55−E001

欧州における情報処理教育等の
実態調査報告書

昭和56年3月

財団法人　日本情報処理開発協会
情報処理研修センター

欧州における情報処理教育等の実態調査報告書

ソフトウェアの生産性に関する

— ヨーロッパ調査報告書 —

1980 年9月7日から9月23日まで17日間にわたり，フランス，西ドイツ，イタリアの3か国を訪問し，ソフトウエアの生産性に関する調査を行なった。参加者は

宇都宮　公訓　（筑波大学）

小瀬村　克也　（財団法人日本情報処理開発協会）

の2人である。本書はその調査結果の報告である。

目　　次

1. はじめに

情報処理技術の発達は目覚ましく，社会生活に広く，深く浸透し，成熟した情報化社会は着実に近ずきつつある。

集積回路技術の発達によって計算機ハードウェアの価格が急速に下がり，もはや，4ビット（単位で処理する）1チップ(の)マイクロコンピュータは1,000円もしない。このように安価な計算機が従来の道具，機械装置，システムで使われるようになり，1980年代はまさにコンピュータの大衆化時代である。しかし，必ずしも問題がないわけではない。ハードウェアのコストは下がったが，それを利用するためのソフトウェアの方は多くの難問を抱えたままなのである。この点について少々考察し，解決の方向をさぐってみたい。

情報処理において，ハードウェアとソフトウェアは不可分である。ソフトウェアの問題を考えるとき，それに光と影を与えているハードウェアの強い影響を見逃すことはできない。そこで，ハードウェアの発達をざっと見ておきたい。表1.1はＩＢＭの計算機の性能，性能価格比である。また，図1.1はマイコンの性能の推移である。Z 8000（というマイコン）は過去のベストセラー・ミニコンPDP 11／45の性能を凌いだといわれている。このように計算機ハードウェアは高速になり，安くなってきている。この傾向は順調に続き，1980年代に計算機の性能は約10倍も向上すると予測されている。計算機メモリについては，1978年から3年ごとに，メモリ・チップの密度は4倍の割で高くなり，それにつれビット当りの価格は3分の1の割で低下すると予測されている。すなわち，1978年の16キロ・ビット／チップが1987年には1メガ・ビット／チップになり，価格は1／27に下がるというわけである。さらに，図1.2に示すように磁気ディスクの価格も4年に3分

の１の割合で着実に下がっている。もちろんそれと同時に性能も向上している。例えば，1979 年に発売されたＩＢＭ3370 磁気ディスクを，

A．大 型 機

1965 年	360／65	0.4 MIPS
1971 年	370／165	1.98 〃
1973 年	370／168	2.42 〃
1976 年	370／168MP	3.18 〃
1978 年	3,033	4.96 〃
	3,033 MP	8.31 〃
1980 年	3,081	10.3 〃

B．中 型 機

1972 年	370／135	0.2 MIPS
1977 年	370／138	0.27 〃
1980 年	4,341	0.88 〃

C．性能価格比

3,033 は 370／168 の約２倍

4,341 は 370／198 の約５倍

MIPS：Million Instructions Per Second

MP：Multi－Processor

表 1.1　ＩＢＭの計算機の性能，性能価格化

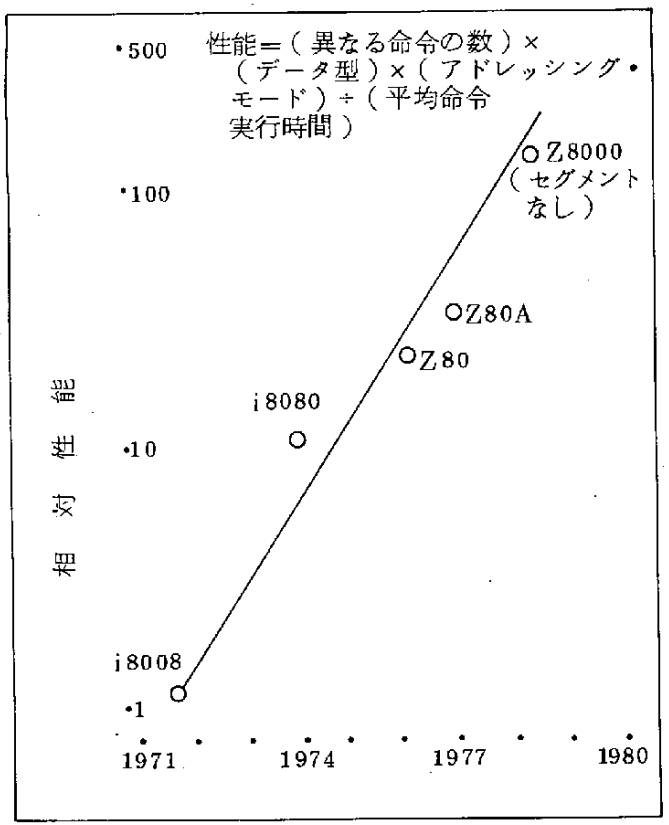


図 1.1　マイコンの性能の推移

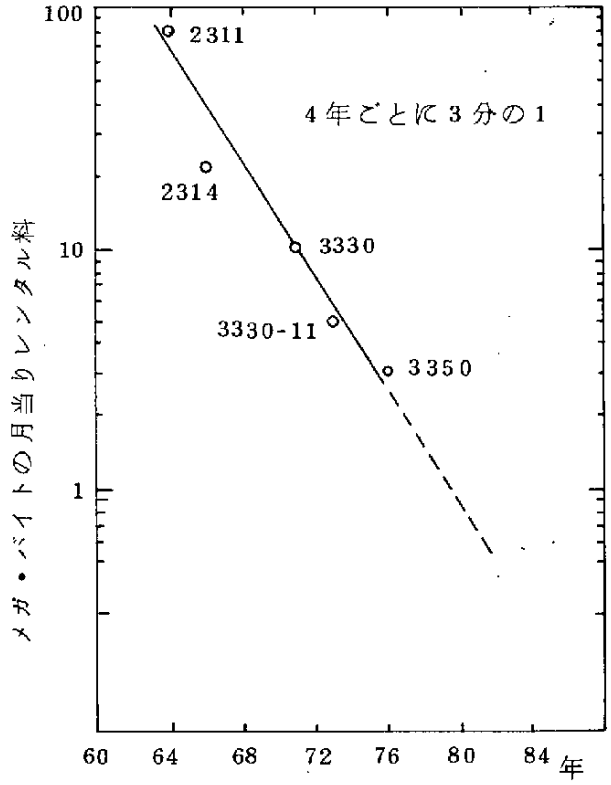


図 1.2　磁気ディスクの価格の推移

同社の3330－1型磁気ディスクと比較すると，次のようになっている。

1メガ・バイト当りの月当りレンタル料：　246円で3330—1の⅛

平均アクセス時間：　20ミリ秒で3330－1の⅔

1秒当りのデータ転送量：　約1,859キロ・バイト（3330—1の約2.3倍）

さて，このようなハードウェアの目覚ましい発達に対して，ソフトウェアの発達は思うにまかせない状態が続いている。1974年に行なわれたＩＢＭのSHARE会議では，表1.2のような数字が報告されている。すなわち，計算機（ハードウェア）の性能／価格は，1955年から，10年に100倍の割で向上しているが，プログラマの生産性向上は（10年に）2倍強の割でしかない。性能／価格の向上の割合に対する見方がやや甘い気もするが，大勢において誤ってはいないと思う。別の予測（「Data Processing in 1980—1985」）によれば，1980年代に，ハードウェアの性能が約10倍向上する一方で，ソフトウェアの生産性向上は（手をこまねいていれば）年率約3％でしかない

	1955	1965	1975	1985
性能／価格	1	102	104	106
プログラマの生産性	1	2.4	5.6	13.3
システムの信頼性	1	5	24	120

1975，1985は予測

表1.2　ハードウェアとソフトウェアの発達の差

とのことである。年率3％では10年経っても2倍にもならない。

1979年の労働白書によれば，わが国の実質賃金は1970年を100とすると1977年には167と大幅に上昇している。計算機ハードウェアが速く安くなり，人件費が高くなっていることは，計算機を用いた省力化への圧力が今後ますます強くなることを意味している。計算機の性能が10倍になるだけでも，それに見合うだけのソフトウェアが要求される。要求されるソフトウェ

アの量も10倍とまでは言えないとしても，それに近いことは確かである。いや賃金の上昇がはげしければ，10倍を超えることも考えられる。例えば，ソフトウェアの要求量が10倍になるとして，それを開発する生産性向上が（10年で）2倍でしかないとすれば，現在の5倍のプログラマが必要ということになる。しかし，この要求は到底かなえられるはずがなく，別の角度からのの解決を図らなければならない。データ処理化が進んでいる時代ではあるが，（企業における）データ処理部門も決して省力化の対象から外されているわけではない。

さて，ハードウェア・コストが安く，ソフトウェア・コストが高くなると，データ処理コスト全体に占めるソフトウェア・コスト（あるいは人件費）の割合が非常に高くなる。米空軍では，1955年にソフトウェア（コスト）の割合が15％であったものが，1985年には逆転して85％になると予測されている。1979年のわが国のコンピュータ白書でも，図1.3のように，ソフトウェア・コスト（人件費と外注費）が増加傾向にあることを強調している。ま

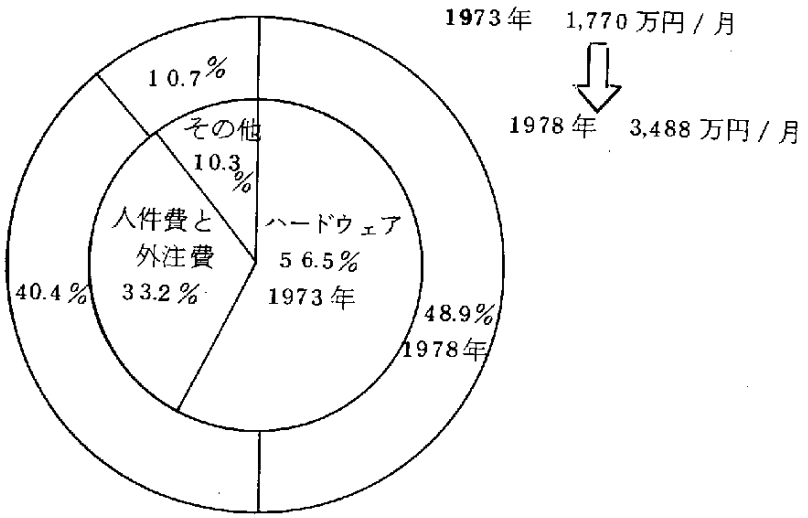


図 1.3　データ処理部門の費用

た，インテル（intel）社は，ハードウェア価格が24,000円のマイコンで動くソフトウェアの開発に，（高水準言語を使っても）2千万円（アセンブリ言語では9千万円）かかると言っている（「日経エレクトロニクス」1980年6月23日号）。

ソフトウェアの生産性はどうして上がらないのだろうか。それは，ソフトウェアの製作がほとんど人間だけの力によるからである。ハードウェアは確かに速く，安くなった。しかし，決して利口にはなっていない。計算機を利用する上で重要な知能（intelligence）はほとんどソフトウェアが背負っている。ソフトウェアに少々肩入れした言い方をすれば，「ハードウェアは，難しいところをすべてソフトウェアに押しつけて，やり易いところだけをつまみ食いしている」と言えなくもない。確かにこれは今後の課題である。ソフトウェアの負担を軽くする計算機をどう作るか，もっと真剣に議論しなければならない。しかし，ソフトウェアにはその完成を待っている余裕はない。少なくとも，1980年代にソフトウェアの生産性や信頼性を援けるアーキテクチャの計算機が広く使われるようにはならないと判断される。ハードウェアの機能は基本的に現在と変らないとして，ソフトウェアの問題をどう解決するかを考えなければならない。

ソフトウェアの問題は，

- 開発コストが高い
- 維持（保守）コストが高い
- 信頼性が低い
- 大きなソフトウェアは開発そのものが難しい
- 開発管理が難しい

ことだといわれている。これらは原則的に「（ソフトウェアを）つくる」場合の問題である。ソフトウェアをつくる場合に心がけるべき戒めを1つだけ

に絞ると，「Readability（読み易さ）を上げよ」ということになると思う。Readabilityを上げることは，上記5つの問題に対してプラスの方向に働く。ただし，Readabilityを上げることは，一般に，計算機ハードウェアの使用効率を低下させる。しかし，これは，人件費が高く，ハードウェアの性能価格比がいいことでじゅうぶんお釣りがくるので，問題にはならない。確かに，ハードウェアをふんだんに使っても，ソフトウェアの生産性を上げるようにする」ことを基本姿勢にしなければならない。いまはハードウェアは安い。ハードウェアの効率的利用のために，ソフトウェアで苦労するのは，一般に，上策ではない。現在，ハードウェアがソフトウェアに唯一貢献できることは「速くて安いこと」であり，それを活用しない限り，ソフトウェアはいつまでも辛い思いをするばかりである。

ソフトウェアの開発で使える技法として，ＩＰＴ（Improved Programming Technologies，効果的プログラム技法）をはじめいろいろな技法が開発されている。ＩＰＴを使うことによる生産性向上はほぼ2倍と報告されている。しかし，見方を変えれば，効果的開発技法を使っても2倍にしかならずこれらに頼っているだけでは，この10年（10倍のソフトウェア量）は支えきれないということになる。すなわち，開発技法にだけ頼るのではなく，もっと総合的な解決策が必要といえる。とくに，企業のデータ処理部門の状態を考えるとその感が強い。

企業のデータ処理部門の立場は非常に厳しくなってきている。人件費の上昇を抑える傾向は全社的なもので，データ処理部門も例外ではなく，要員は増やしてもらえない。これまで開発してきた適用業務プログラムが累積しており，その保守・変換のために要員を投入しなければならない。ソフトウェアには必ずといっていいほど保守・変換がつきまとう。従って，適用業務プログラムが多くなればなるほど，保守・変換作業量も増える。その結果，（デ

ータ処理部門としての）新規開発能力は下がる一方で，ＩＢＭの調査では，1975年を100とすると1978年は77ぐらいになっている。他方，適用業務プログラムの開発要求の圧力は強く，開発すべき適用業務の量は毎年20%の割で上昇しており，1975年を100とすると，1978年では175となっている。このままでは，開発待ちの適用業務がどんどんたまってくる，ということになる。これを解決する最善の策は，既製の適用業務（プログラム）パッケージを買うことである。要求される膨大なソフトウェアを社会全体として効率よくまかなうことを考えるべきである。お金をかけたいいソフトウェアを多くの人で使い合うようにすれば，社会全体として安くてよいソフトウェアが使えるようになる。すなわち，「ソフトウェアの流通」こそ．ソフトウェア生産性向上の最善の策なのである。これに近い考え方として，ソフトウェア・ハウスなどへの外注，買ってきて手を加えることなどもある。

さて，どうしても企業内でつくらなければならないとする。その場合でも必ずしもデータ処理部門がつくらなければならないというわけではない。すなわち，エンド・ユーザにつくってもらい，データ処理部門はその支援にまわればよい。驚くことにデータ処理部門はエンド・ユーザからは決してよく思われてはいない。日本データ・プロセシング協会の調査では，表1.3のように，データ処理部門は自分達の要求にじゅうぶん応えてはくれないと思われている。適用業務プログラムの開発をデータ処理部門に見積ってもらったところ，（エンド・ユーザが）自部門で開発する場合のコストの5倍以上も要求されたという例もある。また，開発してもらっても，自分の思ったものとはほど遠いとか，完成が遅すぎるといった苦情も絶えない。従って，エンド・ユーザにできる場合，エンド・ユーザにつくってもらった方が生産性，完成時期，質がよくなる場合は，エンド・ユーザに積極的につくってもらいデータ処理部門はその支援にまわる方が利口である。

・非常に業務オリエンティッドな（一過性に近い）適用業務の増加

・オペレーショナルな利用から意思決定型の利用へ

・定型的利用から非定型的利用へ

と利用の質が変化してきており，高い適合性と即応性が要求されている。従

突発作業	1975年	1979年
相当無理でも応じる	41.2%	33.7%
応じない	8.1%	10.5%
新しいシステム開発		
積極的	73.9%	66.5%
消極的	4.3%	8.3%

表 1.3　エンド・ユーザからデータ処理部門を見た場合

来のようにデータ処理部門が要求をきき，人を割り当て，業務の内容を理解し，開発したので余りにも生産性が低く，場合によっては，意味をなさなくなることさえもある。では，データ処理技術に関しては素人に近いエンド・ユーザに適用業務プログラムの開発が可能であろうか。エンド・ユーザは自身の業務に精通しているがデータ処理技術にはうとい。しかし，

・データベース／データ通信機の整備

・表示型端末による対話式計算システムの普及

・（関係データベース，プロンプティング，サイド，HELPコマンドなどによる）エンド・ユーザ向きソフトウェアの充実

などにより，エンド・ユーザ自身による適用業務開発の可能性は大きくなっている。もちろん，データ処理部門による支援，教育が不可欠なことは言うまでもない。エンド・ユーザ参加の問題点を以下にまとめておく。

(1) コンピューティング・パワーの自由な使用
対話式計算，分散処理など

(2) 業務の考え方，用語によるコンピュータの利用

エンド・ユーザの世界観の確立

（業務告きパッケージ，マクロによるテーラリング，特殊目的言語，ＡＰＬ関数）

(3) 完全に知ってから使うシステムではダメ

できるだけマニュアルを使わず，端末の前に座って使いながら理解する

（メニュー方式，サイド，HELPコマンド）

(4) 理解しやすいロジック表現

ＱＢＥ的手法による照会・検索，ＡＰＬの関数

(5) 業務とデータ処理システムの強い連繋

思考の中断のない使い方

試行錯誤的な使い方

迅やかな対応

アルゴリズムの可変性

既存のプログラムやデータの容易な利用

いくぶんニュアンスは違うが，既存のプログラムやデータベースだけを使って，計算したり，作表，作図したりするエンド・ユーザをサポートすることも必要である。既製のものを自在に組み合わせて使わせるようにすることは現実には難しいことであり，それが可能なシステムを（エンド・ユーザのために）整備してあげることも重要である。

最後に，どうしてもデータ処理部門でつくらなければならない場合を考えよう。その場合はまず高水準インタフェースを採用すべきである。データ処理部門はメーカ提供の機能をベースにし．適用業務プログラムで機能を補足し，要求を満たすデータ処理機能（と性能）を実現する。適用業務プログラムで補足しなければならない機能の量（深さ）が大きいほど開発工数が増え，

保守労力が増える。従って，メーカにできるだけ要求仕様に近い高い水準の機能を提供してもらい，それを利用することにより適用業務の労力軽減を図ろうというわけである。データ処理部門（一般にユーザ）とメーカのこの（高い水準の）インタフェースを高水準インタフェースと呼んでいる。高水準化には次のような例がある。

処理系：　機械語→アセンブリ言語

→高水準（コンパイラ）言語

記憶系：　ファイル・システム

→データ管理システム

→データベース・システム

通信系：　基本データ伝送制御手順

→ハイ・レベル・データ伝送制御手順

→通信ネットワーク体系

→通信制御パッケージ

高水準言語：　FORTRAN→APL

データベース：　木構造→関係構造

記憶管理：　実記憶→仮想（主記憶）→一元化記憶

また，高水準インタフェースを採用することによる利点には

- 適用業務の開発保守労力が軽減される
- 適用業務プログラムの寿命がながくなる
- システム技術の動向に左右されない，カスタマ本位の適用業務開発ができる
- 新しいシステム技術（の効果）を適用業務に容易に吸収させることができる。
- ハードウェア資源の効率的運用の自由度が高くなる

・複雑なシステムの信頼性が向上する

などがある。

次に，計算機による支援，とくに対話式プログラム開発を利用することである。これは単に端末を通して計算機を使うことを意味しているだけではない。本質的に対話式である言語，対話式で使うようにつくられたパッケージなど，対話式ソフトウェアの充実がなければその価値は著しく下ってしまう。このような優れた開発環境，さらにはツール類を整備し，その上で，ＩＰＴなどの手法を採用することである。適用業務のプログラムの開発で今後問題となるのは，漠然としたデータ処理化の要求を，データ処理技術をふまえた（具体的）要求仕様にまで具体化する過程で使える手法の開発である。本調査の中にある，DB／DC 指向の適用業務設計手法はこれにあたる。

以上，ソフトウェアの生産性に関する種々の問題を総合的に解決する基本姿勢を述べた。本調査はこの基本姿勢にもとずいて行なった。ＩＰＴに代表されるコーディングに近いレベルでの手法は多く報告されているので，本調査では意識的に外している。本調査における具体的テーマと，その調査のための訪問先の対応は次のようになっている。

(1) ソフトウェア開発保守の実態：Cii－Honeywell Bnll

(2) データ処理部門の生産性：ＩBMドイツ本社，ＩBMイタリア，ローマPPDC

(3) ソフトウェア開発導入保守環境：ＩＲＩＡ－ＬＡＢＯＲＩＡ，ＩBMドイツ本社，ＩBMイタリアＰＳＩＣ

(4) データベース／データ通信・システムおよびその上で動く適用業務プログラムの開発保守用パッケージ：ＩＢＭドイツ本社，ＩＢＭイタリア・ローマＰＰＤＣ

(5) 適用業務向きプログラム・パッケージ：ＩＢＭドイツ本社，ＩＢＭイタリアＰＳＩＣ，ＩＢＭフランス・ラゴード研究所

(6) エンド・ユーザ向きシステム：ＩＢＭドイツ・ハイデルベルグ・サイエンティフィック・センター

(7) 要求仕様決定までに使える開発手法：ＩＢＭフランス教育センター

(8) ソフトウェアの開発保守を支えるハードウェアの発達動向：ＩＢＭドイツ・ボブリンゲン研究所，ＩＢＭフランス・ラゴード研究所

2. 調査訪問先

本調査で訪問した組織，対応してくれた方，主たる調査内容を日程順にまとめる。

(1) Cii Honeywell Bull, Paris

所在地： 94 Avenue Gambetta - 75960 Paris, France

内　容： • GCOS 64の開発保守について（Claude Carré氏）

(2) IRIA-Laboria

所在地： Domaine de Voluceau Rocquencourt BT 5 78150 Le Chesney, France

内　容： • PASCALプログラミング環境（Monsieur G.Kahn氏）

(3) IBM France Education Center

所在地： 75 Avenue Edouard Vaillant, 92 Boulogne-Billancourt, Paris, France

内　容： • データベース／データ通信指向の適用業務設計手法（Alain Nagler氏）

(4) IBM Germany Headquarters

所在地： Pascalstrasse 100, Stuttgart, Germany

内　容： • データ処理部門の生産性（P. Kirn氏）
• APL-DI, QBE, ADRSの比較（Muller氏）
• SIPO/E. とELIAS （Klaus Kosalla氏）
• 計算機を利用した教育（Klaus Schneider氏）
※セットアップ：Knut E Wuruster氏

(5) IBM Germany Boeblingen Laboratory

所在地： Schoenaicher Strasse 220, Boeblingen, Germany

内　容：　・IBM4331モデル2型（Max Briner氏）

※セットアップ：　Uwe Wehlers氏

(6)　IBM Germany Heidelberg Scientific Center

所在地：　Tiergartenstrasse 15, Heidelberg, Germany

内　容：　・Integrated Data Analysis and Management System（Ulrich Schauer氏）

(7)　IBM Italy Public Sector Industry Center

所在地：　VIA IV Novembre 103, Rome, Italy

内　容：　・PSICの任務，教育などについて（Pierlurgi Arnao氏）

・研究教育適用業務パッケージの開発について（Giorgio Dori氏）

・MUSIC（Ulrich Schönbaumsfeld氏）

・DOBIS（Ulrich Schönbaumsfeld氏）

※セットアップなど：　Tom Goldschmid氏, Luciano Costa氏）

(8)　IBM Italy Rome Program Product Development Center

所在地：　Eur-VIA Oceano Facifico 71/73, Rome, Italy

内　容：　・Entry Level Interactive Application System（Guido Mazzeo氏, Paolo Giorgi氏）

(9)　IBM France La Gaude Laboratory

所在地：　06610-La Gaude France

内　容：　・La Gaude研究所の沿革，任務など（Claude Pouget氏）

・通信技術の動向（Philippe Nay氏）

・ITIRC（Philippe Nay氏）

3. 調 査 結 果

調査結果を訪問日程の順に報告する。

3.1 CII−HB Paris

Cii Honeywell Bullグループは，資本金の53％をCompanie des Machines Bull （CMB）が，47％をHoneywell Information Systems Inc. が出資してつくられた会社組織で，Companie Internationale pour I' Informatique CII-Honeywell Bull（CII-HB）と，Compagnie CII-HB Internationale N.V.（CII-HBI NV）の2つの会社からなる。CII-HBの本部はパリにある。CII-HBI NVはアムステルダムにあり，オランダ籍の会社である。これら2社はグループとして同じ会長の管轄の下にある。

CII-HBは，グループ（全体）としての一般のサービス，研究開発，製品生産，フランス，中東，アフリカ地区での販売活動を担当すると同時に，外国市場での直接セールスも行なう。一方，CHH−HBI NVはヨーロッパ14か国と南アメリカ地区での販売を担当している。

Cii Honeywell Bullグループは，当然のことながら，Honeywell Information Systems Inc. グループと強い提携関係をもっており，技術的な協力，交流を行なっている。

Cii Honeywell Bullグループの概要を表3.1にまとめた。

CII−HBでは，（Level64 計算機の）ソフトウェア・アーキテクチャ担当マネジャーであるClaude Carré 氏に会い，Level64 計算機，その上で動くオペレーティング・システムGCOS64 の開発，維持（保守），Cii Honeyell Bull グループの研究開発等について調査した。

3.1.1 Cii Honeywell Bull グループの研究開発

Cii Honeywell Bullグループはフランスに（グループとしての）センターを7つもっており，年間約5億5千万フランス・フランの予算で研究開発を行なっている。これらのセンターはAngers, Belfort, Grenoble, Les Clayers-sous-Bois, Louveciennes, Paris, Saint-Ouen にあり，それぞれ次のようなテーマを担当している。

Angers：

従業員（1979年12月31日現在）
人数　19,054人
そのうちフランスに14,530人，フランス以外に4,524人
職業分布
販売：26.8％，保守：23.6％，研究開発：13.2％，
製造　22.7％，管理：13.7％，
研究開発
1,313人のエンジニアをふくめて2,523人
フランスに7つのセンター
予算は5億5千万フランス・フラン
製　造
4,518人の従業員
フランスのAngersとBelfortに工場
製　品
Series Mini6, Series 61 DPS, 62,
Series 64DPS, DPS7, Series 66DPS, DPS8
Series Iris, 7000 Series Terminals
市場占有率
全世界で10,000の顧客数
フランス市場の25％～30％
ヨーロッパ市場の10％～20％
CII−HBとHISを合わせれば世界第2位のEDPメーカ

表3.1　Cii Honeywell Bull グループの概要

大型，中型システム（ハードウェア）マイクロ・パッケージング（高密度回路実装技術）

Belfort

計算機周辺装置（プリンタ，カード装置）

データ収集装置

文書読取り装置

Grenoble

Grenoble大学との協同による（高等）研究

Les Clayes-sous-Bois

大型，中型システム（ハードウェア）

磁気周辺装置（ディスク，テープ）

端末（ハードウェア），

素　　子

Louvecieness

大型，中型システム（ソフトウェア）

端末（ソフトウェア）

特殊システム（ハードウェア，ソフトウェア）

ネットワーク体系

新しい活動

科学部門

標　準　化

Paris

大型，中型システム（ハードウェア，ソフトウェア）

小型システム（ハードウェア，ソフトウェア）

新しい活動

Saint-Ouen

マイクロパッケージング

信　頼　性

標　準　化

Cii Honeywell Bullグループは Honeywell Information グループと提携しているので，そちらの研究センターについても少々触れておく。

Honeywell Information Systems Inc, はアメリカ合衆国に次の4つのセンターをもっている。

Billerica (Massachusetts)

ミニコン (ハードウェア，ソフトウェア)

Cambridge (Massachusetts)

MULTICS オペレーティング・システム

Los Angeles (California)

大型システム (ソフトウェア)

Phoenix (Arizona)

大型システム (ハードウェア，ソフトウェア)

Honeywell Information Systems Ltd, はイギリスに次のセンターをもっている。

Hemel Hempstead

ミニコン (ハードウェア，ソフトウェア)

特殊ミニコン・システム

最後に，Honeywell Information Systems Italia Spaはイタリアに次のセンターをもっている。

Pregnana

Level 62 システム (ハードウェア，ソフトウェア)

計算機周辺装置（ＬＣＳＰ 小型プリンタ）

Cii Honeywell Bullグループの研究成果の１つであり，プログラム言語分野の研究に最近大きな話題を提供しているものに新プログラム言語ＡＤＡがある。ＡＤＡは，Jean Ichbiahが率いるＡＤＡ プロジェクトで開発されたものである。IchbiahはCii Honeywell Bull グループ研究センターのソフトウェア部門の責任者である。ＡＤＡは，アメリカ国防省の計算機のリアル・タイム・プログラム用言語として採用されることが決まった。この決定は1979 年の５月になされたが，選定作業は1975 年に始まっており，長い月日を要している。国防省からの提案要求は全世界に出された。その目的は，1980 年代に普遍的な標準となり得，しかも，ソフトウェア・コストの節減につながるという国防省の要求を満たす，新しいプログラム言語の開発を促すことであった。国防省が費やしているソフトウェア・コストは年60億ドルという巨大なものであり，国防省の苦労が感じられる。

ＡＤＡは，数値データの処理能力だけでなく，並列動作の管理，並列動作の実行タイミング用の基本機能を有しているので，基本ソフトウェアや，リアルタイム・システムを書くのにとくに適している。なかんずく，統合システム，すなわち，フライト制御システムや，機械装置制御システムのような（計算中心ではなく）より一般的なシステムのデータ処理部分を書くのに適している。ＡＤＡは，STEELMAN レポートと呼ばれている。国防省がつくった非常に正確な要求（の組）にそって設計されている。このレポートでは次の目標を掲げている。

- 広い範囲の業務に適用可能であること
- プログラムの信頼性，保守性が高まること
- （言語として）簡潔であること

- コンパイラーがつくり易いこと
- プログラムの効果が大きいこと
- プログラムが移植しやすいこと

すなわち，ＡＤＡはこのような目標にそった言語である。ＡＤＡについては，種々報告書も出ている（例えば，Ada基準文法書，bit,1981年，1，共立出版）ので，ここではこれ以上述べない。

Cii Honeywell Bull グループは，Honeywell Information Systems グループとともに，ＰＴＴなど通信会社と提携して，新しいネットワーク体系ＤＳＡ（Distributed System Architecture）を発表したが，当然ながら，これは，ＩＢＭのＳＮＡや，日本電信電話公社のＤＣＮＡによく似ているので，割愛する。

面白い報告を見つけたので表3.2に掲げておく。1978年にFredrik Bull Foundation が情報処理の将来について125人のエキスパートの意見をきいた結果である。

3.1.2 Level64 計算機とGCOS

Series60 Level 64はＩＢＭコンパティブルの（互換性を有する）計算機で，ＩＢＭのSystem／3，System／38から43××，3031の対抗機種である。ＩＢＭ3031から3033までの対抗機種はLevel66（ＧＥ系）である。Level64は現在世界で約2,000台使われている。その内訳は，アメリカが400台，フランスが600台，ドイツが400台，イギリスが200台である。

Level64上で動くオペレーティング・システムは唯一GCOS64（General Comprehensive Operating System 64）だけである。この他，Level 64上で動かすように提供されているソフトウェア（プロ

	1980	1985	1990	1995	2000	それ以上
全銀行が計算機システムで結合される	22	69	23	6	3	
電話等を介して個人がデータ処理ネットワークにアクセスする	16	40	38	18	7	2
テレビを通して情報センターから情報を得る	7	33	34	22	17	10
会社用通信手段として電子郵便が使われる	1	18	51	25	20	3
ペーパーレス・オフィス	5	16	21	28	25	30
５０％以上の製造作業が自動化される	1	15	39	29	19	16
５０％以上の組立て作業が自動化される	0	1	18	23	37	37
あいまいさのないテキストの自動翻訳（数千語）	2	17	27	17	26	35
一般開業医で計算機援用診断が実施される	9	32	39	24	11	9
高度のＣＡＩ	4	18	38	15	23	25
オンライン投票（国民の家からの直接投票）	1	3	11	17	24	60

表 3.2　データ処理技術による将来の社会的成果の予測

グラム製品）には次のようなものがある。

- Transaction Driven System（ＴＤＳ）
- Integrated Dota Store Ⅱ（ＩＤＳⅡ）
- File Catalog
- Network Supervisor
- Interactive Operation Facility（ＩＯＦ）
- Message Access Method（ＭＡＭ）
- Sales Order Processing（ＳＯＰ）
- Distribution and Inventry Management System（DIMS）
- Industrial Management System-Transaction Driven（IMS-TD）
- Statistical and Data Analysis Package（STATPAC）
- Mathematical Subroutine Library（MATHLIB）

- Linear Programming（ＬＰＩ）
- Product Management and Control System-Extended（PMCS−X）
- Program Checkout Facility（ＰＣＦ）

Cii Honeywell Bull は，1980年代の新しい計算機システムとして，ＤＰＳ７シリーズを発表している。ＤＰＳ７には60，70，80，82のモデルがある。それぞれのモデルの要約を表3.3に示す。DPS7／82は2プロセッサ計算機でＳＩＲＩＳ８−Ｅオペレーティング・システムの下で動く。ＤＰＳの特徴としては，

- 非中央化アーキテクチャ（図3.1参照）
- 新しい回路，実装技術（Current Mode Logicとマイクロ・パッケージング）

モデル	7／60	7／70	7／80	7／82
主プロセッサの数	1	1	1	2
主プロセッサのサイクル時間（＋1秒）	210	160	110	110
主記憶の容量（メガ・バイト）	2，3，4	2，3，4	3，4	4，8
入出力プロセッサ・グループ	1	1	1	2
入出力プロセッサ（最小／最大）	4／12	4／12	4／16	4／32
入出入プロセッサ・グループの最大処理能力（メガ・バイト／秒）	18	18	19	29
入出力プロセッサの最大処理能力（メガ・バイト／秒）	2.5	2.5	2.5	2.5
ユニット・レコード・プロセッサ（組込み＋選択）	1+1	1+1	1+2	1+5
ユニット・レコード装置	10	13	21	45
大容量プロセッサ（組込み＋選択）	1+2	1+3	1+3	1+7
接続可能ディスクの装置	27	36	36	72
オンライン容量（１０億バイト）	16	21	21	42
磁気テープ・プロセッサ（単一アクセス）	2	4	4	8
磁気テープ・プロセッサ（デュアル・アクセス）	1	2	2	4

磁気テープ装置	16	32	32	64
Datanet 7100（最大）	1	2	2	4
・ライン（最大）	48	128	256	512
DCC4380	1	1	1	—
・ライン	15	15	15	—
オペレーティング・システム	GCOS64-E SIRIS8-E SIRIS3-E	GCOS64-E SIRIS8-E SIRIS3-E	GCOS64-E SIRIS8-E SIRIS3-E	SIRIS8-E

表 3.3　DPS／7 の各モデルの要約

- 可能性の向上
- ファームウェアの積極的採用
- 複数操作環境（64／DPSなどの従来の計算機システム上で動いていた適用業務プログラムはすべてDPS／7で使えるどのオペレーティング・システム上ででも動く）
- ＤＳＡ（Distributed Systems Architecture ）を用いた分数処理が可能

などがある。

3.1.3　GCOS 64 の開発と保守

GCOS64（General Comprehensive Operating Systen 64 ）はLevel 64などの上で動くオペレーティング・システムで，1975年に Release 1（初版）が出荷されてから，1980年のRelease 5（第5版）まで改訂が続けられてきている。GCOS64（Level 64 ）は，約90％の多数を占めるCOBOLユーザを強く意識してつくられている。ＡＰＬインタプリタ，ＢＡＳＩＣインタプリタもサポートしている。将来は，ＡＰＬ，ＢＡＳＩＣのユーザが増えると予想している。PL／Iもサポートしているがユーザの数は少なく，今後の増加も見込めないと考えている。

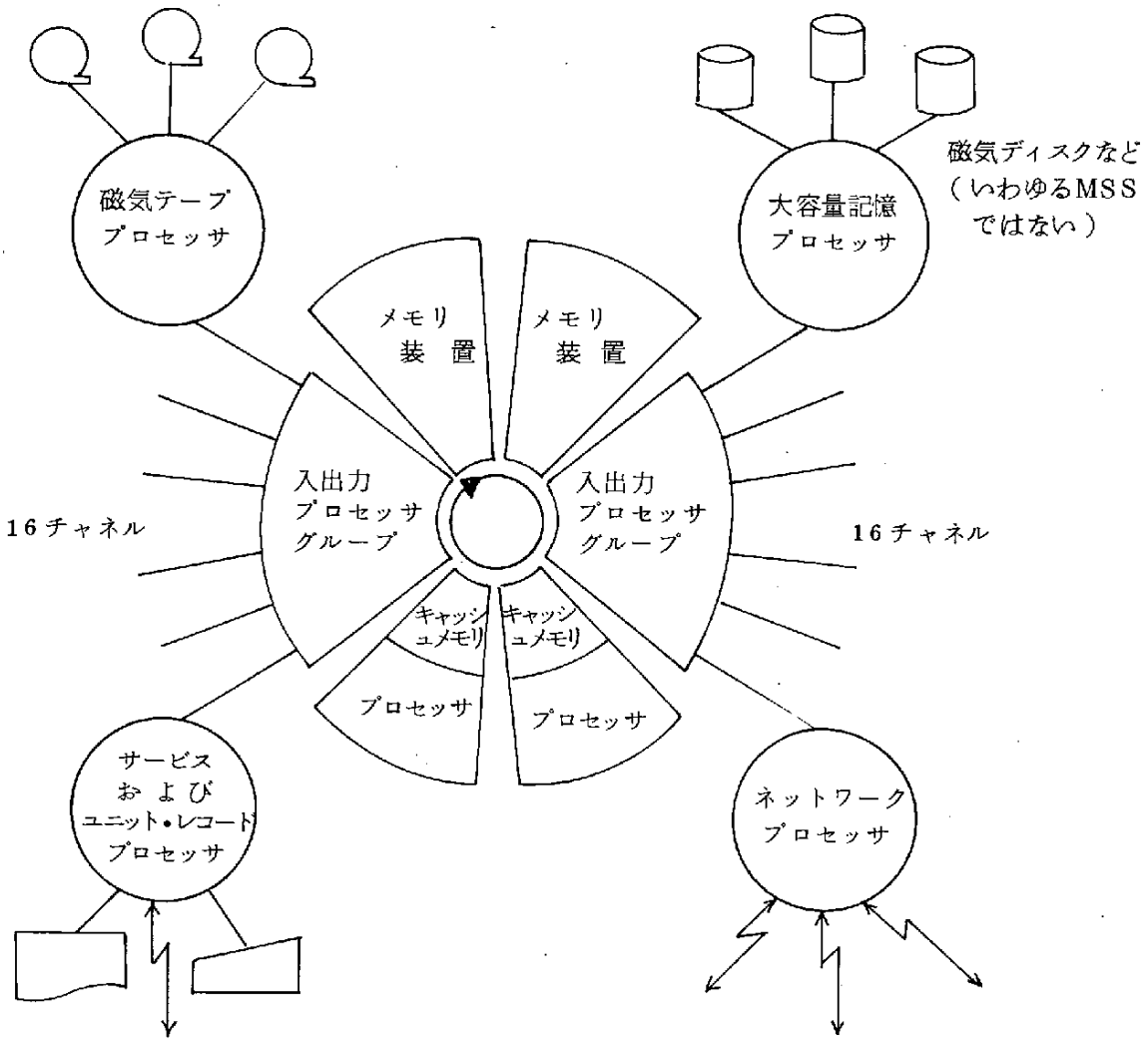


図 3.1　　ＤＰＳ７のシステム構成

GCOS 64 は大部分社内専用の高水準言語でかかれている。この言語はＨＰＬ（Honeywell Programming Language）もしくはＧＰＬ（GCOS Programming Language）と呼ばれており，PL／Ｉのサブセットに組込み関数を追加したものである。この組込み関数は主としてプロセス（process）制御，プロセス間通信（例えば，P,V操作）のためのものである。ＨＰＬはほとんど純粋手続き（pure procedure，再入可能手続きのこと）を生成する。GCOS 64 の手続き（procedure）はほとんど純粋手続きになっており，ソフトウェアの生産性に大きく貢献している。GCOS 64 は，ＨＰＬの他に，アセンブリ言語やMLP（ＨＰＬとアセンブリ言語の中間的なもので，マクロに似ている）も使われている。初期の頃は，メモリをできるだけ節約するためやむを得ずアセンブリ言語を使用したが（ＨＰＬの３倍の効率），現在は，その量を減らすように努めている。また，Level 64はマイクロプログラムを使った機械であり，オペレーティング・システム部分も含めて基本機能のファームウェア化が進められている。

MULTICSに参加した人達がGCOS 64 の開発を担当したので，GCOS 64 の考え方はMULTICSに似ている。例えば，プロテクション（protection）は４つのリング（ring）で実現している。セグメンテーションを採用しているが，ページングの概念はない。

GCOS 64 は約 300 人で開発した。300人を，４つの開発グループ（１つのグループの大きさは50人から70人）と１つのシステム先進ユーザ・グループに分けた。システム先進ユーザ・グループは，先進的な大ユーザと提携し，ギブ・アンド・テイク（give and take）の精神でシステムのカスタマイズを担当した。残りの４つの開発グループのうち，３つはオペレーティング・システム本体（制御プログラム・データ

ベース，データ通信を担当し，1つは言語プロセッサを担当した。GCOS64全体の生産性は，10行／人・日，200行／人・月程度であったと推定されている。この数字はIBMによるOS／360 開発の生産性ほぼ2倍で，条件の違いを割り引いても，高い生産性であったと判断される。

GCOS 64 Release 5は（1980年時点で）200～300万行からなっている。手続き（procedure）の数でいうと，合計約4,300，そのうちHPLでかかれたものが約1,900，アセンブリ言語でかかれたものが約750，MLPでかかれたものが約1,600である。開発担当者を犠牲にしないように，オペレーティング・システム本体の保守は，開発担当者以外の人間が担当する。コンパイラ（部分）はコードの量が少なくバグも少ないので，開発担当者がそのまま保守も担当する。保守は次の3種類に分けて行なっている。

- 安定化部分
- 発展部分
- エラー処理

安定化部分とはオペレーティング・システムのうち，安定させることだけを図る部分である。例えば，Release1では，G100，IBM360，H200／2000のエミュレータを開発したが，それは安定させることだけが保守の目標である。ところが，機能を変更，追加しながら保守しなければならない部分もあり，それを発展部分と呼んでいる。例えば，Release 4の発展部分の保守担当者がRelease 5の開発を担当するようにしている。エラー処理とは，それ以外のすべてのエラー・レポートに対処することを指す。

保守のため，システムのソース・コードは決してユーザに出さない。オペレーティング・システムに関する全責任はCii Honeywell Bull

がとる。ユーザには手を入れさせない。保守コストはプログラムの大きさにもよるが結構高くついている。ユーザから出てくるエラー・レポートの累積件数については，次のような実験式が得られている。

エラー・レポートの累積件数

$$= k\sqrt[3]{\text{システム導入個数の数} \times \text{使用月数}}$$

但し　kは定数

GCOS 64 の開発，テスト（検査）終了時，およびそれ以後の累積修正件数を図 3.2 に示す。

最後に，Release 3 にバージョン・アップしたときの数字を報告しておく。この改訂作業は 1975 年 8 月に開始され，1977 年 1 月に終了した。約 570,000 行が追加され，合計 1,370,000 行にふくらんだ。205 人・年の工数が費やされたが，それは約 900 万ドルに相当すると考えられている。

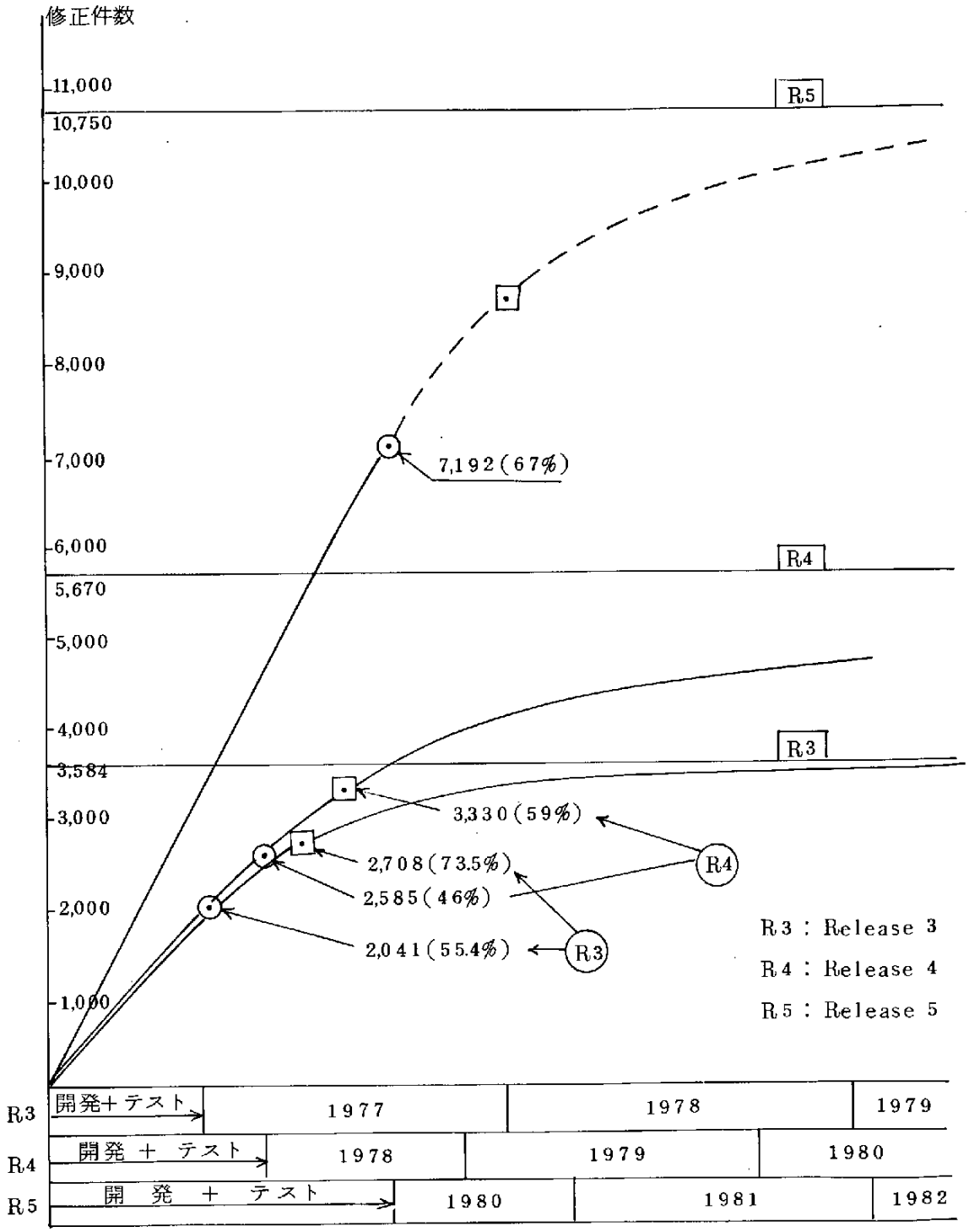


図 3. 2　　GCOS 64 の修正件数

3.2 IRIA－LABORIA

IRIA (Institule de Recherche d' Informatique et d' Automatique)は，フランスのMinistry of Industrial and Scientific Developmentの管轄下にある国立研究所である。この研究所の任務は，情報処理と自動機械に関する研究開発，教育である。

IRIAは，(財)日本情報処理開発協会派遣の調査ではしばしば訪問されているところであり，その沿革，組織等は詳しく報告されているので，ここでは述べない。

IRIAでは，M. G. Kahn氏に会い，最近のIRIAの研究活動，MENTORを用いたPASCAL プログラミング環境等について話をきき，デモを見た。

3.2.1 最近の研究など

1979年のIRIAの研究レポートの一覧を表3.4に示す。とくに，理論的研究を実践的研究に応用することを目指している。

№323 The metric space of infinite trees algebraic and topological properties
A. Arnold, M. Nivat

№324 An Ω (Cn/lg n)½) lower bound on the number of additions necessary to compute O－1 polynomial over the ring of integer polynomials
R. L. Rivest, J. P. Van de Wiele

№325 Identification d' un systême à non minimum de phase par approximation stichastigue･Application en
A. Benveniste, M. Bonnet, M. Goursat, C. Macchi, G.Ruget

№326 Le problème du partage dans I' évaluation des λ-expressions J. J. Levy

№327	A study of flows in an X25 environment J.Labetoulle, G. Pujolle, N. Mikou
№328	Arbres infinis et systemes d'équations B. Courcelle
№329	Update semantics of relational views F. Bancihon, N. Spyratos
№330	A Complete machine-checked definition of a simple programming language using denotational semantics V. Donzeau-Gouge, G. Kahn, B. Lang
№331	Analytic Methods for multiprocessor system modelling A. Kurinckx, G. Pujolle

表3.4（その1） IRIA研究レポート一覧
（1978年9月から1979年9月まで）

№332	Nonlinear programming and nonsmooth optimization —A unification C. Lemaréchal
№333	Proof and synthesis of semantic attributes in compiler definition P. Deransart
№334	Semantics of procedures an algebraic abstract data type M-C. Gaudel, ph. Deschamp, M. Mazaud
№335	Performance evaluation of a cache memory for a mini —computer M. Badel, J. Leroudier
№336	Damaines concrets G. Kahn, G. Plotkin
№337	Problems of clustering and recent advances E. Diday

№ 338 Pattern recognition by a piecewise polynomial approximation with variable joints
Ch. Charles, Y. Lechevellier

№ 339 Hadamard's variational formula a mixed problem and an application to a problem related te a Signorini-variational inequality
Bernadetle Palmerio, A. Dervieux

№ 340 A study of flows in queueing networks and approximate method for solution
G. Pujolle, Ch Soula

表 3.4（その２） IRIA研究所レポート一覧
（1978年9月から1979年9月まで）

№ 341 Distribution of the flows in a general Jackson network
J. Labetoulle, G. Pujolle, Ch. Soula

№ 342 The abstraction process in tho relational model
N. Spyratos, F. Bancilhon

№ 343 Une méthode d'éléments finis mixte pour les équatlons de Van Karman

№ 344 Un algorithme adaptatif de balayage de Péano
J. Quinqueton

№ 345 Conditions nécessaries d'optimalité pour des problémes de contro le optimal associés à des inéquations variationnelles
Ch. Saquez

№ 346 Computing integrated costs of sequences of operations with application to dictionaries
Ph. Flajolet, J. Francon, J. Vuillemin

№ 347 Performance analysis of new file organizations

based on dynamic hash-coding
M. Scholl

№348 Note on the M/M/1 feldback process
J. Labetoulle, G. Pujolle, Ch. Soula

№349 Denotational definition of properties of programs computations
Véronique Donzeau-Gouge

表3.4（その3） ＩＲＩＡ研究レポート一覧
（1978年9月から1979年9月まで）

№350 Etude de codes binaries abeliens modularies autoduaux de petites longueurs
P. Camion, F. Preparata

