TECHNOLOGY REVIEW

UNISYS 技報

通巻 29

1991年5月発刊 Vol. 11 No. 1

論 文

日本ユニシス

ISSN 0914-9996

システム移行には，移行ツールは不可欠である．最近の移行システムの増大と多様化に伴い，移行手法も複雑になり，従来の移行ツールでは補いきれない点が出てきた．移行に際しての個別改造要求も増えてきた．井上博行の**COBOL移行作業におけるAI指向ツールの作成**は，これらの問題点を解決するために，AIの考え方を導入して移行情報の移行ツール本体からの独立，プログラムの流れを追う機能の搭載を目指した新移行ツールとしての検証ツールを開発したことを報告している．

シリーズ2200・1100の初期の頃からのユーザには，FORTRANで事務計算システムを開発してきたところが少なくない．今回あるユーザから，FORTRANプログラムをCOBOLプログラムに自動的に書き換える言語変換ツールの提供を求められた．日本ユニシスでは，AIマシン・AI技法を活用し，FTC（Fortran-To-Cobolトランスレータ）と呼ぶ言語変換ツールをKS 303（Explorer II）上のLisp言語で開発した．瓜生和史の**Fortran-To-Cobolトランスレータの開発**は，FTCトランスレータの概要および運用について述べるとともに，変換実績の評価，今後の課題について言及している．

オフィス・コンピュータを利用しているユーザでも，エンドユーザの関連部門にコンピュータを設置し，導入展開から運用フォローまで独自の方法で支援する例が増えつつある．このような大規模分散型システムをかかえたユーザにとって，工数のかかることが必至な展開後の運用フォロー等の問題をいかに最小限にとどめるかが成功の可否を握っている．日高修一・寿賀徳静は，**大規模分散処理環境下における開発および保守**の中で，この種のシステムの実現を目指しているユーザの事例を通し，開発準備段階の構想，システム設計段階での事前対策，ネットワークによる遠隔保守の実現，現状の問題点とその取り組み，等の観点からその有効性と今後の課題について述べている．

大規模なソフトウェアの設計情報を構造化・電子化された情報として保存することは，構造情報を利用した設計時・設計修正時の新たな計算機支援の可能性を有し重要である．内田修市・平川豊・門田充弘の**通信ソフトウェア設計支援環境——設計行為からの履歴情報獲得と修正支援機能**は，設計情報とその間の関係を表現する設計履歴のモデルを紹介し，次に，信号シーケンス図作成ツール・SDL（Specification and Description Language）図作成ツールの操作から設計履歴情報を半自動的に獲得する手法を提案している．さらに，蓄積した設計履歴情報を活用した設計修正時の支援機能を提案している．

毛利幸雄の**フォールト修正文に基づくフォールト混入工程の分析**は，ジャクソン法（JSP法）に従ったソフトウェア開発を対象として，いわゆるバグに相当するフォールトを分析し，フォールトがプログラム中に混入した工程を求める分析手法を提案している．なお，本稿では，JSP法の開発過程で開発者が行うフォールトの発見・除去の作業は，フォールトリポートとして報告されるものと仮定している．

マイクロ・メインフレームの中心となるホストとのセッション接続を行うプログラミング手法として状態遷移図に基づくプログラミング手法を採用した．本手法の採用により，セッション接続の失敗の確率を大幅に下げることができた．竹内征勝は，**状態遷移に着目したプログラミング手法**の中で，状態遷移図をパラメタの形で定義し，そのパラメタをプログラムで直接扱う方法について述べている．さらに，プログラム・ロジックの一部を外部パラメタ化できることにより，プログラム開発の生産性にも寄与できることを示している．

高品質なソフトウェアを作るためには，周到な教育・訓練が必須であり，開発現場における教育といえども，長期的な視野を持ち，絶えざる工夫・配慮が必要である．勝田祐輔の**ソフトウェア開発現場におけるプログラミング教育**は，ソフトウェア開発の現場において実施したプログラミング教育の経験に基づき，その考え方および方法を整理したものである．

COBOL移行作業におけるAI指向ツールの作成

An AI-oriented Tool for COBOL Program Conversion

井 上 博 行

要 約 ホストコンピュータの入れ替えに伴うシステム移行には,移行ツールは不可欠である.しかし,従来は単純なバッチプログラムの移行が主体であったシステム移行も,近年はデータベースや画面を使ったプログラムの移行の増加とともに移行手法も複雑になり,その結果,従来の移行ツールでは補いきれない点が増えてきている.同時に,移行を行う各ユーザの仕様に合わせた個別改造要求が非常に増え,移行ツールの個別改造作業が追いつかない状況にある.このような状況から,現在・今後の移行作業に対応できる移行ツールが必要となってきた.

今回,新たな移行ツールの一つとして,本稿で紹介する検証ツールを開発した.その際の開発目標として,上記二点を解決するために「個別対応作業の容易化,従来の移行ツールの限界に対応できる機能の搭載」を設定し,作成のためのアプローチを次のように行った.

1) 移行情報を移行ツール本体から独立させる (AIの考え方を導入).

2) プログラムの流れを追う機能を搭載する.

最後にこれらのアプローチに対する評価を行うことによって,今後のツールのあり方を示唆する.

