェアとしてUNIshuttleを採用しシステムの再構築を実現させた.

本稿では, システム再構築の経緯とTIPPLER, UNIshuttle, LINCでの開発方法についての考察を述べていく. 今後の開発方法を考えていく上で参考になればと思われる.

Abstract　This paper is intended to present a case where a client/server system has been reconstructed at company A, for which a mainframe (A 19), UNIX workstations (US 120s), EWSs (AS 1000s) and PCs are configured. The crux of this system creation was the choice of an optimum software tool to ensure a good integration of existing system resources and new ones.

For the new system, made up of user-friendly EWSs and PCs both requiring less physical space, applications development took place with the help of TIPPLER, the integrated applications development support tool. For the mainframe having total control of centralized data, applications were developed with the use of LINC. For the linkage of the host mainframe and the EWS/PC equipment, UNIshuttle was used as a middleware tool.

This paper discusses how the new system has been developed in addition to how TIPPLER, UNIshuttle and LINC were used for the development. The author hopes this report will be of some help for readers in considering their future systems development efforts.

1. はじめに

A社のMP（ミート・パッカー）事業本部では, 輸入肉と国内肉の取引相場を把握しながら食肉の売買を行っており, その業務を支援するシステムとして情報検索システムと窓口支援システムがある.

情報検索システムは, 販売実績や在庫状況, 需給バランス等を把握し, 相場等の動向を睨みながら生肉輸入と販売という商売の方針決定を支援する業務である.

一方, 窓口支援システムは, 現場ディーラーの取引（受注, 発注, 引当）を直接支援する業務である.

しかし，従来の情報検索システムは検索できる情報量が少なく，利用する部門も人も限られており，ほとんどがディーラーの勘と経験に頼っていた．

また，窓口支援システムも完全にリアル処理化されておらず，手作業に頼っている部分もある．

上記のような課題を解消するためにシステム再構築にあたって，次の二つのポイントに重点をおいた．

第1は，エンドユーザにとって使い勝手の良いGUIによるオペレーションと大量の情報をただちに検索できるシステム作り，第2は，直接商売に結びつくディーリング処理をタイムラグなく実現できるリアルタイム処理化である．

本稿では，まず2章で，MP事業本部システムの概要を紹介し，3章で既存のシステムの説明と既存システムに対する問題点やそれに対するエンドユーザの要求をもとにシステムの再構築の経緯を説明している．次に，4章では新システムの構成を，5章では新システムの構成要素であるTIPPLER，UNIshuttle，LINCの観点から見たシステム構築のポイントおよび評価をまとめている．6章では，全体のまとめと今後の課題を述べている．

2. MP事業本部システムの概要

MP事業本部基幹システムは，輸入契約管理/為替管理，仕入統計管理/買掛金管理，在庫管理(外冷(外部冷蔵倉庫)/先物)，販売統計管理，債権預信管理から成っており，それらの支援システムとして情報検索システム，窓口支援システムが提供されている．

情報検索システムは，販売実績，在庫状況，需給バランスの検索に使用され，それらの情報をもとに生肉輸入と販売に対する商売の方針が決定される．検索情報は，全社レベル，部門レベル，担当者レベルのそれぞれで活用されている．

窓口支援システムは，受注システム，発注オーダーシステム，外冷引当システム，先物契約システムから成っており，現場での取引を直接支援している．つまり窓口業務担当者が，前年/当年相場，中間相場などの情報を得てトレンドを分析すると同時に

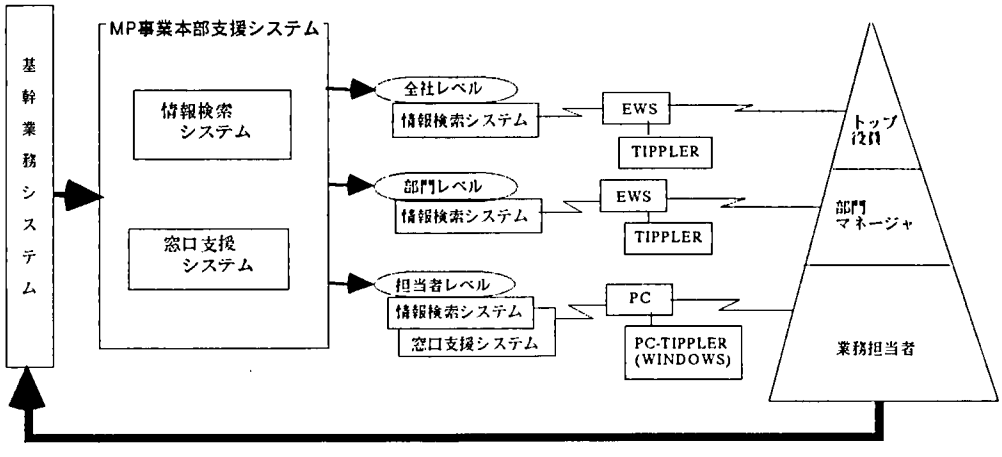


図1　MP事業本部システムの概要

在庫情報を参照して，どの在庫を売りに対応させるかのチェックをしながら売買に関する処理を行っている(図1参照)．

3. システム再構築の経緯

3.1 システム再構築の背景

MP事業本部のシステム再構築の背景には，既存システムにおける次のような課題やエンドユーザの要求があった．

システム共通の課題として，ダム端末で操作を行っているため，GUIのような操作環境と比べて，利用者がコマンドに精通した人に限られている等必ずしも使い勝手の良い操作環境ではなかった．

情報検索システムの課題としては，検索できる情報が限定され，情報量としても満足いくものではなく，エンドユーザの業務における方針決定や問題点の発見は個人の資質や能力に依存していた．

また，窓口支援システムにおいては，現場のディーラーの直接取引に結びつくディーリング処理にオンラインバッチ処理的なものが多く，完全にはリアル化されていなかった．そのためディーラーが最新のデータを用いて業務ができない等の点である．

3.2 システム再構築のポイント

前述の背景と課題に対して，次の5項目を再構築のポイントとした．

① 検索はメニュー方式とし，グラフや表を利用してGUIによる扱いやすい操作環境を実現する．

② 多角的な情報検索を可能とし，需給バランスを最適化するためのシミュレーション機能や最少二乗法での計算結果でグラフを作りこれを利用した不良在庫の削減，商品回転率をチェックして警告を促すためのアラームシステムの実現により，個人の資質や能力だけに依存しない方針決定支援システムを構築する．

③ トップ役員・部門マネージャが頭を悩ます，販売実績・在庫状況・生産状況などの膨大な情報検索とその加工を即時にできるようにする．

④ ディーラーが直接取引に結びつくディーリング処理のタイムラグをなくし，一元管理されているメインフレーム上の最新データの即時検索および即時更新処理を実現する．

⑤ データの即時検索，即時更新が求められる中でメインフレーム（A 19）の負荷を分散させるためにUNIXサーバ（US 120）上にORACLEデータベースを構築し，EWS（AS 1000），PC等相互の接続を可能とする環境を実現する．

次に今回の再構築に当たり，情報検索システムの再構築をフェーズ1，窓口支援システムの再構築をフェーズ2とした．

1) 情報検索システムの再構築（フェーズ1）

利用頻度が高く，膨大なデータの検索を行わなければならない販売実績データや在庫データを夜間のバッチ処理でA 19上のデータベースから抽出し，ファイル転送でUS 120に送り，ORACLEデータベースを構築する．そのデータを活用してGUIを駆使したTIPPLERアプリケーションで各種の情報検索処理を実現させる．

2)　窓口支援システムの再構築（フェーズ2）

更新処理を含む受注システム，発注オーダーシステム，外冷引当システム，先物契約システム等の窓口支援システムのリアルタイム処理を実現するためにメインフレーム（A 19）上の在庫，取引情報，相場データベースをEWS/PCのGUI環境より直接参照および更新を行う．また，マスタ情報等の静的なデータはUS 120上のORACLEデータベースに持たせて分散処理を実現する．

なお，フェーズ2の本番開始は，1996年3月に予定しており，本稿の執筆時点ではプロトタイプ版の開発を完了している．

4. 新システムの構成

4.1 EWS/PCの役割

新システム構築のためには，EWS/PCとA 19との連動が必要となり，EWS/PCの役割は次のようになる．

EWS/PCを用いることによりエンドユーザにとって使い勝手が良く，物理的にもスペースを取らない環境を実現する．EWSとPCの使用形態は，図1のようにトップ役員・部門マネージャは大量の情報検索を行うために高解像度で大画面のEWSを使用し，業務担当者は多くの部署での窓口業務を行うためにコスト面と実用性を考慮してPCを使用することとした．

4.2 フェーズ1におけるシステム構成

フェーズ1は，A 19，US 120，AS 1000，PCをLANで接続し，US 120をデータベースサーバ，AS 1000，PCをクライアントとするクライアント/サーバシステムである．

メインフレーム（A 19）のデータベースから抽出プログラムを用いてシーケンシャルファイルを作成し，ファイル転送（FTP）でUS 120にデータを転送する．そして，US 120上のORACLEデータベースにデータをロードしTIPPLERで作成されたアプリケーションを用いて検索処理を行っている．

図2は検索処理形態を，図3はソフトウェア構成を示している．

4.3 フェーズ2におけるシステム構成

メインフレーム（A 19），EWS，PCをミドルウェアであるUNIshuttleを用いて統合する．A 19上では，LINCを用いて更新処理のアプリケーションを開発し，LINCの無編集入出力機能（LINC-NOF機能）でUNIshuttleとインタフェースをとる．

図4は更新処理形態を，図5はソフトウェア構成を示している．

4.4 TIPPLERの採用

TIPPLERは，知的活動支援システムを構築し，実行環境を提供するシステムであり，経験やデータを知的化し，組織としての活用を支援する．また既存の業務系基幹システムや情報系システムとのデータインタフェースを図りながら，使う側の意思や直感を操作につなげるようなユーザインタフェースを持ち，使い勝手の良いシステムを超短期間で開発できることを可能にした＜プラットフォームソフトウェア＞である．

システム利用者の要求であるマウスを使用した「ボタン操作」，マルチウィンドウ環

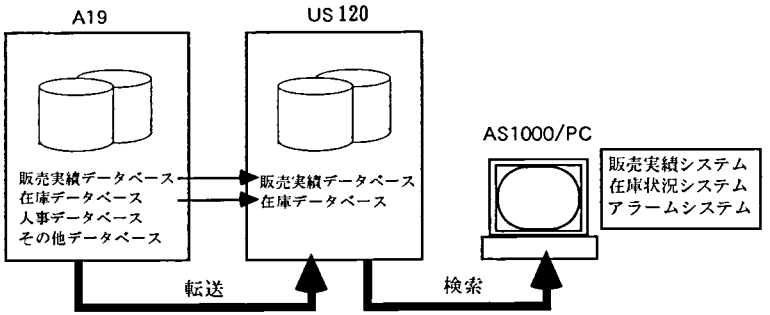


図 2　データの流れと検索処理形態

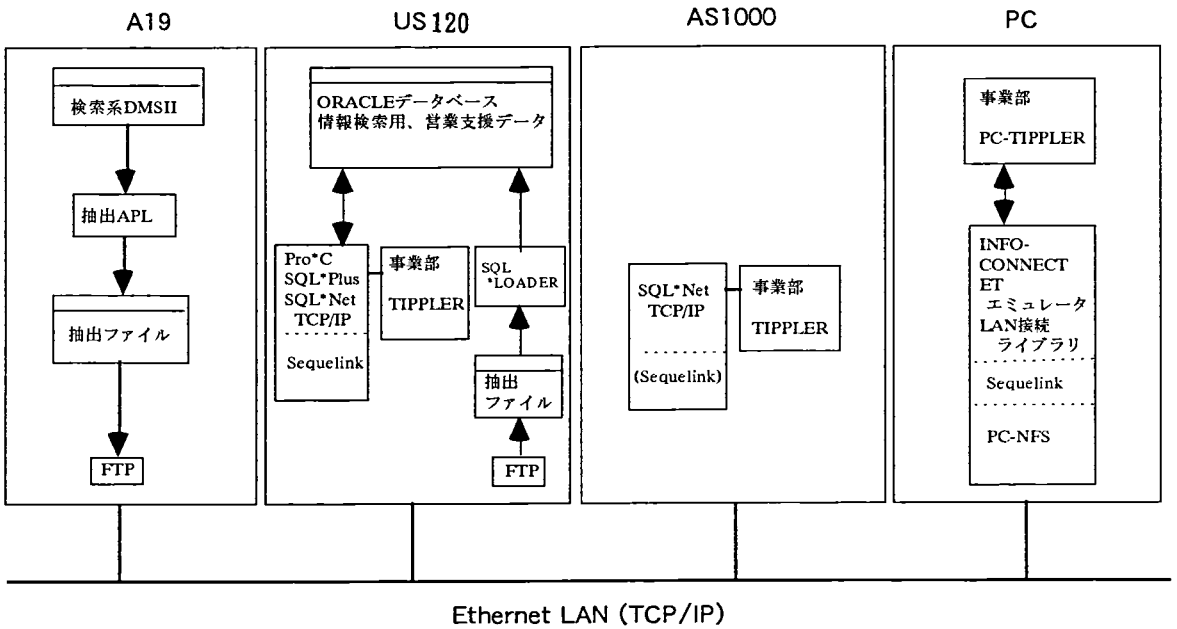


図 3　フェーズ1のソフトウェア構成図

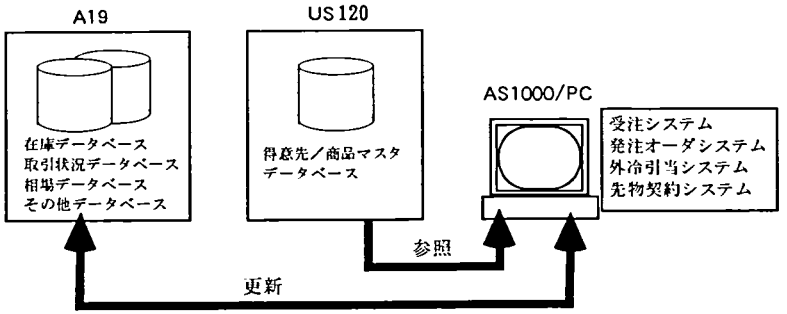


図 4　データの流れと更新処理形態

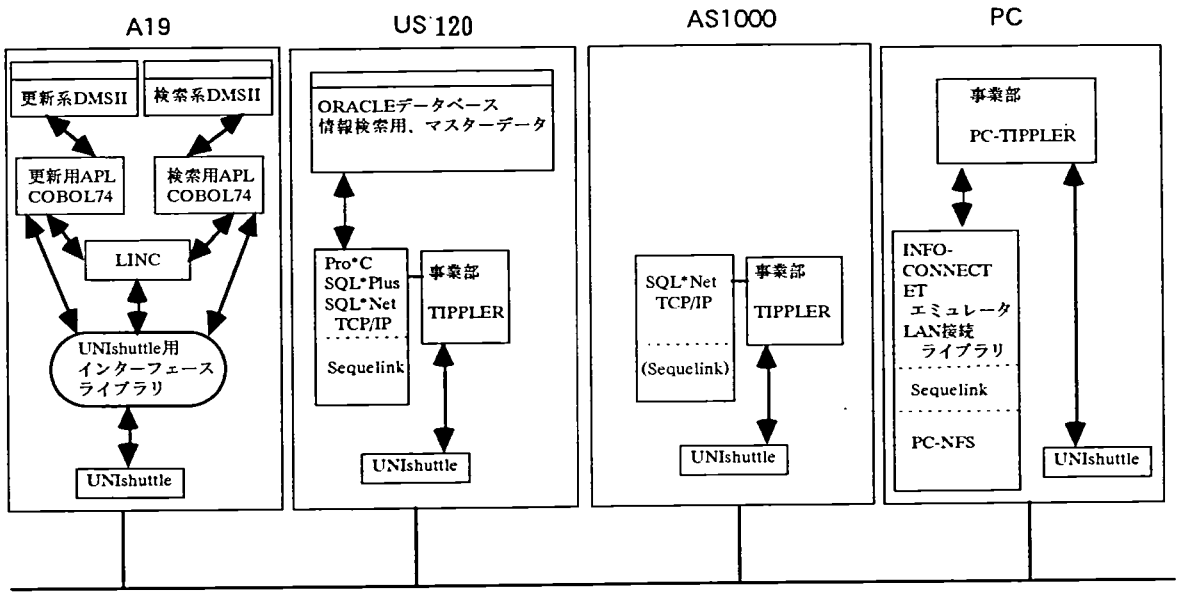


図 5　フェーズ2のソフトウェア構成図

境，グラフ表示等のGUI環境を提供し，思考プロセスにそった情報の掘り下げが可能となるシステムを構築できる．以上の理由でTIPPLERを採用することにした．

また，EWSとPCで同じオペレーション環境とアプリケーションインタフェースを提供できることも採用に当たっての大きな理由である．

4.5 UNIshuttleの採用

UNIshuttleとは，異機種間接続を可能とする分散システム構築支援ツールである．

分散化を行う上でメインフレーム，EWS，PCそれぞれの特徴を活かしたシステム作りが必要不可欠であり，異機種間のネットワーク環境を支援するプロダクトとして，各機器から利用できるAPIを持つUNIshuttleが最有力候補にあがった．さらに異機種間の複数サーバ/クライアントに対する運用および管理面でも同一環境として扱えることからUNIshuttleを採用することにした．

4.6 LINCの採用

LINC（Logic & Information Network Compiler）とは，LDL（Linc Definition Language）単一言語を使用し，LINCコンパイラと対話形式で総合的な情報システムを生成する4GLソフトウェアである．

LINCを用いることにより，高い生産性が得られ，簡単なコーディングでデータベース同期リカバリが可能となる．また，ユーザがLINCに精通している，Aシリーズの LINCアプリケーションをすでに構築していた，ユーザからの強い要望があった等により LINCを採用することにした．

5. 新システム構築の実現と評価

新システムの構築に当たり，TIPPLER，UNIshuttle，LINCを採用することによりユーザの要求の実現が可能となった．これらを用いた開発時の注意点等の説明を次に行う．

とくに，本章では当社初の試みであるUNIX版TIPPLERからPC版TIPPLERへの移行とAシリーズでは初めて使用したUNIshuttleに関して重点的に述べることにする．

5.1 UNIX版TIPPLERからPC版TIPPLERへの移行

フェーズ1ではAS 1000で情報検索システムの構築を行ったが，PC上でも同一アプリケーションを稼働させる必要があったため，PC上でTIPPLERを使って開発する時の互換性を検討した．

TIPPLER for WINDOWSへの移行対象システムとして在庫状況検索システムを選んだ．この理由は当システムがORACLEデータベースを持つUS 120サーバとPCとの典型的なクライアント/サーバシステムであり，ORACLE SQL * NetのAPIを使用したTIPPLER（Open-LOOK）のアプリケーションであったためである．

PC側は，ミドルウェアとしてSequelinkを採用した．この理由は，TIPPLER for WINDOWS β版が出荷された時点で，外部データベースとのインタフェースを行うためのAPIは唯一SequeLinkであったからである．

5.1.1 移行の実現性

移行作業で発生した問題や互換性について以下に記述する．

1)　開発環境と実行環境の区別

PC用の実行モジュールを作成（make）する場合（WINDOWSのDOSプロンプトからの実行）に注意しなければならない点は，コンベンショナルメモリを十分空けておく必要があるということである．

今回の作業では，外部データベースをアクセスするためにミドルウェアやネットワークソフトウェアが常駐するため，コンベンショナルメモリが大量に使用される．また，一台のPCに開発環境と実行環境を持つと，さらにコンベンショナルメモリ不足となるために環境を分ける必要があった．

2)　64Kの問題

TIPPLERの記述言語であるUNIscriptは，C言語に変換されてコンパイルされるが，DOS（16ビット）端末はコンパイル時に（コードセグメント，デフォルトセグメント共に）64Kを越えてはならないという制限があり注意が必要である．これを回避するには，UNIscriptの各ソース（*．us）を分割しなければならない．

個々のソースファイルを作成する際は，コンパイルした結果，自動的に生成されるファイル（usmain.c，usf.c，usi.c，usv.c）のステップ数の合計が10,000ステップ以下になるようにする必要があり，そのためのおおよその目安として，virtual宣言を利用して個々のソースファイルのステップ数を約1,000〜2,000ステップ以下になるよう分割する．

3)　互換性について

TIPPLER for WINDOWSは，シンタックス上で引数の一部が使えないなどの細かい制限はあるが，基本的にUNIX版のTIPPLERで使用しているものとは互換性がありUNIXの資産を活用することができる．

4)　画面の大きさと解像度の問題

ノート型PCを使用する場合，解像度が小さいことにより，画面で表示できる情報量は少なくなる．TIPPLERの画面設定ファイル(tippler.ini)のdefaultfont(高さ，幅）を小さく設定することで，ある程度は画面で表示できる範囲を広げることが可能にはなるが，A社の要望は満足できなかった．

今回こういった制限を解消するためにC社製の大型液晶フラットパネルディスプレイFLCD（1280×1024ドット，15インチ）を表示装置として採用し，本体は当社のPCを使用することにした．

5)　ソースの一元管理について

UNIXからの移行については，ほぼ互換性が保たれているが，変更を全く必要としないわけではない．開発完了後のアプリケーションの変更・保持を考慮すると，PCとUNIXとの間でソースの一元管理をすることは困難であり，個別にソース管理することにした．

今回の移行作業においては，基本的にシンタックスチェックおよび機能テストまでをUNIX機で行い，画面レイアウトの調整および単体テスト以降はPC側で行うことで効率的に作業ができた．

5.1.2 TIPPLER for WINDOWS 移行時の考慮点

1) 画面レイアウトの変更

単純に PC に移行すると，各アイテム(panel，field 等)の間隔が広がり，UNIX で作った時のレイアウトと違ってしまう．

これは，GUI の相違に原因があり，UNIX 版 TIPPLER の場合でも Open-LOOK と Motif 間でこの現象が発生する．UNIX 版 TIPPLER では実行時にオプションを付けることによって回避しているが，TIPPLER for WINDOWS ではロジック(X, Y 座標指定) 変更が必要となる．

2) 外部データベースインタフェースの相違による変更

PC 側の Sequelink API では，アプリケーションで使用しているデータベースのテーブルフォーマットを明示的にロジック内に定義する必要がある．これは，UNIX 側の ORACLE SQL * Net API を使用する場合にはなかったことである．

3) UNIscript 作成時の制限

現在の TIPPLER for WINDOWS には，前述の C 言語ソースステップの制限に加えて，UNIscript において下記の制限がある．

- 総ステップ数は 30,000 ステップ以下
- クラス数，スロット数の和は 3,000 以下（ただし，継承した場合はそのスロット数も数える）
- メソッド，フレーム，メニュー，デーモンの総数は 1,800 以下
- 列挙型定義の数×4＋フレーム名の数は 1,000 以下

5.2 UNIshuttle のトランザクション転送機能とプログラム間通信機能の比較

EWS/PC からメインフレーム上のデータを直接検索する処理を実現するために UNIshuttle の機能であるトランザクション転送機能とプログラム間通信機能を用いてそれぞれの利点，欠点の評価を行った．

構成は図 6 のように，EWS/PC の TIPPLER で作成されたアプリケーションから A シリーズの検索プログラムを介してデータの検索を行うものである．

トランザクション転送機能，プログラム間通信機能ともにノード間のセッションを確立して電文を転送し，指定したプログラムを起動し，その結果を受け取る仕組みになっている．

この二つの機能の最大の違いはセッションの確立の保持である．

トランザクション転送機能が処理ごとにセッションを確立するのに対して，プログラム間通信機能は一度セッションを確立するとその後のアプリケーション間の会話は継続して行える．しかし，セッションを確立したままなので，同時実行端末台数が増えるというリソース面での問題は生まれてくる．

実際にテストを行った時のパフォーマンスを以下に記述する．

テストは，TIPPLER からデータベースを検索するキー項目を A シリーズに転送し，それに対する検索結果を TIPPLER に戻すようにした．なお，A シリーズ側の検索プログラムはどちらも数 10 ミリ秒程で処理が完了した．

その結果は，

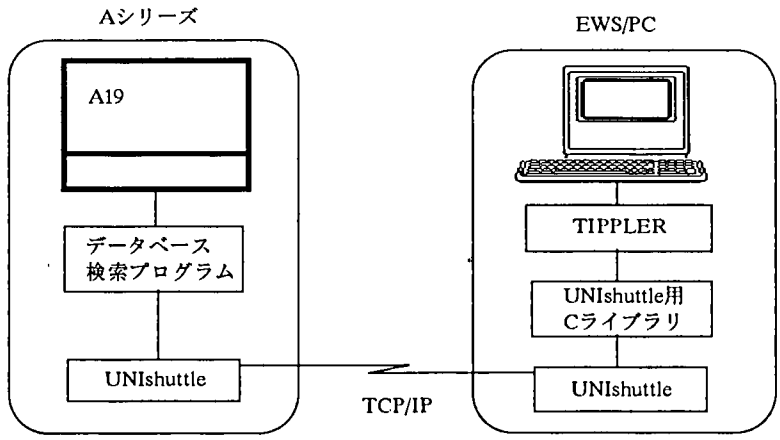


図 6　評価システムの構成

トランザクション処理　…　15〜20秒
プログラム間通信　　…　4〜6秒

であった.

ただしプログラム間通信機能を使用したケースは，セッションを確立した後の結果である.

トランザクション転送機能は処理ごとにノード間のセッションを確立する処理があるために10数秒程度時間がかかってしまうことが判明した.

プログラム間通信機能は，一度ノード間のセッションを確立してしまうとそれを保持し，その後のアプリケーション間での会話は自由に行うことができるので，トランザクション転送機能と比べるとこの時間が短縮される.

今回は業務システム形態がリアルタイムであるため，やはりプログラム間通信機能を用いてアプリケーションを開発する方針に決定した.

しかし，一回の会話で許されている電文の長さは，トランザクション転送機能の電文長が7000バイトに比べプログラム間通信機能は3000バイトである．したがってTIPPLERで作成された情報量の多い画面の場合では，複数回のやり取りが必要となり，結果的に処理が遅くなる場合があるため効率改善策の検討の必要があるであろう.

5.3 LINC-NOF機能

フェーズ2のAシリーズ上の環境を構築するためには，参照および更新処理を行うアプリケーションを作成しなければならない．そのアプリケーションをCOBOL等で作成する場合には，データベースの排他制御や端末管理を考慮する必要がある．しかし，アプリケーション作成にLINCを用いることにより，データベースの排他制御や端末管理は意識することなく行うことができる.

また，ユーザがLINCに精通しておりAシリーズ側のアプリケーション開発をLINCで行いたいという強い要望があったため，LINCを用いたシステム開発を進めた.

UNIshuttleのプログラム間通信機能は，COMS（Communication　Management

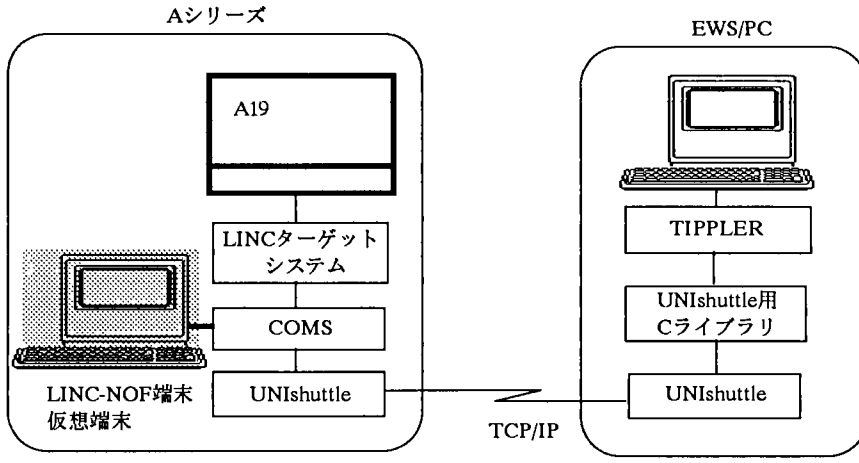


図 7　LINC を用いたシステム構成

System；A シリーズの通信制御システム）に対応しており，LINC の無編集入出力（以下：NOF）機能を使用することによりフェーズ 2 のシステム環境（LINC～UNIshuttle～TIPPLER）を構築することが可能となる．COMS に対応していることで，各ノードより複数の処理が発生してもメインフレーム側は 1 本のプログラムで対応できる．その時の構成は，図 7 のようになる．

LINC の NOF には無編集端末に対するインタフェース機能があり，表示項目や画面編集用の端末制御文字を含まないデータを無編集端末と LINC システム間で授受できる．

今回使用する無編集端末（LINC-NOF 端末）はメインフレーム内の仮想端末で，業務システム環境にサインオンしている 1 台の端末として扱われ，実際には無編集のデータは，TCP/IP で接続された EWS/PC の TIPPLER アプリケーションに対して送られる．

LINC で作成されたオンラインプログラム（ターゲットシステム）をサーバプログラムとし，TIPPLER で作成したプログラムをクライアントプログラムとするこの構成では，従来の LINC の画面が EWS/PC の TIPPLER で作成された画面に置き換えられる．

LINC を用いたときの長所として UNIshuttle を意識したコーディングの必要がないことがあげられる．作成されたシステムは，従来のダム端末でも使用することができ，また逆に従来 LINC で作成されたシステムを EWS/PC を用いて使用することも可能になる．

LINC を用いたときの制約事項として，LINC で開発されたシステムは対話型のために 1 入力に対して 1 回のメッセージの送信しかできず，送信できるメッセージサイズが LINC の制限（1920 バイト）となる．

送信メッセージの電文長が LINC により制約される問題があるものの端末管理や排他制御等の処理を LINC が行うので生産性の高いシステム開発を行うことができるのが特徴である．

6. おわりに

システム開発における状況は，システムが大規模化・複雑化してきたことにより，さらに生産性・信頼性の向上が求められるようになってきている．保守・運用面での負荷が大きくなる一方で，アプリケーション開発においての要求も大規模化・複雑化してきている．

最近は，PCのハードウェア/ソフトウェアが脚光を浴びており，汎用機がそれらに追いつけない面もある．しかし，システム開発においては汎用機の存在を無視することはできない．新しい技術と既存のシステムをうまく統合し，システム開発を推し進めて行く上で，メインフレーム，UNIXサーバ，EWS，PCを統合的に支援するUNIshuttleの存在は大きなものと言える．

とくに重点を置いた「AシリーズとEWSおよびPC間の相互接続とデータの一元管理」に対しては様々な案が挙げられたが，UNIshuttleを採用することで実現することができる．また，UNIshuttleがシステム稼働状況管理やリモートジョブ処理，トランザクション転送機能，プログラム間通信機能などの多くの機能を持っているためシステムに対して発展性を持たせることができる．

メインフレーム，EWS，PCの共通のミドルウェアとしてのUNIshuttleは，レスポンスに関してもユーザの要求（平均レスポンスタイム10秒）を満たすレベルにあり，十分システムを構築するのに耐えうるものである．

また，Aシリーズを中心にシステム開発を行ってきたユーザにとってUNIshuttleを用いてシステム開発を行うことは，今までの開発とかけ離れたものではないので開発しやすいというメリットがある．そのうえ，LINCを活用することができるので，生産性の向上はもとより，過去の資産を活用することができるという大きなメリットを得ることができる．

EWS/PC側のアプリケーション開発ツールとしてTIPPLERが採用され，開発の主な対象となる画面処理に適していたこと，オブジェクト指向開発ツールであるため機能追加などが容易にできること，プロトタイピング型開発手法に適していたことなどにより，開発コストを下げることが可能となり，良い結果が得られた．ユーザの要求の中には，TIPPLERよりも表計算ソフトを利用した方が良いと思われるような非定型の処理および表のみを表示するシステムもあった．しかし，A社の大部分を占める定型業務にはTIPPLERが適していたことやEWSとPCでの共通のユーザインタフェースを保証するためのツールとしてTIPPLERを選択することとした．

今後のシステム開発についての我々の課題としては以下のことが挙げられる．

分散システムに対するユーザの要求が多様化していく中で，我々はその要求に最適のソフトウェアとハードウェアを判断していかなければならない．マルチベンダ環境が一般化している中で，他ベンダとの共同開発，他ベンダ製品とのネットワーク，システム構成などに対応できるノウハウと各種ミドルウェア，開発ツール等の組み合わせ技術がさらに要求されるであろう．

最後に本稿の作成にあたり，ご協力してくださったA社の方々にお礼を述べると共に，執筆にあたりご援助，ご指導を賜った皆様に感謝の意を表する．

執筆者紹介 門 本 光 央 (Mitsuo Kadomoto)

昭和40年生. 平成元年近畿大学理工学部経営工学科卒業. 同年日本ユニシス(株)入社. 4GLセンタにてLINCプロダクトのサポートに従事. 3年より流通関連のユーザサービスに従事. 現在関西支社I&Cシステム1部第2課に所属.

石 倉 哲 善 (Tetsuyoshi Ishikura)

昭和38年生. 62年甲南大学理学部化学科卒業. 同年日本ユニシス(株)入社. アパレル関連のユーザサービスに従事. 現在関西支社I&Cシステム1部第2課に所属.

林 健 一 (Ken-ichi Hayashi)

昭和30年生. 55年京都工芸繊維大学繊維工学科卒業. 同年日本ユニシス(株)入社. 流通関連のユーザサービスに従事. 現在関西支社I&Cシステム1部第2課に所属.

Windows上でのアプリケーション開発の実際

Actual Applications Development on Windows

翠　秀幸

要　約　ここ数年，ダウンサイジング，エンドユーザ・コンピューティングといったキーワードとともに，UNIX, PCによるクライアント/サーバ型のシステムが数多く構築されるようになった．とくに，Windowsの登場以来，PCの能力，機能の発展には，目覚しいものがあり，クライアント端末の選択肢の最右翼ともいえる存在になっている．ビジュアル開発環境を実現するためのツールも，続々とWindowsアプリケーション市場に投入されている．その中でも“Microsoft Visual Basic Ver 2.0”(以下VB)が最もメジャーな地位を獲得している．また，これを継承する形で，統合型表計算ソフトウェアである“Microsoft Excel Ver 5.0”の投入を契機にVBと同様の言語仕様を持つ，“Visual Basic Application Edition”(以下VBA)が搭載され登場してきている．この二つの言語手段を用いて，実際にプログラム開発が行われる機会も非常に多くなっている．また，その際，各ベンダから提供されるWindows上の複数アプリケーションにまたがって，制御やデータを相互にやり取りする方法として，DDEプロセス間通信を取り上げる．そのDDEについて，詳細に解説し，VBとExcelとの間でデータのやり取りをDDEを使って動作する実際のアプリケーション例を紹介する．

Abstract　With the wide spread of buzzwords such as downsizing and end user computing, the past several years have seen new UNIX- and PC-based client/server systems in great numbers. The availability of Windows, in particular, has contributed to remarkable progress in PC performance and functionality; thus having turned PCs into the best choice of all client terminals. The Windows applications market has also witnessed the successive announcement of software tools which provide visual system development environments. Of all those tools, “Microsoft Visual Basic Version 2.0” (referred to as VB hereafter) has established the most influential position. The unveiling of integrated spreadsheet “Microsoft Excel Version 5.0” as successor triggered the entry into the marketplace of “Visual Basic Application Edition” (called VBA hereafter) which has language specifications similar to VB's. There have been more and more cases where programs are developed through the use of those two languages. The author zeroes in on dynamic data exchange (DDE) process-to-process communication that serves to mutually exchange controls and data across multiple applications on the Windows programs supplied by different vendors. Besides referring to DDE in detail, this paper portrays some actual applications for which data are transmitted via DDE between VB and Excel.

1. はじめに

Windows Ver 3.1日本語版(以下Windows)が'93年春に発売され，約2年がたった．Windowsの登場によって，アプリケーションの形態が変わってきている．エンドユーザのアプリケーションへのニーズをシステム開発担当者が分析し，システム化要件を決定し作り上げるといった開発形態では，時間的にも，ニーズを正しく反映する

本稿に記載の会社名，商品名は一般に各社の商標または登録商標である．

ことも難しくなってきたためである．

その結果，日々変化し発生する複雑なシステム化要件の積み残しに対応すべく，エンドユーザ自身が開発し，自由に変更が可能なシステム作り，つまりエンドユーザ・コンピューティング(EUC)の考え方がさらに発展してきた．また，Windows上のビジュアル開発ツールの登場により，視覚的にもわかりやすく，複雑な処理が内部に隠蔽された高度なインタフェースや十分な機能と使いやすさを持ったアプリケーションが従来に比べて容易に作成できるようになったことも，EUCをより現実的なものとしている．

その数あるビジュアル開発ツールの中でも，VBは標準とも言えるポジションを獲得している．また，VBと全く同じ言語仕様を持ったExcelのマクロ言語であるVBAも投入されている．これらを使ってWindowsアプリケーションを作る機会は，確実に増えてきていると言える．

さらに高度な技術として，Windows上の複数のアプリケーションにまたがる制御やデータを相互にやりとりする手段として，OLE, DDE, DLLやクリップボードなどの方法があるが，その中で最も汎用性が高く，手軽なものとして，DDEがあげられる．最近では，OLEの方が話題にのぼっているが，実装レベルを考慮すると(現在，日本語版Excel 5.0は，OLE 2.0を提供しているが，日本語版VB 2.0は，OLE 2.0は未提供である．一方，最新のVB英語版は，バージョン3.0が出荷されており，こちらは，OLE 2.0が提供されている．このVB 3.0は，日本語化の予定がなく，日本語版VBでOLE 2.0の提供は次バージョンまで待つ必要がある)，現時点での最適解はDDEであると考えられる．

2. Windowsアプリケーション

Windowsは，ユーザにとってはやさしいが，プログラマにとっては難しいといわれる．つまり，Windowsを使用する場合，GUIベースのオペレーションにより容易に操作できる．しかし，Windowsアプリケーションを開発するのは，多くの従来のCOBOL, C等の手続型のプログラミングとは，全く異なった考え方であるイベント駆動型プログラミングを理解してプログラムを作る必要がある，それに加えて，windowsが提供している1000個前後の関数を使いこなさなければならないのも大きな障壁となっている．

2.1 イベント駆動型プログラミング

イベント駆動型プログラミングとは，制御の主導権がWindows本体にあり，待ち状態にあるWindowsプログラムがWindowsから送られてくる各メッセージ（たとえばマウスがクリックされた等）に対して一定の処理をランダムに実行するようにプログラムを作ることを指している．常に一定の順序でプログラムが実行されるとは限らない．これに対して，従来のCやCOBOLといった言語は，手続型プログラミングで記述される．手続型プログラミングの場合，プログラムソースの先頭から順（分岐処理，GO_TO処理，ループ処理を含む)に処理命令を実行していくものを指す(図1)．

Windows上で動くアプリケーションプログラムは，すべてこのイベント駆動型で作る必要がある．また，従来の手続き型プログラムと根本的に考え方が異なることか

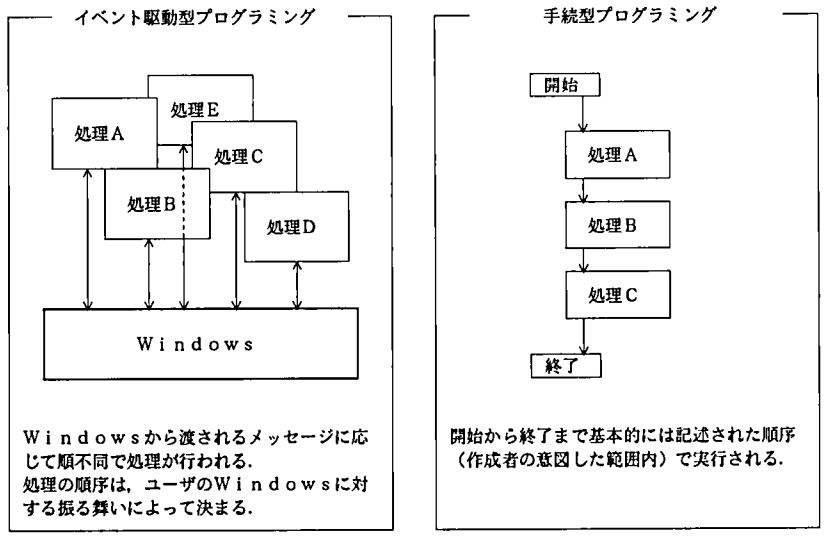


図 1　イベント駆動型プログラミングと手続型プログラミングの違い

ら，設計書等のドキュメントが従来のものに適用しにくい，テストケースの複雑化といったインパクトも発生する．

2.2 実際の Windows プログラム

実際に C/C++言語を使用して Windows アプリケーションを作る場合，主関数とウィンドウプロシージャの二つの関数を定義する．

主関数はウィンドウを作り，Windows からのメッセージを受け取る構造を持ち，Windows プログラムを作る際に必須となるオーバヘッドともいえる部分である．また，ウィンドウプロシージャは，主関数から呼び出され，各メッセージごと（イベント種類ごと）の処理を記述すべき部分である．

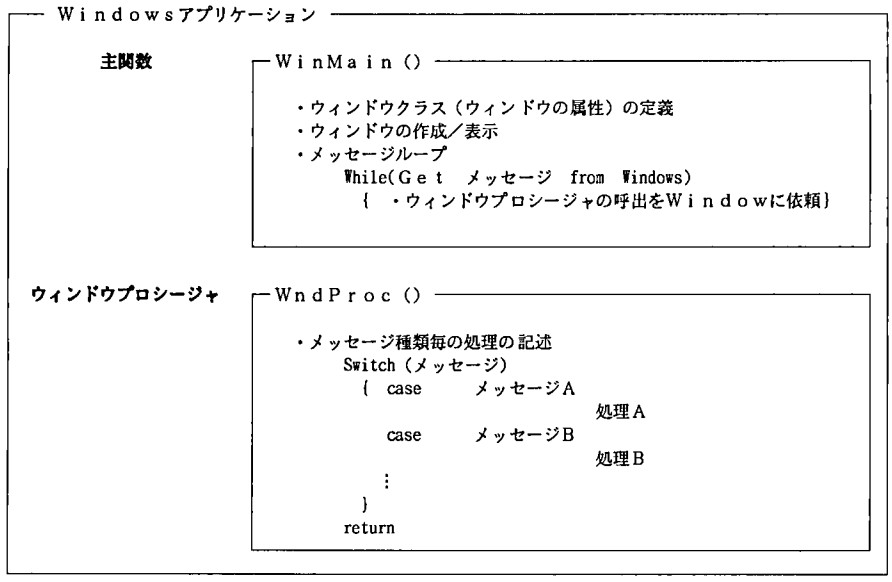
プログラム構造は図 2 に示すとおりである．

何かの処理を行うために，OS を呼び出すという考え方は，今までの手続型プログラミングにもあった．実際，Windows ファンクションコールと呼ばれる API が 1000 種類以上存在する．しかし，Windows アプリケーションでは，それと同時に Windows の方からもプログラムを呼び出す．つまり，何かのイベント（ウィンドウのサイズが変更された，ウィンドウがアクティブになった，など）の都度，Windows は，WinMain を通して，WndProc を呼び出すのである．

通常，Windows システムを Windows のインストールディスクからインストールしただけでは，ユーザは，Windows システムを使うことは可能だが，Windows プログラミングを行うことはできない．

Windows プログラミングを行うためには，Windows 開発キット（SDK：Software Development Kit）と MS-DOS もしくは，Windows 対応の C コンパイラが必要となる．

以降の章では，Windows アプリケーションを実際に作る際の二つのアプローチ方法を紹介する．ビジュアルプログラミング言語である VB と VBA を使って，Win-



メッセージループのGetメッセージでWindowsからメッセージを受け取る（SENDされたメッセージ）か，アプリケーションのメッセージキューからメッセージを取り出し（POSTされたメッセージ），ウィンドウプロシージャを呼び出す．受け取るメッセージが存在しない場合，別のアプリケーションに制御を移す．（ここで，Windowsは，疑似マルチタスクを実現している．）

図 2 Windows アプリケーションの基本構造

dowsアプリケーションを開発する手順とその実際を示す．

3. Visual Basic for Windows Ver 2.0（以下 VB）

VBは，リレーショナルデータベース（以下RDMS）へのアクセス機能を持たないビジュアルプログラミング言語に分類される．(RDMSにアクセスするには，別途ミドルウェアと呼ばれるRDMSインタフェースが必要となる．）一般に，VBを使用すれば容易にWindowsアプリケーションを作成できると言われる．VBを使うことによって，ビジュアルの部分つまりユーザインタフェース作成の部分は，マウス主体で極めて容易に作ることが可能となったが，それ以外のプログラミングのコードに関しては，その名のとおり，構造化BASICで記述する必要があり，プログラム作成全体が自動化され容易になったわけではない．

3.1 VB アプリケーションの作成

ここで，実際のプログラム作成例にそって，VBを使用した開発の流れを以下に示す．

1) 画面（グラフィカルインタフェース）を作成する．(図3(a)(b))
コントロールと呼ばれるインタフェースを構成するオブジェクト（ボタンやテキスト・ボックス等）をフォームと呼ばれるウィンドウ上に配置する．

2) プロパティを設定する．(図3(c)(d))
作成した各々のコントロールのプロパティと呼ばれる属性（色，サイズ，キャプション，フォント等）をプロパティウィンドウに設定する．

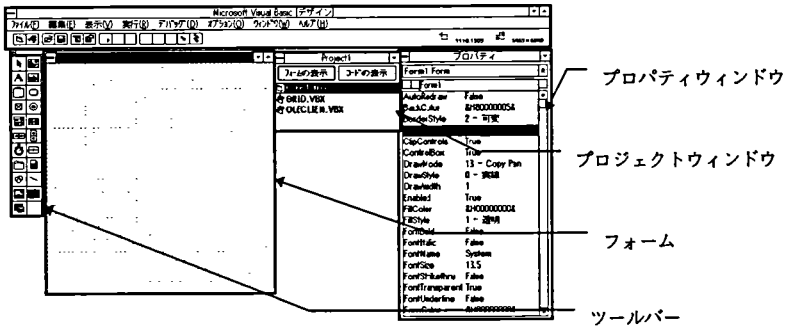


図 3(a)　Visual Basic Ver2.0 を立ちあげたところ

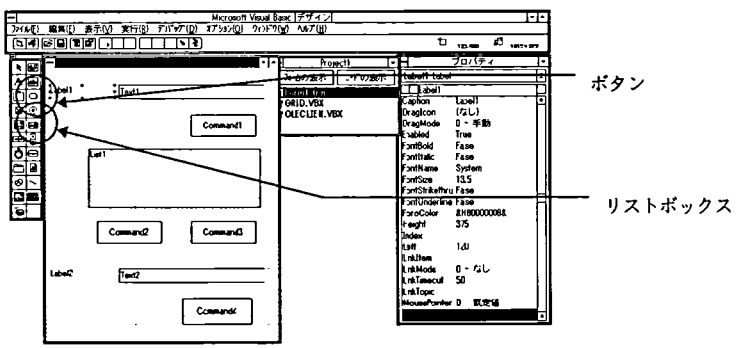


図 3(b)　ツールボックスより，各コントロールを選択し，マウスでフォーム上に張り付けた．選択したコントロールのプロパティ（属性）の設定（デザイン時）は，プロパティウィンドウで行う．