№351 Models for distributed computing
D. B. MacQueen

№352 Performance of tho HDLC control
J. Labetoulle

№353 Synchronization for distributed systems using a single broadcast channel
J-S Banico, C. Kaiser, H. Zimermann

№354 Analysis of two hash-coded file reorganization policies under steady-state conditions
M. Scholl

№355 Etude numérique de lécoulément d'un fluide viscoplastique de Bingham par une méthode de Lagrangien augmenté
D. Begis

№356 The cube-connected cycles : a versatile network for a parallel computation
F. P. Preparata

№ 357 Throughput analysis of a two-computer system
G. Pujolle, A. Wolisz

表3.4（その4） ＩＲＩＡ研究レポート一覧
（1978年9月から1979年9月まで）

これまで，計算機システムはＩＲＩＳ80を使ってきたが，これからはHoneywell 68（いわゆるMULTICS，2CPU，2,500キロ語，1語は36ビット）を使う。MULTICSは以下の点でソフトウェア開発に向いていると考えている。

- （Kahn氏の研究などで必要な）記号処理ではオン・メモリの考え方が大切であるが，仮想記憶であることは，それに適している。
- 動的連繋（dynamic linking）ができる。
- ファイル・システムがよい（ファイル保護や，ファイル管理が優れている）
- （MULTICSは）PL／Ｉでかかれており，ソース・リスティングを入手し，システムに手を入れることができる。
- 時分割システム（Time Sharing System）である。

ＩＲＩＡは，グルノーブル大学と共同で，このHoneywell 68上でPASCALコンパイラを動かすことを計画している。PASCALコンパイラはすでに2つ作った経験があり，PASCALコンパイラ作成工数は約2.2人・年と予測している。

ＩＲＩＡでは，1973年からPASCALを使っている。その主な理由は当時PASCALがＩＲＩＡ80上でもっともよく動くコンパイラであったこと，DEC（Digital Equipment Corporation）社の機械やマイクロコンピュータ上でも使えたこと，PASCALには副作用（side effect）がないこと，などである。

3.2.2 PASCALプログラミング環境

これまでは，プログラミングをよくするということは，新しい，より強力なトランスレータ（translator）をつくることを意味していた。その考え方からそろそろ脱皮すべきではなかろうか。新しい，より強力なトランスレータをつくることより，充実した，より知的なプログラミング環境を整備することの方がもっと賢いアプローチと言える時期になっていると考えるべきである。

ここでいう（知的）プログラミング環境の提供とは，1つの考え方の枠組，1つのプログラミング・システムの中で，次のようなプログラマの仕事を支援する統合的な（プログラミング）環境をつくり上げることである。

- プログラムをつくる。
- プログラムのドキュメントをつくる。
- プログラム・テキストを変更する。
- プログラムを実行する。
- プログラムを（記号レベルで）デバッグする。
- プログラムの性能を測定し，改良する。
- プログラムに関する種々の事実を記憶し，検索し，推論する。

通常このような仕事はほとんど人手で行なわれるが，その多くは計算機の支援が可能であり，高い効果が期待できると考えられる。計算機はプログラマの肩から覗き込み，プログラマの要求に論理上の矛盾がないかをチェックし，プログラマの事務的作業を手助けし，プログラマの判断を援けるべきだと考える。

このような知的プログラミング環境を作るに際して最初に採りあげるべき技術上の問題は，もっとも基本的な対象であるプログラムを意のま

まに取り扱うことを可能するシステムを，どのようにすればつくることができるかということである。過去10年間に行なわれた計算の理論に関する研究成果をふまえれば，そのようなシステムをつくることはいまや可能であると少なからず自信を抱いている。

プログラムを意のままに取り扱うにはプログラム・エディタ（ program editor ）をつくればよい。テキスト・エディタ（ text editor ）ではないことに注意する。プログラム・エディタは，単なる文字列を取り扱うのではなく，高度な構造をもったテキスト（すなわち，プログラム）を取り扱うエディタである。プログラム・エディタを使う場合，プログラムテキストの取扱いはプログラムの直接の制御の下でなされる。その結果は直ちに表示される。

MENTORはこのような考え方にそってつくられたプログラムエディタであり，それを核にして，PASCAL（プログラミング）のための（知的）プログラミング環境がつくられている。そのデモを見せてもらった。デモを見ながら観察できたこのシステムの特徴には，

- 清書機能をもつ
- 自動字下げ機能をもつ
- プログラムの論理的構造，性質を種々のレベルで表現する
- 他のシステムに移植するためのプログラムの変換（program conversion）を容易にする機能，例えば，プログラムを72桁までに書く，注釈部の指定方法を変えるなどがある。
- デバッグに役立つ
- コマンドはユーザ・マクロであり，拡張が可能である。
- 応答時間が短い

がある。このシステムはPASCALで書かれており，可入可能プログラ

ムになっている。ブートストラッピング（bootstrapping）手法を採用しており，拡張可能なシステムである。実行時のテスト，デバッグ機能について質問したところ，それはこれからの課題ということであった。

3.3 ＩＢＭフランス，教育センター

ＩＢＭフランスの教育センターも，日本アイ・ビー・エムの教育センターと同じように，顧客と社員の教育を行なっている。この教育センターにおける教育は次のように５つの部門に大別され，実施されており，講義が50％，事例研究が50％という感じである。

第１部門　　ＯＳ関係

ＯＳ／ＶＳ，ＭＶＳ，ＩＭＢ，……

第２部門　　ＤＯＳ関係

ＤＯＳ／ＶＳ，ＣＩＣＳ，ＤＬ／Ｉ，……

第３部門　　通信関係

ＳＮＡ，ＶＴＡＭ，8100分散処理システム，……

第４部門　　社内教育

第５部門　　手法，技法コース

ＩＰＴ，ＢＳＰ（Business System Planning），ＤＢ／ＤＣ指向の適用業務設計手法，……

1977年，この教育センターは，通称フレンチ・テクニック（French Technigue）と呼ばれているデータベース／データ通信（ＤＢ／ＤＣ）指向の適用業務設計手法を開発した。この手法はとくにデータベース（ＤＢ）の構築に力点がおかれており，データベースを持ちたいけれども，実際には難しいデータベースの構築をどうすればいいか悩んでいる人のために，データベース構築を手順化したものである。計算機利用上，ソフトウェア開発上，もっとも難しく，使用範囲の広い手法であり，価値の高いものである。ここを訪問した最大の主題はこのＤＢ／ＤＣ指向の適用業務設計手法の調査であり，Alain Nagler 氏から熱意あふれる説明を受けた。この手法の詳細は付録Ａにまわし，本節では，情報システム開発の統一的アプ

ローチだけを述べる。

3.3.1 情報システムの統一的開発

企業などの組織体の情報システムを，組織全体として，統一的に，トップ・ダウンに開発することを考える。この場合，システムの開発は次のような5つのフェーズ（phase，段階）に分けることができる。

- 情報システム全体の設計
- （サブシステムごとの）機能仕様の決定
- データベース／データ通信システムの設計
- 製作，移行等の計画
- 適用業務プロジェクトの開発，導入

以上5つのフェーズについて説明する。

(1) 情報システム全体の設計

まず，組織体のトップが必要とする情報を洗い出し，その情報がどの程度役に立つかを調べ，必要性と問題点をまとめる。サブシステムに展開する。データ処理で解決する部分とそうでない部分を識別する。経済を予測し，データ処理すべき部分について，開発の優先順位を決める。このフェーズでは，BSP（Business System Planning，IBMが開発したビジネス・システムの分析，企画手法）などを利用する。

(2) 機能仕様の決定

各サブシステムごとに，現在の解決策を分析し，将来のシナリオを考え，将来の解決策のモデルをつくり，機能仕様を決定する。そしてその妥当性を検査する。

(3) データベース／データ通信システムの設計

各シナリオを検討し，DB／DCを設計する。すなわち，データ管理をどうするか，データの流れやネットワークをどうするか，処理形態（バッチ／オンライン）をどうするかなどを決定する。計算機化するシステムを詳細に定義し，技術的妥当性を検討する。

(4) 製作，移行等の計画

シナリオを選択し，移行を検討する。プロジェクトを定義し，製作計画をたて，コスト，要員，周囲環境等を評価する。

(5) 適用業務プログラムの開発，導入

プロジェクトごとに適用業務プログラムを開発，あるいは購入し，導入する。

フレンチ・テクニックは上記(2)のフェーズの一部と(3)のフェーズで使える手法であり，1977年に開発され，1978年全ヨーロッパ，ニューヨークで紹介された。このコースは現在（教育センターの）女性1人を含む3人で担当している。

3.4 ＩＢＭドイツ本社

米国ＩＢＭを除けば，ＩＢＭドイツは，日本アイ・ビー・エムとともに，ＩＢＭにおいて優れた業績をあげているところである。その本社はシュツッツガルトの樹々の繁る，静かな，小高いところにある。

ここでは，データ処理部門の生産性に関する（ＩＢＭの）考え方，いくつかのプログラム製品とその狙いなどを調査した。具体的には，P.Kirn氏からデータ処理部門の生産性についてMueller氏からＡＰＬ－ＤＩ，ＱＢＥ，ＡＤＲＳの比較をKlaus Kosalla氏から SIPO/EとELIASについて，Klaus Scheider 氏から計算機を利用した教育について話をきいた。なお，ＩＢＭドイツ本社としてのすべてのセットアップはKnut Wurster氏がやってくれた。

3.4.1 データ処理部門の生産性

(1) データ処理部門の生産性

次の３つの任務を総合的にとらえながら考えなければならない。

- 適用業務の開発
- 導　　入
- 維持（保守）

(2) 計算機システムの能力とデータ処理部門の能力のギャップ

計算機システムの処理能力が急速に伸びる一方で，デー処理部門は，一定の人員で，開発，導入，維持をこなしていかなければならない状況にある。ところが，データ処理化が進み，適用業務プログラムが増えれば増えるほど，その維持にさかなければならない能力が大きくなり，絶えず要求のある新規適用業務プログラムの開発にまわせる（データ処理部門としての）能力は低下の一途を辿っている。そのようす

を図 3.3 に示す。

(3) プログラムの数と走行時間

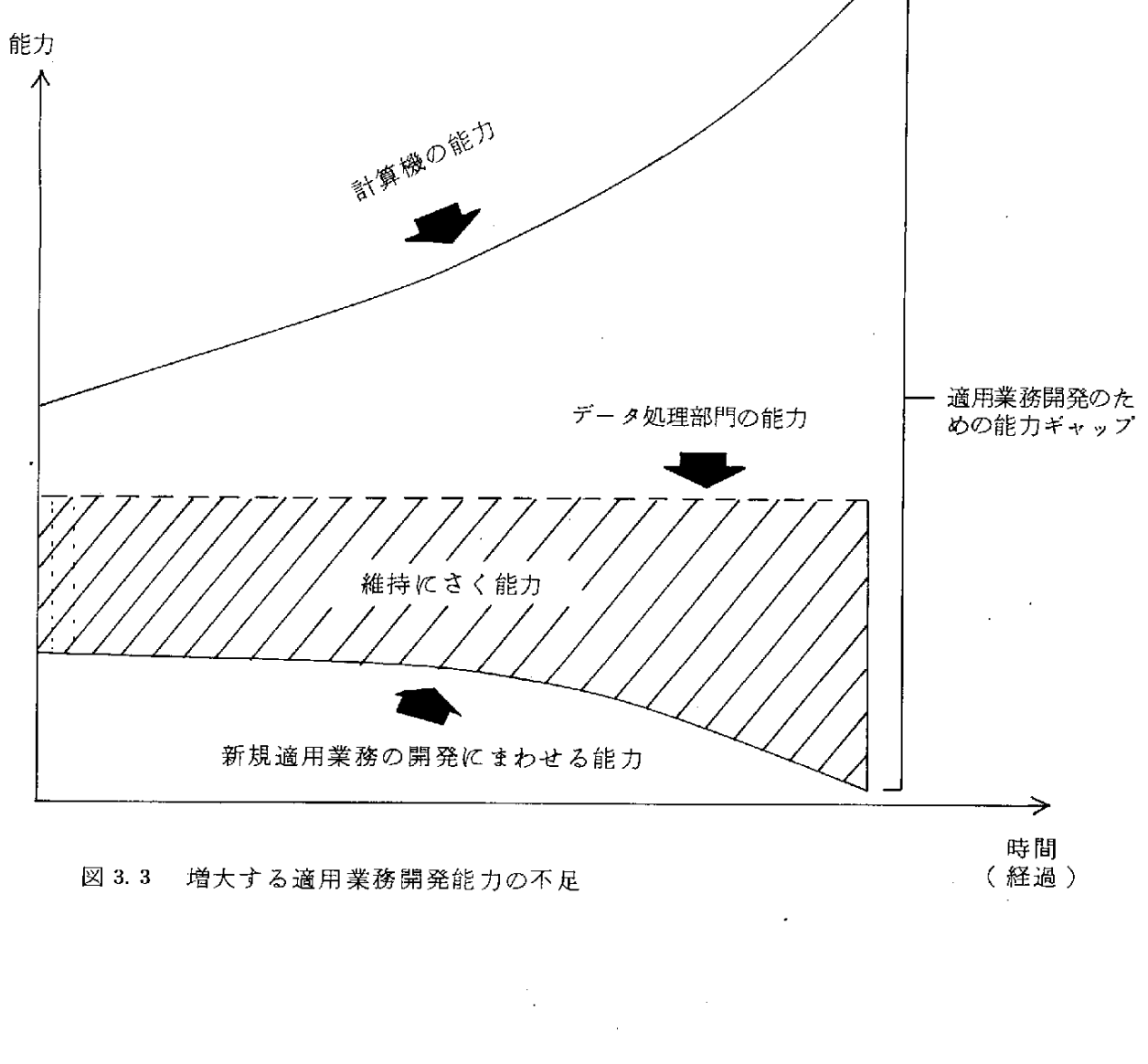


図 3.3　増大する適用業務開発能力の不足

ある調査によれば，図 3.4 に示すように，全プログラムのうち50％のプログラムは，（全プログラムの）走行時間全体の２％しか占めていず，逆に走行時間全体の50％を食っているのは，高々２％のプログラムに過ぎない，という結果がでている。ハードウェア価格の著しい低下とも考え合わせると，一般に，「プログラムの作成において，その走行効率を高めることが第一義的に要求されることはほとんどない」と言ってよい。

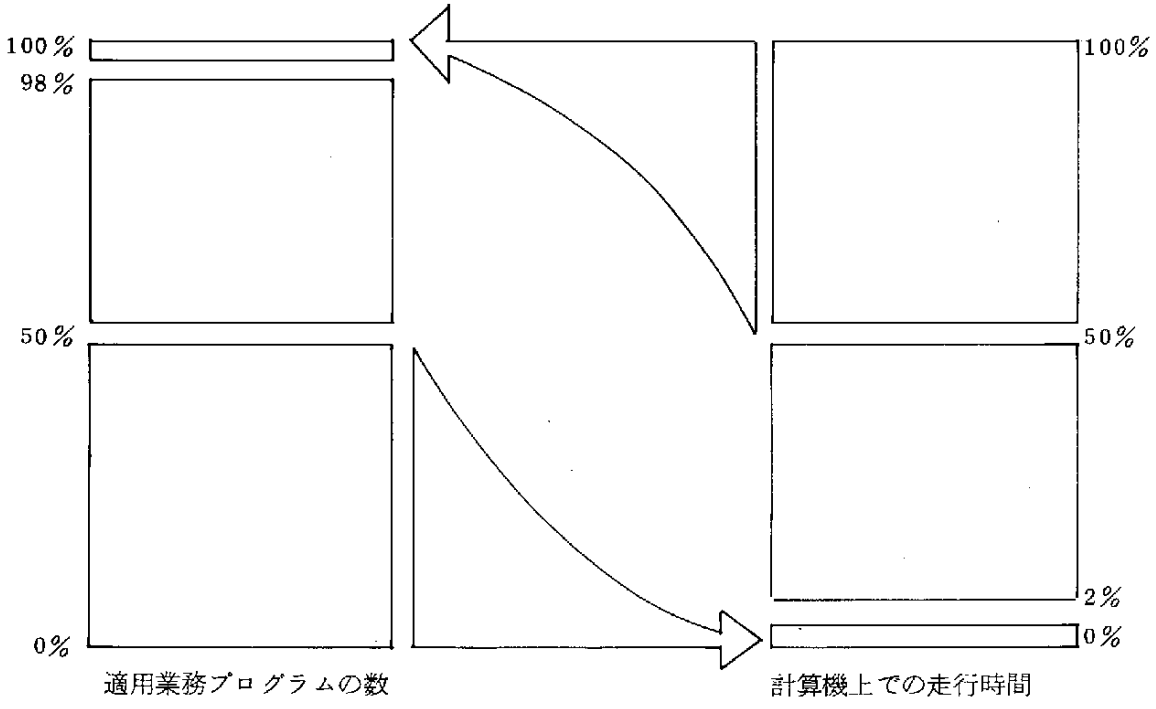


図 3.4　適用業務プログラムとその走行時間の関係

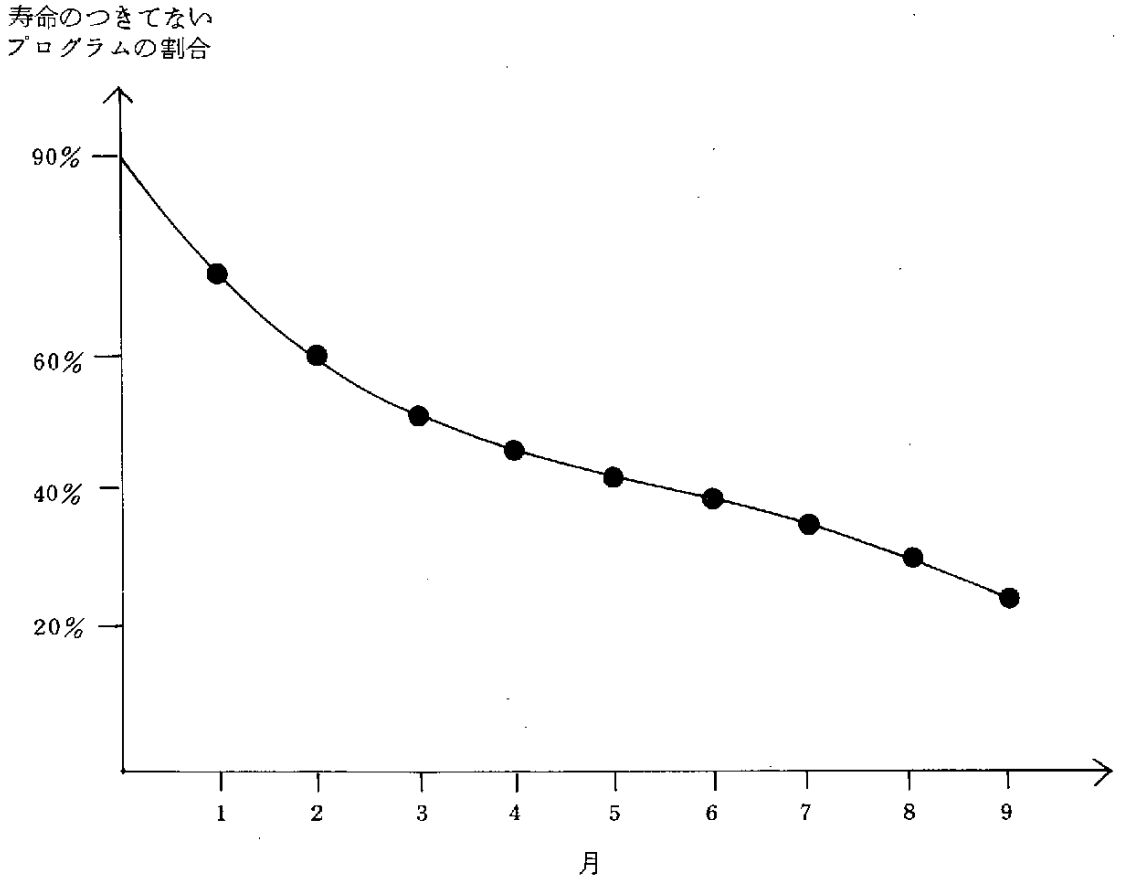


図 3.5　プログラムの寿命

(4) プログラムの寿命

図 3.5 はプログラムの寿命を調査した結果である。開始時に走行したプログラムが，（何度か走行したのち）再び走行されることがなかった場合，そのプログラムは，最後の走行時点で寿命がつきたと考えられる。この図からはプログラムの平均寿命は 5 か月弱と判断される。

(5) 開発コストと走行コスト

図 3.6 は図 3.4 とほとんど同じである。図 3.4 から開発(総)コストの50%を占めるプログラムは，全走行コストの 2 %しか食っていないと考えても差支えない。

(6) 開発の考え方

上記の50%のプログラムが合計 530万ドルをかけて開発され，CPU 時間の 2 %，例えば4万1千ドルを費していたとする。このとき，開発コストを10%削減したため，ＣＰＵ時間が200 %に増加したとしても，

開発削除コスト	53 万ドル
ＣＰＵコスト	8 万 2 千ドル
差引削減コスト	44 万 8 千ドル

となる。ハードウェア・コストが安く，人件費が非常に高い現在，ＡＰＬなどの言語で生産性を上げてプログラムを開発することが（，走行コストがたとえ100倍に増えても，）コストの節減につながる（図 3.参照）

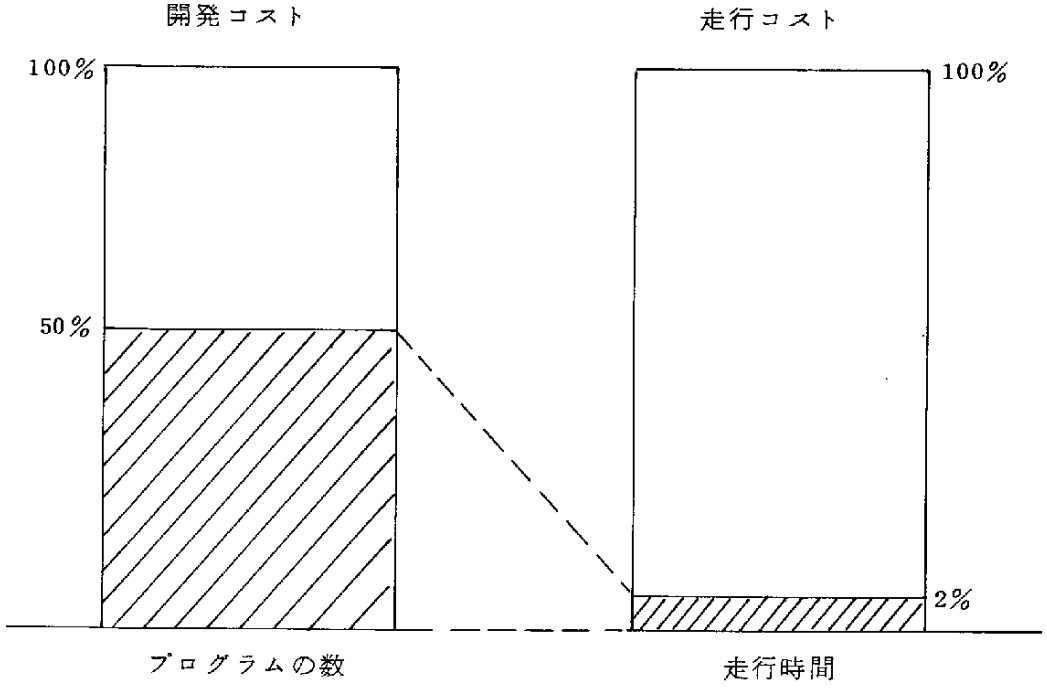


図 3.6　開発コストと走行コストの関係

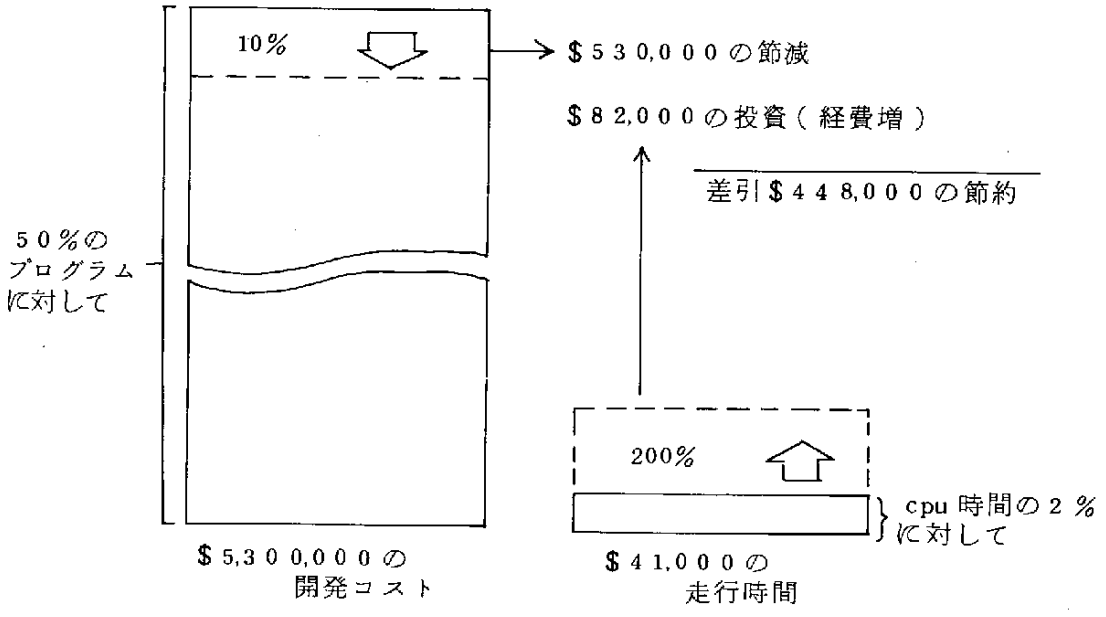


図 3.7　50%部分のプログラムの開発コストを10%節減する効果

(7) 適用業務開発の要求に応える道

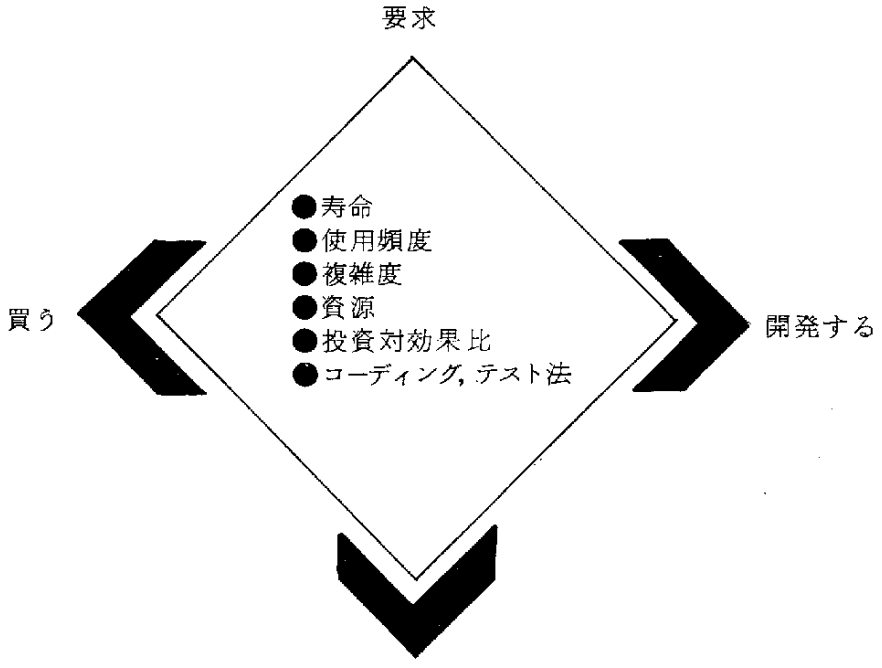
適用業務の要求があっても，「（データ処理部門が）開発する」ことだけが道ではない。その適用業務プログラムの寿命，使用頻度，複雑度，（開発のための資源の状態，投資対効果比，コーディング，テスト法などを総合的に判断して，「買う」道も，「（エンド・ユーザに作ってもらい，それを）支援する」道も検討してみる必要がある。（図 3.8 参照）

(8) 新しい方法，処理形態，コーディングによる解決

前記図 3.4 のように走行するプログラム群のそれぞれに対しては，図 3.9 のように，開発アプローチ，処理形態，コーディング・テストを使い分けることが，総合的にみて，）生産性を向上させ，コストの節減につながる。

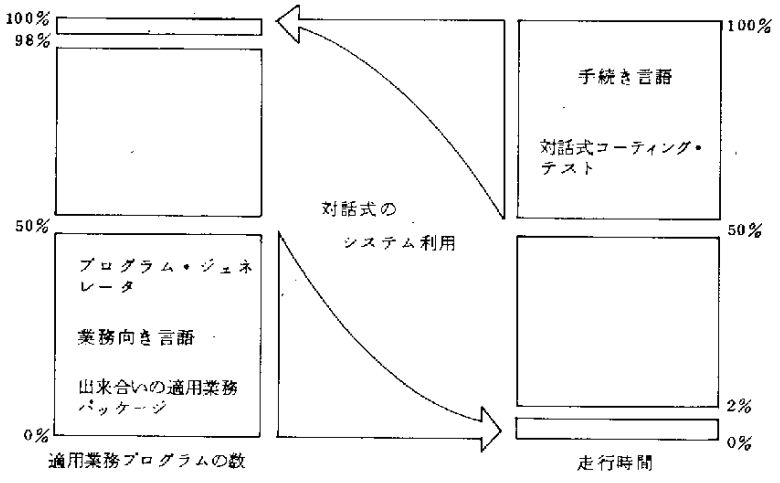
(9) 開発することによって要求に応える道には

図 3.8 の（適用業務プログラムの）開発のためにＩＢＭは図 3.10 に示すようなツール類を用意しており，これらによってコストの節減を達成することができる。



買う＝「適用業務の要求に対する完全な解になる」
支援する＝「共通的なものはデータ処理部門，そうでないものは現場で」
開発する＝「従来のようにデータ処理部門で開発する」

図 3.8　適用業務の（開発）要求に応える道



走行時間の長い２％のプログラムに対しては：
- 処理形態，開発方法
- 対話式コーディング，テスト
- プログラム・チューニング

を考える。

走行時間の短い５０％のプログラムに対しては：
- 出来合いの適用業務パッケージ
- プログラム・ジェネレータ
- 利用の支援

を考える。

図 3.9　生産性向上のための基本的考え方

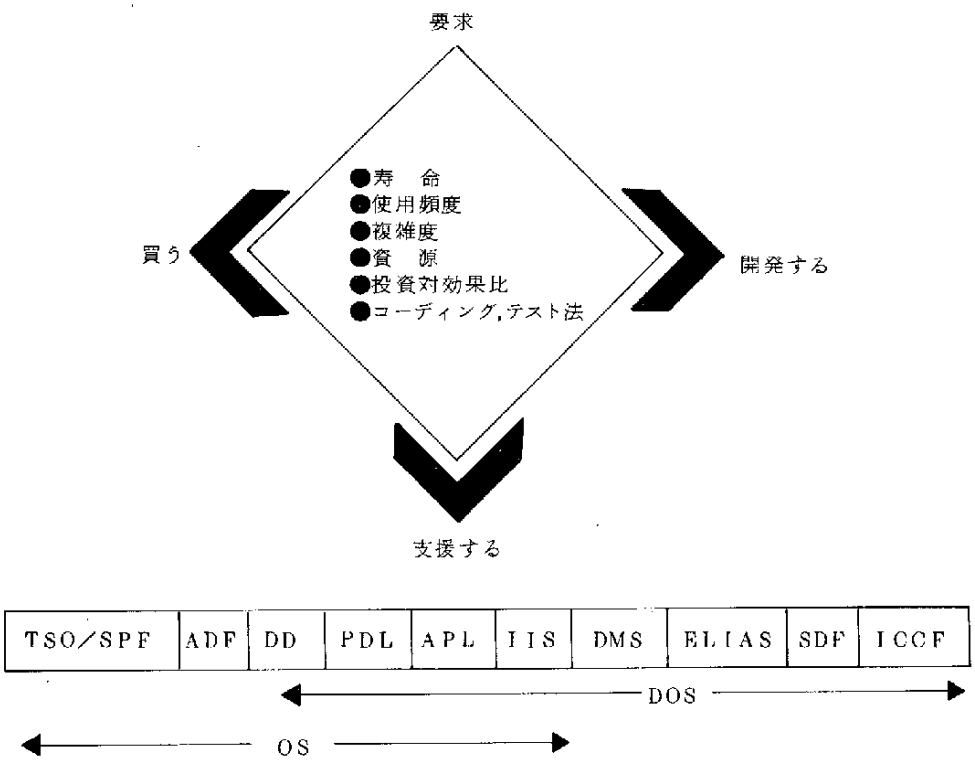


図 3.10　データ処理部門による従来型の適用業務開発に対して
IBMが用意しているツール

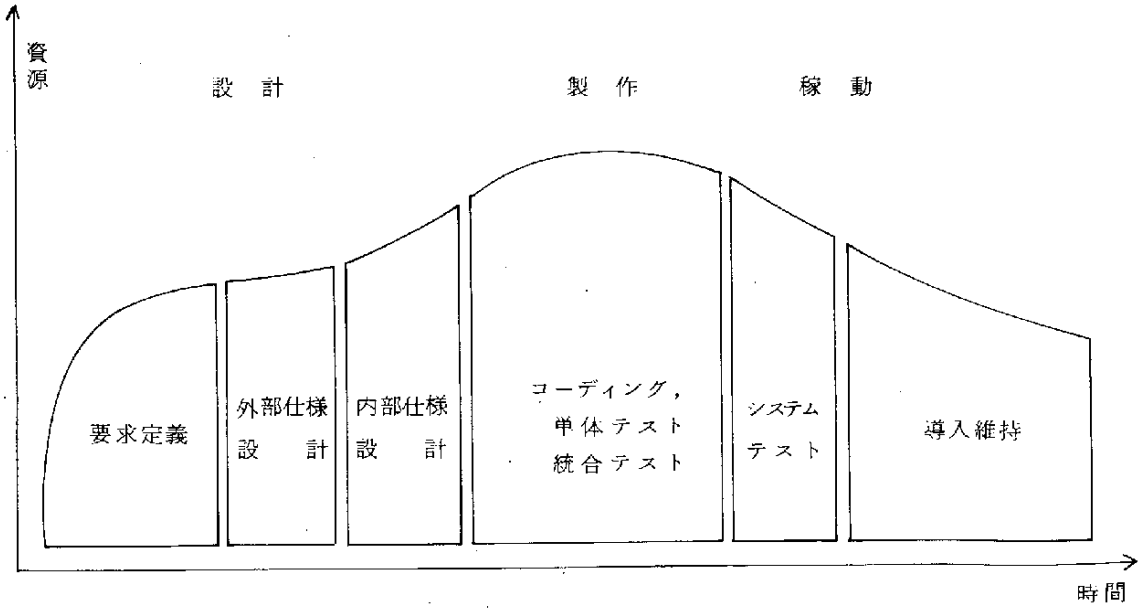


図 3.11　適用業務の開発サイクル

(10) 適用業務の開発サイクル

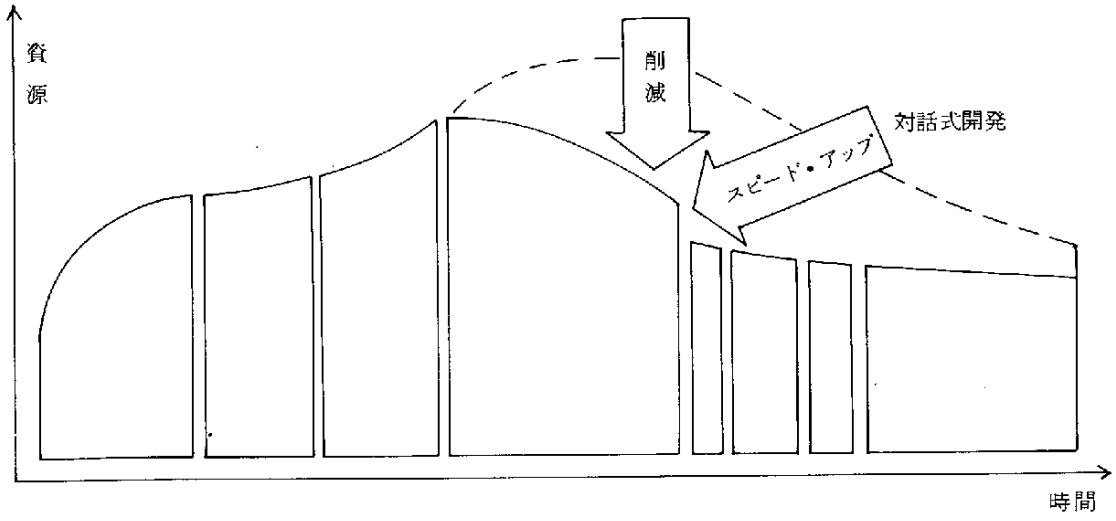
適用業務開発サイクルと，サイクルの各過程で必要となる（開発）資源の大きさを図 3.11 に示す。

(11) プログラム製作時の生産性向上

図 3.12 は，図 3.11 のプログラム製作（コーディングおよび単体テスト）過程から後の過程で，新しい開発技法，道具，形態を採用した場合，資源がどのように節減されるかを表わしている。

(12) 買うことによって要求に応える道

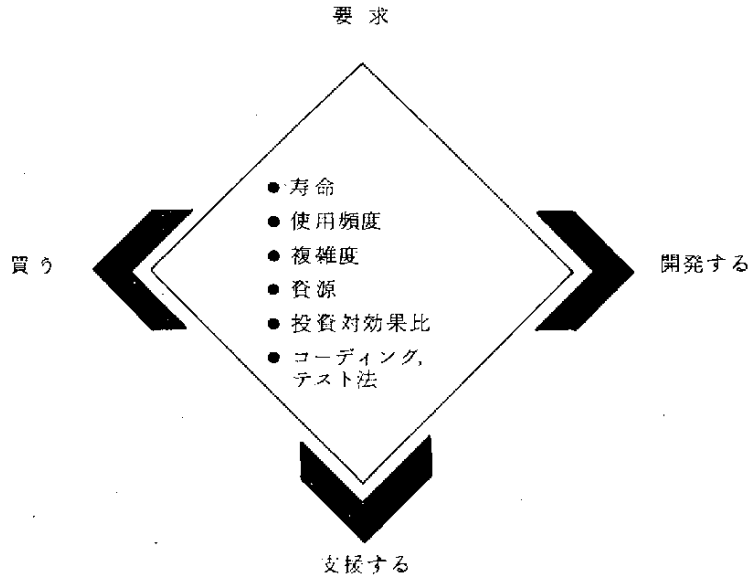
適用業務（プログラム・）パッケージを買うことによってもコストを節減することができる。そのためにＩＢＭが提供している適用業務パッケージの一例を図 3.13 に示す。



開発する場合，以下によってプログラム・フロー／システム利用を最適化する

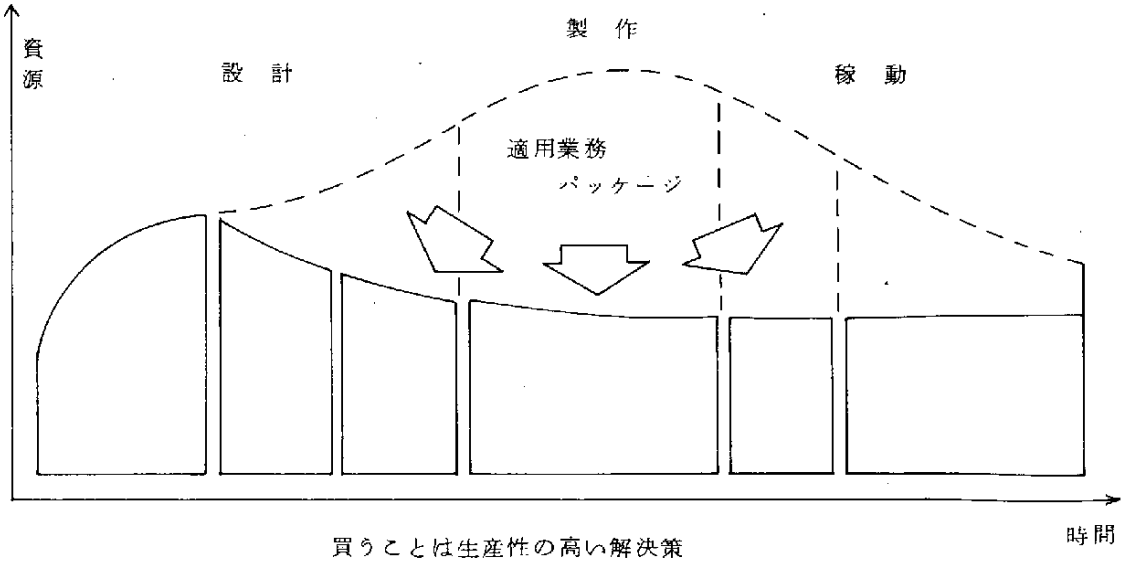
- 対話式開発
- データベース／データ通信システム
- データ・デリバリ環境など

図 3.12　生産性の高いプログラム開発



分野ごとの適用業務	APL	APL EPLAN	ATMS	DCF	IIS + teachware	INTER PERS

図 3.13　既製の標準ソフトウェア・パッケージを買う解決策をとるカスタマのためにＩＢＭが用意しているパッケージ



買うことは生産性の高い解決策
早期入手による早期稼動，早期効果

図3.14　適用業務パッケージを買うことによる効果

(13) 既製のプログラムを買うことの効果

従来のプログラム開発に比べて，既製のプログラムを購入することによる資源節減効果を図 3.14 に示す。このように開発コストが節減できるだけでなく，適用業務を早期に稼動させることによる効果利益も大きいことを忘れてはならない。

(14) 「買う」ことがもっとも生産的である。

プログラムをつくる場合と買う場合で，コスト効果が時間とともにどのように累積されていくかを図 3.15 に示す。買うことがいかに利益が大きいかよく分る。

(15) エンド・ユーザを支援することによって，要求に応える道

エンド・ユーザによる適用業務の開発を支援するためにＩＢＭが用意しているツール類を図 3.16 に示す。

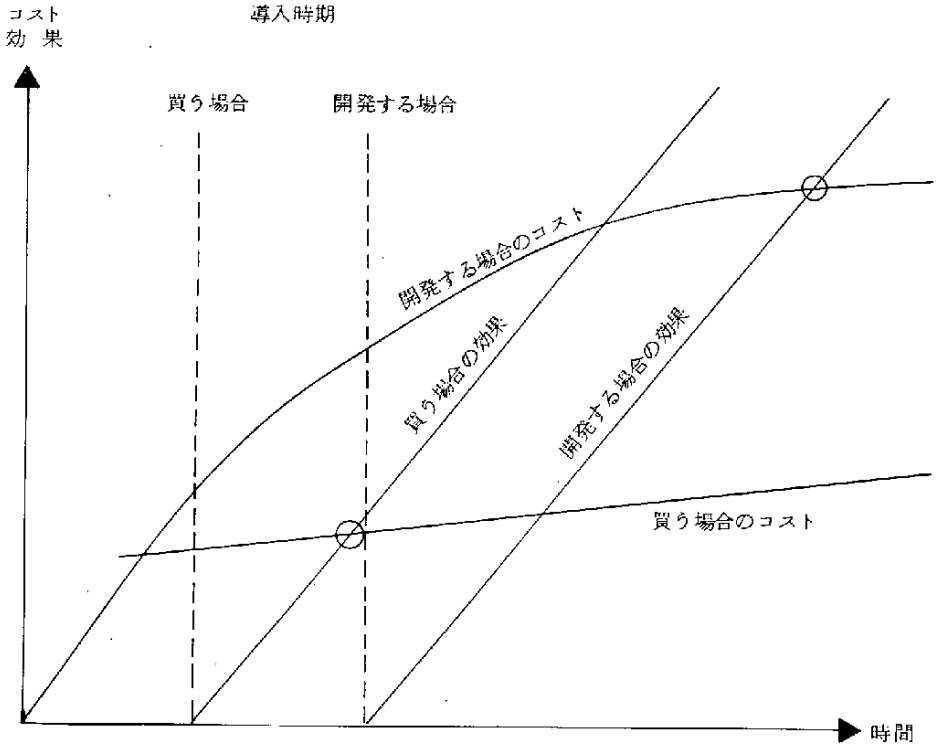
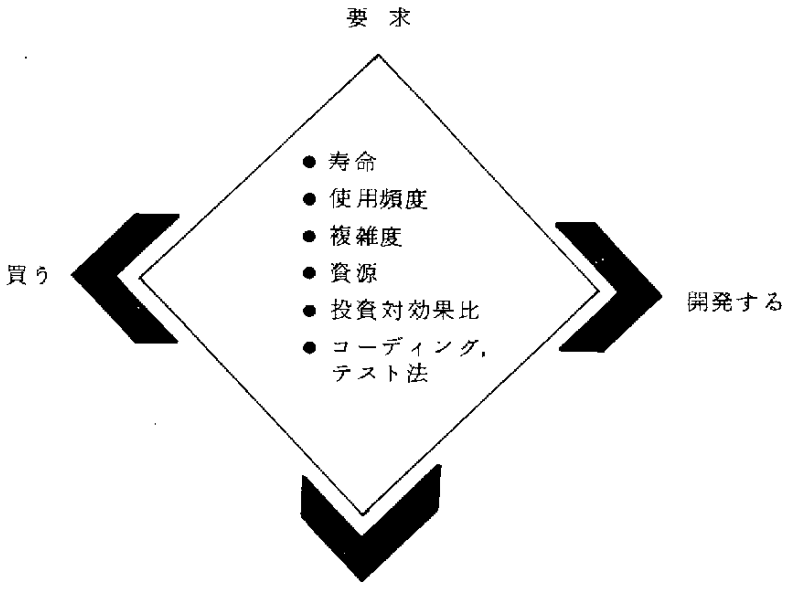


図 3.15　　買う場合と開発する場合の累積コスト，累積効果の差



（エンド・ユーザによる製作）

APL	APLDI	FORTRAN	APL-ADRS	LIS	STAIRS	APL-EPLAN	INTER PER S PDL

図 3. 16　　エンド・ユーザをデータ処理部門が支援しやすいように
ＩＢＭが用意しているツール類

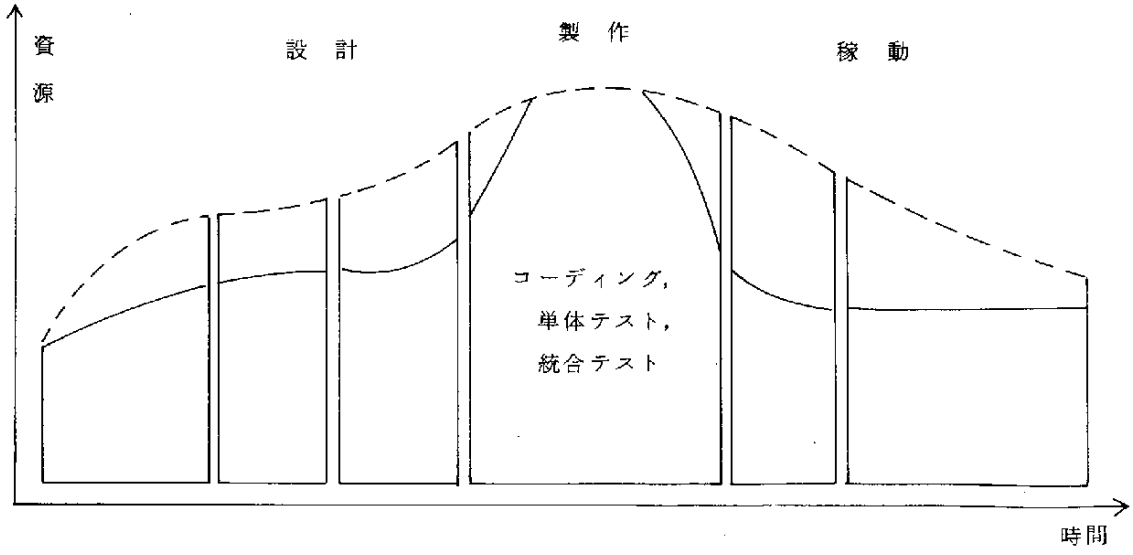


図 3. 17　　エンド・ユーザを支援しながら開発する場合の効果

(16) エンド・ユーザをデータ処理部門が支援することによる効果

エンド・ユーザを支援することによって適用業務の要求に応えることは，「買うこと」の次によい解決方法であり，図 3.17 のような資源の節減効果がある。このようなエンド・ユーザの支援がデータ処理部門の任務に加わることは，データ処理部門に従来とは違った機能，能力の配分が要求されていることを意味する。

(17) 利用サービス・センター

エンド・ユーザを積極的に支援するために，従来のデータ処理部門の中に利用サービス・センターを設けることを（ＩＢＭは）提案している。この構想はもともとＩＢＭカナダで提案され，一般には「情報センター」と呼ばれており，すでに大きな効果を挙げている。利用サービス・センターが果すべき機能等を図 3.18 にまとめた。

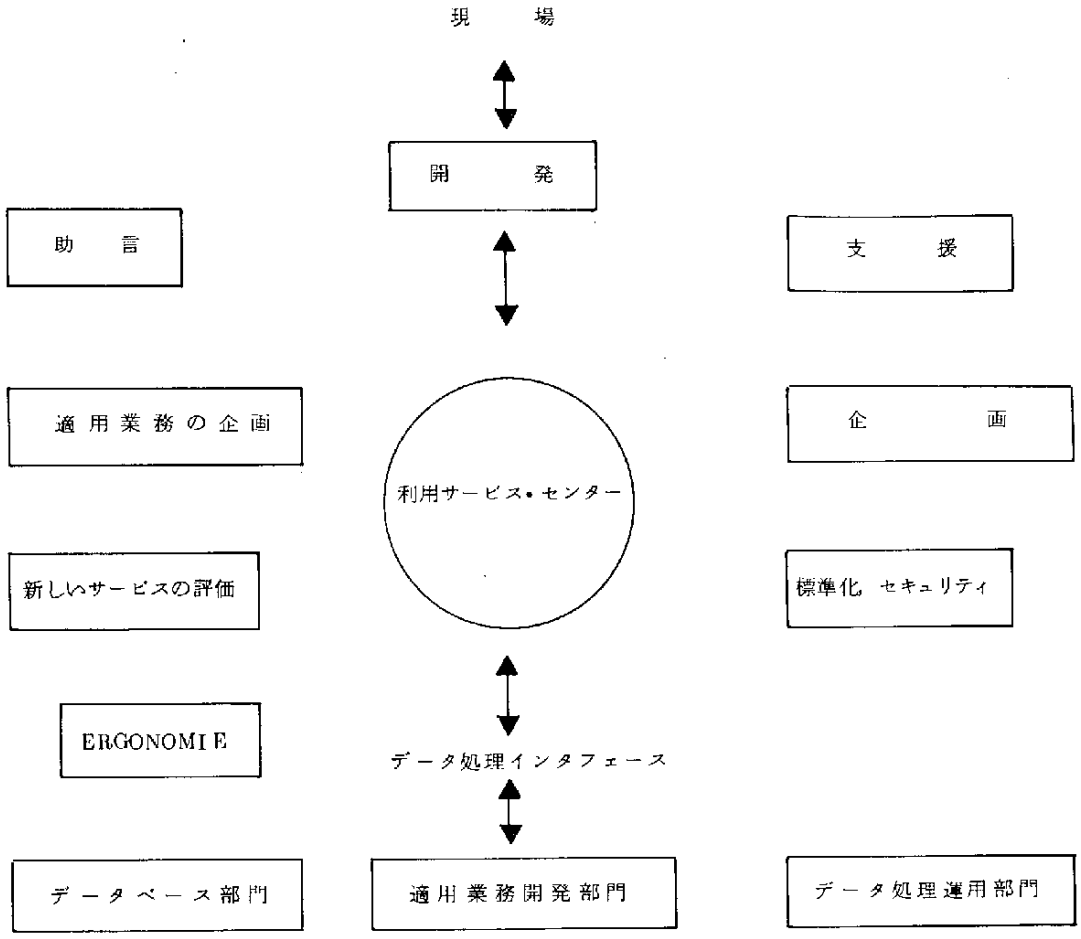


図 3. 18　利用サービス・センターの位置付けと役割り

3.4.2 APL—DA, QBE, ADRSの比較

データ処理には，データベースを操作するという見方と，データベースを情報そのものとして取扱うという見方がある。データ処理部門では，データベースをDL／Iなどで操作するという見方をとるが，エンド・ユーザなどによる個人計算システムでは，データベースを関係表の集まりと見る方が適切であろう。関係ファイルには正規形と非正規形がある。