Abstract The program conversion tool is indispensable for systems conversion resulting from the replacement of host computers. Systems conversion, which used to be done by converting arrays of simple batchprocessing programs for the most part, has become complex and complicated in its methods in parallel to an increase in the number of program conversion needs which deal with data bases and graphics. As a result, there are more and more cases where traditional conversion tools are not helpful enough to satisfy all conversion requirements. At the same time, independent modification requirements based on the specifications of computer users intending to convert their systems are augmenting so enormously that independent modifications to conversion tools can no longer catch up with their requirements. Triggered by those changes, a need has emerged for the conversion tool which is capable of meeting both the current and future conversion requirements.

The author has developed a verification tool as depicted in this paper, which is one of the new conversion tools. The development goal adopted was "to make it easier to work on independent coversion requirements and to let it have capabilities of responding to what conventional tools have been unable to handle" for the solution of the problems mentioned above. The approaches taken for the development are as follows:

1) Separation of conversion data from the main conversion tool framework (on the basis of AI concepts)

2) A newly imbedded function which serves to trace program flows

The last part of the paper suggests what future conversion tools ought to be after those approaches are evaluated.

1. はじめに

コンピュータ業界の競争の中では，他社からのリプレイス，自社内のレベルアップによるホストコンピュータの入れ替えが活発に行われている．その際，それまでのソフトウェア資産を有効に活用するために，システム移行（現在稼働している業務処理システムを最大限に有効利用し，短期間・低コストで新しいシステム環境へ移植する作業）を行う場合が多い．各社とも，その作業をいかに円滑に進めるかに苦心し，そのための手段としての移行ツールを多種多様にそろえている．その「いかに円滑に進めるか」というテーマに対し，現在のシステム移行作業において改善すべき大きな点が二点ある．一つは移行作業工程の見直しであり，もう一つは移行ツールの見直しである．

現在のシステム移行作業の各工程は，図1の通りである．実施工程の手修正・検収作業（図2）と準備工程以外は移行ツールによって多くの作業を自動化している．各工程とも改善の余地はあるが，検収作業ではとくに

1) 人間が目で調べるため非常に時間がかかる，
2) その作業のために大量のリストを出さなくてはならない，
3 プログラムの流れを追って判断しなければならない修正もあり，修正漏れを見逃す可能性もある，

等の大きな問題を抱えている．今回，移行作業工程の見直しとして，これらの問題を解決する「検収作業を自動化する移行ツール（ツール名『検証ツール』）を新たに手掛けることにした．

準備工程	調査	最適なシステム移行を行うため，現行システムを調査，分析，検討し新規システムに反映する．
	プランニング	移行マスタスケジュール，移行体制等を検討し決定する．
	準備	調査の結果，必要に応じてツールの改良・開発，各種マニュアル類を準備する．
↓		
実施工程	資料作成	プログラムコンバージョン用資料を準備する．
	プログラム変換	文法変換ツールを用いて，ソースプログラムを自動変換する．
	手修正	自動変換を行わない部分を手修正し，クリーンコンパイルする．
	検収	作業担当者が正しく修正が行われているかチェックを行う．
↓		
システムテスト工程	総合テスト	一連の通しテストを行い，システムとして結果を検証する．
	併行テスト	現行システムと新規システムとで併行に実行を行い，本番移行に万全を期す．
↓		
本番		

図1 システム移行の各工程

Fig. 1 Conversion system process

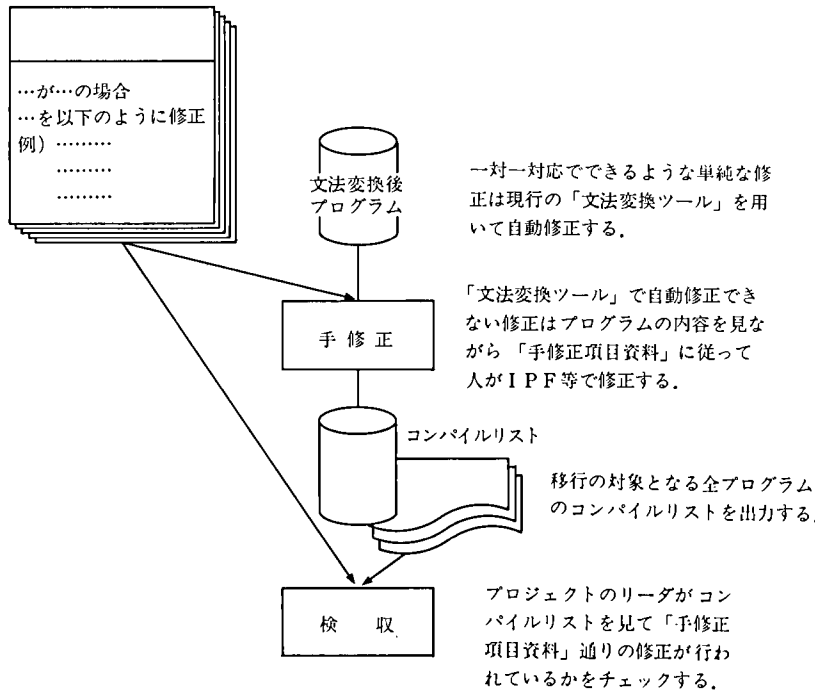


図2　手修正作業と検収作業の説明

Fig. 2　Explanation of checking amendment by persons

一方の移行ツールの現状を説明する．通常，移行する各ユーザの仕様に合わせるために移行ツールに個別改造を行うが，最近の移行すべきシステムの増大と多様性に伴い個別改造の要求が非常に増え，従来の移行ツール（今回新規に作った移行ツールと区別するために以下，従来ツールと呼ぶ）の個別対応が追いつかない状況にある．同時に，従来は単純なバッチプログラムの移行が主体であったシステム移行が，近年はデータベースや画面を使ったプログラムの移行の増加とともに移行手法も複雑になり，その結果，従来ツールでは補いきれない点が出てきた．