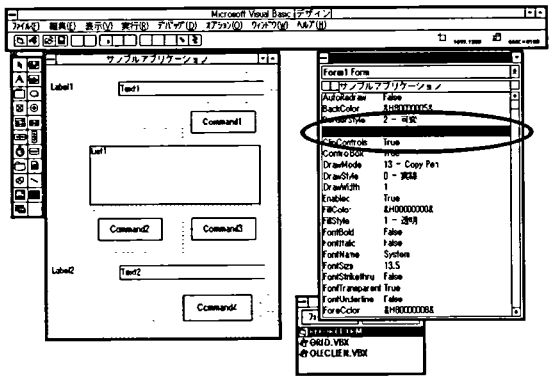


図 3(c)　例として，フォームの Caption プロパティを“Forml”から“サンプルアプリケーション”に変更した．

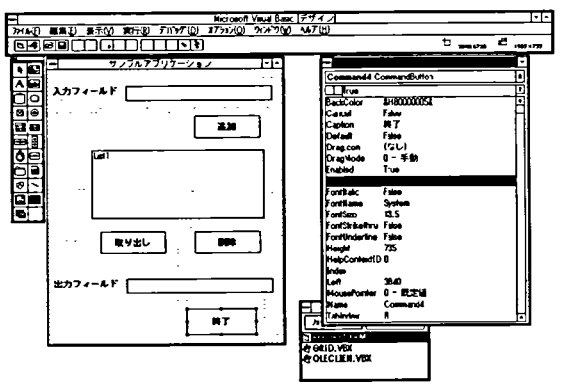


図 3(d)　全てのコントロールのプロパティを設定し終えたところ．(デザイン時には，設定できないプロパティや参照のみ可能なプロパティもある．例：リストボックス内の項目数を表す List Count プロパティ等)

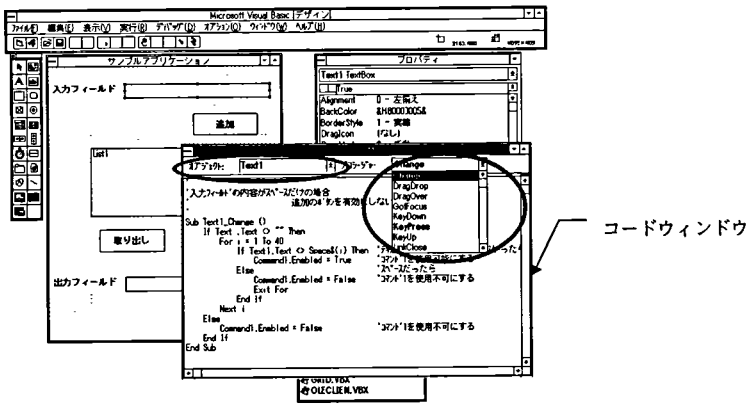


図 3(e) 各コントロールに発生するイベントに対する処理を記述する．例では，選択されているテキストボックス(Text 1)に変更が発生した時の処理を記述している．

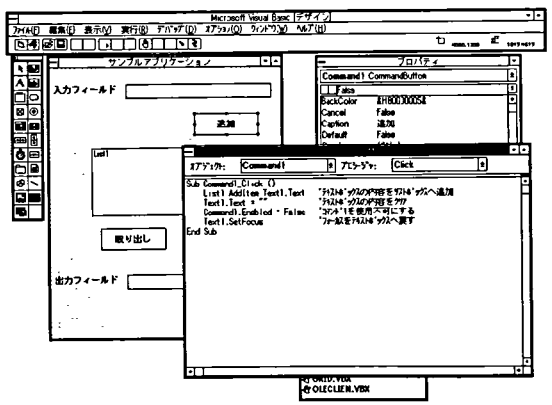


図 3(f) 追加ボタン（Command 1)がクリックされた時の処理の記述

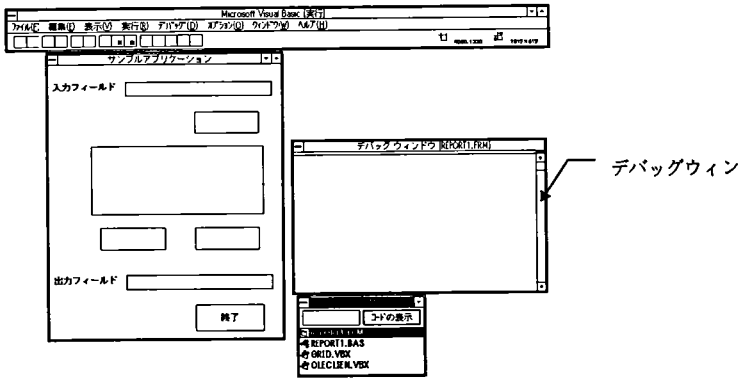


図 3(g) 作成したアプリケーションを開発環境上で実行した．アプリケーションとしては，入力フィールドへの入力待ちの状態である．

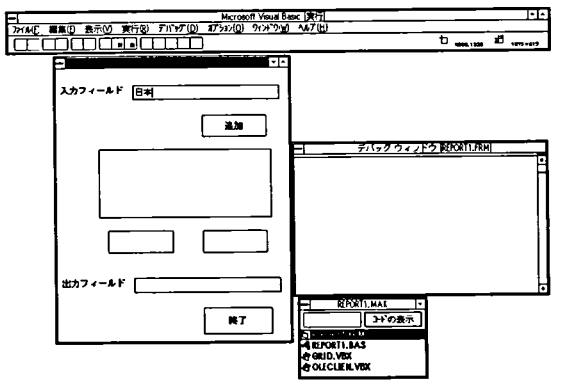


図 3(h) 入力フィールドに“日本”と入力した．追加ボタンが有効になっている．

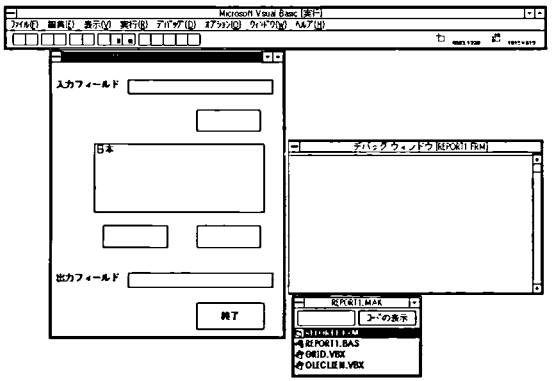


図 3(i)　追加ボタンをクリックし，入力値がリストボックスに転記され，再度追加ボタンが無効となった．

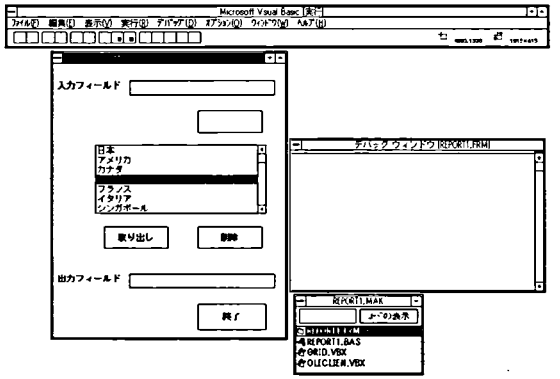


図 3(k)　リストボックス内の項目を選択する．“取り出し”と“削除”ボタンが有効化される．

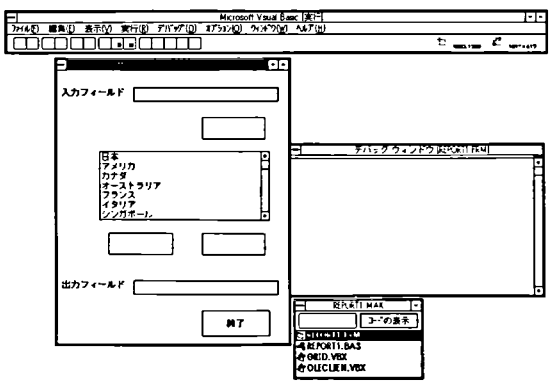


図 3(j)　複数件データを入力した．リストボックスに項目が入りきらないため，自動的にスクロールバーが付加されている．

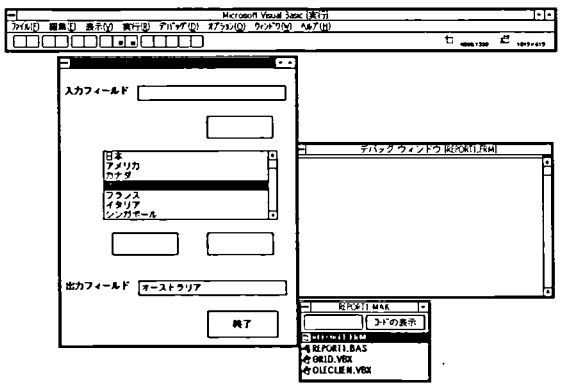


図 3(l)　“取り出し”ボタンをクリックした．選択した項目が出力フィールドに転記されている．

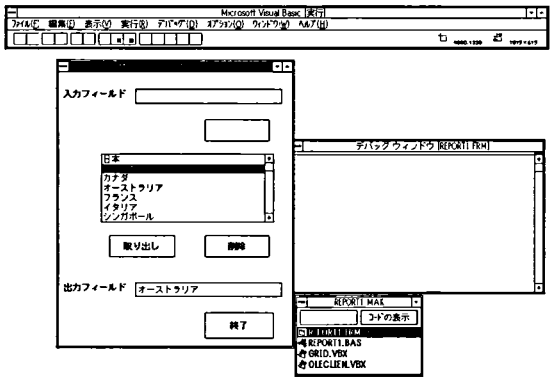


図 3(m)　アメリカを選択した．再度，“取り出し” と “削除” ボタンが有効化される．

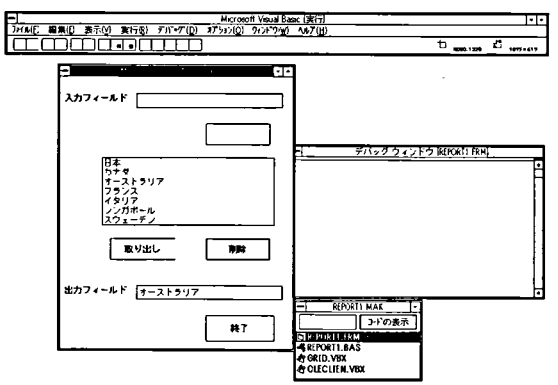


図 3(n)　“削除” ボタンをクリック．選択項目が削除される．

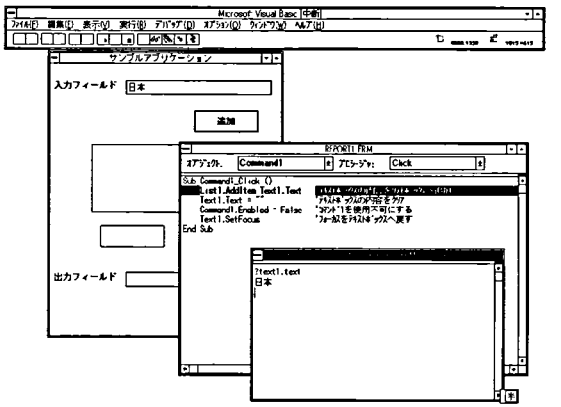


図 3(o)　ブレークポイントを設定し，処理を止めたところ，デバッグウィンドウ上での現在の Text 1, Text の値を表示させた．

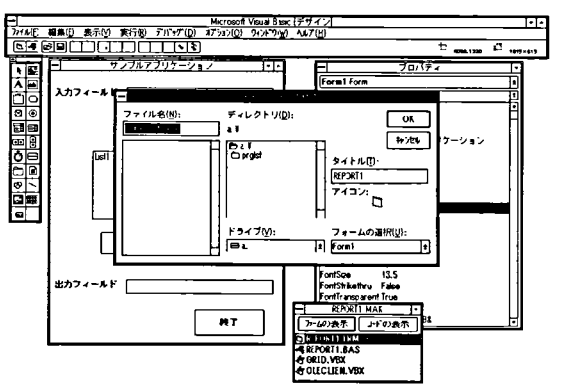


図 3(p)　完成したアプリケーションを実行モジュールにコンパイルする．

3)　コードを記述する．(図3(e)(f))

各コントロールに発生するイベントごとに処理をコードウィンドウに記述する．シンタックスの検査は，コード記述と同時に行われる．

4)　テスト/デバッグ

開発環境上でのテスト実行を行う．(図3(g)～(n))

ブレークポイントの設定，ステップ実行，データ監視等のデバッグツールを使用して，論理エラーやランタイムエラーの原因追求を行う．(図3(o))

5)　実行モジュールを作成する．(図3(p))

コンパイルを行い，実行モジュール（.EXEタイプ）を作成する．

実行には，VBに付属する再配布可能なランタイムライブラリ(無償)が必要である．

各コントロールには，あらかじめ取得可能なイベントプロシージャの枠(図3(e)コードウィンドウ右上のリストボックス参照）が準備されており，そこに処理を記述していく．このサンプルプログラムでは，8個のイベントプロシージャがある（図3(q)).

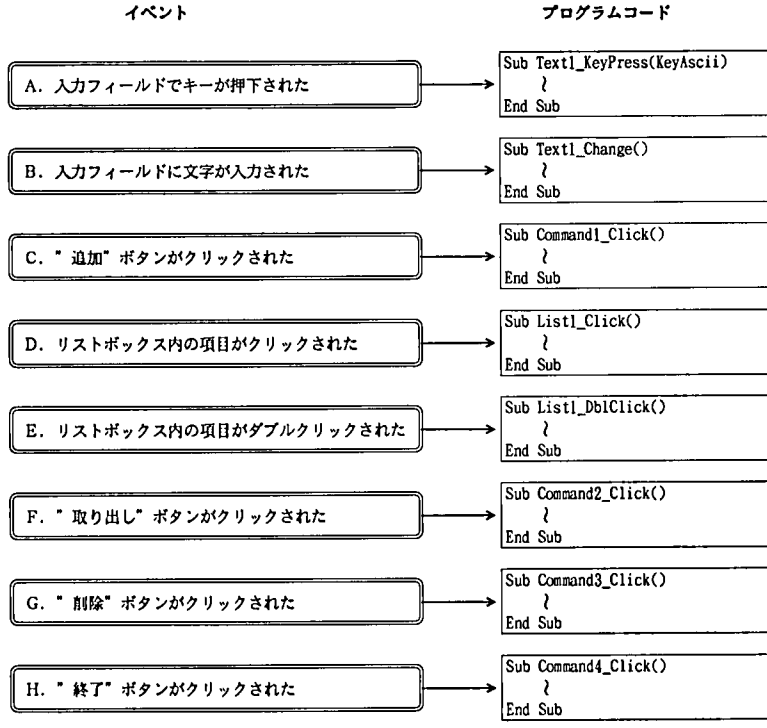


図 3(q)　作成したVBサンプルアプリケーションのイベントとコードの対応

Text 1で文字が一文字入力されるごとにAのKeyPressとBのChangeイベントが順に発生する．また，DのClickとEのDblClickでは，Eのイベントが発生する前に，2回のDのイベントが発生する．このため，AとB, DとEのイベントコードでは，相互で問題が発生しないよう留意してコードを記述する注意が必要である．

3.2 VBのメリット/デメリット

ここでは，VB 2.0 を実際に開発に使用する上でのメリット/デメリットについて述べる．

1) メリット

① 入力画面の作成/修正が容易であり，プロトタイプ開発手法に向いている．

② 開発環境として完成度が高い．

VB を構成するセグメント(メインウィンドウ，プロジェクトウィンドウ，ツールボックス，プロパティウィンドウ，デバッグウィンドウ，フォーム）が，機能的に統合されていて，使いやすい．

③ イベント駆動型のプログラミング言語について理解できれば，短期間で修得が可能である．

コードの BASIC 部分も構造化 BASIC を取り入れているため，COBOL 等からの移植も大きなインパクトはない．また，入力と同時に Syntax の検査が行われる．たとえば，1 ステートメントを入力後，カーソルを移動させると，エラーに対してメッセージが表示され，予約語に関しては，色の変化によりコードを見易くするといった機能がある．

④ DDE, OLE 1.0 (Object Linking & Enbedding)，DLL (Dynamic Link Library) が提供されており，市販の VBX が使用できる．

VBX (Visual Basic Custom Control) は，ファイルの拡張子が VBX のためこう呼ばれる．VB 本体にインクルードすることで VB のコントロールを追加でき，ソフトウェアの部品化を図っている．市販のもの（ミドルウェア的な RDMS 接続コントロールもある)も多く出回っており，また C/C++言語等で自作することも可能である．(ちなみに，VB で使用できる OLE 機能，Grid コントロールも VBX により実現されている.) しかし，VBX は VB と Win 16 メッセージアーキテクチャに依存しているため，他 OS (ex. WindowsNT, Windows 95 等の Win 32 環境)や他の開発ツールとの互換性が得にくいといった問題点がある．そうした欠点を解消するため，OLE Control(旧 OLE Custom Control (OCX)) と呼ばれる，OLE 2.0 をベースにした Custom Control が現在提唱・普及がなされている．ただし，VB での OLE 2.0 の提供は VB Ver 3.0 (英語版のみ) からである．

2) デメリット

① プログラム上の処理の順序が特定できないため，いろいろなイベント発生順序について十分考慮してプログラミングする必要がある．

② Visual Basic で DLL モジュールを作成できない(呼び出すことは可能)．

DLL モジュールは，C/C++で作成する必要がある．

③ VB コードに関してライブラリ化の概念がない．

同じコードをアプリケーション間で共有するためには，各アプリケーションがソーステキストを実行モジュールごとに読み込む必要がある．

④ ソース・モジュールの管理機能（バージョン，複数人員開発）を持っていないため大規模開発には向いていない．

VB は，小人数での開発向きである．

⑤ 印書機能が弱い．

行単位でバッファに積み込み，ページごとに打ち出す機能があるだけである．

4. Visual Basic for Application Edition（VBA）

VBA は，Excel Ver 5 から新たに搭載され，VB とほとんど同様の言語使用を持ったマクロ言語である．

VBA は，純粋なプログラム言語ではなく，Excel の操作を包括し，Excel 上での定型処理を記述するためのマクロ言語であり，Excel 自身の豊富な機能のほとんど全てをコードで記述できるようになっている．

4.1 VBA アプリケーションの作成

VBA による開発では，VB ほど固定化された手順は存在しない．たとえば，VBA のコードを記述する場合でも，Excel での会話型のオペレーションをマクロ記録させ，必要箇所を修正していく方法や，いきなりコードを記述するといった，いくつかの手順が考えられる．基本は，1）ボタン等のオブジェクトの作成/配置，2）処理の記述，と，3）起動するタイミングの決定，4）テスト/デバッグの四つの手順を必要に応じて選択して行うことになる．

VBA を使用した開発の流れを以下に示す．

1） ボタン等オブジェクトの作成/配置（図 4（a）～（d））
- ダイアログシート上でのダイアログボックスの作成
- ワークシート上へのボタン等の作成
- ツールバーへのボタンの作成
- メニューバーへのメニュー項目の追加

2） 処理の記述（図 4（f））
- 一連の操作をマクロ記録機能によりモジュールシート上にコード化/修正
- モジュールシート上での新規/既存マクロの記述/修正

3） 起動するタイミングの決定（図 4（g））
- ボタン等とコードの連結
- イベントプロシージャの登録
  （ダイアログボックスの表示時，テキストの変更時，シート/ウィンドウのアクティブ化/非アクティブ化時等）
- 任意のマクロを実行時に選択

4） テスト/デバッグ（図 4（h）～（k））
- （テスト）実行
- ブレークポイントの設定，ステップ実行，データ監視等のデバッグツールを使用して，論理エラーやランタイムエラーの原因追求を行う．

このサンプルプログラムでは，3 個のイベントプロシージャがある（図 4（l））．

すべてのイベントプロシージャの登録は，メニューのマクロ登録か各オブジェクトの OnAction プロパティで設定する必要がある．

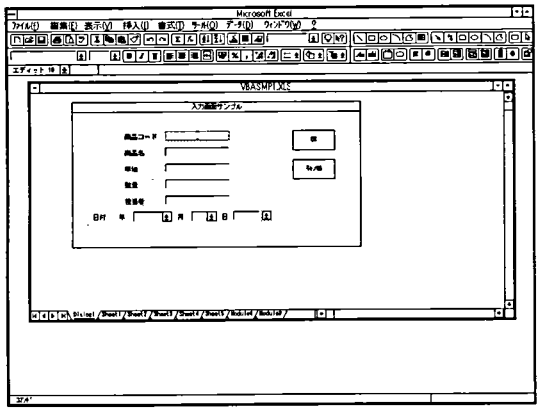


図 4(a)　Excel ダイアログシート上にコントロールを配置し，ダイアログボックスを作成

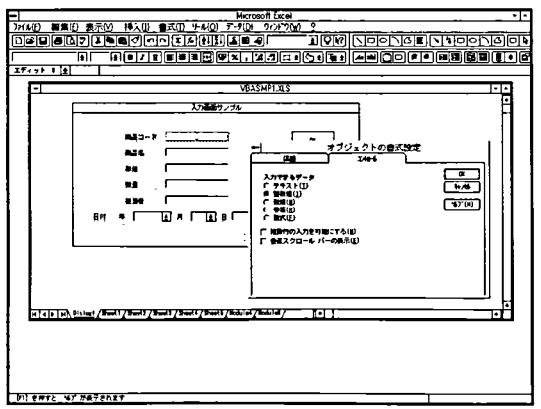


図 4(b)　オブジェクトごとの書式設定．この例では，商品コードを入力するテキストボックスの入力データを整数値に設定

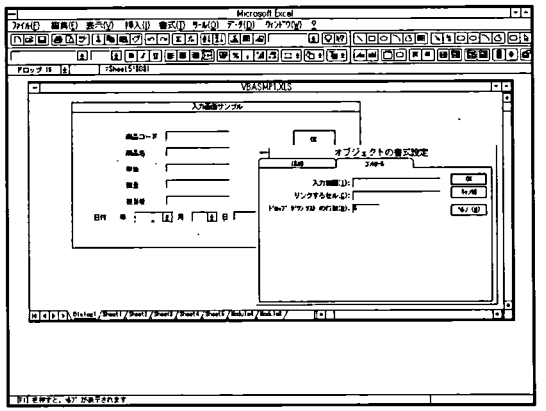


図 4(c)　ドロップダウンリストの書式設定を行う．選択時の表示枠の行数を設定

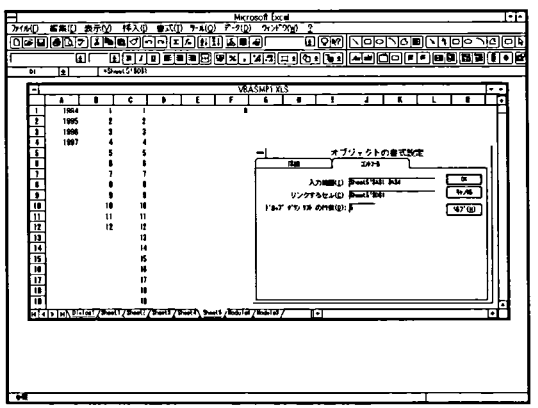


図 4(d)　ドロップダウンリストに表示する値のリストをワークシート内のセル範囲で指定する．また，選択された結果を反映するセル位置の指定を行う．

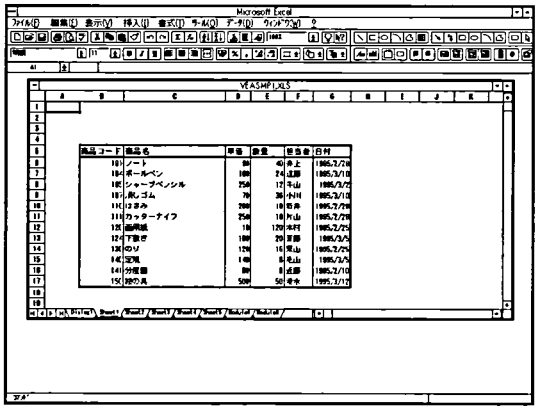


図 4(e)　このサンプルアプリケーション用の表の初期リスト

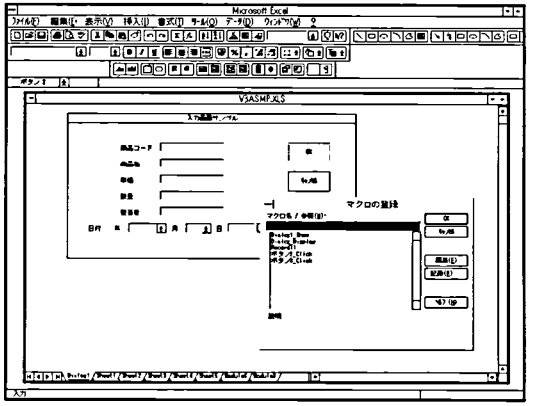


図 4(g)　ダイアログ上のボタンにマクロを登録

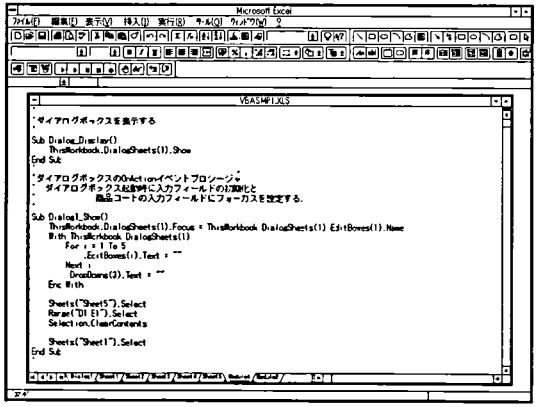


図 4(f)　モジュールシート上に処理を記述

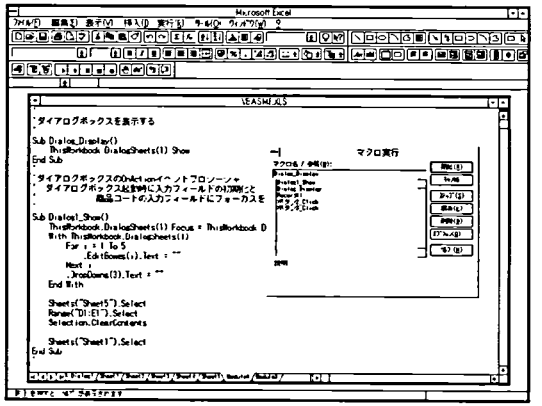


図 4(h)　記述したマクロの実行

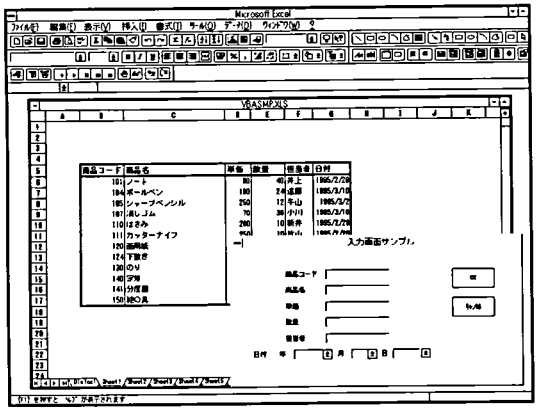


図 4(i)　実行直後の状態．表が前面に表示され，ダイアログが表示される．

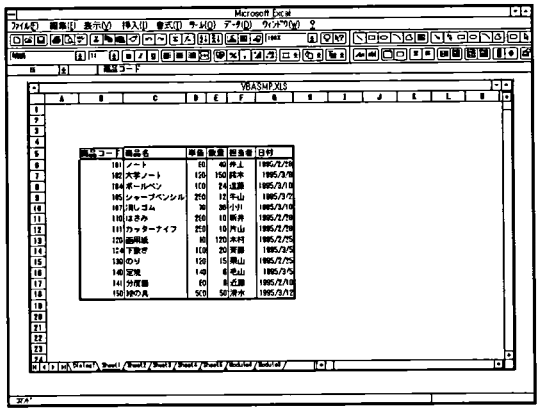


図 4(k)　実行結果として，2レコード目に新しいレコードが挿入(ソート済み）される．

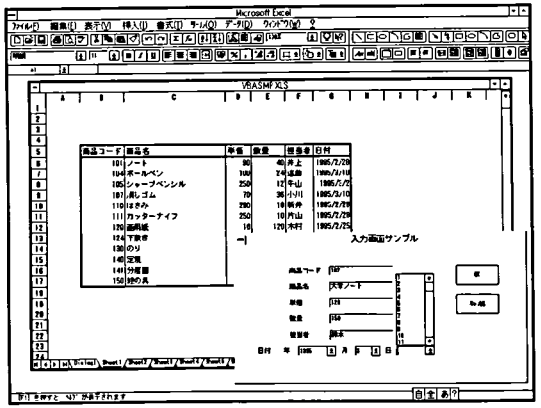


図 4(j)　ダイアログに値を入力し，OKボタンをクリックする．

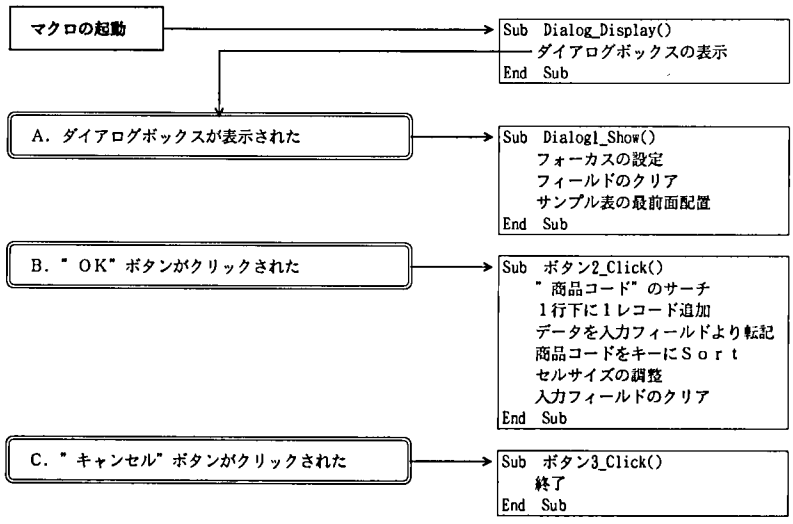


図 4(1)　作成したVBAサンプルアプリケーションのイベントとコードの対応

4.2 VBAのメリット/デメリット

VBとVBAでは，言語仕様の点では非常に似通ってはいるが，VBが純粋にWindowsアプリケーションを作るためのプログラミング言語であるのに対して，VBAは，Excel上で動き，基本的にはExcelを制御するためのマクロ言語である点で，存在目的がまるで違っている．たとえば，VBでは，作成したアプリケーションをコンパイルして，EXEタイプの実行モジュールを作ることができるが，VBAで作成したアプリケーションは，中間コードに落としたアドインが作成されExcelに包含されて，Excel上で稼働する形になる．また，VBとVBA間相互に互換性は全くない．VBには，Excelが持つオブジェクト（ブック，シート等）はなく，VBAではVBの持つフォームやイベントをハンドリングする手段をもたない．

1)　メリット

①　複数のWindowsアプリケーションにまたがる定型の処理を統一的に記述制御できる可能性がある．

（現時点では，VBAを搭載する日本語版アプリケーションは，Excel Ver 5のみであるが，VBAから先に触れたOLE 2.0を提供するアプリケーションを制御可能）

②　エンドユーザが自由に使用することのできるエンドユーザ向け開発環境として提供できる．

③　Excel従来のマクロが，ワークシート上に書き込まれるのに対して，モジュールシートを呼ばれるVBAコード専用のシートが用意され，旧マクロに比べてよりプログラミング言語に近くなったため，コードの可読性が高い（インデント，豊富な制御構造等による）．

④　Excelのマクロ記録の機能を使って，Excelに対する操作を記録でき，プログラミングを支援する．

⑤　DDE，OLE 2.0，DLLが提供されている（ただし，先に触れたVBXをVBA

から使用はできない).

2) デメリット

① 基本的に，VBA は手続き型の言語であり，VBA では，VB ほど明確にイベントの概念が定義されていない．VB のコーディングでは，共通モジュールコード以外は，各イベントに対して処理を記述していくが，VBA のコーディングでは，各オブジェクトやイベントに対してユーザが必要に応じて，任意に登録・記述を行う．

② 各コントロールオブジェクトに対するイベントの種類が VB に比べて少ない．たとえば，ボタンに対するイベントは，VB では，Click, GetFocus, KeyPress 等，8 種類存在するのに対して，VBA では，OnAction (VB で Click に当たる) イベントの 1 種類しか存在しない．(VBA では，ボタンに限らずチェックボックス，オプションボタン，リストボックス，ダイアログボックス等も，OnAction イベント 1 種類しか存在しない．例外は，Excel 本体，ブック，ワークシートのみで，これらは，複数のイベントが存在する.)

③ ビジュアル開発環境としてみた場合，入出力を受け持つ画面を専用のダイアログボックス (VB では，フォームにあたる) の作成機能，操作性は，プロパティウィンドウがないことなどから，VB に比べて劣る．

④ VB の場合，制御するオブジェクト (フォームや各コントロール) は 30 種程度であるが，VBA は，Excel 本体の操作のためのオブジェクト (ブック，ワークシート，ダイアログシート，チャート等) が 130 種以上存在し，さらにそれらオブジェクトに付随するメソッド，プロパティの数を考えると膨大なものになり，習得に対する負荷が大きい．

⑤ アプリケーション稼働環境としての Excel 自身の設定が明文化されない/保証されない．たとえば，アドイン・マクロの登録状況，Excel 立ち上げ時の状態，オブジェクトとコードの関係等は，通常 (Excel メニューで登録した場合 (図 4 (g)) VBA のコード上に表されないし，ユーザによって容易に変更が可能である．これは，別の端末への移植時やメンテナンス時に困るだけでなく，トラブルの原因になりかねない．これらを回避するためには，メニューによる登録ではなく，VBA コードでそれぞれの登録を行わなければならない．

5. ダイナミックデータエクスチェンジ

ダイナミックデータエクスチェンジ (DDE: Dynamic Data Exchange) は，Windows が提供する三つのプロセス間通信機能の一つである．他の二つは，クリップボードとダイナミックリンクライブラリである．

DDE 通信とは，Windows 上のソースとディスティネーションと呼ばれる二つのアプリケーションが相互に，連続的にメッセージ交換する機能である．基本的にデータは，ソースアプリケーションからディスティネーションアプリケーションに渡される．Windows 3.1 では，Windows 3.0 からのバージョンアップに伴い，DDE 管理ライブラリ (DDEML) が提供されている．これは，以下に説明する Windows メッセージを使った DDE よりさらに高水準かつ単純化されたファンクションコールレイアを提供

し，より簡単にプログラミングができるようになっている．ここでは，メッセージ交換の様子をよりリアルに認識するため，一番プリミティブなWindowsメッセージを用いて，DDE通信の処理の流れを説明する．DDEMLも，この従来のWindowsメッセージによるDDE通信の上位に位置するものであり，互換性を持っている．

DDE通信は，ディスティネーション側から起動され，Windows上で実行中の全てのプログラムに通信開始要求を相手の識別子（後述）を示してブロードキャストする．該当するソースアプリケーションは，ディスティネーションからの通信開始要求に対して応答メッセージを返し，DDE通信が開始される．

一つのWindowsプログラムは，同時に複数のDDE通信を行い，同時にディスティネーションとソースを兼ねることも可能である．その場合，それぞれのDDE通信に対して，別々のDDE通信のリンクを開設しなければならない．

5.1 アプリケーション，トピック，アイテム

ディスティネーションアプリケーションでは，相手のソースアプリケーション名，トピックと呼ばれるDDE通信の主体名を指定して，DDE通信の開始要求を発呼する．また，実際にやり取りするデータとして，アイテムを指定する必要がある．また，一つのDDE通信内（アプリケーション名とトピックの組み合わせ）で一つ以上のアイテムを動的に変更しながらデータのやり取りを行うことが可能である．

表1に，VBアプリケーション，Excelをソースにした場合のディスティネーションから見たアプリケーション，トピック，アイテムの例を示す．

表1　アプリケーション，トピック，アイテムの例

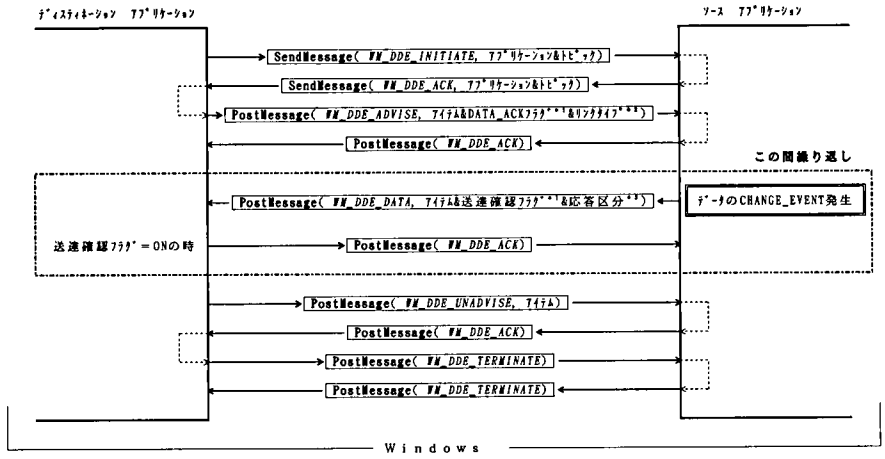
	アプリケーション	トピック	アイテム
VBアプリケーション	実行可能ファイル名	フォーム名	ラベル ピクチャーボックス テキストボックス
Excel	Excel	［ブック名］シート名	セル

5.2 DDE通信の種類

DDE通信には，データの交換方法の違いによって，自動リンク，手動リンク，通知リンクの3種類がある．

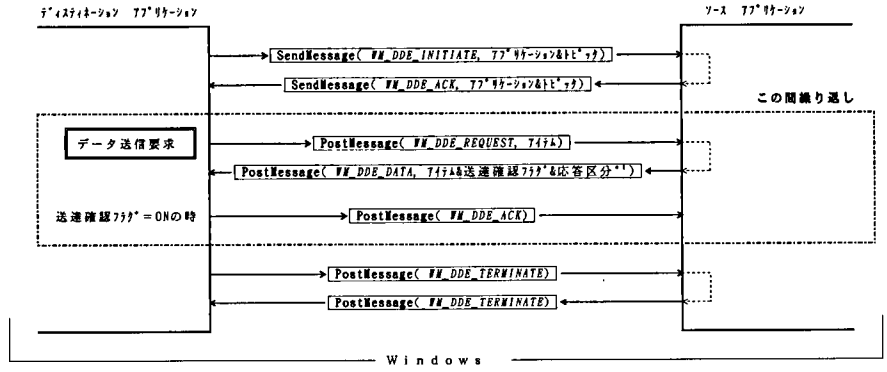
1) 自動リンク：ソースアプリケーションのアイテムが変更されるごとに，ソースからディスティネーションにデータを送る．
2) 手動リンク：ディスティネーション側からソースアプリケーションに要求があったとき，データを送る．
3) 通知リンク：ソースアプリケーションのアイテムが変更されるごとに，ソースからディスティネーションに変更があったことのみ通知する．ディスティネーション側では，通知を受信後，ソースアプリケーションに要求を出し，ソース側からデータが送られる．

図5（a）～（c）に，Windowsメッセージを使用してDDE通信を行う場合の概念的な処理の流れを示す(実際のWindowsプログラミングは，従来の手続型プログラミングとは異なり，イベント駆動型プログラミングで組まれる．図では，処理の流れを分かりやすくするため，手続型でプログラムが組まれているように書いた．また，関数



＊1：送達確認フラグの設定は，ディスティネーション側からのＡＤＶＩＳＥメッセージ送信時のＤＡＴＡ＿ＡＣＫフラグに合わせて，ソース側でＤＡＴＡメッセージの送信時に設定を行なう．
＊2：リンクタイプには，＂自動リンク＂のフラグ設定を行なう．
＊3：応答区分には，＂ＡＤＶＩＣＥメッセージの応答＂のフラグ設定を行なう．

図 5(a)　自動リンク時のメッセージの流れ



＊1：応答区分には，＂ＲＥＱＵＥＳＴメッセージの応答＂のフラグ設定を行なう．

図 5(b)　手動リンク時のメッセージの流れ

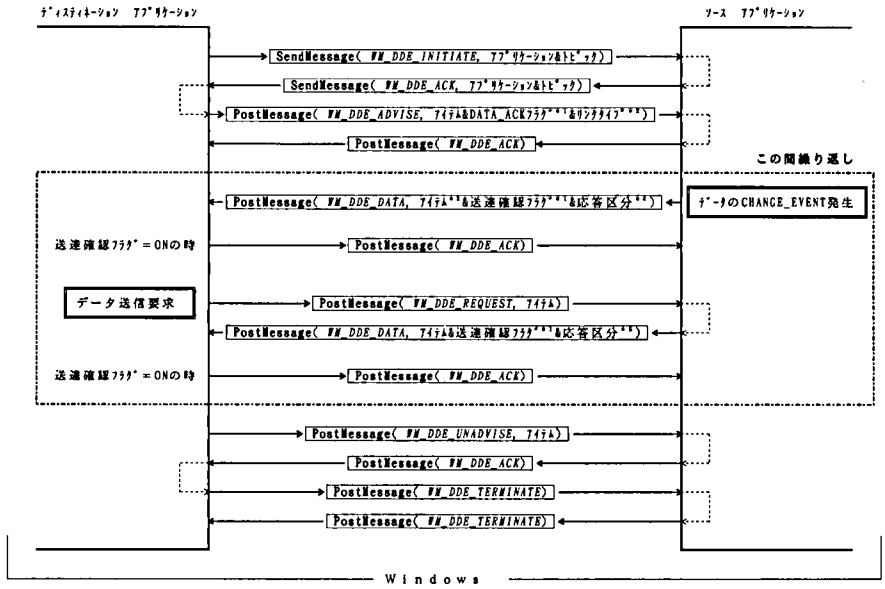
の引き数についても，必要と思われるもののみを記述しているので注意されたい)．

なお，各メッセージの送受信時のデータ(アプリケーション，トピック，アイテム)のやり取りは，グローバルアトムと呼ばれる共有メモリブロックにStringsが格納され，格納時に入手できるアトム値がSendMessage, PostMessageの引き数として渡すことで実現している．

以上の3種類のDDE通信の他にWindowsでは，次の二つのメッセージも支援している．

・WM__DDE__POKE　　：DDE通信の確立中にサーバ側からディスティネーション側に一方的にデータを送る．

・WM__DDE__EXECUTE　　：ディスティネーション側からサーバアプリケ



＊1：送達確認フラグの設定は，ディスティネーション側からのＡＤＶＩＳＥメッセージ送信時のＤＡＴＡ＿ＡＣＫフラグに合わせて，ソース側でＤＡＴＡメッセージの送信時に設定を行なう．
＊2：リンクタイプには，"通知リンク"のフラグ設定を行なう．
＊3：通知リンクのイベント発生時のＤＡＴＡメッセージのアイテムには，ＮＵＬＬの設定を行なう．
＊4：応答区分には，"ＡＤＶＩＣＥメッセージの応答"のフラグ設定を行なう．
＊5：応答区分には，"ＲＥＱＵＥＳＴメッセージの応答"のフラグ設定を行なう．

図 5(c)　通知リンク時のメッセージの流れ

ーションに対してコマンド文字列を送る.(支援されるコマンドは，各サーバアプリケーションに依存する.)