IBMはデータベースを情報そのものと見る立場に立った製品として，APLDI（APL Data Interface），QBE（Query by Example），ADRS（A Department Reporting System）の3つをサポートしている。APLDIとADRSはともに，ベース言語がAPLで，非正規形ファイルを使っている。一方，QBEはPL／Iをベース言語とし，正規形ファイルを使っている。

APLDIの使用例を図3.19と図3.20に示す。

```
DATEI?      auftrag
SELEKTION?  prod = 8100 ∧ menge > 10
FUNKTION?   sum
KONTROLLFELDER?  filiale
SUMMENFELDER?  umsatz, menge > 0 ⊂ anzahl
```

⇓

FILIALE	UMSATZ	ANZAHL
BERLIN	3,850	3
HAMBURG	12,535	10
KOELN	5713	7

図3.19　APLDIの使用例（その1）

DATEI： umsatz
SELEKTION： prod = 5813
FUNKTION： cross
ZEILEN： gs
SPALTEN： menge < 100, 100−800, > 800

GS	≤100	100−800	>800	TOTAL
60	→37	→20	→43	
	↓30	↓15	↓12	↓21
65	→22	→10	→68	
	↓27	.....	.....	↓46
70	.....	.....	.....	
	.....	.....	.....	.....
.....	.....	.....	.....	.....
TOT	→32	→15	→53	100

図 3.20　APLDIの使用例（その２）

APLDT　APLDIでは

SUM, CROSS, AVER, COUNT, FREQ, PRINT, REG, REPORT, LNK, SUB, TABL, TOP, DATA,……などの30のコマンドが使える。

ＱＢＥはあまりにも有名であるし，紙面の関係もあり，ここでは述べない。

ＡＤＲＳは主として（関係ファイルから）報告書を作成するためのプログラムである。ＡＤＲＳの機能を図 3.21 に示す。

ＡＰＬＤＩ，ＱＢＥ，ＡＤＲＳの比較表を表 3.5 に示す。基本的に，ＡＰＬＤＩはデータ分析に，ＱＢＥはデータ照会に，ＡＤＲＳはテーブル処理に向いている。

3.4.3 SIPO／EとELIAS

SIPO／E（System Installation Productivity Offering／Extended，拡張システム導入の生産性向上機能）はシステムＩＰＯ（ＳＩＰＯ）の拡張版である。ＳＩＰＯがシステム導入の生産性，すなわち，データ処理要員，機械の生産性を主目的にしているのに対し，SIPO／E

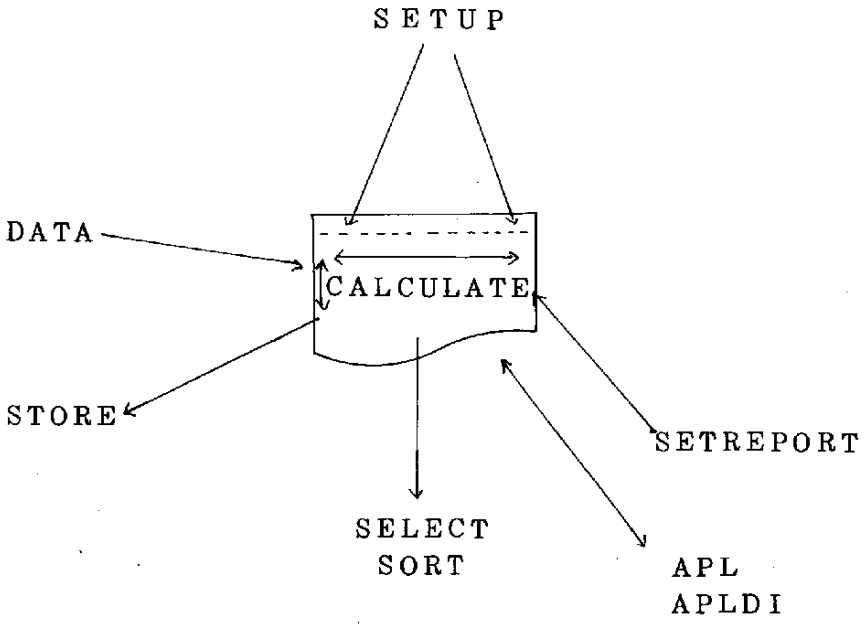


図 3.21 ADRSの機能

機　　能	ADRS	APLDI	QBE
データの選択	＋	＋	＋
テーブルの結合	－	0	＋
データの圧縮	0	＋	＋
レポート	＋	0	0
自由な計算	＋	0	0
統計	－	＋	0
拡張性	＋	0	－

適用業務			
企画	＋	－	－
管理	＋	－	－
報告	＋	0	0
照会	0	＋	＋
処理されるデータ	ユーザ，DP	DP	DP
使い方	汎用テーブル処理	データ分析	照会
＋：得意，－：不得意，0：普通			

表 3.5　ADRS，APLDI，QBEの比較

は，導入時はもちろんのこと，導入時の（システムの）管理，運用の生産性までも向上させることを目的としている。SIPO／EとSIPOの大きな違いは，ダイアローグ・マネージャー機能が加わったことである。

SIPO／Eは，IBM4,300とシステム／370の上で動く。システムを即時に導入し，使用するための既製システムである。SIPO／E自身に導入手順がふくまれており，オンライン，対話式の説明／案内によって，大量の解説書を参照しなくても導入操作ができる。

SIPO／Eを採用する利点には次のようなことがある。

(1)　構成済みのシステム（ハードウェア，ソフトウェアともに）である。

(2)　対話式機能を用いた画面表示による容易な利用が可能である。対話式機能としては，CMS（Conversational Monitor System）またはVSE／ICCF（Virtual Storage Extended／Interactive Communication Control Facility）を使っている。

(3)　システムが生成済みであり，導入が容易である。JCL（Job Control Language）は準備済みであるし，PTF（Program Tewporarily

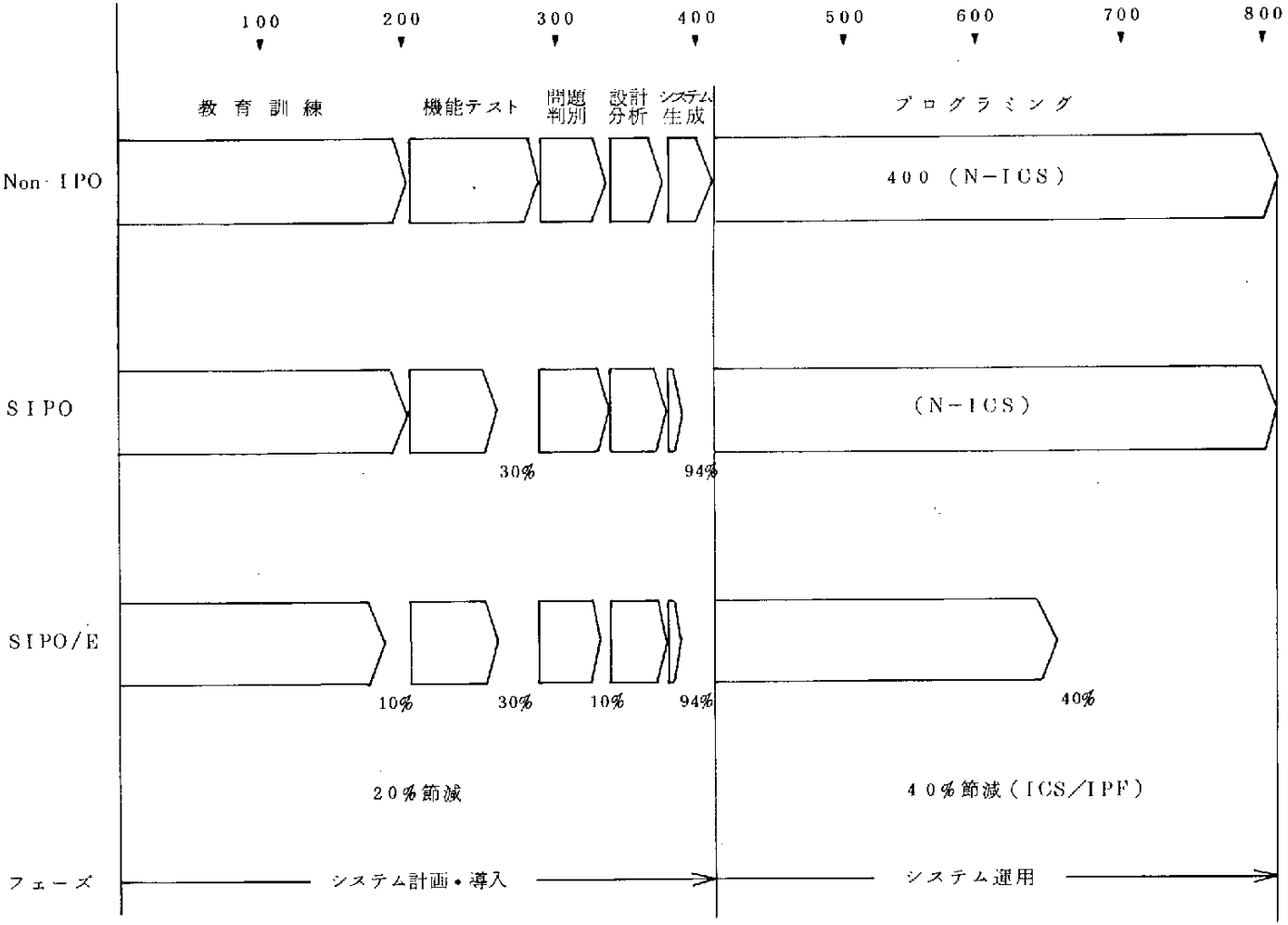


図3.22　SIPO/Eの生産性（見積例）　　単位：人・日

Fixed ）も適用済みである。

(4) 統合テストが済まされており，信頼性が高い。

(5) もっとも新しいレベルのプログラムを集めている。

(6) 参考情報が与えられるので，導入のための準備，教育の時間が短縮される。導入先の仕様に合わせてチューニングする。

以上により，

- 導入生産性の向上
- 取扱いが容易
- 信頼性の向上

などが達成される。

ＳＩＰＯとＳＩＰＯ／Ｅの比較を表 3.6 と 3.7 に，ＳＩＰＯ／Ｅによる生産性向上の見積り例を図 3.22 に示す。

ＥＬＩＡＳについては，ＩＢＭイタリア，ローマ・プログラム・プロダクト開発センターでより詳細な調査がなされ，その結果を 3.8.2 節と付

項　　目	ＳＩＰＯ	ＳＩＰＯ／Ｅ
目　　的	システム導入生産性の向上	システム導入のほか ＋操　作 ＋エンド・ユーザ の生産性向上
内　　容		
• 基本要素	ＳＣＰ	ＳＣＰ＋基幹製品（SPL）
• 選択要素	ＰＰ，ＦＤＰなど	ＳＣＰ＋エンド・ユーザ構成要素
• ドキュメント	部分的	拡　充
• ツール／エイド	ユーティリティ＋ＪＣＬ	対話式機能
• 例　　示	ＳＣＰのみ	すべてのSIPO構成要素

表 3.6　ＳＩＰＯ／ＥとＳＩＰＯの比較（その１）

項　　目	ＳＩＰＯ	SIPO／E
構　　成	最少SCPとハードウェア	指定済運用環境
用意された機能	生成済の操作可能なシステム	生成済で チューニング済の 総合的なシステム機能
ユーザ•インタフェース ／援助機能	ＳＣＰ機能とＩＰＯで ドキュメンテーション	総合的な対話式機能 • 対　話 • ツール • プロンプタ • 説明•案内 • ドキュメンテーション
生産性向上の利益	システム•プログラマ 適用業務プログラマ DP 部門の業務	システム•プログラマ 適用業務プログラマ システム操作員 エンド•ユーザ DP 部門の業務

表 3.7　ＳＩＰＯ／ＥとＳＩＰＯの比較（その２）

録Dで報告してあるのでここでは割愛する。

3.4.4 計算機を利用した教育

計算機を教育の場に応用することはふるくからやられており，目新しいことはほとんどない。しかし，教育現場でどの程度実用になっているかを調べると，心もとない限りである。ＩＢＭでは，計算機を利用した教育用のプログラム製品としてＩＩＳ（Interactive Instruction System）を提供している。もっとも肝心なのはＩＩＳ上で動くティーチウェア（teachware）をテーマごとに開発しなければならないことであり，一般に，それはＩＢＭなどメーカの仕事ではなく，教育の場近くの課題であろう。

ＩＢＭではデータ処理教育に的を絞ったティーチウェアを開発しているとのことである。計算機を用いた教育のテーマとしては，計算機技術はもっとも適しているはずである。シミュレーションによるオペレータの訓練など，効果の高いテーチウェアが開発されているようであったが時間がなくなってしまい，詳細を調べることはできなかった。

3.5 ＩＢＭドイツ，ボブリンゲン研究所

ボブリゲン研究所は1953 年に設立された，世界中の全ＩＢＭに対して製品開発の任務を負う研究所である。ここでいう全ＩＢＭとは，ＵＳ ＩＢＭ（米国ＩＢＭ），ＡＦＥ，ＥＭＥＡのすべてとの意である。ＡＦＥ（America and Far East ）とは，（米国ＩＢＭを除いた）アメリカ地区にあるＩＢＭと極東地区にあるＩＢＭの上位組織で，日本アイ・ビー・エムはＡＦＥに所属する。同様に，ＥＭＥＡ（Europe，Middle East and Africa）は，ヨーロッパ地区，中東地区，アフリカ地区にあるＩＢＭの上位組織で，ＩＢＭドイツ，ＩＢＭフランス，ＩＢＭイタリアはすてＥＭＥＡに所属する。日本アイ・ビー・エム藤沢研究所もボブリンゲン研究所と同様，全ＩＢＭに対して製品開発の任務を負っている。

ボブリンゲン研究所はＩＢＭドイツ本社から車で15分位のところにあり，現在約1,500人が働いている。そのうち約90％が技術関係で，研究員は（1,500人のうちの）およそ70％，大学卒以上はその（70％のうちの）70％ 強である。

ボブリンゲン研究所の任務は，基本的に，中下位機種の製品開発である。具体的には，ＩＢＭ4331などプロセッサ，64キロビット／チップのメモリやＩＩＬ（Integrated Injection Logic）など半導体技術銀行端末など特殊製品，ラインプリンタ，さらには，ＤＯＳ／ＶＳＥ（Disk Operating System／Virtual Storage Extended）などオペレーティング・システム，ＳＩＰＯ（System Installation Productivity Offering）などソフトウェア製品を開発している。ボブリンゲン研究所で以前開発され高い評価を得た製品にシステム／360モデル20，システム／370モデル115，125などがある。また，製品開発や製品の品質保証に用いるツール（道具）類の開発もボブリンゲン研究所の任務の１つである。

ここでは，Uwe Wehlers 氏とMax Briner の両氏に会い，議論し，所内を視察した。Wehlers 氏から研究所の組織の紹介があり，Briner 氏から64キロ・ビット／チップのＬＳＩメモリ，ＩＢＭ4331－2型プロセッサのアーキテクチャなどについて話があった。

64キロ・ビット／チップのＬＳＩメモリは，512キロ・バイト単位のカードに実装され，4331 などで使われている。このメモリの生産量はＩＢＭが最高であろうとのことであった。

4331 －2型プロセッサのアーキテクチャについては，次にあらためて節を設けて報告する。

3.5.1　ＩＢＭ4331－2型プロセッサのアーキテクチャ

プロセッサ4331 －モデル2型は今年発表された最新鋭の中型計算機で，昨年発表された4331 －モデル1型と，上位機種プロセッサ4341 の中間の性能をもち，幅広いシステム構成が可能である。基本記憶モジュール（ＢＳＭ，Basic Storage Module，主記憶にあたる）は，最大4ＭＢ（メガ・バイト）まで実装できる。高速ブロック多重チャネルの提供により，ＩＢＭ3830 または3880 経由で，高速大容量ディスクＩＢＭ3330，3333，3340，3344，3350，3370，3375 などを接続できる。

システム構成は，図3.23 に示すように，基本的にはモデル1型と同じであるが，性能や機能の面で次のような強化，拡張が図られている。

- ＢＳＭは，1ＭＢから，1ＭＢ単位で，4ＭＢまで拡張可能に変更。
- 内部処理能力は，モデル1型の約1.8 ～ 2.3 倍（モデル1型で約0.25 ＭＩＰＳ，ＭＩＰＳはMillion Instructions Per Second）
- 統合チャネル（ＩＣ，Integrated Channel）の転送能力は，モデル1型の3.33 ～ 3.67 ＭＢ／sec（毎秒3.33 ～ 3.67 メガ・バイト）に対

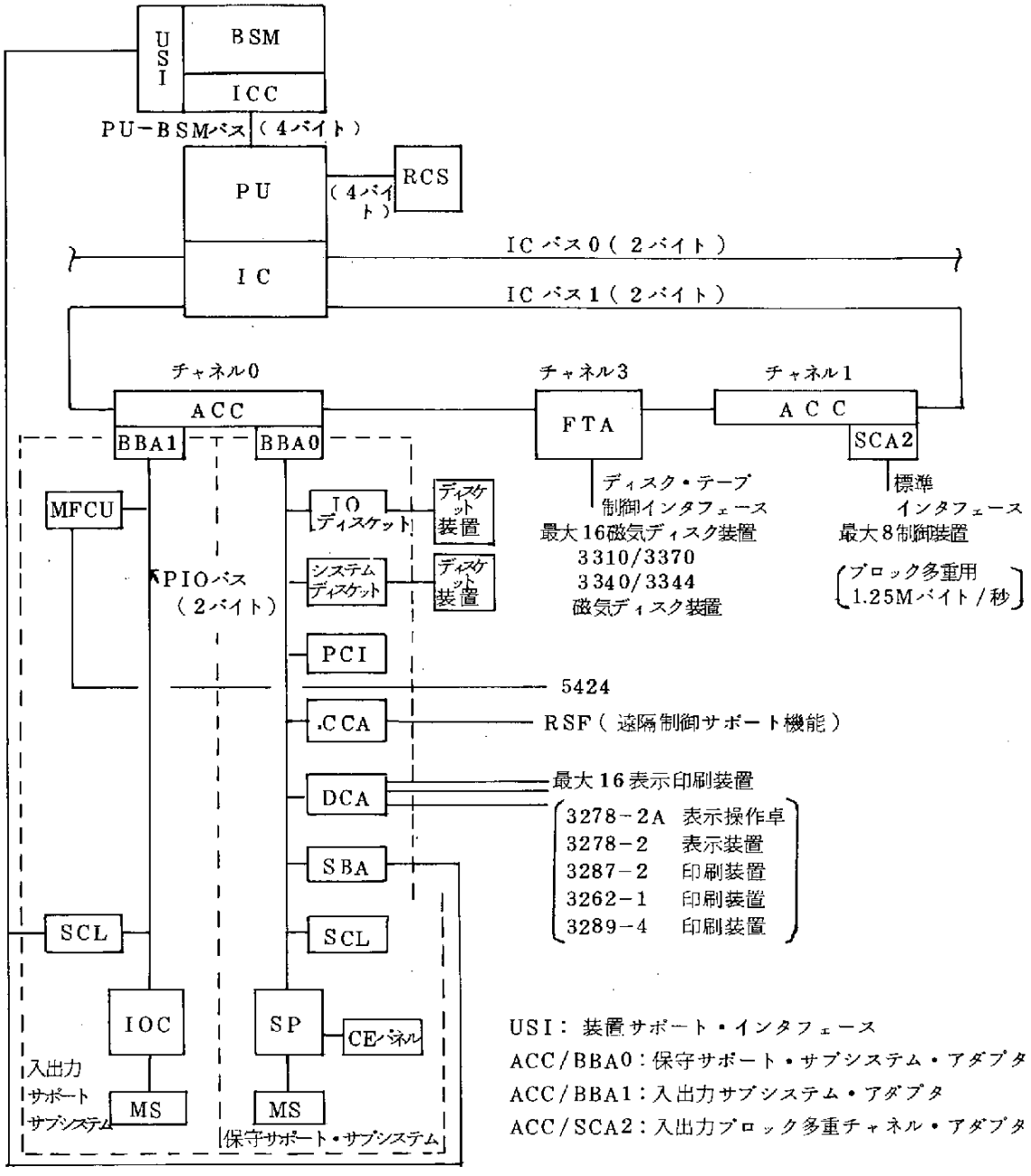
して，ＩＣバス当り５ＭＢ／ｓｅｃで，ＩＣには２本のＩＣバスが接続可。

・ＩＣバス上の各種アダプタの組合せに対しても３種の選択が可。

・論理チャネルは０～６まで合計７個使えるが，同時に使える最大論理チャネル数は５個。

モデル２型では，図3.24に示すようにＲＯＳ（Read Only Storage）が追加されている。ＢＳＭのアクセスは，モデル１型では1，2，3，4，16，32Ｂ（バイト）単位で行なわれ，４Ｂ単位がほとんどであったが，モデル２型では，１～64Ｂまで任意の単位で行なうことができ，ほとんどの場合64Ｂで行なわる。ＰＵ（Processing Unit，処理装置）からＢＳＭへのアクセスはＣＡＣＨＥ（キャッシュ記憶）経由で，ＩＣからＢＳＭへのアクセスはデータ・バッファ経由で行なわれる。１型では，４Ｂ当り，取出し900ns（ナノ秒，10^{-9}秒），記憶（書込み）1,300nsであったが，２型では，64Ｂ当り，取出し2,600ns，記憶3,100nsである。モデル１型のＥＣＣ（Error Correcting Code，エラー訂正符号）は４Ｂ単位で構成され，１ビット・エラーの自動訂正，２ビット以上のエラーを検出しているが，モデル２型では，８Ｂ単位で同様のことを行なっている。

ＲＣＳ（Reloadable Control Storage，再ロード可能な制御記憶）は，モデル２型では標準で128ＫＢに拡張されている。マイクロコードをＢＳＭからＲＣＳバッファにロードするとき，モデル１型は32Ｂ単位であったが，モデル２型は64Ｂ単位になっている。あらたにＲＯＳを導入した目的は，マイクロコードの量を減らしてユーザ記憶域を拡げること，マイクロコードの取出し時間を短縮することである。ＲＯＳに入っているのは高速処理が要求されているコードである。ＲＯＳからのマイクロコードの取出しはＲＣＳバッファを経由しないで，直接オペレーション・レジスタ



SP：サポート・プロセッサー1，IOC：サポート・プロセッサー2，MFCU：5424多能カード処理装置アダプタ
PCI：電源制御インタフェース，CCA：遠隔操作サポート・アダプタ，DCA：表示印刷装置アダプタ
SBA：サポート・バス・アダプタ

図 3.23　IBM4331－モデル2型プロセッサのシステム構成

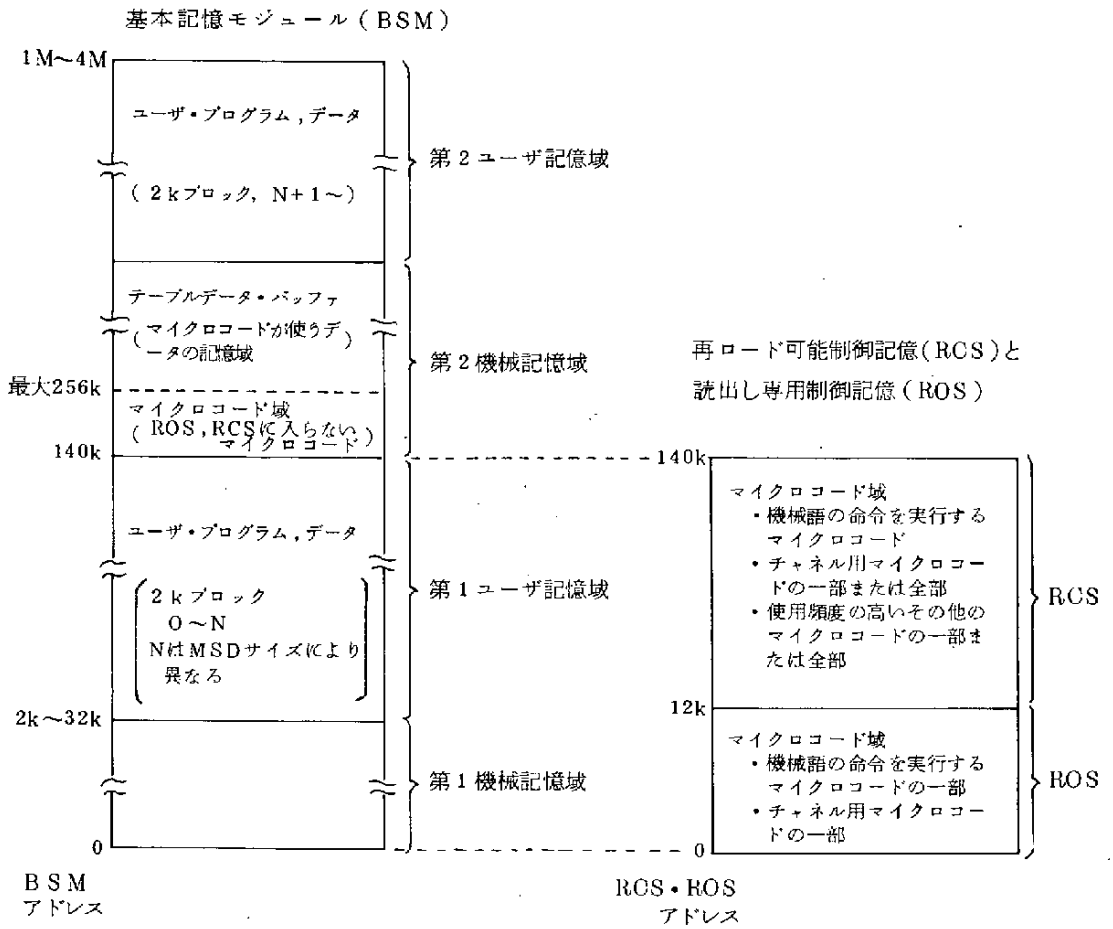


図 3.24　基本記憶モジュール（BSM）の4つの記憶域と，マイクロコードを保持する再ロード可能制御記憶（RCS）と読出し専用制御記憶（ROS）

に取り出される。その所要時間は100nsで，ROSの容量は12KBである。

モデル2型の処理装置（PU）もモデル1型と同じく，4B幅の処理を基本にしている。基本サイクル時間は10ns，マイクロ命令の実行時間は，200 ～ 1,600ns であり，マイクロ命令により異なる。PUの内部構造には種々の改良，機能の補強がなされている。CACHEは8KB，1エントリ当り2KBで4エントリある。書換えは64B単位で行なわれるため，各エントリは64Bを1単位として32単位に分かれている。個々の単位の64BのデータがBSMからCACHEにロードされているが，書換えがあったかなどが，CACHEディレクトリに記録されている。CACHEへのアクセスは，取出し，記憶ともに200nsである。CACHEとCACHEディレクトリの構成を図3.25に示す。その他，ALU（Arithmetic Logic Unit，算術論理演算装置）演算を高速化するための局所データ保管記憶の装備，浮動小数点計算高速化のためのハードウェアの改良，マイクロコードの強化・改良が行なわれている。

モデル1型のそれと異なり，モデル2型のICはPUから独立して入出力処理を行なうことができるし，データ・バッファを装備してICバス上のデータ転送能力を大幅に向上させている。モデル2型のICは自分のハードウェアを使ってデータを転送する。IC内に装備されたデータ・バッファは2KBである。アダプタ当り256B使うので，8個のアダプタが同時にデータ・バッファを使える。データ・バッファとBSM間のデータ転送は最大64Bである。入出力装置に転送すべきデータがCACHEにあれば，CACHEから最大64B単位でICのデータ・バッファに移される。

ICバスは2本接続でき，そのうちの1本は，ICバス上のアダプタの組合せを変更することを可能にするため，3種類のアダプタの組合せが用意されている。モデル1型と同様，サイクル・スチールでICバス上のデ

ータ転送が行なわれる。ＩＣバス上のアダプタの組合せにより，磁気ディスク・アダプタは最大２個，ブロック多重チャネルも最大２個，高速ブロック多重チャネルが最大１個接続できる。ブロック多重チャネルのデータ転送能力は，ＰＵとＩＣの改良で，モデル１型の 0.5MB／sec に対して，1.25MB／sec に改良されている。高速ブロック多重チャネルにより，磁気ディスク制御装置 3830 または 3880 経由で，磁気ディスク装置 3330，3340, 3350, 3370, 3375 などの接続が可能になっている。また，従来のＣＫＤ（Count Key Data）アーキテクチャを使った磁気ディスク装置，新しくＦＢＡ（Fixed Block Architecture）を使用した磁気ディスク装置，あるいはそれらの組合せと，幅広い入出力装置やオペレーション・モードの組合せが可能になっている。

なお，モデル２型で使用されているＬＳＩメモリは，アクセス時間 400 ns，サイクル時間 900ns，チップ・サイズ 6 mm × 6 mm，集積密度 64 Kb／チップ（キロ・ビット／チップ）で，１枚のカード上に 512 KB 実装されている。

ついでながら，モデル１型については，日本アイ・ビー・エム社の鍋島幸敏氏による詳細な解説が，bit臨時増刊ダイナミック・アーキテクチャ（共立出版，Vol.12, №10, 1980 年）に掲載されている。

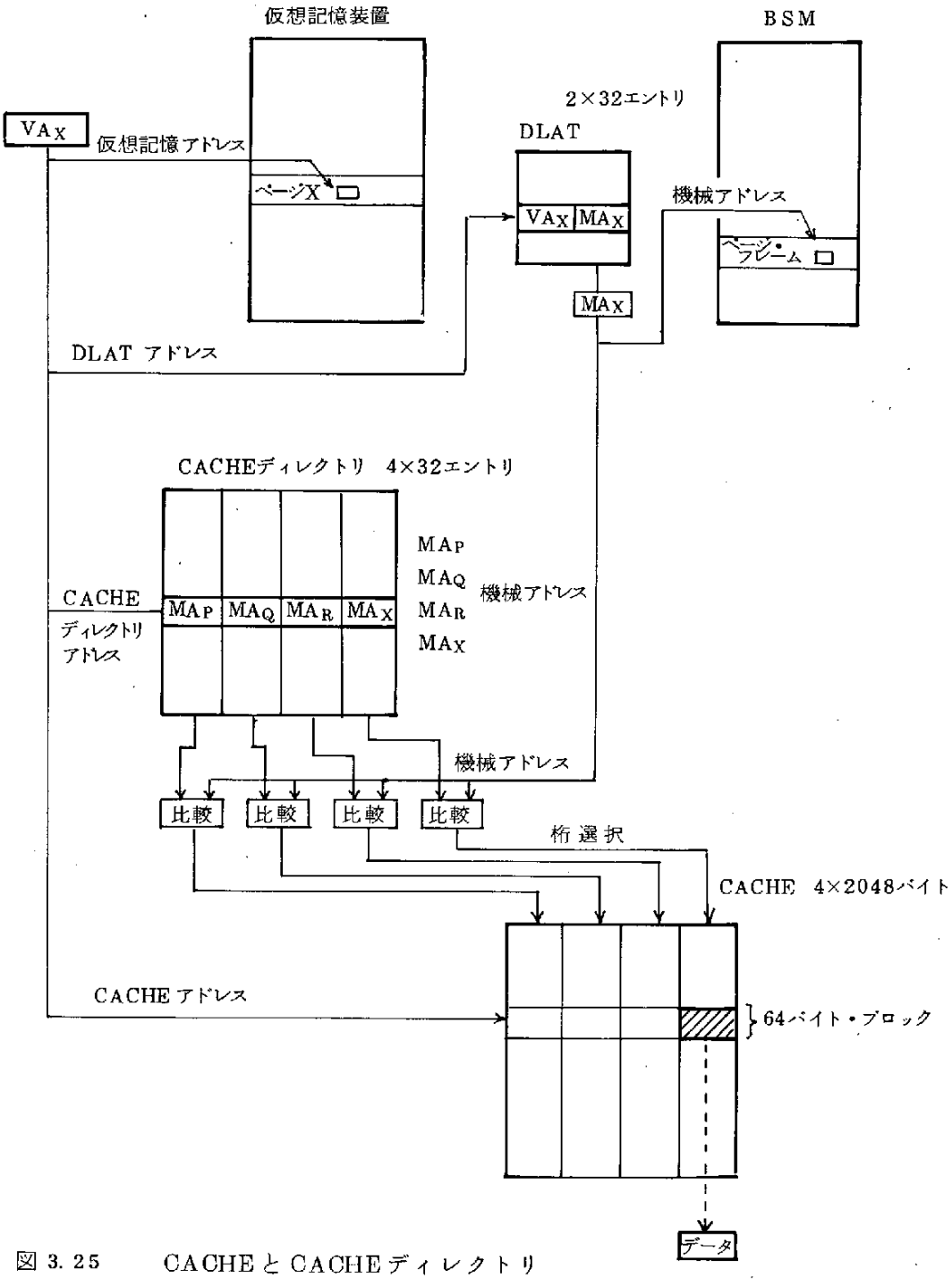


図 3.25　CACHEとCACHEディレクトリ
その構成とCACHEアクセス

3.6 IBMドイツ，ハイデルベルグ・サイエンティフィック・センター

ハイデルベルグ・サイエンティフィック・センター（Heidelberg Scientific Center，HDSC）は，世界中に散在するIBMの12のサイエンティフィック・センターの1つであり，1968年設立された（日本アイ・ビー・エムにも東京にサイエンティクィック・センターがある）。（ボブリンゲン研究所など）研究所と異なり，サイエンティフィック・センターは新しいハードウェア技術や製品を開発することが任務ではない。むしろ，既存の製品の新しい応用技術，応用分野を開拓することである。プロジェクトの選定に際しては，そのプロジェクトのもつ社会的意義がもっとも重視され，それぞれの国の社会的要請などによって決定されている。例えば，イタリアのピサ（Pisa）にあるサイエンティフィックセンターでは，Arno川（Arno River）の流れ方をシミュレートし，氾濫を防止するための方策を見つけるための数学モデルが開発されており，東京のサイエンティフィック・センターでは日本語処理の研究が行なわれている。サイエンティフィック・センターは他の（種類の）研究所と異なり，研究はすべて公開され，大学など外部組織との交流も盛んである。

HDSCでは，Ulrich Schauer氏に会い，HDSCの研究活動と，HDSCの最近の大きな研究プロジェクトであるIDAMS（Integrated Data Analysis and Management System）について調査した。

3.6.1 HDSCの研究活動

1973年以来，HDSCでは，データ処理に精通してないユーザでも計算機を用いて創造的な仕事ができるようにするためのプログラミング・システムの研究を行なってきている。この研究プロジェクトは「エンド・ユーザ（end user）による対話式プログラミング」と言われており，適用業

務に精通している（が，データ処理についてはくわしくない）人（すなわち，エンド・ユーザ）が，その人自身に特有な，よく理解（定義）できている問題を解決するための簡単なプログラムをつくることを可能にする技法をみつけることが目標である。エンド・ユーザは，自身の適用業務の用語を用い，計算機と対話しながら作業することができるようでなければならない。そのため，HDSCの研究者達は簡単な対話様式をつくっている。例えば，エンド・ユーザは表示型端末の前に座り，あたかも紙と鉛筆を使って仕事をしているようにスクリーンの形式を設計することができる。その形式中に論理演算，算術演算，データ入出力操作などを指定する。このようにして，ルーチン・ワーク的な作業を計算機に代行させることができるようになる。

現在行なわれているもう1つのプロジェクトに，「エンド・ユーザによる問題解決システム」がある。このプロジェクトは，エンド・ユーザによる，繰返し使用されるプログラムの開発を支援するためのツールを提供することではなく，エンド・ユーザが（1回限りの）問題を解くことを直接手助けしようとするものである。すなわち，ユーザはデータベースに対し質問を発し，その結果として取り出されたデータを既存のプログラムで処理することができるようになる。この場合にも対話技法が中心になる。解決すべき問題にとって必要なデータは何かを指定し，処理に用いるプログラムを計算機に指示し，取り出したデータをそのプログラムで処理することを指令することが，すべて対話式に行なわれる。ユーザは，問題解決のための（計算機との）対話において，彼自身が抱えている（解決すべき）問題しか理解できないのだ，という点に特別な注意が払われている。具体的には，このプロジェクトの結果としてIDAMSが開発されている。

以上もふくめて，1978～9年に行なわれた研究のテーマは次のように

なっている。

(1) Enduser Systems Research

- Interactive Programming by Endusers
- Integrated Data Analysis and Management System
- User Specialty Languages
- APL Complementary Functions

(2) Automatic Information Management Research

(3) Operating Systems Research for Very Large Systems

エンド・ユーザ向きシステムに関する研究に重点が置かれていることがよく分る。

HDSC で使われている計算機システムについて簡単に述べておく。HDSC に設置されている計算機システムの構成と（組み込まれて）使用されているソフトウエアをそれぞれ図 3.26，図 3.27 に示す。研究者が能率的に研究を進めるためには，できるだけ制限事項を少なくし，自由に計算機資源を利用できるようにしなければならない。そのような環境を実現するためには，CPU 能力，磁気ディスク，特殊ハードウェア，多くのツール類を効率よく経済的に使わなければならない。多くのツール，例えば，VM（Virtual Machine，仮想計算機，オペレーティング・システムの１つ）モニター，SMART などが上記の目的のために HDSC の事情に合わせて組み込まれている。

3.6.2 IDAMS の概要

IDAMS（Integrated Data Analysis And Management System，統合データ分析管理システム）は，プランナ，エンジニア，サイエンティストのためのすぐに使える問題解決，意志決定を支援するシステムで

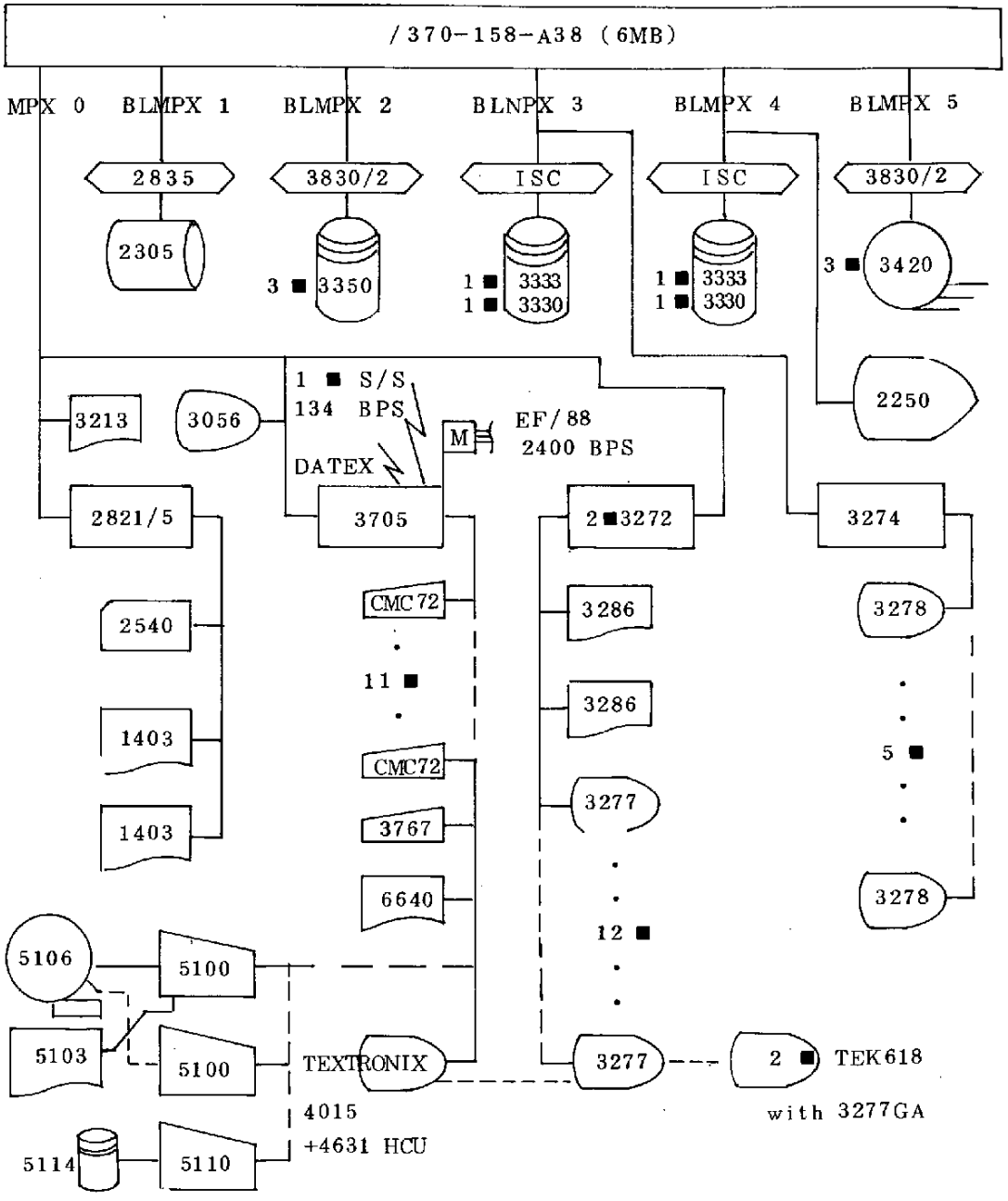
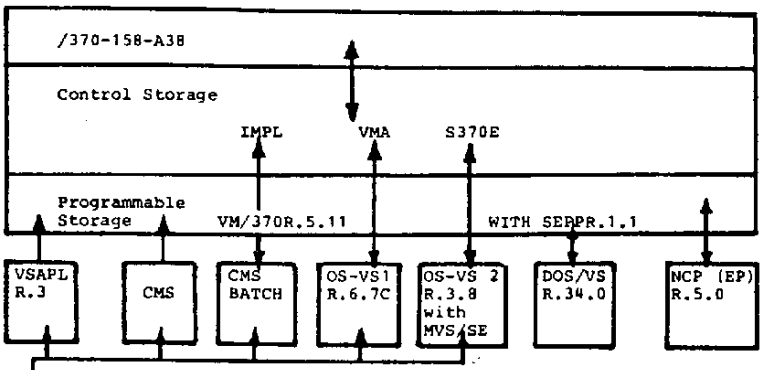


図 3.26　　HDSC 計算センターのハードウェア構成
（1979年12月現在）



PP's	Installed under	Version
CMS EXEC Language Extensions	C	1.0
VMPR, SCP, VMAP, VMGM	C	-
Online Tape and Disk Library Mgmt.	C	1.0
VS/REPACK Memory Analysis	C	1.0
Resource Measurement Facility	M	-
VNET - VM/370 Networking PRPQ	C	1.1
PL/1 Optimizing Compiler	V,C	3.0
PL/1 Checkout Compiler	C	3.0
PL/1 Resident Library	V,C	3.0
PL/1 Transient Library	V,C	3.0
FORTRAN HX + Optimization Enhancement	C	2.2
FORTRAN CODE & GO	C	3.0
FORTRAN G1	C	2.0
FORTRAN Interactive Debug	C	2.1
FORTRAN Library Model 1	C	2.0
FORTRAN Library Model 2	C	2.3
FORTRAN Preprocessor	C	1.0
ASSEMBLER H	V,C	5.0
ASSEMBLER-H/CMS Interface	C	1.0
DEBASS VM/370	C	1.0
3705 SCP	V	3.0
SAM Extended	M	-
VSPC OS/VS	V	2.0
VSAPL	V,C	3.0
CICS/VS	V	4.0
STAIRS/VS WITH TLS & PMR	V	2.5
Query by Example	C	1.1
OS/VS Sort/Merge	V	3.1
VM 370-CMS Sort	C	2.0
SCRIPT/370	C	2.0
DCF	C	1.0
EDGAR Display Editing System	C	1.1
SL-MATH/PL-MATH/SSP	V,C	-
GPSS	V	1.4
FAMS	V	1.1
CSMP	V	1.3
PLANCODE/I	V	2.1
GRAPL/1	-	2.0
26 VSAPL Application Programs	-	-
MISCELLANEOUS (IBM Internal Use Only)		
PL/S II	C	7.2
SCRATCHPAD, LISP, RESQ	C	-
XRAM, PRTV, SYSTEM/R, USAGE	C	-

図 3.27　HDSC計算センターに導入されているソフトウェア

（M：OS／VS 2，V：OS／VS 1，C：CMS）

ある。IDAMSは，データ処理技術の訓練をほとんど受けていない（エンド・）ユーザを支援するために必要ないろいろな（システム）構成要素を1つに統合化したシステムである。具体的には，以下の構成要素から成っている。

(1) （スカラの数値，文字など）通常のデータ型に加えて，ベクトル，行列，測定値などの高水準データ構造もサポートする拡張されたデータベース管理，測定値などでは単位も識別（し自動的に変換する）

(2) 例えば，APL，PL/I，FORTRAN，Assemblerなどで書かれている汎用プログラム，適用業務向き特定プログラム用の関数ライブラリ

(3) グラフ表現，（例えばデータの時系列配置などの）データ操作，報告書作成のサポート

(4) IDAMS自身，およびIDAMS中のプログラムやデータに関する機械処理形式の情報だけでなく，（散文的）説明情報も記憶されているが，それらを管理する構成要素

(5) 操作に精通していないユーザのための対話式での誘導機能

(6) データの抽出および分析のための高水準の照会言語。QBE（Query by Example，例示照会言語，M.Zloof によって考案された例示による照会言語で，エンド・ユーザとシステムの優れたインタフェースである）の構文とAPLを組み合わせた，強力で，柔軟性に富み，使い易い言語

(7) データ分析のためのホスト言語はAPL

とくに重要なことはVSインタフェースをもっていることである。このインタフェースはIDAMSの中心的構成要素であり，VSAPL−CMS システムをPL/I環境と結びつけ，コンパイルされて記憶されているプログラムをVSAPLから簡単に効率よく呼び出すことを可能にしている。

Schauer 氏とともにIDAMSを使ってみた。仕事の指定は，QBE をベースにしてあらかじめつくられているテーブル中に必要なデータを入れていくというやり方で行なう。分析のもとになるテーブルの名前，それをグラフにするための縦，横の軸のとり方，軸の単位，軸につける名前，プロッティング（実線，点線のプロッティング，頻度分布表など），表示型端末の画面の分割，グラフの平滑化に用いる関数，などを指定することができる。また，一定の規準でデータベースから取り出したデータ（のセット）に対してAPL を用いて処理を行なうこと，さらには，FORTRAN，PL／I 関数を呼び出し，それを用いて加工することもできる。

IDAMSはQBE と同等の構文をもつ言語がベースになっているが，QBE と異なり，データを選択するだけでなく，選択して取り出したデータを処理するためのプログラムの指定も同様に行なえるなどいくつかの追加機能をもっているので，EQBE（Extended QBE，拡張QBE）と呼ばれている。この追加機能には種々の条件設定機能などが含まれており，APLとして正しい表現であれば，何でも使えるようになっている。

照会検索は最初は簡単なところから始め，だんだんいろいろなものを付け加えていくことによって複雑な形にまで積み上げていくことができる。その際，APL の機能が最大限に利用される。このような使い易い基本的な枠組がシステムから提供され，それにいろいろな項目を挿入指定していくというやり方であるので，ほとんど予備知識をもっていなくてもEQBE を使うことができる。

IDAMSでは，EQBEの他に，ドイツ語そのものに近い照会検索言語USL（User Specialty Language）も考えられている。USL は，ドイツ語のサブセット（部分集合）ともいえるもので，EQBE に比べれば，非常に多くの規則をもっている。しかし，ドイツ語を話す人達にとっては

ごく当り前の（言語としての）規則であり，何ら負担にはならない。データ処理における自然言語の採用は，このようなところから徐々に進めていくのがよいと考えられる。

IDAMS中のデータやプログラムの内容や属性が分らないときにはIUG（Interactive User Guidance，対話形式によるユーザの案内）がはたらき，ユーザが自分の仕事をする際に，どんなデータを使うかという情報を提供すると同時に，どのプログラムをどのように使ってそのデータを加工するかという補助情報も提供する。

なお，IDAMSの詳細は付録Bで述べている。

3.7 IBMイタリア，公共部門インダストリ・センター

公共部門インダストリ・センター(Public Sector Industry Center, PSIC)の任務は，公共部門インダストリのデータ処理適用業務をサポートすることである。公共部門インダストリとは，具体的には，官公庁，教育研究機関，電信電話公社，鉄道・バス会社などのことである。

ここでは，Pierlurgi Arnao氏からIBMの組織，PSICの任務，適用業務プログラム開発の考え方，教育など，Giorgio Dori氏から教育機関における計算機利用，適用業務開発の動向，Ulrich Schönbaumsfeld氏からMUSICシステム，DOBISシステムについて説明をうけた。

3.7.1 PSICの任務，適用業務開発の考え方，および教育について

IBMの組織は，マーケティングではUSA，AFE，EMEAに分かれているが(3.5節参照)，研究，製品開発，製造などは世界126か国のIBMが互いに連繋しながらそれぞれの任務を果すようになっている。適用業務の開発についてもそれぞれの分野ごとに国際的センターが設けられている。具体的には，表3.8のようなセンターがある。この表にあるように，PSICの任務は，官公庁，教育研究，郵便，電信電話，陸上海上の交通輸送，医療，公共事業などのための適用業務(プログラム)の開発，運用管理である。過去の実績にもとづき，将来の需要の傾向予測をにらみながら，現在の問題に対処するよう配慮している。従って，この分野において将来どのような適用業務の需要が強くなるかを正確に予測することが非常に重要で，約5年前に適確に識別し，それにもとづいて(5年後に使われる)適用業務プログラムを(その時点から)開発するようにしている。

(IBMの)ヨーロッパにおける教育は，原則的に，基本コースは各国内で，上級コースは国際間で(各国から派遣して共通に)行なっている。

表 3.8　適用業務ごとの担当センター

製造装置インダストリ・センター
製　造
プロセス
金融インダストリ・センター
銀　行
保　険
証　券
流通インダストリ・センター
流　通
仲　介
サービス
公共部門インダストリ・センター
官公庁
教育研究
郵便，電信電話
公共事業
陸上および海上交通運輸
医　療
航空センター
航　空

例えば，ベルギーのブラッセルにあるIEC（International Education Center，国際教育センター）では，各国における（基本コース）教育を補なう形で，