今回新たに移行ツールを作成するに当たって，従来ツールの持っている上記の二つの問題を解決するということを念頭におき，本稿で紹介する二つのアプローチを行った．

2. 従来ツールの問題点

2.1 保守性の悪さ

従来ツール作成方法の一番の問題点は，保守性（一般には，デバッグのしやすさや標準機能の追加・修正のしやすさをいうが，本稿ではそれらに加えて，個別対応による個別機能の追加・修正のしやすさも含んだ意味で使用）の悪さである．具体的には，以下の問題を抱えている．

1)　従来ツールの個別改造作業は，そのツールの仕様をよく理解している担当者以外にはむずかしい．

2) 従来ツールの仕様は，プロジェクトごとの仕様に合わせるので個別改造が大変多く，個別対応にかかる時間が大きい．

3) 個別改造を重ねるごとに従来ツールのプログラムが膨れ上がり，ますます個別改造作業を複雑にしている．

これらの問題の原因は，従来ツールの構築方法にあろう．つまり，ユーザの要求仕様をすべて取りまとめた上で，それに関するデータをすべてプログラムの中に取り込むという，プログラム中心の方法である．もし，検収作業を自動化する移行ツールを従来通りの方法で作るならば，次に示すような問題が依然残るであろう(図3)．たとえ構造化・サブルーチン化を図って上記問題点の改善を試みても，解決にはならない．

1) 手修正に必要な全情報を，VALUE 句・MOVE 文等を使って移行ツールのプログラムの中にセットしなければならないので，冗長なプログラムになる．

2) 数多くの手修正項目と対応した数の IF 文が必要なので，冗長なプログラムになる．

3) 各 IF 文の判別条件は個々の手修正項目に特有のものではないので，各 IF 文の中で同じようコーティングが現れ冗長なプログラムになる．

4) 個別改造仕様の中で明らかに PROC-C 2 を実行するケースがないとわかっている場合でも，既存のコーディングを削ることはないので，無駄なコーディングが多くなり，作業効率が悪くなる．

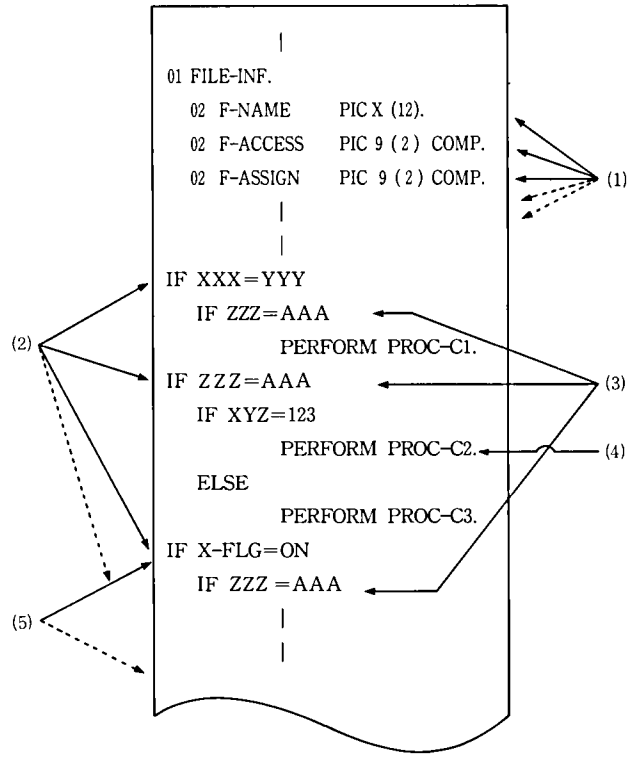


図3 従来のシステム構築例

Fig. 3 Usual method of making conversion-tools

5) 機能追加の場合，追加した処理の結果（レジスタ・フラッグ・ワークエリア等）が他の結果に影響を与えないかどうか，ということにいつも気をつけなければならないので，作業効率が悪くなる．

2.2 機能の限界

従来ツールの一つである文法変換ツールは，単純なイメージ変換だけでなく，内部フラッグの状態を見て条件を満足した場合のみ変換する，という機能を持っている．しかし，変換の内容がプログラムの流れ(PERFORM 文や GO 文等によるプログラム制御の移り先を考慮した，プログラムの始まりから終わりまでの流れ）の内容によって変わる，というような変換には対応していない．最近のシステム移行では，この機能を必要とする作業が大幅に増えないでは済まされなくなってきている．

3. 新移行ツールの作成

3.1 保守性の悪さに対して

3.1.1 AI の考えの導入

保守性の問題より，次のような対策を考えた．

1) 個別改造作業の内容は移行情報（手修正情報）によって左右されるので，それをツール本体から独立させ，個別対応を移行情報群の対応だけで済むようにする．ツール本体の動きは，この移行情報群が決めるようにする．

2) 一つ一つの移行情報による処理結果を他の処理結果から独立させる．

このように，プログラムとデータを分離してデータがプログラムの動きを決めるようにさせるという考えは，とくに画期的なものではない．AI(Artificial Intelligence；本稿では AI の一分野であるエキスパート・システムの意味で使用）の考えというのがまさしくこれに当たるからである．その使われ方は，医療システムのような推論を行う高度な専門知識を必要とする分野に限られているので，推論をとくに必要としない検収作業に応用しようというのは多少仰々しいかもしれない．しかし，AI のシステム構築方法は上で述べた考えを確かに具現化できると考え，新たな移行ツール作成に AI の考えを取り入れることにした．