DDE 通信に関して，最も Windows プリミティブな位置づけの解説は以上である．次の章では，先に解説した VB/VBA を使って，VB アプリケーションと Excel 間で DDE 通信を行うアプリケーションの作成紹介を行う．

6. VB/VBA による DDE プログラミング

ここでは，先の章で説明した VB/VBA および DDE に関して，実際のアプリケーションを紹介する．このサンプルアプリケーションでは，VB で作ったモジュールから Excel に対する DDE のリンクと VBA から VB のモジュールへの DDE リンクの両方を使って，必要な情報，データのやり取りを実現している．

紹介するサンプルプログラムの処理形態は，図 6 (a) に示すような形になっている．また，図中網掛けの部分は図 6 (h)(i) に実際のコードの抜粋を掲載した．

コードを見ても解るように，どちらも DDE 通信の開始～Execute/Poke～終了の処理を行っているが，大幅な相違が見られる．また，2 章，5 章で説明した Windows プログラミング，DDE の複雑さは，VB/VBA によってかなり吸収されていることが解るであろう．この例では，Excel に渡すデータを乱数によって作り出しているが，現実のアプリケーションでは，このデータをファイルから読み込んだり，LAN 上につながるサーバ上の DB から取り出すといった作りが考えられる．

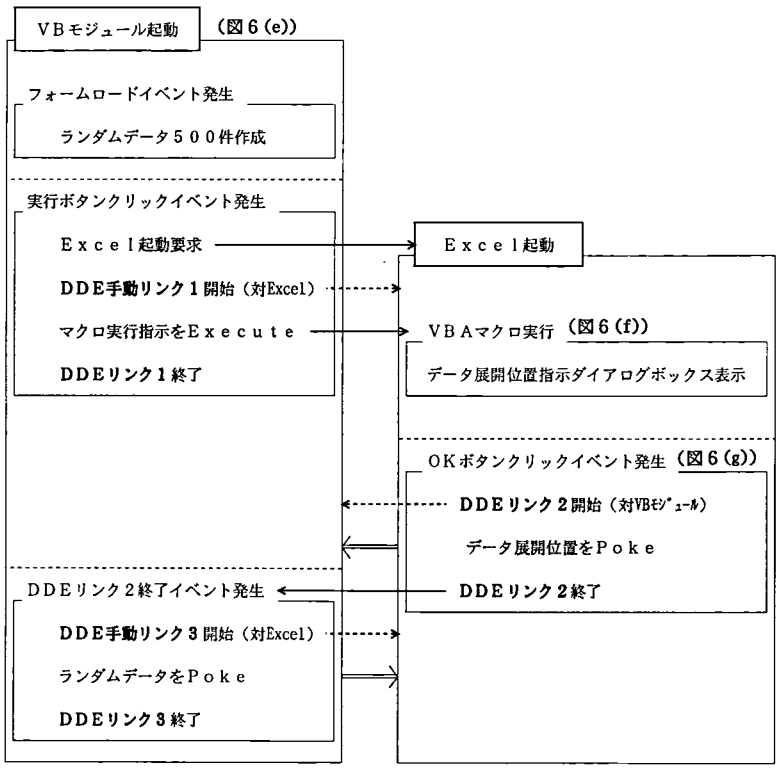


図 6(a)　DDE通信を使ったサンプルプログラムの構造

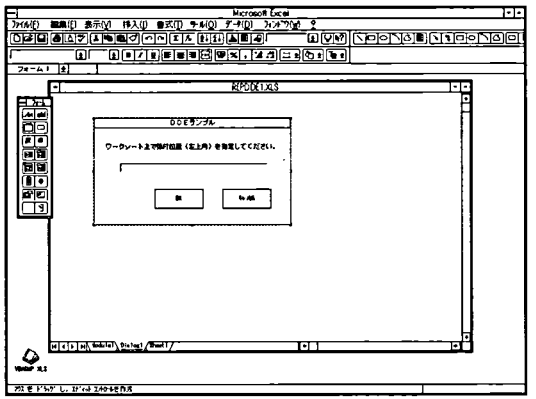


図 6(b)　ダイアログボックスの作成

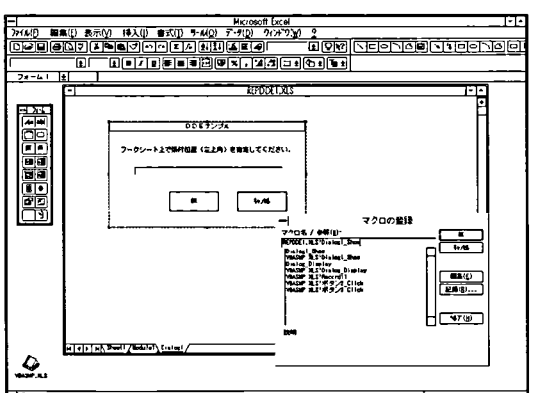


図 6(c)　ダイアログボックスが表示される時に実行するマクロの登録

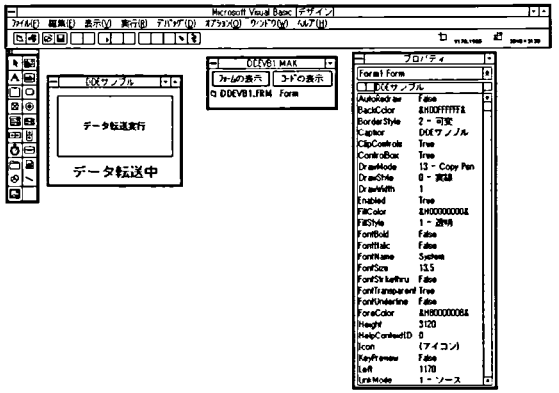


図 6(d)　VB フォームの作成

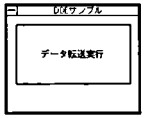


図 6(e)　実行モジュールを実行したところ．
データ転送実行ボタンをクリックする．

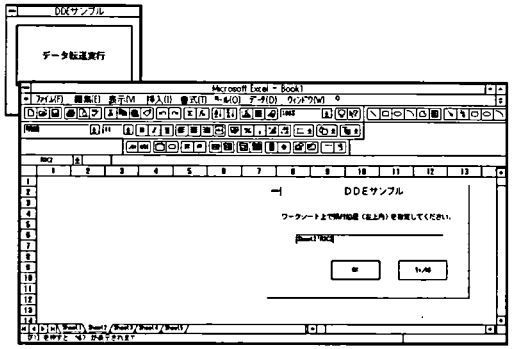


図 6(f)　Excel が起動され，展開位置指定ダイアログボックスが表示された．マウスにより位置を指定して OK ボタンをクリック．

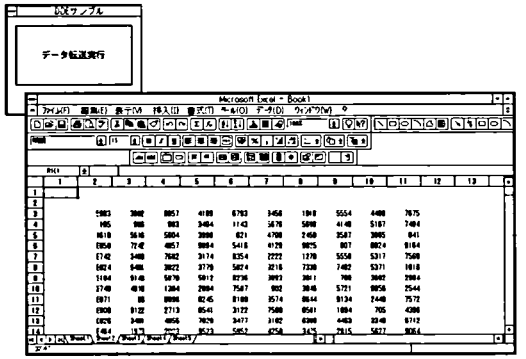


図 6(g)　VB モジュール側からランダムデータが DDE 転送され，指定位置より展開表示される．

```
Q = Chr(34)                                          ' 文字 "  (ﾀﾞﾌﾞﾙｸｫｰﾃｰｼｮﾝ)
Cmd = "[RUN(" & Q & "DDEVBA1.XLA!Dialog_Display" & Q & ")]"
                                                     ' EXCELｱﾄﾞｲﾝﾏｸﾛの起動要求するｺﾏﾝﾄﾞ文字列
Text6.LinkMode = NONE                                ' DDEﾘﾝｸの初期化
Text6.LinkTopic = "Excel|DDEVBA1.XLA"                ' DDEのｱﾌﾟﾘｹｰｼｮﾝとﾄﾋﾟｯｸの設定
Text6.LinkItem = "R1C1"                              ' DDEのｱｲﾃﾑの設定
Text6.LinkMode = 2                                   ' DDE（手動ﾘﾝｸ）のﾘﾝｸ開設
Text6.LinkTimeout = 400                              ' ﾀｲﾑｱｳﾄ時間(40秒）の設定
Text6.LinkExecute Cmd                                ' EXCELに対してｺﾏﾝﾄﾞを実行
Text6.LinkMode = 0                                   ' DDEのﾘﾝｸ切断
```

図 6(h)　VB サンプルコード（抜粋）

```
以前にダイアログボックスにより展開セル位置を入手済み

VB_channel = DDEInitiate("ddevbl", "Form1")          ' DDEの初期化
ActiveSheet.Cells(1, 1) = Selected_Topic             ' ｾﾙA1にﾌﾞｯｸ名とｼｰﾄ名をｾｯﾄ
ActiveSheet.Cells(1, 2) = Selected_Item              ' ｾﾙA2にﾄﾋﾟｯｸ名をｾｯﾄ
DDEPoke VB_channel, "Text1", Cells(1, 1)             ' VBｱﾌﾟﾘｹｰｼｮﾝにｾﾙA1をPokeする.
DDEPoke VB_channel, "Text2", Cells(1, 2)             ' VBｱﾌﾟﾘｹｰｼｮﾝにｾﾙA2をPokeする.
DDETerminate (VB_channel)                            ' DDEの終了
```

図 6(i)　VBA サンプルコード（抜粋）

7. おわりに

本稿では，VB と VBA(Excel)を使った Windows アプリケーション作成について，実際の経験を踏まえて述べてきた．とくに VB と VBA に関しては，その存在目的を十分に考慮して使用範囲を決める必要がある．基本的に VB でできる部分は可能な限り VB を使い，VBA でしかカバーできない部分，つまり Excel に対する制御部分にのみ VBA を使用するようにした方が，作成効率が上がる．

PC を取り巻く環境は，ハードウェアの高性能化・低価格化，ソフトウェアのバージョンアップや新しいソフトウェアが次々に発表され，急速な発達・広がりを見せている．Windows をエンドユーザのインタフェースに採用する開発が増え，今までのメインフレーム時代のインタフェースとは，明らかに大きな違いが発生してきている．従来のキャラクターベースのインタフェースが影をひそめ，マウス主体のインタフェースが当たり前になってきた．また，エンドユーザ・コンピューティングといわれる，いわゆるエンドユーザが，各自のその時点での一番ホットな要求を，ユーザ自らが PC 上で解決するといった動向も見逃すことのできないトピックスである．

現実に，エンドユーザ・コンピューティング，ネットワークや今後の技術動向，コストといった観点から見ると，“オープンシステム”を選択せざるをえない処まで来ている．ただ，WindowsPC を採用したクライアント・サーバシステムが万能と言うわけではない．汎用機ベースのシステムと比べると，まだ未成熟な部分も多く残っている．たとえば，運用まわり，セキュリティ，リアル処理といった技術的な部分とマルチベンダー化，トラブル時の責任の所在等の環境的な部分である．こうして挙げてみても，確かに越えるべき課題は多い．しかし，こうした課題を徐々にクリアし，適用できる部分から確実にオープン化は進んでいる．

本稿では，ビジュアル開発環境，プロトタイピング，開発効率，EUC（エンドユーザ・コンピューティング），EUD（エンドユーザディベロップメント）等のキーワードを踏まえ，実際に開発に用いたVB/VBAとDDEについて主に取り上げた．現時点では，さらに多くの類似/発展した同様の製品が現れてはいるが，同様のアプリケーション開発の参考になれば幸いである．

参考文献

［1］ Charles Petzold, プログラミングWINDOWS Version 3.1（マイクロソフト(株)監訳，(株)エー・ピー・ラボ/長尾高弘翻訳），(株)アスキー，1993年9月21日，pp. 889～948.

［2］ James L. Conger, Programmer's Selection Windows API バイブル1（(株)パセイジ翻訳），(株)翔泳社，1993年9月20日，pp. 874～884, 972～991.

［3］ JAMES L.Conger, Programmer's Selection Windows API バイブル2（(株)パセイジ翻訳），(株)翔泳社，1994年6月25日，pp. 735～814

［4］ Microsoft Visual Basic Programming System for Windows プログラミングガイド，マイクロソフト(株)，1994年1月20日.

［5］ Microsoft Visual Basic Programming System for Windows リファレンス，マイクロソフト(株)，1994年1月20日.

［6］ Microsoft Excel for Windows Visual Basic ユーザーズガイド，マイクロソフト(株)，1994年2月1日.

執筆者紹介　翠　　秀　幸（Hideyuki Misu）

1965年生．1988年上智大学理工学部卒業．同年日本ユニシス(株)入社．以降，証券システム開発，オープンシステム開発に従事．現在，金融システム第一本部 金融システム3部第4課に所属．

病院におけるEUC
——診療支援における病歴検索システム

End User Computing (EUC) at a Hospital
——A Disease History Inquiry System for Medical Treatment

森　良行

要　約　病院における情報システム化は，医療オーダプロセシング・システムを核とした総合情報システムの構築の時代に入ったところである．このシステムによって一元的に管理された診療行為にまつわる情報は，医師への診療支援システム機能として有効活用の期待が高い．ここで求められるのは，オーダ情報をベースに特定の病名，薬剤，処置や患者個人に注目し，その推移を検査結果などから分析しうる情報へ加工できる情報である．

山形県立日本海病院では，この観点からエンドユーザである医師が，自由に検索・加工の行える病歴検索システムを開発した．このシステムは，オーダ情報をもとに情報検索が行いやすい構造を持つ意味型データベースシステムと，汎用検索システムから構築され，任意の検索加工を実現している．このシステムの利用者インタフェースは，医師が取り扱うパソコン操作の一貫として病歴検索が行え，データベース構造の知識を必要とせず，関連付けされている項目を選択できる形態を提供している．

今後，病歴検索システムは，取り扱う情報の網羅性を向上させるために情報の種類の拡大や，広範囲なネットワークの活用による利用層の拡大に対応できる機能を備えていくことが望まれる．

Abstract　The construction of information systems at hospitals has just entered a new phase where total information systems are being built, of which the kernel is a medical ordering system. What has been hoped for is to effectively use medical treatment data, which are collectively managed by such total information systems, for providing medical treatment support for doctors. What is sought here is the kind of information capable of being edited into data that enable users to analyze, on the basis of order infomation and the results of examination, how patients are recovering by picking out any specific disease name, medicine, treatment and individual patient.

The *Nihonkai* Hospital of *Yamagata* Prefecture has developed a treatment history inquiry system that allows doctors, as end users, free inquiring and editing. The system's infrastructure is made up of both a semantic database system so constructed as to provide easier access to required data according to given order information, and an ad hoc inquiry system ; thus making possible any form of inquiry and editing. The user interface of this system helps users, who have no knowledge of the database structure, operate personal computers to retrieve medical history information and sort out related database items.

For a larger coverage of information to be processed, the future requirements of the system include the need to extend the types of data to be handled, and to respond to much more users with the use of large-scale networks.

1. はじめに

病院における情報システム化は，医事会計業務を代表とする各部門ごとのシステム化に始まり，医療オーダプロセシング・システムを核とした総合情報システムの構築の時代にようやく入ったところである．オーダプロセシング・システムは，院内の情報データベース化とネットワークの構築によって経営体質を改善し，対外的には患者サービスの面で，待ち時間の減少という効果を発揮している．

このシステムによって一元的に管理された診療行為にまつわる情報は，医師への診療支援システム機能として有効活用の期待が高い．医師へのフィードバックとしては，単に蓄積されたオーダ情報のみでは付加価値として乏しい．求める情報は，オーダ情報をベースに，特定の病名，薬剤，処置や患者個人に注目し，その推移を検査結果などから分析しうる情報への加工が必須である．意味付けされたデータベースから医師は任意に情報を検索し，自らのパソコンに取り出して研究テーマの合致した分析の実現を望んでいる．

山形県立日本海病院において，この観点からエンドユーザである医師が自由に検索・加工の行える病歴検索システムを開発した．このシステム基盤は，オーダ情報をもとに情報検索が行いやすい構造を持つ意味型データベースシステムと，汎用検索システムによって構築され，任意の検索加工を実現した．

本稿では，2章で病歴検索システムの役割を述べ，3章で病歴管理用のデータベースについて，4章で病歴検索システムの機能と医師が情報利用することを意識した利用者インタフェースについて説明を行い，5章では，病歴検索システムが今後どのように拡大するかを提示する．

2. 病歴検索システムの目的

2.1 病歴検索システムの位置づけ

病院における総合医療情報システムは，業務系システムとしてのオーダプロセシング・システムと，そのデータをベースにした診療支援としての情報系システムから構成され，病歴検索システムは診療支援のサブシステムと位置づけられる．

オーダプロセシング・システムとは，病院内における各部門間での指示（オーダ）伝達の仕組みをコンピュータ化したシステムである．すなわち，病院における診療に関わる作業が，医師の指示により各部門の作業として発生し，それに基づいて別の指示が発生するという流れを保持してコンピュータ化するシステムである．たとえば，医師は患者に対して診断を行い，それに基づいて診療行為の指示をする．その結果として，検査や注射や薬を出すという診療行為が実施され，さらに，医事会計という指示が出され，精算するという一連の流れがコンピュータ化される．したがって，このシステムは，医師が診療の指示を行うために端末（含むパソコン）からの入力操作を行うことを前提としたシステムであり，また医師の診断の局面では特定の患者の診療歴を必要とするので，患者単位で診療行為に対する診療情報が管理される．

オーダプロセシング・システムは，患者単位で診療情報を管理し，診断すべき患者の病歴を提供するので，その患者への診療支援に役立つ．しかしながら，この診療情報は，投薬歴，検査歴，注射歴等の病歴に基づいた診療支援や，病歴データからの統

計情報の入手には利用することができず，そのような場合，医師は個々人のカルテを検索していた．病歴検索システムは，診療情報に対して医師の自由な情報検索を可能にし，医師の行う診療や研究を支援するものである．

日本海病院では，日本ユニシス（当社）のオーダプロセシング・システム（MEDI-ORDER/EX：以後 MO/EX と呼ぶ）を導入するとともに，診療支援を目的とした病歴検索システムの導入を必要とした．該病院における診療支援としての情報系システムには，医師の自由な汎用検索，すなわち，カルテに記載されている内容に対して，条件を設定して対象となるカルテを抽出するのと同じレベルの情報検索機能が要求された．最終的には，現行の MO/EX で格納されているデータに基づき，病歴検索に必要なデータベースを構築し，定期的に更新し，汎用の検索システムを加え，要求された病歴検索システムを提供することになった．

2.2 病歴検索システムのねらい

病歴検索システムの狙いは，提供するデータベースを「患者の医学的に関わりのある私的情報と医学的状態に関する全ての診療記録を整理・保存したもの」と定義して，医師が必要とする見方でデータを抽出できる仕組みを提供することにある．

提供するデータベースを構成する病歴データは，患者の診療記録であるカルテを原本としている．カルテには，診療行為から得られるすべてのデータが記載されている．たとえば，血液検査・肝機能検査（たとえば，GOT）・投薬量などの数値データ，薬品名・所見・抗体などの＋/－などの質的データ，X 線・CT/MRI などの画像データ，心電図・脳波などの時系列データ，診療指示に伴う医師のコメント情報などである．しかしながら，カルテの内容は医師個人の記述にまかされている部分もある．病歴検索システムとして管理すべきデータは，医師との検討の結果，オーダプロセシング・システム上で管理されているカルテの数値，文字列のデータと，カルテ上に記載されているキーワード情報から構成することにした．

このようなデータベースに対しての検索システムの狙いの一つは，決められた画面入力により，決められた項目が戻される，いわゆる定型照会ではなく，データベース上で定義されている全項目が条件式に設定でき，データ関連を持つデータがすべて一度に抽出できることにある．この汎用抽出機能によって，医師が薬品・病名・麻酔などの条件を設定することにより，対象の患者の診療情報を検索したり，対象患者を抽出したり，また診療時に，同一病名の患者の病歴・投薬歴などを事前に把握し，当該患者の治療の参考にする診療支援が可能になる．さらに，ある病名や診療行為を病歴にもつ患者群の検索・抽出，ある薬品を投薬した患者群の検査歴の検索・抽出などの研究を支援することも可能となる．また，病名単位の性別，年齢別の患者数の把握などの統計解析も行える．

二つめの狙いは，単に診療行為の結果としての情報検索にとどまらず，診療の履歴データに基づいて，医師がそれらのデータを独自に加工・編集することを支援することである．この場合，医師はパソコンを使用してデータを検索して加工・編集を行う．このため，病歴検索システムは，データベースを検索する機能に加えて，パソコン上に検索データを取り込む機能を持ち合わせていなければならない．一度パソコン上にデータを取り込むことができれば，パソコンの市販ソフトウェアによって加工編集が

できるので，このシステムでその機能を提供する必要はない．

三つめの狙いは，いわゆる EUC (End-User Computing) の世界で使用できる形態のシステムを形成することである．EUC の一部としてこのシステムを使用するためには，ホストシステムの端末として起動していることを意識させないパソコン上のソフトウェアの提供が必要となる．パソコンのプログラムを立ちあげる，あるいは，終了させることで，ホストシステムとの接続や切断を可能にし，利用者にとってパソコン操作ができれば使用できるという環境が提供できなければならない．これは，ホストに対するパソコン起動型システムである．今までのホストシステムと接続されて稼働しているソフトウェアが，他のパソコン・ソフトウェアの操作と異質に感じさせている部分が多々ある．この部分を排除しない限り，病歴検索システムを提供しても一部の医師しか実際には使用しないことになり，このシステムの目的である診療行為により得られた情報の医療現場へのフィードバック・後利用が図れない．また，検索操作も日本語名による検索項目の指定や，マウスを用いた GUI (Graphical User Interface) による操作を備えたソフトウェア機能の提供も必要である．

日本海病院に提供する病歴検索システムは，医師による蓄積された病歴情報の有効活用を目的としたシステムであり，上記のような狙いを達成する機能要件を備えたソフトウェアでなければならなかった．

3. 病歴管理のデータベース

3.1 SIM データベースの概要

病歴管理のデータベースは，病歴情報の検索をさまざまな見方によって行えるように SIM (Semantic Information Manager) データベースで作成した．SIM データベースは，意味型データベースモデルを実現している A シリーズにおけるデータベース管理システムである．病歴情報システムへ，なぜ SIM を適用したかを述べる前に，SIM の用語を簡単に説明する．

- エンティティ (ENTITY)

  データの最小単位，実世界の興味対象の構成要素（たとえば，患者一個人，ある薬，ある注射など）を表す．

- クラス (CLASS)

  同じ属性（後述）をもつエンティティの集合で興味対象（たとえば，患者の集まり，薬の集まり，注射の集まりなど）の集合である．SIM 上でクラスを定義し，そのクラス内のエンティティを生成・更新・削除といった操作を行う．

- サブクラス (SUBCLASS)

  クラスが形成するエンティティ集合の部分集合で，クラスの特性以外にサブクラス固有の特性を持つことができる．たとえば，患者の集合に対して，死亡患者は部分集合を形成するので患者のサブクラスとなる．

- 属性 (ATTRIBUTE)

  属性は，エンティティの特性を表す要素で，クラス，サブクラスの要素として定義する．属性には，データ値属性 (DVA) とエンティティ値属性 (EVA) がある．

●データ値属性（DVA）

データ値を持つ属性（たとえば，患者の名前や住所，血液型など）である．

●エンティティ値属性（EVA）

クラス内のエンティティとエンティティの関係，および別クラス間のエンティティとエンティティの関係を表す属性．このエンティティ値属性を用いれば関係付けられた相手エンティティの属性を参照することができる．

●単一値属性（SV）と複数値属性（MV）

エンティティの属性（データ値属性，エンティティ値属性に関わらず）で複数の値を持つ属性は複数値属性，単一の値しか持たない属性を単一値属性と定義する．

3.2 病歴管理のデータ構造

病歴を管理するための情報には，患者の固有情報（氏名，生年月日，血液型，体重等），その患者に行った診療情報（診断日，診療科，担当医師等），診断に基づく診療行為情報がある．各患者は，病歴情報として診療情報と診療行為情報を持つ．診療情報では，診療を行った回数の履歴を必要期間保持することになる．診療行為情報は，診断情報に付随して発生するものであり，投薬や注射行為であれば使用する薬，量，単位，医師コメントの情報を個々に管理しなければならず，検査であれば実施された検査の数だけ情報が必要となる．図1に，その概念図を示す．この概念図での関連は，データの発生とその管理を考えた時の関係であり，病歴検索としての関係，すなわち，診療行為を事象別に分類した関係を示してはいない．

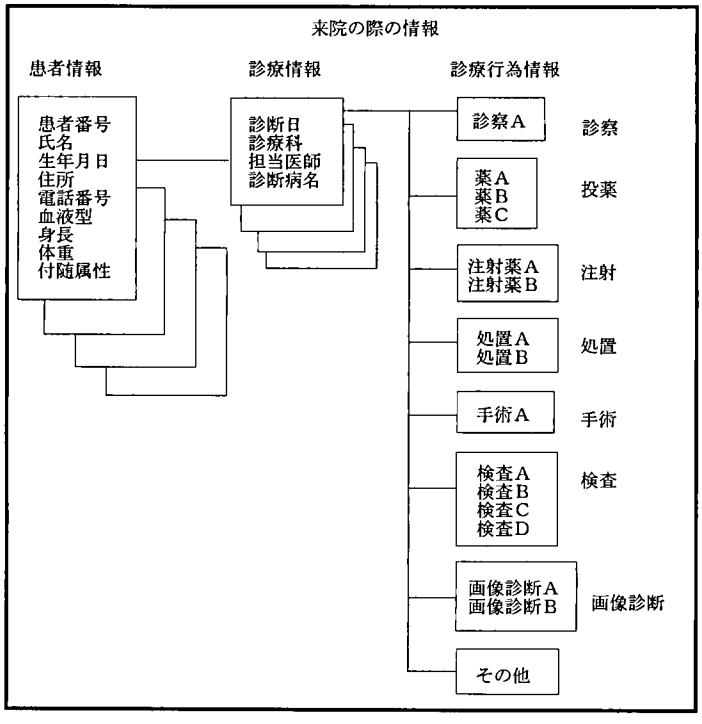


図 1　病歴情報データの概念図

病歴情報は，オーダプロセシング・システムの情報として MO/EX のデータベース上に作成される．このデータベースでは，病歴情報を患者単位に管理するため，患者をまたがった検索，たとえば，ある期間に特定の薬を使用した患者の抽出する検索を行うと全患者の病歴情報を参照して該当の患者を抽出することになる．このような検索形態では，病歴検索ができないので病歴検索に適した形態でデータベースを展開し直す必要がある．すなわち，病歴管理のデータ構造を表現したデータベースを作成することである．

病歴情報を利用する医師が，自らの診療や研究に合った自由な検索という要求を満たすためには，データベーススキーマ上でデータ間の関連が表現でき，かつ，その参照整合性を定義できるデータベース管理システムが必要である．また，医師が行う検索操作において，必要とする関連情報を参照でき，その関連データに対する条件が設定できる検索ができなければならない．このような関連情報の検索は，ネットワーク型のデータベース（A シリーズの DMSII データベース）を使用した場合には，データ関連のために関連テーブルを作成し，各データ間のキー項目を用途に応じて設定して，その関連操作をユーザプログラムのアルゴリズムによって実現することになる．SIM データベースでは，エンティティ間の関係をエンティティ値属性としてスキーマ上で表現できるため，データベースの項目（SIM では属性と呼ぶ）を参照する形態で取り扱うことができる．このため，ネットワーク型に比べて，プログラムでの処理が非常に軽減され，汎用検索を考えた場合にも取り扱いが容易になる．この汎用検索は，SIM が備えているデータベース操作言語の OML（Object Manipulation Language）インタフェースを使用すれば，実行時に OML を生成し，それを解釈・実行することで直接データベース操作ができるので容易に行うことができる．病歴検索システムでは，パーソナルコンピュータ（パソコン）上で OML を生成してホストシステムへ送信して，ホストシステムのデータベースよりパソコン上にデータを抽出する仕組みを提供している．

また，データベース上に保存されているデータに対する論理的な整合性は，DMSII データベースではユーザプログラムで検証していたが，SIM においては，データ値属性単位，エンティティ値属性単位の定義が可能なので，参照整合性をデータベースで保証できる記述をスキーマ上に反映できる．このため，ユーザプログラムや，原始データの検証誤りによるデータの破壊を回避することができ，データの保守を効率良く行える．

日本海病院の病歴情報データベースのスキーマを図 2 に示す．このスキーマは，患者の診療情報が中心となって各情報への関係が結ばれている．各患者ごとの診療歴が診療情報に納められており，その診断に伴って行われた行為，すなわち，診察・処置・検査・投薬・注射などの診療行為が行為情報に繋げられている．このスキーマ上では，診療情報と診療行為情報の関係が一対多の関係になっている．これは，診療行為情報が投薬・注射・検査等のその行為別に区分けされ，各行為の中で複数の種類に分別して管理する必要があり，実際の行為では，これらの行為を複数実施（複数の種類の注射をうつこと）することになるためである．このような診療行為情報の関係をスキーマ上でどの様に定義するかが，病歴情報データベースの設計上の留意点になる．

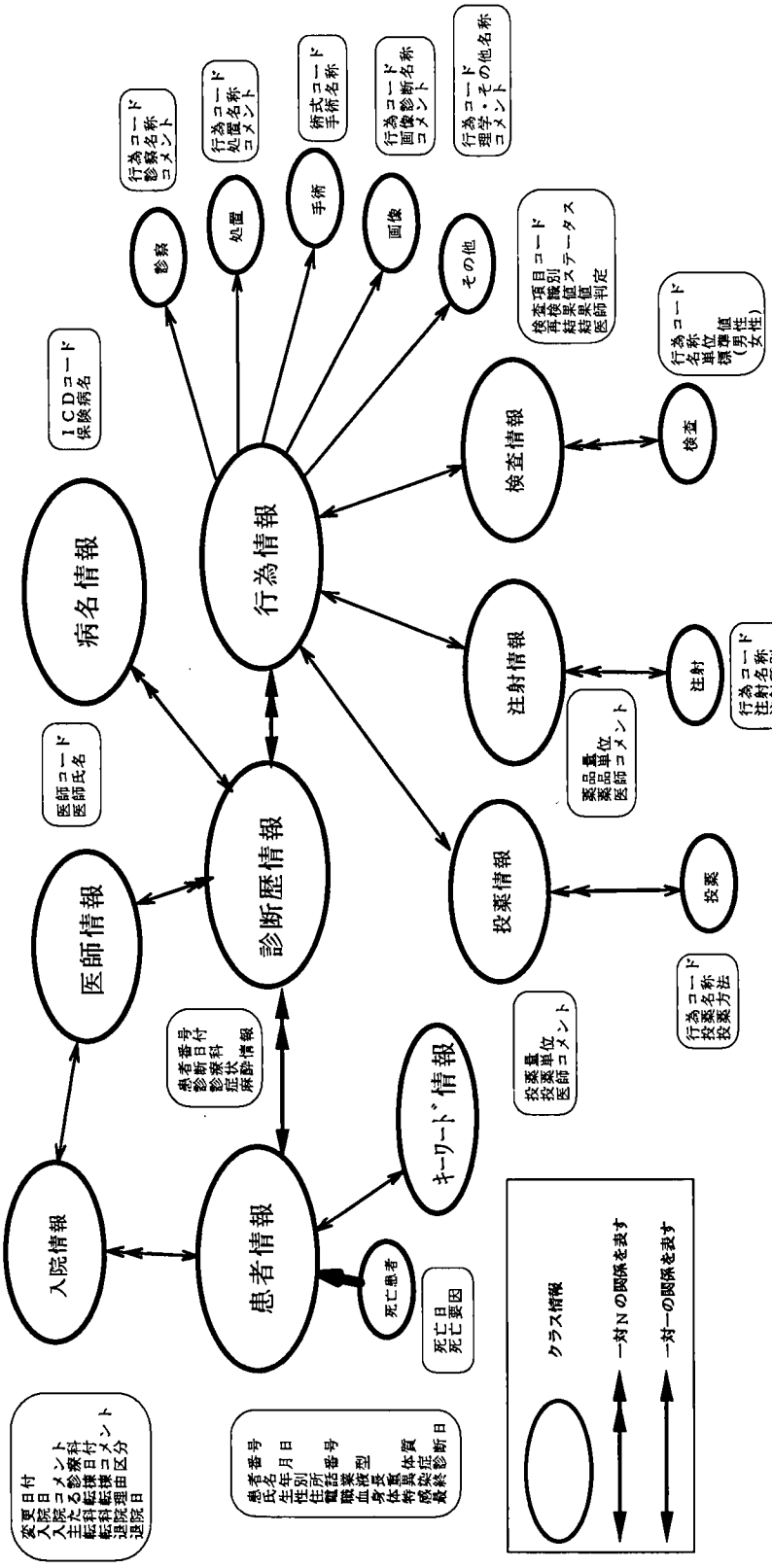


図 2　病歴管理データベーススキーマ図

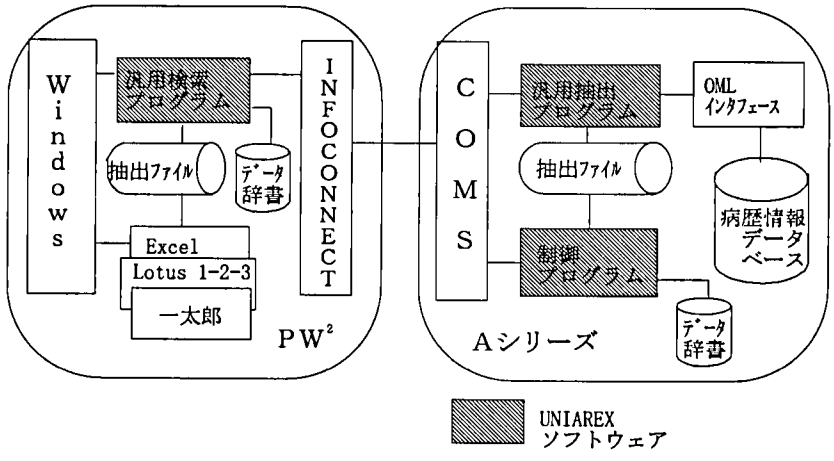
図2のスキーマ上の行為情報クラスは，上記のような関係を取り扱うためにつくられた情報集合である．このクラスは，診察をどのように行ったか，どのような処置をしたか，どのような手術が必要となったか，といった行為名称を得るための関係と，診断にともなって使用する各々の薬・注射の名称とそれらをどの程度使用したのか，どのような形式で投与したかといった情報や，検査の名称とその検査結果の値を行為情報として参照できる関連を持っている．このクラスは，病歴検索の利用者が，診療行為情報を検索して表展開した場合に，参照しやすい形式でデータを抽出できるように作成されている．

行為情報クラスの属性は，ほとんどが関連情報となっていて，一つの診療歴に伴う診療行為をまとめる機能を有する．このため，行為情報クラスから各診療行為のクラスには一対一の関係になるので，SIM が取り扱うデータ抽出の形態が，行為情報クラスを基にして各診療行為の情報をまとめて一行で抽出できる．すなわち，注射情報・投薬情報・検査情報などが一行に並べて抽出可能である．行為情報クラスがなく，診断歴情報クラスより直接的に各診療行為のクラスに関係が繋げられていると，これらの関係はすべて一対多の関係となる．この場合，各診療行為のクラスの持つ情報は診断歴情報クラスから見れば，複数値項目として取り扱われる．したがって，診断歴情報クラスを視点とした検索では，各診療行為のクラスの情報は，クラスごとに異なる行に表示されることになり，行数が増えて，同一データの表示が多くなり，照会上非常に乱雑になる．利用者の手間を省くためにこのようなクラスを作成してある．

4. 病歴検索システム

4.1 病歴検索システムの構成

日本海病院における病歴検索システムの検索部分は，当社のデータベース汎用検索システム UNIAREX（UNIsys A series Report and EXtract）のメニューインタフェース版を使用している．このソフトウェア構成を図3に示す．



Windows, Excel は米国 Microsoft 社の商標である．Lotus 1-2-3 は米国 Lotus Development 社の，一太郎は(株)ジャストシステムの登録商標である．

図 3　UNIAREX の検索ソフトウェア構成

UNIAREX は，A シリーズ上で稼働するソフトウェアとパソコンの WINDOWS 配下で稼働するソフトウェアを提供し，これらのソフトウェア間で，クライアント・サーバ型のシステムを形成する．パソコンと A シリーズの接続は，パソコン上の INFOConnect（通信インタフェース用ソフトウェア）を経由して行うので，INFOConnect が支援する接続形態であれば，回線や LAN のいずれであってもかまわない．パソコンから SIM のデータベース操作言語である OML (Object Manipulation Language) をホストシステムに送信し，問い合わせを実行し，抽出結果データをホスト側で作成後，パソコン上にダウンロードする機能を持っている．

図 3 の検索ソフトウェア構成において，A シリーズ上のプログラムとして次の二つがある．

- ●制御プログラム

  パソコンソフトウェアとのトランザクションの受け渡しを行うプログラムで，このプログラムによって，データ辞書・抽出ファイルのパソコンへのダウンロード，検索要求の汎用抽出プログラムへの渡し処理が行われる．

- ●汎用抽出プログラム

  パソコンからの検索要求である OML 構文を受けて，SIM の OML インタフェースを経由して病歴情報データベースよりデータを抽出し，抽出ファイルを作成するプログラムである．

パソコン（PW²）上には，検索対象のデータベース固有のプログラムは存在せず，汎用検索プログラムを提供した．この検索プログラムは，データベースのスキーマ情報を A シリーズからダウンロードしてデータ辞書として取り込み，その辞書情報に基づいて当該のデータベースの検索処理を行うプログラムである．データ辞書は，A シリーズ上でデータ辞書生成ユーティリティによって作成する．

データ辞書には，データベースの項目情報・項目名とそれの日本語名への変換情報，各クラス間の関連と経路情報，データベース上の項目値を名称変換する情報等が格納される．この結果，パソコン上で指示された検索対象項目と検索条件から，当該データベースへの検索要求としての OML 構文を生成することが可能となる．このプログラムは，検索要求として OML 構文を A シリーズに送信し，その結果として検索要求受諾を受け，次に抽出データをパソコン上に取り込むためのデータ要求を送信し，データを受信するといったクライアントとしての機能を持つ．A シリーズ上では，パソコンからの検索要求の OML 構文に基づいて当該データベースよりデータを検索し，抽出ファイルを作成する．また，パソコンよりデータ要求があれば，抽出ファイルをパソコンに送信するといったサーバとしての機能を持つ．

4.2 病歴検索の利用者インタフェース

4.2.1 利用者インタフェース要件

病歴検索システムの利用者インタフェースは，その対象が，医師または医療従事者であることに基づいて検討する必要がある．すなわち，コンピュータを利用する立場であって，コンピュータを利用させる立場にない人が操作するのであり，ホストシステムに携わったことのない人が，自分の意思でパソコンを使用してホストシステムから必要とするデータを取り込み，そのデータを加工・編集するといったエンド・ユー

ザ・コンピューティング（以後，EUC という）を前提としなければならない．

病歴検索システムにおいて，利用者はパソコンを操作することはできるが，ホストシステムの操作はできないことを前提としている．すなわち，病歴検索での利用者インタフェースはすべて，パソコン操作の一貫としてできなければならず，ホストシステムに対して行う必要のある操作は，直接利用者が行う必要がないようにしなければならない．パソコン上のプログラムが，ホストシステムに対する指令や，ホストシステム端末として利用者に操作させるインタフェースを代替して行うことで，利用者は，他のパソコンソフトウェアと操作性が変わらない病歴検索の操作が行えることになる．

病歴検索システムでは EUC が前提となっているので，検索結果データは，表計算や資料作成に利用される．すなわち，パソコン上の表計算ソフトウェアを使って利用者が自らデータを投入したと同じ感覚で，検索結果データが扱えなければならない．このため検索結果ファイルに，検索したデータの各項目が，どの項目かを判断できるように見出しを付加している．また，病歴管理データベース上のデータには，性別，診療科コード，感染コードのようなコード化されたデータがある．これらのデータをパソコン上に取り込んで使用する場合，コード値ではなくその値に対応する名称を必要とすることがある．このようなデータは，コード値で抽出するか，名称に変換して抽出するかの選択ができなければならない．さらに，これらのコード値項目を用いた検索条件の設定を行う場合にも，名称一覧の中より選択できることが必要となる．

4.2.2　IQF による利用者インタフェース

当初，病歴検索システムとして SIM の持つ汎用検索ツールである IQF (Interactive Query Facility) を使用することを検討した．検討結果として，IQF を使用しなかったのは，以下の三点にある．

IQF によってパソコン上に検索データを取り込むための方法は，最初にパソコン上で，データ転送エミュレータ（DTE：Data Transfer Emulator）と呼ばれるデータ転送ソフトウェアを起動して，ホストシステムのダム端末にする．このとき表示される画面から，ホストシステムの指令そのものを入力し，ホストシステム上の IQF のプログラムを実行する．これは，パソコン上で行う一連の操作の中で，ホストシステムに対する操作を行うことになる．すなわち，利用者は，IQF を起動するために使用するホストシステムの指令を理解する必要があるし，パソコン操作に異質な操作を強要されることになる．また，DTE は，ダム端末を模倣しているので，IQF のプログラムを終了させることなくパソコン側を終了させても，ホストシステム上の IQF のプログラムは，起動されたままで，次に再度 DTE を起動すると，以前の状態を継続することになる．利用者は，この様な状態に対処しなければならないので，ホストシステム上でのプログラムの起動・終了を意識して操作することが強いられる．

IQF を使用した検索操作は，ホストシステムと接続するダム端末仕様になっているので，すべてキャラクタ・ユーザ・インタフェース（CUI：Character User Interface）が前提となる．検索項目は，スキーマ定義で使用されたデータ名の一覧が，クラス単位表示されるのでこれにマーク付けすることで選択する．このデータ名は，スキーマ定義に使用するオブジェクト定義言語（ODL：Object Definition Language）で記述

されたものであり，ローマ字と英語混じりのデータ名で，日本語名称で表現できない．また，クラス間のデータ関連をたどって，他のクラスのデータを抽出しようとすると，データベーススキーマがわからないとデータ抽出ができないという問題もある．すなわち，クラス間の関係が，どのエンティティ値属性（EVA）で関連づけられているのかを知らないと指定できないのである．IQF を用いた病歴検索では，スキーマ図を参照しながら，必要なデータ項目がスキーマ上で，どんな名前で定義されているかとか，データベース構造まで意識させた検索を，利用者に強制することになる．

コード値データに関しては，SIM の機能として名称変換があるが，この名称は英数字しか使用できないので，日本語名称への変換はできない．また，条件設定時には，名称によって設定したくとも，名称一覧を参照して設定することができない．

医師や医療従事者が，IQF による病歴検索を行おうとすると，病歴検索システムの操作以外にホストシステムの操作や病歴管理データベースの構造を理解する必要がある．また IQF の操作性が，他のパソコンソフトウェアとは異質であり，親しみやすくない．病歴検索システムの利用者が，IQF を用いて病歴検索を行うことは困難であると判断し，IQF の上記のような問題点を解決した UNIAREX の提供を決めた．

4.2.3 UNIAREX による利用者インタフェース

UNIAREX は，パソコン上で稼働するソフトウェアと A シリーズ上のソフトウェアが連動して稼働するので，パソコン上の汎用検索プログラム（4.1 節を参照）の操作によって，ホストシステムとの接続・送信・切断を行う．利用者が病歴検索を行うには，この汎用検索プログラムのアイコンをマウスでクリックするといった他の WINDOWS プログラムと同じ操作によって立ち上げ，このプログラムの基本画面（図 4 参照）を表示する．図 4 の病歴検索の基本画面上のメニューバーにある接続のプルダウンメニューより接続・切断を選択して行うことができる．WINDOWS のプログラムと同じような操作性を提供することは，利用者が病歴検索を行いながら他のパソコンソ

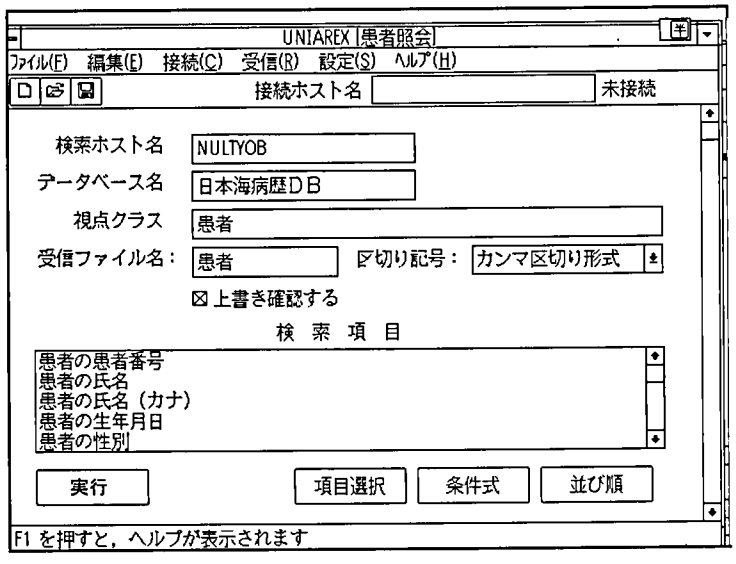


図 4　病歴検索の基本画面

フトウェアを使用する EUC 環境上では重要な要素である．利用者にとって，パソコンのプログラムの操作か，ホストシステムの操作かの頭の切り替えを行う必要がないからである．

パソコン上の汎用検索プログラムは，ホストシステムから取り込んだデータ辞書を参照することにより，そのデータ辞書に登録されているデータベースの検索を可能にしている．すなわち，使いやすい利用者インタフェースを提供しながら，このデータ辞書を介して，検索要求としてホストシステムが望む OML 構文への変換を可能にしている．OML 構文は，以下の例にあるように，スキーマ定義で用いられ項目名を使用して作成する．

[OML 構文例]
Example of OML statement
FROM KANJYA RETRIEVE KANJYANAME, KANJYANAME-K, KANJYA-ID
WHERE SHINDAN-DATE OF SHINDAN = 10/15/1994
＊ 1994 年 10 月 15 日に診断した患者の氏名，ふりがな，患者番号を照会する OML 構文

利用者がデータベース項目を選択するためには，その項目の意味を適切に表現できる項目名になっている必要がある．現行のデータベース定義言語では，英数字とハイフンしか項目名に使用できないので，項目の意味を適切に表現できない．[OML 構文例]にあるように，KANJYANAME が患者氏名であり，KANJYANAME-K が患者かな氏名を表しているが，明らかに日本語で表現した方が項目の意味を理解しやすい．このため，UNIAREX では，データ辞書上に定義名と日本語項目名の変換テーブルを持っている．これにより，利用者は，図 5 の項目選択画面上の日本語項目名の一覧より検索項目を選択でき，ホストシステムへの検索要求は，変換テーブルより定義名を

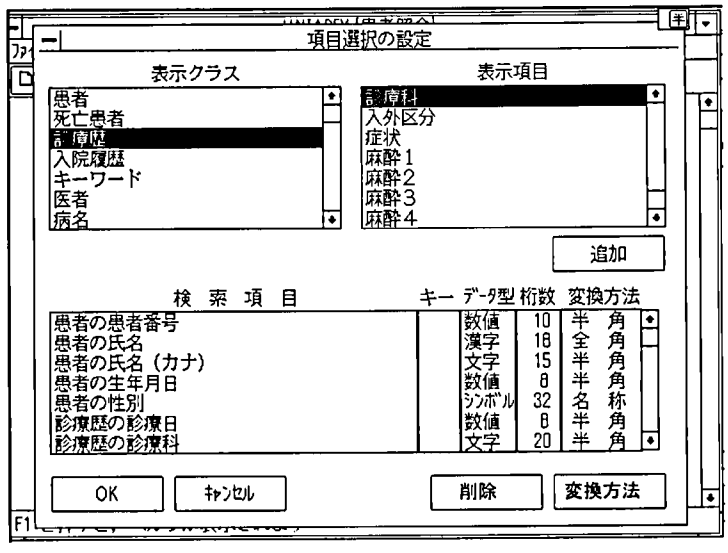


図 5　検索項目の選択画面

使用してOML構文を作成できる．

病歴検索において，検索対象のデータがどのような関連を持って構造化されているかは，利用者にとって興味のないことである．利用者は，病歴情報データがどのような関連を持つかは，図1で提示したデータ概念図のレベルで理解できればよい．この図から，利用者は，情報のくくり（クラス）にどのようなものがあり，そのくくりの中の項目に何があるかを知り，検索対象の項目を探し出すことができるはずである．しかしながら，情報のくくりと他の情報のくくりの関係，すなわちクラスとクラスとの関連がどの様につながっているのかは，利用者が持つデータの概念には兼ね備えられていないし，これを理解させることは，利用者にデータベースのスキーマの理解を強要することになる．

利用者が要求する病歴情報を得るには，図2のスキーマ図にあるようなクラス間の関連経路を使用して，関連クラスの持つ情報を検索する必要がある．病歴検索システムでは，データベースのスキーマ情報と関連データへの経路情報を，4.1節で述べたようにデータ辞書としてパソコン上に取り込み，利用者が指定しなくともいいようにしている．パソコン上の検索操作は，興味対象のデータをもつクラス（視点クラス）を選択することで，図5の検索項目の選択画面のような視点クラスに関連したクラスを，表示クラスの一覧として表示し，表示クラスを選択することでそのクラスの項目一覧を表示する．利用者は，情報のくくりを表示クラスより選択して，必要とする項目を選択することが可能となり，データ関連を意識することなく項目が選択できる．また，選択した項目はすべて条件項目として使用できるので，データ間の関連を意識することのない検索操作も可能となる．視点クラスと関連を持たないクラスは表示されないのでその項目は選択できない．

コード値データに関しては，データ辞書上にコード値データの指定ができ，使用する名称変換テーブル名が設定できる．項目選択時点で，コード値データの項目が選択されると，変換方法として「名称」を選択でき，この項目を条件指定で選択すれば，図6にあるような名称一覧が表示される．図6は，診療科コードを選択した時の名称一覧の表示である．利用者は，この一覧より選択して条件値を設定できる．名称変換テーブルは，データ辞書と同様にホストシステム上で保守され，パソコン上にダウンロードする形態をとっている．

データ辞書，名称変換テーブルは，ホストシステム上で保守されるが，これらに変更があれば，パソコンの利用者がホストシステムと接続を行った時に，「変更されている」という警告が促されるので，必要であればダウンロードできる．

UNIAREXが提供する機能は，4.2.1項で提示した利用者インタフェースの要件を満たしており，IQFでの問題点も解決しているため，日本海病院では，病歴検索システムで使用することになった．

4.3 病歴検索システム適用にあたっての配慮

病歴検索システムを実際に使用するにあたり，次のような配慮がなされた．

まず第一に，利用者にホストシステムを意識させずに，パソコンソフトウェアの操作環境での操作を可能にするような利用者インタフェースの準備である．

次にパソコンソフトウェアの操作環境で，利用者が適切な項目を選択できることへ

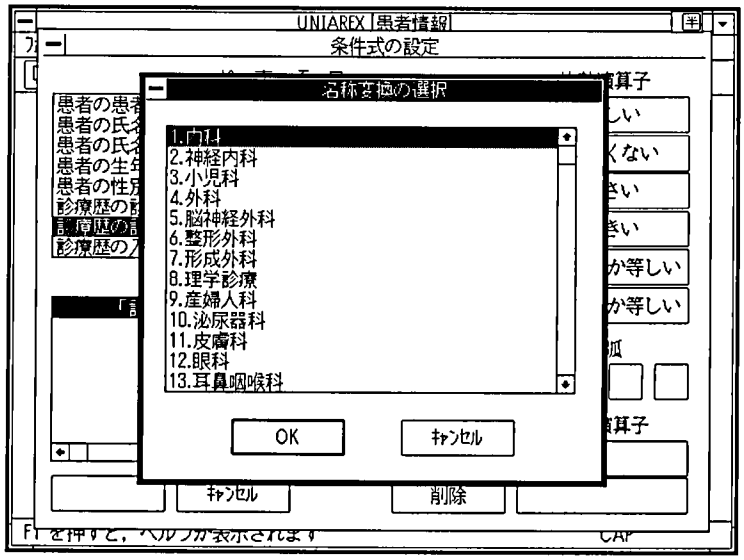


図 6　条件設定の名称一覧画面

の配慮である．日本海病院では，データベース項目に対し，医師や医療従事者が日頃使用している日本語名称を決めて登録し，利用者は，その日本語名称を用いて項目選択ができる．

最後に，検索処理時間に対する利用者への配慮がある．利用者が行う病歴検索の中には，検索結果が戻されるまでに時間を要することがある．そのような場合，利用者は，その検索時間が妥当かどうかは判断できないし，データベーススキーマ上すべての検索時間を早くすることもできない．汎用的な検索を行う上での検索時間は，数秒から数時間の幅を持つ可能性がある．これは，データベースのデータ量によってさまざまである．

日本海病院では，この対応としてデータベーススキーマを理解している管理者が，効率の良い検索要求をあらかじめサンプルとして作成し，それを利用者に提供することで利用者が妥当な検索時間で病歴検索を行うことができるようにした．

病歴検索システムには，検索要求に標題をつけて検索指示ファイルとしてパソコン上に保存する機能を持っており，このファイルを利用者に配布した．利用者が検索を行う場合に，検索指示ファイル一覧を参照して類似した検索要求を見つけ，その検索要求を修正して利用者が新たな検索を行うことができる．また，修正した検索要求を別の検索指示ファイル名で保存することもできる．

5. 病歴検索システムの展望

病院における EUC システムとしての病歴検索システムは，エンドユーザである医師のアドホックな検索要求に対して，その情報を管理しているオーダプロセシング・システムのホストシステムとしてのコンピュータ知識を必要とせずに，求める情報を得るという目的を達成できた点で，一定の効果を上げたと評価できる．

今後の病歴検索システムの向かう方向は，情報の網羅性の追求と広範囲なネットワークの活用による利用層の拡大にある．

情報の網羅性を見ると，今回のシステムは院内のオーダプロセシング・システムで取り扱われる情報のみであり，図2で表現された情報をすべて取り込み得ていない．院内の情報は，各部門の独立したシステムで保有している情報も多く，オーダプロセシング・システムとの接続性を確立させることが望まれる．とりわけ，画像診断情報，病理検査情報や理学療法の諸情報は，数値や文字によるテキスト情報のみならず画像等のアナログ情報を含むことから，病歴データベースへの取り込みについて，データ表現やデータ量と妥当な検索時間についての研究が必要である．

利用者層の拡大については，現状の院内情報から外部に開かれたシステムと位置づけるべきであり，研究的観点から有益な情報は院内に留めることなく，広く提供できる仕組みを備えることが求められる．

当然のことながら，情報は一方的なものでなく相互に流通すべきものと考えるならば，施設間の相互の利用者は，相互のコンピュータの知識を必要とせず目的の情報を得られなければならない．このためには，先ずネットワーク上に流通する情報の表現方法の標準化が必要である．現在，インターネットや地域独自のネットワークによって，情報を流通させることは可能であり現実に一部実施されている例もある．

さらに，画像等の大量データの流通を考えたとき，行政府が進めるネットワークの整備は確実に医療情報の流通を促進させるはずである．病歴情報の流通は，患者のプライバシをいかに保護し得るかが今後の課題であるが，データベース上の氏名，住所等の該当情報に対する保護や置換操作の機能のみならず，ネットワーク上のテキストの暗号化機能などがEUCを支える基盤ソフトウェアに備わっているのが望ましい．

6. おわりに

本稿の病歴検索システムの紹介は，オーダプロセシング・システムで蓄積されたデータを利用者が自由な観点より検索できるシステムとして提供した例である．診療支援における病歴検索システムは，オーダプロセシング・システムで蓄積したデータ以外に様々な付加情報を必要とする局面があると考える．この対応として，付加情報にはどのような種類のデータがあるかの分析が必要であるし，これらのデータを病歴検索システムのデータベース上に直接反映できるような機能拡張が今後必要となると考える．このように本システムは，病歴検索システムの第一段階であり，機能拡張を繰り返すことで充実したシステムになると考えている．

最後に本システムの構築にあたり御指導，御支援いただいた山形県立日本海病院の皆様に感謝の意を表したい．

参考文献 ［1］ 佐柄木秀郎，“オーダプロセシング・システムの基本概念とそのシステム化”，技報43号 Vol.14 No.3 1994年11月.