・上級社内教育，上級顧客教育

・ＩＢＭ幹部経営者の啓発

・顧客の経営者の国際的視野に立つ教育

などを行なっている。

3.7.2 ＭＵＳＩＣシステム

ＭＵＳＩＣ（システム）は，カナダのモントリオール（Montreal）にあるMcGill大学で開発された対話式オペレーティング・システムで，正式には，「ＭＵＳＩＣⅣ対話式オペレーティング・システム」と呼ばれている。もちろん制御プログラムとしての機能だけでなく，各種ユーティリティ・プログラム，種々のコンパイラとのインタフェース，適用業務パッケージもそなえている。そして，全体として，高性能で，コスト効率のいい，管理可能なタイム・シエアリング（による計算機利用）環境を与えることを目標としている。（ＭＵＳＩＣの）ユーザには，問題解決，プログラム開発，ファイル編集，ワード・プロセシング，CAI（Computer Assisted Instruction，計算機援用教育），バッチ処理（batch processing，一括処理）というような幅広い利用が多人数で同時に可能である。もともと大学で開発されたものであるので，大学での計算機用にもっとも適している。ＭＵＳＩＣは，小型のシステム／３７０モデル１１５から大型の３０３Ｘ処理装置（３０３Ｘ Processing Complex）まで，ＩＢＭの広い範囲の計算機ハードウェア上で動く。ＭＵＳＩＣは専用ＣＰＵ上だけでなく，ＶＭ／３７０（Virtual Machine／３７０）の下でも効率のいい多重アクセス（multiaccess）仮想計算機として動作する。

ＭＵＳＩＣ－Ⅳでは，従来の機能に加えて，新しく次のような機能が使えるようになっている。

・VS／APLのサポート

・IIS（Interactive Instructional System，対話式教育システムという意味のプログラム・パッケージ）のサポート

・PL／I最適化コンパイラのサポート

・VS BASIC拡張機能のサポート

・ユーザ領域の拡大

・LPA（Link Pack Area）とロード・ライブラリ（Load Library）

・Context Editor（文脈エディタ，高度な機能をもつエディタ）の拡張。具体的にはHELPコマンド，フル・スクリーン編集（Full Screen Edit）機能，作業ファイルの使い方の最適化など。

・VM／370環境の下でのMUSICユーザに対して，OS 仮想計算機にサブミット（Submit，処理してもらうために投入すること）したジョブの出力を検索することを可能にしている。

・Auto Speed Detect（というプログラム）をサポート

MUSICシステムの設計では，以下の点に力点が置かれている。

・人間的要因の尊重

・タイプ操作がへたな人にとっても，芸の細かな計算機のプロにとっても，エラーの確率を小さくするような，やさしく，誤りにくいコマンド言語

・最先端のセキュリティ（安全保護）技法の採用

・管理のいい利用

・使用料金計算

・バックアップ／回復機能

MUSICシステムは，1都市（にある学校間での計算機の共同利用である）学校システム，大学，製造会社，商業タイム・シエリング・サービス

会社をふくむ多くの環境での計算機利用に適した成熟した，多目的，高性能の対話式オペレーティング・システムである。（１台の）専用CPU上で最小のMUSICシステムを動かすには，10 台の端末（による対話式利用）とバッチ処理をサポートするとして，250Kバイトの主記憶をもつ必要がある。主記憶を増やせば最大250台までの端末をサポートすることができる。端台を１台増やすごとに約800バイトを必要とする。

MUSICシステムでは，対話機能をもつ広範なプログラム言語，多くのサブルーチンからなるオンライン・ライブラリ，既製の適用業務プログラムを使うことができる。ユーザは，端末からでも，バッチ処理用のカードリーダ，プリンタの組からでもプログラムを投入，コンパイル，実行，（ライブラリに）記憶することができる。以下は，利用可能な言語処理系（言語プロセッサ）の一覧である。もちろん，ユーザがこれ以外のコンパイラを追加することも可能である。

・FORTRAN(OS FORTRAN G1)

直接アクセス文(direct access statement)，デバッグ機能，リスト型人出力などが使える。従って，FORMAT文による固定フィールド（固定欄）ではなく自由形式の入出力欄が使える。簡単で，多様なデータ入力を可能にするためMUSIC Ⅳ 多くの機能が追加されている。MUSICの下では，FORTRAN Mod.1ライブラリ・プログラム・プロダクトをこのFORTRANコンパイラとともに用いることができる。MUSICのソート（sort）機能をFORTRANから呼び出して使うこともできる。実行時の記号デバッグもできる。

・VS APL

・PL／I最適化コンパイラ

・VS BASIC

・COBOL（OS/VS COBOL，もしくはOS ANS COBOL）

・MUSIC/APL

・OS/VS Assembler

・MUSIC/BASIC

MUSICでは，次のような適用業務プログラムも使える。

・Interactive Instructional System

・MUSIC/SCRIPT

SCRIPTはテキスト処理プログラムで，マニュアル，技術論文，手紙などを書くときに役に立つ。タブ（tab），バックスペース（backspace）文字のような端末依存性をなくし，キー・ストローク（keystroke）の数を減らし，熟練者でない人でも使えるようにするようにとくに配慮されている。MUSIC/SCRIPTへの入力はFORTRANその他の言語で生成したものでもよい。

・STATPAK

STATPAKはMUSICの一部である会話式統計計算パッケージである。プログラミング経験のない人でもこのパッケージを上手に使うことができる。分散分析，因子分析，回帰その他の統計関数の計算を対話式に行なうことができる。

・COGO（Civil Engineering Coordinate Geometry Program，土木工学用座標幾何プログラム）

MUSICでは，対話式でないジョブをバッチ処理することもできる。どのユーザ端末もサービスを要求していないときだけバッチ処理にサービスがまわってくるように優先順位付けされている。バッチでも（遠隔）端末ユーザと同じ言語プロセッサ，ファイルを使っているので，端末ユーザとバッチ・ユーザでプログラムやデータを共有することが可能である。例え

ば，データ・ファイルを端末から作って編集し，その後，バッチ・ジョブの入力とすることができる。また，プログラムを端末からコンパイル，テストし，目的プログラムをファイルに記憶（セーブ，save)しておいて，後でバッチ・モードで使うこともできる。さらに，磁気テープのような外部記憶媒体から読み込んだデータをバッチのユーティリティ・プログラムでオンライン記憶装置に格納し，それを端末からやバッチで使うこともできる。

ファイルをつくったり，修正したりするための汎用の対話式テキスト・エディタであるContext Editorが使える。Context Editorは，ファイルを変更せずに，そこから特定の情報を探し出すためにも用いることができる。簡便ではあるが強力なコマンドを用いて行や文字列を挿入し，変更し，削除することができる。別のファイルを変更中のファイルに合併する（その一部として組込む）こともできる。このエディタでは，桁条件や論理条件にもとずいた高度の文字列処理ができる。16 進形式での変更が可能である。変更した行すべてに自動的に日付をつけることもできる。

MUSICはシステム使用料の請求に役立つ記録をとっている。端末時間，CPU 使用時間，入出力した文字数などを記録している。読み込んだカードの枚数，印刷した行数，パンチしたカードの枚数はMUSICのバッチ機構（部）が記録している。テープやディスクの装填要求メッセージはログ・アウトされる。ディスクの使用空間の大きさも記録されている。これらの記録をもとにMUSICの料金計算プログラムが使用料を計算する。その結果を用いて，（MUSIC）導入個所ごとに，請求書作成プログラムで請求書をつくる。

MUSICのソートは（IBMの）OSとの互換性を有するソート機能をもち，固定長のレコードをソートする。FORTRAN，PL／I，COBOL か

ら呼び出して使うことができる。

MUSICのコマンド言語は，記述形式で，おぼえやすく，簡単な命令のの集まりであり，ユーザはこれを用いてMUSICシステムと意志の疎通を図る。MUSICでは，／INCLUDEのような実行時コマンドをプログラムとともにセーブ・ライブラリ（Save Library）にとっておくことができる。この／INCLUDEはPL／Iの％INCLUDE，COBOLのCOPYに相当するものである。MUSICのコマンド言語は，初心のユーザが計算機を使うことを支援すると同時に，経験の深いユーザが巧妙な機能を駆使することもできるように設計されている。バッチ処理でも，端末（からの対話式）処理でも同じコマンド言語を用いる。

許可されてない者が，システム，プログラム，データ・ファイルにアクセスしたり変更したりすることができないよう，セキュリティ機能が組み込まれている。MUSICシステムのセキュリティは，登録コード，サブ登録コード，パス・ワード（pass word）で実現されている。セキュリティ管理者は，CODUPDプログラムに会話式で新しいコードを入れるだけで（その新しい）コードを追加することができる。種々の利用上の制限，省略時解釈も同じやり方で追加，変更することができる。パス・ワードはユーザが自由に変更することができる。MUSICのこのセキュリティは，（MUSICを）VM／370のもとで動かすときには組み込まれない。

MUSICは，それぞれの属性，利用上の制限条件情報をともなう約3,000の個人ユーザ・コードをサポートしている。さらに，任意のコードをつけることができるので，多くのユーザが同じ（登録）コードをもつことができ，同時にオンラインで使うことができる。許可されたコードの表はオンラインで更新が可能であり，パス・ワードをはじめいくつかの欄（フィールド）はユーザが（自由に）変更することができる。

3.7.3 DOBISシステム

Gesamthochchule Dortmund（ドルトムント大学群）の図書館は計算機を用いて利用者や司書の要求を満たすオンライン図書館システムを開発した。「Gesamthochschule」はいくつかの大学のグループという意味で，ドルトムントでは，Universitaet Dortmund（ドルトムント大学），Paedagogische Hochschule Ruhr（Ruhr教員大学），Fachhochschule（工科大学）からなっている。この図書館システムは，次の4つの大原則に立って設計されている。

- データ入力は1回だけ
- カタログやファイルはすべて分散アクセス
- 重複作業の排除
- 図書館活動，処理に関する統計情報の自動作成

上記の原則を体系的に実現したシステムを開発した。このシステムにより

- 特殊技能を有するスタッフはその能力を充分発揮できるようになった
- 間違いが極端に少なくなった
- 運用コストが下った

ドルトムント大学群には，3大学のそれぞれの（中央）図書館の他に，25の専門別図書館がある。これらの図書館の集まりを1つの（分散化された）図書館と考え，ドルトムント大学群図書館と呼ぶことにする。ドルトムント大学群図書館（全体）では，1975年時点で約800,000冊の蔵書を有し，その値は年々70,000冊の割で増加している。

DOBIS（DOrtmund BIbliotheks System，ドルトムント図書館システム）はドルトムント大学群で（1台）もっているIBMシステム／370上で動いている。もちろん，この計算機は，大学の他の業務，すなわち通常の管理業務，研究，教育にも使用されている。図書館ファイルは計

算機に集中して集められているが，分散して配置されている端末からアクセスできる。

図書館ファイルは高速直接アクセス記憶装置上に記憶されている。蔵書（論文などもふくむ）はカタログされ，著者，書名，分野，呼出し番号，ISBN，ISSN，などによって検索できるようになっている（図3.28参照）。貸出しファイルも集中化されており，利用者名，利用者番号，書名索引などでアクセスすることができる。これらのデータへのアクセス時間は1,000 分の1秒単位で計算するほど高速である。索引の数が多く，柔

```
Catalog search
Names (author, editor, etc)

 1                    Nord  Merkur GmbH
 2                          Minsky, James A
 3                          Minsky, Marvin
 4                          Minsky, Stratton
 5                          Minsky, Susan
 6                          Moskopp auf der Heide, Johann
 7                          Müller, Anton
 8                          Müller, Georg
 9                          Mueller, Otto
10                          Müller, Wieland
11                          National Industries, Ltd.
12  South East Region       National Standards Institute
13                          Neumann, Caryl
14                          Nord Merkur GmbH

Enter line number or code
3_
t term    fd forw
w new ind b back
                  m more
```

図3.28　著者名（のアルファベット順）の一覧

軟性があること，アクセス速度が速いことにより，利用者と司書にとっても重要な作業である検索（探索）が簡単かつ効率的になっている。

ドキュメント（書籍，論文等をさす）が必要になると，図3.29 に示す

手順が開始される。借り手（利用者）は，件名データをアクセスできるが，貸出し情報，購入情報にはアクセスできない。司書はDOBISファイル中のすべての情報にアクセスできる。オンライン・カタログ（オンライン目録）には，システム中のすべての蔵書の状態だけでなく発注中のドキュメントの件名情報も入っているので，問合せに対しては，すべてのファイルをもとにした最新の回答が与えられる。図書館のスタッフ・メンバーには端末からオンラインでファイルを変更する権限が認められている。従って，変更はDOBIS中で直ちに実施され，次の問合せに対しては最新の情報をもとにした回答が与えられる。

DOBISによる図書館業務手順は，入手による場合と同じように，カタログ，貸出しの3つの手順に大別することができる。

ドキュメントの取得が決まると，件名情報注文情報がDOBISに入力される。DOBISはこれらの情報から（購入）注文書を印刷する。詳細（完全）な件名情報が後刻MARC（Machine Roadable Catalogue）サービスからもらえる場合は，この時点で入力する件名情報は最小限に抑える。ドキュメントを受け取ると，その旨をシステムに記録する。送り状が届いたとき料金情報を入力するので，図書館予算は常時正確かつ最新状態に維持される。ドキュメントが届かないときは，督促状が自動的に印刷され業者に送られる。ドキュメント発注中の間でも，業者索引，注文番号索引だけでなく，オンライン・カタログを用いてそのドキュメントの状態を知ることができる。購入手順がオンラインであり，（状態の）更新が直ちに実行されるので，

- 人手による購入ファイルの作成，維持の手間が省ける
- 発注後の状態追跡が容易であり，必要な対処が自動化できる
- カタログ化する（目録をつくる）ときにMARCを使う準備ができる

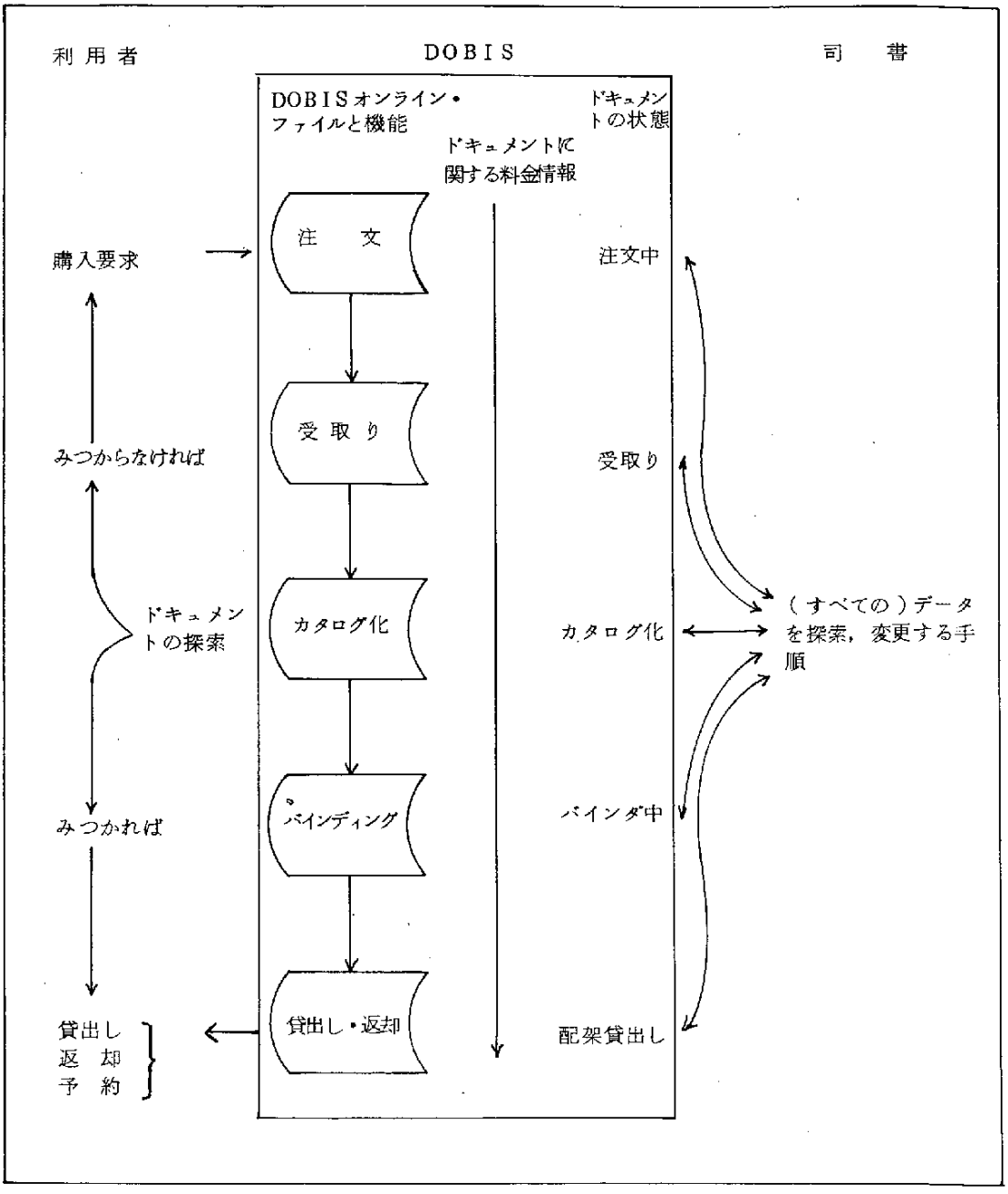


図 3.29　　DOBISによる図書館業務の流れの概要

・購入手順が簡単になる

・予算情報が直ちに利用できる

ドキュメントが到着すると，購入手順中に入力されたカタログ情報は更新され，完全な形にされる。これらの変更，追加は端末からオンライン・カタログに対して直接実施される。カタログ化はMARC 標準規格にそって行なわれるが，MARC タグや関連するデータ処理プログラム等を知っていなければできないというわけではない。購入したドキュメントのカタログ情報をMARCテープから抽出する場合は，システムがテープを探索し，その部分を見つけ出したとき，既存のカタログをテープ上のカタログで置き換えるというやり方をする。ある一定時間中に（テープ上に）カタログが見つからないときは，システムは司書にその旨を伝え，人手による分離処理を依頼してくる。オンラインでのカタログ化と探索により次のような多くの利点が得られている。

・MARCテープをシステムに直接読み込むことができ，司書の手作業を省力化している。

・カタログ・カード（目録カード）のタイプ，カタログ・カードの（カタログ中の正しい位置への）挿入などを省力化している。

・ドキュメントの（配架までの）処理がずっと速くなり，利用者により早く利用してもらうことができる。

ドキュメントの貸出し（および返却）は，利用者カードと，ドキュメント上のバー・コード・ラベル（bar-coded label）を読むことによって自動的に行なわれる。DOBIS は利用者が本を借りる資格を調べ，貸出し期間を計算し，貸出しの記帳を行なう。ドキュメントが返却されたときは，利用者カードとバー・コード・ラベルを読んでチェックする。次に，システムは課金計算をし，その（返却された）ドキュメント（と同じタイトル

のドキュメント）が予約されていないかどうかをチェックし，係員に適切な処置をとるよう指示する。

期限切れ通知書が一定期間ごとに印刷される。返却通知書，予約した本が借りられるようになったという通知書も要求するとシステムが印刷してくれる。特定のドキュメント，同一タイトルのすべてのドキュメント（のコピー），特定の利用者に関する状態情報は表示装置から直ちに知ることができる。貸出し（および返却）情報を集中して記憶し，オンラインによる問合せできるようになっているので，どんなドキュメントでもそれがどこに存在するかが直ちに分り，図書館はすべての蔵書を完全にコントロールできる。加えて，

- システムが貸出し（および返却）の統計情報を把握しているので，同じドキュメント（のコピー）を何部注文すべきかが正しく判断できる。
- 貸出し予約を即座に処理し，返却されたドキュメントが予約されていないかどうか自動的にチェックすることができる。
- 期限切れ，返却，その他の情報通知書類をすべて計算機で作成することができる。

図 3.30 に示すように，DOBIS に新しい別の言葉を話すよう教え込むことができる。実際，DOBIS では多国語が使える。すなわち，別々の利用者に別々の国語で語りかけてくるようにすることができる。図書館を使うために司書や利用者が異なる国の言葉を学ばなければならない，というようなことであってはならない。

DOBIS では，読み分け記号（例えば，$\bar{a}$，$\tilde{a}$，$\ddot{a}$ など）や特殊記号だけでなく，大文字，小文字すべてを入力し，表示することができる。DOBIS のすべての文字は紙，マイクロフィッシュ，テープに出力することができる。

```
DOBIS Functions

 1 Catalog search          ..sear.      10 Date reminder queues   ..date.
                                        11 Print queues           ..prin.
 2 Ordering                ..orde.
 3 Receiving               ..rece.      12 Standard codes         ..code.
 4 Serials, Periodicals    ..seri.
 5 Cataloging              ..cata.
 6 Binding                 ..bind.
 7 Accounting              ..acco.

 8 Circulation             ..circ.
 9 Interlibrary loan       ..ill .

Enter line number
-
```

図 3.30　（その１）DOBISの多国語能力

```
Hauptfunktionen

 1 Registersuche           ..regi.      10 Terminen               ..term.
                                        11 Druckausgabe           ..druc.
 2 Bestellung              ..best.
 3 Zugang                  ..zuga.      12 Standard codes         ..code.
 4 Zeitschriften           ..zeit.
 5 Katalogisierung         ..kata.
 6 Einbandbearbeitung      ..einb.
 7 Rechnungsbearbeitung    ..rech.

 8 Ortsleihe               ..orts.
 9 Fernleihe               ..fern.

Bitte lfd  Nr eingeben
-
```

図 3.30　（その２）DOBISの多国語能力

DOBIS funtics

1 Katal. Zoeken	..zoek.	10 Datum-bestanden	..data.
		11 Druk-bestanden	..druk.
2 Bestelling	..best.		
3 Ontvangst	..ontv.	12 Standard Kodes	..kode.
4 Reeksen,tijdschriften	..reek.		
5 Katalogisering	..kata.		
6 Inbinding	..inbd.		
7 Boekhouding	..bkhd.		
8 Uitlenen	..uitl.		
9 Interbibl. leenverk.	..ilv .		

Lijnnummer inbiengen

図 3.30　（その３）DOBISの多国語能力

Fonctions DOBIS

1 Examen Catalogues	..exam.	10 Rappels en attente	..rapp.
		11 Impressions en attente	..impr.
2 Commande	..comm.		
3 Reception	..rece.	12 Codes standards	..code.
4 Periodiques	..peri.		
5 Catalogage	..cata.		
6 Reliure	..reli.		
7 Fonctions comptables	..fcbl.		
8 Circulation	..circ.		
9 Pret entre Biblioth.	..pret.		

Introduire le numero de la ligne choisie

図 3.30　（その４）DOBISの多国語能力

3.8 IBMイタリア，ローマ・プログラム・プロダクト開発センター

このプログラム・プロダクト開発センター（PPDC，Program Products Development Centre）は，1979年，それまでLidingo,Sindelfingen,Vienna,Paris,Rome の5個所に散在していたEPPDC（European PPDC）が統合されたものである。より正確には，ローマPPDC（Rome PPDC）と呼ばれている。ローマPPDCは，次の3つのすべてのクラスに属するプログラム・プロダクト（プログラム製品）を担当している。

- 基本ソフトウェア
- DB/DC（Data Base/Data Communication，データベース/データ通信）ソフトウェア
- 適用業務ソフトウェア

とくに，イタリアとヨーロッパにおける計算機利用で効果が高いプログラム・プロダクトの開発を第一義の重点と考えている。イタリア向きのプログラム・プロダクトをとくに重視しているのは，イタリアの法令，イタリアの行政，銀行活動，金融上の規制に制約されるプロジェクトがあるからである。IBMイタリアには，ヘッドクオータ（本部）にあたるローマPPDCの他に，Verona,Florence,Bolognaにそれぞれ PPDC のブランチ（支部）がある。これらのブランチは，その地域にある会社などの組織とたえず密接な連繋をとりながら，そこで用いられる適用業務プログラムの開発を担当している。

ここでは，Paolo Giorgi 氏とGuido Mazzeo氏に会った。Giorgi氏からローマPPDCの使命や組織について，Mazzeo氏からソフトウェア開発に対する考え方とELIAS（Entry-Level Interactive Application System）について話をきいた。また，実際にELIAS を使ってデータベースの定義等を行なってみて，その効果を確認した。

3.8.1　プログラムの設計開発環境

ＩＢＭ社がユーザに提供するソフトウェア製品は，異なる多くのユーザの個々の要求に応えると同時に，保守が容易であり，成長（改良）可能でなければならない。さらに，ある程度骨格が定まり既に動いている適用業務システム中に（スムーズに）組み込むことができなければならない。このような要求と取組みながら，ローマＰＰＤＣは，種々のプログラム製品，例えば，ELIAS．PROJACS(Project Analysis and Control System，複雑なプロジェクトを企画，管理するためのシステム），MPSX／370（Mathematical Programming System Extended／370．線型計画法の問題を完全に，効率よく解くために開発されたソフトウェア・システム），DSX（Distributed Systems Executive，分散処理のための管理システム）などを開発している。

ローマＰＰＤＣでは，プログラムの設計段階から計算機が積極的に利用されている。設計レベルで使うＨＩＰＯ図や，スター図も計算機表示端末から作成，修正，編集できる。従って，プログラマは設計中のプログラムの流れを端末を通してながめることができると同時に，修正，改良も（端末から）容易にできる。プログラマは，設計中に繰返し強いられるいろいろな作業，例えば記入項目の更新などから解放されるばかりでなく，計算機と対話しながら，一歩一歩チェックして，設計中のプログラムがユーザの要求を満たすようにつくりあげていくことができる。さらに，大部分の異なる操作（使用）条件をシミュレートすることによってプログラム設計の完全性（completeness)を継続的にチェックできるので，誤動作の可能性を最小限に押えることができる。

SCRIPTという名前のプログラムを用いて，プログラムの開発，テストの文書も計算機で管理できるようになっている。従って，文書は，常時，

その状況に応じてもっとも適切な部分をもっとも適切な形式にして，取り出し，使用することができる。

ローマＰＰＤＣの活動には，プログラムの開発だけでなく，既製のプログラムの保守サービスも含まれているが，保守サービスでも計算機が有効に使われている。計算機端末を介して保守対象のプログラムの開発時の文書を見つけ出し，ユーザの要求に対する回答（保守の内容）をより迅やかに，より完全に与えることができる。また，同じ問題を抱えているＩＢＭの他の研究所中に保存されているデータを直接アクセス（使用）することもある。

現在ローマＰＰＤＣは合計数十万行のコードに対して責任を負っている。

ローマＰＰＤＣの（開発）要員は120人，計算機は5ＭＢ主記憶のシステム／370モデル158をＶＭ（Virtual Machine，仮想計算機システム，オペレーティング・システムの１つ）で使っている。磁気ディスクは3,700ＭＢであり，所内には約50台の（表示型，ディスプレー）端末が配置されている。2.4人に１台の割りの端末台数であり，対話式利用が定着していることがわかる。ローマＰＰＤＣの計算機は，ＩＢＭ，ＲＥＴＡＩＮ，ＳＥＣＯＭ，ＶＮＥＴなどのネットワークで他の研究所やソフトウェア開発センターなどと接続されており，設計，開発技法，個々のソフトウェア製品等に関する情報を交換し合っている。

3.8.2　ソフトウェア開発環境とＥＬＩＡＳ

ＥＬＩＡＳそのものの詳細は付録Ｄにまわし，ここでは，Guido Mazzeo氏による本節のタイトルを内容とする話を報告する。

(1)　標準的なＤＰ部門の現状は次の如くである。

・管理者層，（社内）ユーザ層，ＤＰ（データ処理）部門内部からの（不

満，要求などの）圧力が強くなってきている。

・（プログラムの）保守のロード（負担）が大きい。

・（人的，資金的）資源や，（高度な技術を必要とする適用業務，例えば，データベース／データ通信適用業務を開発するための）能力が限られている。

・（とくに，人間の）コストが急速に高騰している。

・新しい適用業務が開発待ちの状態のままになっている。

(2) 適用業務の開発について，（ＤＰ部門は，管理者層やユーザ層から）次のように言われている。すなわち，管理者層からは，

・ＤＰ予算は大き過ぎる。

・プログラム・コストはどうしてそんなに高くなっていくのか

・もっと迅やかにＤＰの効果は出ないものか

・多額の金をかけて開発したこの新システムでもわれわれは少しもハッピーにならないではないか

・質のいいプログラマを採用していたのではないのか

と言われ，ユーザからは

・この適用業務（プログラム）を今すぐ動かす必要がある，２年先ではない

・（プログラムの）こんな小さな変更にどうして30日もかかるのか

・（ＤＰ部門が開発してくれた）この新しいシステムは気に入らない

・データ処理はどうしてそんなに複雑なのか

・（ユーザにとって）わたし自身のシステムが欲しい

と言われている。

(3) ＤＰ部門の（人的）資源が一定である以上，図3.31 に示すように，適用業務の数が増えれば増えるほど，新しい適用業務の開発にまわすこ

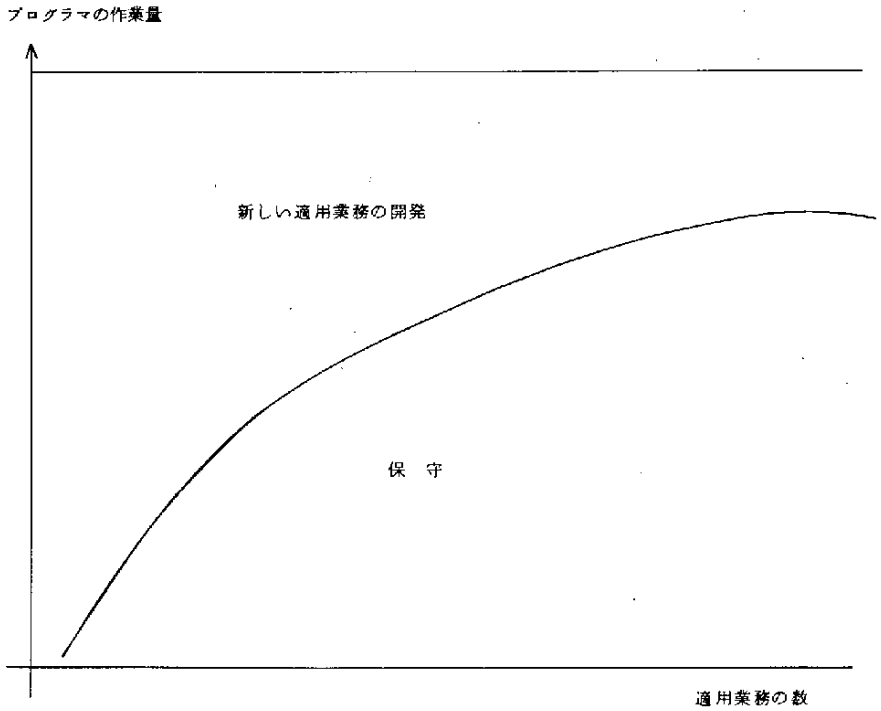


図 3.31　適用業務が増えるにつれて，その保守にさかなければならないプログラマの作業量も増える

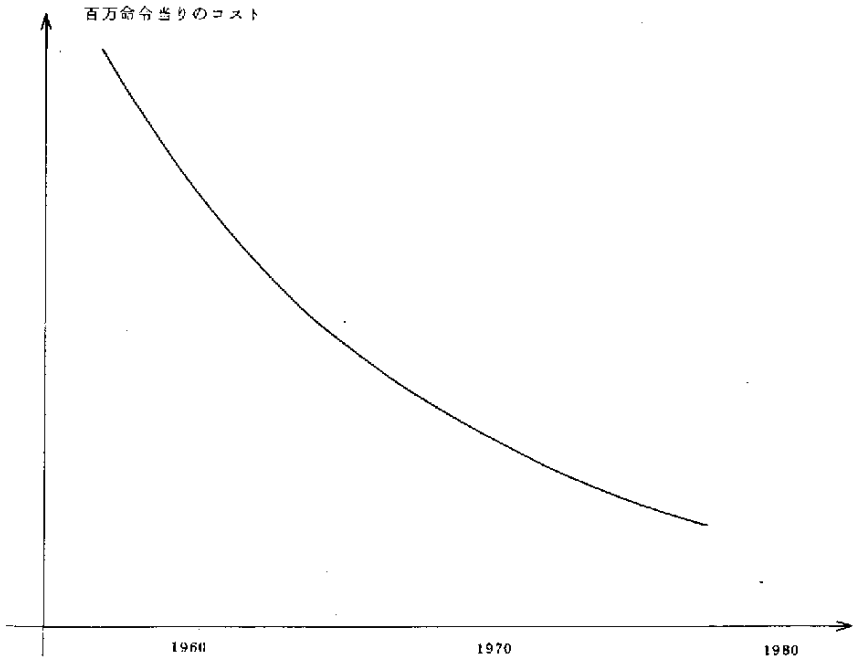


図 3.32　ＩＢＭのプロセッサの価格／性能

とができるプログラマの人数は少なくなる。

(4) 一方，ハードウェアの価格／性能（性能当りの価格，価格性能比）は急速に下ってきている。ＩＢＭのプロセッサに関しては図 3.32 のような状況である。

(5) このように，ＤＰにおいては，機械の生産性が著るしく向上している一方で，（主としてソフトウェアの開発保守を担当する）人間の生産性は思うように上がらず，図 3.33 に示すように両者の生産性の差は拡がる一方である。

(6) 現在のデータ処理環境をまとめると次のようになる。

- ハードウェア・コストは急速に低下している。
- 人間のコストは上昇している
- 従って，（ＤＰ技術の適用による）採算のとれる新しい適用業務が増えている。

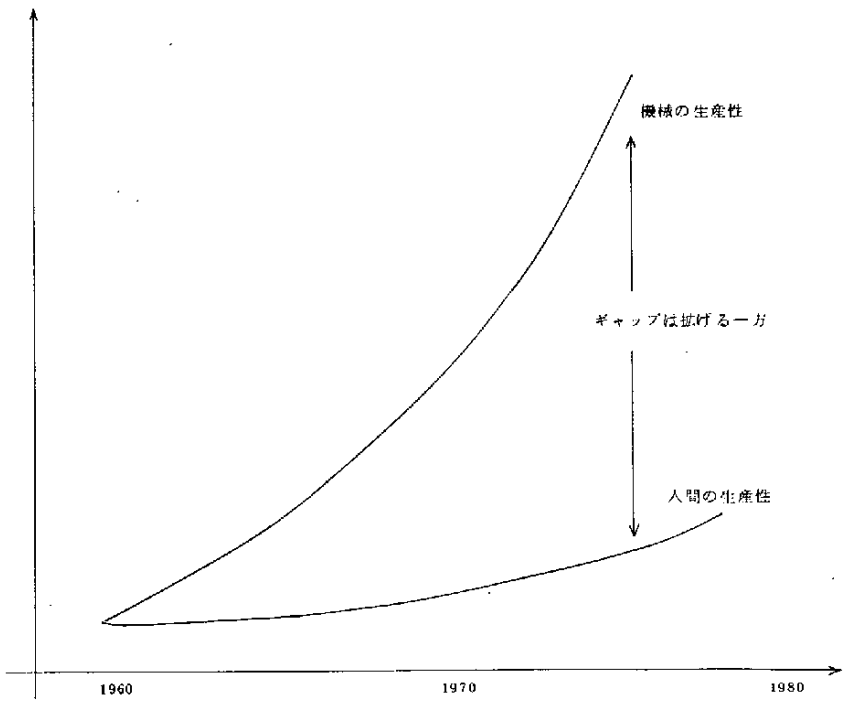


図 3.33　　ＤＰ　部　門　の　生　産　性

・ところが，プログラマの生産性は機械（計算機ハードウェア）の生産性に追いつかず，そのギャップは拡大の一途をたどっている。

・プログラムをつくる（人的）資源は限られている（ＤＰ部門の人数は増やしてもらえない）。

・その結果，適用業務のバック・ログ（開発待ちの適用業務）が増えている。

・プログラマの生産性を向上させることが急務である。

(7) ＥＬＩＡＳ（Entry Level Interactive Application System）は，ＤＰ部門の生産性を向上させるツールとして，とくにＤＢ／ＤＣ（データベース／データ通信）適用業務の開発保守の作業を軽減させることを目的に開発された。

(8) ＥＬＩＡＳの目標は次のようにまとめることができる。

・ＤＰ部門の生産性の向上

・ＤＢ／ＤＣ適用業務の簡単，迅速な開発

・より多くのＤＢ／ＤＣ適用業務の採算性を実現すること

・対話式に利用できること

・ＣＩＣＳ（Customer Information Control System，顧客情報管理システム，オンライン制御パッケージ），ＤＬ／Ｉ（Data Language／Ｉ，データベース・システム）に対する標準的なインタフェースであること。

・プログラム保守を軽減すること

(9) 容易に利用できるようにするため，ＥＬＩＡＳは次のような設計上の配慮がなされている。

・対話式

・案内方式（ＥＬＩＡＳ側から質問を発し，それに答えることによってプ

ログラムがつくられたりなど作業が進行する方式）

・オンラインの質問回答機能（ＥＬＩＡＳが出す問いの内容等が分らないとき，マニュアルを見なくても，端末から質問コマンドを出し，説明を要求することができる。

・英語，日本語（カタカナ）など６か国語が可能

・適用業務の設計のガイド

(10) ＥＬＩＡＳはデータベース管理に関して次のような手順（作業）をサポート（支援）している。

・データベースの定義

・データ・アクセスの定義

・ＤＬ／Ｉ適用業務制御ブロックの作成

・データベースに対するオンライン・アクセスの定義

・データベースの作成（生成）と維持（保守）

(11) ＥＬＩＡＳは適用業務プログラムの作成に関して次のような手順をサポートしている。

・プログラムの骨組み部分の指定

・スクリーン・マップ（オンライン適用業務における表示型端末のスクリーンのレイアウト）の生成

・データ構造の定義

また，以下に対するブリック（機能（function）や手続き（procedure）のセット（set））も用意してあり，プログラマのコーディングの手間が省ける。

・ＣＩＣＳに対するコマンド（・ストートメント）

・ＤＬ／Ｉ呼出しステートメント

・プログラム・コード

COBOLステートメント

PL／Iステートメント

DMS（Development Management System，表示型端末を使う適用業務の開発をサポートするプログラム・パッケージ）を（ELIASから）使う場合，そのユーザ出口

標準のエラー処理

(12) ELIASのブリック（brick）とは，あらかじめ用意されている標準コード（コーディング），ユーザ・プログラムの一部に組込まれる。ブリックは図3.34のような構造になっている。

(13) ELIASでかかれたプログラムは，例えば図3.35のような構成になる。

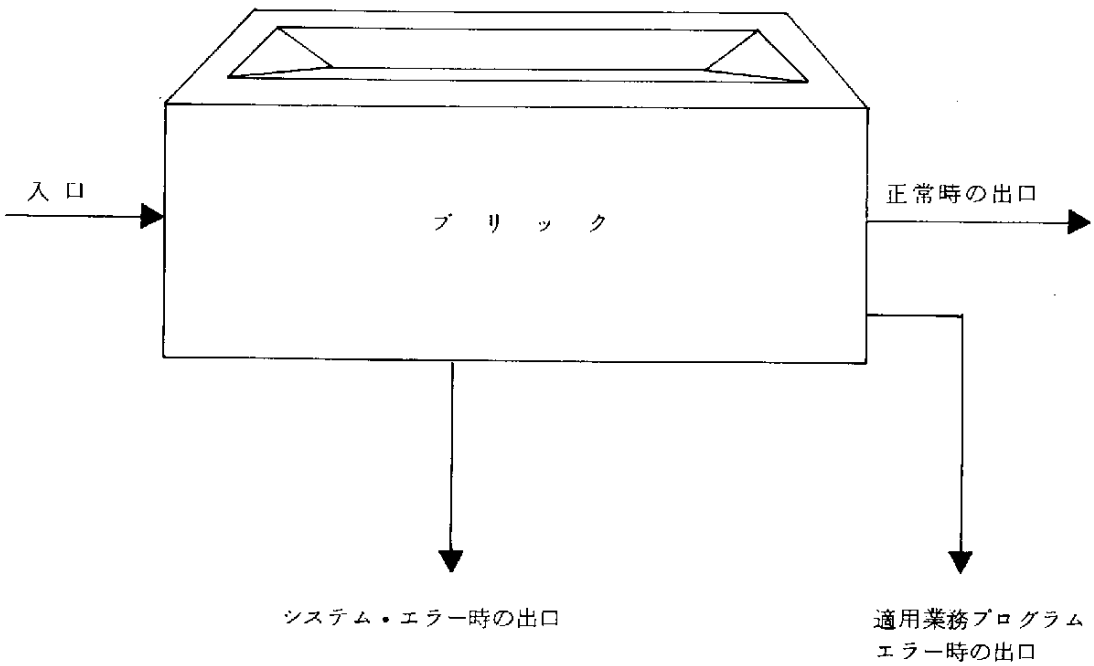


図3.34　ブリックの構造（入口と出口）

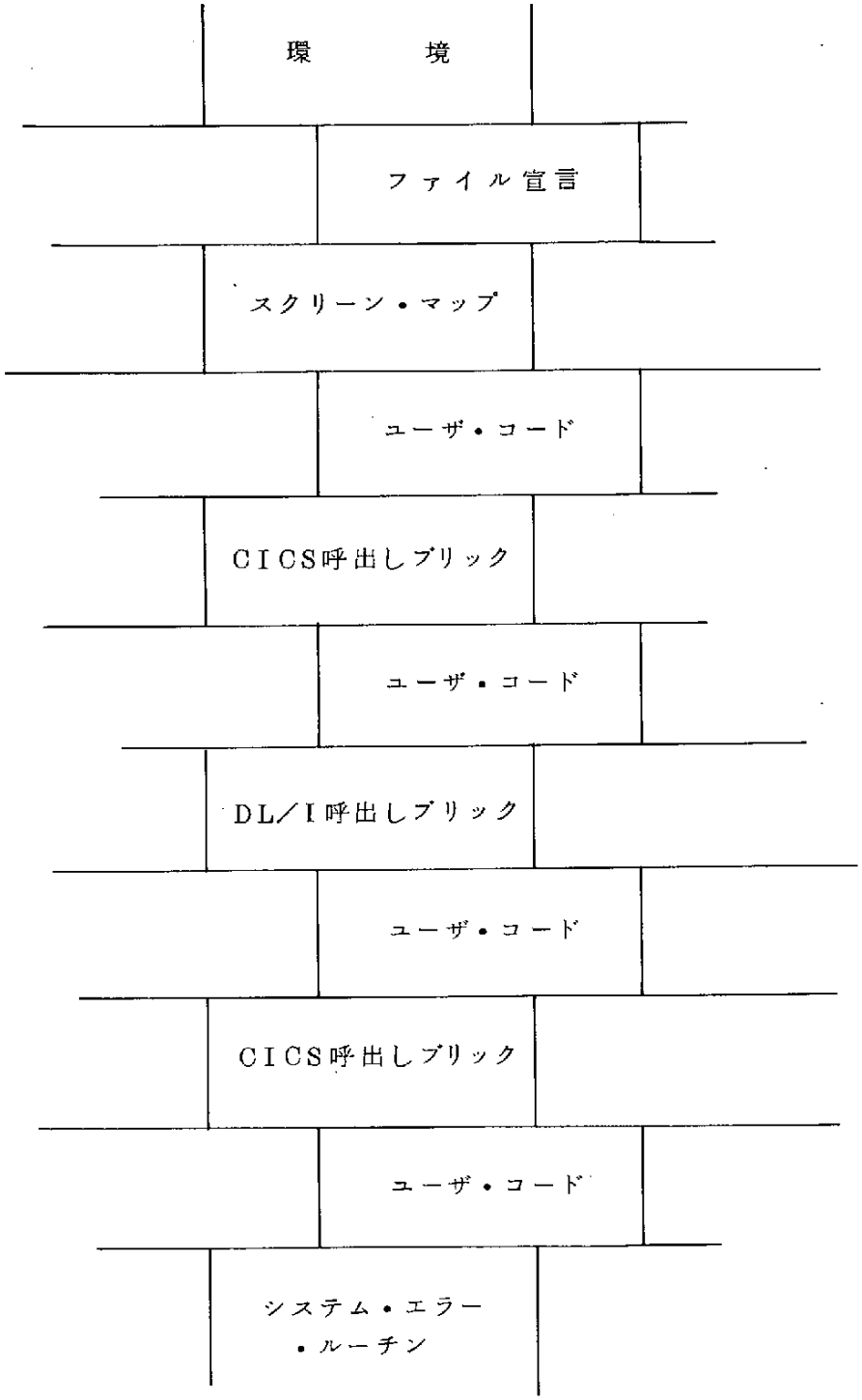


図 3.35　ELIAS（を使って作られた）
プログラムの例

(14) DMS／CICS／VSを使う適用業務に対しては，ELIASは以下をサポートする。

・DMSパネル（画面）の定義

・DMSファイルの定義

・プログラムの骨組みの指定

・ブリックによるDMSユーザ出口の作成

・DMS出口から出た後でのELIAS機能の使用

(15) ELIASシステム自身は次のような構成要素からできている。

・（ELIAS本来の）コード

・ブリック

・（ELIASの使い方を示す）使用例

・説明文書（ドキュメンテーション）とユーザ教育手順

(16) ELIASの利用により，以下の理由から，より簡単かつ迅やかなDB／DC適用業務の開発が可能になり，ソフトウェア開発維持の生産性が向上する。

・開発作業の軽減

・保守作業の軽減

・DP能力を最大限に引出す

・適用業務開発意欲の昂進

・ドキュメンテーション，教育の改善

(17) ELIASは端末から対話式に使う。そのときの（ユーザの）端末操作はログされており，その後の保守などで使うことができる。ELIASによりプログラムの生産性は1.5倍に向上，（適用業務プログラムの）60～70％のコードはELIASにより自動生成できている。

(18) ELIASはPL／S.EXEC2に似たIBM社内専用の言語で書かれている。マクロ機能も利用している。

3.9 ＩＢＭフランス，ラゴード研究所

La Gaude研究所は地中海に面した最大の都市ニース（Nice）から車で僅か20分のところにある。この研究所の沿革，任務等については，3.9.1節にまとめたような説明がClaude Pouget氏からあった。ここでは，Philippe Nay氏にも会い，通信技術の動向，ＩＴＩＲＣ（ＩＢＭ Technical Information Retrieval Center）の利用などについて調査した。

3.9.1 La Gaude研究所の沿革，任務など

La Gaude研究所はもともとパリにあったものを1962年ここに移したものである。地中海に面したこのLa Gaude（ラ・ゴード）を選んだ理由は，第1に太陽であり，第2にニースの国際空港である。La Gaude研究所の建物の床面積は合計36,000m^2である。

ＩＢＭの研究所のうち，Yorktown(U.S.A)，San Jose(U.S.A)，Zurich（スイス）は基礎研究を，残りは製品開発研究を担当している。ヨーロッパでは，La Gaude，Boeblingen（ドイツ），Hursley（イギリス）が主要研究所である。ＩＢＭは従業員の7％，全予算の7％を研究開発に投入している。研究所員は，アメリカ合衆国で20,000人，全ヨーロッパと日本で5,000人である。La Gaude研究所員は1,300人，そのうち59％が研究員（プロフェッショナル），23％が研究補助（テクニカル），18％が管理（アドミニストレーション）である。管理部門をできるだけ小さくし，研究員の割合を増やすことが重要である。（Pouget氏個人としては）研究者の生産性は考えたくないと言っている。

La Gaude研究所の任務は通信技術に関わる将来の製品を開発することである。基礎研究ではなく製品開発研究であるところに難しさがある。この研究所では，パリにあった25年前から通信に関する研究を開始し，

La Gaudeに移ったあと1962年から2年間通信の質をじっくりテストした。その結果ＰＴＴ（フランスの通信会社）の技術等についてよく分った。翌1963年ＩＢＭ3863モデムを開発，さらに，ライン・スイッチング装置等を開発（フランス，イギリス，ベルギー，イタリアなどで販売）している。1979年には，La GaudeとSan Jose（U.S.A，カリフォルニア）間を通信衛星で結ぶ実験に成功している。

ここでは，システム／370モデル158が4台，モデル168が1台，モデル145が1台使われている。4台のモデル158はすべて研究所内のサービスに，モデル168は外部との通信に，モデル145は保守用として使われている。所内には約400台の端末が配置されており，端末の普及率は非常に高い。端末の普及については次のように予測している。

	1980年	1986年（予測）
U.S.A.全体	48人に1台	10人に1台
ＩＢＭの顧客	25 〃	6 〃
ＩＢＭ全　体	4.8 〃	2 〃

また，グラフィック表示装置も有効に使われている。

3.9.2　通信技術の動向

(1)　ＩＢＭの通信関係の歴史

1960～1964年

ドイツ，フランス，イギリスで1,200b/s（ビット／秒）の伝送テキスト。Error Checking Code

1964年

システム／360

1970年

システム／370

1974年

SNA（Systems Network Architecture，システム・ネットワーク体系）

1977年

通信衛星による1.5Mb／s（メガ・ビット／秒）の伝送

1979年

通信衛星による2.5Mb／s での4セットの計算機システムのネットワーク

(2) 計算機と通信の結合

図3.36 のように計算機技術と通信技術が結合され，その結果人間社会（全体）の能力が拡げられる

(3) 計算機コストと通信コスト

計算機の1命令当りの価格は図3.37 のように下っている。一方，通信コストはそれほど急速には下っていない。データ記憶／処理のコストと通信コストの下り方の差を図3.38 に示す。