ここで，もう少し具体的に AI とはどういうものか，また，AI の考えを取り入れることでどのような効果が期待できるのかを簡単に説明する[1]．

AI の一番の特徴は，あるシステムをプログラム部分(図 4 で示す推論部，以下推論部と呼ぶ）とプログラムの動きを決める部分（図 4 で示す知識部，以下知識部と呼ぶ）とに分離し，知識ベース（専門家の知識を一定の形式で表現し蓄えたもの[1]）を個々の断片知識に分割することである．この分離・分割によって次のような効果が得られる．

1) 中核の推論部と最低限の知識部が用意できれば，ユーザのすべての要求が確立するのを待たずに，システムの設計・制作段階へ進むことができる．

2) 知識ベースは後からも追加できるので，システムは半永久的に拡張することができる．

3) 個々の断片知識の内容が明確になる．また，知識ベースは，断片知識のさまざまな組み合わせによって変えることができる．

4) 知識ベースの内容は，すなわちユーザの仕様要求なので，知識ベース作り（以

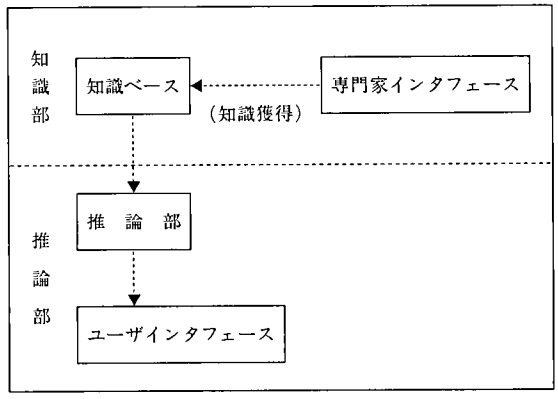


図4 AIの概念図

Fig. 4 Conceptual configuration of AI

下，これを知識獲得と呼ぶ）をユーザに解放すれば，ユーザ自身がシステムの保守作業を行える．

今，推論部を検収作業，知識ベースを手修正情報として，図5のようにわれわれの移行環境をAIの環境に当てはめてみると，以下のことが言える．

① 1)，2)の効果により，移行ツールの設計の着手からリリースまでの時間を短縮することができる．

② 3)の効果により，各プロジェクトに合わせた知識ベースが用意でき，個別対応は個々の知識DBの追加・削除だけで済む．

③ 4)の効果により，移行ツール作成者の個別対応作業負担がなくなる．

このように，AIの考えを取り入れた移行ツールを作ることができれば，2.1節で述べた，どの問題にも対応できる見通しがたった．

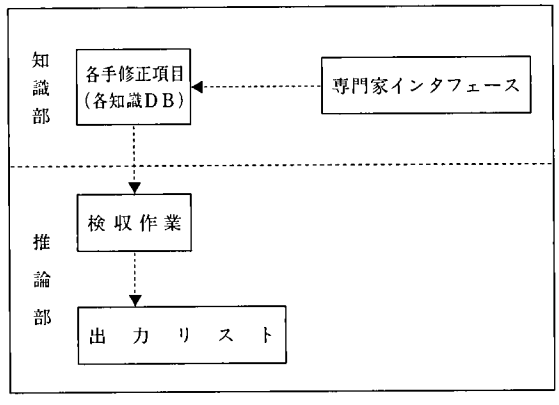


図5 AIの考えを取り入れた移行ツール概念図

Fig. 5 Conceptual configuration of Kensyotool

3.1.2 AIの考えの具現化

これらのAIの考えを具体的な形にした新しい移行ツールが実際にどのような動きになるかを図6に示す．

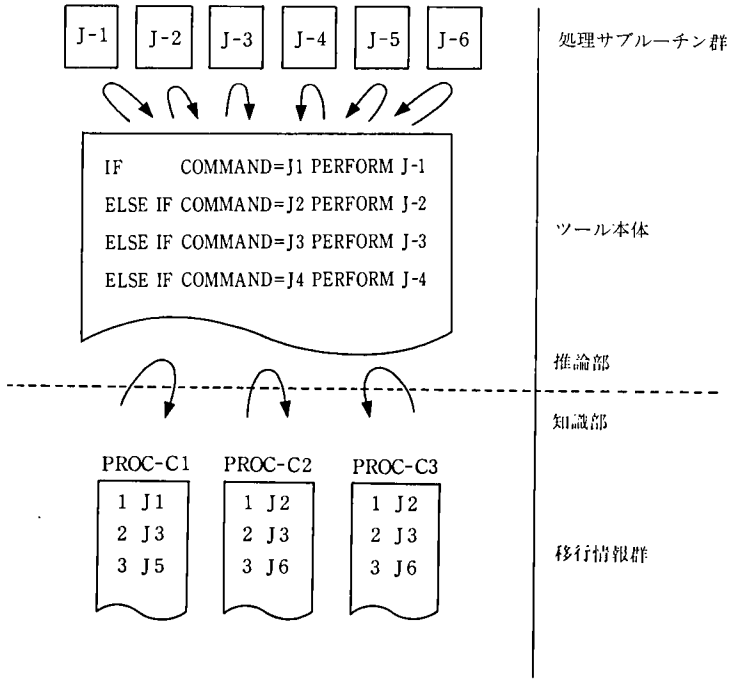


図6　AIの考えを取り入れたシステム構築例

Fig. 6　New method of making conversion-tools

推論部および知識部の機能概要は次のとおりである．

1)　推論部……ツール本体．ここでは，知識部から入力されたデータの順にサブルーチンを実行する．手修正項目のチェックの順番・内容は，すべてツールの外からデータとして指示するので，この中では一切考慮する必要がない．