執筆者紹介　森　　良　行　(Yoshiyuki Mori)

昭和 28 年生．50 年上智大学理工学部数学科卒業．同年日本ユニシス(株)入社．金融機関の客先サービスに従事した後，A シリーズの利用技術サービスに従事する．現在，システムプロダクト本部 A シリーズソフトウェア三部に所属．

レプリケーション機能を用いた分散システム構築

The Construction of Distributed Systems Based on Database Replication

山田和司, 今泉正雄

要　約　最近エンドユーザ・コンピューティングを実現する環境として，分散データベースシステムが注目を集めている．本稿では，分散データベースシステムを作成する場合のシステム構築方法として，レプリケーション機能を用いることを提案する．

分散データベースを構築したときの処理は，単一のデータベースシステム内の処理とは違った工夫が要求される．ここで登場するのが，2PC(2フェーズコミット)であり，またレプリケーション機能である．

本稿ではまず，2PCとレプリケーション機能とを，分散システムの構築という観点から比較検討する．また，レプリケーション機能についてはベンダによって機能や実装が異なることから，筆者らの取り扱うUNIX上のいくつかのDBMSに対して可能な限り客観的な調査・比較を試み，それぞれのシステムにおけるレプリケーションの特徴と有効性を明確にする．最後に，実際のシステム構築において，レプリケーションをどのように用いていくべきかを，いくつかの事例をとりあげ考察する．

Abstract　Today's attention is focused on distributed database systems as a way of making end user computing more effective. This paper recommends a replication approach as a method for building distributed database systems.

Data processing involved in the use of the distributed database requires approaches different from the way data are processed within a single database system; that is, "2-phase commit (2PC)" or "replication".

This paper first compares 2PC and replication functions in view of the effort to construct distributed systems. Then, the authors examine and compare, as objectively as possible, some existing DBMSs that run on the UNIX operating system because the functionality and implementation of replication differ from vendor to vendor; thereby clarifying the traits and advantages of data replication that each DBMS provides. By giving some actual cases, the last chapter discusses how to adapt the replication feature.

1. はじめに

現在多くの企業で分散システムが稼働している．この背景には，UNIXやPCの普及があげられる．UNIXやPCの性能が上がり，システムを構築したときの価格対性能比が向上したことや信頼性が増したことにより，これらUNIXやPCをデータベースサーバとする分散データベースシステムを構築することが多くなっている．

このように，分散データベースの利用と稼働実績が増えるに従い，構築するシステムの形態も，業務処理内容や分散システムの構成に最も適したシステムの構築を考えるようになってきた．実際，企業の中でコンピュータ化される業務は多種多様なものであり，分散トランザクション処理だけが分散システムの処理形態ではないことは明

らかである．このような状況の中で，実用面での効果が高いと言う点でレプリケーション機能が注目されている．

我々は業務において，INFORMIX，ORACLE，およびSYBASEを扱っているが，これらUNIX上の代表的なRDBMS（Relational Database Management System）では，2 PCやレプリケーション機能を実装し*，分散データベースシステムの中心的な役割を演じることを目指している．

一方，分散データベースシステムを構築する立場から見ると，さまざまなシステム構成の可能性が考えられ，はたして，どの機能を利用するのが最も効率の良いシステムを構築できるのかの選択が難しくなっている．

今後，分散データベースシステムの核として，多く利用されるRDBMSを提供する立場として，レプリケーション機能および2 PCを理論，実装，さらに応用という一貫した観点で調査しまとめることにより，分散システムを構築する時のヒントを提供できると考える．

2. 分散データベースの構築

企業内でエンドユーザが活用しているデータの分散化傾向は，UNIXやPCといった，ロー・エンド・システムの性能や，処理能力の飛躍的な向上と共に加速している．従来，これらのデータは基幹業務とは関係のない範囲で取り扱われることが多かった．しかし，厳しい企業環境の中で勝ち残っていくためには，企業内に蓄えられている情報を全て活用できるかどうかが，大きな鍵を握っている．これは，「分散データベースシステム」をどう構築し活用するかが，企業の戦略として重要であることを意味する．

ここまで，「分散データベース」という言葉を漠然と使ってきたが，本稿でのこれからの議論に当り，共通した概念として文献[1]の「分散データベース」の要件を考えることにする．

これは，次の12件からなる．

- ローカルサイトの独立性の保証
- サイトの同等性
- 分散システムの連続運用
- 位置等価性の保証
- 断片化されたデータに関する位置等価性の保証
- レプリカデータに関する位置等価性の保証
- 分散問い合わせが可能
- 分散トランザクション処理が可能
- ハードウェアからの独立
- オペレーティング・システムからの独立
- ネットワークからの独立
- DBMSからの独立

これら12の要件には，「分散データベース」と言ったときに，そこに含まれる全てのデータは，あたかもローカルなデータベースとして構築されているように扱うこと

* SYBASE System 10，INFORMIX 6.0，ORACLE 7.1など．

ができる，という共通の概念がある．分散データベースシステムを構築する上で，これら12の要件を全て満す必要はないだろうが，「分散データベース」を考える上での指針とはなり得る，と我々は考えている．

本章では2PCとレプリケーションという二つの分散データベース構築のアプローチを，それぞれ先の要件を念頭に考察したい．はじめに，2PCの概要にふれ，分散システムの要件と照らし合せながら，その問題点を指摘する．次にレプリケーションについて，機能面からの定義を行い，これが2PCにおいて指摘した問題を解決することを示す．

2.1 2PCの概要

2.1.1 分散トランザクション

ネットワークにより接続された，複数の独立したデータベースシステムにまたがるトランザクション処理を，分散トランザクションと呼ぶ．分散トランザクションのコミットおよびロールバック処理を行うために必要なプロトコルとして2PCが存在する．

分散トランザクション処理でのコミットは，全ての関係するデータ操作が全てのデータベース・システム上に反映されることを意味し，また，なにかの障害がたとえ関係するデータベースシステム上の一箇所でも発生した場合，全ての関係するデータ操作がロールバックされなければならない．これらシステム間にまたがるコミットおよびロールバック処理はコーディネータと呼ばれるシステムが行う．通常コーディネータはそのトランザクションが入力されたデータベースシステムの役割である．

2.1.2 2PCプロトコル

2PCのプロトコルを図1を例に簡単に説明してみる．

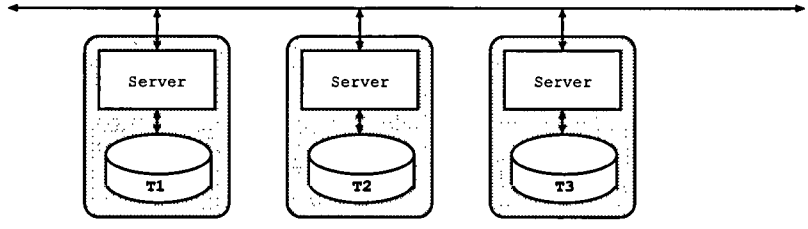


図1 分散トランザクション

まず，T1システムで分散トランザクションを開始する．このトランザクションには，T1，T2，およびT3システムの各データの更新が含まれる．T1システムでトランザクションが開始されたので，通常T1システムがコーディネータの役割を担うこととなる．トランザクションの最後の処理としてコミットまたはロールバック要求が出されると，コーディネータがその要求を処理するために2PCを開始する（図2）．

第一フェーズ（PREPARE）

コーディネータは，T1，T2，およびT3システムに対し，それぞれの持つローカルなトランザクション・ログに情報を書き込むよう要求する(PREPARE)．T1，T2，およびT3システムは，物理ログへトランザクションの情報を書き込むことにより，

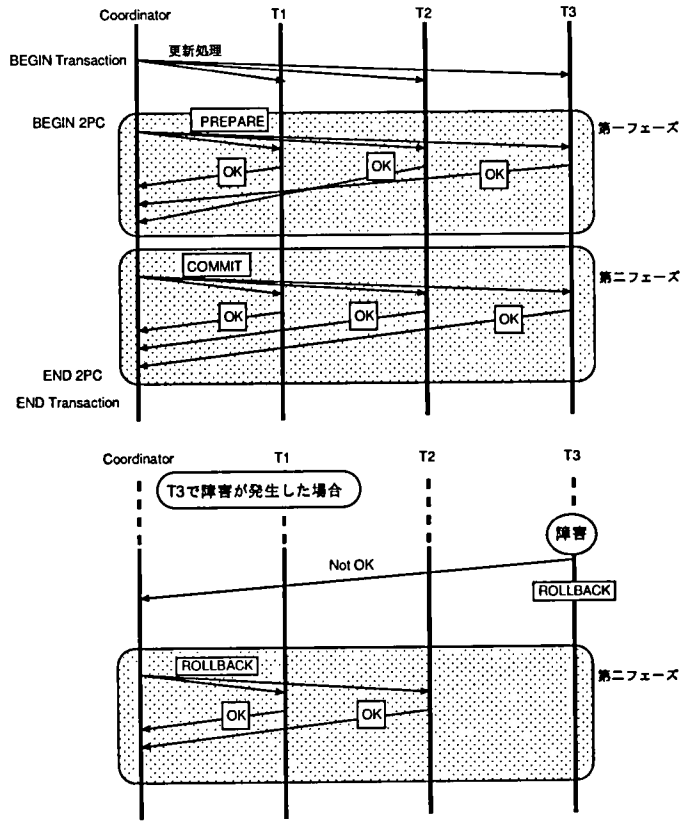


図 2　2PC プロトコル

最終的なデータのコミットまたはロールバックに対する準備を行う．

それぞれのシステムは，物理ログへの書き込みが正常に終了した時点で，コーディネータに対し，“OK”* を送る．なんらかの異常により書き込みできなかった場合は，“Not OK” を送ることとなる．

第二フェーズ（COMMIT/ROLLBACK）

コーディネータは，T 1，T 2，および T 3 システム全てから “OK” が返された場合 “commit” 処理を，また，一つでも “Not OK” の場合は “rollback” 処理をそのトランザクションに対して行うことを決定する．この決定に従って “COMMIT” または “ROLLBACK” 要求を各システムに送る．各システムではこの要求に従って処理が行われ，完了した時点でコーディネータに終了を報せる．全てのシステムから処理終了が返されてトランザクションは終了する．

このように，複雑なメッセージの受渡しが，各データベースシステムの間で行われるが，2 PC を使うことにより，それぞれのシステムの障害やネットワーク障害からトランザクションの一意性やデータベースのデータ整合性を保証することができる．

---

* これらの文字列は，本稿での仮想的なものであり，実際のプロトコルにおいては，実装している RDBMS により異なる．

2.2 2PCの問題点

2 PCは分散システムの基幹となる技術ではあるが，まだまだ2 PCを取り巻く多くの技術的な問題や，適用業務が限定されるなど考慮すべき点が多い．ここでは，2 PCを実際に適用した時に考えられる問題点をあげる．

2.2.1 ローカルサイトの非独立性

分散データベースでは，分散トランザクション処理を行えることが一つの要件として挙げられている．2 PCはもちろんこの要件を満たすためのものであるが，この分散トランザクション処理でのリカバリを考えてみる．

2 PCを利用することによって，一つのトランザクションで処理するデータが複数の独立したサイトに分れていたとしても，データの整合性を保証することが可能となる．ただ，2 PCプロトコルでは分散トランザクションに含まれるサイトが相互に関連して処理が行われるため，あるサイトに障害が発生した場合にシステム全体に及ぶ問題となってしまう．

2.2.2 サイトの非同等性

サイト間に依存関係がないこと，とくにマスタサイトなどの概念がないことが望ましい，とされる．しかし，2 PCプロトコルでのコーディネータとしての役割りを，トランザクションが入力されたサイトが行うこととなってしまい，それぞれのサイトが平等な関係ではなくなってしまう．また，コーディネータはそのトランザクションに関係する全てのサイトと相互に通信をする必要があり，ネットワークの負荷が高くなってしまう．

コーディネータと実際にトランザクションを処理するサイトの関係を考えてみても，各サイトで最終的にコミットするのかロールバックするのかはコーディネータの決定に依存してしまうこととなり，サイトの自立性がなくなってしまう．

さらに，2 PCプロトコルを使用することにより，各サイトまたはネットワークの障害が発生した場合でも，分散トランザクションプロセスは実行され，障害が回復した時にただちに処理が継続されることを期待する．しかし，それらの障害全てに対処可能なシステムを作成することは不可能である．

2.2.3 分散システムの運用の中断

近年，分散環境の中にPCによるデータベースサーバを設置し，分散トランザクションの一つのノードとして利用することが可能となった．しかし，分散トランザクションの中での一つのノードとしてPCが利用されたとき，PC自体の信頼性が問題となってくる．2 PCでは，たとえ一つのノードからでもコーディネータからの要求に返答がないと，その分散トランザクションは終了しない．ロックされているデータによってはそのシステム全体が止ってしまうこともある．このように，あるノードに置かれているデータの重要度やそのノード自体の障害がシステム全体に与える影響が2 PCでは考慮されていない．

2.3 レプリケーションの概要

データベースの世界では，単一のデータを複数のアプリケーションで利用することを目的の一つとしていた．これは，アプリケーションごとにデータを持つことの格納に対する物理的な負担やデータを最新に保つことの困難さを排除するためである．

ところでレプリケーションとはマスタデータの複製（レプリカ）を持つための技術である．これは一見，これまでのデータベースの基本方針と逆行するように見える．しかし，分散システムが発達するに従い，同一のデータを複数のサイトでアクセスできることが要求されるようになると，各サイトで「レプリカ」を持つことの重要性が認識され始めてきたのである．本節では，まずこの利点について，具体的な例をもとに考察する．

もちろんそのようなシステムでは同一のデータを複数持つために，格納の負担の問題以外に，データを正しく更新することの困難さが発生する．次にこの問題についても，データの整合性の観点から取り上げてみる．

最後に，2 PC で見た分散システム構築上の問題と対比させ，レプリケーション機能を考察することにする．

2.3.1 レプリカを持つことの長所

図1の各サイトが，ロンドン，東京，ニューヨークにあり，これらが WAN で接続されていることを示したのが，図3である．

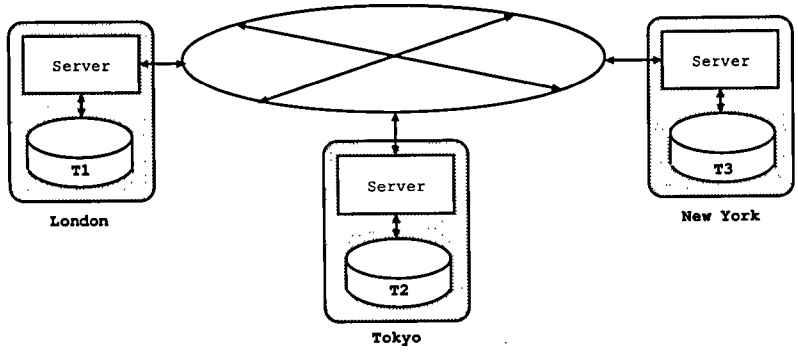


図 3　WAN で接続された 3 サイト

T1はロンドンで，T2は東京で，T3はニューヨークでそれぞれ更新されるが，各サイトで実行される，これらのすべての表を参照するアプリケーションがあるとすれば，図4のように各サイトは互いに，他サイトのデータのコピーを持っていると都合

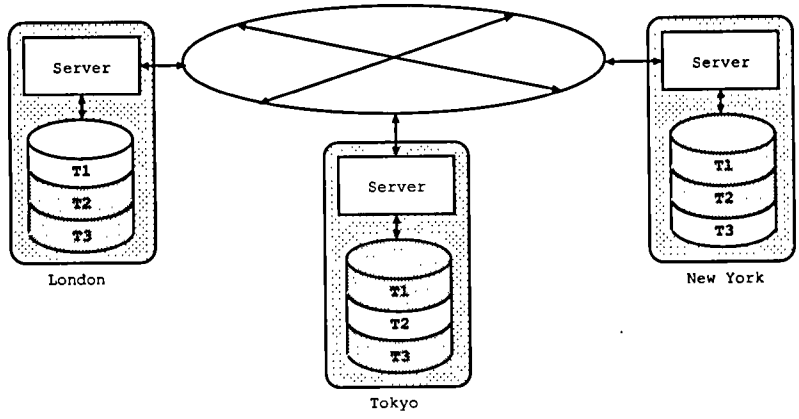


図 4　互いのレプリカをもつ 3 サイト

がよい．

このようなシステムを例に，レプリカを持つことの利点を考えてみる．

・信頼性の向上……ディスク破壊などの致命的障害からデータを守るための冗長性保持．たとえば，東京のデータが破壊された場合，ロンドンからそのデータを取り寄せることができる．

これは，とくに分散ではないシステムでも「ミラーリング」として知られるもので，DMBSの世界でも以前から多くの実装が存在する*．

・可用性向上……同一のデータを複数持つことで，あるサイトが故障しても，別のサイトのデータをアクセスできれば，システム全体の可用性は向上する．たとえば，ロンドンへのアクセスが何らかの事情でできなくとも，東京でそのデータを含め参照することが可能である．

・アクセス時間の短縮……頻繁に参照されるデータをローカルサイトで持つことにより，アクセス時間の短縮が見込まれる．この例では，もしロンドンやニューヨークのデータを頻繁にアクセスするのであれば，その度にWANを経由する必要がないことを意味する．

・スループットの向上……更新系と参照系を別々のサイトで持つことで，ロックを減らすことができ，並列度が向上し，結果としてスループットが向上する．この例では，東京でニューヨークのデータをアクセスすることと，ニューヨークでそのデータを更新することとが同時に行われても，なんら問題が生じないことを意味する．

2.3.2 レプリカを持つことの問題

すべてレプリカは，論理的にはマスタとともに，一つのオブジェクトである．したがって，マスタとレプリカは同一内容であることが保証されなければならないが，これにはいくつかの問題がある．

・表間の整合……図4においては，T1，T2，T3の各レコードは論理的に独立していることを仮定していた．ここでこれらが別の表であり，かつ表間の関連があり，これら複数の表を同時に更新するトランザクションが発生する場合を考える．

例えば，東京で，T1，T2を同時に更新しようとする．このとき，T1はロンドンにマスタが存在するので，こちらはロンドンに要求を出し，T2の更新のみを先に行ったとする．この状況では，少なくとも東京のT1とT2の間の整合性は，ロンドンのT1のマスタの更新が終り，東京のT1のレプリカの更新要求が処理されるまでの間は保たれないことになる．（実際にこのような更新でも整合性を保つのが，2PCのようなプロトコルであった．）

一つの解決法として，東京でT1の更新も許してしまうことが考えられる．この場合は，マスタからの更新要求を先取りした形になるが，別の問題が発生する可能性がある．そこで，このような表のマスタを一箇所に集中管理したのが図5である．

先の例では，T1，T2を更新するトランザクションは，東京でまず処理される．

---

* ただし実現技術は大きく異なる．ミラーリングは，マスタとミラー（レプリカ）のデータを一般に二つのディスクに分けて置き，システムがディスクのI/Oレベルで複製を行おうとする．これに対してレプリケーションでは，レプリカはネットワークを介した別マシンを想定するため，このような単純なディスクI/Oレベルの要求をレプリカに出すことはできない．（実際には，より論理的なデータ単位を，レプリカ側のサーバに渡すことになる．）

また，レプリケーションではネットワークやマシンなどに発生する様々な障害への対応を考慮した，高度なプロトコルが要求される．それは障害が起きたからといって，単純にレプリカ側を切り捨てるわけにはいかないからである．

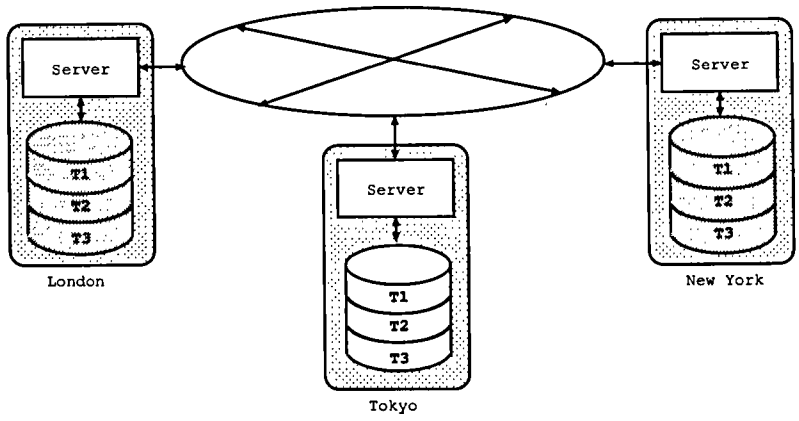


図 5　マスタを1サイトに集中

次にロンドン，ニューヨークにおけるレプリカを，更新トランザクションの発生により作成すればよい．

また，複数の表の間の整合性を保つためには，トリガ機能を用いる方法も考えられる．つまり，T2がT1の更新により自動的に更新されるようなトリガが作成でき，これを各サイトで保持できれば，東京でのT2更新のトランザクションがロンドン，ニューヨークに伝搬されるときにすべてのサイトでT1の更新がトリガによってなされ，各サイトにおけるT1とT2の間の整合性が保たれる．

・同一レコード変更の競合……レプリカでの更新を許す場合，同一レコードの同時更新という問題が起こる．たとえば，図4において，T1の更新をロンドンだけでなく，東京やニューヨークでも行えるようにすると，その更新の伝搬が互いに起こり，しかもそれが同一レコードに対して行われるものだとすれば，混乱を招くであろう．これはレプリケーションの形態の一つとして，次章で考察する．

2.4　レプリケーションによる問題解決

レプリケーションの長所については，すでに前節でまとめたが，ここでは特に分散システムの構築という観点から，2PCにおいて見られた問題と対比させて考察する．

2.4.1　ネットワークの問題

レプリケーション機能がネットワーク・トラフィックの解消に貢献する可能性として，

・レプリケーションを用いたシステムでは，原則として必要なデータはすべてローカルに持っているので，参照のみの場合にはネットワークに負担をかけない．

・マスタの更新に伴うレプリカへの更新要求は，リアルタイム性を要求されない限り，トラフィックの空いている時間帯を選べるなどの柔軟な対応をとることができる．

こうしたトラフィックの軽減は，システムの地域的な拡大に対しても適用の可能性が広がるだろう．

2.4.2　ローカルサイトの独立性

複数サイトの表の同時更新では，2PCのようなプロトコルを用いると，あるサイト

の障害により全体が止まってしまう問題がどうしても起こり得る．レプリケーションを用いれば，処理に必要な表をすべてローカルで保持することができるため，2 PC の場合に起こるこうした問題を防ぐことはできる．

ただし表を複数持つため，表間の整合性をとることが困難となる．これを解決するためには，2.3.2 項で見たような工夫が，各システムで不可欠となる．

2.4.3 サイトの同等性

レプリケーションではほとんどの場合，2.3.2 項で見た問題が発生しないよう，更新可能なマスタと読み出し専門のレプリカという区別があり，その意味ではサイトは同等とはいえないかも知れない．

しかしながら 2 PC と比べると，システム全体の整合性を保つために各サイトが組織の一員としてそれぞれの役割を担うということは少ない．マスタが責任を持つのはそのレプリカだけであり，そのこと以上にトランザクションに依存して別のサイトの表の更新に責任を持つ，ということはないのである．

そうしたサイトの独立性を生かしたシステム構築として，複数のサイトを階層的な木構造に組み合わせることがある．これにより，木の根にあたるマスタの更新を効率良く多くのサイトに伝搬させることができる．この場合多くのサイトはレプリカサイトでありながら，その下にあるサイトに対してマスタでもある．

2.4.4 分散システムの連続運用

2 PC は，更新の必要となるすべての分散されたサイトにコミットの確認をとらなくてはならない．したがって，その中のある一つのサイトへの経路のある一箇所にでも障害があれば，その更新トランザクションはロールバックするしかない．しかし，レプリケーションでは，そのようなサイトへの更新要求はあとまわしにして，うまくいくところだけを先に更新させることができる．更新の非同期性を積極的に利用することで，システム全体の可用性を高めることができるわけである．

またあるシステムにディスク障害など起こった場合にも，冗長的なレプリケーションの技術は有効に働くことがある．必要なデータは別サイトにも存在しているため，必要に応じてこれを参照するように変更もできるだろう．また復旧不能なサイトではそうしたデータを送ってもらうこともできる．

3. 実装と機能

2 PC が手続きの定義であるのに対し，レプリケーションはデータベースの複製（レプリカ）を持つ程度の意味しかない．ごく最近になりハードウェアメーカや DBMS ベンダが実装を始めた段階であり，標準化の動きも今のところはなく，実装，機能も様々である．

ここでは，そうしたレプリケーションの様々な実装，機能を概観し，また具体的に SYBASE，ORACLE，INFORMIX を取り上げてこうした面での違いを明らかにする．

3.1 レプリケーション機能の諸方式

3.1.1 複製データベースの形態

マスタ表とレプリカ表の対応を見て，次の 5 形態に分類する．

・1対1……マスタの表とレプリカの表とが1対1になっているもの．たとえば，基幹業務系（マスタ）から情報活用系（レプリカ）に更新データを供給するシステム．この場合，マスタは更新を行い，レプリカは参照専用で利用することになる．またミラーリングに対応する，システムの可用性を考慮した冗長なシステム構築も，この形態となる．

・多対1……論理的には一つの表だが，実際は各サイトにあるマスタのレプリカの集まりであるもの．更新処理は各サイトで行われる．たとえば，複数部門のサーバで業務処理を行い，これを本部に連絡するシステムの構築に使用する．

・部分集合……マスタの部分集合（一般にviewと言われるもの）をレプリカとする．これは，1対1の変形とも考えられるが，更新情報の形式によってはその実現は単純ではない．

・水平分割……一つの表をレコードのまとまりごとに，マスタとしての所有権を分割．図4を参照．この場合，マスタとレプリカが，同一表に混在することを許すことになるが，次の双方向とは異なり，同一レコードが複数サイトから更新されることはない．

・双方向……複数の表があり，共にマスタであり，レプリカであるもので，サーバにより更新できる行を制限しない．この形態を許すと，ある論理的には一つの行が同時に更新された場合，どちらを有効にするか（あるいはどちらも無効にするか）をあらかじめ決めておく必要がある．

3.1.2 システム形態

マスタサイトとレプリカサイトとを物理的に接続するには，次の形態が考えられる．

・スター型……一つのマスタの表に対し，レプリカの表が複数あるもの．たとえば基幹業務系（マスタ）情報を複数部門（レプリカ）に送るシステム．この場合，マスタの表のすべてをレプリカとする必要がないとすれば，そのサブセットを作る方法が用意されることが必要となる．

・階層型……マスタとレプリカの関係が入れ子になっている．すなわち，あるサーバから見たレプリカが，別のサーバから見てマスタになっている．

3.1.3 更新情報の形式

レプリカ側のサーバに送る更新情報には，次の二つの形式が考えられている．

・更新レコード情報……データ内容の他，更新時刻（タイムスタンプ），更新の種類（insert/update/delete）など．

・更新要求……SQL文のような形式で要求そのものを送る．

ここで，更新要求の場合，レプリカがマスタの「部分集合」であると，単純にマスタの更新要求をレプリカ側に送ることができないことに注意しておく．

たとえば，マスタのキー値が100以上と100未満によって，二つのレプリカR1，R2が存在していたとする．このとき，マスタ側で値が99のレコードを100に更新したとしよう．もし，更新要求を送るとすれば，R1には「キー値100のレコードの挿入」を，R2には「キー値99のレコードの削除」をそれぞれに送る必要がある．

3.1.4 配 信 方 法

レプリカに更新情報を送るタイミングにも，いくつか考えられる．

・即時……更新が発生するごとにレプリカに情報を送る．準リアルタイム．
・インタバル指定……10分おき，1時間おきなどの指定をする．
・日時指定……毎晩まとめて送るなどの処理が可能．
・件数指定……更新ログが20件たまったら送る，など．

3.2 SYBASE System 10

SYBASEは，System 10というバージョンでレプリケーションを実装した．

3.2.1 プロセス構成

この実装ではレプリケーションの実現に，図6のようないくつかのプロセスが起動される．とくに注目すべきことは，データベースサーバではなく，レプリケーションのための専用のサーバが用意されていることである．

・SQL Server……データベースサーバである．プライマリ側のデータベース（PDB）のサーバをPrimary Database Server（PDS），レプリカ側のデータベース（RDB）サーバをReplicate Database Server（RDS）と呼ぶ（図6）．

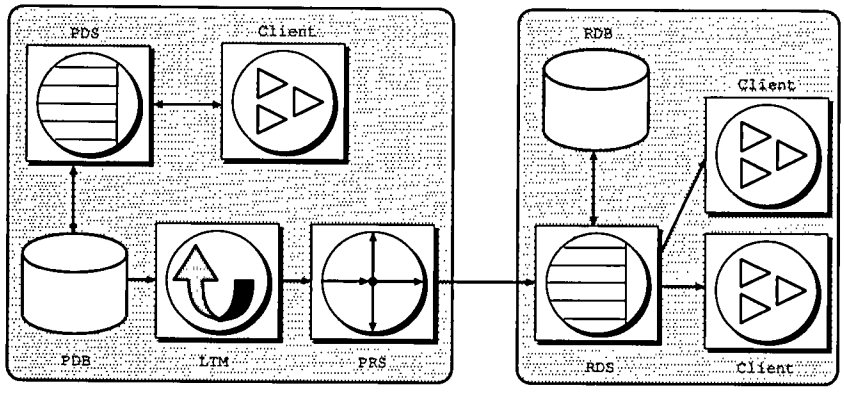


図6 LAN上のレプリケーション

・Replication Server（RS）……レプリケーションのための専門サーバであり，PDB側におかれたRS（PRS）は，PDBに更新がかかると，これをRDSに伝える．

図7では，RDBがPDBと同一LAN上になく，WANを介していることを示す．

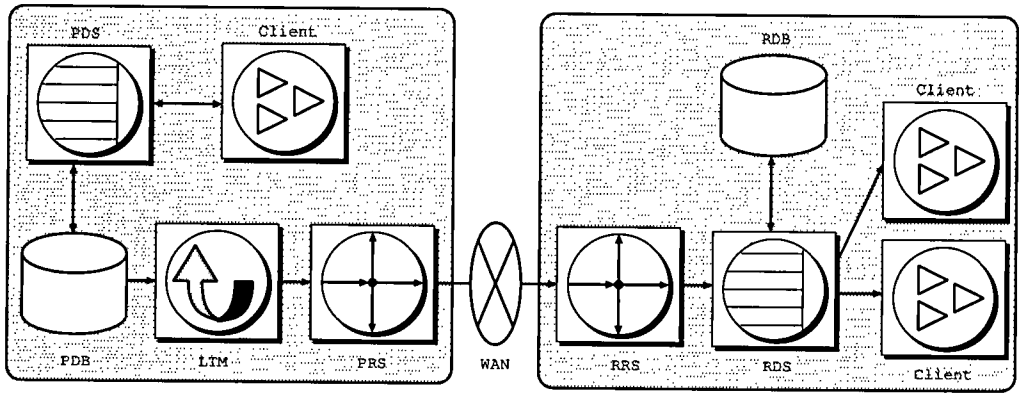


図7 WAN上のレプリケーション

このような場合，図6に比べるとネットワーク障害などが起こる可能性が高い．これを想定し，RDB側にもRS（RRS）を置くことが可用性の面から推奨される．この場合，RS同士が通信し合うことになる．

・Log Transfer Manager（LTM）……RDS側に置かれ，RDBの更新を監視する．更新されたテーブルにレプリケーションの設定があれば，更新トランザクションをPRSに送る．

3.2.2 レプリケーション情報のデータベース

RSSD（Replication Server System Database）というデータベースにより，レプリケーションの情報管理を行う．このデータベース自体はあるSQL Serverの管理下に置かれるが，レプリケーションの定義は既存のデータベースに対し，容易に設定，変更することができる．

管理情報は，Replication DefinitionおよびSubscriptionとして設定される．前者はどの表をマスタ表とするかの定義であり，後者はどのレプリカサイトへどのロー/カラムを送るかの定義である．

これらは，次のReplicaion Serverコマンドにより行われる．

・Create Replication Definition…… プライマリサイトのサーバ名，データベース名，テーブル名，レプリケーションを行うカラム名，データタイプを記述．

・Create Subscription……レプリケーション側のサーバ名，データベース名，テーブル名を記述．さらに，レプリケーションを行うカラムの範囲を，SQL文におけるWHERE句を用いて記述することも可能．

3.2.3 複製データベースの形態

1対1，多対1（4.3節の図10を参照），部分集合，水平分割（4.4節の図11を参照）を許す．

とくに部分集合については，レプリカ側にSubscriptionというものを後から定義することで，柔軟にシステム構築を行える．

双方向は支援されない．すなわち，レプリカデータへの更新は物理的に許されない．ただし，ユーザから見て論理的にこれを許す方法が考えられている．これは，非同期プロシージャコールと呼ばれるもので，レプリカ側に更新をかけるようなプロシージャを発行すると，これがプライマリ側に自動的に送られ，プライマリ側の更新が再度レプリカ側に配送される，というものである．

3.2.4 更新ログの形式

更新情報が転送される．トランザクション単位での転送ではなく，ある時点でマスタがロールバックされれば，レプリカ側にもロールバック要求が起きる．

3.2.5 配 信 方 法

原則として，即時配信である．

ただし，実際にはLTMプロセスがログを見て配信するために，リアルタイム性は保証されない．

また，レプリカサイトが障害を起こしていた場合には，その更新トランザクションをマスタサイトの該当するキューに入れ，障害が復旧した時点で溜っていたトランザクションを一括転送する仕組みを持つ．

3.3 ORACLE7

1) バージョン7.0

ORACLE 7 Ver.7.0では，「スナップショット」と呼ぶレプリケーション機能を実装している．この機能により，分散データベース内の他のノード上に，マスタ表のレプリカを作成できる．このレプリカは読み取り専用であり，マスタ表だけが更新可能である．

マスタ表に対する更新は，ORACLE 7 サーバにより，ユーザの設定した時間間隔で自動的にレプリカに反映されるか，あるいは手動で反映される．さらに，マスタ表に対するスナップショットログが作成でき，レプリカへの更新効率を上げることが可能となる．このように，Sysbaseで実装されているレプリケーションサーバの機能と比べると単純な作りとなってはいるものの，

- ローカルに問い合わせが可能であり，ネットワークへの負荷が軽減される．
- このため，パフォーマンスの向上がはかられる．
- ネットワークや他のノードの障害に対するローカルシステムの可用性が向上する．

などの，レプリケーションを利用することによる基本的な効果は期待できる．

2) バージョン7.1

ORACLE 7 Ver.7.1では，ORACLE 7 Ver.7.0での「スナップショット」の機能をさらに発展させた，「シンメトリック・レプリケーション」が実装されることとなる．

具体的な機能としては，

- 非同期RPC
- シンメトリック・レプリケーション

が提供される．

このシンメトリック・レプリケーションでは，レプリカ側の更新も可能となる(Update-Anywhere Data Replication)．

3.3.1 同期RPC

従来の同期RPCでの処理は，RPC処理をそのトランザクションと同時に処理する．このため2PCを利用して同じトランザクションのその他の更新処理との同期を取る必要がある．

非同期RPCでは，一旦要求をスケジュールしあるタイミングで目的とするサイトに送ることにより処理される．このため，リモートのサイトがなんらかの障害で利用可能な状態でない場合でも，RPC要求はシステムにより受け付けられる．

3.3.2 シンメトリック・レプリケーション

ORACLE 7 Ver.7.1で提供される，シンメトリック・レプリケーションの特徴としては，

- どのレプリカに対しても更新処理が可能である．
- 同じ論理データに対する同時更新が発生した場合，システムが自動的に整合性を取る仕組みを提供する．
- トランザクションの整合性を保証する．

ORACLE 7 Ver.7.0 をはじめ，他の RDBMS で提供されているレプリケーション機能とは，レプリカに対する更新を許しているという点で大きく異る．これにより，一層複雑さが増している業務処理へのレプリケーション機能の適用範囲が大幅に拡大する．

その反面，特徴でも挙げているが，状況によっては複数のサイトで同時に同じデータに対する更新処理をしてしまう可能性がある．ORACLE 7 Ver.7.1 で提供されるシンメトリック・レプリケーション機能では，同時更新を検知し，自動的に解消するための機能が提供される．同時更新解消ルーチンはシステムで提供する，最新時間更新，追加更新，積算更新，および最大/最小更新の他に，利用者が業務に応じて独自のルーチンを作成することも可能となっている．

このように，従来の参照のみ可能なレプリケーションと比べ，より柔軟に分散システムを構築することが可能となり，システムの可用性がさらに向上する．

3.4 INFORMIX version 6.0

INFORMIX version 6.0 では，"OnLine Dynamic Server" という新たなサーバを提供し，この新しい機能の一つとしてレプリケーション機能を持たせる．

3.4.1 複製データベースの形態

1 対 1 である．

あるサーバの情報全てがマスタとなり，これと物理的に同じレプリカが他のサイトに作成される*．その主な目的は，ディスク故障やマシンダウンなどの，全てのサーバのトラブルからシステム全体を保護することである．レプリカは，読み込み専用として用いることができる．

3.4.2 更新情報の形式

更新レコード情報(論理ログ)である．論理ログは，INFORMIX システムにおいて，ロールバックおよびロールフォワードを行うのに必要十分な情報が蓄えられているものだが，このためのバッファ（論理ログバッファ）と同内容のデータ・レプリケーション・バッファ（DR バッファ）が作られ，これがレプリカ側に送られる．レプリカ側では，これをリカバリバッファにおき，リカバリのスレッドがこれをレプリカ・データベースに反映する(図 8)．

3.4.3 配 信 方 法

同期モードと非同期モードがある．論理ログバッファが DR バッファにコピーされると，次にこれがレプリカ側に送られるが，同期モードではこのタイミングでマスタ側の論理ログバッファをフラッシュしてしまう．しかし，非同期モードでは，レプリカ側の Acknowledge を受けてから，フラッシュを行う．

これには一長一短ある．同期モードでは，マスタの更新がネットワークの影響を受けない代わり，レプリカとの整合がとれない可能性がある．非同期モードは逆にネットワークのトラフィックなどが影響し，論理ログバッファのフラッシュが遅れること

---

* 文献[14]によれば，INFORMIX のレプリケーションは次のフェーズで，図 4 程度の機能を持たせることを発表している．さらにここで，パーティション・レベル・アプリケーション，SQL "select" レベル・レプリケーション，という二つの記述がある．後者が表より細かいレベルのレプリカを構築することができるものかは，現在のところ不明である．

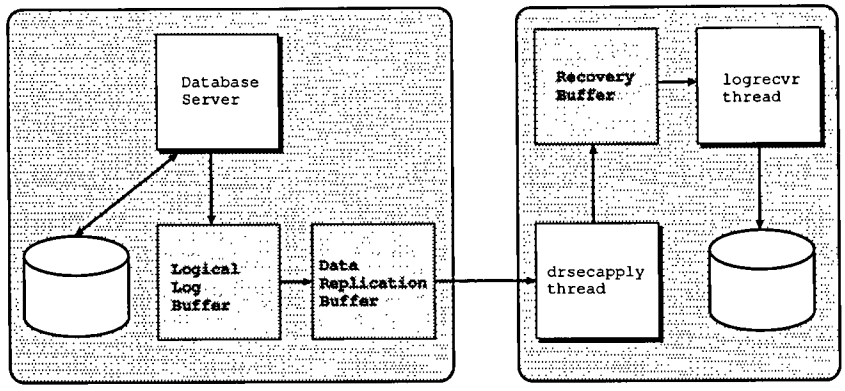


図 8 INFORMIX version 6.0 のレプリケーション

があるが，マスタレプリカ間の整合性は保証される．

4. レプリケーションを用いたシステムの構築

本章ではいくつかのシステムを例に，レプリケーションの取り入れ方を考える．

4.1 デュプレックスの実現

ディスク障害などのトラブルに対処するため，プライマリを二重化する方法がある．最も一般的で簡単な実現方法は，ミラーリングであるが，マシン自体のトラブルにも対処しようとすれば，やはりレプリケーションの機能が必要となる．

図 9 では，白いディスクがマスタを表し，もう一方の黒いディスクがレプリケーションを表している．すべての更新処理は，マスタに対してなされることになるだろう．INFORMIX のレプリケーションは，これを目的としたものである．