(4) 通信ネットワークの解放

計算機から見た場合，通信ネットワークとのインタフェースのどの部分から解放されているかは国によって異なり，図3.39 のようになっている。

(5) 通信に対する要求

次のような要求がある。

・非常に短いメッセージの通信

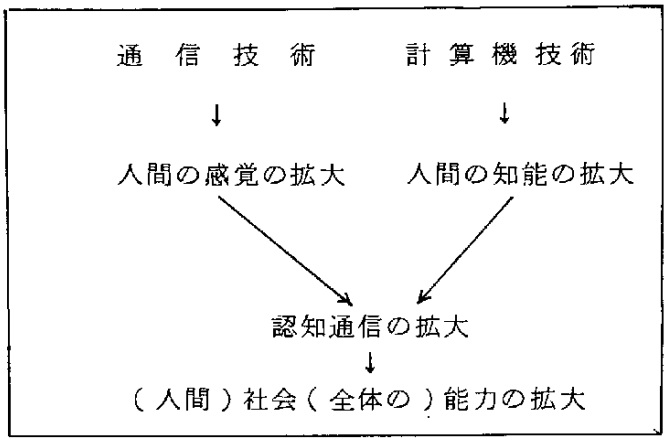


図 3.36　通信技術と計算機技術が結合すると

- 非常に長いメッセージの通信
- 非常に密度の高い通信
- 非常に密度の低い通信
- 高速な応答
- 正確度
- 信頼性
- 複数アクセス

(6) X.25

IBMはX.25 パケット・スイッチングをフランス，イギリス，ドイツ，オランダでサポートしている。EURONET で官庁用にX.25 をサポートしている。

(7) 光ファイバー

銅線に比べて，安く，信頼性が高く，大容量で，損失が少なく，敷設しやすいのでこれから急速に普及すると考えられる。

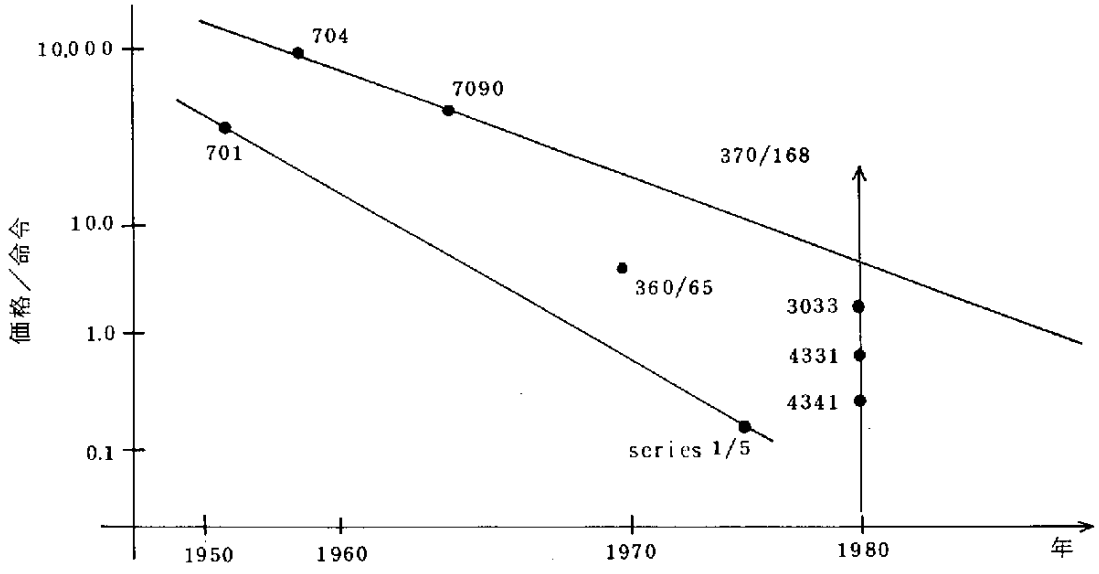


図 3.37　　価格／命令の推移

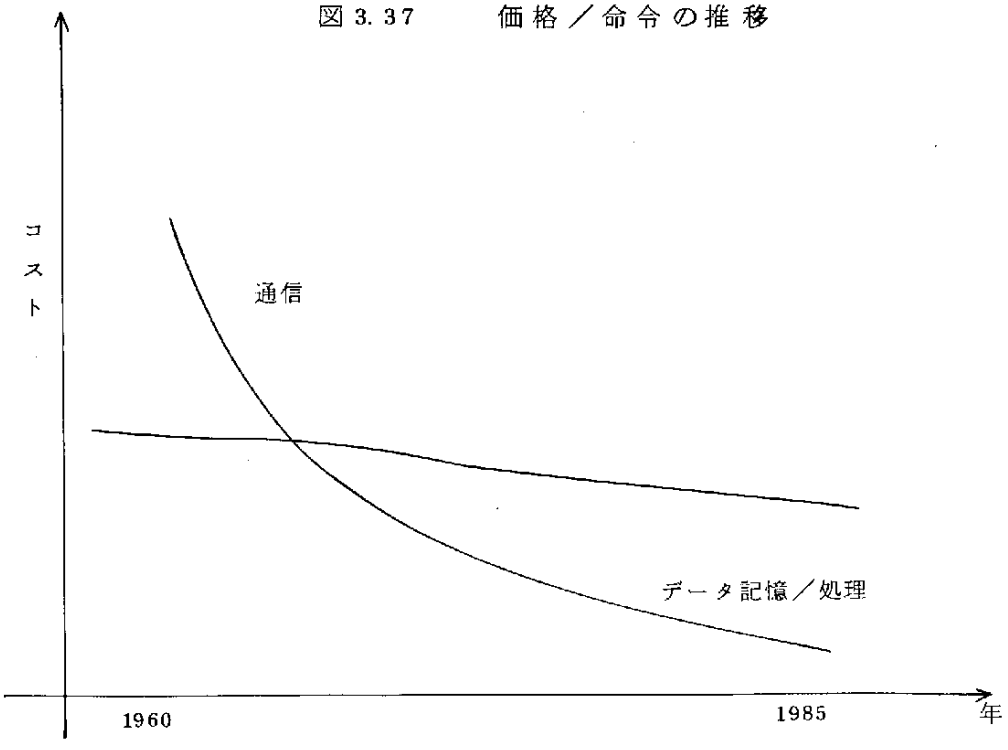


図 3.38　通信コスト，データ記憶／処理コストの推移

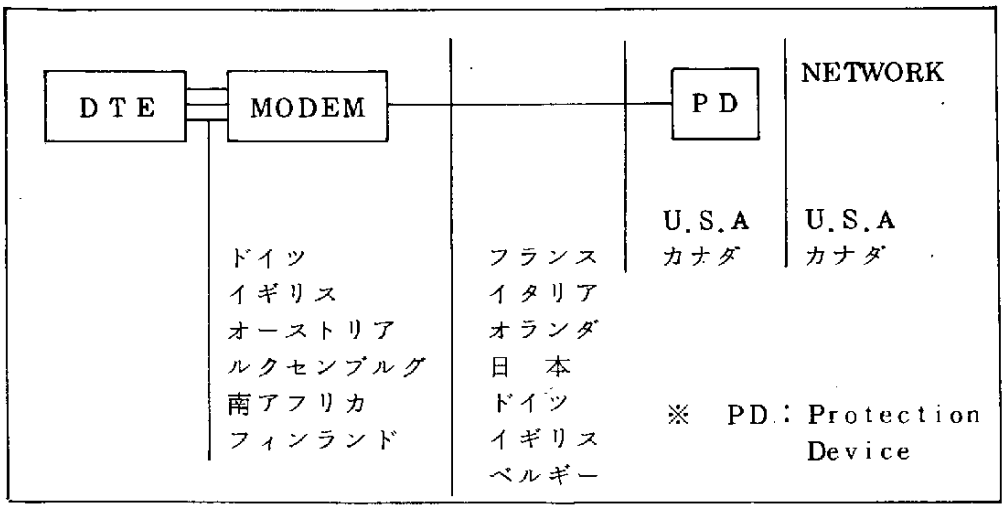


図 3.39　通信ネットワークと計算機とのインタフェース解放の違い

(8)　衛星通信

データ伝送能力が高いこと，ブロードキャスティングであることがとくに重要で，この特長を活かした応用に広く使われるようになる。通信衛星には，

INTELSAT（国際用）

MOLNIYA（ソ連）

SYMPHONIE（ヨーロッパ）

PALAPA（インドネシア，国内用）

ANIK（カナダ，国内用）

SBS（U.S.A，国内用）

などがあり，これから 20 年の間に，総数 200～300 になると予想される。

(9)　衛星通信のコスト

衛星通信の宇宙コストと地下コストの推移を図 3.40，図 3.41 に示す（J. Martin より）。

(10) ＩＢＭの衛星通信の歴史

1962年　TELSTARを用い1200～2400b／sで

1965年　EARLEY BIRDを用い40.8Kb／sでフランスとU.S.A間で

1973年　ANIKを用いU.S.A.国内で

1977年　SYMPHONIEを用い，1.5Mb／sでLa GaudeとGaithersburg(U.S.A.)の間で(IBM+COMSAT+PTT)

1979年　SYMPHONIEを用い，2.5Mb／sでヨーロッパ，U.S.A.の4つの計算機システムを結ぶネットワーク(IBM+COMSAT+PTT.DFVLR)

(11) 衛星ネットワークの実験

まず，SYMPHONIEを用い，point-to-pointでデータ伝送を実験し，エラー率10^{-8}～10^{-9}という上々の成果を得た。

これをふまえて，COMSAT(Clarkuburg.U.S.A.)，IBM Gaithersburg(U.S.A.)，DFVLR(Weilheim，ドイツ)，IBM La Gaude（フランス）の4つの計算機システムを結ぶ通信衛星によるネットワークを完成した。

さらに，このネットワーク上で，ロード・シェアリング(load sharing)，資源シェアリング(resource sharing)を実験中である。とくに，分散データベースに力を入れている。

3.9.3 ＩＴＩＲＣ

ITRIRC(IBM Technical Information Retrieval Center)は，計算機を利用した検索，配布システムで，1965年に(全)IBM本社に設置された。ITIRCの目的は，全世界のIBMおよび関連の科学者，

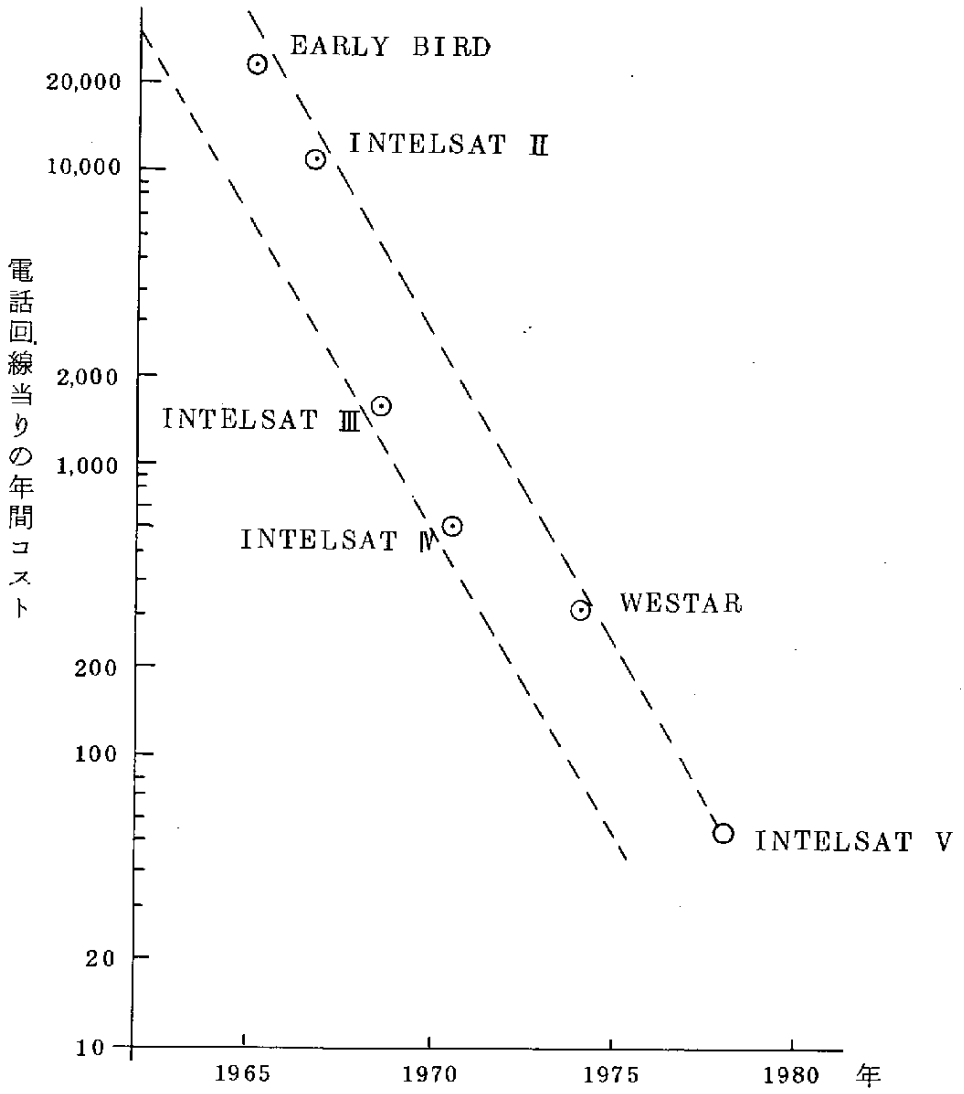


図 3.40　通信衛星の宇宙コストの推移

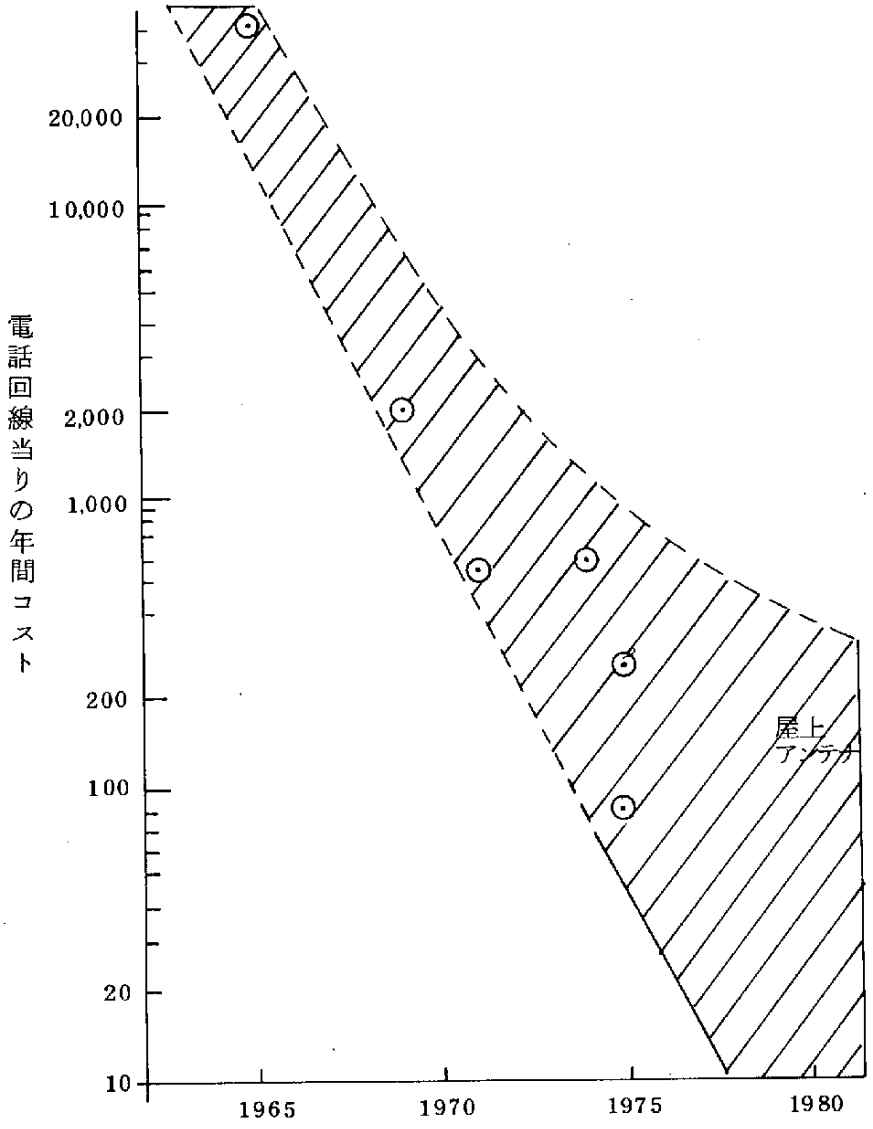


図 3.41　地上局のコストの推移

技術者，その他専門的仕事をする人々に必要な情報を提供することである。

まず，情報は，ＩＢＭの出版物（研究所報告，標準規格，特許，参照マニュアル，技術ジャーナル，その他多くの文書），さらには社外のドキュメント類から必要とされるものを広く収集する。これらのドキュメントに関する説明情報（タイトル，著者名，出典その他，100～200語の梗概）をキー入力する。社外のドキュメントの多くはまとめて磁気テープに記録され売られているので，そのテープを購入し，計算機によってITIRC形式に変換する。

問合せには2つの形式がある。1つは全体質問で，（履歴）ファイル全体の（積みかさね）すべて対して質問するものである。他の1つは興味の概観と呼ばれる持続質問で，システムに入力されてくる新しいデータだけを探索する。いずれの場合も利用者は普通の英語で質問し，訓練された検索スペシャリストがそれを適切な計算機（用）ロジックにかえる。全体質問の多くは回顧探索といわれ，毎日バッチ処理される。印刷された梗概は翌朝郵送される。しかしながら，端末を介してネットワークにつながる（図書）施設の数が増えてきたので，利用者は対話式でファイルにアクセスし，リアル・タイムで質問の精度を高め，最適の答を得ることができるようになっている。もし出力が多い場合には，夜中にオフラインで印刷して利用者に送るようにしている。持続質問の処理は毎週行なわれ，その印刷出力は郵送される。この出力はＣＩＳ（Current Information Selection）といわれ，毎週何千人もの利用者に送られている。

利用者は質問と一致した出力を受け取り，ドキュメントに関する梗概等を調べる。ＣＩＳ出力には，ドキュメント全文のコピーを各地のＩＢＭの図書館あるいはITIRCに要求するための申込み用紙がついている。同様に，回顧探索でもＣＩＳと同じような説明報情が出力され，梗概を調べた

後必要なドキュメントのコピーを申込むことができるようになっている。

1975 年末でのドキュメントの収集，記録状況を表3.9に示す。

ＩＴＩＲＣ検索配布プログラムでは，「標準テキスト」探索技法が使われている。計算機による探索は，入力，すなわち，タイトル，出典など，キーワード索引，梗概のすべての語に対してなされる。ＡＮＤ，ＯＲ，ＮＯＴ，ＡＢＳＯＬＵＴＥなど多くの論理演算子が使える。これらを用いて，独立の単語，語根，同義語，句，文，段落に対して演算操作することができる。従って，非常に柔軟かつ正確な探索が可能である（表3.10参照）

入　　力	年間追加件数	累積履歴ファイル（1974年）
ＩＢＭ社内	8,000	125,000（1950年から）
National Technical Information Service，その他の外部文献	20,000	109,000（1965年から）
Engineering Index（2000を越える技術ジャーナル，会議，シンポジウムの報告）	40,000	285,000（1966年から）
Chemical Abstracts Service	30,000	228,000（1969年から）
American Institute of Physics	30,000	127,000（1971年から）
1975年末で約1,000,000の梗概		

表3.9　ＩＴＩＲＣに登録されているドキュメント

ＩＴＩＲＣプログラムはＴＥＸＴ－ＰＡＣともいわれ，バッチ・モードで，高速に動かすことができる。従って，大量の利用者プロフィルや，非常に大きなデータ・バンクに対する探索質問を経済的に処理することができる。単価は，現実には，システムが大きくなるにつれて減少する。ＴＥＸＴ－ＰＡＣはタイプⅢパッケージとして公開されており，多くのＩＢＭカスタ

マに導入されている。

ITIRCでは，STAIRS(STorage And Information Retrieval System)というプログラム製品が使える。これにより端末ネットワークが可能になり，回顧質問に新しい可能性が出てくる。すなわち，多くのIBM（の図書館）施設（そこに端末がおかれている）の端末からITIRCファイルに直接アクセスできるようになり，バッチの探索プログラム論理）を用いて，対話式で，問合せ条件をつくり，テストし，変更することができる。

論理演算子	機　能	使い方
OR	同義語を集める	visual OR video
AND	共通にふくむ	design AND automation
ADJACENT	語の順序	cathode ADJ ray
WITH	文中に共通にふくむ	remote WITH terminal
CONTROL	段落の指定	1967 CONTROL 000
MASKING	語の終りを集める	automat $ $ $
ABSOLUTE	必ずふくむことを指定	ABS laser
NOT	ふくまないことを指定	NOTsoftware
CAPITALIZATION	上段／下段の区別	ⓐⓐ and

表3.10　問合せで使える論理演算子

従って，非常に柔軟で，効率のいい探索ができる。簡単化された回答や小さな出力は直ちに表示，あるいは印刷されるが，大部の出力は中央の計算機でオフラインで印刷するよう指定できる。端末による遠隔利用が着実に増えており，バッチでの探索と人手による検査という使い方は，対話式利用に置き換っていると思われる。

この研究所内の図書施設にある端末からキー・ワード「UTSUNOMIYA」で回顧検索を試みた。その結果のＣＩＳを図3.42 に示す。論文そのものはオランダにあるということであった。このようなシステムをもち，研究に活用していることは，当然そうあるべきだし，できるとも思うが，やはり立派である。当り前のことを当り前のようにやることは案外難しいことだし，研究にこれだけのお金をかけることも難しい。また，１社の１つのシステムで，これだけ大きな累積履歴ファイルをもち，利用していることも注目に値する。

```
   ***  DATA BASE : EIXF -  UNCLASSIFIED  ***                    PAGE  0001@

SEARCH - QUERY
00008 UTSUNOMIYA .AUTHORS.5
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 5
TITLE      ON A UNIVERSAL DESIGN PROCEDURE TO REALIZE THE SEMIMODULAR STATE
           TRANSITION GRAPH.5
           LOGIC DESIGN5
DOCNO           EIX780400228
AUTHORS         Nakumura, T.5
                Utsunomiya, K.5
ABS-NO          026689
SOURCE          019634
                Digital Processes v 3 n 3 Autumn 1977 p 237-257
                DGPRB
CORP-AUTH       Nagasaki Univ, Jpn5
ABSTRACT        A universal design procedure for the semimodular state
           transition graphs is described.  The circuits constructed from the
           procedure comprise three logic devices: AND, NOR and MAJORITY in5
           their usual sense.  The most fundamental idea is that the
           semimodularity makes it possible to realize a whole circuit by
           interconnecting unit circuits which are designed separately, while a5
           difficulty still exists in constructing each unit circuit to be
           hazard-free where the input changes occur independently of one
           another.  A concept of completely regular extensions is introduced,
           which is claimed to be more convenient than that of good extensions
           by I.  Kinura when designing a speed-independent circuit by
           connecting unit circuits.  A completely regular extension C* of a5
           circuit C is another circuit with hidden nodes in addition to the
           nodes of C so that the behavior of C* may be considered equivalent to5
           that of C.  17 refs.5
CATEGORY        00-A721
                00-A723
KEYWORDS        LOGIC DEVICES
                ASYNCHRONOUS CIRCUITS

        - - - -   E N D   O F   P R I N T O U T   - - - -
```

図 3.45　ＩＴＩＲＣ出力の例

4 おわりに

訪問した先々で，誠意ある，熱心な応対をうけ，所期の調査目的を達成することができたと思っている。ヨーロッパ各地に散在する施設を9か所も訪問したため，必ずしも議論する時間がじゅうぶんにはとれず，うしろ髪をひかれる思いで辞したところも何か所かあった。

ソフトウェアの生産性向上は1980年代の情報処理分野の最大の課題である。毎日々々がそのために苦しみ，あえいでいる。しかし，スマートな特効薬的な手法，ツール等はまず期待できない。すでに考えられ，開発されている手法，体制，技術，ツールなどを総合的，効果的に組み合せていく以外に道はないという気がする。平凡で，ごく現実的であるが視野を拡げ，腹をすえて，展望した道を進みつつ着実に実績を積み重ねていくことが唯一の解決策ではなかろうか。

最後になりましたが，訪問先で観迎，協力して下さった方々はもちろん，国内で種々お骨折りをいただいた，海外推進担当をはじめとする日本アイ・ビー・エムの方々，筑波大学の西村敏男，今野浩両先生，（財）日本情報処理開発協会の方々に心から感謝致します。また，本報告書作成にあたり，不明な個所を一部日本アイ・ビー・エムを通して補足したことをお断りしておきます。

代表して　　宇都宮公訓

附　　録

A．データベース／データ通信指向の適用業務設計手法

（IBM資料より）

（1） 適用業務の設計

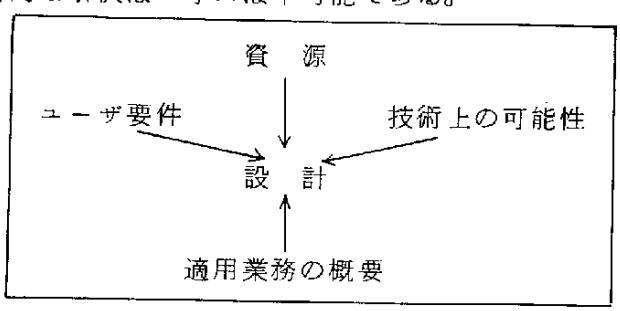
図A 1に示すように，適用業務の設計に際しては，ユーザ要件，（開発，資源，技術上の可能性といった対立し合う要素間のトレード・オフを考えて，作業を進めなければならない。

（2） 理想的データベース環境の特長

データベースを前提とした適用業務の導入・実施でもっとも重要なテーマは次の3つである。

- データの冗長性
- プログラムの独立性
- データ間の関係付け

データベース・アプローチをしなければ，適用業務は図A 2のような性格を示す。しかし，設計者が目指しているのは図A 2の右側のような理想的な解決である。しかし，通常は資源や時間がかかり過ぎて，この極度に理想的な解決は一挙には不可能である。



図A 1.　適用業務の設計

（3） データベース設計上の主要なトレード・オフ

現実のデータベースによる解決は，図A 3のように，図A 2に示す両極端の間のどこかになる。実際の設計では，できるだけ理想に近ずこうとし

	非データベース環境	理想的データベース環境
データの冗長性	複数個所の更新不一致	冗長性の排除一貫性
プログラムの独立性	プログラム構造は物理ファイル構造に依存	プログラムは必要なデータだけを参照
データ間の関係付け	ファイル生成時にデータ間の関係が固定	必要なときにデータ間の関係を設定

図A 2. 理想的データベース環境とは

て，余りにも多くの資源を投入したり，時間の無駄をしないようにすべきである。そのため，設計者は，理想と現実の制約とのトレード・オフを詳細に定義しなければならない。

(4) データベース指向の適用業務を設計する際に解決すべき問題

データベース指向の適用業務を設計する際には図A 4に示すような問題を解決しなければならない。ここで紹介しようとしている手順化された手

非データベース環境 →	トレード・オフ	← 理想的データベース
複数個所の更新不一致	コントロールされた冗長性	冗長性の排除一貫性
プログラム構造は物理ファイル構造に存存	コントロールされたプログラムの独立性	プログラムは必要なデータだけを参照
ファイル生成時にデータの間の関係が固定	もっと頻繁に使われる関係付けはまえもって固定	必要なときにデータの間の関係を設定

図A 3. データベース設計上の主要トレード・オフ

検討すべき業務処理活動は何か
これらの活動の発生場所はどこか
どのようなデータがこれらの活動に関係するか
どのようにデータが使われるか
詳細度の適正規準をどこに設定すべきか
データ構造を表わす言語があって，それでどのように
データ構造を定義できるか
データベースの論理構造および物理構造をどのように
定義すべきか

図A 4. データベース指向の適用業務を設計する際に解決すべき問題

法はそのためのものである。

(5) なぜ方法論（手順化された手法）が必要か

場当り的な方法ではなく，手順化された手法を用いる利点には次のようなことがある。

- データベースは社内の異なる部門の共通のシステムである。データ通信ネットワークと結合して使うことが多く，適用業務環境が複雑になる可能性がある。
- データベースの設計過程は繰返しされる。すなわち，各適用業務に対して，決められた順番のステップをたどることが多い。
- 設計過程の管理（日程やコスト）が可能になる。
- 設計技能を他人に伝えることができる。

(6) データベース／データ通信システムの設計と導入

データベース／データ通信システムの全体的設計は，同じデータ，同じ通信ネットワークを使う種々の適用業務の導入実施よりまえに行なわなければならない。

その段階では，同一資源を使うと考えられるすべての適用業務を詳細に分析することは不可能である。詳細分析に長い時間がかかり，解決案が出来た時点では，それははや陳腐化してしまった状態になりかねない。ここで述べる手法は，適用業務を概略的に記述することから始め，数週間のうちに，情報システムの主要構成要素を定義し，解決策の技術評価ができるようにする。この適用業務予備検討段階では，適用業務開発グループ，適用業務保守グループとは異なる役割りをもつ，設計グループ（ユーザとデータ処理部門）が設計を行なわなければならない（図A 5 参照）。

(7) データベース／データ通信システムの概略設計と評価

（適用業務の）予備検討段階では，次の4つの作業を行なう（図A 6 参照）。

- 目的の定義

  検討対象分野，同一データベース・同一通信ネットワーク上で動く（と考えられる）適用業務，およびその目的を定義する。

- 機能仕様

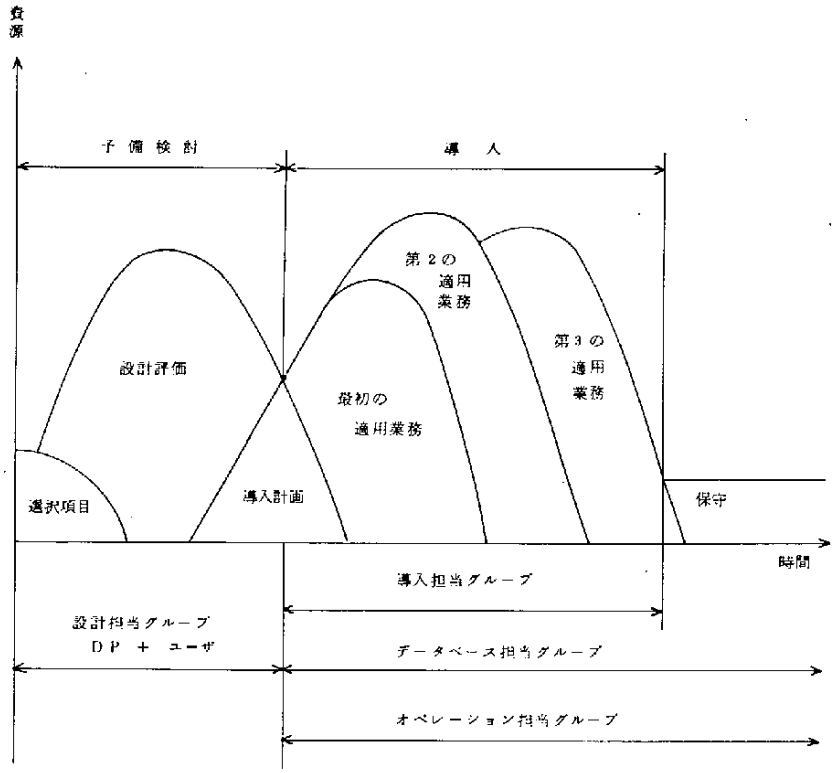
  2つのケースがある。すなわち，既存の適用業務を設計し直したい場合と，新規の適用業務を設計したい場合である。前者の場合，適用業務は既知であり，作業はすぐ終る。後者の場合は，機能仕様を作成するためユーザのニーズの理解から始めなければならない。

- システム設計と評価

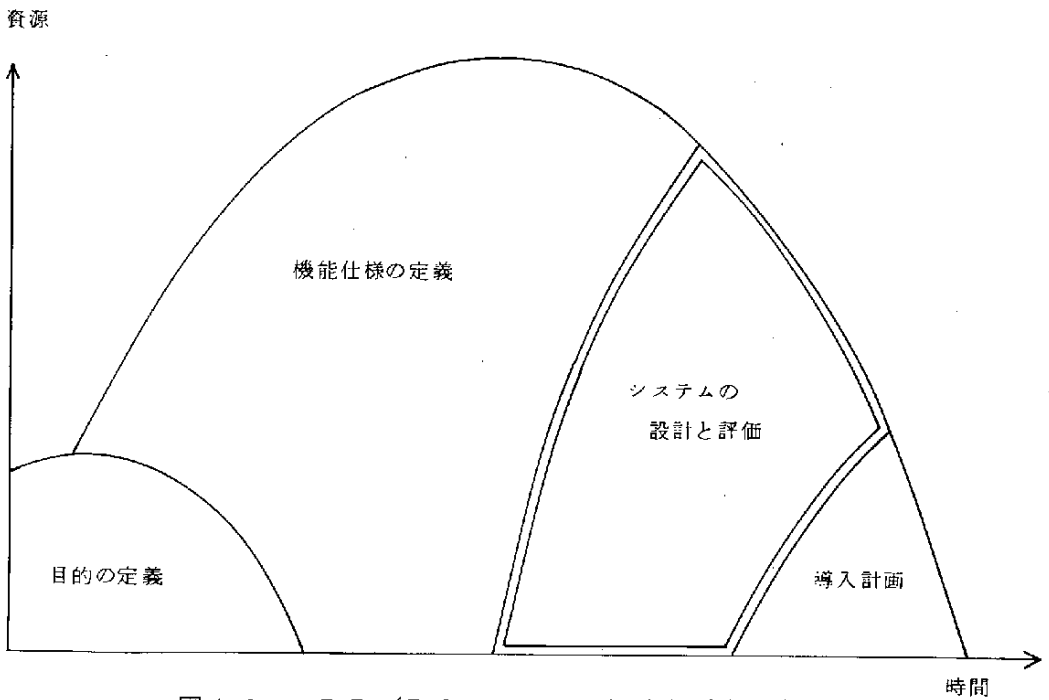
  機能仕様から始めて，情報システムのすべての構成要素を設計し，それをもとに技術評価を行なう。

- 導入計画

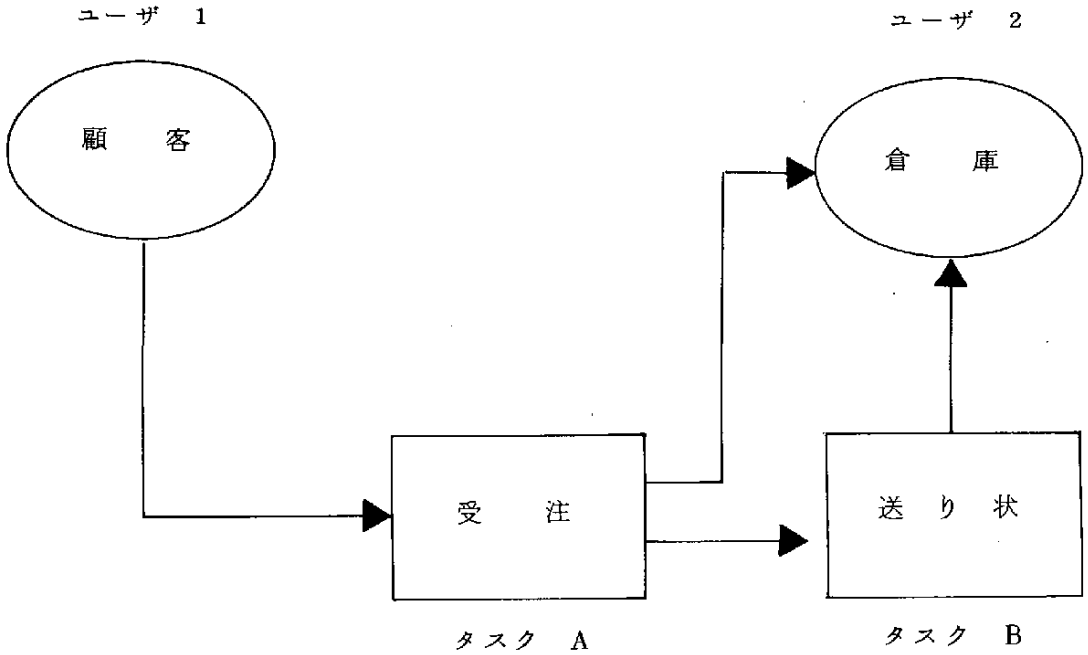
  これまでの過程で得られた資料をもとに異なる適用業務ごとに導入計画を作成する。



図A 5　　DB／DCシステムの設計と導入の計画と資源



図A 6　　DB／DCシステムの概略設計と評価



図A 7　　タ ス ク の 識 別

(8) 機能仕様の定義

ユーザの（情報システムに対する）ニーズを理解する効果あるやり方は，タスク，ユーザ，情報交換の観点から，（従来，将来の）適用業務を記述してみることである。

(9) タスクの識別

ユーザは，直接，あるいはデータ処理システムを介して情報を交換する。タスクとは，システムとユーザの間での情報交換の橋渡しにあたる作業のことをいう。例えば，図A 7 で

タスクA：・顧客からの注文を記録する。

・積込み伝票を印刷し，送り状の作成を要求する。

タスクB：・関連する品物を配達するため倉庫に送られる送り状を印刷する。

のようなものである。

(10) タ　ス　ク

業務の基本作業を「オペレーション」と呼ぶことにすると，タスクは，「組織化された一連のオペレーション」ということができる。タスクは，

- 手作業，自動
- 情報の生成，受取り
- 反復される情報交換

などのオペレーションからなる。タスクは，次の3つの属性をもつ。

- アクション：タスクの結果は論理的かつ恒久的に一貫性をもっていなければならない。
- 時　　間：結果は各情報交換ごとに繰り返えされる中断のない一連のオペレーションによって作り出されなければならない。
- 場　　所：一連のオペレーションは，オペレータ，機械で，同一のワーク・ステーションで行なわれなければならない。

(11) タスクの識別と実行順序

タスクを識別するためには，サブシステム（適用業務の範囲）において処理される主要な「オブジェクト」を調べなければならない。例えば，受注書，パッケージ，品物，信用ファイル，保険証書などのオブジェクトは時間とともに変化する。タスクが生成されたときあるいはサブシステムに入ったときから，キャンセルされるまであるいはサブシステムの出口までを，これらのオブジェクトの変化あるいは「流れ（flow)」という。

オブジェクトの変化は通常不連続的である。受注書がつくられ，送られ，送り状がつくられて，最後に出荷がなされる。各段階での状態の変化を生じさせるものが「イベント」である。例えば，新しい顧客の受注書を受け

とることは，システムに対して受注レコードの作成を開始させるイベントである。

各段階において，状態の変化に対して起る1つ以上の情報の変更は互いに関連し合っている。従って，タスクはオブジェクトの状態を変化させると考えられる（図A 8参照）。

(12) ユーザ・タスク図

タスクの数が多い場合，「ユーザ・タスク図」を使うと便利である。図A 9の例では，

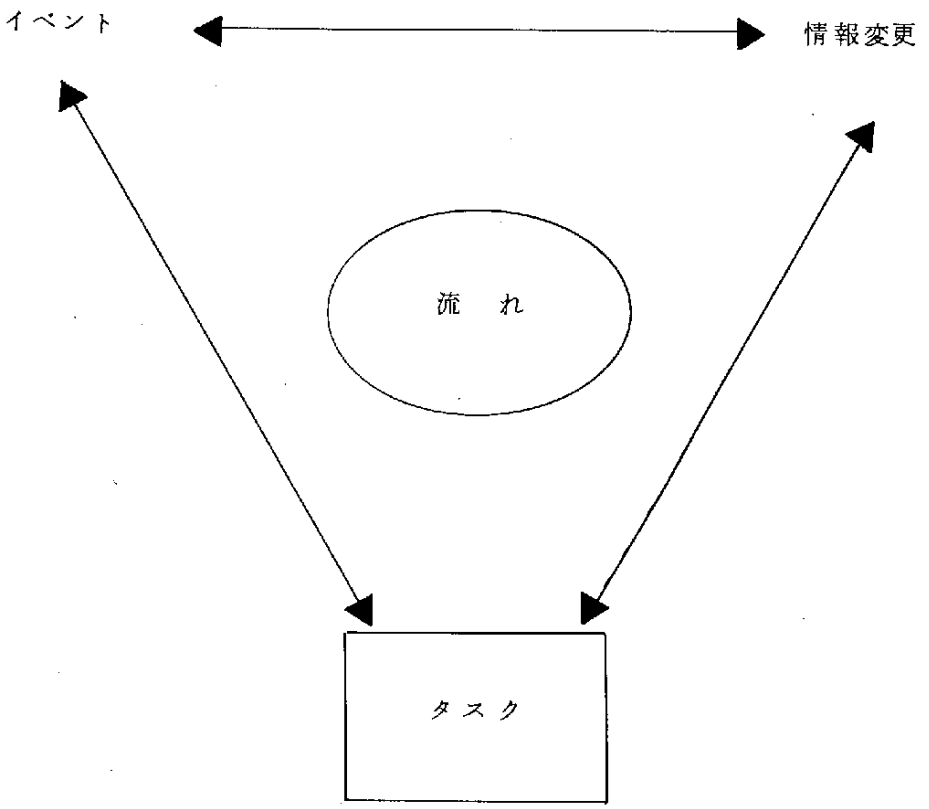
- 受注書は磁気テープで，サブシステムの外から到着し，DP（データ処理）センターでバッチ・モードで収集される。
- 受注品目の取扱いでは，自動割当てが行なわれる（与えられた個数の部品が各受注に割り当てられる）。
- 自動割当てされなかった受注は例外レポートとして出力される。このレポートを用いて，在庫管理者は，画面を通して選択的に受注を割り当てる。
- 自動的であれ，在庫管理者による選択割当てであれ，割当て後の（次の）タスクは，包装リスクを作成することである。これらのリストが倉庫にとどくと，品目の出荷が開始される。

これは現行のシステムと考えた例であり，これから将来のシステムを表わすように漸次変形される。

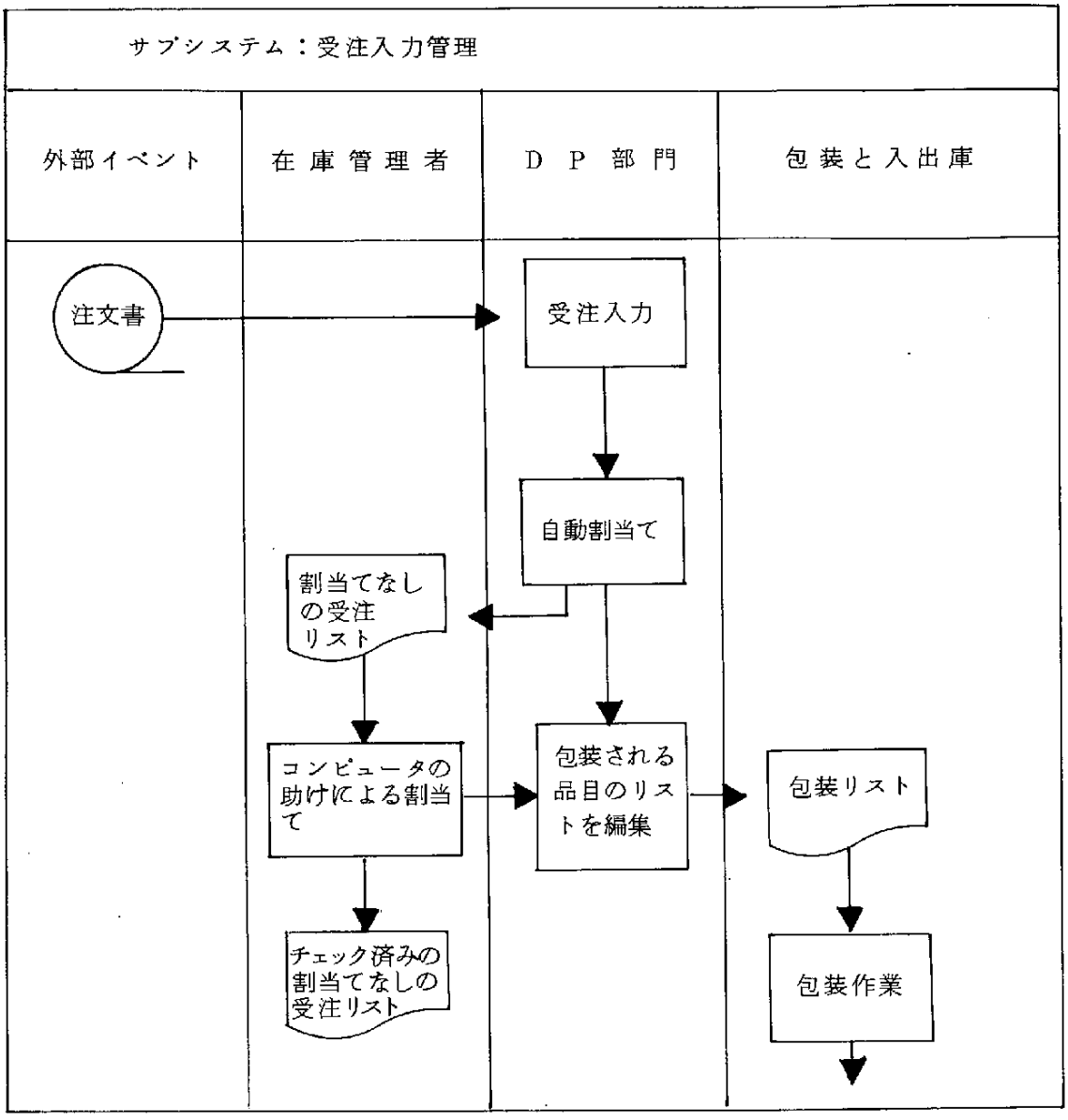
このように，「ユーザ・タスク図」は，機能仕様を簡単にながめたものと見ることができる。

(13) ユーザ・タスク記述書

サブシステムの機能仕様があまり明白でないときは，「ユーザ・タスク図」に，各タスクの短い記述を加えるとよい。そのための書式が「ユーザ



図A 8　　タスクの識別と実行順序


サブシステム：受注入力管理
外部イベント
在庫管理者
DP部門
包装と入出庫
注文書
受注入力
自動割当て
割当てなしの受注リスト
コンピュータの助けによる割当て
包装される品目のリストを編集
包装リスト
チェック済みの割当てなしの受注リスト
包装作業

図A 9　　ユーザ・タスク図

タスク記述書」である（図A 10）。

この例では，対話式（計算機とやりとりする）タスクの記述になっている。左側の列には手作業のタスクだけを，右側の例には計算機のタスクだけを記入している。15 行もあれば，タスクを記述できる。

この書式には，次の情報も記入されている。

- タスクの名前，モード，頻度
- 実行される場所
- タスクが作り出す文書とその行き先

可能であれば，ユーザ・タスク記述書」はユーザやシステムによる各タスクの頻度も記入しておくとよい。

「ユーザ・タスク図」，「ユーザ・タスク記述書」は，本手法で提案している機能仕様の表現方法であり，これ以後の（設計）段階と一貫性をもって使われるものである。

(14) データの構造化の方法

データを（互いの関係を保たせながら）構造化する方法を述べる。この方法は次の 3 つの部分からなっている。

- タスクごとに行なうデータ分析
- データ関係図の作成
- データ関係図を用いたデータ構造の決定

(15) 情　報　交　換

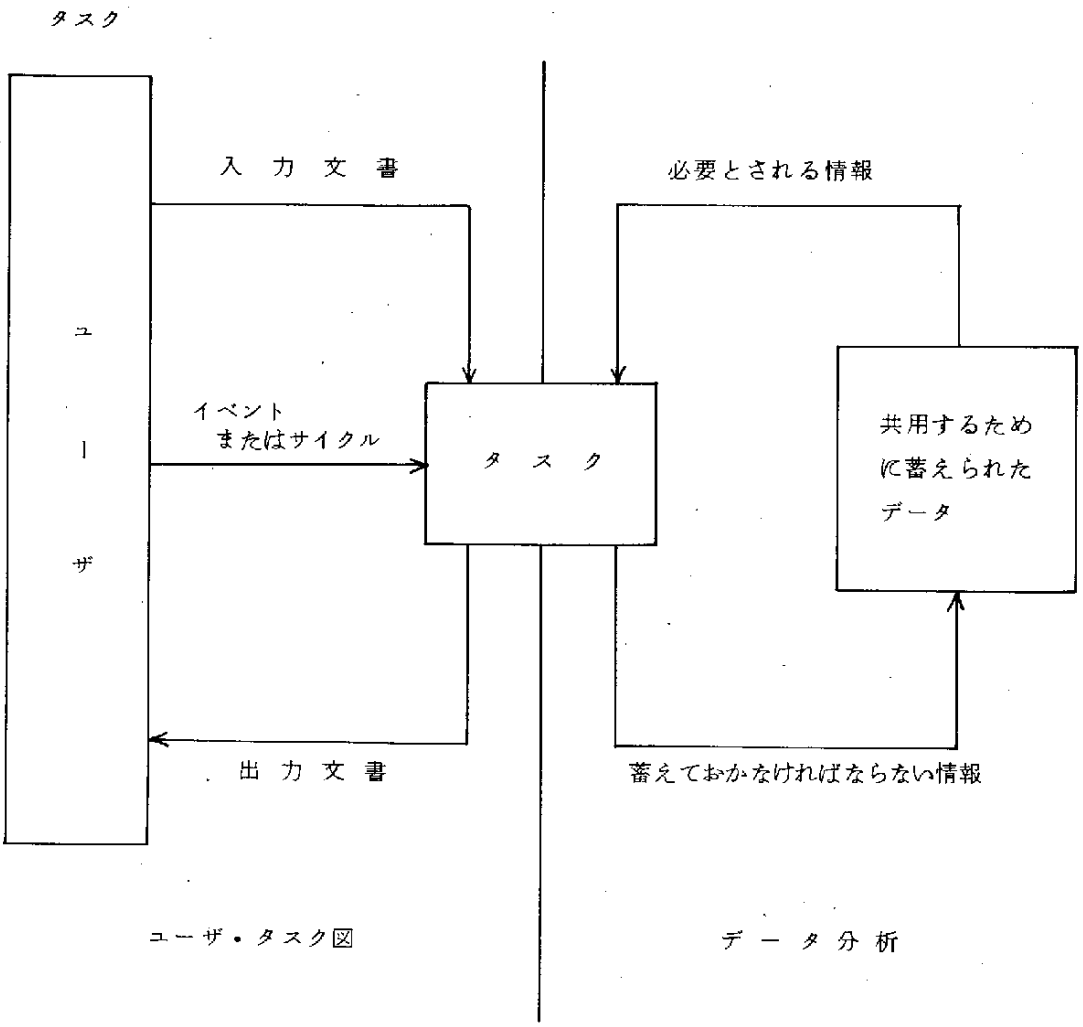
図A 11 で，左側はタスクの識別になっており，右側はタスクがデータにどのように関係ずけられるかを示している。タスクの実行（自動化されているか，これから自動化されるか）のためには，例えば，あるコントロールのために一般に補足情報にアクセスしなければならない。タスクの実行によって，後日使われると考えられるデータができるかも知れない。そ

UTDF	タスク名：コンピュータの助けによる割当て	コード ORD06	モード オンライン	回数 540
場所：受注センター		位置：本　社		
イベントまたは入力文書	割当てなし受注リスト			

F	ユーザ・アクション	F	システム・アクション
1.	割当てなし受注リストから，事務係が優先度の高い注文を選び，注文番号を入れる。		
		1.	受注書をみる
		3.	品目の特性をみる
		3.	品目の在庫を問合わせる
	受注書，受注品目，各品目に対する在庫の表示		
3.	事務係が各品目に割当てる数量を入れる		
		3.	各品目に対する在庫の更新
		3.	割当てを記録
		3.	この品目に対する注文の特性情報を更新
	割当て終り		

出力文書	チェック済み割当てなし受注リスト	行き先 在庫管理者	
注：在庫管理者は担当の製品のグループとその在庫に対して責任をもつ			

図A 10.　ユーザ・タスク記述書


タスク
入 力 文 書
必要とされる情報
ユ
ー
ザ
イベント
またはサイクル
タ ス ク
共用するため
に蓄えられた
データ
出 力 文 書
蓄えておかなければならない情報
ユーザ・タスク図
デ ー タ 分 析

図A11　　情 報 交 換

れゆえ，タスク（複数）と，共用するため貯えられデータ全セットとの間の情報交換を分析し，それをもとにデータの最適な構造化を図る。この段階での検討は，識別されたタスクすべてに対して行なう必要はない。重要なタスクだけを検討すればよい（とくに問題がない場合，低（使用）頻度のタスクは次の段階では分析されない）。例えば，比較的複雑なサブシステムで 100 個のタスクが識別された場合，（使用）頻度で分けて，重要なタスク 20～25 個に絞ることができる。

(16) ユーザ・タスクのデータ分析

重要なタスクに対するデータ利用度を分析し，データの関係の定量的概念モデルをつくり，全タスクによるその使い方を明確にする。この定量的モデルの表現を「データ関係図」を呼ぶ。