　手修正の結果をチェックする処理はサブルーチンにして保守作業を楽にしている．各々のサブルーチンは独立して処理を行うので，お互いの処理内容・処理結果をまったく気にする必要がない．

2)　知識部……移行情報群．移行ツールのプログラムの動きを指示するものをここで作成する．PROC-Cnの一つ一つは一つのプログラムと同等の働きをするので，ここで得られた結果は他で得られた結果に影響を与えない．知識ベースの保守作業は，これらPROC-Cnの組み合わせだけになる．また，PROC-Cnの中の一つ一つのJnをコマンドにすれば，PROC-Cnの保守作業は，これらJnの組み合わせだけになる．

3.1.3　知識部の作成

知識部では，専門家の頭の中にある知識をいかにデータ化するか，という知識獲得の方法が重要である．AIにおける知識表現方法として，これまでいくつかの方法が提唱されてきている．たとえば，概念や物事を表すデータ構造の枠組みを階層的に構造化するフレームベースシステム，複数の知識が協調して問題解決する作業場（黒板）を用意して推論を行う黒板モデル，フレームベースシステム・黒板モデル・ルールベースシステムのそれぞれの特徴を活かして組み合わせたハイブリッド型システム，等

がある．今回は対象とする知識の特徴から，ルールベースシステムを選んだ．

ルールベースシステムの表現方法には，

① 「IF 命令，THEN 命令」

② 「IF データパターン，THEN 行動」

の 2 種類がある．①は「IF 風が吹く，THEN 砂ぼこりが舞う」というような三段論法の推論を行うルールである．

②は知識ベースを個々の断片知識（以下，知識 DB と呼ぶ）の蓄積と考え，その断片知識を②のように記述してデータ化し，内部のデータが「データパターン」に合うと「行動」に記述されたデータ処理を行うシステムである．このルールでは不確実な知識は扱えないが，プログラミング言語的な性格を持つので，われわれにとっては非常に扱いやすい[1]．このデータパターンや行動を表現するための手段としてわれわれは LAKET（LAnguage for KEnsyo Tool）という言語を作成し，LAKET の記述で知識獲得を行うことにした．具体的には，一つ一つの手修正項目資料の内容を図 7 に示すフォーマットに従い，表 1 に示すコマンド（全コマンド数は 74）を用いて記述する．[IF] から [END] までが一つの知識 DB として管理される．さらに具体的に LAKET のコーディング例とその説明を図 8 ・表 2 に示す．

LAKET は，当社の文法解析用言語である GSA（General Syntax Analyzer）とシステム記述言語である PLUS（Programming Language for Unisys System）で作成した．一般に，AI システムの構築といえば，AI システムを記述するのに優れているといわれる LISP・PROLOG といった言語が使わわれるようである．しかし，これらの言語でなければ AI システムが実現できないのではなく，より簡単にできるという程度の違いでしかない[1]．また，検証ツールは汎用機上で動かすので，汎用機でよく使われている言語を使う方が開発上都合がよい．このような理由より，入力のプログラムソースの文法を解析してさまざまな情報を読みとる処理には文法解析に適した GSA を，その他のデータ処理には PLUS を採用した．他の候補として COBOL も上がった

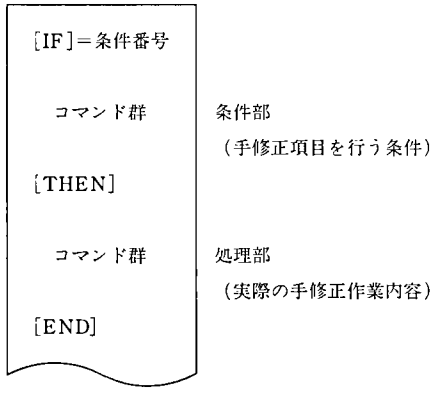


図 7　知識 DB の概要

Fig. 7　Configuration of knowledge base

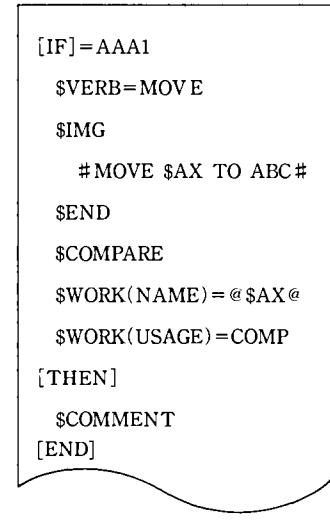


図 8　LAKET のコーディング例

Fig. 8　Coding of LAKET

表1　LAKETコマンド例
Table 1　Commands of LAKET

$FILE (NAMEI)=ABC	ABCという内部ファイル名についての情報を検索する.
$FILE (ASG)=PRINTER	PRINTERに割り当てられているファイルを検索する. または,あるファイルがPRINTERに割り当てられているかどうかをチェックする.
$WORK (NAME)=XYZ	XYZというデータ項目についての情報を検索する.
$WORK (USAGE)=COMP	COMPタイプのデータ項目を検索する. または, あるデータ項目がCOMPタイプかどうかをチェックする.
$VERB=MOVE	MOVE命令のあるラインを検索する.
$INSERT	このコマンドに続くイメージを挿入する.
$COMMENT	位置づけられたラインをコメント化する.