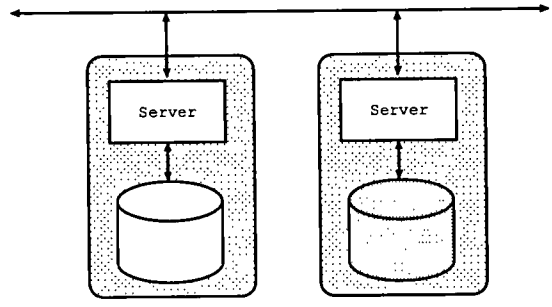


図 9 レプリケーションを用いたデュプレックスシステム

4.2 更新系と参照系の共存

更新系として，一度に少数レコードを更新する OLTP アプリケーションを，参照系として，全レコードの情報を見るような決定支援アプリケーションを考える．

参照系が，インタラクティブな時間のかかる処理を行っているとすると，更新系のトランザクションとしばしば重なり合うことになるだろう．

このとき問題となるのは，ロックの競合である．もし参照系がデータの整合を要求するために，なんらかのロックを必要とするならば，これが動く間は更新系が止まっ

てしまう．

SYBASE では，更新系にマスタを，参照系にレプリカを置くことで，この問題を回避できる．

ORACLE では，各レコードデータのバージョン管理を行い，参照系がロックをすることなく整合性を保証する仕組みを持つため，この用途にレプリケーションを行う利点はない．

4.3 複数サイトでの同一表の更新

東京の本社には全社レベルの表があり，名古屋，大阪，福岡の各支社に関するデータは，それぞれの支社で更新する，という場合を考える．これをレプリケーションを用いて実現するとすれば，図 10 のように，各支社にマスタをおき，本社に各支店のレプリカをまとめればよい．

これは，データベースの形態でいうと，多対 1 になる．SYBASE では各サイトのデータを，「プライマリフラグメント」といい，これらのレプリカを統合して一つの表にする機能がある．

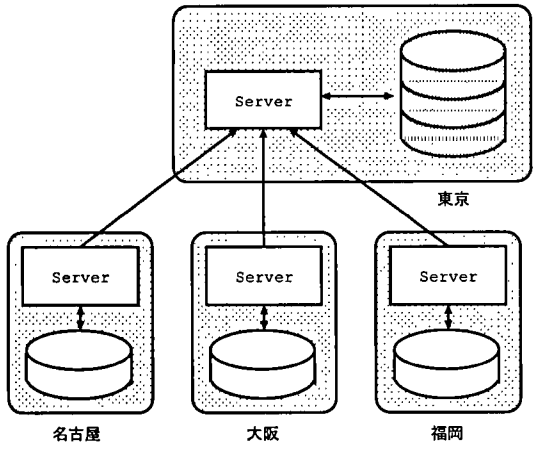


図 10　多対 1 の表の利用

4.4 複数サイトで互いの表の共有

東京，ニューヨーク，ロンドンの各社が，互いの表の参照を必要とする場合を考える．これをレプリケーションを用いて実現するとすれば，図 11 のように，各社にマスタ（白の部分）とレプリカ（黒の部分）を混在した表を持たせればよい．

4.5 双方向を許すマスタ表

東京，ニューヨーク，ロンドンそれぞれに同じ表が存在し，それぞれ更新処理を必要とする場合を考える．ここまで説明してきたシステムの構築例とは，同じ表がマスタとレプリカの両方の機能を持つ，と言う点で異なる．これは，ORACLE 7 により提供される，シンメトリック・レプリケーションを使うことで構築できる(図 12)．

しかし，あくまでもレプリケーション技術の中での機能であり，一時的にデータの不整合が発生することを防ぐことはできない．さらに，論理的に同一のデータを別々に更新してしまった場合には，どちらのデータを有効にするか決定する必要がある．

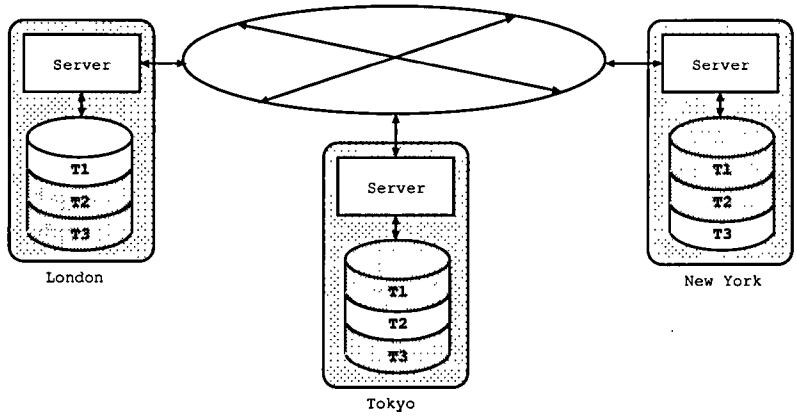


図 11　水平分割された表の利用

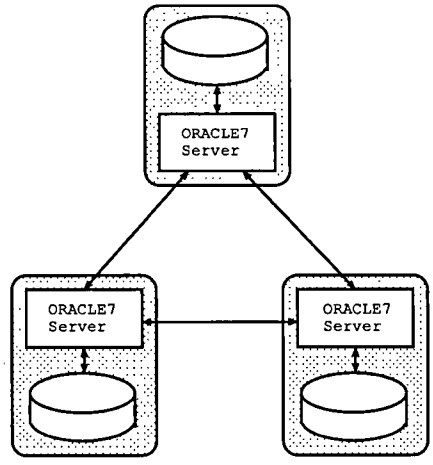


図 12　双方向のマスタ表とレプリカ

このように，新たな問題を含んではいるものの，レプリケーション機能が利用可能な範囲が大きく広がる技術として注目していきたい．

5. 結　　論

企業レベルのネットワーク環境と，それにともなう必然的なオープン化は，エンドユーザ・コンピューティングを促進させるとともに，その実現基盤としての分散データベースに対しても，様々なレベルでの構築可能性の追求を余儀なくした．このときユーザから見て最も期待されている点の一つは，物理的なシステムがいかに複雑であっても単純な操作によりデータをアクセスできる，という透過性の実現だろう．

ネットワークが十分高速で信頼性も高いと仮定すれば，レプリケーションも 2 PC も，この透過性の実現の鍵となる技術であるといえる．もちろん透過性の意味も問題

にするべきであろう．実装によっても異なるが，アプリケーションを構築する立場から見ると，レプリケーションではほぼ透過であるが，2 PC の場合は透過にはしにくい．

逆にアプリケーションを使用するユーザは，レプリケーションではマスタとレプリカとの違いを意識したアクセスが必要な場合もあるが，2 PC の場合は透過的なアクセスを保証する，という違いはある．

とは言うものの，現実にネットワークを介して複数のサイトのデータを更新する分散トランザクション処理を考えた場合，基盤技術としてのネットワーク環境では，速度や信頼性，また普及度といった点でまだ実用レベルとは言えない部分がある．

このような基盤環境を考えたときレプリケーション機能は 2 PC に比較して，実用的，実際的な機能であることが，本稿の考察により明らかになった．さらに，UNIX 上のいくつかの DBMS におけるレプリケーション機能の調査および比較は，実際のシステム構築の際に有効な資料となると考える．ただし，入手できた資料の内容や質の問題もあり，とくに実装面での調査が手薄になったことは残念である．

今後は，この点でのより踏みいった調査と，また実際にシステムを構築することで，計量的な評価を行いたい．

参考文献

[1] C.J.Date An Introduction to Database Systems Volume I.
[2] 前川守・所真理雄・清水謙多郎編「分散オペレーティングシステム」共立出版.
[3] BUSINESS NEEDS GUIDE ALL OR NOTHING 2 PC USE SOFTWARE MAGAZINE 1993.8.
[4] SYBASE Replication Server Administration Student Guide ver.1.0.
[5] SYBASE Replication Server Overview.
[6] SYBASE Replication Server Administration Guide.
[7] SYBASE Replication Server Design Guide.
[8] Oracle 7 Symmetric Replication-Asynchronous Distributed Technology 1994.6.
[9] 「ORACLE 7 Server 管理者ガイド」
[10] 「ORACLE 7 Server 概要」
[11] INFORMIX Case Study : Using Data Replication in Large ProductionEnvironment 1994.7.12.
[12] INFORMIX Data Replication Strategies : Triggers, Stored Procedures, and Two-Phase Commit Versus DSA's Data Replication Features 1994.7.13.
[13] INFORMIX Data Replication Strategies for the Distributed Enterprise 1994.7.13.
[14] INFORMIX Dynamic Scalable Architecture.

執筆者紹介　山　田　和　司 (Kazushi Yamada)

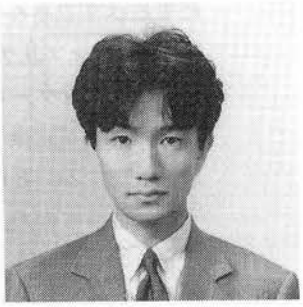
1980 年東京理科大学理工学部経営工学科卒業．同年日本ユニシス(株)入社．XE 550 システム上の ingres の日本語化などを経て，現在 UNIX 上の RDBMS である，Oracle のリリースおよび保守に従事．並列・分散処理，およびマルチメディアなどに興味を持つ．日本ソフトウェア科学会会員．オープンソフトウェア二部データベース課所属．

E-Mail : Kazu. Yamada @ unisys, co. jp.

今 泉 正 雄 (Masao Imaizumi)

1988年都立大学理学部数学科卒業. 1990年電気通信大学大学院情報工学博士前期課程修了. 同年日本ユニシス(株)入社. U 6000上のC++のポーティング, その他の言語処理系の保守などを経て, 現在UNIX上のRDBMSである, SYBASEのリリースおよび保守に従事. 計算機言語, オブジェクト指向, 並列・分散処理などに興味を持つ. オープンソフトウェア二部データベース課所属.

E-Mail: Masao. Imaizumi @ unisys, co. jp.



エンドユーザ・コンピューティング環境におけるリポジトリ

The Role of a Repository in the EUC Environment

阪口喜好

要　約　90年代のエンドユーザ・コンピューティングの中心は，ワークステーションを用いたソフトウェアツール環境である．これまでソフトウェアツール環境は，個々のツールの持つGUIや機能に関心が集まってきたが，今後はソフトウェアツールの統合による情報活用が重視されるようになるであろう．いかに個別のツールが優秀であっても，単なるツールの寄せ集めでは企業情報システムとはなりえない．データベースとリポジトリに基づいて統合されたソフトウェアツール環境が必要になる．このような環境において，リポジトリの果たす役割は，きわめて重要である．

本稿では，リポジトリの概要と実装における当社の考え方を紹介する．

Abstract　The requisite of end user computing in the 90s is an integrated software tool environment in which PCs and workstations are used. Although the software tool environment has so far seen user interest focused on the GUIs and features that individual tools provide, greater importance will be attached to the more effective use of information through the integration of software tools. No matter how powerful they are, a mere collection of them never leads to productive enterprise information systems. Required is the environment where software items have been integrated according to database and repository requirements. Under such circumstances, the role of a repository is getting very crucial.

This paper discusses what the repository is all about and Nihon Unisys' concepts of its implementation.

1. はじめに

システム開発を取り巻く環境は，「ソフトウェア開発の時代」から「ソフトウェア・コモディティの時代」に変化してきた．すなわち，情報システム部門を中心とする「オーダーメイド」のシステム開発からエンドユーザ主導の「イージーオーダ」あるいは「レディメード」のソフトウェア部品の集積によるシステム作りへと変化しつつある．膨大なバックログを抱えた情報システム部門によるシステム開発の行き詰まりは，エンドユーザ・コンピューティングによるバックログ解消への期待を大きく膨らませている．このような環境変化は，組織のシステム開発能力の重視から組織の情報活用力の重視へとその力点が変わっている．しかし，エンドユーザ・コンピューティングが広まるにつれて，これまで情報システム部門が担ってきた情報の一元管理の枠がゆるみ，情報の不整合や重複データの散在といった弊害が出始めている．企業活動に係わる情報が，どこで，だれによって，どのように蓄積されているかを管理し企業情報の一貫性維持を図らないかぎり，エンドユーザ・コンピューティングは，企業の情報資産に突きつけられた諸刃の剣となるであろう．

そのようにならないために，リポジトリの果たす役割はきわめて重要である．本稿では，リポジトリの概要と実装における当社の考え方を紹介する．

2. リポジトリとは

リポジトリとは，企業の情報資源に関するさまざまな情報を，目的に適合する記法で表現・格納し，その変更を制御し，利用・維持することを助ける汎用の情報システムである．あるいは，リポジトリとは，企業情報資源管理に関する情報を管理するための特化したデータベース・アプリケーションであると言うことができる．その構成要素は，次のとおりである[1]．

1) 情報モデル……情報を表現するためのスキーマ体系のこと．情報モデルは，実装レベルとは独立した，リポジトリに蓄積された情報を理解するための共通の枠組みである．実装レベルでは，データベースの分野で検討されてきた各種のデータモデルが使われる．たとえば，実体と実体間の静的な関係を表現する実体関連モデル，情報の表現能力の限界を指摘されている関係モデル，クラス階層や継承により，対象世界のさまざまな抽象を表現できるオブジェクト指向モデルなどはその一例である．
2) 情報ベース……上記モデルに従って格納された情報そのもの．情報は，レコード，表，オブジェクトなどを使って表現される．
3) 情報ベース管理システム……情報ベースのデータ生成，アクセス制御を支援するソフトウェア群．
4) リポジトリ・サービス……版管理や構成管理などリポジトリに固有の制御機構．

図1は，リポジトリの構成要素を示している．

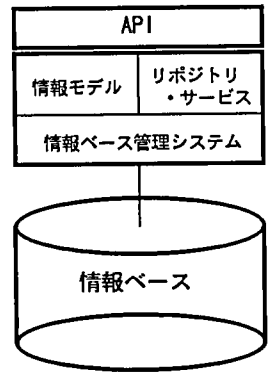


図 1　リポジトリの構成要素

一方，リポジトリが管理対象とすべき情報は，情報システム・ライフ・サイクルにおけるすべての活動で生成または使用される情報であり，多岐にわたる．たとえば，次のようなものが考えられる．

- 企業モデル
- 情報システム・プランニング
- 概念的なデータとプロセスのモデル
- 論理データベースとプログラム設計
- データベース・スキーマとソース・コードの情報

・単体テストおよび機能テスト情報と目標実現のための構成要素
・アプリケーション複合情報
・アプリケーション操作と実行の規則

これらの情報を，定義情報，管理情報そして運用情報に分類したものを，表1に示す．

表1　リポジトリに格納される情報の分類

情報の種類	説　　明
定義情報	データ，企業モデル，ビジネス・プロセスおよび規則，アプリケーション要件，データ要件とデータ・モデル，アプリケーション設計とアプリケーション・モジュール，データベース・スキーマ，ユーザ・インタフェース情報，製品情報，システム・オブジェクトとそれらの間の関連など
管理情報	プロジェクト管理情報，版管理，構成管理およびセキュリティ対策，ソフトウェア開発ライフ・サイクルの各フェーズを管理するのに必要な情報など
運用情報	システム構成，バックアップ情報，スケジュールと他の同様な情報など

表2は，ソフトウェア開発世界を対象として，リポジトリが管理すべき情報や支援すべきソフトウェア・エンジニアリング活動を九つの分類で表現したものであり，参考文献[1]に基づいている．行方向では，「支援対象活動」をプロダクト管理支援，プロセス管理支援，コミュニケーション支援の三つに分類している．他方，列方向では，「開発の規模」を個人レベル，チームレベル，プロジェクトレベルの三つに分類している．

表2　ソフトウェア・リポジトリの管理対象情報

支援対象活動	個人レベル	チームレベル	プロジェクトレベル
コミュニケーション支援	・設計ノウハウの解明と活用 ・業務/技術知識の獲得と伝授	・作業分担と調整 ・収束型討論の支援	・組織階層における効果的な作業指示法と結果の確認方法 ・進捗状況の確認
プロセス管理支援	・ツールの操作法 ・ツール操作手順	・作業の連続性の保持 ・作業間のインタフェースの整合性の保持	・プロジェクト計画立案支援と危機管理 ・プロセスデータの収集とプロセスの改善
プロダクト管理支援	・ソース，文書類の個人的管理	・追跡可能性と制約の表現 ・版管理 ・部品の再利用	・ベースライン成分の管理 ・プロジェクトデータの収集

過去において，情報資源に関する情報の格納と管理を行う方式として，システム・カタログ，ディレクトリ，データ辞書などが提供されてきた．これらとリポジトリの違いを，次に述べる．

・システム・カタログとは，データベース管理システムによって使用され提供されるスキーマ情報表のユーザビューである．
・ディレクトリは，データベース管理システムやその他ソフトウェア群が参照するスキーマ情報であり，位置情報やアクセス情報も格納される．
・データ辞書は，データベース管理におけるメタ・データの蓄積，維持，管理から出発し，データの物理的，実装レベルの記述を主として蓄積してきた．情報

システムの成熟につれて，このような狭義のデータ辞書から，データの記述とともにプロセスおよび管理情報の構文と静的セマンティクスを格納する広義のデータ辞書へと進化しつつある．

リポジトリは，このような広義のデータ辞書の機能をも包含し，さらに動的セマンティクスを付加したものである．表3は，データ辞書とリポジトリの主な違いを示している．

表3　データ辞書とリポジトリの違い

データ辞書	リポジトリ
一般に，単一ベンダによって提供される少数のツールによって使用されるため，拡張性に乏しい．	多数のベンダによって開発されたツールを統合するため，本来的に拡張可能である．
単一ベンダの機能要件に操作が限定される．	他のベンダのリポジトリとの相互運用が可能である．
データ辞書はアプリケーション開発中に使用され，要求に関する開発データのみを格納したり問い合わせるために使用される．	アプリケーション開発中と実行時には，能動的な機能を持ち，情報システム環境内の出来事に応答することができる．
情報の静的セマンティクス，データの記述および位置情報を持つ．	情報の動的セマンティクスと静的セマンティクス，データの記述および位置情報も持つことができる．
ソフトウェア開発環境でのみ使用できるものと見なされる．	情報システムの開発，実行および運用時に環境の中核に位置する基盤と見なされる．
データ項目の固定された集まりを格納し管理するか，または拡張可能性を制限する．	自己自身による定義が可能．格納され管理される新たなオブジェクト型の追加ができ，オブジェクトへ操作することもできる．

リポジトリの効用として期待できるものを，エンドユーザ・コンピューティングという観点からまとめると，次のようになる．

- あらゆるアプリケーション，ツールに，システム情報を提供する．
- 複数ユーザが利用可能な，統合ソフトウェア・ツール環境を可能とする．
- ユーザのコミュニケーションと情報の共有を促進する．
- 企業内で重複するデータを整理し排除する．
- システムの完全性を増進する．
- システムの保守を単純化する．
- マルチベンダのツールの組み合わせを円滑にする．
- 情報をシステム・ライフ・サイクルを通して再利用可能とする．
- システムの移行，変換を単純化する．

リポジトリは，全社的情報の一元的な源泉となり，次のような情報統合の主役となるものと期待できる．

- ディクショナリの統合
- ソフトウェア・ツールの統合
- ソフトウェア・システムの統合
- ソフトウェア・ライフ・サイクル工程の統合

3. リポジトリの要件

エンドユーザ・コンピューティングの進展，クライアント/サーバ・システムの開発が本格化する流れの中で必要とされるリポジトリの要件とは，どのようなものであろ

うか．

これまで実用化されたデータ辞書やリポジトリでは，実体関連モデルに基づいた情報モデルを関係データベースにマッピングし，関係データベース管理システムをディクショナリ・エンジンとして実装されたものが多かった．このような基盤技術に立脚したリポジトリに対して，次のような問題点が指摘されている．

1) 情報モデルを構築する際の核となる実体関連モデルの表現能力が不十分である．実体関連モデルでは，静的セマンティクスは表現することができるが，動的セマンティクスの表現はできない．

2) エンジンとしての関係データベースの能力が不足している．すなわち，実体関連モデルを関係モデルにマッピングした場合，リポジトリに蓄積される豊富な実体と関連の型を十分に支援できるのか，また，リポジトリのパフォーマンス要件を満たしうるのかという疑問がある．

さらに，リポジトリ全般の問題として重要な問題点を指摘しておく．

3) 実装形式を統一するための標準が定まっていない．リポジトリについての標準が重要なのは，ソフトウェア・ツールの統合，異なるベンダのツールの組み合わせ，リポジトリ間の情報交換に必須だからである．標準が定まった時点で，いかに早く追従できるかが課題となる．

このような問題点や課題を解決するために，新しい基盤技術としてオブジェクト指向技術を取り込むことが求められている．

リポジトリの要件をまとめると，次のようになる．

- オープン・アーキテクチャ
- 拡張可能な情報モデル
- 段階的なツール統合（疎な統合から完全統合まで）
- マルチ・プラットフォーム対応
- オブジェクト指向
- マルチ・パラダイム支援
- 標準準拠の共通環境

4. ユニシス・リポジトリ（UREP）

米国ユニシス社は，1990年に発表したUnisys Architecture (UA) の中で，Unisys Repository (UREP) 像を明確に示した．本章ではUREPを例として，3章で述べた要件を満たすリポジトリの実装について解説する．

1) 共通リポジトリ環境……リポジトリのユーザ（ソフトウェア・ツール，リポジトリ管理者，エンドユーザなど）が，異種プラットフォームからアクセスするインタフェースとサービスのセットが提供され，リポジトリが存在するすべてのプラットフォーム上で同じリポジトリ・インタフェースと機能が使用できる．

2) 相互運用性……異なる種類のリポジトリを有する複数プラットフォーム環境において，リポジトリの共存と相互運用性を可能にする．リポジトリは任意のプラットフォームに導入することができ，リポジトリ・データの分散化が可能であるため，インポート/エクスポートサービスやその他のインタフェースを介して任意

のプラットフォームからそれらのデータをアクセスすることや，プラットフォーム間でデータを移動することができる．

3) 相互運用性とリポジトリ間通信……複数プラットフォーム上に分散されたリポジトリは，ブリッジ，ゲートウェイあるいはインポート/エクスポート・サービスを使用して，リポジトリ間の通信を実現する．ブリッジは，通常，既存アプリケーション・プログラムで作られているリポジトリをロードするのに用いられる．ゲートウェイは，プロトコルの対応付けまたは変換機構である．たとえば，ツール統合システム（ATIS）や可搬共通ツール環境（PCTE）が提供するアプリケーション・プログラム・インタフェース（API）をUREPが提供するサービス・インタフェースに変換するものが該当する．

4) 拡張可能性……リポジトリによって格納され管理されるデータの対象範囲は，企業や情報システムの進展にともなって拡張される．それゆえに，リポジトリによって格納され管理されている情報は，拡張可能でなければならない．リポジトリの情報モデルがリポジトリ中で定義されている既存のオブジェクトに基づいて新しいオブジェクトを追加して拡張できる場合，そのリポジトリは「拡張可能」であると呼ばれる．既存のオブジェクト型に基づいて新しいオブジェクト型を定義することにより，リポジトリはツールの統合を実現する．新しいオブジェクト型は，既存のオブジェクト型の属性と操作を継承することができ，また，新しい操作を追加することもできる．このように，既存オブジェクトを拡張する機能は，定義と操作の両方の再利用と共用を促進する．

5) 特定の技術に偏重しないサービスの提供……リポジトリは，情報の共用を促進し，オブジェクト指向プログラム言語のみならず，4GLおよび従来の3GLといったアプリケーション開発言語，DBMS技術（オブジェクト，関係，ISAM，ネットワーク）に適用できるサービスを提供する．

6) 複数レベルのツール統合……ソフトウェア・ツールによる情報の共用と再利用を押し進めるためには，ツール統合に関して複数のレベルを考慮する必要がある．これにより，ツールを提供するサードベンダは，自社のツールをリポジトリに統合することができる．ツール統合により，以下のことが可能となる．

- 異なったツール同士が，与えられた環境の枠内で効果的に会話する．
- 環境の構成要素間を関連付ける．
- 情報システム環境の構成要素（NIST/ECMA参照モデル）間のシームレスな相互作用で生まれる相互運用性，移植性，生産性，スケーラビリティなどが実現する．

7) 標準の準拠……受け入れられた標準には準拠する．標準の準拠は，分散コンピューティング環境と異種システム間の相互運用性を容易にする上で重要である．リポジトリに関係する標準規格は，5章を参照のこと．

8) リポジトリ情報モデル……情報モデルは，リポジトリに格納されているデータとリポジトリによって用意されているサービスを定義しており，リポジトリに蓄積されている情報とリポジトリが提供するサービスを理解するための共通の枠組みである．リポジトリは格納された記述をオブジェクトとして取り扱うオブジェ

クト・モデルであるので，オブジェクト間の関係（継承）とオブジェクトに適用できる操作に関する情報も，リポジトリに登録することができる．図2は，情報モデルの一部分を示しており，情報モデルは，リポジトリ・オブジェクトモデル，レポジトリ・サービス，そしてツールモデルの3層から構成されている．ツールを統合する場合は，リポジトリ・サービスに基づいて，ツールモデルを拡張すればよいことがわかる．リポジトリのAPIは，C++のオペレータあるいはCのファンクションを介してインタフェースがとられる．

9) リポジトリ・サービス……UREPが提供するリポジトリ・サービスは情報モデルに定義されている．表4は，提供されるリポジトリ・サービスを要約したものである．

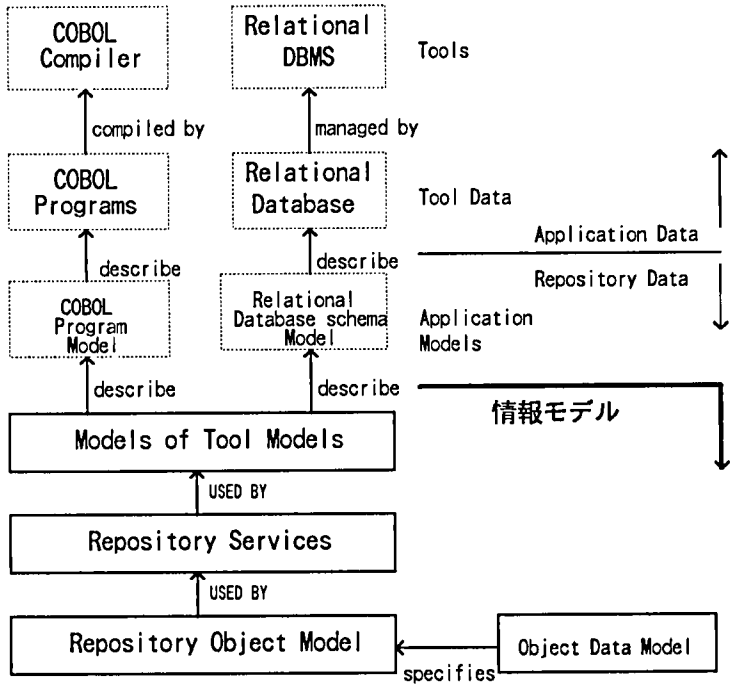


図2　リポジトリ情報モデル

5. リポジトリの標準規格

米国ユニシスは，リポジトリのサービス・インタフェース，インポート/エクスポートの形式，そして情報モデルに関する規格を検討している種々の標準規格委員会に協力し，リポジトリ標準規格の制定を積極的に支援している．.

1) 情報資源辞書システム（IRDS）……米国規格協会（ANSI）と国際標準化機構(ISO)は，どちらもIRDSを定義する標準規格を作成している．ANSIとISOは，IRDSサービス・インタフェース標準規格のすり合わせを行っている．

ANSIのIRDS標準規格のうち重要なものは，次の二つである．

・ANSI X 3.138 情報資源辞書システム標準規格

表4　リポジトリ・サービスの要約

サービス	サービスの要約	重要性
名前管理	ユーザが理解するオブジェクトの名前と，リポジトリが格納している永続オブジェクトの間の関連を保持する．	命名は対象を識別することである．ユーザはオブジェクトを名前で管理することを望む．
セキュリティ	不法なアクセスからリポジトリ情報を保護する．	リポジトリには大量の企業機密に触れる情報が含まれており，この情報は保護されねばならない．
版管理	リポジトリ情報の変更を管理し，変更の履歴を保持して，逐次開発および並行開発の両方を支援する．	リポジトリに格納されている情報は常に変化している．変化を記録し管理することが非常に重要である．
構成管理	複合的なシステムを表現する複合オブジェクトを定義し管理する．構成管理サービスは，個々の部品ではなく，複合オブジェクトをそのまま操作できるようにする．	一部の永続オブジェクトは，複数の下位部品から構成される複合的なシステムを表現している．
ワークフロー	ビジネス・プロセスとこれらのプロセスの各段階と，そこで発生する事象に関する情報を保持し管理する．	多くの企業プロセスは，特定の事象の発生に従って複数の段階を経て進展する．
ツール統合	ツールの登録，統合および起動を行う．	ツール統合によってもたらされるソフトウェア・ツール間の円滑な情報フローがアプリケーション開発の流れを改善する．
インポート/エクスポート	リポジトリ間の双方向の情報転送を行うことができる．	リポジトリ情報を複数のサイトからアクセスし，サイト間で移動することができる．
管　理	他のリポジトリ・サービスのためのすべての管理インタフェースを提供する．	リポジトリを導入した企業は，ツール統合，ビジネス規約，リポジトリ情報へのアクセス，リポジトリ環境管理などに対して指針を立てる必要がある．

・ANSI X 3 H 4 情報資源辞書システム要件文書

ANSI X 3 H 4 分科委員会は，次世代の IRDS 標準規格を検討中であり，オブジェクト指向リポジトリのサービス・インタフェース，基本的なリポジトリ・サービス，コンテンツ・モジュールのための標準規格機能などを検討している．ISO は，次の標準規格を作成している．

・ISO 情報資源辞書システム・フレームワーク（IS 10027）と ISO 情報資源辞書システム参照モデル

・ISO 情報資源辞書システム・サービス・インタフェース（IS 10728）

2)　オブジェクト・マネジメント・グループ（OMG）……オブジェクト指向技術を採用する統合ソフトウェア・システムを開発し，使用するという共通の目標を持つソフトウェア・ベンダのコンソーシアムである．OMG オブジェクト・モデルはリポジトリ情報モデルの基盤である．

3)　ECMA/NIST の参照モデル……欧州電子工業会（ECMA）は，ECMA 参照モデルを技術報告書 ECMA TR/55 として出版した．ECMA/NIST の参照モデルは，CASE 環境を支援するために共通に要求されているオペレーティング・システムとデータベース管理システムによって提供されるもので，サービスのセット，

主要統合構成要素，構成体のサポートを表わしている．米国連邦標準技術局(NIST)は，ECMA 参照モデルを採用し拡張して NIST 特別出版物 500-201 として出版した．

4) ECMA 可搬共通ツール環境(PCTE)……ECMA PCTE 標準規格には，PCTE ツールによって使用される情報のリポジトリの役目を果たすオブジェクト管理システム (OMS) が含まれている．また，PCTE 中のリポジトリのための情報モデルを構築し管理するためのスキーマ管理サービスが含まれている．

5) EIA CASE データ交換形式 (CDIF) ……米国電子工業会 (EIA) は，リポジトリ間，ツール間，ツールとリポジトリ間で情報を通信するため，インポート/エクスポート・ファイル形式用の CDIF を定義している．EIA/CDIF 委員会は，標準規格化のためのコンテンツ・モジュールも定義している．

6. おわりに

Windows NT* のリリースにともない，より小さな組織体の中でクライアント/サーバ・システムを導入できるようになり，開発に弾みがついている．それはまた，本稿の冒頭で述べた危惧が現実化する日が近づいていることでもある．リポジトリを核とした，EUC 環境の構築とシステム・ライフ・サイクル管理体制の確立が急がれる．

UA で規定された Unisys Repository に基づいたサーバ上で稼働する UREP/U (UNIX** 版) が，現在開発中であり，リリースが待たれる．

---

* Windows NT は米国 Microsoft 社の商標である．

** UNIX は，X/Open カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標である．

参考文献

[1] 落水浩一郎，ソフトウェア・リポジトリ，情報処理，Vol.35 No.9, 1994.

[2] Carma McClure,"THE THREE Rs OF SOFTWARE AUTOMATION. Re-engineering, Repository, Reusability", Prentice-Hall.

[3] Unisys Architecture Technical Reference Volume 2 Information Management Service (IMS).

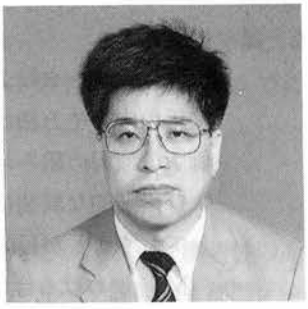
[4] 堀内一，情報資源管理とリポジトリ，第3回 CASE ジャパン '94 セミナー要録．

[5] ISO/IEC 10027 Information technology-Information Resource Dictionary System (IRDS) framework.

[6] Lois Wakeman&Jonathan Jowett,"PCTE The standard for open repository", Prentice Hall.

執筆者紹介　阪　口　喜　好(Kiyoshi Sakaguchi)

1970 年京都大学工学部機械工学科卒業．1973 年日本ユニシス(株)入社．応用ソフトウェア部にて DSS 1100 の開発に従事．以来データベース関連プロダクトの開発，保守および適用支援に従事．1987 年 4 月より 1990 年 9 月まで米国ユニシスにて XTC-TIP/UDS の開発に従事．現在，システムプロダクト本部サーバソフトウェア 1 部部長．



人事システム再構築における UNIshuttle の適用事例

An Application of UNIshuttle to a New Personnel Information System

伊　奈　　健

要　約　システム構築の内容や方法が今までのメインフレーム一辺倒から，処理に適合した高機能・廉価版の機器の組み合わせによる形態になってきている．このことは利用者自身によるデータ利用の環境を確保することもシステム構築の目的の一つとなっていることを意味している．今回の人事システムの再構築も利用者自身によるデータ活用を近い将来の目的に，ホスト，部門サーバ，利用者端末の3階層のハードウェア構成で再構築した．3階層での構築を実現する上では，

・部門サーバに持つデータベースの維持管理

・利用者端末とホスト業務システムとの分散環境での連動

等をいかに解決していくかがシステム構築での課題となる．

本稿では，システム構築する上での上記課題を解決するためにミドルソフトウェアとして UNIshuttle を適用して実現した事例を紹介する．

Abstract　The method and objectives involved in the building of new systems have shifted from traditional concentration on mainframe computers to the use of various combinations of high-performance and low-priced devices that most comfortably satisfy data processing needs. This trend also means that the creation of an environment that makes a database available to end users has been one of the goals of the efforts to build systems. With the aim of allowing end users to have access to the database in the near future, a new personnel information system, of which the development the author has just taken part in, has been rebuilt in the form of a three-tier hardware configuration consisting of a host machine, departmental servers and end-user terminals.

The creation of a three-layer system entails the need for

— departmental servers' capability to maintain the database

— the efficient linkage of end-user platforms with the host system in the distributed data processing environment.

This paper presents a case where those requirements have been successfully met with the use of UNIshuttle as a middleware tool.

1. はじめに

コンピュータによる業務システム構築の構成が大きく変わってきている．システム構築の内容や方法は今までのメインフレーム一辺倒から，処理に適合した高機能・廉価版の機器の組み合わせによる形態となってきている．このことは全体費用の低減および段階的な展開はあるものの近い将来の利用者自身によるデータ利用の環境を確保しておくことが目的となっている．

M社における今回の新人事システムも，操作性の向上，社員数増大への対応，頻繁に行われる人事諸制度の変更への対応をいかに効率良く行えるようにするか，また近

本稿に記載の会社名，商品名は一般に各社の商標または登録商標である．

い将来の利用者自身によるデータ利用の環境を構築しておくことを要件として開発された．

これらの要件を実現するために現行人事システムの情報基盤整備を行うこととし，

・整合性のとれたデータベースをもつホストを中心とした人事システムの再構築
・LAN 環境を前提とした利用者使用端末とホスト業務システムとの分散環境での連動
・情報公開用としてデータベース・サーバの部門設置

を具体的方策とした．

これらのことは，ネットワーク技術がデータの発生・蓄積・保守を行う部署とデータを自分の手元に持ち込み，自分の環境で利用しようとする利用者との間の時間的・空間的隙間を埋めることであり，このネットワーク技術により実現可能と考へネットワーク技術のミドルソフトウェアとしては UNIshuttle を適用し実現することとした．

本稿は，この UNIshuttle を適用した事例として UNIshuttle を中心に以下の内容で記述する．

・新システムの要件および開発の課題
・新システムのハードウェア，ソフトウェア構成
・課題解決のための UNIshuttle 適用の具体例
・今後の課題

2. 新システムの要件および開発の課題

導入より 10 数年を迎えた既存システムは，機能の改善・拡張により要求を満足させては来たが，その半面，肥大・複雑化を招く結果となり，数々の問題点が指摘され，新たなシステムを次のような要件のもとに開発した．

① 頻繁に行われる人事諸制度の変更に対しての迅速な対応

② 操作性の向上

今までの入力方式がテキストベースのシンプルな機能により，操作に熟練した利用者による高速入力を可能としていた半面，入力データチェック機能や補助情報表示機能は乏しく，データ入力ミスの発生やコードブックなどの資料管理が必要となっていた．これらを解決すべくデータ入力にかかる作業負荷の軽減や入力ミス減少を目的として GUI の採用を行う．

③ オンライン・リアルタイム方式によるデータ入力

今までのバッチベースのデータチェックではデータ入力からチェックまでに時間がかかる事などからこれをオンライン・リアルタイム方式とする．

④ 整合性のとれたデータベースの構築

⑤ 利用者によるデータ活用の環境構築

④の整合性のとれたデータベースの構築が前提となるが，さらに機密性の高い人事データから公開用のデータを抽出し利用者の手元へ配置すべく部門データベース・サーバの設置を行う．

⑥ 既存システム資産の有効活用

今まで保持しているホスト・システムのプログラム資産を有効活用し新システムの早期立ち上げおよび開発費用の低減を計る.

上記要件実現のためのハードウェア，ソフトウェアとしてはホストに2200/310，TIP 1100，RDMS 1100（既存システム資産の有効活用，オンライン・リアルタイム方式の採用，データベース・システムの構築），部門データベース・サーバとしてU 6000/65，ORACLE(利用者によるデータ活用の環境構築)，利用者端末としてUS 40 E，開発言語としてはTIPPLER（操作性の向上ー GUIの導入ー）の採用を決定した（ハードウェア，ソフトウェア構成は図1，図2，アプリケーション構成は表2参照).

これらの構成で開発していく上でのネットワーク技術との関連での課題は，

① GUIの実現とオンライン処理によるデータ入力をどのように連動させるか

GUI実現のための利用者側ハードウェア，ソフトウェアをUS 40 EおよびTIPPLERとしたが，これらの環境とホスト・プログラムとの連動をどのように効率よく行うかが課題となる.

② 既存システムのプログラム資産をどのように新システム環境へ移行するか

既存システムではデマンド処理が中心であり，バッチ起動，パラメータの入力は0 S 1100が持つ基本機能により実現していた．これらの機能を現行プログラムに与える影響を少なくして実現するかが課題となる.

③ 部門データベース・サーバのデータをどのように維持するか.

ホスト側で更新されたデータをどのタイミングで，またどのような方式で部門サーバへ送信し更新させるかが課題となる.

これら課題を解決させるためにミドルソフトウェアとしてUNIshuttle機能を使用している.

UNIshuttleの使用機能は表1の通りである.

表1 UNIshuttle使用機能

実 現 内 容	UNIshuttle機能
GUIの実現ツールTIPPLERとホストのリアル／バッチ・プログラムとの連動(課題①②)	プログラム間通信機能
バッチ・プログラムの起動(課題②)	統合メニューおよびRJC機能
部門データベースの維持(課題③)	ファイル転送およびRJC機能

3. 課題解決のためのUNIshuttle適用の具体的内容

本稿では課題解決のためのUNIshuttle適用の具体的内容については，GUIの実現とオンライン処理によるデータ入力の連動について述べる．ただし，TIPPLERでのGUI実現およびTIPPLERのプログラミングについては本稿では割愛し，UNIshuttle適用のためのアプリケーションでの考え方を中心に述べる.

3.1 GUIの実現とオンライン処理によるデータ入力の連動

3.1.1 デマンド・プログラムとのプログラム間通信機能の実現

既存デマンド・プログラムは，利用者からのプログラムに対する指示はCOBOLが持つDISPLAY，ACCEPTの機能により実現している．このプログラムを大きく変更

表 2　新システムアプリケーション構成

	主な業務	業務内容
ホスト	TIP 業務プログラム バッチ業務プログラム	伝票入力処理(DB 更新処理) DB 検索処理 大量帳表作成処理 大量 DB 更新処理
部門サーバ	部門 DB 維持プログラム	
利用者端末	クライアント・プログラム	伝票入力画面インタフェース (入力内容の基本確認) 検索画面インタフェース ホストバッチ・プログラム起動 リスト・ボックス表示 ・ ・ ・

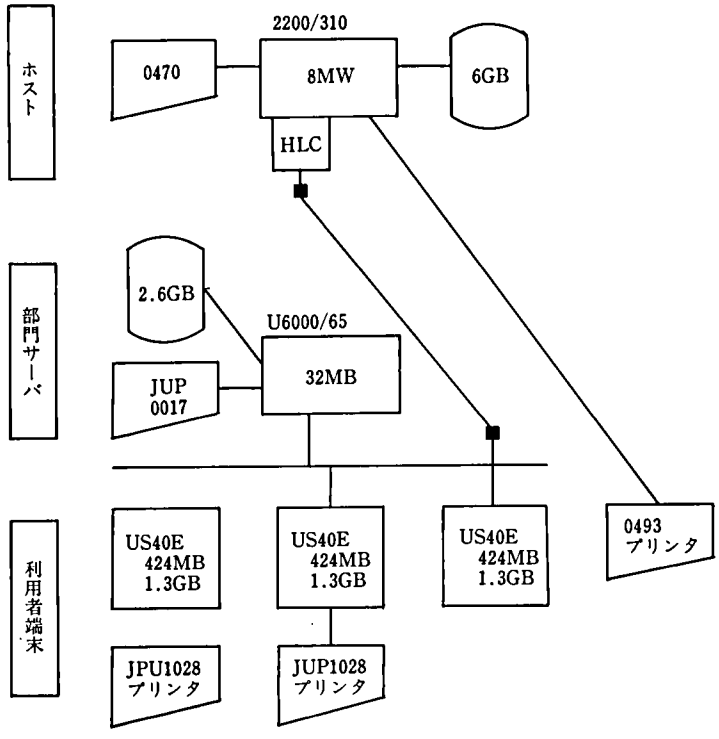
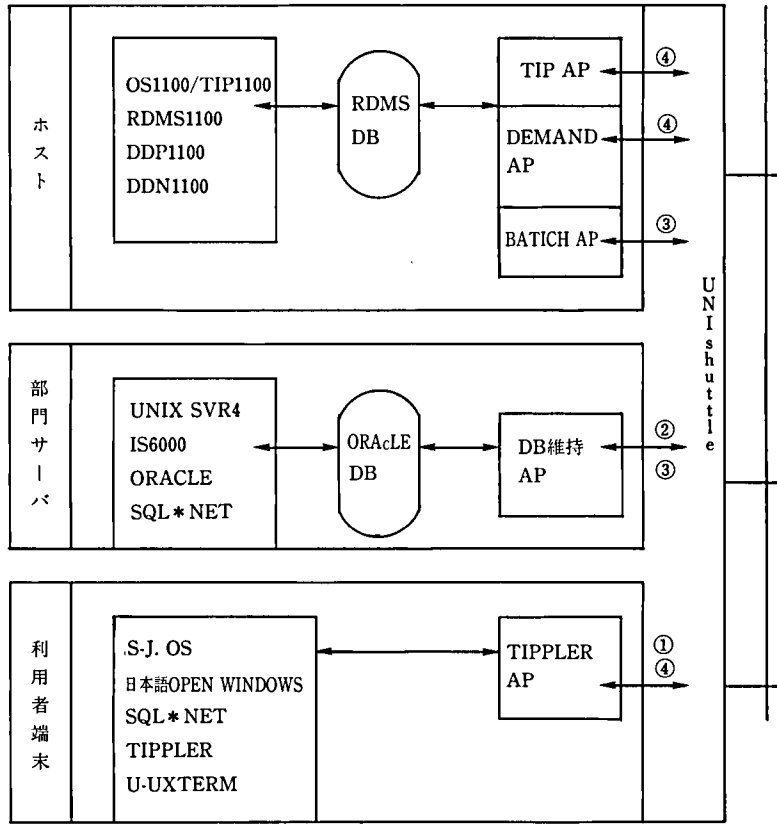


図 1　新システムのハードウェア構成



UNIshuttle 使用機能
①統合メニュー，②ファイル転送，③ RJC，④プログラム間通信

図 2　新システムのソフトウェア構成

することなく移行するために，UNIshuttle のプログラム間通信機能を使用している．処理の手順を図 3 に示す．図 3 には UNIshuttle が提供している各ファンクションで記述しているが，実際にはこれらのファンクションをコールするサブルーティンの形式で実現している．

サブルーティンを提供することにより，ホストの既存プログラムは自動変換をして移行している．

3.1.2　リアル・プログラムとのプログラム間通信機能の実現

リアル・プログラムは主として伝票処理の業務を行っている．伝票の形式から見ると，伝票のヘッダー部と伝票の明細部とに分かれているのが通常の形式である．利用者端末の操作は GUI を意識していることと，伝票設計は今までの 80×25 桁の制限から抜け出し自由度を持たせて設計することが可能となったことから，ヘッダー部と明細部の区分けはあるもののプログラム間通信での電文の構造も柔軟な構造にする必要があった．

今回のシステムではプログラム間通信の機能は UNIshuttle の機能を使用しているが（図 3 のバッチ・プログラム間通信のファンクションを使用），伝票の形式およびプ

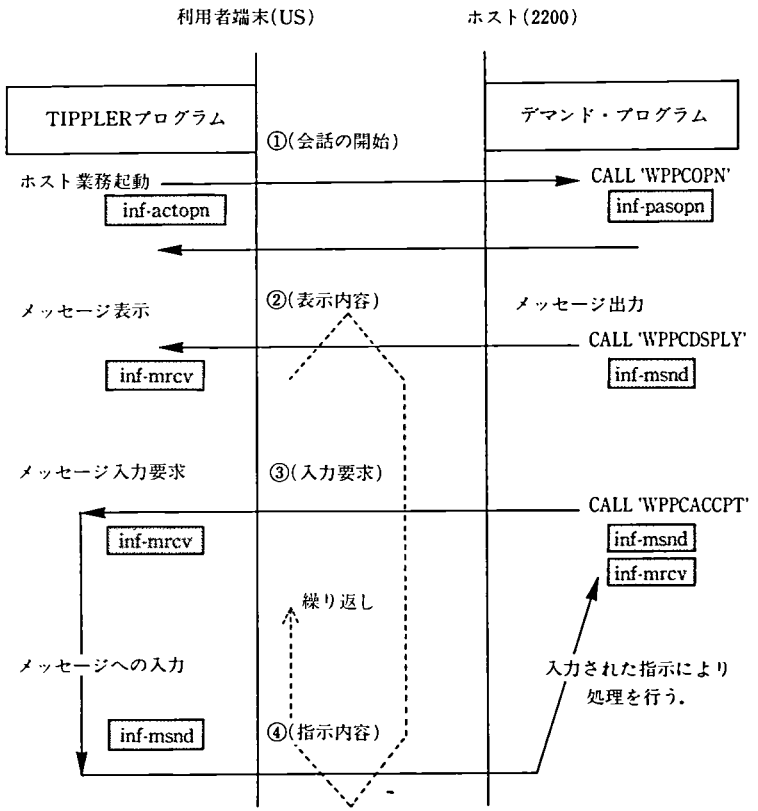
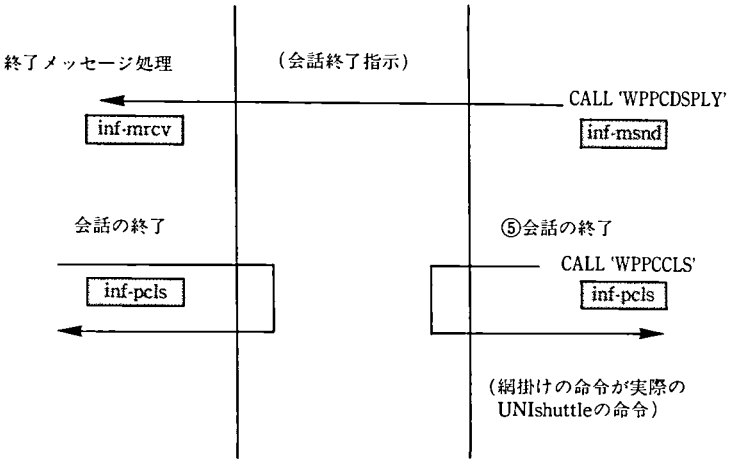


図 3　バッチ・プログラム間通信



① 会話の開始

ホスト・バッチ業務プログラムが会話の開始前に必ず呼び出す．この手順より会話を確立する．

CALL 'WPPCOPN'で実現する．

② 会話メッセージの表示

ホスト・バッチ・プログラムが起動元にメッセージを表示する時に呼び出す．

(COBOLのDISPLAY命令)

CALL 'WPPCCLS'で実現する．

③④ 会話メッセージの受取り

ホスト・バッチ・プログラムが起動元にメッセージ受取り要求を行い、メッセージを受け取る．

(COBOLのACCEPT命令)

CALL 'WPPCACCPT'で実現する．

⑤ 会話の終了

ホスト・バッチ業務プログラムが会話の終了前に必ず呼び出す．このことにより会話を終了する．

CALL 'WPPCCLS'で実現する．

ログラムの作成を以下のように規定した.