(17) データ関係図

図A 12 に示すように，データは2つのカテゴリに分けられる。すなわち，

- 識別属性：受注や品目など，システムが管理するオブジェクトと，オブジェクトの組合せ（「関係」），例えば，受注と（受注書にかかれている）品目の1つとの間には，「関係（受注品目）」が存在する。
- データ群（楕円形で示されている：オブジェクトもしくはオブジェクト間の関係に結びつけられる基本データの集まり。例えば，データ群「受注特性」は受注に関連があり，「品目在庫」や「標準品目価格」はその品目に関連している。データ群「受注品目特性」は，受注や品目には関連せず，受注と品目の関係，すなわち，関係「受注――品目」に関連している。データ群「受注した品目の補足的な特性」も同様であ

る。データ群「受注推移特性」は，受注がどのように取り扱われたかを追跡するため，受注がファイルに記入されるときつくられる。これも関係「受注――品目」に関連している。「割当て情報」はこの品目の購買スケジュール情報を含んでおり，やはり，「受注――品目」に関連している。

(18) 4 つ の 概 念

これまで，オブジェクト，関係，データ群という3つの概念を導入した。実は，データ関係図に量的情報を記入する場合，着目の仕方によって動的データ関係図と，静的データ関係国の2種類が得られる。この動的データ関係図のためには第4の概念，すなわち，データの，アクセス経路が必要になる。

以上4つの概念を例とともにまとめると次のようになる

オブジェクト：受注，顧客，品目など。

関　　係：オブジェクトの組，受注―品目，倉庫―出庫など。

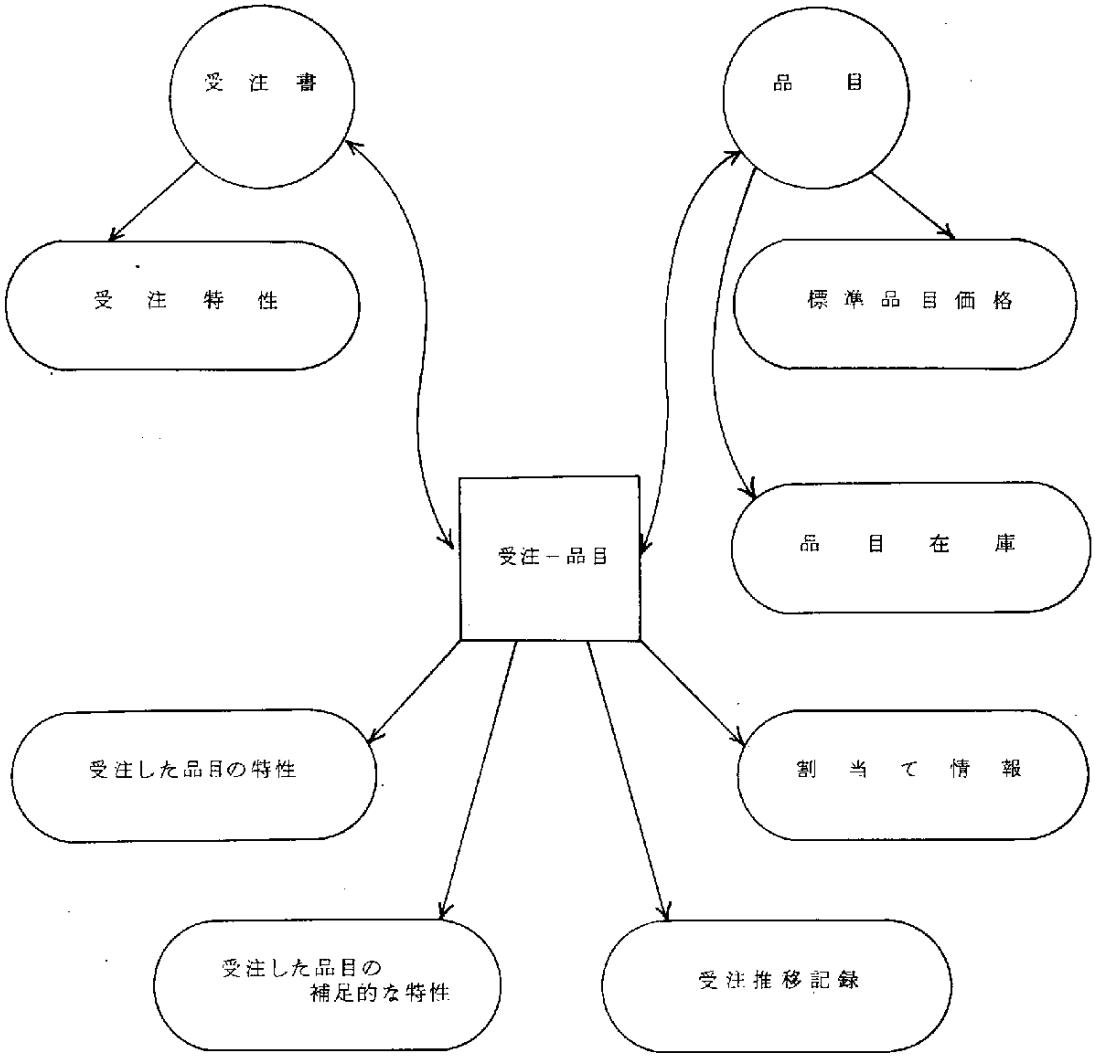
データ群：オブジェクトもしくは関係の属性。名前，住所，品目記述，配送計画など。

アクセス経路：データ群に到達するために通過するオブジェクトと関係の順序列。

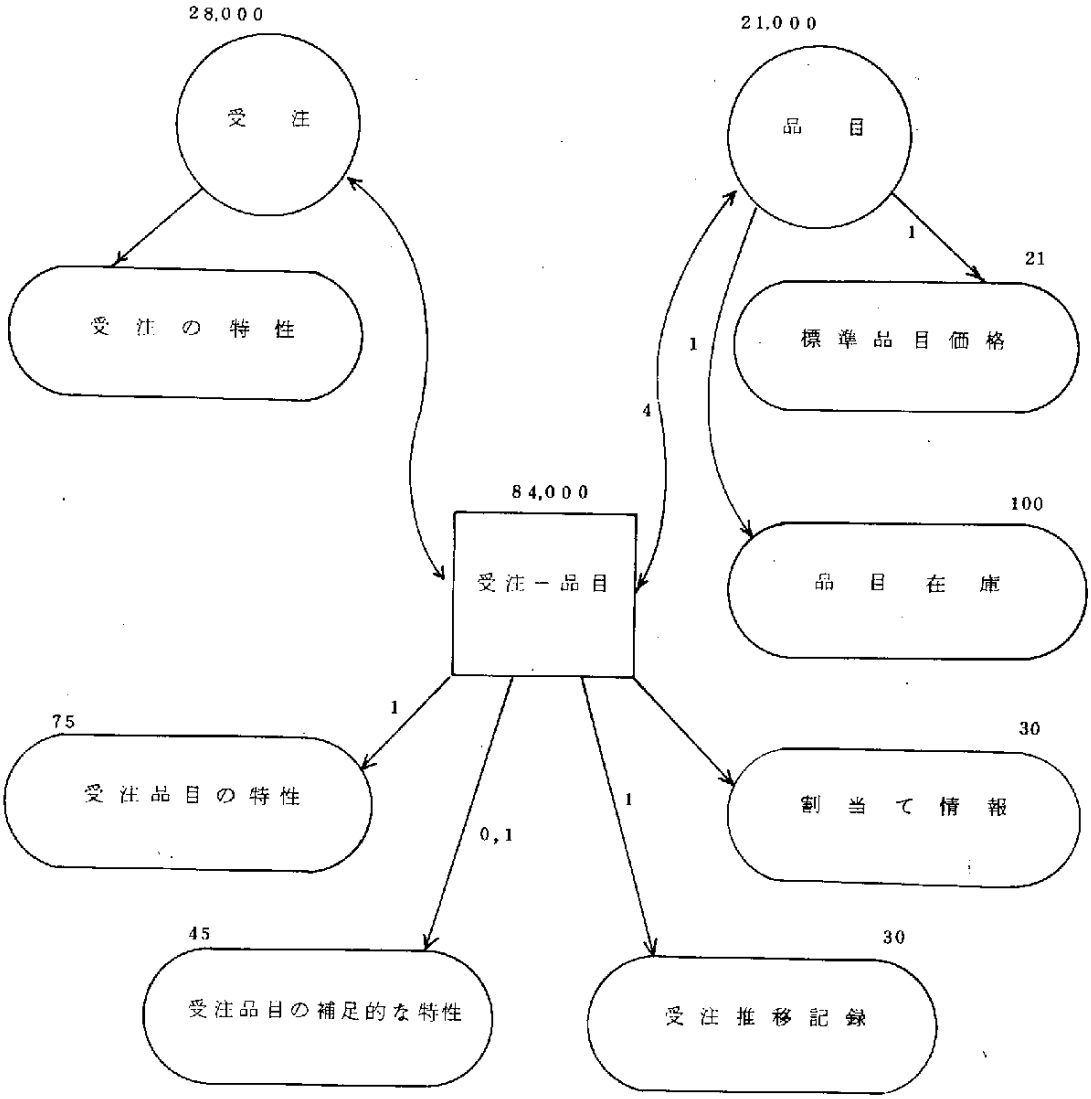
(19) 静的データ関係図

図A 13 は，図A 12 とほとんど同じであるが，タスクで使われるデータの量が記入されており，静的データ関係図といわれる。

21,000 個の品目に関連した平均28,000 件の受注があることが分る。受注1件に平均3品目が含まれ，1品目は4つの受注と関連ずけられている。それゆえ，関係「受注―― 品目」は84,000 個発生している。これらの数は重要で，注意してチェックしなければならない。これらは設計の重要な


受注書
品目
受注特性
標準品目価格
受注－品目
品目在庫
受注した品目の特性
割当て情報
受注した品目の
補足的な特性
受注推移記録

図A 12　　データ関係図



図A 13　　静的データ関係図

キーになる。

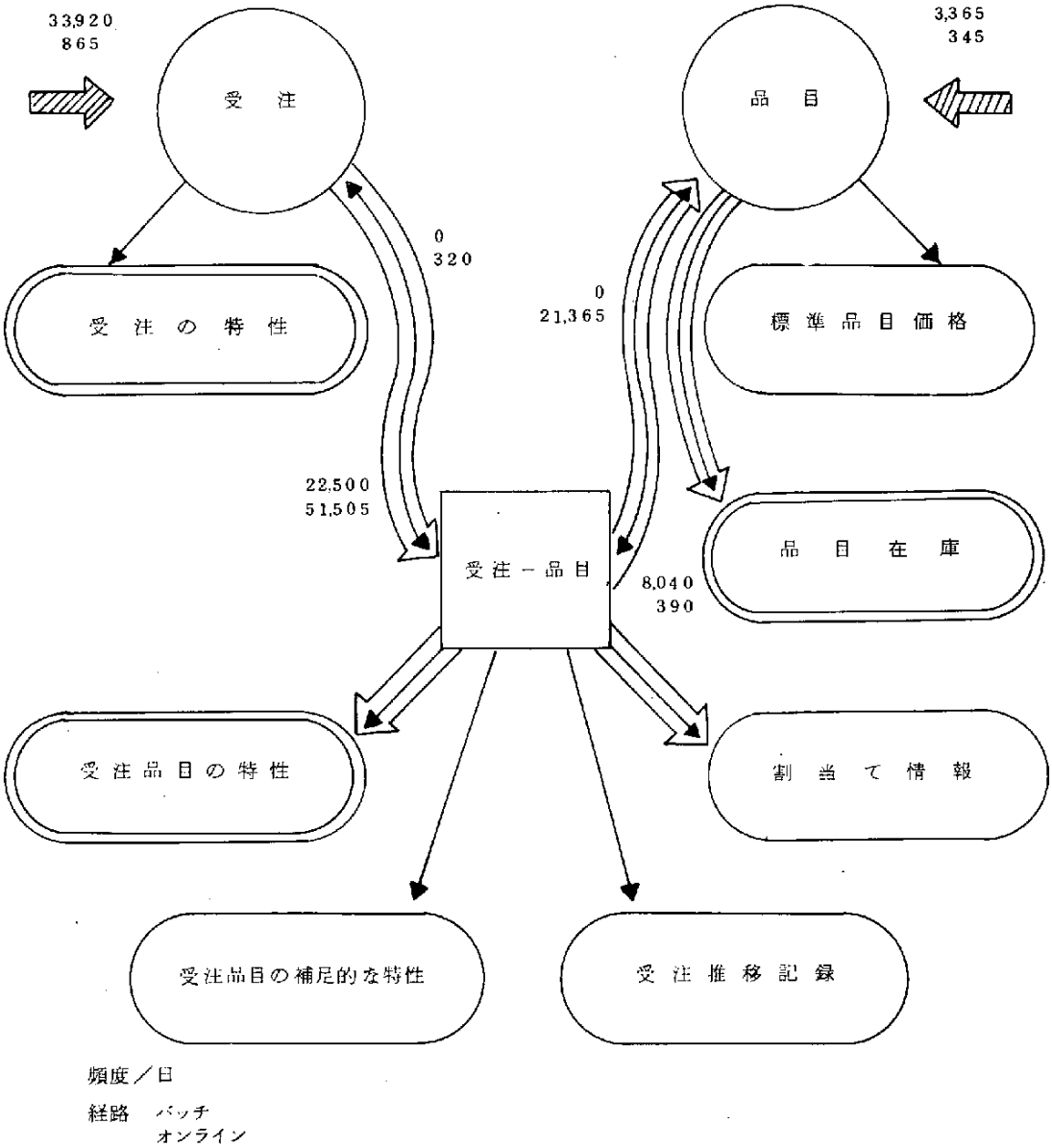
また，図A 13は，各「受注」ごとに「受注特性」データ群があり，その大きさは約35バイトであることを示している。各「受注　品目」の関係に対して，1つの「受注推移記録」データ群と3つの「割当て情報」データ群がある。これは平均的な受注は3個の品目を割り当てていることによって満たされることを意味している。このように，静的データ関係図は，タスクが使うデータ量の評価に使うことができる。

(20) 動的データ関係図

動的データ関係図に記入されている数値は，重要な一連のタスクを実行する際にたどる種々のアクセス経路の，1日の利用頻度を表わしている。

図A 14の動的データ関係図で，上の数値はバッチ利用の，下の数値は端末によるオンライン利用の頻度を表わしている。1日に約35,000件（33,920件＋865件）の受注があり，1日に約74,000回（22,500回＋51,505回）「受注」から「受注── 品目」に行くことがわかる。これらの数値は，すべての重要なタスクによる経路の1日の使用を加えた結果である。

例えば，図A 10にかかれている在庫管理者のタスク「コンピュータの助けによる割当て」を考えてみる。「受注」から入って，必要とする受注について本当に作業しているか検討するため，データ群「受注特性」を入手する。それから，「受注」から「受注──品目」を通って，データ群「受注品目特性」を入手する。そこで，この特定の場合に，送り届ける数量を見出す。「受注── 品目」から「品目」に行き，データ群「品目在庫」中にある在庫水準を知る。在庫が十分な場合，割当てに必要な量だけ取る。そこで，「品目」から「受注── 品目」にもどり，データ群「割当て情報」を作り出し，データ群「受注品目特性」の中の割当て在庫を更新する。こ



図A 14　　動的データ関係図

のオペレーションは受注書の各品目に対して繰り返される。

このように，データ関係図は重要なタスクによるデータの利用を簡潔に表わしている。

(21) ユーザ・タスクごとのデータの分析

タスクごとのデータの分析は図A 15 のような用紙を用いて行なわれる。中心となる情報は，

- タスクで必要なデータ群
- アクセス・モード（読取り，追加，更新，または抑制）
- アクセス経路
- 利用頻度

「まとめの記述」は，データ・アクセスがともなうオペレーションについて，「叙述調」で，上記の内容を説明する。

(22) データの構造化

データ関係図をもとに，データの物理構造をつくる。そのときの基準は，

- データへのアクセスの性能を最適化する
- データの冗長度を最小にする
- 発展性を残し，将来のために，オープン・エンドにしておく

ことである。これらの基準の相対的な重要度は，それぞれのケースで異なるはずである。

(23) 動的データ関係図による構造化

図A 16 は，サブシステム「受注入力」の動的データ関係図である。前の図に，「代理店」，「ケース」，「小包み」が加えられている。「代理店」は「品目」を要求する「受注」を送る。「品目」は「小包み」に荷造りされ，「小包み」は「ケース」に入れられて「代理店」に送られる。このようにしてシステムの働き方が定義される。

UTDA	サブシステム	注　文　管　理		日付　2.11.78
	タ　ス　ク	コンピュータの助けによる割当て		記入者：
所在地：本　社		処理　対話式 タイプ　頻度 540/日	期　間	コード：ORD06
開始イベントあるいは入力文書		在庫管理者の割当て決定 割当なし注文の毎日の状況		発　生　源

データ群	大きさ	アクセス・モード	アクセス経路	利用頻度
受注の特性	36	R	ORD	1
受注品目の特性	75	R	ORDER,ORD.PART	3
品目の在庫	30	R	ORDER,ORD.PART,PART	3
		U	ORDER,ORD.PART,PART	3
割当て情報	30	A	ORDER,ORDER・PART	3
受注品目の特性	75	U	ORDER,ORDER・PART	3

まとめの記述　各注文に対し，注文の特性を読む
　各受注品目に対して　受注品目の特性（数量）を読む
　品目在庫を読む
　割当てにより在庫を更新する
　割当て要素を挿入する
　品目の特性を更新する

出力文書	割当てなし注文リスト，チェック済み	送り先：在庫管理者
入力点　受注	アクセス・モード　Read ＝R　Add ＝A Update ＝U　Suppress ＝S	

図A 15.　ユーザ・タスクごとのデータの分析

この構造化作業は，使おうとしているデータベース管理システムに依存した作業であり，ここではDL／Iを仮定する。

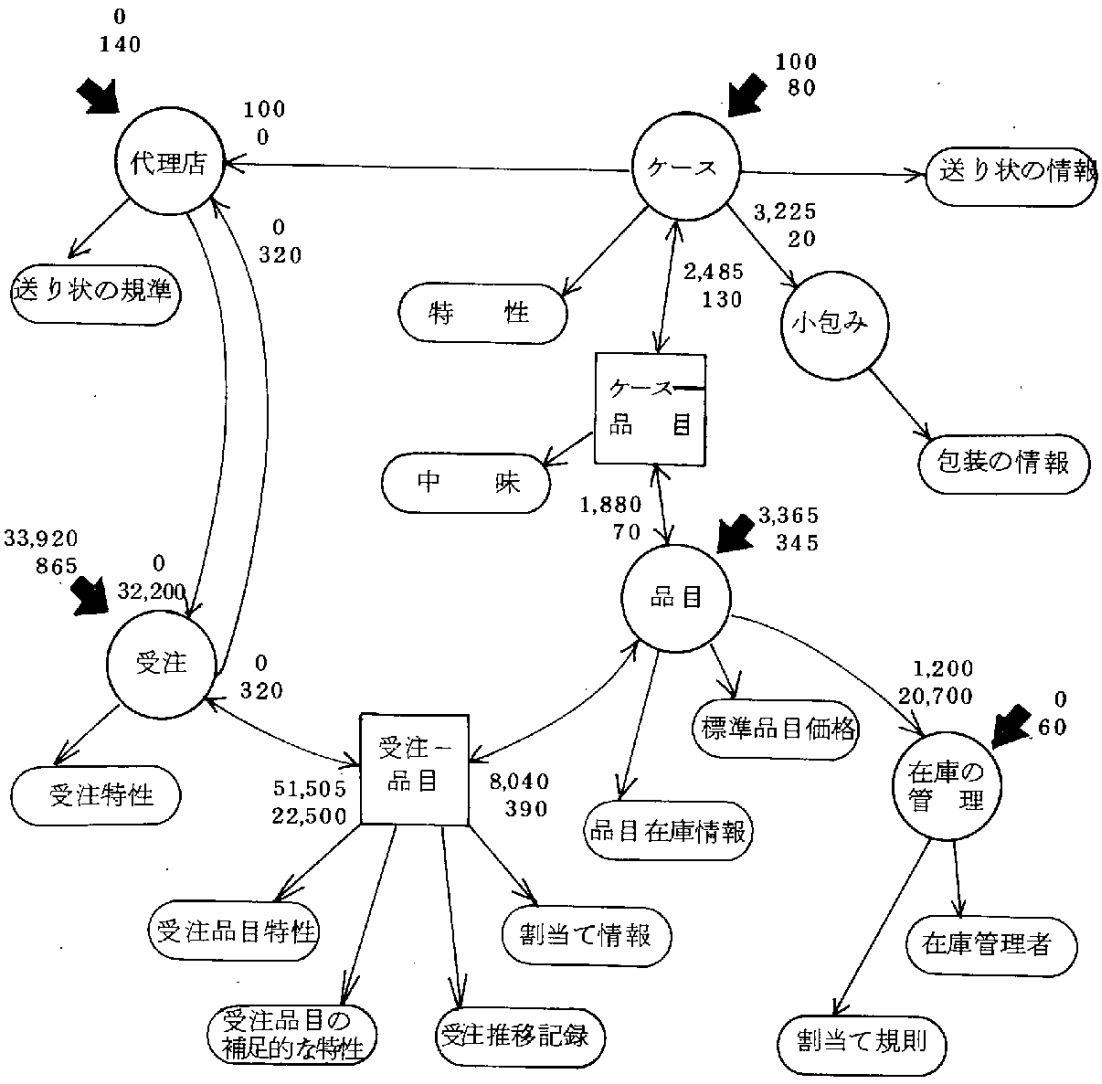
前項の最初の基準「アクセスの性能」から，動的データ関係図内での「統合データ群」を区別することができる。例えば，関係「受注――品目」とこれに関連するデータ群を考えてみる。「受注」から「受注――品目」までの経路は1日に約74,000回使われると仮定され，他方，「品目」から「受注　　品目」の経路が1日に8,400回だけ使われる。アクセスの性能の観点からは，「受注」と「受注――品目」に関連したデータ群をまとめて，物理的に近接して記憶すべきだということになる。同様に，「ケース」，「小包」，関係「ケース――品目」のデータ群へのアクセス経路を調べてみると，「品目」から「ケース――品目」は決して使われないので，データ群は必ず「ケース」からアクセスされる。それゆえ，「ケース」のまわりに，「ケース」「小包」，「ケース――品目」に関連したデータ群から構成された第2のグループをつくる。また，「代理店」，「品目」，「在庫管理者」のまわりにも（第3の）統合データ群をつくる。

「品目」から「在庫管理者」への経路だけが使われるとしても，第2の基準（コントロールされたデータの冗長度）により，「在庫管理者」とそのデータ群は「品目」の統合データ群に含ませるべきではないと判断される。

(24) 中　間　段　階

図A 16では，コントロールされたデータの冗長度という基準はまだ満たされていない。

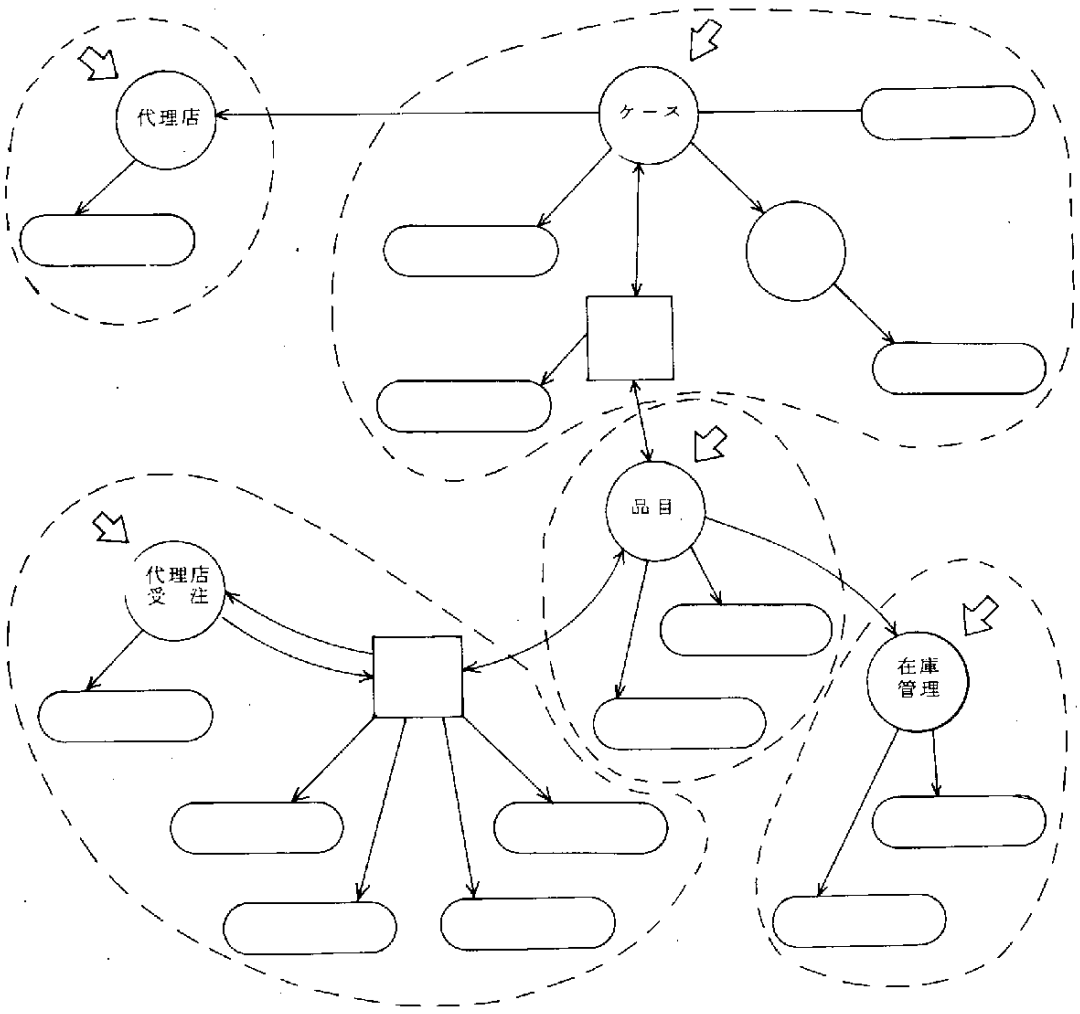
実際には，各統合データ群内でアクセスの性能が最大であれば，タスクの実行は，動的データ関係図に示された頻度によって，1つの群から他の群に与えられた数の動きだけを必要とする。これらの動きは一般に記憶装



頻度／日

経路　バッチ
　　オンライン

図Ａ１６　　構造化のための動的データ関係図の例



図A 17　　データ構造化の中間段階

置へのアクセスを必要とする。これらのアクセスはシステムのパフォーマンス（とコスト）に影響を与える。それゆえ，意図的にデータの冗長度を導入して，できる限りそれらを減らすことを試みる。例えば，「代理店」統合データ群から「受注」統合データ群への動きは，キー「受注」の中にアクセス・キー「代理店」を導入することによってまったく取り除くことができる（図A 17 参照）。

図A 17 では，関係「代理店——受注」は単に「代理店」の「受注」への所有権を表わしているに過ぎない。これは入力点として「受注」を使うすべてのタスクはいかなる変更もなしに，新しい入力点「代理点＊受注」を使ってもよいことを意味する。

(25) DL／I データ構造への展開

図A 17 の5つの統合データ群に対応する5つのDL／I データ構造を図A 18 に示す。

(26) データベース・システム指向の適用業務設計のステップ

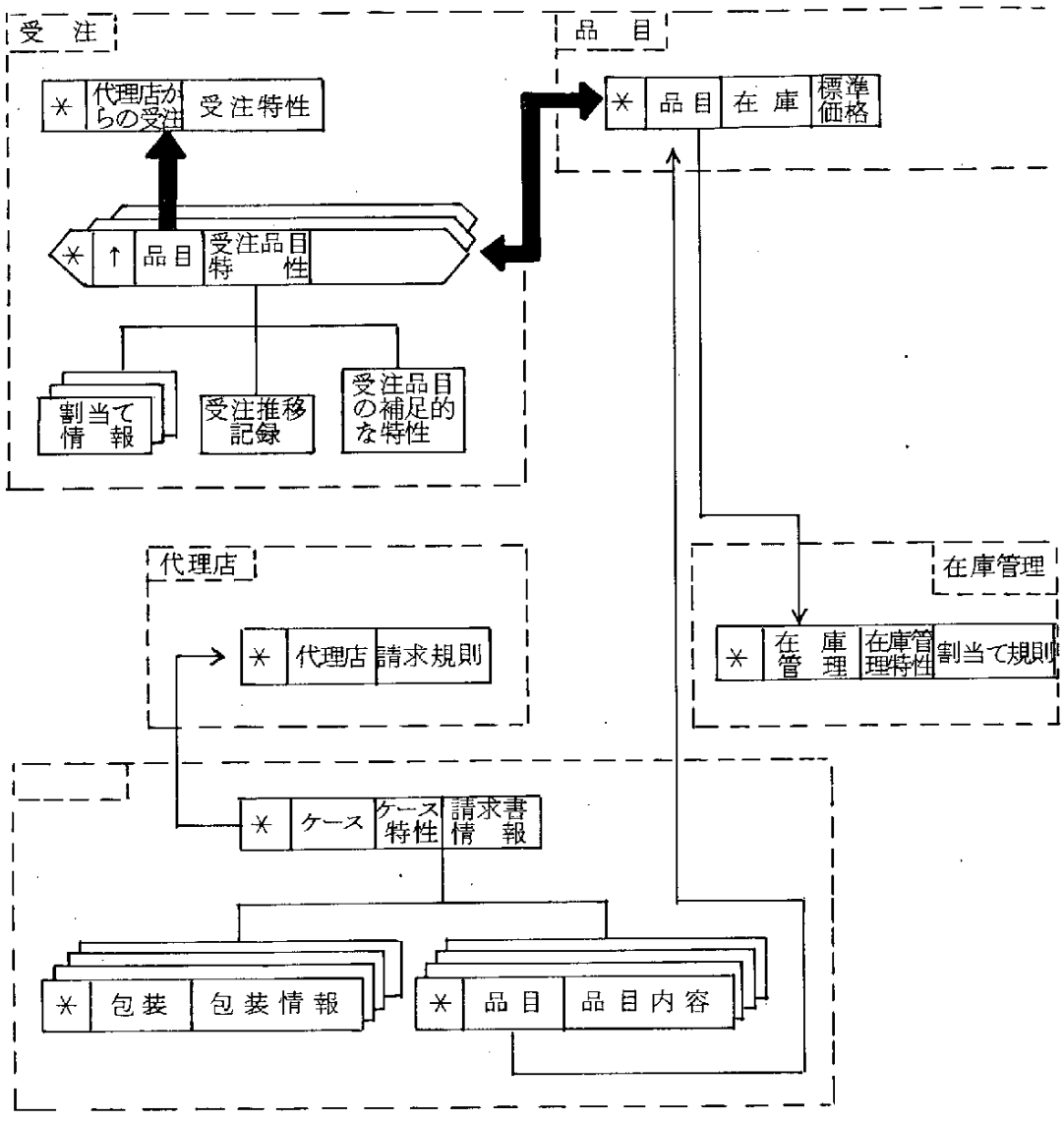
以上説明した方法の流れを要約すると図A 19 のようになる。まず，適用業務の機能仕様にもとずくタスクを定義し，次に，タスクを分類して重要なタスクを識別する。データ収集とデータ分析の段階はデータ関係図で表わされ，最後に，データの構造化の段階でデータベースのDL／I 物理構造を定義する。

以上の結果は，技術的検証によってチェックしなければならない。

(27) 完全なデータベース／データ通信適用業務設計サイクル

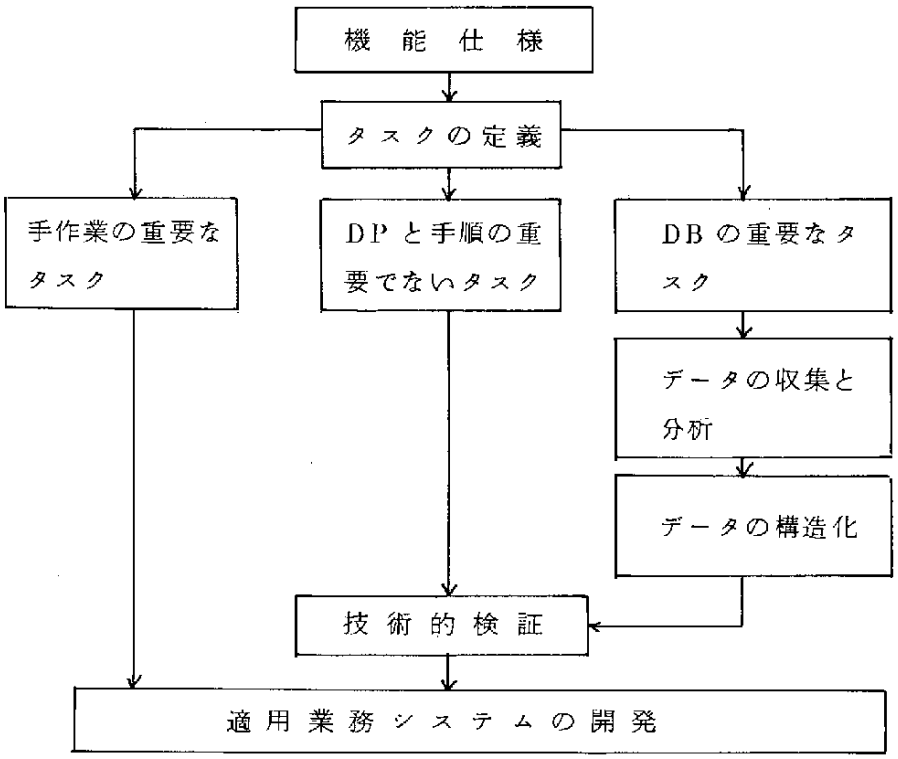
完全なDB／DC 適用業務の設計サイクルを図A 20 に示す。ワーク・ステーション，通信ネットワークの設計など，データ通信に関す部分は左側に書いている。

・ ワーク・ステーションとはユーザとシステムがコミュニケートする場



DL/L管理のリンク ⟶ 適用業務プログラム管理のリンク

図A 18　　DL/Iのデータ構造に展開した結果



図A 19. DBシステム指向の適用業務設計のステップ

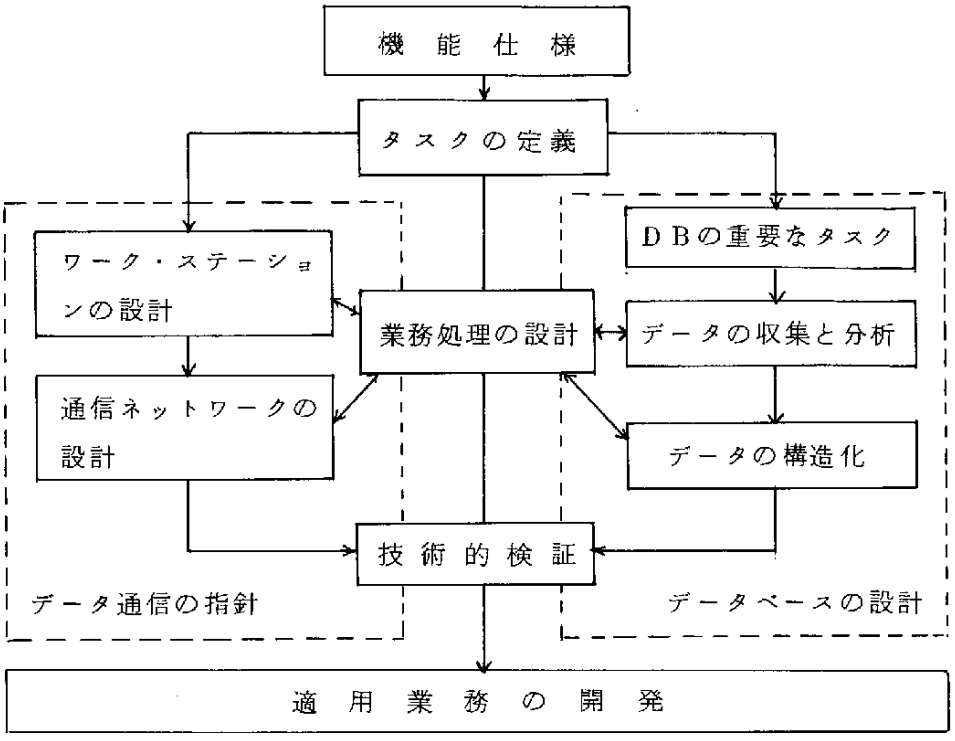
所であり，通常，知能端末，もしくは一般端末を用いてユーザが毎日の作業を行なっている操作環境である。

- 通常ネットワークの検討により，ワーク・ステーションがシステムとどのようにコミュニケートされるかが定義される。
- 処理の部分の設計は，類似の処理をひとまとめにすることによってプログラム開発がより易しくなる。

(28) ワーク・ステーション構成の検討

次のようにしてワーク・ステーションの構成を検討する。

- ユーザの実際の業務活動をユーザ・トランザクションとシステム・トランザクションに分けて適用業務を設計し，ワーク・ステーションを検討する。ユーザとシステムが互いにどのようにコミュニケートするかを



図A 20. 完全なDB／DC適用業務設計サイクル

考える。

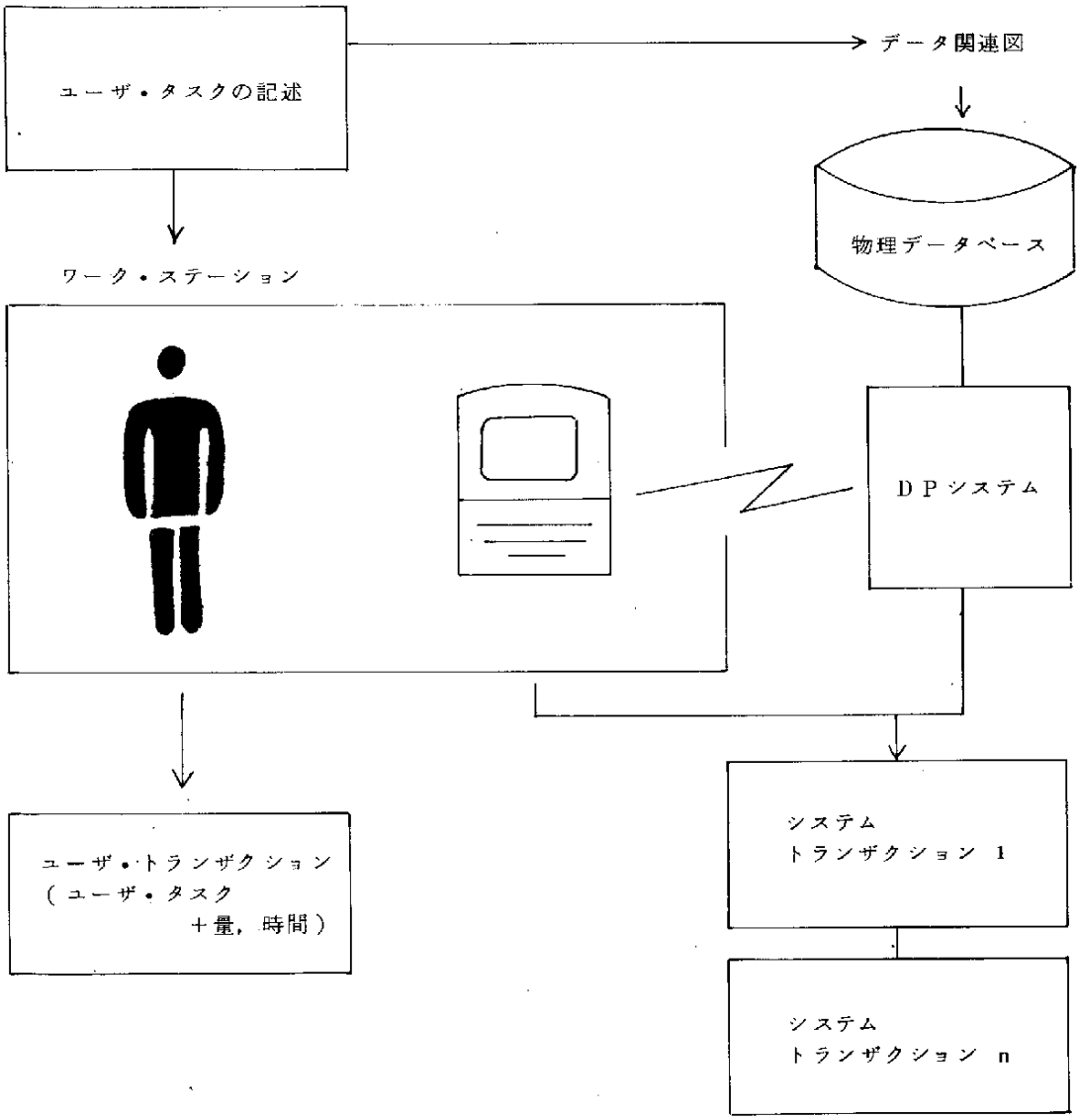
- まだ決定がなされていない場合，どの種類の端末を何台どこに置くかを決める。
- エンド・ユーザに検証してもらう。

(29) ユーザ・トランザクションとシステム・トランザクション

図A 21はワーク・ステーション構成方法のキーになる概念の関係を示している。

タスク（複数）は，オペレーションの集合であり，データ分析を通じてデータ関係図を定義する。タスクはデータをもっとも効率的かつ効果的に利用することができる程度の詳しさで表わす。

上のオペレーションを行なう単位とは，ユーザがシステムとコミュニケ



図A 21　ユーザ・トランザクションとシステム・トランザクション

ートするワーク・ステーションである。ここで言っているコミュニケーションとは「ユーザのトランザクション」であり，ユーザのトランザクションはタスクと対応するが，ユーザのオペレーションのタイミングと量を考慮する必要がある。「システム・トランザクション」とは，ユーザの要求とデータベースの間でデータ処理システムが扱うコミュニケーションである。

データベースは，データ関係図にもとずいて，ユーザ・タスクに従って設計されるべきである。

(30) 適用業務設計の最終ステップ

このステップですべきことは，図A 20 にもあるように，設計結果の技術的検証である。選択したハードウエアとソフトウエアをベースにしてシステムのパフォーマンスを評価する，通常，評価ツールが複雑であればあるほど見積りはより精密になる。

- 選択したハードウエアとソフトウエアの構成のもとに見積るべきことがら

  資源利用度

  システム・トランザクション応答時間

  ユーザ・トランザクション応答時間

  バッチ処理ジョブ時間

- 使うもの

  手計算

  分析のツール

  シミュレーション・ツール

(31) DB／DC 設計作業の結果

最初の原案：

- ワーク・ステーションの設計と配置
- 主要なユーザ・トランザクション
- 主要なバッチ処理ジョブ
- データベース
- システムの物理的構成

見　積　り

- 利　用　率
- トランザクション応答時間
- 処理のタイミング

以上が設計作業の結果である。データ処理システムは，定義され，技術的に検証されたと見なされる。

(23) 最初の設計作業チーム

データ関係図の作成はいかなる技術的知識も必要としない。従って，どのユーザもデータの構造化段階まではその検討に参加できる。

設計チームは

- ユ　ー　ザ（複数）
- データ処理アナリスト（複数）
- システム・アナリスト（複数）
- プロジェクト・リーダ

から構成されるべきである。チームは2～6人で構成され，半分の時間を適用業務設計作業に費やすような例が多い。

(33) 適用業務設計／導入実施時間の例

図A 22は，これまでに述べてきた方法による設計作業時間の例である。実際の時間はそれぞれの状況によって異なる。この例は，ユーザの要件が充分よく理解されている中規模の適用業務の場合と考えてくれればよい。

一般的に言って，この方法による設計作業では，技術的技能をもっている要員は，技術的検証の段階になって，集中的に必要となる。

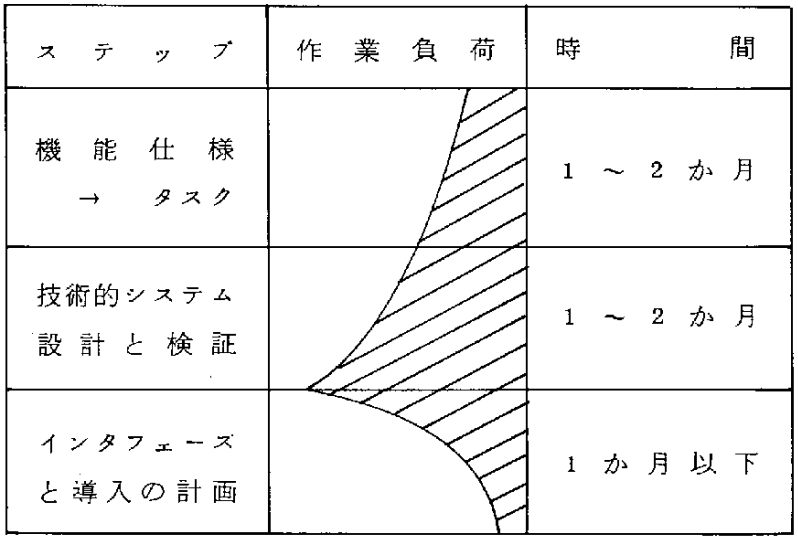
(34) 成 功 の 条 件

第一に，よい機能仕様作業書に記述されたユーザ要件のよき理解

第二に，よく動機ずけされた適用業務をよく理解しているチーム

(35) 適用業務設計のトップダウン・アプローチ

図A 23 のよう設計アプローチになる。このようなアプローチを採用することにより，トップ・マネジメントは，初期の段階において，トップマネジメントからみたビジネス・ニーズ，優先権をはっきり表現（して要求）することができるようになる。

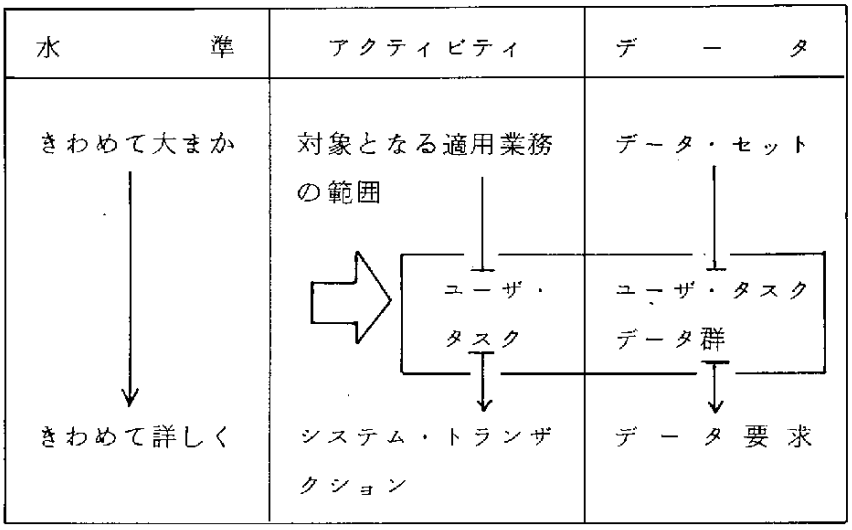


図A 22. 適用業務の設計／導入実施に要する時間の例

(36) 本設計手法を採用する利点

この手法の特長は次の通りである。

- データ関係図の定義まではユーザも理解できる。
- 目に見える結果を出すまでに短時間しかかからない。
- アプリケーション・システムを簡潔に眺めることができる。



図A 23.　適用業務設計のトップダウン・アプローチ

- トップダウン・アプローチと（組み合わせて使っても）一貫性がある。

その結果

- 迅速な導入実施
- 確実な導入
- 一貫性

が約束される。

B．統合データ分析管理システム（IDAMS）

（Integrated Data Analysis and Management：R.Erbe, R.Hartwig，H.Lehman，G.Muller，U.Schauer，Proc.of APL 80 より）

要　　約

典型的なエンド・ユーザ（科学者，エンジニア，経済学者など，プログラミングの能力や意欲はほとんど，あるいはまったくないユーザ）が問題解決のために行なう非定型的なデータ分析では，強力な対話式言語，柔軟性のあるデータベース管理システム，適用業務プログラムの（意のままの）挿入実行，データのグラフ表現機能，プログラムやデータの用法に関する（オンラインでの）情報（提供），データやプログラムの適用業務に合わせた記述(説明）が強く要求される。

IDAMS（Integrated Data Analysis and Management System）は，それらの要求をすべて満たすように，いくつかの構成要素を組み合わせてつくられている。IDAMSは大部分APLで書かれている。IDAMSでは，（データベース・システムで管理される）データと，（APLライブラリ，非APLライブラリ中の）プログラムを，簡単ではあるが強力な照会言語を用いて利用することができる。データとプログラムの属性は辞書中に記述されており，その意味は，対話式ユーザ・ガイダンス（Interactive User Guidance，IUG）の情報ネットワークで記述されている。ユーザ・ガイダンスおよびIDAMSインタフェースのユーザ能力に対する（ゆるやかな，やさしい）適応性は，データ処理の専門家でない人達が，容易に気持よくIDAMSを利用して問題を解決するキーになっている。

1. はじめに

適用業務専門のユーザ（例えば，科学者，エンジニア，経済学者など）は大量のデータを，非定型的，多少実験的に，できれば対話式で分析，評価しなければならないことがよくある。とくに現在，データ処理経験の浅い適用業務志向のエンド・ユーザは，非定型のデータ分析作業を対話式で快適に行なうことは難しい。問題解決を行なうエンド・ユーザが扱う情報は次の4つに大別できる（図B1参照）。

(1) 問題データ

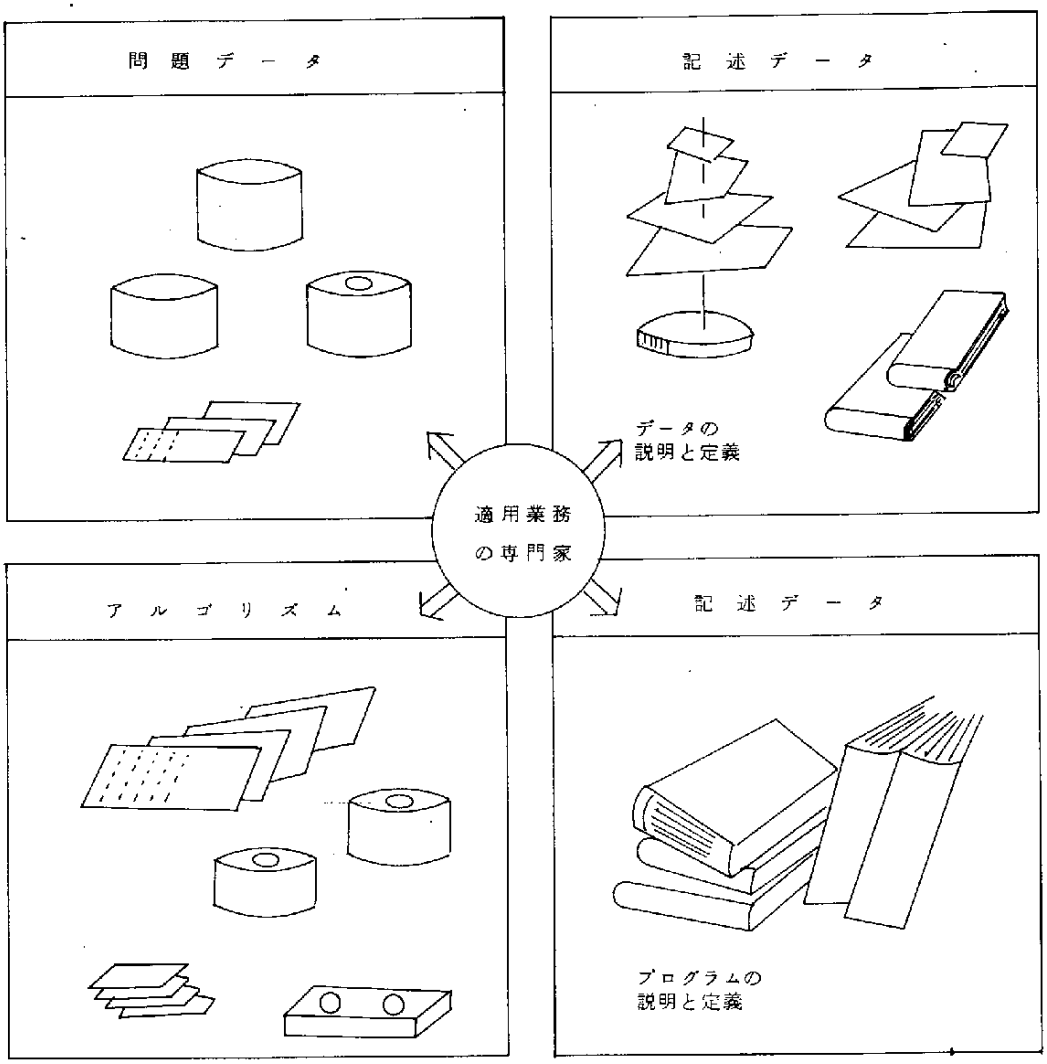
(2) プログラム（手法）

(3) データの定義と説明

(4) プログラムの定義と説明

今日のデータベース・システムやプログラム・パッケージの多くは，データを管理し，計算する強力な機能はもっているが，それらを簡単に組み合わせることができないし，データ処理の経験がかなりなければ使えないことが多い。データ処理経験の浅いユーザに対しては，データ管理機能，データ操作機能を組み合わせて，（インタフェース条件は）同一であるが，ユーザにとって使い勝手のいいインタフェースをもつ（一つの）一様な環境にしてやる必要がある。