表2　図9の説明
Table 2　Explanation of Fig. 9

[IF]=AAA 1	条件番号を「AAA 1」とする.
$VERB=MOVE	MOVEの出現するラインにポインタを位置づける.
$IMG #…# $END $COMPARE	位置づけられたラインのイメージが#で囲まれた中のイメージと同じかどうか比較. 同じならば, $AXというワークエリアにデータ項目名を保存. 同じでなければ, 次のMOVEの出現するラインへポインタを位置づけ, 同処理を続ける.
$WORK(NAME)=@$AX@	$AXに保存されたデータ項目名のデータ項目についての情報を検索する. その名前のデータ項目がなければ, 次のMOVEの出現するラインへポインタを位置づけ, 同処理を続ける.
$WORK (USAGE)=COMP	その項目がCOMPタイプかどうか調べる. COMPタイプでなければ次のMOVEの出現ラインへポインタを位置づけ, 同処理を続ける.
$COMMENT	現在位置づけられているラインの7カラム目に削除マークをつけたイメージを正解ファイル上に作る.

が, 文字操作処理が機能的にGSA・PLUSに比べ劣る等の理由で見送った.

3.1.4　推論部の作成

推論部の役割は,「知識ベースのデータ (LAKETで得たデータパターン) が内部データと合うかどうかを調べ, 合えばデータ処理を行う」ことである. ここで大事な点はそのチェック方法である. もし, AIの手法に則るならば, 個々のデータパターンが入ってくるごとにGSAを用いて検収するプログラムの構文を解析し, その結果による内部データと知識ベースのデータとのチェックを行えばよい. だが, 品質の保証性, コーディングの量等を考えると, 推論部を全部GSAでまかなうのは困難である. そこで, 推論部をプログラムの構文を解析する部分とそれ以外の部分に分け, GSAで記述するのはその特徴を活かして構文を解析する部分 (以下, 構文解析と呼ぶ) だけにとどめた.

構文解析では, 知識ベースのデータパターンに対応する内部データを先に洗い出して解析・加工し, 表3に示す数種類のファイルに編集した. 構文解析以外の部分は, ファイル処理に優れたCOBOLで作成した.

表3 推論部における各種編集ファイル
Table 3 Files in reasoning part

ファイル情報ファイル	COBOL プログラムの SELECT 句および FD 句情報を編集. 内部ファイルを主キー，外部ファイル名を副キーとする MSAM ファイル主キー項目名・副キー項目名・レコード長・LINAGE 情報…を持つ.
レコード情報ファイル	COBOL プログラムの DATA DIVISION 情報を編集. データ名を副キー，そのデータの出現するライン番号を主キーとする MSAM ファイル. レベル・USAGE・桁数・そのデータを使っている命令語情報…を持つ.
命令語出現ファイル	COBOL プログラムの命令語情報を編集. 命令語コード(COBOL 命令語に番号をつけたもの)をキーとする MSAM ファイル. 各命令語がどのラインに出現するかの情報を持つ.
ラベル対応ファイル	各ラベルによるパラグラフの開始ラインと終了ラインを持つ.
PERFORM 飛先ファイル	PERFORM 命令の飛先ラベル名を持つ.
ロジックファイル	各ラベルがどこから呼ばれているかの情報を持つファイル.
加工ファイル	COBOL プログラムの記述の個人差を取って編集. ・複数のスペースは一つにする. ・複数の記述方法が許されているものは省略形に変える. PICTURE → PIC ZEROES → ZERO ・省略可能なものは省略する. ASSIGN TO → ASSIGN ・1 ラインに命令語は一つにする. MOVE A TO B C→MOVE A TO B MOVE A TO C MOVE A TO B →MOVE A TO B

3.2 機能の限界に対して

従来ツールの機能の限界に対しては，プログラムの流れの情報によって修正内容が変わる修正項目のチェックができるように，プログラムの流れを追うことができる機能を加えた.

まず，検収するプログラムの構文解析時に，プログラムの流れを追うのに不可欠な PERFORM 文・GO 文・EXIT 文・IF 文・ラベルについての各種ライン情報をファイル（以下，ロジックファイルと呼ぶ）に蓄えておく．次にツール作動時に，あるラインからの流れを追う要求（$AFTER, $BEFORE）が知識部から出されるごとに，そのラインを起点とした流れの情報を詰め込んだテーブル（以下，ロジックテーブルと呼ぶ）をロジックファイル情報より作成する．さらに，推論部はロジックテーブルの情報を読み込んで，処理するプログラムの流れを 1 ラインずつ追う，このような仕組みで，プログラムの流れを追う機能を実現している．流れの方向は，開始から終了へ($AFTER) と，終了から開始へ ($BEFORE) の両方向を用意した.