① 伝票の形式

・ヘッダー部のみの電文

・明細部のみの電文（明細部としては1行～n行まで自由度を持たせる）

を考え，これらに合わせてプログラム間通信での送受信電文のフォーマットを規定した.

フォーマットの概念図は図4の通りである.

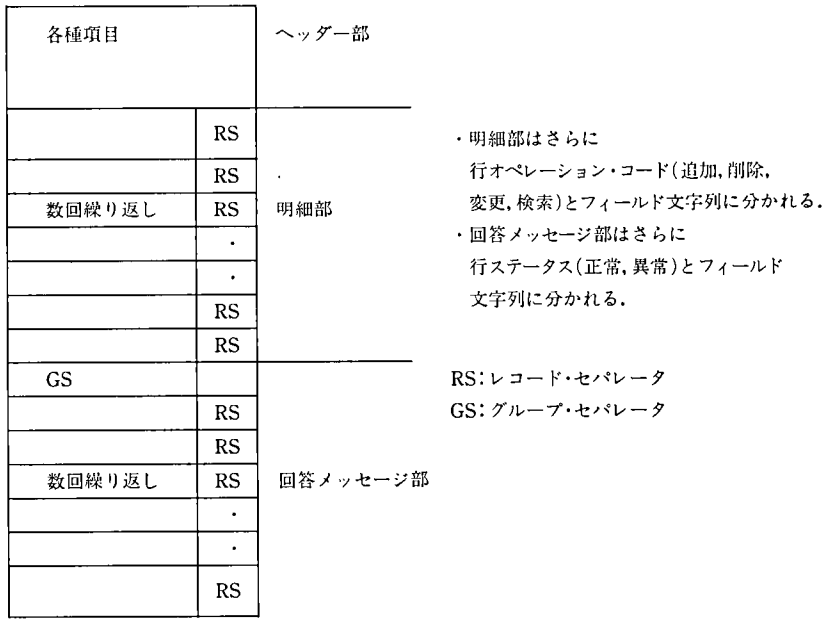


図 4　プログラム間通信電文フォーマット

② プログラム作成

リアル・プログラムの作成においてはプログラム間通信のコマンドやメッセージ送受信のタイミングに関わる設計やプログラミングに，多大な労力を要する．したがって，今回は伝票形式に合わせたプログラム・スケルトンを作成し標準化と省力化を計った.

スケルトンの概念図は図5の通りである.

4. おわりに

本稿は『エンドユーザ・コンピューティング特集』の一つであるが，今回述べた内容は，利用者自身によるデータ利用の環境を構築しておくことを要件として，基幹システムを再構築した事例である.

利用者自身によるデータ利用の環境を前提に考えたとき，どうしても分散環境が必要となり，現在の基幹システムを分散環境にどのように再構築するかが課題となる．ホスト側にまだアプリケーションが存在しクライアント側アプリケーションとの連動が必要となった今回の基幹システムは，今までのホストでの開発資産を生かしつつ新

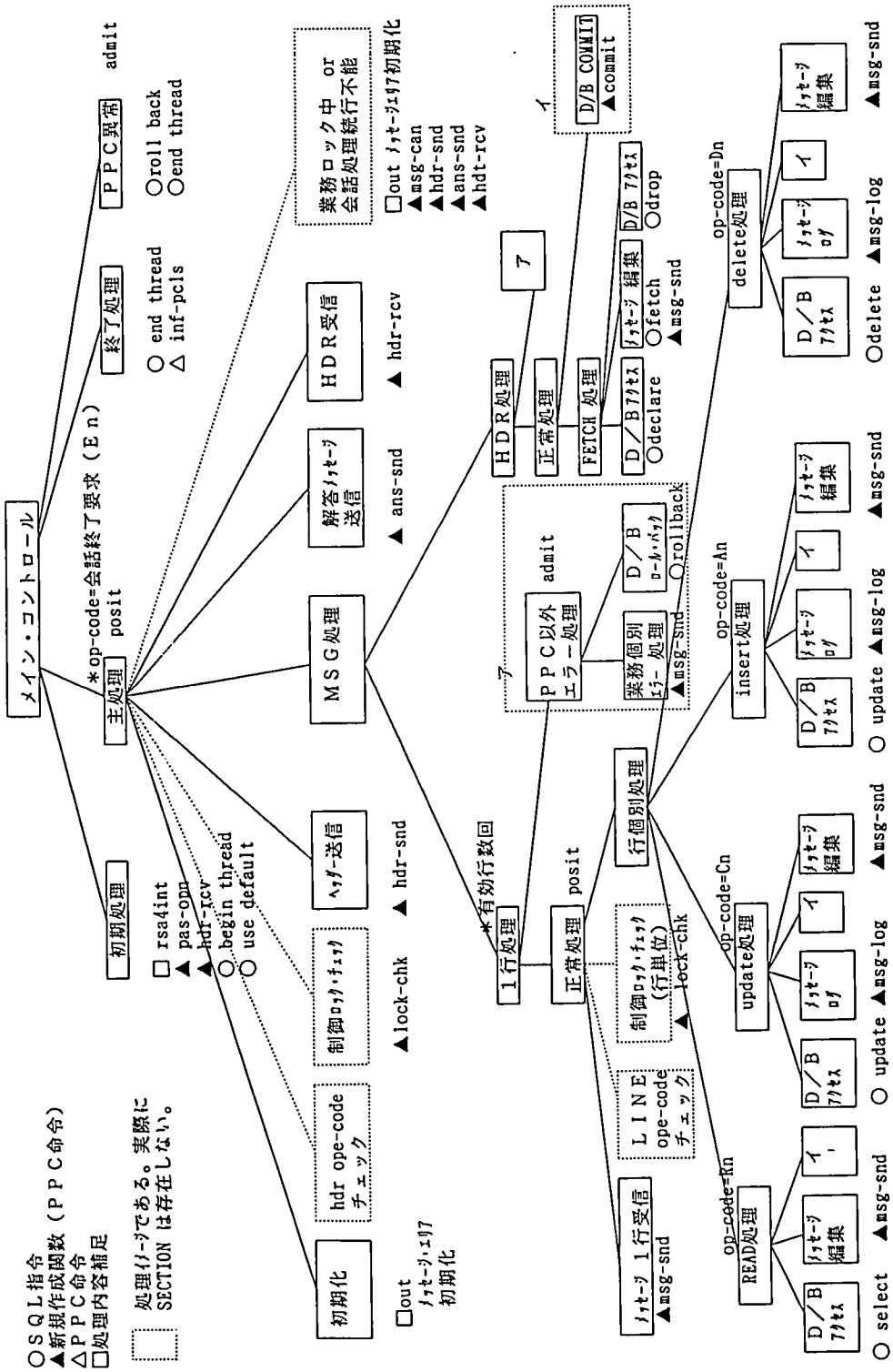


図 5　リアルプログラム標準スケルトン

しい環境で新システムを構築する事例であり，そのためのミドルソフトウェアとしてUNIshuttle は必要な機能を具備しており，有益なツールであったと考えている．

本システムは基幹システムを分散環境に構築した段階であり，今後は本来の目的である利用者自身によるデータ利用へと展開していく予定である．

執筆者紹介 伊　奈　　健（Takeshi Ina）

昭和 24 年生，47 年慶応義塾大学工学部電気科卒業．同年日本ユニシス入社．主として客先 SE サービスに従事，現在製造工業システム第 2 本部システム 1 部に所属．

UNIshuttleの概要
——分散処理システム構築向けの基盤ソフトウェア

An Overview of UNIshuttle——A Set of Infrastructure Software Tools designed to support Distributed Data Processing

本田親光

要約 UNIshuttleはオープンな分散処理システム構築の開発工数削減を狙いとした基盤ソフトウェアである．UNIshuttleは，多くのユーザに適用できる汎用性の機能とコンパクトな開発を期待されたために，開発に際し与えられた要件は，矛盾も含んだ多岐にわたるものであった．

UNIshuttle開発の特徴は，多くの要件の中から中核要件を探索しながら実現方案を案出し，これらを基本構造として機能の拡大を進め続けるアプローチにある．

このように開発したUNIshuttleは汎用性のある機能を持ち，分散を意識しないで開発でき，さらに機能拡張や変更ができる成長型の基盤ソフトウェアになっている．

本稿ではUNIshuttleの開発アプローチ，主要な実現方案・機能，その実現状況を概説する．つけ加えてUNIshuttle開発での中核要件設定の視点が，分散処理システム構築の開発者にとって汎用基盤ソフトウェア評価の視点に活用できることに言及する．

Abstract UNIshuttle is a set of infrastructure software tools aimed at reducing the number of development steps for open, distributed data processing systems. UNIshuttle was expected to provide a complete set of general-purpose functions applicable to as many users as possible, and to make it feasible to create systems in a simpler and less expensive way, and for this reason, the requirements raised for its development were far-reaching and included even some inconsistencies.

Typical of UNIshuttle development was the approach that required the author's team to find out how to implement core needs out of all different ones while trying to look for those nuclei and to continue extending capabilities with such kernels positioned as the elements of its basic structure. UNIshuttle, thus developed, has become a growth-type collection of infrastructure software tools, which offers general-purpose functionality, makes it possible to create systems without developers being aware of distributed data processing, and allows its capabilities to be changed and expanded.

In addition to briefly discussing the approaches to the development of UNIshuttle, major implementations/capabilities, and the status quo of the efforts, this paper also refers to the fact that the viewpoint from which to determine the core needs for UNIshuttle can also serve the developers of distributed data processing systems as the standpoint from which to evaluate infrastructure software tools.

1. はじめに

情報処理システムの新しい潮流として，オープンなネットワークを中心にした分散処理システムへのダウンサイジング化が進みつつある．

この分散処理システムは，処理特性に合わせ選択されたマルチプラットフォームの

本稿に記載の会社名，商品名は一般に各社の商標または登録商標である．

上で，分散データアクセス，分散プログラムを実行するクライアント・サーバシステム処理が多用される．

このように分散したプラットフォーム，データ，プログラムを相互連携させたり，開発したアプリケーション・プログラム（以降 AP と略す）を異なった OS のプラットフォームへ容易に移行するには，分散処理環境の標準化に基づくソフトウェアの採用が前提となる．

しかしながら，分散処理環境の標準化制定は遅れており，多くの場合，相互連携の保障のない市場で勢力を占めるソフトウェア群を組み合わせて構築する場合が多い．

したがって，これらのソフトウェアの適切な組み合わせを探るためのスタディ，および確認のテストに多くの時間を掛ける必要がある．また，標準化も進行中のため，その動向に合わせた対応も検討する必要があり，構築には解決しなければならない課題は多い．これら課題の解決策として，オープンな分散処理システムを早急に構築したい開発者は，どのプラットフォームでも動く AP が作れ，将来の標準動向にも対応でき，ソフトウェアの組み合わせを考えずに済む分散処理環境用の基盤ソフトウェアの出現を望んでいる．

UNIshuttle は，どのユーザでも使用する機能の網羅性を持ち，かつ将来の技術に追随する構造を持った分散処理システム構築向けの汎用基盤ソフトウェアである．この UNIshuttle 開発の特徴は，利用者に対し分散を意識させず，汎用性を持ち，成長する構造を持ち，かつ短期間の提供などの多くの要件を満足させることにあった．

本稿では 2 章で UNIshuttle 開発の進め方を述べ，3 章で実現方案，4 章で基本仕様，5 章で UNIshuttle の実現状況を報告し，6 章ではオープンな分散処理システム構築の開発者に対し，基盤ソフトウェアに対する評価の視点を明かにする．

2. UNIshuttle 開発の進め方

2.1 UNIshuttle 開発の背景

情報処理システムの現状は，高性能で多機能なワークステーションへのダウンサイジングが進み，また PC の普及によりコンピュータがエンドユーザに身近なものになっている．さらに LAN，WAN の物理的ネットワーク環境が整備されてきており，分散システムを推進する基盤はできているといえる．

しかしながら，実際の構築に際しては，WINDOWS NT，GUI 等の新しいソフトウェア，技術は多岐にわたっており，システム部門でさえも全てを修得するのは困難になってきている．また，多様で進歩の早いオープンなハードウェア，ソフトウェアの組み合わせ方法，将来の再配置対策，制約などの情報入手は難しく，自ら体験しながらノウハウの蓄積をする場合が多い．

このような状況でマルチベンダ提供の製品を組み合わせた分散システムを構築する場合，AP の可搬性を保持し，プログラム開発工数を少なくする基盤ソフトウェアの採用が有効な対応策となる．

UNIshuttle はオープンな分散システム構築の開発工数削減を狙いとし，機能の面で汎用性があり，将来の技術進歩にも追従できる構造を持つ分散処理システム構築向けの基盤ソフトウェアとして開発した．

2.2 UNIshuttle の要件

開発要件の洗い出し方法は，標準化団体の分散環境機能モデルを参考に，分散システム構築を早急に目指すユーザの要望，あるいは過去の事例から網羅的に収集した．この収集した要件を，分散 AP 実行面の要件，保守・運用面の要件として分類した．また，提供する上での前提・制約も要件として追加している．

2.2.1 分散 AP 実行面の要件

1) AP は OS，ハードウェア，ならびにコード体系のタイプに依存しない可搬性を保持すること
2) 物理的な経路やプラットフォーム，データベースの物理的位置を指定しなくて済むこと
3) 実行方式は，クライアント・サーバモデル，かつマルチ・クライアント，マルチ・サーバ形態が可能なこと
4) 各種分散サービス機能の提供
   - トランザクション処理
   - プログラム間通信
   - ファイル転送
   - SQL によるデータアクセス
   - リモート・ジョブエントリ
   - プリント
   - メイリング
   - メニュ（GUI）
5) AP ごと，データベースごとにセキュリティの設定，制御が可能なこと

2.2.2 保守・運用面の要件

1) 一元管理のできるネットワーク・サービス機能の提供
   - システム構成管理
   - 稼働状況管理
   - 使用状況管理
   - データ・バックアップ管理
   - プログラム・リリース管理
   - セキュリティ管理
2) 業務運用面のサービス機能の提供
   - 日次/月次でのデータの締め機能（時間指定，同期確認など）
   - トランザクション/ジョブ/ファイルのスケジュール管理
   - エラー時のリカバリ支援

2.2.3 UNIshuttle 開発の前提・制約

1) マルチ・ベンダ提供のマルチ・プラットフォーム（メインフレーム，ワークステーション，PC）への対応が可能なこと
2) タイミング良い供給，期間 1 年以内の開発にすること
3) 多くのユーザで適用できる汎用性を持つこと
4) コンパクトであること（構造がシンプルで開発量は少なく）

5) 将来の技術標準へ対応できる成長型ツールであること

2.3 要件抽出後の進め方

網羅的に要件の抽出を行ったことにより，実現すれば多くの機能を提供でき多くのユーザに対応できるが，反面一部のユーザには不用な機能も混在し，コンパクトな開発にならないと予想された．また，広範囲な要件のためどこから着手すれば良いか分からないという汎用ソフトウェアの開発で起きがちな問題も発生した．

この解決方法として，汎用基盤ソフトウェアとしての備えるべき要件を層別し，中核となる要件を設定し，それらを満足させる実現方案の中で基本構想，基本構造を策定し，そのあと残っている要件の可能性を探る進め方を採用した．

2.4 UNIshuttle 中核要件の設定

汎用基盤ソフトウェアの中核要件設定の考え方としては，ユーザの狙いであるオープンな分散システム開発工数の削減，維持工数の削減に対し大きく貢献できる点が何であるかに着眼する．

一つめの着眼点としては，ベンダ製品中心の従来型集中処理システムの開発・維持では考慮しない作業であるが，分散処理システムでは工数が発生する，あるいは工数が増加する作業に焦点を当て，それらを抑制する要件を中核に設定する．

開発段階において，分散処理システムゆえに工数が増大する作業は以下のとおりである．

- 分散したプラットフォーム間をつなぐネットワーク接続作業
- 分散したプラットフォームの運用管理システムの構築作業
- 可搬性を考慮するプログラム開発環境の構築作業

維持段階において工数が増大すると想定される作業は以下のとおりである．

- 将来の技術，標準化への進歩に合わせ拡張，変更する作業

これら項目と網羅的に抽出した要件を照らし合わせ，工数抑制の中核要件としての設定を行った結果は以下のとおりである．

1) 開発段階での工数増大抑制のための中核要件
   - マルチ・プラットフォームを接続可能とするオープンなネットワークとする
   - ネットワーク・リソース（プラットフォーム，データベースなど）の物理的な位置をユーザ指定の論理アドレスで一元管理できること
   - プラットフォームに依存しない AP 構造になること
2) 維持段階での工数増大抑制のための中核要件
   - 既存の AP に影響を与えず，将来の技術の進歩に合わせサービス機能の拡張，変更ができること

二つめの着眼点は，多くのサービス機能を提供することであるが，短期間の開発では一挙に提供はできないため，段階的にサービス機能を提供できる構造を中核の要件に設定する．

3) 多くのサービス機能を提供するための中核要件
   - 多くのサービス機能を提供し，かつ汎用性を失わない構造であること

上記中核の要件に対する個々の実現方案を 3 章にて述べる．

3. 実現方案について

3.1 マルチ・プラットフォームを接続可能とするオープンなネットワークの実現案

分散ネットワークの通信プロトコルとして，業界標準のTCP/IPを採用する．TCP/IPプロトコルは，どのプラットフォームにおいても適用されており，これにより接続に関する作業は，TCP/IPプロトコルを支援する各社のソケット（SOCKET）インタフェース（IF）ライブラリの接続確認のみになる．UNIshuttleは，ソケットIFライブラリを分散ネットワークの最下層入出力インタフェースとして提供し，接続可能の保障をこのレベルで行うこととする［標準技術の採用］．

3.2 論理アドレスによるネットワーク・リソース一元管理の実現案

1) 分散ネットワーク内のハードウェア，ソフトウェア資源の物理的位置を含めた情報を特定のプラットフォーム（マネジャと称す）のシステム構成ファイルにユーザ指定の論理アドレスをキーにして一元管理する．

そのシステム構成情報は，マネジャから各プラットフォーム（エージェントと称す）に複製情報として配布し，ネットワークのシステム構成内容を維持する形態をとる［マネジャ/エージェント方式の採用］．

2) システム構成情報が送られた各プラットフォーム内のサービス機能は，相手先位置の特定には，常にこのシステム構成情報（ネームテーブルと称す）を参照し相手先の物理的位置を自動的に算出する機能を使用する［ネームサービス機能の新設］．

3.3 プラットフォームに依存しないAP構造の実現案

1) アプリケーションの実行処理形態は，処理を要求するプラットフォームをクライアント，要求された処理を実行するプラットフォームをサーバという，クライアント・サーバモデルに限定する．その他の分散実行処理形態であるPIER TO PIER処理モデル，オブジェクト処理モデルもあるが，クライアント・サーバモデルの早期リリースを優先する［優先度による設定］．

2) APに対するサービス機能のインタフェースは，異なるプラットフォームでも共通なAPI（アプリケーション・プログラム・インタフェースの略）を設定する［共通APIの採用］．

これは図1に示すようにUNIshuttleがAPとプラットフォームの中間に入

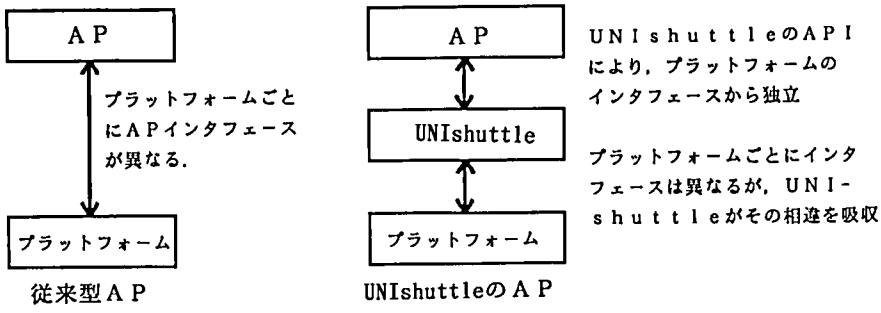


図1　APの可搬性保持の構造

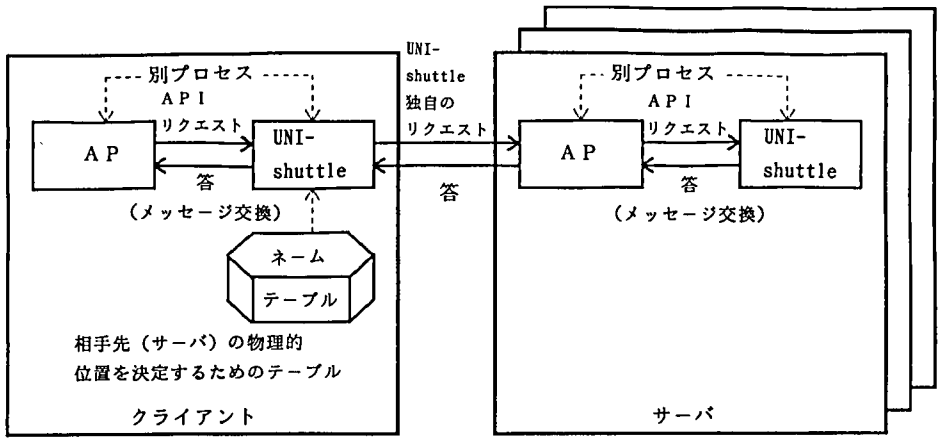


図 2　UNIshuttle 実行構造の概念図

り，プラットフォームの相違を吸収する．AP は UNIshuttle の API を使用すれば，異なるプラットフォームに移行することが可能となる．これにより AP は可搬性を保持することになる．

3)　AP と UNIshuttle は，図 2 に示すように，API が定めたメッセージ交換のプロセス間通信にて連結する．UNIshuttle のクライアントとサーバのプロセス間では AP には影響しない UNIshuttle 独自の定められたメッセージ交換を実行する．この形態は，クライアントとサーバの AP は UNIshuttle により完全に切り離されることになり，サーバとクライアントの AP が異なるプラットフォームに移動しても UNIshuttle が存在すれば，影響をなく作動できる構造になっている．これにより AP は UNIshuttle のプロセスが存在するプラットフォームへ自由に移動することが可能となる［プロセス間通信方式の採用］．

UNIshuttle 独自のメッセージ交換の概要は 4.3 節の分散プログラム実行制御方式にて後述する．

4)　AP，データベースがどのサーバに存在するか，あるいは変更するかへの対応のため，クライアント側サービス機能の実行時にネームテーブルを参照して自動的に AP，データベースの物理的存在場所を特定するディレクトリサービス(ネームサービスと称す）をプラットフォームに組み込むこととする．

実現方案として，図 3 に示すとおり，クライアント側 AP からのサービス機能要求を受付ける UNIshuttle のプロセス（ファンクション・マネジャと称す）の中にネームサービス機能を実装する．これにより AP は物理的存在場所を意識しない実装となる［ネームサービスの AP からの独立採用］．

5)　それぞれコード体系を持つ異なるプラットフォーム間を行き交う API パラメータのコード体系への変換は，UNIshuttle のファンクション・マネジャが行う．これにより API はプラットフォーム機種ごとのコード体系に影響されない．

またファイル/メッセージの送受信における，データのコード変換も指定により UNIshuttle が行う．

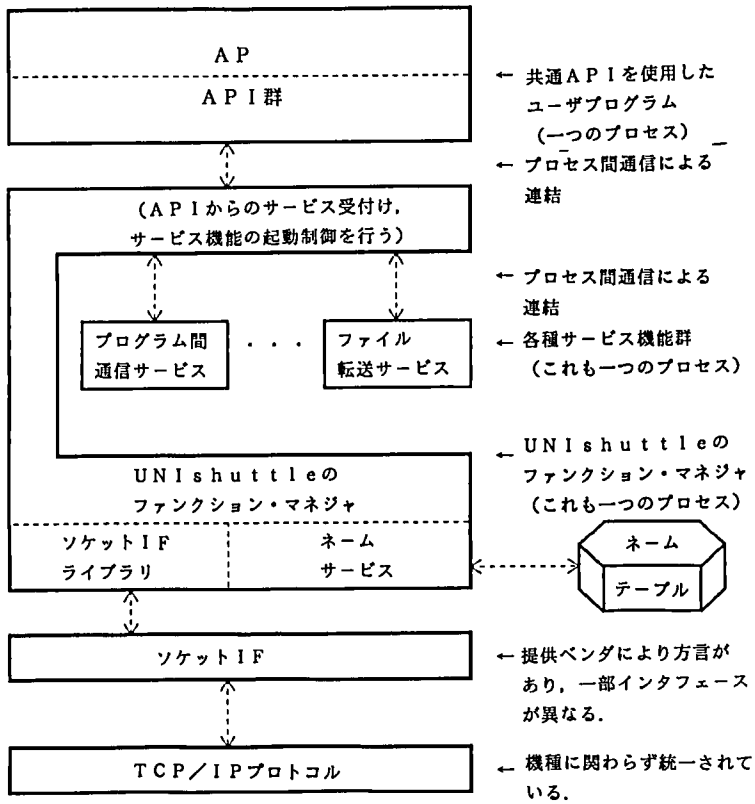


図 3　クライアント側実装の関連図

3.4　既存 AP に影響を与えないサービス機能の拡張，変更の実現案

AP と UNIshuttle 部分は前述の共通 API の採用，プロセス間通信方式の採用により API 処理を除き実装レベルでは分離される．これにより共通 API の変更がない限り，UNIshuttle のサービス機能の変更，新規追加を行っても既存 AP への影響を与えない構造になる．

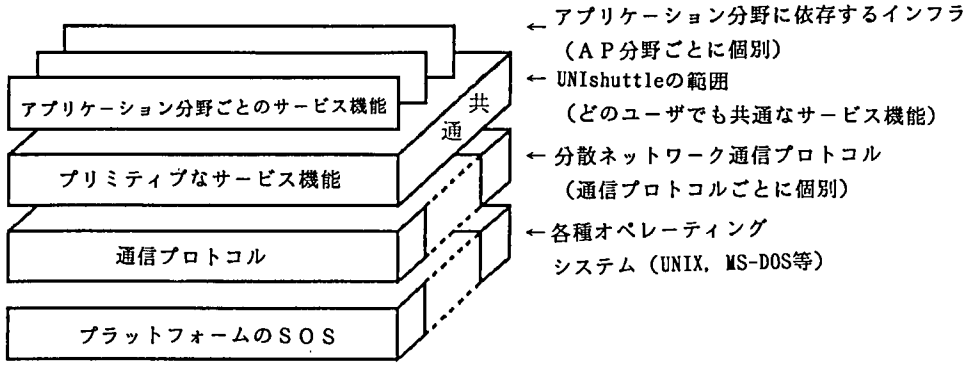
3.5　多くのサービス機能が実現し，かつ汎用性を失わない実現案

1)　サービス機能は，どのユーザでも使用するプリミティブな機能とアプリケーション面の要求に関る機能に層別する．この層別化により，プリミティブな機能階層は汎用性を保持でき，アプリケーション面の要求機能は，プリミティブなサービス機能階層の上にアプリケーション分野ごとに構築することにより，多くのサービス機能を段階的に提供できることになる．

開発の考え方としては，アプリケーション分野のサービス機能は，アプリケーション分野用基盤ソフトウェアに委ね，UNIshuttle の対象範囲として図 4 に示す汎用性のあるプリミティブな機能階層に定める［対象層別化の設定］．

2)　UNIshuttle のネットワーク間の入出力インタフェースは TCP/IP プロトコルを使用するソケット IF ライブラリにする．

この上に UNIshuttle 用メッセージ交換ライブラリを置き，これをもとにサービス機能を構築する．そのサービス機能のプログラム階層は図 5 に示すとおりで



・UNIXはX/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標である.
・MS-DOSは米国Microsoft社の登録商標である.

図 4 基盤ソフトウェア（インフラ）サービス機能の層別化

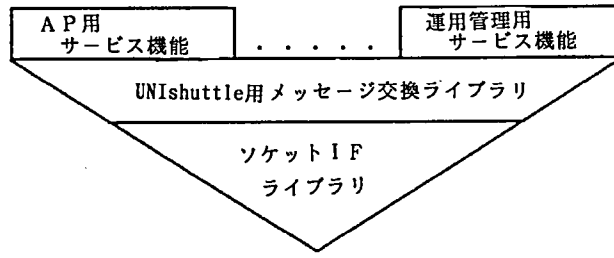


図 5 サービス機能の階層構造

ある.

この構造は, TCP/IP プロトコルを使用するだけであり, 他の AP, ソフトウェアには影響を与えないことを示しており, したがって他のソフトウェアとの組み合わせで多くのサービス機能の実現もできる.

4. UNIshuttle 基本仕様の設定

3章での実現方案をもとに, 今後の標準化を意識しながら汎用的に使用される AP サービス機能の設定, 分散システム構築に必要な操作の簡単な運用機能の設定を行った. これらを UNIshuttle の基本仕様として述べる.

基本仕様の内容は適用範囲, 対象プラットフォーム, プログラム実行制御方式, 提供サービス機能である.

4.1 適 用 範 囲

3章での実現方案により適用範囲は必然的に定まり, 以下の項目を前提とした適用とする.

1) クライアント・サーバモデルのコンピュータ実行処理であること.
2) TCP/IP プロトコル採用の分散ネットワーク環境であること.
3) 大規模トランザクション処理でないこと. (大規模トランザクションとは, トラ

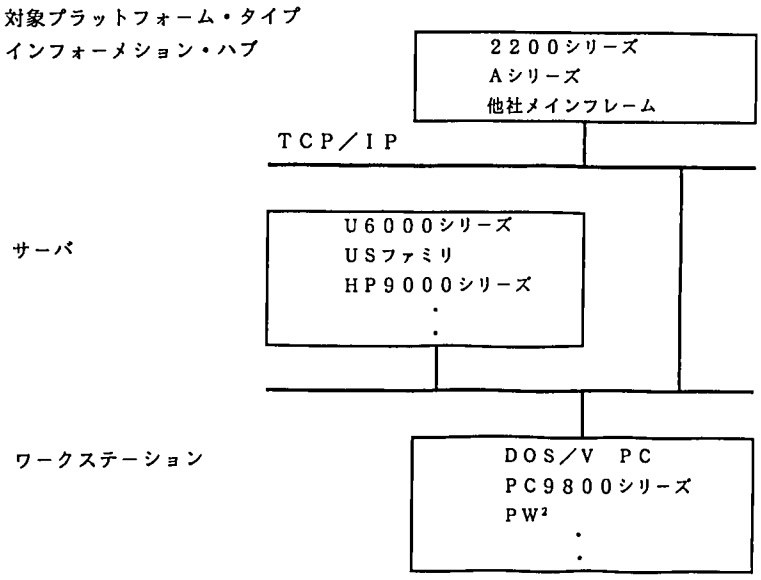


図 6　対象プラットフォーム・タイプ

ンザクション・リカバリが必要であり，トランザクション流量制御ができ，2フェーズ・コミットメントが必要な処理)

このサービスは，他のトランザクション処理用のソフトウェアに委ねることとする．

4.2　対象プラットフォーム

対象のプラットフォームは，図6に示すTCP/IPプロトコルでLANに接続可能なメインフレーム，UNIXワークステーション，パーソナル・コンピュータとする．これらのプラットフォームは，UNIshuttleではノードと呼ばれシステム構成ファイルの中で一元管理する．

4.3　分散プログラム実行制御方式

UNIshuttleクライアント・サーバモデルの基本的流れを，図7にUNIshuttleのサービス機能の一つであるプログラム間通信サービス機能をもとに示している．

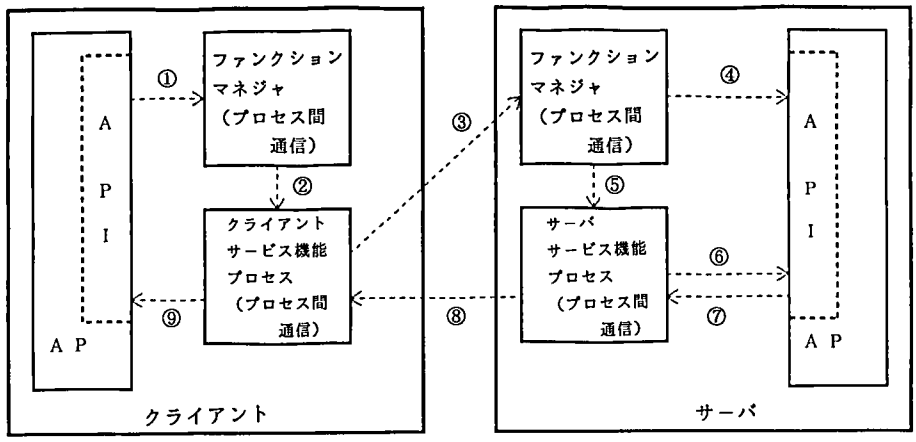
図7の処理の流れは，UNIshuttleが提供する各種サービス機能において基本的に遵守する．これは，APIとサービス機能をプロセスとして分離し，これにより，基本ソフトウェアなどの変更による影響をAPに与えず，APの独立性を保つためである．

4.4　UNIshuttleの提供サービス機能

UNIshuttleが提供するサービス機能は，分散システムを構成するリソースを管理するシステム構成管理とこれに連動するネームサービス，APと運用管理へのサービス機能である．

4.4.1　システム構成管理とネームサービス

UNIshuttle内の管理するリソースとして，ノード情報（ノード名，IPアドレスなど)，ファイル情報（ファイル名，存在ノード名，セキュリティ情報など)，データベース情報(データベース名，存在ノード名，セキュリティ情報など)，プログラム情報



① クライアント側ファンクション・マネジャのコネクション確立，
入力パラメタの引き渡し
② プログラム間通信用のプロセス起動，入力パラメタ引き渡し
③ サーバ側ファンクション・マネジャへのコネクション確立，入力
パラメタの引き渡し
④ 実行要求サーバ側ＡＰ起動
⑤ サーバ側プログラム間通信用の子プロセスを起動
⑥ 入力メッセージの引き渡し
⑦ 出力メッセージの引き渡し
⑧ 出力メッセージの転送
⑨ ＡＰへの出力メッセージの引き渡し

図 7　UNIshuttle の実行処理の基本的流れ

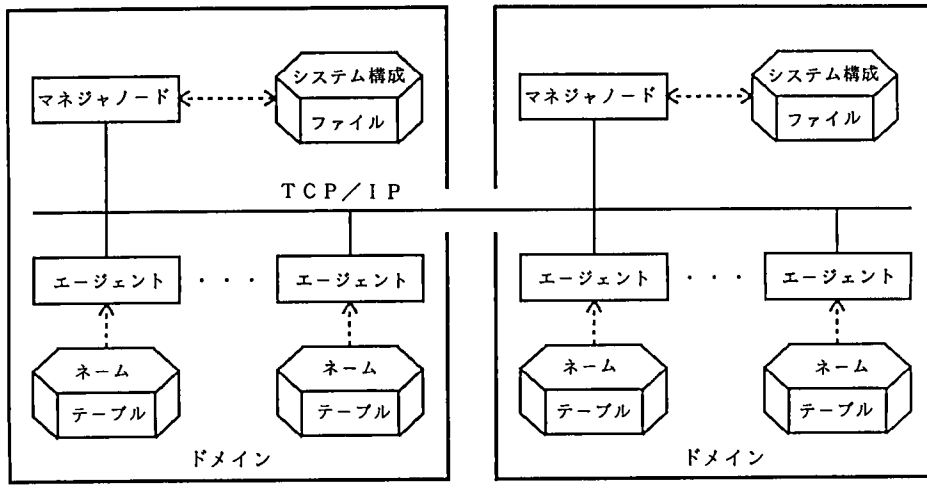


図 8　ドメインの構成

(プログラム名，存在ノード名，セキュリティ情報など)，ユーザ情報（ユーザ名，ユーザ ID，グループ名など）などがある．

これらリソースを管理する作業領域単位として図 8 に示すドメインという概念を設定している．

ドメインは一つのマネジャノードと一つ以上のエージェントノードから構成される．マネジャノードは，自分のドメイン内リソースをエージェントノードをとおして一元的に管理する．

システム構成管理は，ドメインの定義，ドメイン内のリソースの登録，および登録されたシステム構成ファイル内の情報を高速検索可能なネームテーブルに変換し，ドメイン内のすべてのエージェントに対し自動配布する．

ネームサービスは，エージェント側に自動配布されたネームテーブルからリソース情報を獲得し，論理名（ノード名，ファイル名など）から物理的位置に変換する．

この機能により，APはリソースの物理的位置を意識することなく論理名を指定するだけでUNIshuttleのサービス機能を呼び出せる．

4.4.2 APへのサービス機能

UNIshuttleがAPへ提供するサービス機能は以下のとおりである．

1) ファイル転送機能……クライアント，サーバ間でファイルの送受信を行う．転送ファイルとしてはテキストファイルとバイナリファイルの選択ができる．また時刻指定，同報指定の制御も備える．
2) リモートジョブコントロール機能……クライアント側から，JCLまたはシェルプログラムをサーバに転送・実行し，その実行結果を受け取る．
3) トランザクション処理……クライアント側APからサーバ側APを起動したのち，1件のトランザクションを転送し，その結果を受け取る．
4) プログラム間通信機能……クライアント側APとサーバ側APが会話的にデータの送受信を行う．トランザクション処理との違いは，複数件のデータの送受信が自由にできる．
5) SQLインタフェース機能……クライアント側APで生成したSQL文をサーバ側のリレーショナル・データベースに送信し，その結果を受け取る．
6) プリント機能……クライアント側APから，印書装置が接続されているサーバに対し印書指示を行う．出力順の変更，出力停止の制御も備える．
7) 電子メール機能……クライアント側APからマネジャが持つメールボックスまたは掲示板へのメール文を送信する．

4.4.3 運用管理のサービス機能

分散システムの運用管理機能として以下の機能を提供する．

1) 稼働状況管理……ドメイン内の各ノードの稼働状況(稼働中，停止中等)，および負荷状況（CPU利用率，メモリ利用率等）を監視する．
2) 使用状況管理……ドメイン内の各ノードにおけるシステム資源の使用状況を収集し集計する．
3) データ管理……ファイル，データベース，プログラムのテープあるいはディスクへの保存，およびテープあるいはディスクからのリロードを行う．
4) プログラム・リリース管理……ドメイン単位にAPまたはファイルをマネジャからエージェントへ配信（リリース）する．

5. UNIshuttle の実現状況

4章における基本仕様は UNIshuttle で実現しユーザに提供されている．

これらの実現状況を中核要件の観点，網羅的に抽出した要件の観点，業界標準化団体から出された主要な機能の観点から述べる．

5.1 中核要件に対する実現状況

1) マルチ・プラットフォームを接続可能とするオープンなネットワーク分野
   - TCP/IP を標準通信手順として採用しており，TCP/IP を支援するプラットフォームであれば，容易に UNIshuttle 内に組み込み可能となっている．
   - 汎用機(2200 シリーズ，A シリーズ)，UNIX(US ファミリ，U 6000 シリーズ)および PW², J 3100，PC 9800，DOS/V，Mac などの数多くのプラットフォームを標準で支援できており，ニーズに応じた選択の幅は広くなっている．今後もさらにプラットフォーム種類の拡大をはかる．

2) 論理アドレスによるネットワーク・リソースの一元管理分野
   - 一元管理されたネームサービス機能によって，開発者，利用者はデータベース，ファイル，プログラム等の物理的存在位置を意識せずにプログラム作成が可能になっている．
   - 分散システムを構成するネットワークや機器の構成変更およびそれに伴うデータベース，ファイル，プログラム等の物理的存在位置の変更が発生しても AP への影響はなくなっている．
   - 一元管理されたネームサービス機能により，分散システム内のノードに対し，稼働状況監視，各ノードへのプログラム配布，データの配布，各ノードからのデータ・バックアップの集中運用管理が可能となっている．

3) プラットフォームに依存しない AP 構造分野
   - LAN ネットワーク環境で水平・垂直分散処理システムを構築できる API とコマンドを提供している．これによりプラットフォームに関する専門的な技術を熟知しなくても分散システムの開発ができる．

4) 既存 AP に影響を与えないサービス機能の拡張・変更分野
   - AP と API は，プロセス間通信をとおして提供される各種サービス・モジュールと分離されており，サービス機能の拡張・変更，基本ソフトウェア等の変更があっても AP への影響は生じない．

5) 多くのサービス機能が実現し，かつ汎用性を失わない分野
   - UNIshuttle の対象範囲を共通性のあるプリミティブな範囲にしたことにより，どのユーザでも使用する汎用性を保持している．この UNIshuttle を土台にサービス機能の追加ができ，今後より多くの機能拡大が可能になっている．

5.2 網羅的な要件（2.2節記載の要件）に対する実現状況

網羅的要件に対し実現できなかったものは，業務運用面のサービス機能である．これは，これら機能の制御（時間指定，同期処理，起動方法）の組み合わせをパターンとして汎用化するのが難しく開発を見送ったことによる．今後，簡便な組み合わせの機能は提供する予定である．

5.3 業界標準の分散環境機能に対する実現状況

UNIXの標準団体であるOSFのDCE（Distributed Computing Environmentの略）の主要な機能は，以下のとおりである．

1) RPC（リモート・プロシージャ・コール）：処理の呼出し
2) ディレクトリサービス：処理，サーバの自動認識
3) セキュリティサービス：暗号化と認証
4) タイムサービス：ユニバーサル・クロックとタイムサーバ
5) 分散ファイルサービス：ネットワーク・ファイル・システム

UNIshuttleでは上記の機能に比べ，タイムサービス，分散ファイルサービスが提供されていない．分散ファイルサービスに関しては，OSあるいはネットワークOSで提供されており，これらの活用で補完する方が良いと考える．タイムサービスは，時間制御の自動化に重要な機能であり，今後の標準化の動向を踏まえた対応を行う．

6. 分散処理システム構築における汎用基盤ソフトウェアの評価視点

汎用基盤ソフトウェアに対する実現方案，基本仕様を，ソフトウェアを提供する立場から述べてきた．一方オープンな分散ネットワーク環境の中で，APを開発する立場から基盤ソフトウェアに対する主要な評価の視点は，以下のとおりと想定される．

1) 分散を意識せずに開発できるか［相互運用性］
2) 今後の技術，標準の進歩に合わせ変更・追加できるか［柔軟性］
3) 必要とするサービス機能があるか［網羅性］
4) 操作の容易性はどうか
5) 信頼性，効率性があるか

上記の中で4)・5)項は従来の集中処理システムから分散処理システムに移行しても変わらない評価の視点である．

一方，1)・2)項は従来の集中処理システムでは評価のなかった視点であり，3)項は，評価の範囲，レベルが変化している視点である．

したがって，分散システム構築用の基盤ソフトウェアに対する評価項目は，4)・5)項の視点では，従来の評価項目を参考に変更すれば良いが，1)，2)，3)項の視点では新規に評価項目を作成する必要がある．

このユーザの基盤ソフトウェアに対する評価の視点は，本稿で述べてきた中核要件設定の視点と同一であり，本稿の中核要件は評価項目として活用できるはずである．

7. おわりに

要件，課題の多い分散システム構築向けのUNIshuttle開発の進め方，実現方案中心の記述により，汎用基盤ソフトウェアの評価ポイントを示すことができたと思う．反面利用する立場からの機能・利用法・操作性の説明がなく，UNIshuttleの機能を十分に紹介しきれていない．この詳細は参考文献[1]をご一読願いたい．また，開発工数削減の効果の記述が抜けており，この点は今後の事例の積み重ねにより明らかにしていくつもりである．

なお，本稿で紹介したUNIshuttleは平成5年4月にリリースされ，ユーザへの適用

が拡大すると共に，プラットフォームの追加，あるいは機能追加も続けられ，成長し続ける汎用基盤ソフトウェアを実証している．

参考文献 ［1］ 日本ユニシス「分散システム構築ツール UNIshuttle 概説書」.