初心者やたまたまのユーザにとっては，データやプログラムに関する説明情報が計算機利用の成否を分けるキーである。体系立った説明によって，ユーザが要求する情報を処理し，ユーザにデータやプログラム（の使い方）を案内（ガイド，guide）しながら識別，利用させることを可能にできる。今日の問題解決環境をまとめると次のようになる。すなわち，現在のデータベース，プログラム・ライブラリ，文書化システムは，エンド・ユーザ向きにはつくられていないし，独立したシステムとして開発されているので，そ



図B 1. 統合化されたデータの集合

れらを1つのシステムに統合して，処理を担当する構成要素や情報を与える構成要素をインタフェースさせながら，たやすく使い分けていけるようにすることができない。それゆえ，問題解決環境を支援するためには，データ，プログラム，および記述情報（定義，説明）を取り扱うための（構成要素が）統合化された，エンド・ユーザ向きに誂えられたシステムをつくることが必要である。

2. IDAMSの概略

IDAMSは，対話式のAPL言語の上，周囲につくられた実験システムである。IDAMSでは，関係データベース・システム（現在は単一ユーザ用のXRMシステム，近い将来複数ユーザ用のSystem−Rに移行の予定）を，非手続き的で簡単ではあるが，強力な照会言語（ホスト言語であるAPLのすべての関係中に自由に組み込んで使うことができる）でアクセスできるし，APLプログラムのライブラリ，非APLプログラムのライブラリも使うことができる。IDAMSはAPLのすべてのデータ構造，すなわち，スカラー，ベクトル，行列，3次元以上の配列だけでなく，測定値の単位，すなわち単位のチェックや変換もサポートしている。辞書があって，そこにはプログラム，データの定義が記録されている。それらの使い方や意味は，対話式ユーザ・ガイダンス・システム（Interactive User Guidance System）によって説明される。

IDAMSのユーザは，自分のデータを表の集まり（外観としては関係データベース・モデルのデータベース）として眺める。プログラムの特性は入出力パラメータで与えられ，表の形式で表わされている。記述データ（すなわち，データ，プログラムの定義と説明）は表形式の情報と，表形式でない情報とから成っている。表形式の部分は，基本的には，表（データ）やプログ

ラムを識別するためのものである。記述データの表形式でない部分は，ユーザを所望（検索中）の情報をもつ（ネットワーク）ノードに導く情報ネットワークの形式になっている。

不慣れなユーザや初心者は，初めにIUGS（構成要素）を用いて照会（すなわち，自分自身のデータ分析／操作）を行なって（自分の）処理に必要な表やプログラムを識別し（すなわち，自分自身の作業環境を生成し），次にデータ分析構成要素を用いて実際に処理を行なう照会をセットアップし，実行する（図B 2参照）。

2.1 管理される情報のクラス

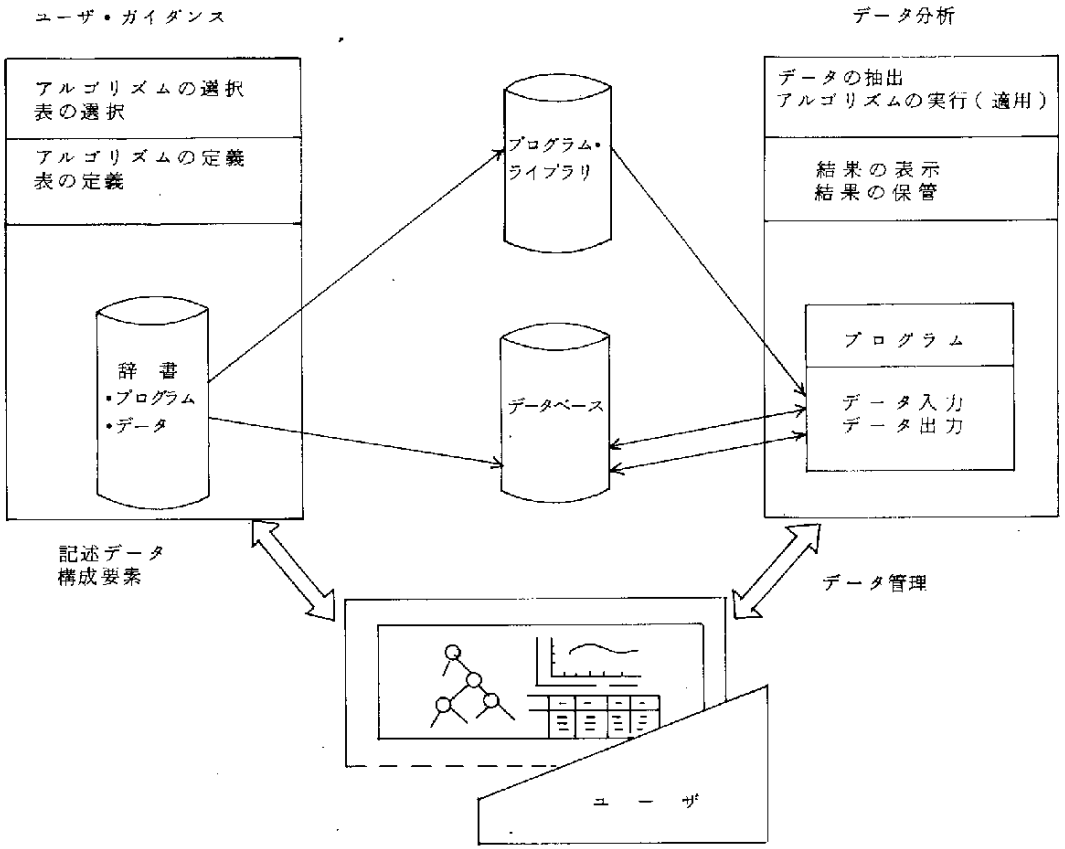
問題解決環境として与えられる情報には3つのタイプがある。それぞれのタイプに応じて，IDAMSは互いに関連する次の3つの（データ）集合を維持しなければならない。

- 適用業務プログラム（群）
- 問題データ
- 記述データ

データベース（問題データ）とプログラム・ライブラリ（適用業務プログラム）に関する説明情報が記述データであり，その点がきちんと（記述データに）反映されるようになっていなければならない（例えば，表やプログラムを追加したり，削除したりする場合，その結果は記述データに正しく反映されなければならない）。

問題向きの表，関数，測定の単位といったIDAMSのオブジェクトを説明している記述データは，次の2つの互いに関連し合の集合に分割することができる。

(1) 表形式の情報：すなわち，照会コンパイル時の形式チェック，関数実


ユーザ・ガイダンス
データ分析
アルゴリズムの選択
表の選択
アルゴリズムの定義
表の定義
プログラム・
ライブラリ
データの抽出
アルゴリズムの実行（適用）
結果の表示
結果の保管
辞　書
・プログラム
・データ
データベース
プログラム
データ入力
データ出力
記述データ
構成要素
データ管理
ユ　ー　ザ

図B 2. 問題解決環境

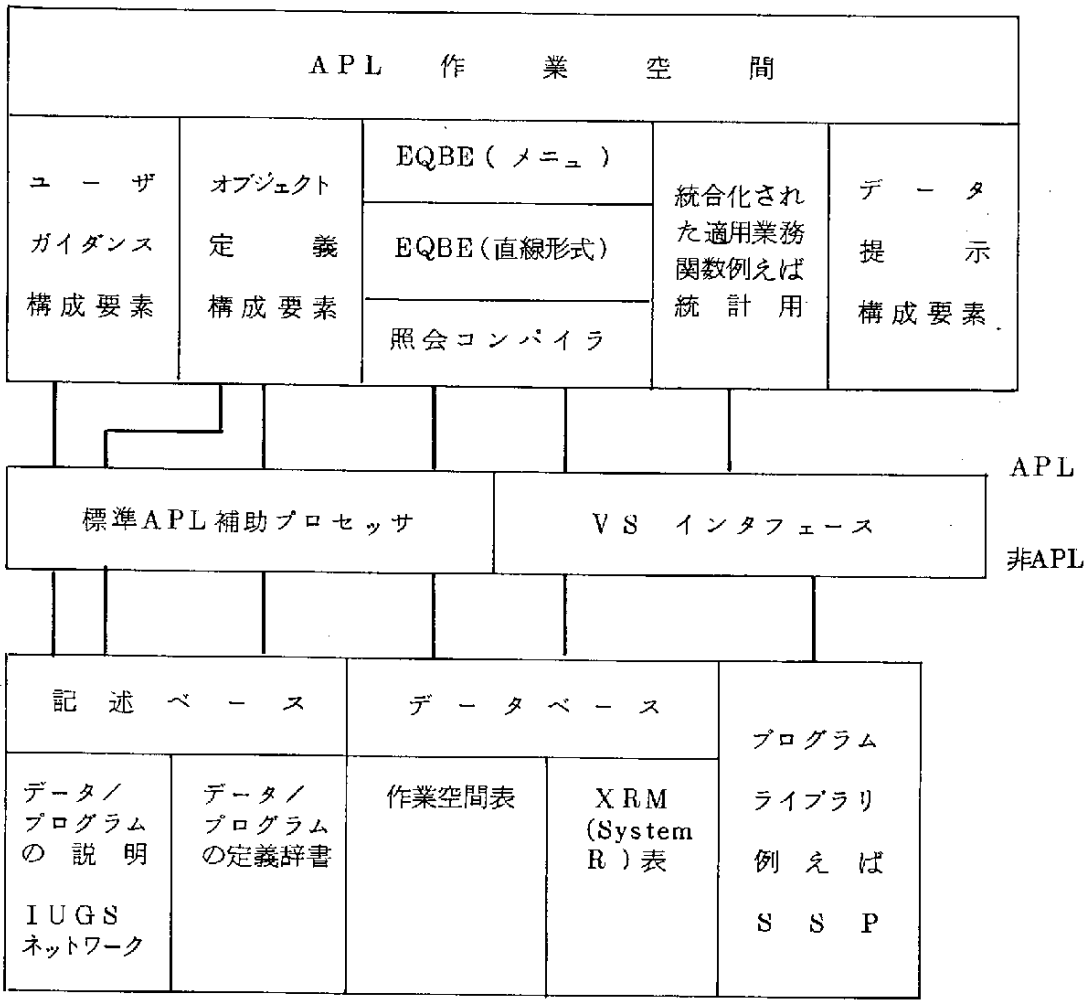
行に必要なオブジェクトの定義である。IDAMSのオブジェクトは，一時的状態，永久的状態のいずれかであることができる。一時（的）オブジェクトは，永久オブジェクトに変えて，いわゆる永久辞書にしまわなければ，セッションの終りで失なわれる。ユーザの作業空間では，一時オブジェクトも永久オブジェクトも取り扱うことができる。それらは，いわゆる活動時辞書中に記入される。活動時辞書中の（記入）項目はセッション中に実際につくられるが，永久オブジェクトは永久辞書からコピーされる。

(2) 表形式でない文書情報：（IDAMS）永久オブジェクトの適用業務環境を記述している情報ネットワークを使用する（ユーザ・）ガイドの過程で，そのオブジェクトの識別をサポートする。この情報ネットワークはIUGSが管理する。辞書中に永久オブジェクトが挿入されたり，（それから）削除されたりすると，それぞれの説明情報もIUGSネットワークに追加されたり，（それから）削除されたりする。

2.2 システム・アーキテクチャ

IDAMSの大部分は対話式APL言語であるVSAPLでかかれている。IDAMSは，IBM VM/370会話モニタ・システムCMS(Conversational Monitor System) の下で動く。図B3はIDAMSシステム・アーキテクチャである。この図は，2つの主要環境が中央インタフェース（標準APL補助プロセッサおよび言語間通信VSインタフェース）によって結合されている様子を表わしている：

- APL環境（APL作業空間）
- 非APL環境（CMSホスト・システム，XRMデータベース，非APLプログラム）



図B 3. システム・アーキテクチャ

APL作業空間は5つの主要ユーザ・インタフェースをもっている。すなわち，ユーザ・ガイダンス，オブジェクト定義，高準位の照会言語，データ提示，統合化された適用業務プログラム・パッケージである。対話式ユーザ・ガイダンスと，オブジェクト定義は，標準APL入出力補助プロセッサを介してAPL作業空間にロードされる記述データ(データ/プログラムの定義と説明)を処理するための高準位インタフェースである。EQ

BE（Extended QBE，拡張QBE，拡張例示照会言語）はデータベースへの照会をセットアップするための高準位照会言語である。照会コンパイラは，これらの照会を，ＶＳインタフェースを通してデータベースにアクセスする，実行可能なＡＰＬ関数に翻訳する。照会処理を通してのデータ取出しに加えて，ユーザは統合化された適用業務プログラムを使うことができる。非ＡＰＬプログラムの場合，ＡＰＬ作業空間との連絡はＶＳインタフェースを通して行なわれる。特別に開発したデータ提示（部）を用いて結果を容易にリスト・レイアウト，グラフ表現することができる。

3. ユーザ・インタフェース

ＩＤＡＭＳには，次の２つの異なる準位（レベル）のユーザ・インタフェースがある。

- 初心者やたまたまのユーザのためのメニュ・レベル（menu level）
- 上級ユーザのためのコマンド・レベル（command level）

メニュ・レベルでは，ユーザは自分の能力に合わせて使うことができる。すなわち，より多くの説明を求めることも，先の選択を指定することによってメニュをとばすことも，コマンド・レベルで，つまりＡＰＬの文で直接使うこともできる。

ＩＤＡＭＳは次の５つの大きなユーザ・インタフェースを提供している：

- 対話式ユーザ・ガイダンス
- オブジェクト定義
- 照会言語
- データ提示
- 統合された適用業務プログラム・パッケージ

対話式ユーザ・ガイダンス（ＩＵＧ）はメニュ・レベルで使う。　オブジェ

クト定義とEQBE照会言語は，メニュ・レベルでもコマンド・レベルでも使える。データ提示と統合された適用業務プログラム・パッケージも，メニュ・レベルとコマンド・レベルで使える。データ提示機能は今なお拡張中である。

3.1 対話式ユーザ・ガイダンス・システム（IUGS）

IDAMSシステムに登録されているプログラムやデータ（の集まり）を使いやすくするために，対話式ユーザ・ガイダンス・システム（IUGS）がユーザ・インタフェースに組み込まれている。IUGSは次の2通りの方法でエンド・ユーザを支援する。

(1) 永久IDAMSオブジェクトに関する説明情報を入力，格納，管理，検索するために使う。

(2) ドキュメンテーション・ツールとして適切に使われていれば，IUGSは，情報ネットワークによってユーザが適切なデータやプログラムを識別することを支援する。それらのデータやプログラムは（情報）ネットワークの末端ノード（グラフ理論でいう「葉」）に（関連づけられ）記憶されている。IUGSは，ユーザを，彼の（適用業務）問題から適切な問題解決方法，プログラム・ライブラリに導いてくれる（図B 4参照）。IUGSは，IDAMSデータベース中に格納されているデータの意味，出所，関係等についての情報を提供する。

IUGSは，適用業務向きのデータやプログラムに関する書式化されないオンライン説明情報を与える（IDAMSの）記述（データ）部分と考えることができる。すなわち，IUGSは，データやプログラムに関する書式化された情報を格納しているIDAMS辞書と（一）対をなすものである。

```
CURRENT NODE:  SEARCH-NODES
SELECT FROM SUCCESSORS
    1 IDAMS-DATA
    2 IDAMS-FUNCTIONS
SELECT:2
CURRENT NODE:  IDAMS-FUNCTIONS
SELECT FROM SUCCESSORS
    1 SSP-STATISTICS
    2 PLOT-FUNCTIONS
    3 GEOMETRIC.FUNCT.
SELECT:1
CURRENT NODE:  SSP-STATISTICS
SELECT FROM SUCCESSORS
    1 SMOOTHING
    2 DISTRIB.COMPARIS
    3 REGRESSIONS
    4 CONTINGENCY-TABL
    5 MULTIV.ANALYSIS
    6 CORRELATIONS
    7 NONPARAMETR.TEST
SELECT:2
CURRENT NODE:  DISTRIB.COMPARIS
SELECT FROM SUCCESSORS
    1 DC.THEOR.EMPIR
    2 DC.EMPIR.EMPIR
SELECT:1
CURRENT NODE:  DC.THEOR.EMPIR
SELECT FROM SUCCESSORS
    1 UNIFORM.DISTR.
    2 EXPONENTIAL.DIST.
    3 NORMAL.DISTR.
    4 CAUCHY.DISTR.
SELECT:3
CURRENT NODE:  NORMAL.DISTR.
SELECT FROM SUCCESSORS
    1 KOLMN
SELECT:1
CURRENT NODE:  KOLMN
RESPECTIVE OBJECT TO BE PROCESSED BY IDAMS  1:YES
```

図B 4　　ユーザ・ガイダンス

```
  1 SSP-STATISTICS 370
  2   SMOOTHING 378
  3     LINEAR CUBIC 376
  4     | SMOOTH 65
  5     E.EXPONENTIAL 377
  6     | EXPON 359
  7   DISTRIB.COMPARIS 380
  8     DC.THEOR.EMPIR 381
  9     | UNIFORM.DISTR. 383
 10     | | KOLMU 344
 11     | EXPONENTIAL.DIST 384
 12     | | KOLME 353
 13     | NORMAL.DISTR. 385
 14     | | KOLMN 355
 15     | CAUCHY.DISTR. 386
 16     | | KOLMC 357
 17     DC.EMPIR.EMPIR 382
 18     | KOLM2 346
 19   REGRESSIONS 387
 20     POLYNOMIAL 388
 21     | REGPOL 358
 22     | REGPOU 348
 23     LINEAR 389
 24     | REGRE 345
 25   CONTINGENCY-TABL 390
 26     CHISQ 339
 27   MULTIV.ANALYSIS 392
 28     EIGENVALUES 393
 29     | EIGEN 340
 30     VARIANCE-ANALYSIS 394
 31     | VARIA 341
 32     FACTOR.ANALYSIS 395
 33     | FACTO 349
 34     DISCRIMINANT.ANA 396
 35     | DISCR 350
 36   CORRELATIONS 56
 37     TWO SETS 139
 38     | CORCAN 352
 39     MORE THAN TWO SE 150
 40     | WTEST 356
 41   NONPARAMETR.TEST 184
 42     UNTIED SAMPLES 185
 43     | WIEST 341
 44     TIED SAMPLES 186
 45     | MP.TWO SAMPLES 187
 46     | | WILCOX 342
 47     | MP.MORE THAN TWO 188
 48     | | FRIED 351
SUPPRESS PRINTING 1:
```

図B 5　IUGSネットワーク

IUGSの基本情報構造はネットワークである（図B5参照）。このネットワークは，特定の適用業務分野の知識を情報要素（ネットワーク・ノード）に分解し，（ネットワーク状に）構造化してつくったものである。

IUGSは次の2つの基本構成要素からなる。

(1) 挿入，更新オプションは，専門家（エキスパート，例えば，適用業務プログラマ，データベース管理者）が，案内情報や，プログラムやデータの体系化（ネットワーク）された説明文書（ドキュメンテーション）をつくるためのものである。

3.3.1. 他のユーザ・インタフェースとのコミュニケーション

永久IDMAS－オブジェクトの管理は，対応するIUGSネットワーク・ノード中に含まれるIDAMSオブジェクト・ドキュメンテーションが行なう。IDAMS辞書の各永久オブジェクトに対して，あらかじめ固定された名前をもつ1つのノードがIUGSネットワークに組み込まれる。

IDAMSオブジェクトを定義し，それを正しく永久辞書に登録すれば，対応するIUGSノードが自動的に定義される。永久IDAMS オブジェクトを削除すれば，対応するIUGSノードも自動的に削除される。

IDAMSオブジェクトを表わすノードには，永久辞書に登録されている書式化された定義を補足する書式化されない（使い方などを示す）情報が入る。ノードへのテキストの入力は対応する辞書情報によって管理される。すなわち，ユーザは，個々の桁，オブジェクト全体に対するテキスト情報を入力するようガイドされる。エンド・ユーザによる識別の必要がある場合は，それをサポートするために，適用業務向きの情報を含むノードを追加することができる。

ガイダンス情報に対応する永久オブジェクト・ノードを結びつけるこ

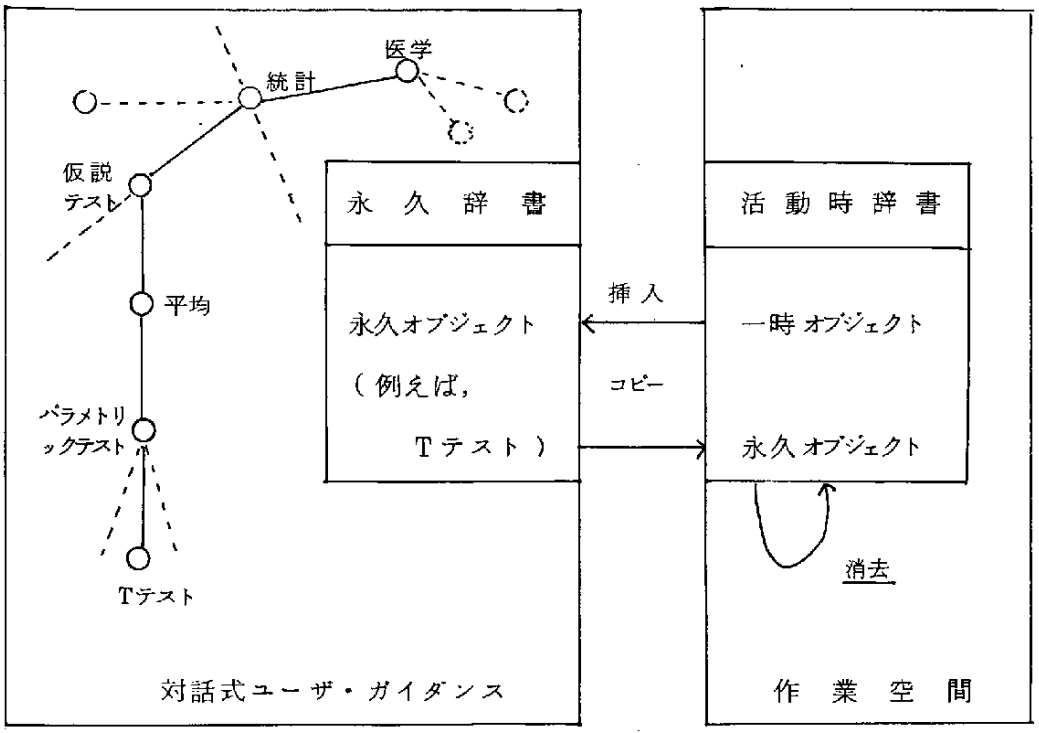
とはもちろん，このガイダンス情報を作成したり，更新したり，削除する場合にも，IUGS挿入更新オプションを使う。それぞれの適用業務のエキスパート（専門家）によって，適用業務向きのガイダンス（情報）ネットワークがつくられ，そのノードと永久オブジェクトの対応がつけられれば，エンド・ユーザはIUGS探索オプションを用いてオブジェクトを識別することができる。

識別されたオブジェクトは，自動的に活動時辞書に入れられる。永久オブジェクトを活動時辞書にロード（load）もしくはコピーすれば，対応するIUGSノード・テキスト活動時辞書にロードされる。すなわち，ノードに対応する説明テキストもIUGS環境の外側，例えば，照会の（条件）設定段階や関数実行時に利用可能になる。オブジェクトを活動時辞書から削除すれば，オブジェクトの対応する説明情報も活動時辞書から削除される（図B 6参照）。

3.2 オブジェクトの定義

表（テーブル），関数，測定の単位のようなオブジェクトはまず定義しなければならない。そのような定義をすべて集めたものが，IDAMS辞書である。使われるオブジェクトはセッション（session）の間活動時辞書中に保持される。端末からのセッションが終っても捨てられない（寿命がつきない）オブジェクトは永久辞書に格納される。

IDAMSでは，データは表として眺めるようなやり方を採用している。IDAMSを使うと予想されるデータ処理の専門家でないエンド・ユーザにとっても，もっともなじみやすい考え方と判断したからである。データは列方向に同じ属性をもつように並べられる。データやプログラムは，QBE（例示照会）スタイルでアクセスするような機構になっている。



図B6　IUGSとデータ辞書

データの属性には次のようなものがある。

(1) IDAMSは任意の階数（rank）の表，例えば，スカラー，ベクトル，行列などをサポートしている。

(2) データには測定の単位，例えば，*kg*，*m*，inchなどをつけることができる。

(3) データの型としては，2進，整数，実数，文字，サンプルされた関数が使える。この「サンプルされた関数（sampled function）という特別なデータ型は，x－開始値と増分，およびy－値とその測定単位の系列（sequence）である。

(4) データ表現法により，データがどのような形で格納されているか，例えば，I2，I4，R4，R8が与えられる。ユーザがデータ表現法を指定しない場合は，システムが自動的に省略時解釈規則を適用して決定する。

(5) データ利用法で表のどの列をキー，もしくは非キーにするかを定義する。

(6) 表のカテゴリにより，APLの作業空間（workspace），XRMデータベースのセグメントのいずれに表を格納するかを決定する。

要約すると，データの定義は，表の名前，列の名前，上記の属性を正しく与えたもののリスティングである。

表の中のデータの変更は，その表の内容の現行の版でしか有効でない。SAVE（もしくはLOAD）コマンドを用いれば，セーブされた版の表の内容を変更できる。この概念は，APL作業空間のセーブ，ロードと似ている。

関数を定義するには，さらに次のような別の属性が必要である。

(1) 関数の言語，例えば，APL，FORTRAN，PL/Iなど。

(2) 関数の型，例えば，関数（function），源泉（source），吸収（sink）など。

(3) パラメータ，例えば，入力，出力。

(4) 単位の写像，すなわち，入力の単位を用いて表わした出力の単位。

(5) パラメータの写像，すなわち，APL変数をFORTRAN，PL/Iプログラム呼出しのパラメータに対応付けること。

源泉とは，通常入力ファイルからデータを読むために使われるAPL関数である。この関数は，呼び出されるたびごとに，1行のデータを表形式による照会に与える。吸収はこの逆で，1回呼び出すと1行分のデータを

対応するAPL関数に渡す。

データの定義や関数の定義は，構文規則にそった文字列として，コマンド・レベルでIDAMS辞書に登録することができる。メニュ選択やメニュ・レベルでの案内によってもIDAMS辞書に登録することができる。

3.3 照会言語（Query Language）

定義された表やプログラムは，非手続き的な拡張QBE言語（EQBE言語）を用いて処理することができる。EQBEは，N.Zloofによってつくられた Query By Example（QBE，例示照会）言語を次の2つの点で拡張したものである。すなわち，EQBEは表だけでなく関数にもアクセスできるようになっており，（照会）条件として任意のAPL述語を指定することができる。EQBEはフル・スクリーン（full screen）で使える。ユーザは，関数や表の枠組（skelton）に直接挿入することができるし，直線形式（linear form）で挿入することもできる。直線形式は照会コンパイラへの入力形式そのものである。上級ユーザは直接直線形式を使うこともある。

メニュー・レベルのEQBEは，APLをほとんど知らない，データ処理の経験素養がまったくないユーザのために設計されたものである。この言語は表（テーブル）による表現形式で，強い記述力をもっている。ユーザは「例示要素（example element）」，すなわち変数やリテラル（literal）を，表示された枠組（skelton）に書き込む。この枠組には，表やプログラムの名前，列やパラメータの名前があらかじめ記入されて，表示される（EQBEは表とプログラムを一様に（区別せず同一の性質のものとして）取り扱う）。さらに，ユーザは述語形式，すなわちAPL表現で照会条件を指定する。この条件は（そのユーザが指定した）例示要素に対

して真値をとるものとして与えられる。EQBEではホスト（host）言語（親言語）であるAPLのすべての機能が使える。通常，別々の（に指定された）述語は「AND条件」で結合される。しかし，別々の述語を選択的なグループ，すなわち，「OR条件」で結合するように指定することもできる。EQBEで指定された照会は実行可能なAPL表現にコンパイルされ，実行される。照会（条件）をみたす結果の値の組は，

(1) 表に記入される（そのあとで照会処理するための準備）

(2) 端末にリストされる（そのあとでAPL処理するための準備）

(3) グラフ表示される（そのあとで表示処理するための準備）

(4) リストされずにAPL変数に代入される

表への挿入(1)は一時的な表に対しても，永久的な表に対しても行なうことができる。一時的な表に挿入した場合は，データはセッションの終りで消滅する。

単純な照会を繰返し，一時的な表を使うことにより，複雑な照会もできる。端末への表示(2)は表の形式で行なわれる。各々の列の内容はAPLの変数に割り当てられる。このAPL変数の名前は列の名前でもユーザが付けた名前でもいい。従って，このAPL変数の名前を用いて，表示された値をさらにAPLで処理することが可能である（図B 7，B 8参照）。

グラフ端末上への表示はデータ提示構成要素がサポートしている（3.4節，図B 9，B 10参照）。

適用業務の関数を照会（条件中）に組み入れることができる（例えば時系列のスムージング，3.5節参照）。

照会セット・アップの際中に（すなわち，枠組に記入中に），選び出した表やプログラム，関連する列やパラメータに関する説明情報がオンラインで使える。

```
 1|APL|EXPRESSIONS    |
--|-0-|-1------------|
1 |0  |T,RAW,ADJUSTED|
2 |⍝  |RAW←R[I]      |
3 |⍝  |ADJUSTED←A[I] |
4 |⍝  |              |

 2|STOCK|SHORT▵NAME|RAW▵RATES|ADJUSTED▵RATES|
--|-0---|-1--------|-2-------|-3-------------|
1 |⍝    |T         |R        |A              |
2 |⍝    |⍝         |⍝        |⍝              |

 4|DATE|RATE▵INDEX|DAY|MONTH|YEAR|
--|-0--|-1--------|-2-|-3---|-4--|
1 |⍝   |-1        |⍝  |D[1] |D[2]|
2 |⍝   |⍝         |⍝  |⍝    |⍝   |
FILLIN:



 STATUS -FILLIN
 1.FILLIN      -CONTINUE FILLIN MODE (EDITING)
 2.CLEAR       -DELETE ALL TABLE-ENTRIES
 3.EXECUTE     -EXECUTE SPECIFIED QUERY
 4.SAVE        -SAVE QUERY FOR REPEATED USAGE
 5.SKELETON    -MODIFY SKELETON SELECTION
SELECT: 3
QUERY NAME: ~T RATES D
```

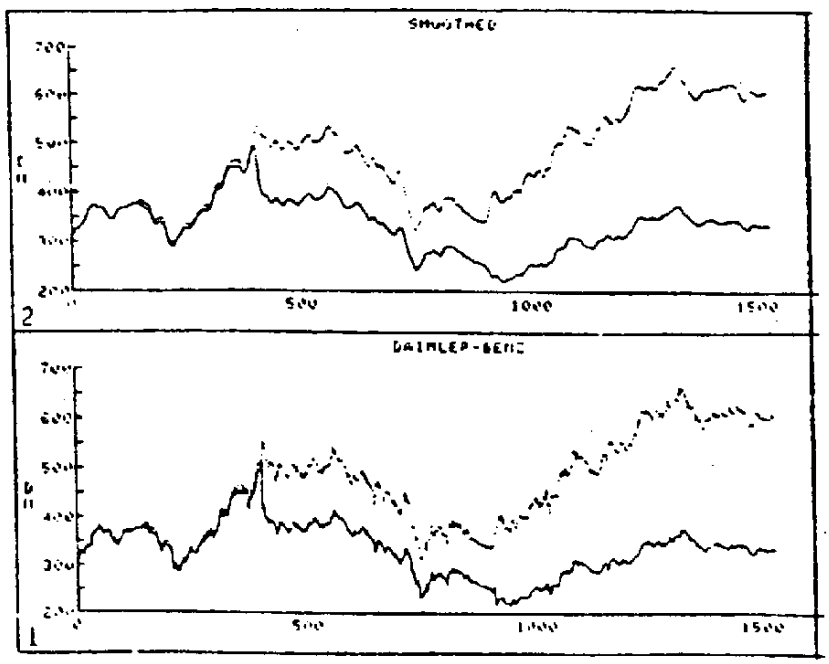
図Ｂ７　　照会入力の例（その１）
（ＡＰＬ変数による入力をふくむ）

```
'DAI'RATES 2 1972

 T          RAW       ADJUSTED

 DAI         318        560
             315        554
             314        553
             314        553
             314        553
             317        558
             316        556
             319        561
             318        560
             315        554
             315        555
             313        551
             312        550
             306        539
             306        539
             309        544
             307        541
             304        536
             305        538
             303        533
             306        539

T/,ADJUSTED-RAW
      242
```

図Ｂ８　　照会出力の例（その１）

```
 1|APL|EXPRESSIONS|
--|-0-|-1---------|
2 |Z  |Z          |

 2|NAME|SHORT|LONG|INDUSTRY&CODE|
--|-0--|-1---|-2--|-3-----------|
1 |Z   |T    |NAME|Z            |
2 |Z   |Z    |Z   |Z            |

 3|STOCK|SHORT&NAME|RAW&RATES|ADJUSTED&RATES|
--|-0---|-1--------|-2-------|-3------------|
1 |Z    |T         |R        |A             |
2 |Z    |Z         |Z        |Z             |

 4|SMOOTH|TYPE|INPUT|-RESULT|
--|-0----|-1--|-2---|-3-----|
1 |Z     |1 20|R    |SR     |
2 |Z     |1 20|A    |SA     |
3 |Z     |Z   |Z    |Z      |

 5|#PLOT|#NUM|#TYP|#GRAF      |#HOR|#VER|#TIT       |#TICS|#LOGA|#SCAL|
--|-0---|-1--|-2--|-3---------|-4--|-5--|-6---------|-7---|-8---|-9---|
1 |Z    |1   |1   |R AND A    |''  |'DM'|NAME       |0    |''   |0    |
2 |Z    |2   |1   |SR AND SA  |''  |'DM'|'SMOOTHED' |0    |''   |0    |
3 |Z    |Z   |Z   |Z          |Z   |Z   |Z          |Z    |Z    |Z    |

FILLIN: EXECUTE SHOW T
```

図Ｂ９　　照会入力の例（その２）
（スムージング，プロッティングを含む）



図Ｂ１０　　照会出力の例（その２）

照会はコンパイルされる。コンパイルされた照会をAPL関数としてセーブすることができる。セーブされた照会をより柔軟にし，繰返し使用できるようにするため，パラメトリック照会の概念を導入している。異なる値を与えることができるプログラムと同じように，パラメトリックの照会では，異なる探索引数を用いて繰返し呼び出すことができる。

直線形式のEQBE言語は上級ユーザ用に設計されたものである。メニュ・レベルのEQBEと直線形式のEQBEの主要な違いは，いわゆる記入段階での使い方の違いである。メニュ・モードではユーザは表示された枠組中に記入するが，直線形式では，照会をセット・アップするための定められた構文規則が適用される。

3.4 データの提示

現在の版のデータ提示構成要素の主要部分は，APL GRAPHPAK プログラム・パッケージにもとずくグラフ機能である。

以下のタイプの1次元プロットがサポートされている。

(1) ライン・プロット（line plot）

(2) バー・チャート（barchart）

(3) ヒストグラム（histogram）

(4) スキャタ・プロット（scatter plot）

(5) スキャタ／ライン・プロット

以下のタイプの2次元プロットもサポートされている。

(1) 透視図

(2) 輪郭線

(3) 3方向投影図

ユーザは1つ以上のフレーム（frame）を同じ画面に表示させることが

できる。1枚の図面（フレーム中に異なる型のいくつかのカーブを重ねることもできる。固定スケーリング，可変スケーリングいずれも可能である。

純粋提示部分に加えて，ズーミング（zooming），値の変更（データベースの更新呼出し），座標変換などのグラフ操作用機能を組み込むことが予定である。さらに，グラフ提示とカーブ・フィティング（curve fitting）を組み合わせたデータ・リダクション（data reduction，カーブに載らないデータを落とすこと）機能も追加することになっている。

グラフ（提示，操作）部分に加えて，報告書作成機能を組み込むつもりである。そうすれば，照会の結果を報告書にし，そのレイアウトを対話式に設計することができるようになる。

3.5 統合化された適用業務プログラム

適用業務の（IDAMSへの）統合化とは，

(1) オブジェクト定義構成要素を用いて入出力パラメータの構文（syntax）を定義する。

(2) 非APLルーチンとのインタフェースとして働くAPLドライバ（APL driver）を生成する。

(3) IUGSを用いて説明情報をつくる。

ことを意味する。

IDAMSでは，適用業務関数は2つの組に分けられる。

(1) 組込み関数。その定義はいわゆる隠し辞書に格納されている。この隠し辞書は常に活動時辞書の一部になっている。この隠し辞書には，グラフ出力を行なう主要プロット関数など共通的に使うと考えられる関数が入っている。

(2) その定義もしくはそのAPLドライバ定義が永久辞書中にあり，実

行に先立って活動時辞書にコピーしなければならない適用業務関数。現在の版のIDAMSには約20の統計用SSP FORTRANサブルーチンが統合化されている。

統合化された適用業務プログラムは，照会の内側で（メニュもしくは直線形式で），照会の外側で入力パラメータのガイドを行なうIUGS実行オプションを通して（メニュ・レベルで），使うことができる。さらに，統合化された適用業務プログラムは，ガイドなしで純粋APLレベルで呼び出すことができる。

4. システムの構成要素

2.2節で概観したように，IDAMSは大きく2つの部分，すなわち，APL作業空間（ユーザ・インタフェースおよび照会コンパイラ）と非APL環境（記述ベース，データベース，非APLプログラム・ライブラリ）に分けることができる。これら2つの環境の間の連絡は，標準のAPL補助プロセッサと特別なVSインタフェースによって行なわれる。

4.1 VSインタフェース

標準のAPL補助プロセッサに加えて，特別に開発した（APL）補助プロセッサの周囲にVSインタフェースをつくり，それをAPL環境と非APL環境の間の主要な橋渡し役にしている。このVSインタフェースを通して，VM/370VSAPLユーザは，APL作業空間にある引数を動的に与え，PL/I，FORTRAM，アセンブリ言語のサブルーチンを実行させることができる。APL引数とサブルーチン・パラメータとの連繋は，APLデータ構造と対応するPL/I，FORTRAN表現をサポートしている構文記述の制御の下で行なわれる。

（APLドライバによる）APLからのFORTRANもしくはPL/I の呼出し（処理）は，次のようなステップに分けられる。

(1) プログラムをロードし，パラメータ情報を生成する（連繋の確立）

(2) 入力引数を割り当てる

(3) 実行する

(4) 出力パラメータを返す。

(5) 連繋を解く

ステップ(2)から(4)は，1回の連繋確立．連繋解除に対し繰返し実行されることもある。単独使用の版のVSインタフェースではステップ(1)から(5)をユーザが指示しなければならないが，IDAMS中ではこれらのすべてのステップが自動的に実行される。

4.2 照会コンパイラ

3節で述べたユーザ・インタフェースに加えて，照会コンパイラもIDAMS・APL作業空間の重要な機能グループである。3.3節で述べたように，ユーザは，表の枠組（EQBEメニュ・レベル），EQBEコマンド・レベルでの構文（直線形式）という2つの方法でデータベース照会を行なうことができる。これらの非手続き的な高水準照会言語のいずれかの形式で書かれた照会セット・アップは，EQBE直線形式に変形され，照会コンパイラに入力される。照会コンパイラはその入力を受けて実行可能なAPL関数を生成する。このAPL関数が，直接作業空間の表にアクセスしたり，XRM（System R）データベースアクセス・ルーチンを呼び出すためのPL/Iドライバを VSインタフェースを通して動かしたりするAPLドライバである。

直線形式の照会は述語の集合からなる。述語を書く順序はどうでも構わ

ない。これらの述語の一部（入力述語）が，出力述語によって処理される「適格化する（qualifying）」値の組の集合を決定する。連繋規則により，適格化する組の集合がきちんと定義されることが保証される。

連繋規則により照会コンパイラが生成するコードの順序が一部決まる。連繋規則によってコードの順序が一意的に決まらない場合は，コンパイラはデータベース・アクセスの回数を最小化しようとする。アクセス経路を決定するために，コンパイラは，表の現在の大きさに関する実際の情報を用い，表の列の中の異なる値の数を見積る。これらの見積りは，表を定義する際，ユーザが指定することができる。

4.3 記述ファイル

ユーザが表，関数，単位を永久IDAMSオブジェクトとして宣言すると，その定義がCMSファイルもしくはXRM（System R）ファイル（上）に格納される。作業空間表（WS表，4.4節参照），関数，単位の定義はCMSファイルに格納される。WS表，関数もしくは単位が活動時辞書にコピーされるとき，その定義が，標準APL入出力補助プロセッサを通して，CMSファイルからAPL作業空間にロードされる。

XRM（System R）表の定義は特別なXRM（System R）ファイルに格納される。永久XRM（System R）表が活動時辞書にコピーされるとき，その定義が，VSインタフェースを通して，XRM（System R）ファイルからAPL作業空間にロードされる。

対話式ユーザ・ガイダンスの情報ネットワークはCMSファイルに格納される。それぞれのネットワークは6つのファイルからなる。IUGSセッション開始時にこれらのファイル（すべて），もしくはその一部が，標準APL入出力補助プロセッサを通して，APL作業空間にロードされる。

4.4 データ・ファイル

データ表は作業空間（workspace table, WS表）として，もしくは，XRMデータベース，System Rデータベース表現のセグメント（XRM一表，System R一表）中に格納することができる。作業空間表としての記憶モードは小さな表に対して使われる。表のすべての列が作業空間に入るからである。作業空間表の利点は，XRM表に比べてアクセス時間が短いことである。APL作業空間のあき空間を越える（サイズの）表に対してはXRM表が使われる。XRMでは，大きい表の組（行）ごとの処理が可能である。

4.5 プログラム・ライブラリ

統合化された適用業務プログラム・ライブラリのメンバーは，永久IDAMSオブジェクトとして保持される。適用業務プログラムは次の2つのカテゴリに分けられている。

(1) 活動時環境を設定するときにAPL作業空間にロードし，直接実行することができるAPL関数

(2) 統合化されている20本のSSP FORTRANサブルーチンのような，VSインタフェースを用い，対応するAPLドライバ（活動時環境を設定するときにロードされる）を通してAPL環境から実行する非APLプログラム，非APLサブルーチンは，テキスト・ライブラリ中に再配置可能目的デック（relocatable object deck）として格納される。

5. 結　　論

1節で議論したように，データ処理に精通していない適用業務の専門家が

データ処理を行なうには多くの困難があるというのが問題解決環境の現状である。問題解決をより快適かつ効率よいものにするため，データベース，プログラム・ライブラリ，案内システム（ガイダンス・システム）にアクセス可能な統合化データ分析管理システム（IDAMS）を提案した。

IDAMSは広い範囲のユーザが使えるのみならず，広い範囲の適用業務に対しても利用可能である。IDAMSは，上級ユーザやデータ処理の専門家だけでなく，プログラマでない適用業務の専門家もサポートするように，ユーザ能力に対する適応性を有している。

IDAMSは実験システムであり，現在ユーザによる評価の際中である。すなわち，IDAMSは，異なる（種々の）適用業務を動かす外部のユーザによる厳しいテストを受けている。IDAMSユーザの評価（公表の予定）は，ユーザの操作を選択的に追跡して自動的に記録したログ・ファイル，および補足的な面接と質問にもとずいて行なわれる。これらユーザ利用の研究結果から，エンド・ユーザ・システム，とくに，意思決定支援システム，対話式データ分析システムへの要求に関するより深い洞察が得られるものと期待している。

6. 謝　　辞

IDAMSの開発には多くの人が貢献した。なかでも，著者等は，VSインタフェースを主として担当してくれたH.Eberle，定義構成要素およびコンパイラのほとんどを作ってくれたH.Schmutzに謝意を表したい。

7. 参 考 文 献

/1/ R.A. Lorie, "XRM an Extended (n-ary) Relational Memory", IBM Technical Report 320-2096, January 1974.

/2/ Donald.D. Chamberlin, "Relational Data-Base Management Systems", Computing Surveys, Vol 8, No 1, March 1976.

/3/ VSAPL for CMS, Terminal Users Guide, IBM Form-No. SH20-9067

/4/ IBM Virtual Machine Facility/370, CMS Users Guide, IBM Form-No. GC20-1819

/5/ R. Erbe, G. Walch, "A General Interactive Guidance for Information Retrieval and Processing Systems", APL76 (ed. G.T. Hunter), pp. 127-140, Ottawa September 1976.

/6/ E.F. Codd, "A Relational Model of Large Shared Data Banks", CACM vol. 13 No. 6, pp. 377-387, June 1970.

/7/ M.M. Zloof, "Query by Example", IBM Research Report RC 4917, July 1974.

/8/ W.H. Niehoff, R.J. Alan, "APL-GRAPHPAK", program description, IBM Endicott Laboratory, 1978.

/9/ IBM Scientific Subroutine Package, programmers manual IBM Form-No. H20-0205, February 1969.

/10/ H. Eberle, H. Schmutz, "Calling PL/I or FORTRAN subroutines dynamically from VSAPL", IBM Scientific Center Technical Report TR77.11.007.