ロジックテーブル（図 11）が作り上げられるまでの $AFTER の具体的な動きを，図 9 のソースプログラムを使って図 10 に示す．この例では，手修正前プログラムの 96000 ライン目(以下，下 3 桁は取って記述する)からプログラムの流れを追う要求が

```
00092000 00 00125    PROCEDURE DIVISION.
00093000 00 00126    START-1100.
00094000 00 00127    DISPLAY '** TESTPRG ** START **'
00095000 00 00128    PERFORM SRINI1 THRU EXINI1
00096000 00 00129    OPEN OUTPUT SRV-PRG
00097000 00 00130    OPEN OUTPUT SRV-JYO
00098000 00 00131    OPEN I-O SYUT-F
00099000 00 00132    PERFORM SRGET1 THRU EXGET1
00100000 00 00133    IF NKEY = HIGH-VALUE
00101000 00 00134    GO ZEND .
00102000 00 00135    Z1 .
00103000 00 00136    PERFORM SRPRC1 THRU EXPRC1
00104000 00 00137    PERFORM SRGET1 THRU EXGET1
00105000 00 00138    IF NKEY = HIGH-VALUE
00106000 00 00139    NEXT SENTENCE
00107000 00 00140    ELSE
00108000 00 00141    GO Z1 .
00109000 00 00142    DISPLAY '** TESTPRG ** END **'
00110000 00 00143    ZEND .
00111000 00 00144    CLOSE SYUT-F
00112000 00 00145    CLOSE SRV-PRG
00113000 00 00146    CLOSE SRV-JYO
00114000 00 00147    STOP RUN
00115000 00 00154    SRINI1 .
00116000 00 00155    ACCEPT CAWK1
00117000 00 00156    IF OPTWK = SPACE
00118000 00 00157    MOVE 'ON' TO OPT-A .
00119000 00 00158    EXINI1 .
00120000 00 00159    EXIT
00121000 00 00163    SRGET1 .
00122000 00 00164    READ SYUT-F NEXT
00123000 00 00165    END
00124000 00 00166    MOVE HIGH-VALUE TO NKEY
00125000 00 00167    GO EXGET1 .
00126000 00 00168    MOVE SYUT-KEY TO NKEY
00127000 00 00169    EXGET1 .
00128000 00 00170    EXIT
00129000 00 00174    SRPRC1 .
00130000 00 00175    MOVE SYUT-KEY TO W-SRV-P-JYO
00131000 00 00175    MOVE SYUT-KEY TO W-SRV-J-JYO
00132000 00 00176    IPF OPT-A = 'ON'
00133000 00 00177    PERFORM SRPUT1 THRU EXPUT1
00134000 00 00178    IPF OPT-B = 'ON'
00135000 00 00179    PERFORM SRPUT2 THRU EXPUT2
00136000 00 00180    EXPRC1
00137000 00 00181    EXIT
00138000 00 00185    SRPUT1 .
00139000 00 00186    WRITE SRV-P-REC FROM W-SRV-P
00140000 00 00187    INVALID
00141000 00 00188    DISPLAY '** SRV-PRG FILE INVALID !!'
00142000 00 00189    CALL 'CANCEL'
00143000 00 00190    EXPUT1 .
00144000 00 00191    EXIT
00145000 00 00195    SRPUT2 .
00146000 00 00196    WRITE SRV-J-REC FROM W-SRV-J
00147000 00 00197    INVALID
00148000 00 00198    DISPLAY '** SRV-JYO FILE INVALID !!'
00149000 00 00199    CLOSE SYUT-F
00150000 00 00200    CLOSE SRV-PRG
00151000 00 00201    CLOSE SRV-JYO
00152000 00 00202    CALL 'CANCEL'
00153000 00 00203    EXPUT2 .
00154000 00 00204    EXIT
```

図9　加工ファイル中の手修正前ソースプログラム例

Fig. 9　Program without amendment in Kako-file

図10　ロジックテーブル作成の過程

Fig.10　Process of making logic-table

出たものとする．プログラムの流れの解析は図10に示す順に行われる．図10の縦横棒は，実際のプログラムの流れを視覚的に訴えられるように使用している．すなわち，縦棒はプログラムの流れを，横棒はプログラムの大まかな区切りを，分岐はプログラムの流れが複数に分かれることを示す．96ライン目から99ライン目までは，図9に記述されている順とプログラムの流れは等しいので，ロジックテーブルの開始ライン・終了ラインにそれらをセットする．99ライン目はPERFORM文なので，ここから先のプログラムの流れは図9に記述されている順と異なる．実際のプログラムの流れは，このPERFORM文で実行される段落を展開したものに等しい．そこで，ロジックファイルからこの段落の開始ライン・終了ライン情報を引き出し，ロジックテーブルにセットする．ここで使われている「*)のケース」という記述は，

P)　パラグラフが，PERFORM文で実行されるケース

F)　パラグラフが，プログラムの上からの流れで実行されるケース

G)　パラグラフが，GO文で制御が移って実行されるケース

の各ケースを表す．これより，上記の3ケースを考慮しながら，プログラムの終了までプログラムの流れを解析し，それにそってロジックテーブルの開始ライン・終了ラインをセットしていく．GO文によるループ等によって展開するラインに重複のある場合は，2回目以降を展開しない．図10では，「展開する→しない」という記述で表している．