執筆者紹介 本 田 親 光 (Chikamitsu Honda)
昭和 22 年生．46 年早稲田大学理工学部電気工学科卒業．50 年日本ユニシス(株)入社．主に製造分野向けのシステムの開発，サポートを担当．現在，製造工業営業本部営業一部に所属．

OLTP——基幹システムとEUCの融合

Interoperation of End User Computing (EUC) and an Open/OLTP Mainframe

佐藤友彦, 久米進也

要　約　近年，高性能でしかも低価格なパーソナルコンピュータ（PC）の提供やGUI（グラフィカル・ユーザ・インタフェース）を駆使したワープロや表計算ソフトなどの出現により，単にパーソナル・ユースとして利用されていたPCが，ローカル・ネットワークを構成する一部として利用されるようになり，PC上での処理が業務系の一部を担うまでになってきた．一方，ホストコンピュータにすべての処理を任せていた中央集中型の基幹システムからのライトサイジング化が現実のものとなりつつあり，単にホストコンピュータにアクセスするためだけの大量な端末を抱えるシステムから，様変わりをしてきている．こうした中で，PCの構成するローカル・ネットワークとホストコンピュータ上の基幹業務を連動し，従来ホストで行ってきた情報の加工処理を個人のニーズにあわせて自由に行えるシステムが求められるようになってきた．

本稿では，PCとホストコンピュータ間のクライアント/サーバ・システムについてOpen/OLTPプロダクトを例にして考察する．

Abstract　Availability of high-performance, low-priced PCs as well as the advent of GUI-loaded word processors and more powerful spreadsheets have upgraded the position of PCs, which were traditionally intended for personal use, to the place where PCs are regarded as part of local area networks; thus enabling data processing by PCs to take on partial responsibility for practical computer applications. On the other hand, what has become a reality is the move to right-size centralized computer systems where all mission-critical applications are processed by host systems. That has boosted a transition from the way in which a large number of terminals are installed only to have access to host computers. Reflecting such changes, information technology users have begun to choose systems that allow end users to freely process the information, according to their own needs, that has conventionally been handled by mainframes through the linking of PC-based LANs with major applications processed by host equipment.

By taking Open/OLTP distributed transaction processing software as an example, this paper discusses a client/server system that exists between PCs and a host computer.

1. はじめに

毎年200万台以上の割合で増え続けるPCは年々その機能を高める一方で，低価格化もさらに拍車がかかっている．さらに，WindowsといったGUIベースのインタフェースを提供するOSとそのもとで稼働するワープロソフトや表計算ソフトの普及により，PCがますます身近なものとなっている．また，ファイルサーバやプリントサーバなどをローカルなネットワークに設置し，限られた範囲でのデータの受け渡しが容易

になってきている．こうしたPCベースのネットワークの拡大は，PCの利用幅を大きく広げる結果となり，PCでの作業と基幹システムで作成される全社レベルのデータとの連携が必要となってきた．

当社では，PC，UNIX，シリーズ2200，Aシリーズといった各プラットフォームを繋いだオンライン・トランザクション処理を実現するプロダクトとしてTransIT Open/OLTPを94年12月にリリースした．

このTransIT Open/OLTPプロダクトを使用して，メインフレーム上の情報系システムと使用者の操作するUNIXまたはPCの連動を考えてみることにする．

2. 現　　状

EUCと基幹システムの連携を考えるにあたって，まずPCの利用形態と基幹システムの現状から述べる．

2.1 PC利用形態の変化

もともと個人のワープロや表計算ソフトの範囲で使用されていたPCは，そのハードウェアおよびソフトウェア機能の発展とネットワーク技術の向上により，個人使用からネットワーク化へと段階を踏んで利用形態を広げていった．図1にPC接続の変遷を示す．

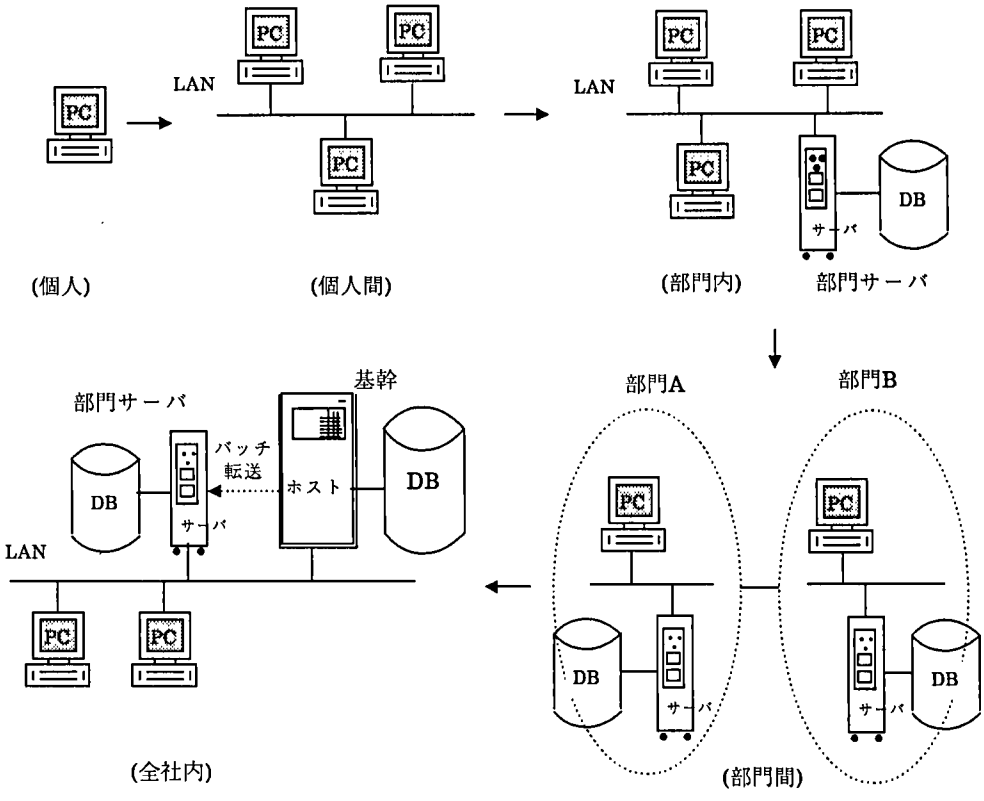


図1　PC接続の変遷

PC がスタンドアローンで個人使用されていた(個人). 個々の PC が LAN 接続されるとローカルなネットワークを形成しお互いが通信できるようになった(個人間). お互いが接続されると今まで個々で持っていたデータを共有するために部門データベースを使用するようになった(部門内). 部門ネットワーク同士の接続がなされ, 部門間にまたがるデータがアクセスできるようになった(部門間). ネットワークが拡大されることにより利用の幅も拡大され, できるだけ全社レベルのデータが必要となってきた. このため, 基幹データから抽出したデータを部門データベースへ定期的にバッチ処理で反映し, アクセスは部門データベースに対して行うようになった(全社内). この形態が定着するにつれてデータの新鮮さが問題となってきた. PC の使用に慣れたユーザが, 自ら全社レベルの情報を必要に応じてリアルタイムで取り出し加工したい, という要求が高まってきている.

2.2 中央集中型の基幹システム

基幹システムという言葉は, 企業あるいは団体が活動する上で最も重要なシステムを指すが, 各企業にとっては基幹システムが情報処理主体であったり, 事務処理主体であったりとさまざまである. 通常, 金融・証券業では勘定系と情報系システムの二系統立てで, 製造・流通業では, 基幹システム自身が業務系・情報系の区切りを持たない一つの大きなシステムで運用していることが多い.

70 年代から 80 年代前半にかけてのシステムのネットワークは, 汎用機を中心としてその下に数百, 数千というおびただしい台数の専用端末があって, PC はあくまでスタンドアローンとして切り離されて使用されていた. 80 年代後半になり, ホストコンピュータへのアクセス専用としての端末は PC にとって替わられ, PC 同士とホストは LAN 接続されるようになった. LAN 接続された PC はホストコンピュータの端末としてだけでなく, PC 機能の両方で使用できるようになった訳である. しかし, 基幹システムと PC 機能の間には相変わらず大きな壁が存在している. PC から基幹システムをアクセスするときにはエミュレータを起動してホストシステムの画面処理プログラムと会話しているケースが圧倒的に多い. 基幹システムのプログラムがすべての処理を一括して行っている形も変わってはいない. 図 2 に中央集中型システムの例を示す.

また, 図 2 のような中央集中型システムで使用者がデータを受け取るまでのプロセ

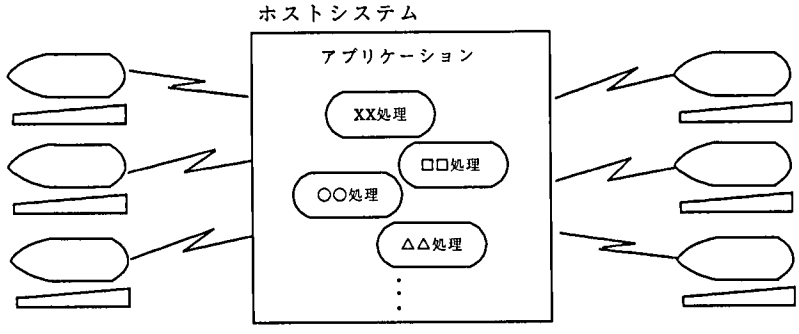


図 2　中央集中型システム

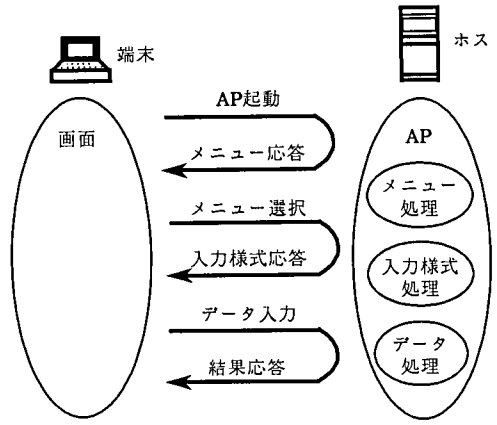


図 3　中央集中型システムのプロセス例

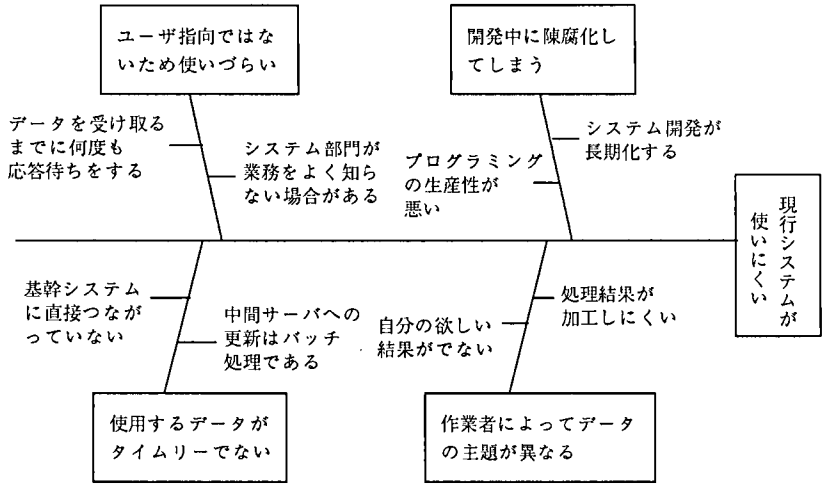


図 4　使用者の不満

スを見てみると，ホスト側がすべての処理を押さえているため，欲しいデータの送受信をするまでに複数回のトランザクションが発生し，使用者としては自分の要求するデータを受け取るまでに何回もホストからのレスポンス待ち状態に陥るという問題がある．図3に中央集中型システムのプロセス例を示す．

このような基幹システムの評価は，本来使用者の満足度によって決まるわけであるが，現行のシステムに不満をもつユーザは意外に多い．PC の使い勝手が良くなり，PC を身近に感じるようになっているので，余計にホストシステムの使いづらさに不満や不足を感じてしまう．使用者から見たホスト上での基幹システムの不満点をまとめると図4のようになる．

2.3　中央集中型からクライアント/サーバ型への移行

EUC の得意とする分野は情報加工である．一方，基幹システムでは情報加工(入口，出口処理)が必ずしも満足される状態ではない．前述の通り，PC と基幹システムの連携は必ずしもホスト側からの分散処理という観点からだけで語られるわけではなく，

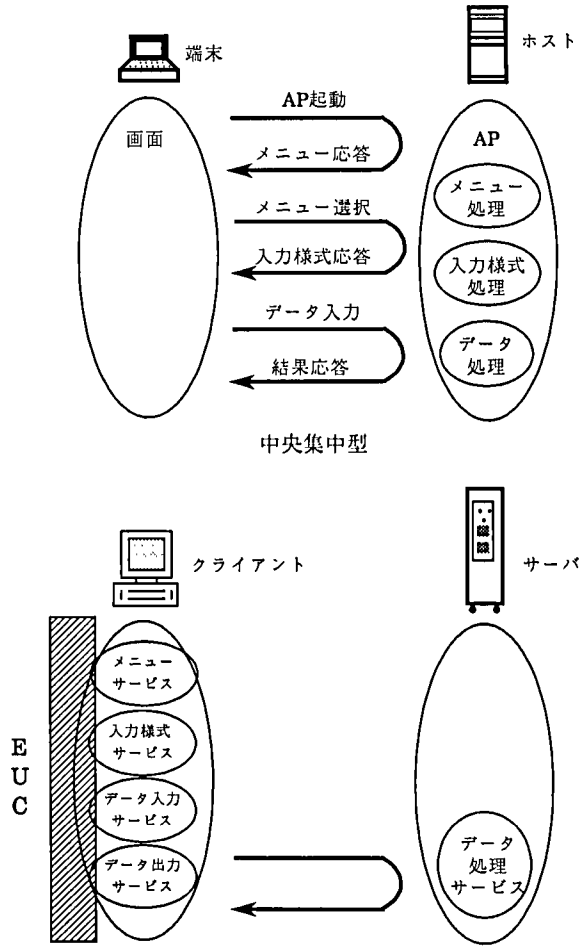


図 5　中央集中型とクライアント/サーバ型のプロセスの違い

PC を操作する作業者側からの使いやすさと最新データの必要性からも生まれるものである．PC の処理を中心に基幹システムとの連携を考えるとクライアント/サーバ型システムの良い点が見えてくるし，クライアント側に EUC を絡めていくと今までよりさらに柔軟性に富むシステム作りが期待できる．

また，データを受け取るまでのプロセスを見てもクライアント/サーバ型システムでは，ホスト上にあった処理の一部を PC（クライアント）側へ任せたために，データ処理要求を行うまでに一回のトランザクションで済むようになる．このため，PC での単独処理を行う部分が大幅に増えるのでレスポンスは安定し，さらに入出力処理に表計算プログラムやワープロソフトを絡めることができ，データ処理に幅が出てくる．図5に中央集中型とクライアント/サーバ型のプロセスの違いを示す．

3.　クライアント/サーバ・コンピューティング

クライアント/サーバ型システムを実現するためのいくつかの方法を以下に示す．

① RPC（リモート・プロシージャ・コール）スタイル：サーバ側で用意しているプロシージャをクライアント側から呼び出して実行する．

② RDA（リモート・データベース・アクセス）スタイル：サーバ側の共有データをクライアント側から活用するスタイル．SQL コマンド送信などもこのスタイルに含まれる．

③ TP モニタースタイル：クライアント側からメッセージをサーバ側へ送信しサーバ側でデータ処理などのカスタム・メイドの処理を行いメッセージをクライアント側へ返送する．

この中で，TP モニタースタイルの TransIT Open/OLTP プロダクトについて紹介する．

TransIT Open/OLTP は業界標準のクライアント/サーバ・アーキテクチャに基づき，部門処理から全社処理および企業間処理までを統合した水平・垂直クライアント/サーバ・システムを実現するソフトウェアである．

以下に，ソフトウェアの特徴を示す．

1) オープン・アーキテクチャの採用

X/Open DTP（Distributed Transaction Processing）モデルに準拠している．DTP モデルの利点は，そのコンポーネントおよびコンポーネント間のインタフェースが明確に規定されていることで，ユーザは DTP モデルに準拠したものの中から必要に応じて最適なコンポーネントを自由に選択・利用できる．また，DTP モデルの共通な API を使用していればアプリケーションの移植性も保障される．つまり，ユーザとしてはマルチベンダー/マルチプラットフォーム下で相互運用性と移植性の高い分散トランザクションが実現できる．

また，X/Open で定められている送受信のデータ型を使用して，構造化されたデータの整合性を保持することもできる．

図 6 に X/Open DTP モデルの構成要素とインタフェースを示す．

① AP：AP とは業務に応じた処理を行う構成要素である．トランザクションの開始/終了などの管理は TM に，トランザクションとして必要なリソースの操作は RM に，他のシステムとの通信は cRM にそれぞれ処理を要求し，応答を受け取ることによって一つのトランザクションの結果を得る．

② TM：TM はトランザクションの調整と制御を行う構成要素であり，これに必要な管理機能を提供する．たとえば，トランザクションが複数のリソースを使用する場合には，複数の RM 間に渡る整合性をとるための調整を行う．

③ RM：RM とはデータベース，ファイルなどのトランザクション処理に用いるリソースを管理する構成要素であり，AP に対してリソースへのアクセス手段を提供する．

④ cRM：cRM とは通信リソースを管理する RM であり，AP が他のトランザクション処理システムに存在する AP と連携（通信）して分散トランザクションを処理する場合に必要な構成要素であり，バッファやコネクションなどの通信リソース管理機能および通信プロトコルの制御機能を提供する．

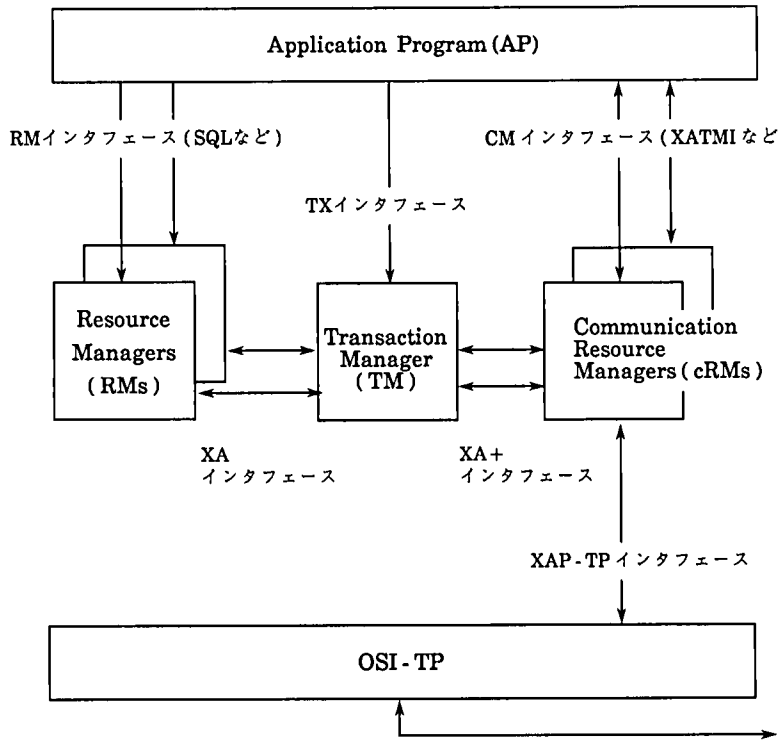


図 6　X/Open DTP モデルの構成要素とインタフェース

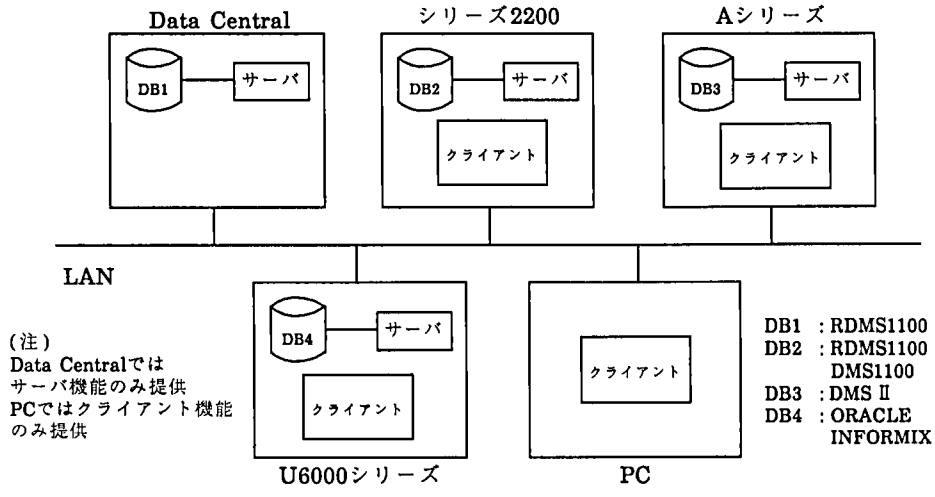


図 7　Open/OLTP におけるクライアント/サーバ構成

2)　メインフレームから PC/WS までを支援

シリーズ 2200，A シリーズ，UNIX，PC(MS-Windows，MS-Windows NT) など各プラットフォームで提供する．また，汎用機間あるいは汎用機と UNIX 機の間で双方向のクライアント/サーバ処理が可能である（図 7)．

3)　分散されたデータベースの整合性保証

2フェーズ・コミットメント機能を提供する．2フェーズ・コミットメント機能とは，複数のプログラムにより変更されたデータが，各自のデータベースに一連のトランザクションの結果として，同期をとって更新されるようにコミット処理を二つのフェーズに分けて行う処理である．2 PC と略すことが多い．これにより複数システムに存在する複数のリソース(DMS 1100/RDMS 1100，DMS II など)の更新処理を管理することが可能となり，障害発生時にも回復動作を行うことができる．

4) 管理機能

Open/OLTP プロダクトは U 6000 上で作動するネットワーク管理システム-CNMS（Common Network Management System）インタネット・マネージャと協調して，Open/OLTP ネットワークの管理機能を利用できる．つまり，各プラットフォームの Open/OLTP プロダクトは SNMP（Simple Network Management Protocol）エージェント機能を有し，CNMS インタネット・マネージャに対して情報を送信できる．

4. PC と基幹システムの連携

4.1 Open/OLTP と EUC の接点

PC 上では Open/OLTP TD（トランザクショナル・デスクトップ）が稼働する．ユーザは PC 上のクライアント・プログラムを開発する時，各種のプログラミング・ツールを使用する方法と，Windows アプリケーションを使用する方法のいずれかを選択することができる．

プログラミング・ツールを使用する場合は，Microsoft Visual C++，Borland C++, MicroFocus COBOL, Microsoft Visual Basic, Powersoft PowerBuilder などが使用でき，ユーザは XATMI インタフェースや TX インタフェースなどの X/Open の API を意識したプログラミングを行う必要がある．

一方，Windows アプリケーションを使用する場合は，このアプリケーションがダイナミックデータ交換（DDE: Dynamic Data Exchange）を使用していることが前提となる．DDE とは Windows 環境において複数アプリケーション間でデータ交換を行ったり，他のアプリケーションのコマンドを実行するためのプロトコルである．この DDE を使用している Windows アプリケーションには，MS-EXCEL，MS-Word，Visual Basic などがあり，DDE の呼び出し側を DDE クライアントと呼び，DDE を通してメッセージを受け取る側を DDE サーバと呼ぶ．つまり，DDE クライアントは MS-EXCEL や MS-Word のもとで作成した AP であり，DDE サーバは DDE インタフェースと Open/OLTP TD のインタフェースの両方を対応づける AP であるが，このうち Open/OLTP では DDE サーバを標準で提供している．つまり，ユーザはサーバとの通信に必要な X/Open の XATMI ファンクションや TX ファンクションを全て DDE サーバにまかせることができ，クライアント・プログラムの作りが簡単かつ明解なものになる．

たとえば，情報加工者が MS-EXCEL で作成した表の項目の中で，すべてのセルに

入る情報が部門情報からではなく，メインフレーム上の情報系から得たいとした時，表のデザインとデータ定義という通常の作業の他に，どのセル部分を入力としてどのセルに出力するかをMS-EXCELのマクロで作成するだけでよい．入力値はPOKE, EXECUTE（"DoService"）といったDDEメッセージの引き数にセットされDDEサーバに渡されるようにしておけば，DDEメッセージはDDEサーバによってX/OpenのAPIに変換され，別プラットフォームのサービスへと引き渡される．サービスからの戻り値もユーザが指定したセルにセットされてくる．受け取った結果はMS-EXCEL本来の機能を使って折れ線グラフや円グラフにすることはもちろんのこと，作業者のニーズにあわせた形へ自由に加工することができるし，ワープロへマージす

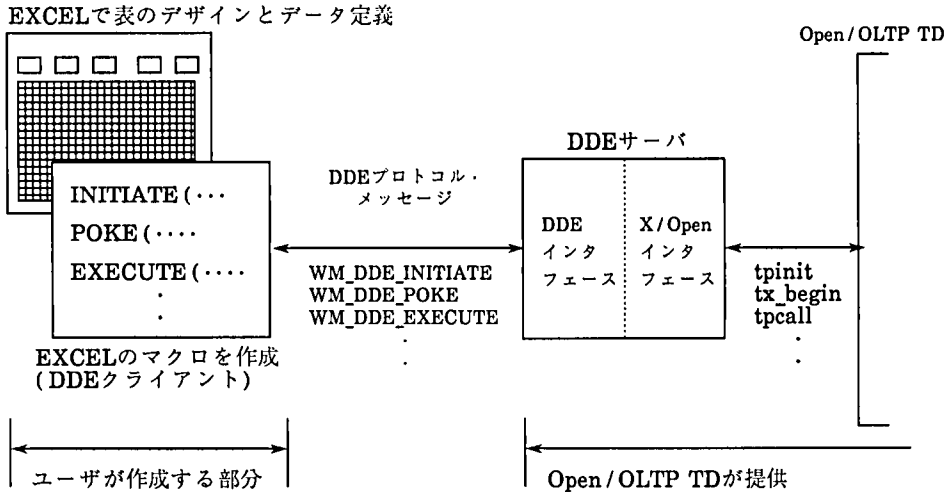


図8　EXCELとOpen/OLTP TDの連携

表1　EXCELで使用できるDDE用マクロ一覧

DDEのメッセージに対応するEXCELのマクロ関数	意　　味
INITIATE("DDEサーバ名",<アプリケーション名>)	DDE通信チャネルをオープンする。このメッセージ後、EXCELとDDEサーバが通信できる。
POKE("[コマンド:]<フィールド名>[(オカレンス)]",'<フィールド値>")	送信バッファにデータを蓄積する。コマンドにはADD,CHG, DEL, DELALLを指定し、バッファ内容を変更できる。
EXECUTE("BeginTrans:<タイムアウト値>")	グローバル・トランザクションを開始する。
EXECUTE("DoService[:<フラグ1>[+<フラグ2>]...]")	別プラットフォームのサービスを呼び出す。
EXECUTE("Clear")	EXECUTE("DoService")コマンドを呼び出した後、次のデータをPOKEする前に送信フィールドをクリアする。
EXECUTE("CommitTrans")	サービスの全ての作業をコミットする。
EXECUTE("AbortTrans")	サービスの全ての作業をアボートする。
REQUEST("<フィールド名>[(オカレンス)]")	サービスから返された応答を受け取る。
TERMINATE()	DDEサーバとのセッションを終了する。

ることも自由にできる．図8にMS-EXCELとOpen/OLTP TDの連携について示す．また，表1にはMS-EXCELで使用できるDDE用のマクロ一覧を示す．

4.2 基幹システムへのOpen/OLTPの適用

メインフレーム上の基幹データベースへはサーバ・プログラムによってアクセスするようになる．サーバ・プログラムは標準で提供されるサーバ・メイン・プログラム，ユーザによって作成されるサービス・ルーチンおよびユーザの必要に応じて独自処理を付加するための初期処理ルーチンと終了処理ルーチンで構成される．各サーバがアクセスするRM（リソース・マネージャ）はコンフィグ情報で関連づけられているため，サービス・ルーチンは具体的にデータベースへアクセスするための命令をSQL命令を主体としてプログラムを作成すれば良い．

また，業務系と情報系と2系統あるシステムでは，業務系と情報系の中継にOpen/OLTPを使用することができる．たとえば，業務系で更新処理が入ると同時にその処理を情報系へ反映させるために，Open/OLTPのもとで新たなトランザクションを発生させ2フェーズ・コミットメント処理で両データベースの同期をとるようにすると，常に情報系でも最新のデータを取り扱うことができる．業務系や情報系などの複数システムが，同一ホスト内に別アプリケーションとして存在する場合でも，Open/OLTPで連携することができる．これにより，複数システムにまたがるトランザクションの実行・管理をすることができる．

5. Open/OLTPを使用したシステムのリエンジニアリング

本稿では，情報系基幹システムとPCの連携を中心に述べてきたが，業務系自身のリエンジニアリングを考えた場合，以下の点がシステム作りのポイントとなりうる．

シリーズ2200の場合，従来のシステムはAIS/XISのもとで開発されたケースが非常に多い．AIS/XISの機能をプログラムから使用でき，かつX/Openに基づいたOpen/OLTPのインタフェースをとるように，これら二つのプロダクトの共存が今後の課題として残っている．この課題は，ユーザにおける開発コストの削減と開発期間の短縮に大きく影響する．

Aシリーズの場合，データベースの分割と統合にOpen/OLTPが有効である．たとえば，一つのデータベースを機能ごとに複数のデータベースに分割したり，別々のデータベースを統合したデータベース・システムとして再構築することが容易である．

6. おわりに

今まで中央集中型のシステムを作成する際にはウォータフォール型と呼ばれる手法が使用されてきた．つまり開発はある工程の反復を行わずに，計画→設計→開発→テスト→保守という一定の手順で行われていた．しかし，クライアント/サーバ型システムでは各開発手順の段階でプロトタイプを公開・評価し，繰り返し開発や手直しをするプロトタイピング手法が用いられることが多く，開発過程は使用者が中心となっている．こういった意味でも，これからはユーザ主体のシステムが主流となり，今後の基幹システムを構築するに当たっては，クライアント/サーバ型でという企業もますます増えるに違いない．

基幹システムとUNIX/PCネットワークの連携は，ファイル転送で部門データベースと基幹データベースの連携をすることもできるが，今回紹介したTransIT Open/OLTPプロダクトはタイムリーな最新情報が扱える点で有効な解決策の一つであり，メインフレームとUNIX機，PCとの効果的な共存を実現できるといえよう．

参考文献

［1］ Unisys, Unisys Open/OLTP Capabilitis Overview June 1994.
［2］ Unisys, Unisys Open/OLTP Object Managers Installation, Administration and Programming Guide June 1994.
［3］ X/Open Distributed Transaction Processing: Reference Model Version 2.
［4］ U 6000シリーズSYSTEM V.4 Open/OLTPトランザクション・デスクトップ導入・操作ガイド.
［5］ Microsoft Windows Version3.1 ソフトウェア開発キット　プログラミングガイド.
［6］ 奥井規晶他, クライアント/サーバソフトウェア開発, リックテレコム.
［7］ 岩宮好宏他, 分散トランザクション処理, リックテレコム.
［8］ 日経コンピュータ別冊, Windows用アプリケーション開発ツール, 1994.7.29.
［9］ 日経オープンシステム, 基幹システム設計, 1994.6.

執筆者紹介

佐　藤　友　彦（Tomohiko Sato）

昭和58年武蔵工業大学工学部 機械工学課卒業．同年日本ユニシス(株)入社．SNAコミュニケーション・ソフトウェアの開発・保守に従事．平成4年よりOpen/OLTP 2200の開発作業に従事．現在，ネットワーク技術本部 ネットワークソフトウェア一部WAN課所属．

久　米　進　也（Shinya Kume）

昭和58年関西学院大学理学部 物理学課卒業．同年日本ユニシス(株)入社．シリーズ2200・1100の基本ソフトウェア・リリース前検査業務に従事．平成4年よりOpen/OLTP 2200の開発作業に従事．現在，ネットワーク技術本部 ネットワークソフトウェア一部分散サービス課所属．

ネットワーク管理

Network Management

水 野 純 一

要 約 近年，ネットワークの経営インフラ化，高速化，LAN-WANインタネットワーク化が進展してきた．このような中で，経営インフラとしてのネットワークにマルチベンダの多様な情報機器（PC，ワークステーション，部門サーバなど）や通信機器（ルータ，ハブなど）が，エンドユーザ主導で接続/使用されるようになってきた．このようなマルチベンダ環境のネットワークを管理するには，多くのベンダに支持された標準的な管理フレームワークに基づく，統一的かつ拡張性のある手法を採用する必要がある．

本稿では，LANおよびLAN-WANインタネットワークにおいてこのような手法として普及している「インタネット標準管理フレームワーク」（SNMPv 1）の概要と，その実装例である「CNMSインタネット・マネージャ」を紹介する．次いで，このフレームワークを改良した「インタネット標準管理フレームワーク第2版」（SNMPv 2）のセキュリティの向上と管理プロトコルの改善につき記述する．最後に，管理対象の拡大に伴うネットワーク管理分野とシステム管理分野のオーバラップや，デスクトップ管理との関連について言及する．

Abstract The latest progress in computer networks is that they are becoming corporate-wide infrastructure, and are being changed into LAN-WAN internetworks. Reflecting this wave of trend, varieties of information processing machines (such as PCs, workstations and departmental servers) and network equipment (including routers and hubs), both supplied by multiple vendors, have come to be connected to those networks on end users' own initiative for their use. The successful management of such multi-vendor networks requires the adoption of an integrated, flexible approach based on a standard management framework supported by a large number of vendors.

This paper is intended to give an overview of the Internet standard management framework (SNMPv1), widely received as an approach for LANs and LAN-WAN internetworks, and discuss the CNMS Internet Manager, an implementation of the framework. The author, then, refers to the revised version (SNMPv2) focusing on two major enhancements in its security and management protocol. Finally described is the recent overlap, which has stemmed from expanded management areas, of network management and systems management as well as SNMP's relationship with desktop management.

1. はじめに

ビジネス・サイクル（企画～意思決定～投資～実現～回収）の短縮や，情報化投資の「適性化」などのニーズに対応して，汎用機のオープン化，クライアント・サーバ・システム化が進んできた．ネットワークについても経営インフラ化（個々の経営活動の共通基盤化），高速化，LAN-WANインタネットワーク化が進展してきた．このような中で，エンドユーザ・コンピューティングに典型的に見られるエンドユーザ（利用部門）の主体的参画に伴い，経営インフラとしてのネットワークにマルチベンダの

多様な情報機器（PC，ワークステーション，部門サーバなど）や通信機器（ルータ，ハブなど)が，エンドユーザ主導で接続/使用されるようになってきた．このようなマルチベンダ環境のネットワークを管理するには，多くのベンダに支持された標準的な管理フレームワークに基づく，統一的かつ拡張性のある手法を採用する必要がある．

本稿では，LANおよびLAN-WANインタネットワークにおいてマルチベンダのネットワーク管理技術として普及している「インタネット標準管理フレームワーク」（SNMPv 1）の概要と，その実装例である「CNMSインタネット・マネージャ」を紹介する．次いで，このフレームワークを改良した「インタネット標準管理フレームワーク第2版」（SNMPv 2）のセキュリティの向上と管理プロトコルの改善につき記述する．最後に，管理対象の拡大に伴うネットワーク管理分野とシステム管理分野のオーバラップについて言及する．

2. SNMPv1

SNMPv 1は，米国を中心とするインタネットを管理するために，その技術タスクフォース（IAB/IETF）の作業をもとに標準化された管理フレームワークである．その基本原則は，「管理対象ノードの特質の最大公約数的要素を考慮して，ノードに管理機能を追加することによる影響を最小限にする」[1]ことである．この基本原則にしたがって，①多様な管理対象の特質をなるべく共通化・抽象化してとらえ，②管理される機器は一般に低価格・小能力であることを前提にして，管理側（マネージャ）が主導権をもち，管理される側（エージェント）の負荷を最小にする，という考えをとっている．

SNMPv 1は，以下の四つのRFC（Request for Comments）文書により規定されている*．このフレームワークに基づくネットワーク管理システム（SNMPマネージャ）は，これらのRFCを実装している多様な機器（エージェント）をベンダを問わず管理することができる．

1) 管理情報構造と識別　：STD 16/RFC 1155[16]
2) コンサイスMIB（管理情報ベース）　：STD 16/RFC 1212[17]
3) 標準管理情報ベース（MIB-II）　：STD 17/RFC 1213[18]
4) シンプル・ネットワーク管理プロトコル（SNMP）　：STD 15/RFC 1157[19]

なお，管理API（アプリケーション・プログラミング・インタフェース），管理オペレータ・インタフェース（操作方法）については標準化の対象外である．

2.1 管理情報の定義

SNMPv 1に基づく管理システムはOSI管理に先んじてマルチベンダ管理を実現した．これには，インタネットを対象とした具体的な管理ニーズがあったために，管理情報を早期に標準化できたことが大きいものと考えられる．なお，管理情報は体系化されて管理情報ベース（MIB）と呼ばれる．

前述の四つのRFCのうち，1)RFC 1155は，管理情報の記述方法と使用できるデー

* 文献[2]ではRFC 1156（MIB-I）も含めているが，すでに「Historic」というカテゴリに属するため[3]，ここでは省略した．なお，表2に示すSNMPv 2のRFCでは，RFC 1213（MIB-II）をSNMPv 1に含めていない．

タ型などを規定している．また，2) RFC 1212 はテーブル形式の管理情報の一つの行（インスタンス）を識別するための情報の記述法などを規定している．一方，3) RFC 1213 は，これらを用いて TCP/IP ネットワークの標準管理情報を MIB-II として規定している．MIB-II はインタフェース，IP，ICMP，TCP，UDP および SNMP などのプロトコルごとの管理情報と，システムの管理情報からなる（表 1）．なお，MIB-II のほかにも，ハブ（リピータ），FDDI，RMON（Remote Network Monitoring）をはじめ多くの標準 MIB が，1) および 2) の規定に基づいて定義されている．

表 1　標準 MIB-II の管理情報概要

グループ	内　　容
system	システムに関する管理情報． システムの記述，初期化されてからの時間，管理者の名前，連絡先，システムの設置場所，など．
interfaces	インタフェースに関する管理情報． インタフェースの数，種別，物理アドレス，インタフェースの状態，送／受信オクテット総数，廃棄パケット数など．
at	アドレス変換テーブルに関する管理情報． 物理アドレス，ネットワーク・アドレス，など．
ip	ＩＰ（インタネット・プロトコル）に関する管理情報． 送／受信データグラム総数，廃棄データグラム数，ＩＰアドレス・テーブル，ＩＰルーティング・テーブル，など．
icmp	ＩＣＭＰ（インタネット制御メッセージ・プロトコル）に関する管理情報． 送／受信ＩＣＭＰメッセージ総数，送／受信したＩＣＭＰエラー・メッセージ総数，など．
tcp	ＴＣＰ（転送制御プロトコル）に関する管理情報． ＴＣＰコネクションの最大数，送／受信ＴＣＰセグメント総数，ＴＣＰコネクション・テーブルなど．
udp	ＵＤＰ（ユーザ・データグラム・プロトコル）に関する管理情報． 送／受信ＵＤＰデータグラム総数，エラーとなった受信データグラム数，ＵＤＰテーブル，など．
egp	ＥＧＰ（外部ゲートウェイ・プロトコル）に関する管理情報． 送／受信ＥＧＰメッセージ数，ＥＧＰネイバ・テーブルなど．
snmp	ＳＮＭＰ（シンプルネットワーク管理プロトコル）に関する管理情報． 送／受信ＳＮＭＰメッセージ総数，エラー・タイプ毎の送／受信メッセージ数，コマンド毎の送／受信メッセージ数，など．

2.2　SNMP 管理プロトコル

RFC 1157 が規定する SNMP（Simple Network Management Protocol）は，マネージャがエージェントに送信する①取得要求（GetRequest），②設定要求（SetRequest），③次の管理情報の取得要求（GetNextRequest）と，エージェントがマネージャに送信する，④応答（すべて GetResponse と呼ぶ），⑤事象通知（トラップ：Trap）からなる管理プロトコルである．エージェントの負荷を小さくするためやメッセージ

が消失する可能性から，トラップの多用は避けるべきこととされており，標準トラップの種類も6種類と少ない．

SNMPプロトコルは実装が容易なため，ルータ，ワークステーション，PC，ハブ，など多くの機器にSNMPエージェントが実装されている．

2.3 管理フレームワークの拡張

SNMPv1フレームワークは標準MIBや標準トラップによる管理機能に加えて，機器ベンダが独自の（被）管理機能を提供する手段も提供している．独自の管理情報をMIBの形に整理したものを「拡張MIB」と呼ぶ．また，独自のトラップを「ベンダ固有トラップ」と呼ぶ．これらは標準MIBおよび標準トラップとほぼ同一の扱いができるため，SNMPv1に基づく管理システムは，拡張MIBや固有トラップにも柔軟に対応できるものが多い．

2.4 SNMPv1とマルチベンダ管理

SNMPv1の標準化により，SNMPエージェントの実装されているマルチベンダの通信ノード，情報ノード，ネットワーク・セグメント（RMON使用）を統一的に管理できるようになった．また，SNMPエージェントを実装していない機器であっても，エコー要求に対して応答するものであれば，稼働/停止の別を推定することができる．

将来的にはOSI管理によるマルチベンダ統合管理が実用化されようが，安価な通信機器などの管理には，SNMP管理が今後も残ると考えられる．

3. 実装例：CNMSインタネット・マネージャ

当社が提供する「CNMSインタネット・マネージャ」（CNMS/IM）は，SNMPv1およびSNMPv2に基づくネットワーク管理システム（マネージャ）で，TCP/IPベースのネットワークの構成管理，障害管理，性能管理，アラーム管理，レポート作成などを支援する．

3.1 分散管理アーキテクチャ

SNMPv1はマネージャからエージェントへのポーリングによる情報収集を主体とする管理フレームワークであるため，管理対象が増えると次のような問題が起きてくる．

① 管理マネージャの負荷が大きくなる．
② 状態監視の周期が長くなり，タイムリに監視できない．
③ 管理のための通信量が増えて，業務通信が妨げられる．
④ WAN接続（ISDN網サービスなど）部分の通信料金が増える．

このような問題に対して，CNMS/IMでは「分散管理アーキテクチャ」を採用して，マネージャの機能を「管理ステーション」と「管理デーモン」の二つに分け，それぞれを一つまたは複数組み合わせて大規模なネットワークでも管理できるようにしている．

1) 管理ステーション……CNMS/IMの管理サービスの中心で，ネットワーク・マップ（3.3節）を表示したり，他の管理アプリケーションを起動するなど，管理オペレータとのインタフェースを提供する．

2) 管理デーモン……バックグラウンドで実行されるプロセスで，状態監視，接続

機器の検出（3.2節），アラーム管理（3.5節）などを行う．状態の変化，未登録機器の検出，アラーム（あらかじめ設定された事象）の発生などを管理ステーションや他の管理デーモンに通知する．この通知にはTCPプロトコルを用いて信頼性を高めている．

複数の管理デーモンを階層構成することにより，管理対象へのポーリング負荷や管理通信量を各管理領域に局所化しながら，ネットワーク管理センタの管理ステーションでネットワーク全体を一元管理することができる（図1）．

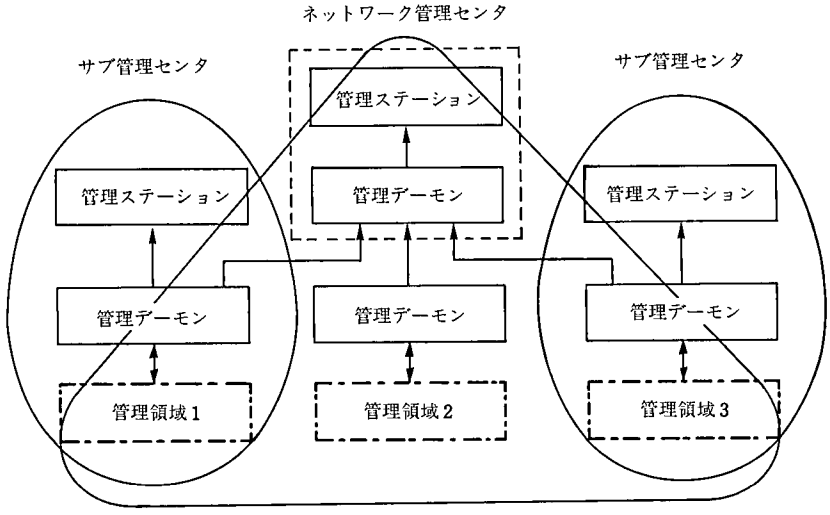


図1 階 層 構 成

この場合，サブ管理センタにも管理ステーションを置いて，一部の管理領域を管理する形態もある．また，複数の管理ステーションが相互に補完しながら各管理領域を管理する形態をとることもできる（図2）．

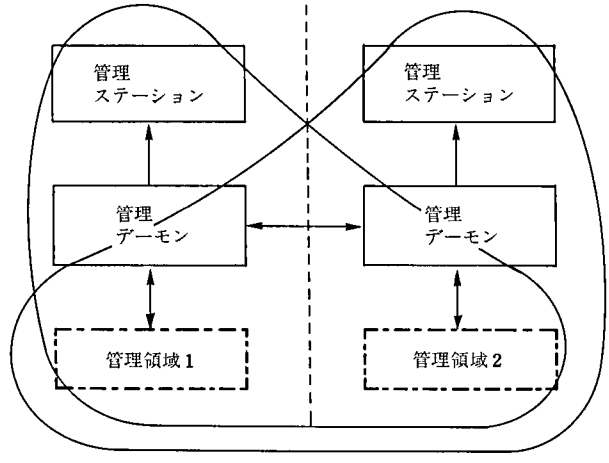


図2 マネージャ・マネージャ構成

3.2 接続機器の検出

指定されたネットワークに接続されている，CNMS/IMに未登録の機器を検出する

機能で，対象となる機器はICMPのエコー応答機能をもてばよく，SNMPエージェントをもつ必要はない．ネットワーク・マップの作成や，不正に接続された機器を見つけるのに役立つ．

3.3 ネットワーク・マップの自動作成

ネットワーク・マップは，ルータ，ホストなどの機器や，イーサネット，トークンリング，広域ネットワーク（WAN）などのネットワークを階層的に図示する（図3）．機器の状態が変化したりアラームが発生すると，対応するアイコン色の変化などで管理オペレータに通知する．管理デーモンが検出して登録した機器情報を元にネットワーク・マップを自動作成することができる．また，マップ・エディタを使って作成・編集することもできる．