C. McGill 大学対話式計算システム (MUSIC)

（Ulrich Schönbaumsfeld 氏の講義資料より）

(1) MUSICシステム

MUSIC(McGill Uuiversity System for Interactive Computing，McGill 大学対話式計算システム）は，高性能で，コスト効果の高い時分割システム（Time Sharing System，TSS）である。多数のユーザが同時に，問題解決，プログラム開発，ファイル編集，ワード・プロセシング，CAI（計算機援用教育），バッチ処理などの幅広いデータ処理活動を行なうことができる。MUSICはVM/370のもとでも，専用計算システム上でも動かすことができる。ハードウェアに対する要求が柔軟であるため，IBMシステム/370モデル115から303X処理装置までの幅広い範囲のIBMの計算機上で使うことができる。MUSICは4300シリーズのプロセッサ上でも動き，3310，3370ディスクをその本来のモードであるFBA（Fixed Block Architecture）でサポートしている。MUSICはカナダのモントリオール（Montreal）にあるMaGill大学で開発され，ISP（Install User Program）としてIBMから販売されている。現在IBMはRelease 4.1（4.1版）のMUSICシステムをサポートしている。以後の説明はこの版に関するものである。

(2) MUSICシステムの機能

対話式計算システムがその目的をじゅうぶん果すためには，以下の機能をすべて具備していなければならない。

- 端末から対話式ですべての言語が使えなければならない。ユーザは端末の前に座るだけで，バッチに投入したり，印刷出力を受け取ったりしなく

ても，彼がしたいことをすべて端末からすることができなければならない。MUSICではすべての言語が対話式で使える。

- 自分のファイルがどこにあるか，ファイルの大きさはどれくらいか，ファイルにどれくらいの空間を割り当てなければならないかなどについてユーザが心配しなくても済むような，強力なファイル管理システムをもっていなければならない。ユーザは自分のファイルはその名前で参照しさえすればよく，その物理的な位置を心配しなくても済むようにしなければならない。
- 多種多様の多くの適用業務パッケージがその上で動くようでなければならない。MUSICにはいくつかの適用業務パッケージが組み込まれているが，ユーザが選んで適用業務パッケージを追加することもできる。
- システムにアクセスするための端末はもちろん，（他の）装置もサポートされていなければならない。すなわち，（中央の）計算機室にある装置も使えなくてはならない。また，対話式システムは，幅広い範囲の型の端末をすべて同時にサポートできなければならない。
- ユーザは端末の前に座っており，カード・リーダ／カード・パンチにはアクセスしないし，またアクセスする必要もないので，ユーザがシステムに情報を入力する方法がなければならない。これは，時分割環境では編集サブシステムを通して行なわれる。MUSICには3つの異なるエディタ（editor）があり，データをつくったり，変更したりすることができる。
- 計算機システムでサポートしたいのは時分割環境だけではないので，時間割環境が他のすべてのシステムに与える影響は最小限に止められていなければならない。効率いい時分割環境とともに，システムの大きな部分が，バッチや，他のオンライン・システムで使えるようでなければならない。

MUSICはこれらの目標をすべて満たしている。以下，MUSICの各機

能についてもっとくわしくみてみる。

(3) MUSIC対話式言語

MUSICはIBMのプログラム製品FORTRAN GIコンパイラとMod 2ライブラリを対話式でサポートしている。VS BASICプログラム製品をVS BASIC言語ユーザ用に組み込むことができる(オプション)。対話式環境での問題解決には，VSAPLを(MUSIC上に)導入することができる。事務計算適用業務用にはVS COBOLの導入が可能である。PL/I言語を使いたいユーザにはPL/I最適化コンパイラを入れることもできる。IIS(Interactive Instructional System)もMUSICの上で動かすことができる。IISにより端末ユーザにCAI環境が与えられる。VSアセンブラもMUSIC上で使える。

提供されるMUSICシステムには，上記のプログラム製品のためのインタフェースとVSアセンブラしかついていない。上記のプログラム製品を導入するには，IBMに(MUSICと)別々に注文し，しかる後にMUSICに組み込まなければならない。

(4) その他のインタフェース

(3)で述べたプログラム製品の他に，MUSICには以下の製品のためのインタフェースが用意されている。

- GPSS(General Purpose Simulation System)：IBMのプログラム製品である。
- SPSS(Statistical Package for Social Sciences)：Chicago(シカゴ)にある会社SPSSから提供されている。
- WATFIV：カナダのWaterloo大学が開発したFORTRANの高速

（コンパイル）学生用コンパイラで，世界中の多くの大学で広く使われている。

- WATBOL：Woteer大学が開発した学生用COBOLコンパイラである。学生用コンパイラは，コンパイルが非常に速く，エンド・ユーザのために非常に強力なエラー・メッセージを出すという特色をもっている。
- PL/C：Cornell大学で開発された学生用PL/コンパイラである。
- ALGOL−W：Stanford大学で開発された学生用ALGOLコンパイラである。
- PASCAL：Stanford大学で開発され，McGill大学で拡張されたPASCALコンパイラである。
- 3270フル・スクリーン・エディタ：3270型端末を使っているユーザ用の，フルスクリーン・エディタである。

(5) セーブ・ライブラリ・ファイル（Save Libray File）

ユーザがデータをディスク・ファイルに格納しようとするとき，その情報は通常MUSICセーブ・ライブラリ・システムに格納される。このセーブ・ライブラリはMUSIC特有のもので，MUSIC環境の外部，すなわちDOSやOSからはアクセスできない。セーブ・ライブラリ・システムの主たる機能は，（記憶）空間を集中してプールし，（複数の）ユーザ間で共有し合えるようにすることである。あらかじめ決められた量の空間を個々のユーザに割り当てる必要はない。従って，空間の無駄がなくなり，効率的に使うことができる。セーブ・ライブラリ・ファイルには端末からでもバッチからでもアクセスできる。

プールからもらった空間をユーザが解放すると，その空間はシステム(中)の他のユーザが直ちに，自動的に再使用可能である。ユーザがそのファイル

に格納していた情報はシステム（中）の他のどのユーザもアクセスできない。これはMUSICがもつ大きなセキュリティ（security）機能である。

MUSICは通常圧縮形式で動く。すなわち，セーブ・ライブラリに情報を書き込んだり，セーブ・ライブラリから情報を読み出すとき，繰返し現われる文字（列），先行する空白（文字），（文の）後に続く空白（文字）は圧縮され，ファイルから取り除かれる。これは，ファイルの保持に必要な空間の量を減らし，ファイルと（内部）記憶装置との間の情報転送時間を短くする効果をもたらす。

MUSICは多数のユーザ・ファイルをサポートするが，ファイル・アクセスは非常に速い，MUSICシステムでは，1ないし2回のディスク・アクセスで，データ・セット（data set）にアクセスすることが可能である。たとえ50000本のファイルが使われていてもそうである。

MUSICはファイルをネスト（nest）する能力をもっている。すなわち，あるファイルが別のファイルを指し，そのファイルがさらに別のファイルを指すというようなことが可能である。プログラムからこのような（ネストされた）ファイル（群）をアクセスするとき，あたかもそれらは連続した1本のファイルであるかのように見える。8レベルまでのネストが可能である。

MUSICはいくつかのレベルの保護をかけてファイルを格納することができる。通常，省略時解釈により，ファイルは「プライベート（private，私用）」で格納される。従って，ファイルを生成したユーザがそのファイルにアクセスできない。ファイルを生成するユーザはそのファイルを「パブリック（public，公用）」として格納することができる。この場合，そのファイルの名前を知っている（システム中の）ユーザはそのファイルを読むことができる。しかし，それを書き換えたり，スクラッチ（scratch）にすることはできない。さらに，ファイルを「実行専用（execute-only）」と

して格納することもできる。この場合，ユーザはそのファイル中のプログラムを実行することができる。しかし，プログラムをリストしたり変更することは絶対にできない。MUSICには，ファイルを「付加専用（append—only）」として指定できるという独特のオプションもある。このモードの場合，ユーザはファイルの終りに書き加えることはできるが，ファイルの内容をみたり，変更することはできない。

ユーザが持てる空間の量を制御できるようにするため，システム上の個々のファイルの最大サイズ，ユーザがセーブライブラリ上に格納することができる空間の総量を指定するユーザには，それぞれの配分空間が与えられる。

一日のうちに生成されたり，変更されたりしたファイルには印がつけられる。従って，変更されたファイルすべてをバック・アップ・テープにとるMUSICバッチ・ジョブを夜走らせることができる。さらに，セーブ・ライブラリ中のすべてのファイルをある一定の期間内バック・アップ・テープにとるように，そのサイクルを設定することができる。この一定の期間は，通常30日である。ユーザが最後のバック・アップ・サイクルからあとファイルを変更するしないにかかわらず，そのユーザのすべてのファイルがバック・アップ・テープにとられる。

(6) ユーザ・データ・セット（User Data Set）

セーブ・ライブラリの他に，ユーザ・データ・セット・ボリューム（User Data Set Volume）を使うことができる。これは，セーブ・ライブラリに格納するには大きすぎるプログラム・ファイルやデータ・ファイルを格納するためのものである。

順次アクセス・データ・セットと直接アクセス・データ・セットがサポートされている。ユーザ・データ・セット・ボリュームには，設置個所による

一意的なボリューム名が割り当てられる。端末ジョブは常時装填されているボリューム（IPLの時点で装填）を使うことができ，バッチ・ジョブは，非・常持装填ボリューム（システムがオペレータに装填メッセージを出す）を使うことができる。

ユーザ・データ・セット・ボリュームはOSで読み，書きできる（512バイト以下の指定されたサイズの物理レコード，論理レコードのサイズもそれに合わせて適当に決める）。

ユーザ・データ・セットの属性は，データ・セット名の一部としてセキュリティ・コードを介して割り当てられる。

パブリック：だれもが読み，書きできる

プライベート：所有者しか読み，書きできない

読出し専用：だれもが読むことができるが，所有者しか書き込めない。

セーブ・ライブラリと同様，ユーザ・データ・セットにもユーザは空間配分を要求することができる。ユーザ・データ・セットの最大サイズは，各ユーザ・コードごとにMUSICシステムを用いて指定することができる。

(7) サポートされている端末装置

MUSICシステムがサポートしている端末の中にはバッチ端末も含まれている。この場合，バッチ端末とは，計算機室に置かれているカード・リーダ，ライン・プリンタ，カード・パンチのようなものをさす。3270系列の端末（3277，3278）が遠隔装置としてサポートされている。2741モードで動作しているIBM3767，IBM2741，IBM通信磁気カードシステム（Communicating Mag Card System, CMCST），および新しく発表されたIBM 3101端末はすべてMUSIC環境で動く。また，テレタイプのモデ

ル33，35，38，43もサポートしている。さらにテレタイプと同じインタフェースをもつTTYもサポートしている。このカテゴリに入るTTYには，Hazeltine，DECWRITER，その他多くの端末製造業者から出されている端末がある。

(8) 端末制御機構

端末からの制御可能度，利用度が大きいことも，MUSICシステムの特色の1つである。これは，MUSICシステムも主要機構の1つである，物理端末特性表の効果である。MUSIC環境中で動く種々の端末の特徴情報をシステム・プログラマがこの表に記入する。システム・プログラマは，端末のバックスペース（backspace）文字，ライン・フィード（line feed），空き文字のような項目を（どのようにするか）選択することができる。それゆえ，新しい，異なる（タイプの）端末をMUSIC環境に導入するときでも，物理端末特性表に（選択結果）を記入するだけで，その端末装置をサポートできるようになる。MUSICでは，ユーザが選んだタブ（tab）の桁に合わせて入出力時のタブ操作が可能である。MUSICの独特の機能は，ユーザがブレーク・キー（break key）を押してもジョブがキャンセルされないことである。その代り，MUSICはブレーク・モードに移り，応えるようにつくられている。ブレーク・モードからいくつかのコマンドを出すことができる。その1つに，スキップ（skip）（・コマンド）がある。端末に出力が表示されており，その次の200行の出力は使いたくないと考えたとき，ブレーク・キーを押し，続いて／SKIP 200とキー・インすればよい。その結果，MUSICシステムは出力を200行飛ばし，（その次の行から）再び印刷を開始する。

ブレーク・モードでのコマンドとしてCOMPRESSがある。このコマン

ドは，１つの行の中の連続する空白文字を１つにまとめるはたらきをする。１行132文字の端末用に設計された出力を１行80文字しか印刷できない端末に出力するときなど，このコマンドが役に立つ。

もう1つ，WINDOWコマンドがある。ブレーク・キーを押してWINDOWとキー・インすると，出力のうちのいろいろな桁（だけ）を（取り出して）覗くことができる。例えば，WIDOW50　100とすると，第50桁から第100桁までの桁だけが端末に表示される。さらに，これらのコマンドはすべて，出力を生成中端末から動的に与えることができ，何回でも必要な回数だけ変更することができる。その時点までにこのジョブが費やした時間を表示させるコマンドもある。自分のジョブがループに陥っているか否か調べることができる。すなわち，TIMEコマンドを使って現時刻と，このコマンドを発した時までにそのジョブが費やしたCPU時間を知ることができる。自分のジョブがループしていると判断した場合は，／CANCELとキー・インすることにより，システムにジョブのキャンセルをしてもらうことができる。ブレーク・キーを押したが，上記の５つのどの（ブレーク・モード・）コマンドも与えたくないときは，リターン・キーを押しさえすればよい。たとえブレーク・モードにあるときでも，ジョブの実行は継続しているということに注意する。

(9) エ　デ　ィ　タ

MUSICシステムには3つの異なるエディタ・サブシステムがある。行エディタ（line by line editor）は1度に1行全体を置きかえる。VS BASIC行番号エディタは，（完全）MUSIC文脈エディタ（MUSIC Context Editor）のサブセットである。VS BASIC行番号エディタは，

MUSIC環境にあるVS BASICを使うユーザのためにとくにつくられたものである。VS BASIC（行番号）エディタは，BASIC言語の文番号で操作するようになっている。文番号を用いて（指定し）BASICの文をリストし，変更し，削除し，入力する。

MUSICの文脈エディタは，汎用の会話型テキスト取扱いプログラムである。その主要機能の1つは，文字列の取扱いである。セーブ・ライブラリ中のファイルに対しても，ユーザ・データーファイルに対しても使える。

この文字列取扱い機能により，1つの行の中間にある文字の置換や削除，プログラムやファイルのすべての文字の置換や削除，文字列の有無にもとずくファイル中の行の表示などの操作を行なうことが可能である。

(10) 文脈エディタ

文脈エディタでは，現在行ポインタ（current line pointer）の概念を使っている。現在行ポインタは，ファイルの中の（ある）特定の場所を指す。ユーザはコマンドを用いて現在行ポインタを必要なだけ上下に動かすことができるし，特定の文字列を指すようにすることもできる。文字列XYZの位置に合わせるということは，ファイル中の文字列XYZの位置を指すように現在行ポインタを動かすことである。その文字列の位置に合わせたとする。例えばその文字列をXYZからYZXに変えたとする。このときはCHANGEコマンドを使う。通常の（位置合わせ（locate），変更（change），削除（delete））コマンドの他に，CHANGE LOGICAL，DELETE LOGICAL LOCATE LOGICALコマンドが使える。これらのコマンドによって次のようなことを指定することができる。

「同じ行に文字列ABCもある行の文字列XYZに位置合わせする」

これは非常に強力な探索コマンドである。論理関数としては，and，or，

not，exclusive-or が使える。MUSIC文脈エディタを16進形式で動かすこともできる。端末から16進形式で入出力できることもその1つである。MUSICシステムでファイルを変更するときは必ず日付がつけられる。従ってそのファイルを最後に書いた日が分る。このエディタの中から，編集中のファイルに別のファイルを組み込んだり，付け加えたり，することができる。ファイルAを編集中，その点でファイルBを組み込みたいときは，ファイルのMERGEコマンドを出せばよい。エディタのコマンドを忘れたり，コマンドの使い方に自信がないときは，エディタ中でいつでもヘルプ（help）機能を使うことができる。この機能を使いたいときはHELPとタイプすればよい。その点で質問のあったコマンドに関して求めている情報を得ることができる。ヘルプ機能を使い終ると，それまで編集していた点に正確にもどってくれる。MUSICは高速入力モードの能力ももっている。システムにデータを入力しているとき，MUSICはタイプ可能な（最大の）速さで情報を収集する。この機能は端末からの入力の受け入れに必要なオーバヘッドを少なくする。ユーザは，自分のプログラムから直接この高速入力モードを呼び出すこともできる。

(11) コマンド言語

MUSICシステムの内側ではJCL（Job Control Language，ジョブ制御言語）は使わない。コマンド言語を用いる。この言語はプログラマでない人にも分りやすい言語である。会話型で英語的である。厖大な数のコマンドが使えるが，平均的なユーザならほんの僅かのコマンドが使えるだけでよい。このシステムは推論能力をもっている。すなわち，コマンドを省略形で与えることができるし，分らないコマンドはプログラム（セーブ・ライブラリ・ファイル）の名前と考える。その名前をもつプログラムがあるかどうか

セーブ・ライブラリをさがし，そのプログラムをロードし，実行する。省略形はオプションで，コマンドは１文字にまで簡単化できる。システムがその文字列を一意的であると認識できる限り，コマンドとして認識し実行する。

キー・ワードはないし，パラメータもほとんど必要ない。バッチ処理でも同じコマンドが使えるので，カード・リーダからジョブを投入するときでも，ジョブ制御言語を使う必要はない。従って，VM370のもとで走っているときサブミット（submit）コマンドが使いやすくなる。また，端末ユーザは，自分の端末からMUSICのバッチ・システムにジョブをサブミットすることができる。

(12) いくつかのコマンド

ユーザが端末セッションで使ういくつかのコマンドを図Ｃ１に示す。これらのコマンドは全体のごく一部に過ぎないが，これだけのコマンドがあれば通常のMUSICの機能は大抵動かすことができる。

直接コマンドは，MUSICが端末からそのコマンドを受け取ると直ちに実行するコマンドであり，遅延コマンドは，環境を設定するためのコマンドである。

(13) 端末セッションの例

図Ｃ２は端末セッションの例である。

「＊CODE……」は，「／ID」で入力されたユーザ・コードで（それまでのうち）最後に使われたときの時間と日付である。「＊GO」が表示されているとその端末はMUSICコマンド・モードにある。「EDIT FORT. TEST NEW」で，編集モードに入り，FORT. TESTという名前の新しいファイルを生成することを指示している。このファイルはFORTRANプログ

直　接　コ　マ　ン　ド

LIBRARY	セーブされたユーザのファイルの名前をリストする
EDIT	文脈エディタを呼び出す
/ID	サイン・オン
BASIC	BASICエディタを呼び出す
LIST	ファイル（の内容）を端末に表示する
EXEC	プログラムを実行する
OFF	サイン・オフ

遅　延　コ　マ　ン　ド

FILE	ファイルの名前と属性を定義する
INCLUDE	ユーザ・ファイルを連結する
LOAD	コンパイラを表わす
SYS	ジョブに対するシステムの要求を指定する

文脈エディタのコマンド

CHANGE	ユーザ・ファイル中の文字列を置き換える
DELETE	ファイルから何行か削除する
LOCATE	ファイル中の文字列を見つける
INPUT	ユーザ・ファイルに1行読み込む
SAVE	ユーザ・ファイルをライブラリにセーブする
NEXT/UP	現在行ポインタを動かす
QUIT	ファイルをセーブしないまま編集を打切る
PRINT	端末にファイル（の内容）を表示する

図C1　コ　マ　ン　ド　の　例

```
*MUSIC  --  SIGN ON.
/ID XXXX
*PASSWORD?
MMMMMM
*CODE LAST USED 17:26 11FEB80
*SIGN ON MON FEB 12, 1980  TIME=13:23  PORT=OBO  RESTART=013
*GO
EDIT FORT.TEST NEW
/LOAD FORTG1
      WRITE(6,1000)
1000  FORMAT(///' SQUARE PROGRAM'/)
100   READ(9,*) VALUE
      SQUARE = VALUE ** 2
      WRITE6,1010) VALUE, SQUARE
1010  FORMAT(' THE SQUARE OF',F9.2,' IS',F11.1)
      GO TO 1000
      END
      (CR)
EDIT
SAVE
FORT.TEST
NEW FILE SAVED
*GO
```

図Ｃ２　（その１）　端末セッションの例

```
EXEC FORT,TEST
*IN PROGRESS
+0006 0005   WRITE6,1010) VALUE, SQUARE
                        $
*****  01)  IG1013I SYNTAX
+0009 0007   GO TO 1000
                        $
*****  01)  IG1005I ILLEGAL LABEL
*END
*GO
EDIT FORT,TEST
EDIT
LOCATE 6,1010
       WRITE6,1010) VALUE, SQUARE
CHANGE /E/E(/
       WRITE(6,1010) VALUE, SQUARE
LOCATE GO TO
       GO TO 1000
CHANGE /000/00/
       GO TO 100
SAVE
FORT.TEST
REPLACED
*END
*GO
```

図Ｃ２　（その２）　端末セッションの例

```
FORT.TEST
*IN PROGRESS
MAIN   =  000196
0044D0 BYTES USED
EXECUTION BEGINS  1.15

SQUARE PROGRAM

?
12
THE SQUARE OF     12.00  IS      144.0
?
-14.1
THE SQUARE OF    -14.10  IS      198.8
?
/CAN
*TERMINATED
*GO
```

図Ｃ２　（その３）　端末セッションの例

ラムを含んでいる。その第1行で使おうとしているFORTRANコンパイラのタイプを定義している。この場合は，FORTRAN GIコンパイラであり，それゆえ「／LOAD FORT GI」となっている。その9行あとの「EDIT」は，システムが印刷したもので，編集モードにあることを確認させ，編集作業を促している。その後，プログラムを実行させ，エラーが見つかったので再び編集モードにもどして訂正している。

図C2（その3）では，訂正したプログラムを実行させている。正しくコンパイルされ，実行に入り，「？」でデータを要求してきている。データを入力すると実行にもどり，結果が印刷される。2つのデータに対してこのプログラムを実行したのち，「／CAN」というキャンセル・コマンドを与えてセッションを終らせている。

(14) 適用業務パッケージ

MUSICシステムでは次のような適用業務パッケージが使える。

- STAKPAK（統計計算パッケージ）
- MUSIC／SCRIPT（(15)参照）
- DISK SORT
- COGO（Civil Engineering Coordinate Geometry Program）
- IIS（対話式教育システム）と以下のコース

  MUSIC OVERVIEW

  MUSIC SCRIPT

  MUSIC CONTEXT EDITOR

  MUSIC LINE EDITOR

(15) MUSIC／SCRIPT

MUSIC/SCRIPTは，テキスト処理サブシステムであり，MUSIC環境でサポートされているすべての端末で使える。MUSIC/SCRIPT は，3270上で非常によく動作する。MUSIC/SCRITは修正標識付け（modification flagging）能力をもっている。再印刷すると，修正個所がその行にあることを示す「縦線」文字が左端に印刷される。現在ATS システムを使っている人のために，ATSファイルからMUSIC/SCRIPT 形式への変換ユーティリティ・プログラムがMUSICに組み込まれている。綴りチェック・ユーティリティ，目次生成ユーティリティ，索引生成ユーティリティも組み込まれている。（配布されている）MUSICのドキュメンテーションはすべてMUSIC/SCRIPTを使ってつくられている。MUSICのドキュメンテーションを見ればMUSIC/SCRIPTがいかに強力か分る。

(16) 料 金 計 算

どんなシステムでもユーザが使用した時間に対する料金計算ができなければならない。MUSICシステムは，システムのすべてのユーザの料金計算情報をレポートする。端末時間に関する情報，端末入出力に関する情報を報告する。さらに，CPU使用時間，ディスクの使用，ディスクとテープの装塡に関する情報を記録する。MUSICシステムにはこれらのデータをいくつかのMUSIC内部カウンタ（値）になおしてしまう料金計算パッケージがある。このパッケージの出力をもとにして，（個々の導入個所固有の）料金計算をすればよい。

(17) ユーザ・プロフィール（User Profile）

システムのユーザはだれもユーザ・プロフィールをもつ。このプロフィールを用いて1日のうちシステムにアクセスできる時間を制限することができ

る。プロフィールを用いて，あるユーザは午前9時から午後5時まではシステムにアクセスできるとか，できないとか指定することができる。1日のある時間は時分割アクセス，バッチ・アクセスができないなどの指定ができる。さらに，ジョブの制限時間も1日のうちの時間帯にもとずいて指定することができる。9時から17時の間ではわずか4秒のCPU時間しか使えないが，それ以外の時間であれば，CPU時間を無制限に使える，というように指定することができる。ユーザが割り当てることができるセーブ・ライブラリ空間の量もユーザ・プロフィール中に保持される。システム管理者は，ユーザが使うことができる空間の最大量を指定することができる。また，システム管理者は，ユーザがセーブできる最大のファイルのサイズを指定することもできる。プロフィール中に，セーブ・ライブラリのセキュリティに対する省略時解釈規則を記入しておくこともできる。ユーザがいつもある特定の端末タイプを用いてサイン・オンする場合，その情報をプロフィールに設定することができる。従って，サイン・オンのときに端末のタイプを指定する必要がなくなる。ユーザのタブ設定の省略時解釈規則もこのプロフィール中に保持させることができる。

プロフィールにより合言葉（password）を変更可能かどうか指定することができる。MUSICでは同じコードでいく台かの端末から同時にサイン・オンできる。このような場合，ユーザによる合言葉の変更は好ましくないことは当然である。

(18) VM／370の下でのMUSIC

MUSICは，VM／370の下で非常に効率よい複数アクセス仮想計算機として動く。MUSICのスケジューラは，MUSIC環境内の個々のユーザをどのように制御すべきかをよく知っている。導入個所で応答時間の基準値を設

定することができる。MUSICのスケジューラは，その実行中設定された基準値を実現するよう努力する。MUSICは仮想計算機環境のいくつかの性能向上機構を利用している。その1つは，V＝R（virtual=real）オプションである。MUSICは1つの仮想計算機の中で多くの端末を取り扱う。それゆえ，VMの下で走る計算機を減らす効果がある。さらに，MUSICシステムはユーザ間でファイルを共有する能力ももっている。

(19) MUSICとVMの連繋

MUSICは，VMと連繋できた最初のIBM提供のオペレーティング・システムである。それは1973年のことであった。MUSICシステムがIPLであるとき，MUSICはVMと一緒に走るかどうか決定する。もし一緒に走るとすれば，V＝Vか，V＝Rかを決定する。これらの状況のもとで，MUSICシステムはそれぞれの環境で最適の性能を出すべく，動的に自分自身を変更する。MUSICがVMとV＝Vで動いているときは，VMのページの競合を小さくするためMUSICはそのバッファリング・アルゴリズムを変更する。さらに，VMと一緒に動いているとき，MUSICが出す特権操作（privilege operation）の数を減らす。これによってもVMのオーバヘッドはいくらか減少する。従って，MUSICはより効率よく（VMと一緒に）動作する。MUSICのIPL時にVMコマンドを出すことによって自動的にMUSICシステムをセット・アップすることができる。これによりオペレータの負担が減る。MUSICにはVMへのサブミット・インタフェースがあり，これによってMUSIC端末から別の仮想計算機に（制御された）サブミットが可能である。このインタフェースはFORTRANでかかれており，JCLの走査，代入を処理するために変更することができる。このインタフェースは，ジョブをMUSICのバッチにサブミットするためにも使うことがで

きる。

(20) ま と め

MUSICは高性能の時分割システムである。MUSICは使いやすく，導入しやすい。そしてひとたび導入されると，高い信頼性を示す。ますます多くのユーザがシステムを使ってみようとしている今日的環境では，MUSICはその最適の解を与えると考えられる。

D. 入力レベル対話式適用業務システム（ELIAS−ONE）

（Entry Level Interactive Application System−One : General Information Manual , IBMより）

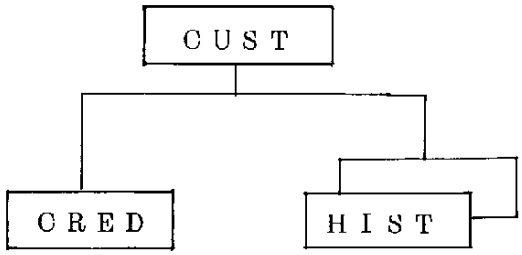
1. はじめに

ELIAS−Iを用いてDB／DC適用業務をどのように開発（維持）するか，例を通して説明する。

2. DB／DCシステムの構成要素

2.1 データベース

図D1のような構造をもつデータベースを例にとる。セグメント「CUST」は長さ114文字で，顧客番号（customer number，通常これを用いてアクススする），顧客名，顧客住所，省略形の名前（2次キーとして使うことができる）が入っている。セグメント「CRED」は長さ10文字で，信用買いの上限値（credit limit）と信用買い残高（credit balance）が入っている。顧客1人についてCREDセグメントが1つつくられる。CREDには識別フィールドがない。セグメント「HIST」は長さ42文字で，（その）顧客がそれまでに出した注文の要約である。



図D 1　　顧客データベースの論理構造

1人の顧客についていくつものHISTセグメントがあり，特定の日付のレコードにアクセスしたいという要求もあり得るので，HISTセグメントには識別フィールド（HISTDATE）がある。

2.2 PSB（論理的な見方）

ここでの問題のために用意するPSB（Program Specification Block，DL／I参照）では，CUST，CREDセグメントに読出し専用でアクセスできるように指定する。なぜなら，それら2つのセグメント中の情報（だけを）表示しさえすればいいからである。HISTセグメントは（ここでの）プログラムには分らないので，あたかもHISTセグメントが存在しないかのようにプログラムをかく。これには2つの利点がある。すなわち，プログラマの負担が軽くなるし，HISTセグメントの不正なアクセスが生じないことを保証することになる。

2.3 マップ（画面の書式）

画面（端末のスクリーン）のレイアウトを図D 2のように（すると）考える。これは書式であるから，レイアウトを変えずに，いくつかの問合せができるようになっている。1つの問合せに対する出力画面をそのまま次の問合せの入力画面にしている。

ここでの例は，初期画面が端末に表示された時点から（作業が）始まるものとしている。

2.4 プログラム

プログラムの（処理）論理を図D 3に示す。この図で使っている記述言語はいくぶん形式的であるが，ELIAS－Iユーザがプログラム仕様を記

```
KMAPS30              *** CUSTOMER and CREDIT ***

  Customer No.     : xxxxxx

        Name:      : xxxxxxxxxxxxxxxxxxxxxxxx
        Address    : xxxxxxxxxxxxxxxxxxxxxxxx
                   : xxxxxxxxxxxxxxxxxxxxxxxx
                   : xxxxxxxxxxxxxxxxxxxxxxxx

        Credit limit      : 9,999,999.99

        Credit balance    : 9,999,999.99

        Credit remaining  : 9,999,999.99


     Key next Cust No. and hit ENTER; any other key ==> Menu
        ---- Error Message (from program to user ----
```

図Ｄ２　　初期画面の書式

```
Receive data from KMAPS30
IF special key pressed on terminal THEN
  EXIT to MENU MAP
ENDIF
IF CNUMBER omitted THEN
  display error message 1
  EXIT retry
ENDIF
Move CNUMBER to data base retrieve key
Get Unique Segment = CUST
IF feedback = 'not found' THEN
  display error message 2
  EXIT retry
ENDIF
Get Next Segment = CRED
IF feedback = 'not found' THEN
  display error message 3
  EXIT
ELSE
  Move data from segments to output area
  Send to KMAPS30 for formatting
  Return
ENDIF

Error message 1: "Invalid customer number.  Please reenter."
Error message 2: "Invalid customer number.  Please reenter."
Error message 3: "No credit segment for this customer."
```

図Ｄ３　　プログラムの（処理）論理

述する技法の１つとして参考になるのではないかと考え，意識的に使っている。

3. データベースの生成

3.1 顧客データベース「KDBCUS」の定義

DB/DCシステムの中心はデータベースである。従って，システム管理者（もしくは，データベース管理者）がデータベースKDBCUSを定義する過程から話を始める。

（ELIASが出した）メニュから，システム管理者はデータベース定義を選択する。そうすると，画面は図D４のようにかわる。次に，システム管理者は（定義しようとしている）KDBCUSに合わせて画面の質問に答えていく。新しいデータベースを定義するわけであるから

- 「１（生成）」
- データベース名「KDBCUS」

とキー・インする。システム管理者がENTERキーを押すと，ELIASはそれらの回答を読み，矛盾がないかどうか（例えば，同名のデータベースが既に存在しないかどうかなど）をチェックし，その回答をしまっておく。

次に，（定義中の）データベースの物理的属性を入力する。図D５はそのための（ELIASが支援する）画面である。この画面から，ELIASがどのようなことに関してデータベースの定義を支援してくれるかが分る。表示されているフィールドは，支援のタイプに応じていくつかのクラスに分けることができる。

第１のクラスは「DIRECT ACCESS BLOCKS」というフィールドである。ここに入力する値は，このデータベースに割り当てる空間の大きさである。この値は100以下（小さな，テスト用のデータベース）から

と答えている。ここでも，分らないことがあればEXPLAIN機能を呼び出すことができる。

3.1 データの論理的見方の定義

システム管理者の最後の仕事は，プログラムからアクセスできるセグメントと，そのアクセスの仕方を決めてやることである。ここでの簡単な例では，物理的見方と論理的見方が一対一に対応しているけれども，実際のデータベースの利用では，通常1つの物理的見方に対していくつもの論理的見方が存在する。

論理的見方は，もう1つのELIAS－I手続きであるKEPSBを呼び出して定義する。この機能は，アクセスされるデータベースの名前を要求し，次にセグメントを1つずつ与えプログラマがそのセグメントにアクセス可能か否かをシステム管理者に決定させる。また，適用業務プログラムから参照可能なセグメントの組に記号名を付けさせる。この記号名が「PCB(Program Communication Block)名」である。

この例では，PCB名は「KDBCUS」にしている。この名前はプログラマがデータベースにアクセスするときに用いる。

以上の情報を入力するための画面を図D8に示す。

4. プログラミング

適用業務を作成するための残りの作業は通常適用業務プログラマが担当する。その作業は大きく2つに分けられる。

- マップ(map)の定義
- プログラムのコーディング

1,600万まで変り得るので，省略時解釈値（ default value ）は与えられていない。

第２のクラスはフィールド「RANDOMIZING MODUL」と「DBD DEVICE」である。省略時には，ELIASに組み込まれている標準のランダム化モジュール（ randomizing module ）が呼び出される。このモジュールは，キーの記号値にもとずいて，データベース中にデータを効率よく分散してくれる。経験を積んだユーザは自分自身のランダム化モジュールを（標準のモジュールにかえて）使うことができる。

大抵の導入個所では，ただ１つの型の直接アクセス記憶装置（ Direct Access Storage Device ）しか使っていない。その場合，導入時にELIASシステムにその（ DASDの ）情報を入れておく。そうすると，KEDBD手続きが実行されるときは，必ず省略時解釈値としてその情報が与えられる。いくつものタイプのDASDを同時に使っている場合は，図Ｄ５の画面上で省略時解釈値を（実際に使うDASDのタイプに）変更すればよい。別のタイプのDASDに（システム）変換する場合は，ELIASの初期化手続きを実行し直して，その新しいタイプを省略時解釈値に指定し直す。

第３のクラスは，「CI BLOCK SIZE」，「FREE BLOCKS」，「FREE BYTES」フィールドである。これらのフィールドには省略時解釈値が表示されているが，それらは，すべてのタイプのデータベース利用に適するような妥協値に過ぎない。経験を積んだユーザは，自分のデータベースの使い方に合わせて調整することができる。

画面上のフィールドの意味がはっきりしないとき，システム管理者は画面の一番下にある「EXPLAIN」を入力する。そうすると，そのときのELIAS手順は一時中断されて，詳しい説明情報が表示される。そのあと中断した点から処理が再開される。

このEXPLAIN機能はその特定の文脈に対応して詳細情報を提供するようにつくられており，端末セッションを中断して，参照マニュアルを見なければならないような必要度を少なくしている。画面定義を議論するときこのEXPLAIN機能の例をあげる。

3. データベースのセグメントの定義

以上でデータベースの物理的属性の指定が終った。次に，システム管理者はセグメントを定義し，データベースのキー・フィールドを与えなければならない。図D6のような画面がデータベースに入れたいセグメント（segment）ごとに（ELIASから）表示される。この例では，入力すべきデータは，第1セグメントの名前（CUST）とその長さ（114）だけである。

それらの情報を受け取ると，（ELIASは）図D7のようなフィールド定義用の画面を表示する。この場合必要なデータは

- フィールド名（適用業務プログラムと共通）
- CUSTセグメント中での（このフィールドの）位置
- 長さ
- このフィールドに入るデータの属性
- 新しく挿入されるセグメントの位置の決定にこのフィールドを使うかどうか

である。この例では，それぞれ

- CNUMBER
- 1（このセグメントの第1番目のフィールドである）
- 6
- C（文字データである）
- Y（「イエス」である）

```
                          DBD  SEGMENT DEFINITION                    ELI$DBE4

 Enter the required data and press ENTER.
********************************************************************************

 CREATE a segment in  HDAM data base  KDBCUS

EXIT                    ==>  N          Y = to leave segment definition

SEGMENT NAME            ==>  ________   Alphameric; starting with alpha
SEGMENT LENGTH          ==>  ____       Length limited by CI-size
PARENT SEGMENT          ==>  0          Enter 0 for root segment
PRECEDING SEGMENT       ==>  0          Enter 0 for root segment
TWIN CHAIN POINTERS     ==>  T          T = twin; TB = twinbackward
PARENT POINTERS         ==>  1          1 = Pointer to first child only
                                        2 = pointer to first + last child in
                                        twin-chain
SEGMENT TYPE            ==>  PC         PC  = Physical (child) segment
                                        RLC = Real logical child
                                        VLC = Virtual logical child
********************************************************************************
 CREATE -- DATASET --   PROCESSED
 ==>
  EXPLAIN                    CANCEL                      RETRY
```

図D 6　　セグメントを定義するための表示

```
                          DBD   FIELD DEFINITION                 ELI$DBE5

 Enter the required data and press ENTER.
********************************************************************************

 CREATE a field for segment  CUST in  HDAM data base  KDBCUS

EXIT                    ==>  N          Y = leave field definition

FIELD NAME              ==>  ________   Alphameric; starting with alpha
START POS. OF FIELD     ==>  ____       Segment starts in position 1
LENGTH OF FIELD         ==>  ___        Maximum length 255 (HIDAM-roots 236)

FIELD TYPE              ==>  C          C / P / X ; data characteristics of
                                        field
SEQUENCE FIELD          ==>  Y          Y/N

********************************************************************************
 CREATE -- SEGMENT -- CUST  PROCESSED
 ==>
  EXPLAIN                    CANCEL                      RETRY
```

図D 7　　フィールドを定義するための表示

```
                          DBD MODE SELECT                          ELI$DBE1

 Enter the required data and press ENTER.
********************************************************************************

MODE                    ==>  1          Please enter option required

DATA BASE NAME          ==>  _______    Alphameric; starting with alpha

   Option   Function                  Applies to
   ------   --------      .-----------------------------------------
      1      create        NEW DATA BASE DEFINITION (DBD)
      2      insert        SEGMENT / FIELD / SECONDARY INDEX
      3      modify        DATA BASE ACCESS METHOD / DATASET INFORMATION
                           SEGMENT / FIELD / SECONDARY INDEX
      4      delete        SEGMENT / FIELD / SECONDARY INDEX
      5      erase         WHOLE DATA BASE DEFINITION
      6      exit          TO SYSTEM ADMINISTRATOR PANEL

********************************************************************************

 ==>
  EXPLAIN                    CANCEL                    RETRY
```

図Ｄ４　　データベース定義のための表示

```
                      DBD   HDAM DATA SET DEFINITION                ELI$DBEA

 Enter the required data and press ENTER.
********************************************************************************

 CREATE dataset definition for  HDAM data base  KDBCUS

EXIT                    ==>  N          Y = leave dataset definition

DIRECT ACCESS BLOCKS    ==>  ________   Number of direct addressable blocks
                                        (Root Addressable Area = RAA)
CI BLOCK SIZE           ==>  1024       Multiple of 512; maximum 4096
RANDOMIZING MODULE      ==>  GENERAL    Alphameric, starting with alpha

FREE BLOCKS             ==>  0          0-100; but not 1; (every nth block
                                        left free)
FREE BYTES              ==>  0          0-99; (percentage of block left free)
DBD DEVICE              ==>  3340       FBA / 3330 / 3340 / 3350

********************************************************************************

 ==>
  EXPLAIN                    CANCEL                    RETRY
```

図Ｄ５　データベースの物理的属性を入力するための表示

```
                    KEPSB  -  DATA BASE ENTRY-PANEL                ELI$PSE2
 Enter the required data and press ENTER.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
You are now in  CREATE mode for PSB  KPSCC30

END OF PSB              ==>  NO          YES to stop entries in PSB
DBD NAME                ==>  kdbcus_     DBD name of Data Base
PCB NAME                ==>  kdbcus_     Symbolic Data Base name (default =
                                         DBD name)
SECONDARY INDEX NAME    ==>  ?           to control the processing sequence if
                                         required
        ----- Select  either one or more of these ------
GET                     ==>  YES         YES or NO   read segments
INSERT                  ==>  NO          YES or NO   insert segments
REPLACE                 ==>  NO          YES or NO   replace segments
DELETE                  ==>  NO          YES or NO   delete segments

             -------  or this one -------
LOAD                    ==>  NO          YES or NO   loading segments
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 ==>
  EXPLAIN                    CANCEL                    RETRY
```

図D 8　　論理的見方を定義するための表示

```
                 KEMAP - SELECT PANEL OR MAP NAME, OR EXIT          ELI$MPE1
 Enter the required data and press ENTER.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ACTION REQUIRED         ==>  1           1 -  CREATE a new screen definition
                                         2 -  UPDATE an existing definition
                                         3 -  DELETE an entire definition
                                         4 -  LEAVE screen definition


SELECT BMS OR DMS       ==>  BMS         -  BMS to create a BMS macro set
                                         -  DMS to create a DMS Panel
                                              Definition Form


PANEL / MAP NAME        ==>  _______     1 to 7 alphanumeric characters


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 ==>
  EXPLAIN                    CANCEL                    RETRY
```

図D 9　　画面定義のための最初の表示

4.1　マップの定義

図D2のようなマップを定義するためには，プログラマはシステムに，画面のサイズに関する一般情報と使用するプログラム言語を与えてやらなければならない。次に，プログラマは，画面の各論理フィールドを詳細に定義しなければならない。

ここの例では，24行の画面，COBOLプログラム用のマップをつくることにする。このマップを「KMAP30」と呼ぶことにする。

図D9はKEMAP手続きから最初に与えられる画面である。この画面に対して入力する回答は

- 1
- BMS
- KMAPS30

である。

図D10の画面は残りの一般情報について回答を求めている。最初の2つの回答は示された通りの省略時解釈がそのまま使える。タイトルは

'*** CUSTOMER and CREDIT ***'

で，その長さは27である。従って，図D10の3番目の質問の回答は「27」である。

最後に，対話式で各フィールドを定義する。第1番目のフィールドを定義するための画面を図D11に示す。

最初の数行で，マップの名前，そのサイズ，マップ上の次のフィールドに移るまえに使える空間（この場合，マップの最後の行）をプログラマにおしえている。画面の残りの部分を使ってユーザはフィールドを定義する。

経験の浅いユーザはこのあたりでEXPLAIN機能を呼び出したくなるかも知れない。ユーザが画面の最終行にある「END OF FIELD」オ

```
                        KEMAP - INITIAL  PARAMETERS                    ELI$MPE3
 Enter the required data and press ENTER.
+------------------------------------------------------------------------------+
 CREATE screen definition  KMAPS30 in  BMS

SELECT LANGUAGE          ==>  COBOL       COBOL or PL/I

SELECT SCREENSIZE        ==>  24          24 or 43 lines


             ------------  For BMS only ------------
ENTER TITLE LENGTH       ==>  56          value 1 to 56

+------------------------------------------------------------------------------+
 ==>
  EXPLAIN                    CANCEL                      RETRY
```

図Ｄ１０　　画面定義のための２番目の表示

```
                      ELIKEMAP - FIELD DESCRIPTION                 ELI$MPE5
 Enter the required data and press ENTER.
+------------------------------------------------------------------------------+
 CREATE screen definition  KMAPS30 in  DMS

Positions free before next field =  1679 - Screensize is  24 lines.
Cursor is set at Row  0 Column  0 Field  0

SELECT FUNCTION          ==>  1           1 - ADD, 2 - DELETE, 3 -  END
FIELD TYPE               ==>  _           BMS: 1- NUM, 2- ALP, 3- INIT, 4- PIC
                                          DMS: 1- NUM, 2- ALP
FIELDNAME                ==>  ?           1 to 7 alphanumeric characters
LENGTH                   ==>  __          Enter value 1 to 79
LINE                     ==>  2           Enter value 2 to 22
COLUMN                   ==>  1           Enter value 1 to 79
CURSOR                   ==>  N           Y, for CURSOR on this field

          ---------- For BMS only ----------
END OF FIELD             ==>  1           1-None, 2-PROT.Fld., 3-ASKIP Fld.
+------------------------------------------------------------------------------+
 ==>
  EXPLAIN                    CANCEL                      RETRY
```

図Ｄ１１　　画面のフィールドを定義するための表示

プションの意味を知りたくなったとしよう。ユーザは画面の最終行のEXPLAINを入力する。この場合，説明を要求できる項目がいくつかあるので，EXPLAINメニュ（図D12）が表示され，どの項目の説明を要求しているかをユーザに選ばせる。「END OF FIELD」オプションの説明を得るには，プログラマはメニュの項目番号5を選ばなければならない。

システムから2ページにまたがる情報が与えられる（図D13）。ユーザが顧客番号をキー・インしていて，その入力が定義されている長さを越える場合，キーボードをロックしてそれを防ぎたければ，END OF FIELDのオプション2を選べばいいことが分る。

画面の一番上の行のフィールドはELIAS－Iが自動的に決定する。必要な情報はただ1つ（タイトルの長さ）だけであり，それはすでに入力済みである。

```
                KEMAP INDEX TO EXPLANATION OF FIELD DEFINITION          ELI$MPE5
 Select the category of information you require.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   1 INFORMATION SECTION          This panel describes the reminder informa-
                                  tion held on the screen.
   2 SELECT FUNCTION              Which functions may be selected for each
                                  field.
   3 FIELD DESCRIPTION            How to define the name, type, and size of
                                  your field.
   4 FIELD POSITIONING            How to specify where your field is to
                                  appear on the panel or map.
   5 CURSOR AND END OF FIELD      How to set the Cursor.
                                  How to avoid operator errors from overrun-
                                  ning the end of a field when using the map.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ==>
              RETURN       INITIAL      END
```

図D12　フィールド定義画面の説明用のEXPLAIN表示

```
                    KEMAP - CURSOR AND END OF FIELD OPTIONS          ELI$MPEE
                                                                    PAGE 001/002
 END leaves explanation, FORWARD continues explanation.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                        CURSOR Option for DMS and BMS

 CURSOR:  by selecting this option, you ensure  that when the map being defined
is sent to the terminal, the default location for the cursor is at the start of
this field.  This may be over-ridden in the  program, (to point to a field with
invalid data, for  example), but will normally  be set to the  first data entry
field on the screen.




++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ==>
  OVERVIEW                    END                                  FORWARD
```

図D13（その1）　CURSORオプションとEND OF FIELDオプション
（1ページ目）

```
                    KEMAP - CURSOR AND END OF FIELD OPTIONS          ELI$MPEE
                                                                    PAGE 002/002
 END leaves explanation, BACK continues explanation.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                        END OF FIELD Option only for BMS

 END OF FIELD: This option controls what will happen if the terminal user tries
to key "off the end" of the field being defined.

 1  means  that the  user  can  go on  keying,  though  the  data will  not  be
transferred to the program.

 2 means that the user will be stopped, because the keyboard will lock.

 3 means that the user will skip automatically to the next unprotected field.




++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ==>
  OVERVIEW                    END                    BACK
```

図D13（その2）　CURSORオプションとEND OF FIELDオプション
（2ページ目）

定義する最初のフィールドの内容はリテラル

'Customer No.　　:'

である。このフィールドを定義する際の入力（キー・イン）を図D 14に示す。

リテラルの実際のテキストはまだ入力されていないことに注意する。実際のテキストは後の編集段階で入力される。

次に定義するフィールドは顧客番号（CNUMBER）フィールドである。この定義を図D 15に示す。

名前CNUMBERを選べば，画面からこのフィールドにキー・インされるデータに記号名CNUMBERI（I＝入力）をつけたことになり，その名前で（キー・インされた）データを操作することができる。またCNUMBERO（O＝出入）に値を入れる（moveする）操作によって，(そのmoveされた）データを（CNUMBERという名前の）このフィールドに表示できる。

フィールドの名前，位置，タイプという基本的な定義をすべて入力してしまうまで，以上のフィールド定義を繰り返す。そして，図D 11 ，D14，D15の画面上の「SELECT　FUNCTION」に対する回答として「3」を入力することにより「End of Map」を指示する。それに応えて，ELIAS－Iの手続きはマップの定義に必要な文（ステートメント）の組を生成する。もしELIASを使わなければ，この文の組はプログラマがコーディングしなければならない。

この時点で，プログラマはシステム・エディタ・プログラムを用いてこれらの文を編集することができ，リテラルに対して実際の値を入力する。

4.2　プログラムのコーディング

```
                    ELIKEMAP - FIELD DESCRIPTION                    ELI$MPES
 Enter the required data and press ENTER.
 CREATE screen definition  KMAPS30 in BMS

Positions free before next field =  1679 - Screensize is  24 lines.
Cursor is set at Row  0 Column  0 Field  0

SELECT FUNCTION          ==>  1          1 - ADD, 2 - DELETE, 3 -  END
FIELD TYPE               ==>  3          BMS: 1- NUM, 2- ALP, 3- INIT, 4- PIC
                                         DMS: 1- NUM, 2- ALP
FIELDNAME                ==>  ?          1 to 7 alphanumeric characters
LENGTH                   ==>  17         Enter value 1 to 79
LINE                     ==>  3          Enter value 2 to 22
COLUMN                   ==>  3          Enter value 1 to 79
CURSOR                   ==>  N          Y, for CURSOR on this field

          ---------- For BMS only -----------
END OF FIELD             ==>  1          1-None, 2-PROT.Fld., 3-ASKIP Fld.

 ==>
  EXPLAIN                    CANCEL                    RETRY
```

図D14　　顧客番号フィールドの定義

```
                    ELIKEMAP - FIELD DESCRIPTION                    ELI$MPES
 Enter the required data and press ENTER.
 CREATE screen definition  KMAPS30 in  BMS

Positions free before next field =  1579 - Screensize is  24 lines.
Cursor is set at Row  0 Column  0 Field  0

SELECT FUNCTION          ==>  1          1 - ADD, 2 - DELETE, 3 -  END
FIELD TYPE               ==>  1          BMS: 1- NUM, 2- ALP, 3- INIT, 4- PIC
                                         DMS: 1- NUM, 2- ALP
FIELDNAME                ==>  cnumber    1 to 7 alphanumeric characters
LENGTH                   ==>  6_         Enter value 1 to 79
LINE                     ==>  3          Enter value 2 to 22
COLUMN                   ==>  21         Enter value 1 to 79
CURSOR                   ==>  y          Y, for CURSOR on this field

          ---------- For BMS only -----------
END OF FIELD             ==>  2          1-None, 2-PROT.Fld., 3-ASKIP Fld.

 ==>
  EXPLAIN                    CANCEL                    RETRY
```

図D15　　CNUMBERフィールドの定義

プログラムのコーディングも，マップの生成と同じような過程をたどる。ELIAS－Iとの最初の対話でコーディングするプログラムの基本的な性質を指定し，ELIAS－Iに適用業務プログラムの基本的枠組をつくらせる。この枠組のサイズはプログラムの性質によって異なるが，約250個の文からなる既製コードであると考えればいい。

次に，ELIAS－Iのエディタ・プログラムを使って，プログラマはこの枠組に自身の（処理）論理を書き加えていく。ここでの例の場合，プログラマは，まず顧客番号を得るために，画面から入力されたデータを読もうとするであろう。

ELIAS－Iのエディタを使って，次のコマンドを書くだけでよい。

MPR KMAPS30

これは，マップKMAPS30からデータを受けとるMape Receiveコマンドである。このコマンドによって，マップからデータを受け取るためのCICSステートメントがつくられ，プログラムに組み込まれる。この時点で顧客番号の値はフィールドCNUMBERIに入る。

データベースからレコードを取り出すため，プログラマは通常のCOBOLステートメントを用いてこの値（顧客番号）をDL/Iが使っている領域に移さなければならない。これはセグメント探索引数（Segment Search Argument，SSA）と呼ばれている。ELIASはCUSTセグメントを取り出すためのSSAを生成し，それにSSACUSTという名前をつける。データベースを読む準備としてプログラマがしなければならないことは，ステートメント

MOVE CNUMBERI TO SSACUSTQ-KEY.

をコーディングすることだけである。

読出しを行なうためには，別のコマンドをもう1つ与えなければならな

い。プログラマは一意的に識別されるレコード（ある特定の顧客のデータ）をデータベースから読み出したいわけであるから，Get Unique 呼出しを行なえばいい。すなわち，

GU KDBCUS

というコマンドを入力する。GUはGet Uniqueの省略形である。論理名KDBCUSは，システム管理者がデータの論理的見方を定義したときつけたものである。

プログラムには書換えを必要とするステートメントが組み込まれている。変更されるステートメントには，プログラマにして欲しいことがCOBOLの注釈ステートメントの形式で書かれている。プログラマは

- 顧客の情報を転送する先の入出力領域の名前
- データベースからの取り出しが正しく行なわれた場合に制御権を移すべき，その移し先のラベル
- データベースからの取り出しが不成功であった場合に制御権を移すべき，その移し先のラベル

などを与えてやらなければならない。

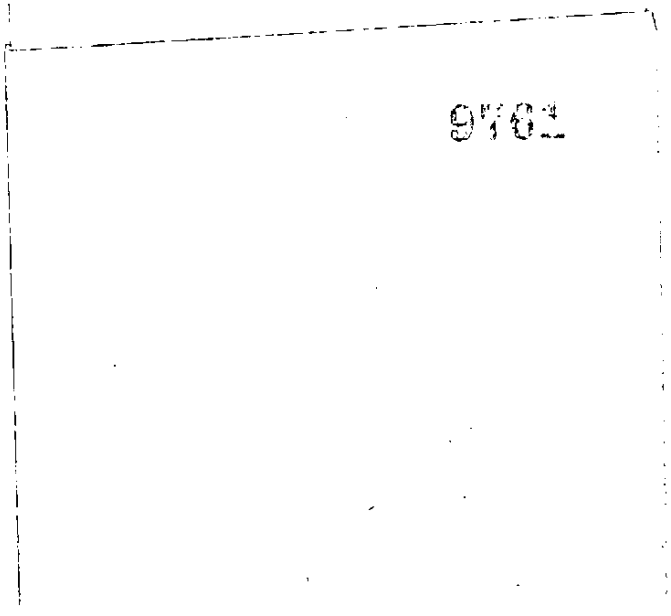
同様にして，CREDセグメントからレコードを取り出し処理するレコードを生成する。

次に，プログラマは通常のステートメントを用いてデータベースからのデータを出力マップに移す。さらに，回答の形式を整え，端末の前のユーザに（その回答を）返すコマンドを出す。

このプログラムは簡単である。5ないし6画面を通しての対話でまず枠組をつくれば，そのあと，COBOLステートメント20個，ELIAS－Iのコマンド10個程度で，ここでの例のプログラムをつくってしまうことができる。

この報告書は，日本自転車振興会の補助金の交付を受けて実施した「昭和55年度上級情報処理技術者養成等の補助事業」の一環として作成したものです。

昭和56年3月発行
財団法人　日本情報処理開発協会
情報処理研修センター
〒105　東京都港区浜松町2丁目4番1号
（世界貿易センタービル　7階）
TEL　03（435）6511（代）


9761