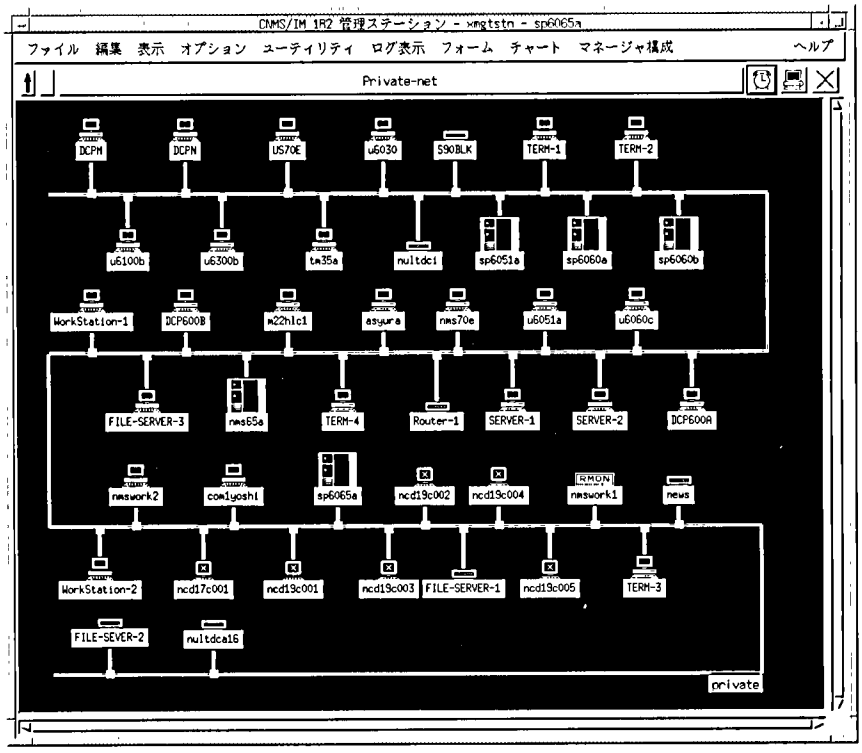


図 3　ネットワーク・マップの例

3.4 管理情報の表示/設定

MIBの階層構造（例えば，MIB-IIの管理情報の一つであるインタフェース状態：ifOperStatusは，iso → org → dod → internet → mgmt → mib - 2 → interfaces → ifTable → ifEntry → ifOperStatusの階層構造により定義される．）に従って管理情報を表示/設定したり，様々な種類の表形式の画面（フォーム：図4）を使って管理情報を表示/設定することができる．CNMS/IMが標準提供しているフォームのほかに，フォーム作成ツールを使用して変更・作成したフォームも使用できる．

また，MIBに定義された管理情報やそれらを使った計算式の値をグラフ形式（チャート：図5）で表示することもできるので，ネットワークの状況分析や将来計画の検討

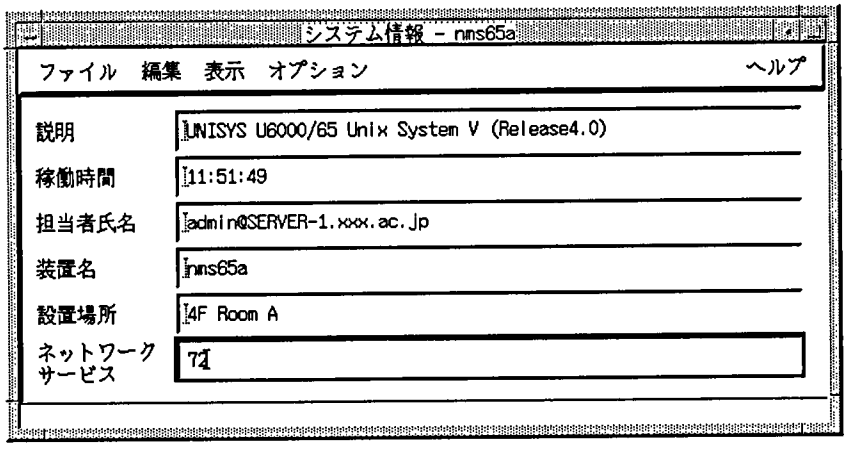


図 4 フォームの表示例

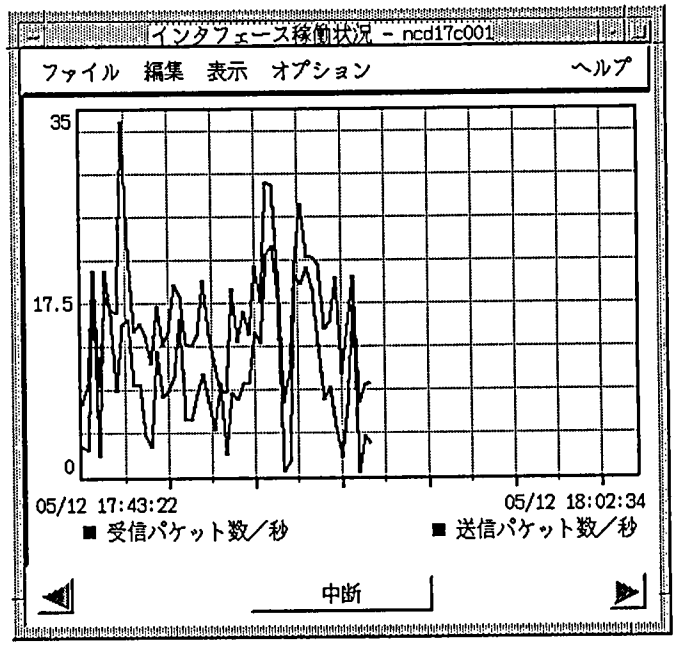


図 5 チャートの表示例

に役立つ．CNMS/IM が標準提供しているチャートのほかに，チャート作成ツールを使用して作成・変更したチャートも使用できる．

3.5 アラーム管理

CNMS/IM では，ネットワークに発生する様々な事象や状況のうちでユーザに通知する必要のあるものをアラームと呼ぶ．機器状態の変化など設定済みのアラームのほかに「エキスパート・システム」と呼ぶツールを使用して，アラームとそれが発生した時に取るべきアクション（コマンド/シェル・スクリプトの実行，ログ，ネットワーク・マップへの通知など）を設定することができる．

3.6 MIBコンパイラ

CNMS/IMは80種類以上の標準/拡張MIBを登録済みであるが，そのほかのMIBもMIBコンパイラを使って登録できる．これにより，エンドユーザが採用した機器に固有な管理情報にも対応できるし，今後定義されるMIBを実装した新しいタイプの機器が現れても管理対象に追加できる．CNMS/IMのMIBコンパイラは，SNMPv 1のRFC 1212およびRFC 1155(2.1節)に基づいて記述されたMIB文書ファイルを入力とする．また，SNMPv 2のMIB記述マクロにも一部を除いて対応している．

3.7 カスタマイゼーション

フォーム作成ツール，チャート作成ツール，「エキスパート・システム」，レポート作成ツールに加えて，メニューの追加/変更，ネットワーク・マップの編集（例，表示色，背景，アイコンの種類，管理対象機器の追加/削除，リングの大きさの変更）などが簡単にできる．また，Oracleデータベースや管理API（アプリケーション・プログラミング・インタフェース）を用いたカスタマイゼーションも可能である．

4. SNMPv2

SNMPv 1に基づくネットワーク管理には次のような制約があった．

① セキュリティが弱い．

② 大きなテーブル形式の管理情報をアクセスするための通信の効率が悪い．

③ マネージャ間の通信プロトコルがない．

表2に示すRFCから構成されるSNMPv 2管理フレームワークにおいては，これらの制約も含めて改善が図られている．

表2　SNMPv2のRFC[4]~[15]

番号	表題	備考
1441	Introduction to version 2 of the Internet-standard Framework	SNMPv2フレームワークの紹介．
1442	Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)	MIBを記述するためのマクロ，有用なデータ型の定義，適合性（コンフォーマンス）要件の記述法などの規定．
1443	Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)	
1444	Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)	
1445	Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)	パーティと呼ばれる仮想的な実行環境や認証プロトコル／機密プロトコルなど，運用に関する規定．
1446	Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)	
1447	Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)	
1448	Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)	SNMP管理プロトコルと様々なトランスポート層へのマッピング，SNMP管理プロトコルに関連するMIBに関する規定．
1449	Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)	
1450	Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)	
1451	Manager-to-Manager Management Information Base	
1452	Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework	SNMPv1とSNMPv2管理フレームワークの共存に関する規定．

4.1 管理プロトコルの改善

SNMPv 2の管理プロトコルでは，テーブル情報などの効率的取得のためのGetBulkRequestと，マネージャ間通知要求用のInformRequestが追加された（表3）．

表 3 SNMPv1 と SNMPv2 の管理プロトコルの比較

SNMPv1	SNMPv2	備　　考
GetRequest	GetRequest	マネージャからの情報取得要求
GetNextRequest	GetNextRequest	マネージャからの次情報取得要求
SetRequest	SetRequest	マネージャからの情報設定要求
GetResponse	Response	エージェントまたはInformRequestを受けたマネージャからの応答送出
Trap	SNMPv2-Trap	エージェントからの通知
	GetBulkRequest	マネージャからのテーブル情報などの効率的取得要求
	InformRequest	マネージャから他のマネージャへの通知要求

4.1.1 GetBulkRequest[11]

SNMPv 1 でテーブル形式の管理情報を取得するには，1 行分の管理情報を GetNextRequest により要求してその応答 (GetResponse) を待つことを，必要な行数分繰り返さなければならなかった．たとえば，100 個のエントリをもつテーブルでは GetNextRequest と GetResponse がそれぞれ 101 回必要となる（マネージャは 101 回目の応答でこのテーブルのエントリがもうないことを知り，取得をやめる)．

これに対して SNMPv 2 の GetBulkRequest では，最大繰り返し回数(max-repetitions) フィールドに指定した行数分の管理情報が一つの応答 (Response) により返される（正常時)．したがって，同じ 100 個のエントリをもつテーブルに対して max-repetitions を 5 とした GetBulkRequest を使えば，GetBulkRequest と Response をそれぞれ 21 回出せばよいことになる（マネージャは 21 回目の応答でテーブルの終りを知り，取得をやめる)．

4.1.2 InformRequest[11]

InformRequest は，マネージャが他のマネージャに情報を通知するのに用いられる．その形式はエージェントがマネージャに通知する SNMPv 2-Trap と同じである．

InformRequest を受信したマネージャは所定の検査を行った上で，Response を作成して InformRequest の送信元に応答するように規定されている．したがって，InformRequest の送信元は Response を受信することで送達確認ができるようになっている（SNMPv 2-Trap では送達確認はできない)．

4.2 セキュリティの向上

SNMPv 1 ではセキュリティに関係する次の事項を識別するのに「コミュニティ名」という文字列（例，"public"）を用いた[20]．

① 管理操作を要求できるエンティティか？

② 許される管理操作か？

③ 管理操作の対象とできる管理情報か？

コミュニティ名は文字列のままマネージャとエージェント間で通信されるので，コミュニティ名が解読されて悪用される危険がある．また，管理操作の要求元の認証が十分得られないため，SNMPによる管理情報の更新機能をエージェントに実装しない，という悪影響も指摘されている[20]．

SNMPv 2の運用モデル（administrative model）[8]では，SNMP管理プロトコルのデータを認証プロトコルや機密（暗号）プロトコルで包み込む方法や，これらのプロトコルの種類，公開鍵，秘密鍵，認証クロックやトランスポート層関連情報などの仮想実行環境である「パーティ」というデータ構造を規定している．コミュニティ名の役割であった他の二つの識別，すなわち許される管理操作の識別には「アクセス・ポリシー」が，また管理操作の対象となる管理情報の集合の識別には「SNMPv 2コンテキスト」および「MIBビュー」が用いられる．パーティ，アクセス・ポリシー，SNMPv 2コンテキストおよびMIBビューの形式は，パーティMIBにより規定される[10]．

4.2.1 セキュリティの目標[9]

ネットワーク・プロトコルの脅威には，①情報の改変，②偽装（masquerade），③メッセージ順序の改変（遅延，再送），④開示（disclosure）などがあるが，SNMPv 2では認証プロトコルにより①と②を防ぎ，その上で機密プロトコルにより④を防ぐ．③については認証プロトコルにより一部を防ぐ．

4.2.2 SNMPv2メッセージの形式

SNMPv 2ではパーティ，SNMPv 2コンテキスト，認証プロトコル，機密プロトコルなどを支援するために，管理メッセージの形式が変更された（図6）．PDUs型で定義されるGetRequestなどの情報にパーティ識別子やコンテキスト識別子が付加されてSNMP管理情報となり，これに認証プロトコルの情報が付加されてSNMP認証メッセージとなる．これを機密プロトコルで暗号化して，機密宛先パーティ識別子を付加したものがSNMP機密メッセージとなる．

SNMPv 2メッセージはSNMPv 1メッセージと形式が異なることや，未だProposed Standardという標準化段階にある（Draft Standardを経てからFull Standardに進む必要がある）ことから，普及には今後数年かかるものと考えられる．

4.2.3 ダイジェスト認証プロトコル[9]

送信側と受信側だけが知っている認証秘密鍵を使って送信側がメッセージ（符号化済み）の一部からダイジェストを計算し，これをメッセージの中のauthDigestに入れて送信する．受信側は受信したメッセージのauthDigestに入ったダイジェストを保存した上で，共通の秘密鍵を使ってメッセージの一部からダイジェストを計算する．このダイジェストが保存したダイジェストと一致すれば，受信したメッセージの送信元の認証が確認され，情報も改変されていないことになる．ダイジェストの計算にはMD 5アルゴリズムが選択されている．このアルゴリズムを用いたダイジェスト認証プロトコルは，v 2 md 5 AuthProtocolというオブジェクト識別子により識別される．この場合には，ダイジェストも認証秘密鍵も16オクテットの長さとなる．

4.2.4 対称機密プロトコル[9]

ダイジェスト認証プロトコルに基づくメッセージ（符号化済み）に対して，受信側

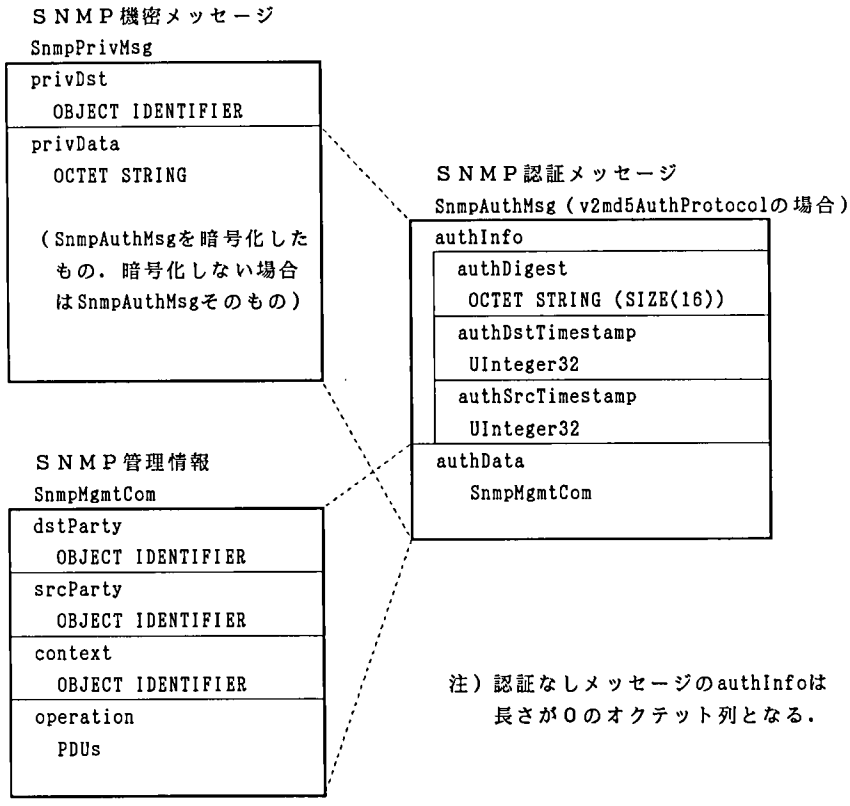


図 6　SNMPv2 メッセージの形式

パーティの機密秘密鍵を用いて送信側が暗号化を行う．これを privData にセットしてから snmpPrivMsg 全体を符号化して送信する．受信側では自分のパーティの機密秘密鍵を用いて受信メッセージ(復号化済み)の privData 部分を平文化する．暗号化/平文化のアルゴリズムには DES (Data Encryption Standard) のサイファ・ブロック・チェイニング・モード（いったん暗号化したメッセージと次のブロックの平文を加えて次の暗号化のための入力をする方法[21]）が選択されている．

4.3　実装例：CNMS インタネット・マネージャ

CNMS インタネット・マネージャ (CNMS/IM) は SNMPv 1 に加えて，SNMPv 2 の MD 5 ダイジェスト認証プロトコルや，GetBulkRequest も含む SNMPv 2 形式の管理メッセージを実装している．一方，次のものは実装していない．

① 機密プロトコル
② パーティのリモート作成
③ Manager-to-Manager MIB および InformRequest
④ SNMPv 2-Trap
⑤ 一部の MIB 定義用マクロ

CNMS/IM は SNMPv 2 を支援しているエージェントには，(設定に応じて) SNMPv 2 形式の管理メッセージを送信する．また，管理デーモンのもつ管理情報(管理対象機器テーブルなど) を管理ステーションから取得してフォーム (3.4 節) に表示

する場合に，GetNextRequest ではなく GetBulkRequest を使用して，管理メッセージ数を減らしている．

5. 管理対象の拡大——システム管理とのオーバラップ

標準 MIB の中で Full Standard になっているのは，MIB-II と Ethernet-like MIB[2]の二つしかない[3]．しかし，MIB を定義した RFC は近年増加している（表 4，図 7）．この中にはホストリソース MIB（RFC 1514），ネットワークサービス・モニタリング MIB(RFC 1565)，メール・モニタリング MIB(RFC 1566)，X.500 ディレクトリ MIB（RFC 1567），RDBMS MIB（RFC 1697）など，通信ノードから情報ノードに管理対象を広げたものが含まれている．これらは，ネットワーク管理とシステム管理がオーバーラップしている領域に属すると言える．

一例としてホストリソース MIB では，PC やワークステーションを含む様々なアー

表 4　MIB 関連の RFC（1500 以降）1995/1/4 現在

番号	状態	管　理　対　象
1749	DS	IEEE 802.5 Station Source Routing
1748	DS	IEEE 802.5
1724	DS	RIP Version 2
1697	PS	RDBMS
1696	PS	Modem
1695	PS	ATM Management Version 8.0
1694	DS	SMDS Interfaces
1666	PS	SNA NAUs
1660	DS	Parallel-printer-like Hardware Devices
1659	DS	RS-232-like Hardware Devices
1658	DS	Character Stream Devices
1657	PS	Border Gateway Protocol (BGP-4)
1650	PS	Ethernet-like Interface Types
1643	S	Ethernet-like Interface Types
1628	PS	UPS
1612	PS	DNS Resolver
1611	PS	DNS Server
1604	PS	Frame Relay Service
1595	PS	SONET/SDH Interface Type
1593	I	SNA APPN Node
1573	PS	Evolution of the Interfaces Group of MIB-II
1567	PS	X.500 Directory Monitoring
1566	PS	Mail Monitoring
1565	PS	Network Services Monitoring
1559	DS	DECnet Phase IV
1525	PS	Source Routing Bridges
1516	DS	IEEE 802.3 Repeater Devices
1515	PS	IEEE 802.3 Medium Attachment Units (MAUs)
1514	PS	Host Resources
1513	PS	Token Ring Extensions to the Remote Network Monitoring
1512	PS	FDDI

S: Full Standard　DS: Draft Standard　PS: Proposed Standard　I: Information

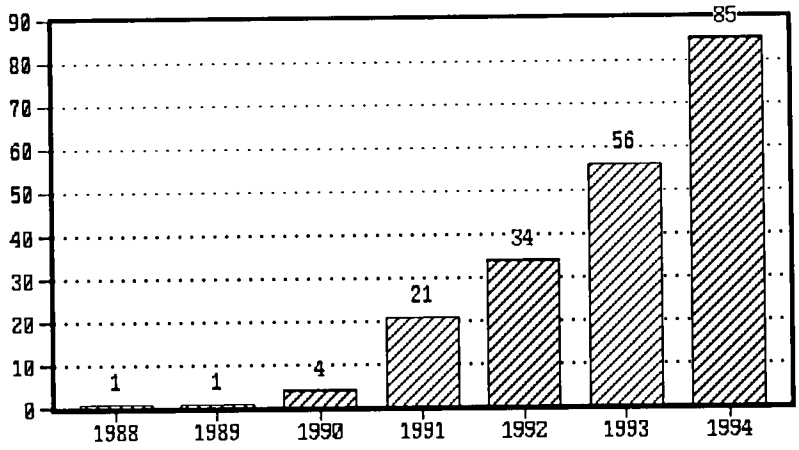


図 7　MIB 関連 RFC 数（累積）*

表 5　ホストリソース MIB の管理情報概要

グループ	内　　容
system	コンピュータ・システムに関する情報 初期化されてからの時間，現在の日付・時刻，初期化用装置，現在のユーザ数，現在のプロセス数，など．
storage	記憶領域（ストレージ）に関する情報 メモリの大きさ，各ストレージの種類・割当て単位・大きさ・使用量・割当できなかった回数，など．
device	装置に関する共通情報，CPU・プリンタなどの情報 各装置の種類・記述・状態・エラーの回数，CPU負荷，ネットワーク機器のインタフェース番号，プリンタの状態・検出したエラーの種類，ディスク装置・パーティション・ファイルシステムの情報，など．
running software	ロード済・実行中のソフトウェアに関する情報 ソフトウェアの記述，種類，状態，など．
running software performance	ロード済・実行中のソフトウェアの性能に関する情報 CPU時間，メモリ使用量．
installed software	インストール済のソフトウェアに関する情報 ソフトウェアの記述，インストール日時，など．

キテクチャのコンピュータに共通な資源（周辺機器など）の管理情報を規定している(表 5)[22]．情報ノードを SNMP で管理するためには，拡張 MIB をベンダが定義してエージェントに実装することが従来から行われている．たとえば日本ユニシスが提供する U 6000 サーバについては，①エージェント構成情報，②プロセス・テーブル情報，③シグナル・テーブル情報，④ファイルシステム・テーブル情報，⑤スワップ情報，

* RFC INDEX (1/4/1995) から「MIB」「Objects」「Managed」「Base」「Management」のいずれかの語を含む RFC を抽出し，MIB に関係ないものを除外した．他の RFC で置き換えられたものも含む．

⑥メッセージ・キュー・テーブル情報，⑦共用メモリ・テーブル，⑧セマフォ・テーブル情報が定義されている．また，他の例ではベンダ独自の管理フレームワークに基づく「エージェント」を使うこともある．しかし，ホストリソースMIBをはじめとする情報ノード対象のMIBが標準化されて実装が普及すれば，通信ノードと比較して使われ方が多様な情報ノードも，管理ニーズに合わせて様々な観点から「ネットワーク」管理システムにより統一的に管理することができることになる．

6. おわりに

本稿では，LANおよびLAN-WANインタネットワークにおいてマルチベンダのネットワーク管理技術として普及している「インタネット標準管理フレームワーク」(SNMPv 1) の概要と，その実装例である「CNMSインタネット・マネージャ」を紹介した．また，SNMPv 1の弱点を改良した「インタネット標準管理フレームワーク第2版」(SNMPv 2)のセキュリティの向上とGetBulkRequestなど管理プロトコルの改善につき記述した．最後に，管理対象の拡大に伴うネットワーク管理分野とシステム管理分野のオーバラップについても言及した．

マルチベンダの多様な機器からなるネットワークの管理には，今後もSNMPによる管理が広く使われよう．一方，PCの高機能化，ネットワーク化につれてデスクトップ管理としてDMI (Desktop Management Interface) [23]に基づく管理も次第に広まることが予想される．両者の統合の一つの形態として，デスクトップ管理とSNMP管理をプロキシ・エージェントにより媒介させる形態がある (図8)．CNMS/IMなどSNMPベースの管理システムはマルチベンダ管理を前提にしているため，このような

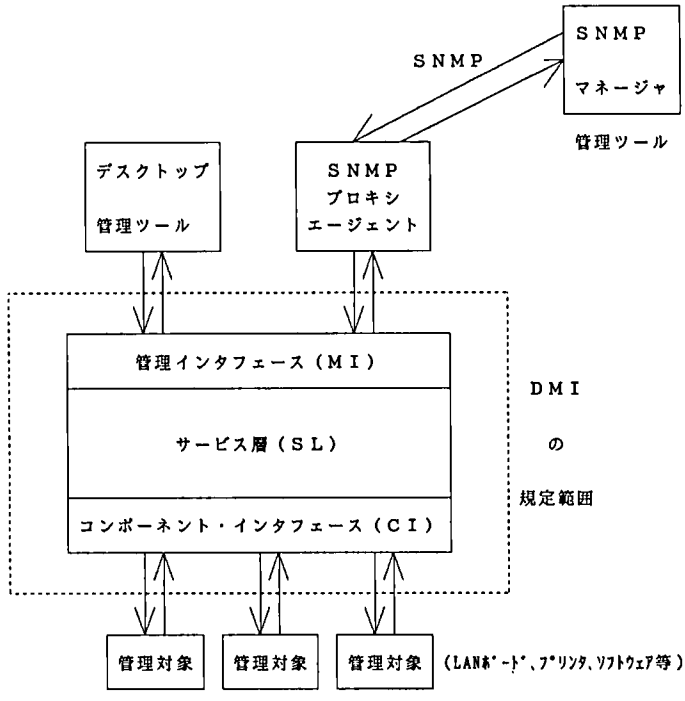


図 8　デスクトップ管理とSNMP管理の連携

プロキシ・エージェントがどのベンダから提供されても，これを介してデスクトップ管理と連携することが可能になるものと考える．

参考文献 [1] M. T. ローズ (西田竹志訳), TCP/IP ネットワーク管理入門－実用的な管理をめざして, トッパン, 1992 年 8 月, p. 66.
[2] F. Kastenholz, "Definition of Managed Objects for the Ethernet-like Interface Types", STD 50, RFC 1643, FTP Software, Inc., July 1994.
[3] J. Postel,"INTERNET OFFICIAL PROTOCOL STANDARDS", STD 1, RFC 1720, Internet Architecture Board, November 1994.
[4] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Introduction to version 2 of the Internet-standard Network Management Framework", RFC 1441, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[5] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv 2)", RFC 1442, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[6] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Textual Conventions for version 2 of the the Simple Network Management Protocol (SNMPv 2)", RFC 1443, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[7] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Conformance Statements for version 2 of the the Simple Network Management Protocol (SNMPv2)", RFC 1444, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[8] J. Galvin and K. McCloghrie, "Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1445, Trusted Information Systems, Hughes LAN Systems, April 1993.
[9] J. Galvin and K. McCloghrie, "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2) ", RFC 1446, Trusted Information Systems, Hughes LAN Systems, April 1993.
[10] K. McCloghrie and J. Galvin, "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv 2)", RFC 1447, Hughes LAN Systems, Trusted Information Systems, April 1993.
[11] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1448, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[12] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1449, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[13] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1450, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[14] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Manager-to-Manager Management Information Base", RFC 1451, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[15] J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework", RFC 1452, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
[16] M. Rose and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", STD 16, RFC 1155, Performance Systems

International, Hughes LAN Systems, May 1990.
[17] M. Rose and K. McCloghrie, Editors, "Concise MIB Definitions", STD 16, RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.
[18] K. McCloghrie and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.
[19] J. Case, M. Fedor, M. Schoffstall and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.
[20] M. Rose, *The Simple Book : An Introduction to Internet Management*, 2nd edition, P T R Prentice Hall, Englewood Cliffs, New Jersey 07632, 1994.
[21] 石崎純夫監修, 上園忠弘, 小林孝夫, 山本明知, 情報システムのセキュリティコントロール, オーム社, 1988年6月.
[22] P. Grillo and S. Waldbusser, "Host Resource MIB", RFC 1514, Network Innovations, Intel Corporation, Carnegie Mellon University, September, 1993.
[23] "Desktop Management Interface Specification, Version 1.0", Desktop Management Task Force, April 1994.

執筆者紹介　水　野　純　一 (Jun-ichi Mizuno)

1951年生. 1976年東京大学大学院工学系研究科修士課程修了. 同年日本ユニシス(株)入社. 基本ソフトウェアの開発・保守を担当後, ネットワーク管理システム (TNAS, CNMS インタネット・マネージャ) の日本化・提供に従事. 現在, ネットワークソフトウェア一部に所属. 情報処理学会会員.

DW/Power Client Edition

1. はじめに

クライアント/サーバ・コンピューティングは，アプリケーションを複数のコンピュータ上で協調して実行する方式の一形態である．今日，エンドユーザはデータベースなどのリソース共有を部門内だけでなく，他部門や他企業に対しても望むようになっている．部門外で従来から稼働しているメインフレーム・コンピュータ・システム上の企業データベースを情報系システムとしてアクセスする必要性は益々高まっている．また基幹システムとしても，クライアント/サーバ・コンピューティング・モデルを導入する機運が高まっている．こうした中で，旧来のシステムからクライアント/サーバ環境への移行を支援する見通しの良い簡便なツールが求められている．

1993年6月に2200シリーズ，Aシリーズおよび U 6000シリーズ向けにリリースされたデザイナ・ワークベンチ（DW）2R1はその後1994年11月に3R1にレベルアップされた．これはクライアント/サーバ環境へ橋渡しをする最初の製品である．LINC, MAPPERまたは3GL言語で作成されたホスト上のアプリケーションを全く変更することなしに，クライアントであるPC側の“ルックアンドフィール”をマイクロソフト社のWindowsに対応したグラフィカルな表現に簡単に変更できる．

さらに，ここに紹介するDWPCE（DW/Power Client Edition）は，クライアント側にPower Builder（PB）やVisualBasic（VB）を用いてローカル・ロジックの組込みを可能にする次世代版デザイナ・ワークベンチである．

アプリケーションの開発者は従来のDWが提供するフォーム・デザイナの代わりにPBやVBのGUIスクリーン・ペインタを使用して，クライアント側のフォーム定義とローカル処理を記述することが可能である．既存のDWのフォームをPBやVBのフォームに変換するユーティリティも提供される．またDWのフォーム・デザイナは，ホストで作成した画面のモダナイズ作業に継続して利用可能である．

2. 導入によるメリット

DWPCEの導入により，次のような効果が期待できる．

1) GUI

LINC, MAPPERあるいは3GL言語で作成されたホスト側のアプリケーションをWindowsの統一された“ルックアンドフィール”環境で操作できる．

2) 生産性向上

アプリケーションの操作が簡単になり，利用者に対する教育期間も短縮できる．

3) 集中管理

フォームとロジックの全クライアントPCへの配布およびバージョン管理ができる．

4) クライアント構築

ローカル・ロジックを組み込むことにより，LINC, MAPPERあるいは3GL言語で作成されたホスト側の既存アプリケーションを変更せずにユーザ・インタフェースの拡張が可能である．また新規にアプリケーションを開発する場合にも，PCクライアント側の画面設計とローカル・ロジックの組込みを簡便にする開発ツールとして市販のPowerBuilderまたはVisual Basicが使用できる．

5) ホスト・アクセスが容易

スクリプトにより異なる複数のプラットフォームへ簡単に接続できる．

6) デスクトップ・アプリケーションとの協調動作

DW-CLASSIC（3R1以前のDW）を使用する場合は，MAPPERでDDEやDLLを提供している．

DW-PB（PowerBuilder対応のDW）やDW-VB（VisualBasic対応のDW）を使用する場合には，サードパーティの提供するDDE, DLLまたはOLEにより他のアプリケ

ーションと協調処理ができる．

7) ホスト・アプリケーションには影響なし

DWPCEは，ホスト・アプリケーションの変更なしに導入できる．DWPCEによりクライアント/サーバ・システムを段階的に拡張していくことができる．既存のシステムを使いながらクライアント/サーバ・システムに移行して，簡単なユーザ教育で生産性を上げることができる．

8) マルチ・プラットフォームの提供

デザイナ・ワークベンチから2200シリーズ，Aシリーズおよび UNIX などの複数ホストのアプリケーションと接続できる．

9) 接続オプション

INFOConnect および WinSock を提供するため，接続形態の選択肢が拡大されている．すなわち，TCP/IP や XNS による LAN 接続，X.25 や ISDN による WAN 接続および Uniscope や Pol/Sel の専用回線接続などが使用可能である．

10) リポジトリ

PC側のリポジトリにフォームやローカル・ロジックを保有することにより，実効速度の向上と回線使用量の削減を可能にしている．

3. 機能概要

ここではDWPCEとPowerBuilderを用いてPCクライアントを構築する場合（DW-PB）の開発環境と実行環境について，その概要を説明する．

3.1 開発環境

DW-PBでアプリケーションを開発するには，DWの他にPowerBuilderを使用する．PowerBuilderで作成したアプリケーションとDWリポジトリとのインタフェースは，DW-PBインテグレーション・ユーティリティにより行う．DW-PBインテグレーション・ユーティリティはチェックアウト，チェックイン，ツール・バーおよびDwCBUtilライブラリから構成される．

1) チェックアウト

チェックアウトはDWリポジトリからDWフォームとPowerBuilderフォームを読込む．DWフォームとはPowerBuilder形式に変換していないSCL形式のフォームである．また，PowerBuilder形式のフォームはPBLの部分とEXEの部分からなる．チェックアウトはDWフォームをPowerBuilderフォームに変換し，PBLファイルとしてコピーする．PowerBuilderフォームに対しては，PBL部分のみ取り出して変換せずにコピーする．PowerBuilderフォームのEXE部分は，更新されたEXEがDWリポジトリにチェックインされるまでそのまま保持される．チェックアウトではフォームの選択とコピー先のディレクトリの指定ができる．

2) チェックイン

チェックインは，新たに作成されたPBLとEXEをPowerBuilderフォームとしてDWリポジトリに保存するときに使用する．すなわちPowerBuilderによるクライアント・アプリケーションの開発が完了し，EXEを生成したあとにチェックインを実行してPowerBuilderフォームをDWリポジトリに保存する．チェックインではDWリポジトリに保存するEXEを選択できる．選択したEXEに対応したPBLも一緒にPowerBuilderフォームとして保存される．

3) ツールバー

DW-PBインテグレーション・ユーティリティのツールバーをPowerBuilderのPowerバーから起動することができる．ユーザはPowerバーをカスタマイズして，DW-PBインテグレーション・ユーティリティを浮動型のアイコンとして表示するように設定することができるのでPowerBuilderの開発環境と一体化される．これらのアイコンから「チェクアウト」，「チェックイン」および「ヘルプ」を起動することができる．

4) DwCBUtilライブラリ

DwCBUtilはPowerBuilderでは提供されない関数もしくはDwCBUtilを利用する方がより簡便な関数を提供する．これらの関数を利用することによりフォーム・ドライバとPowerBuilderの協調処理が促進される．DwCBUtilは3組の関数群から構成され，最初の組はフォーム・ドライバとPowerBuilder間で受け渡しされるデータの処理に利用される．2番目の関数の組はリスト・ボックスやコンボ・ボックスに使用するDATファイルの読込み機能を提供する．最後の関数の組

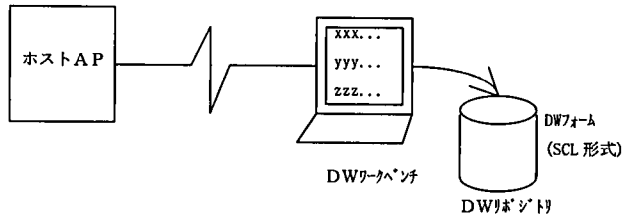


図1 ダウンロード

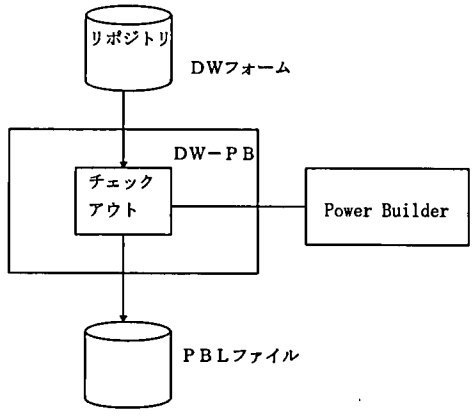


図2 チェックアウト

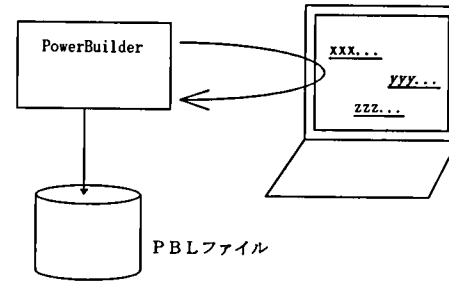


図3 PowerBuilder による変更

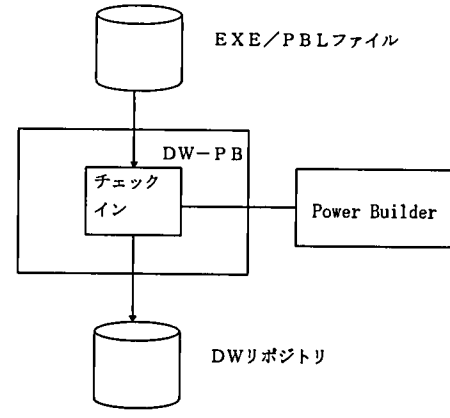


図4 チェックイン

は障害が発生した場合に解析の手助けとなる情報を処理するために使用する．

3.2 開発手順

1) ダウンロード（図1）

ホストの既存アプリケーションから表示される全フォームをDWワークベンチを用いてDWリポジトリにダウンロードする．

2) チェックアウト（図2）

DWリポジトリにダウンロードされたフォームを加工するために，まずDW-PBインテグレーション・ユーティリティを起動しチェックアウトを選択する．

チェックアウト機能は，DWフォームをDWリポジトリからPB開発環境にコピーする．このときDWフォームをPowerBuilderの形式であるPBLファイルに変換する．また，すでにDWを使っていたユーザがDWPCEに移行する場合も，このチェックアウト機能を用いてDWフォームをPBLファイルに変換する．

3) PowerBuilder による変更（図3）

次に開発者は，チェックアウトしたすべてのPBLの画面デザインとローカルロジックに対しPowerBuilderを用いて追加・修正を行う．この作業ではPowerBuilderの開発環境を利用する．

PowerBuilderアプリケーションのテストが完了したら実行モジュール生成機能により，PBLからEXEを作成する．

4) チェックイン（図4）

DW-PBインテグレーション・ユーティリティのチェックイン機能により，作成したEXEとPBLファイルをDWリポジトリに登録する．

5) アップロード（図5）

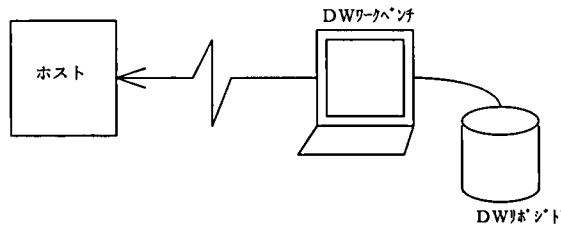


図5　アップロード

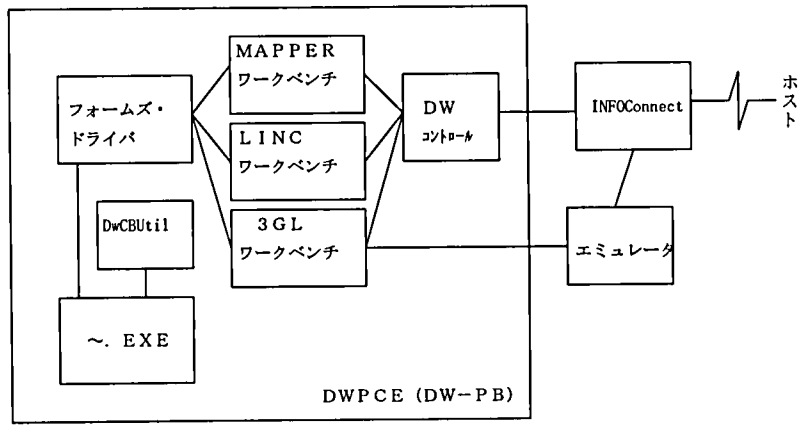


図6　DWPCE構成図

開発者は，DW-PBフォームをDWワークベンチを用いてホストにアップロードする．これでローカル・ロジックを含むフォームを他のクライアントにダウンロードして利用できる準備（開発）が完了することになる．

3.3　実行環境

DW-PBの実行環境は，DWワークベンチ，DWリポジトリおよびPowerBuilder実行用DLLから構成される．利用者は，DWワークベンチを用いて，ホスト・システムを利用する．利用者がホスト・システムのセッションを開き新しいフォームにアクセスすると，DWワークベンチは対応するDWフォームがリポジトリに，すでにダウンロードされているかをチェックする．もし存在しなければ，DW-PBフォームをDWリポジトリにダウンロードする．次にDW-PBランタイムがDW-PBフォームを表示する．ホストから受信したデータやホストに送信するデータは，フォーム・ドライバとPowerBuilderのアプリケーション間でメモリを介してやりとりされる．また，受け取ったデータの解析や送信データの結合はDwCBUtilが提供するPBのスクリプト用関数を使用して処理する．これらの機能を利用すれば入力フィールドにコード番号を入力する代わりに，ガイド画面を絵付きで表示し，その中から選択することも可能である．また，入力の値の範囲チェックも状況に応じて柔軟に設定することができる．

DWのVisual Basicとの連携機能もDW-PBとほぼ同様である．ただし対応バージョンはVB 3.0以降である．

4.　おわりに

ガートナーグループによるクライアント/サーバ・モデルの分類に照らし合わせて見ると，DW 3 R1までは，アプリケーション部分がホストに，プレゼンテーション部分がPC側にあるリモート・プレゼンテーション型である(図7)．DWPCEは，アプリケーション部分がホストとPCに分かれる機能分散型といえる．その特徴は，きめ細かいアプリケーション・ロジックをPC側で組み込めることであり，このことによりホストの処理負荷の軽減化と回線トラフィックの低減化が図れる．さらにこうしたパフォーマンスの向上のみならず，

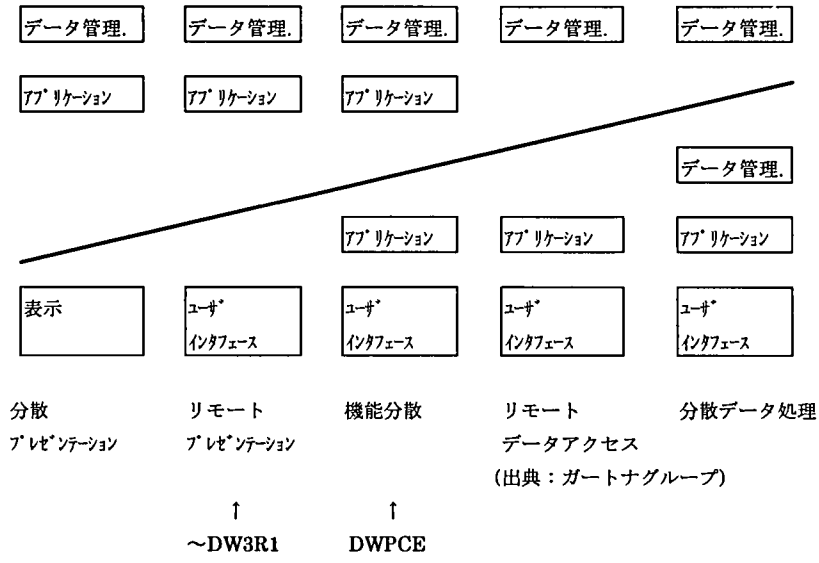


図7　オープン分散コンピューティング・モデルにおける DWPCE の位置づけ

アプリケーション開発のサーバ/クライアントへの分散により，アプリケーションの構築と改造に一層柔軟性を持たせることができる点が注目される．

また，画面やロジックをクライアント側に保有する形態ではそれらの集中管理に困難さが発生するが，DWPCE では常に最新版を自動的に配布するリポジトリ機能によりこの問題を解消することができ，この点も見逃せない特徴の一つである．

掲載論文梗概

システム構築がメインフレーム一辺倒から，高機能・廉価版の機器の組合わせによる形態になってきている．今回，ホスト・部門サーバ・利用者端末の分散構成による人事システム再構築を実施したが，部門サーバのDB維持管理/利用者端末とホストAPの連動等の解決が課題となった．伊奈健の**人事システム再構築における UNIshuttle の適用事例**は，本再構築がUNIshuttleを適用して実現したことを報告している．

UNIshuttleは分散処理システム構築の開発工数削減を狙いとした基盤ソフトウェアである．本ツール開発の特徴は，多くの要件の中から中核要件を探索しながら実現方案を案画し，これらを基本構造として機能の拡大を進め続けるアプローチにある．本田親光は**UNIshuttle の概要——分散処理システム構築向けの基盤ソフトウェア**の中で，開発アプローチ，主要な実現方案・機能，実現状況を概説し，中核要件設定の視点が分散処理システム開発者の汎用基盤ソフトウェア評価の視点に活用できることに言及している．

高性能で低価格なPCの提供やGUIを駆使したワープロや表計算ソフト等の出現により，PCがローカルネットワークの一部として利用され，その処理が業務系の一部を担うまでになってきた．佐藤友彦・久米進也は**OLTP——基幹システムとEUCの融合**の中で，PCとホストコンピュータ間のクライアント/サーバシステムについてOpen/OLTPプロダクトを例に考察している．

経営インフラとしてのネットワークにマルチベンダの多様な情報機器や通信機器がエンドユーザ主導で接続されるようになってきた．水野純一は**ネットワーク管理**の中で，LAN・LAN-WANインタネットワークにおいて，標準的な管理フレームワーク基づいた手法である「インタネット標準管理フレームワーク」の概要と実装例である「CNMSインタネット・マネジャ」を紹介し，更に改良した「インタネット標準管理フレームワーク第2版」のセキュリティ向上と管理プロトコルの改善等に言及している．

掲載論文梗概

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 15　No. 1　(No. 45)

発　行　日　平成7年5月31日
編集発行人　柳　生　孝　昭
発　行　所　日本ユニシス株式会社
東京都江東区豊洲1-1-1 〒135
TEL(03)5546-4111（大代表）
印　刷　所　三美印刷株式会社

UNISYS

日本ユニシス株式会社

本社 東京都江東区豊洲 1-1-1 〒135 電話03-5546-4111(大代表)

OpenMAPPER 増発計画。

OpenMAPPERは直販だけでなく、
日本ユニシスと契約された代理店／販社からも販売いたします。

販売のオープン化により、あのOpenMAPPERが、いちだんと身近になります。

エンドユーザ・コンピューティングで定評のあるユニシスのソフトウェア「OpenMAPPER」が、ついに販売もオープンになりました。PC、部門サーバ、全社サーバのマルチプラットフォームを実現し、情報の共有化によるいちだんと創造的な業務を支援するコラボレーション・コンピューティング・ツールとして、そしてまた、あらゆる形態のクライアント／サーバ・システムを容易に実現するシステム構築ツールとして高い評価を受ける「OpenMAPPER」。ますます身近になってオープンの世界を広げていきます。

OpenMAPPERの特長 ●PCからホストまでの統一された操作環境。●各社UNIXシステムおよびWindows上で稼働。●あらゆる形態のC／Sシステムが構築可能。●業界標準リレーショナル・データベースに対応。●市販PCソフトとのデータ連携。●豊富なビジネス・グラフ機能。●GUIによるアプリケーション開発。●機密保護機能や運用管理機能。

コラボレーション・コンピューティング・ツール

OpenMAPPER

資料のご請求、お問合せは、会社名、住所、所属(役職)、氏名、電話番号を明記の上、日本ユニシス本社広報部まで、FAXまたは郵送にてお願いいたします。 FAX03(5546)7800