このような解析の結果，作成されるのが図11のロジックテーブルである．推論部は，

```
                                        開始ﾗｲﾝ   終了ﾗｲﾝ   現在ﾗｲﾝ
0 00000 00001 00000 00000 00000 00000 00097000 00099000 00097000
0 00000 00001 00001 00000 00000 00000 00121000 00125000 00121000
0 00000 00001 00002 00000 00000 00000 00126000 00128000 00126000
0 00000 00001 00003 00000 00000 00000 00127000 00128000 00128000
0 00000 00001 00000 00000 00000 00001 00100000 00101000 00100000
0 00000 00002 00000 00000 00000 00000 00110000 00114000 00110000
0 00000 00003 00000 00000 00000 00000 00102000 00103000 00102000
0 00000 00003 00001 00000 00000 00000 00129000 00133000 00129000
0 00000 00003 00001 00001 00000 00000 00138000 00144000 00138000
0 00000 00003 00001 00000 00000 00001 00134000 00135000 00134000
0 00000 00003 00001 00001 00000 00001 00145000 00154000 00145000
0 00000 00003 00001 00000 00000 00002 00136000 00137000 00136000
0 00000 00003 00000 00000 00000 00001 00104000 00104000 00104000
0 00000 00003 00000 00000 00000 00002 00105000 00106000 00105000
0 00000 00004 00000 00000 00000 00000 00109000 00109000 00109000
9 99999 99999 99999 99999 99999 99999 99999999 99999999 99999999
```

図11　ロジックテーブルのダンプリスト

Fig.11　Dump list of logic-table

ロジックテーブルの開始ラインから終了ラインの間のラインを順に見ていくだけで，プログラムの流れがつかめる．すなわち，初めに開始ラインを得て，順に1ラインずつ増やしていき，ライン番号が終了ラインと等しくなれば次の開始ラインを得る，ということを繰り返すことで，推論部は実際のプログラムの流れにそった処理を行うことができる．ライン番号が 99999999 はテーブルの終了を意味する．

4. 検証ツールの構成

検証ツールの構成を図12に示す．知識部では，移行情報を元に LAKET で記述された知識 DB ソースから，推論部の入力となる知識 DB(KALET コンパイラで作成された検証ツールのオブジェクトコード）を作成する．検証ツールの実際の動きは，この知識 DB の一つ一つの内容とそれらの組み合わせによって決まる．

一方，推論部は四つの工程から成る．

1) 手修正前・後ソースプログラムを解析する．ここでは対象プログラムのプログラム内のさまざまな情報を解析し，次の工程で処理しやすい形に編集する．また，手修正前・後ソースプログラムは，表3の加工ファイルのルールに従って加工される．

2) 加工した手修正前プログラムの内容と手修正項目の条件（知識 DB のデータパターン）が合致すると，加工した手修正前プログラムに行動をおこし，正解ファイルを作成する．正解ファイルの中のプログラムは，検証ツールが作る正しい修正の入ったプログラムである．

3) 正解ファイルのプログラムと加工した手修正後のプログラムを突き合わせて異なる箇所を見つけ出し，その対応状況を集約した対応ファイルを作成する．

4) 対応ファイルをもとに，どのラインがどのように違っているか等の情報をまとめ，最終的にチェックリストを出力する．

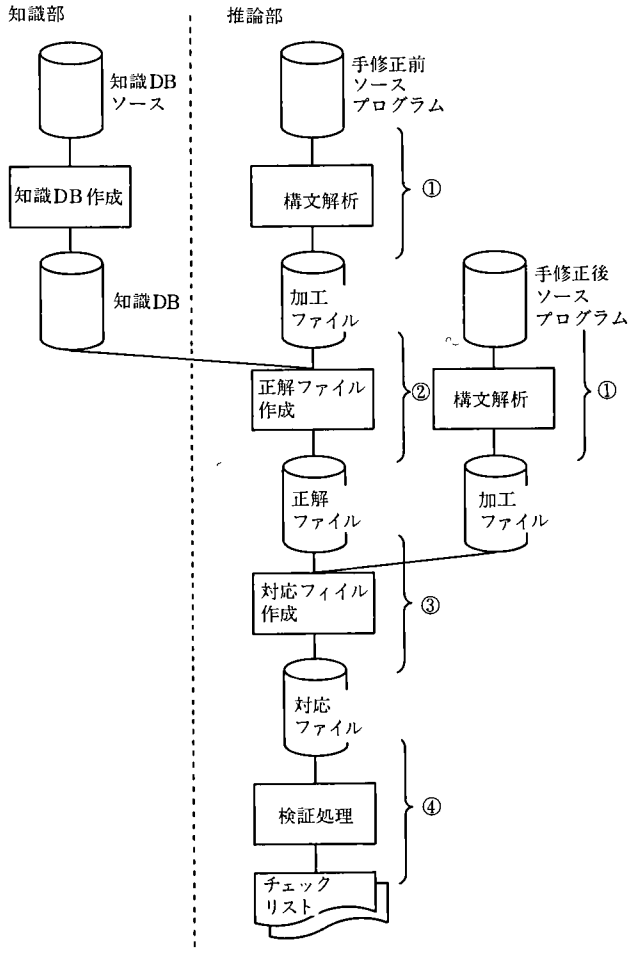


図12　検証ツール構成図

Fig.12　Configuration of kensyotool

5. 検証ツールの用途

移行作業の実施工程を従来と今回とで比較したものを図13に示す.

従来のプログラム移行は,図13の左側の流れで行っていた.既存の文法変換ツールでは完全な文法変換はできないため手修正が必要であり,手修正情報の資料を一つ一つ作るうえ,検収用のコンパイルリストを大量に出力する.プロジェクトのリーダは資料を手に大量リストを目で追ってチェックする.

今回検証ツールを取り入れてからのプログラム移行は図13の右側の流れになる.手修正項目をまとめた資料の山と大量コンパイルリストの出力,および目チェックがなくなる.プロジェクトのリーダは知識DBを作成し,修正者は検証ツールが出力するチェックリストをもとに手修正を行う.チェックリストには修正情報が簡潔に編集されているので,検収の状況を一目で把握することができる.

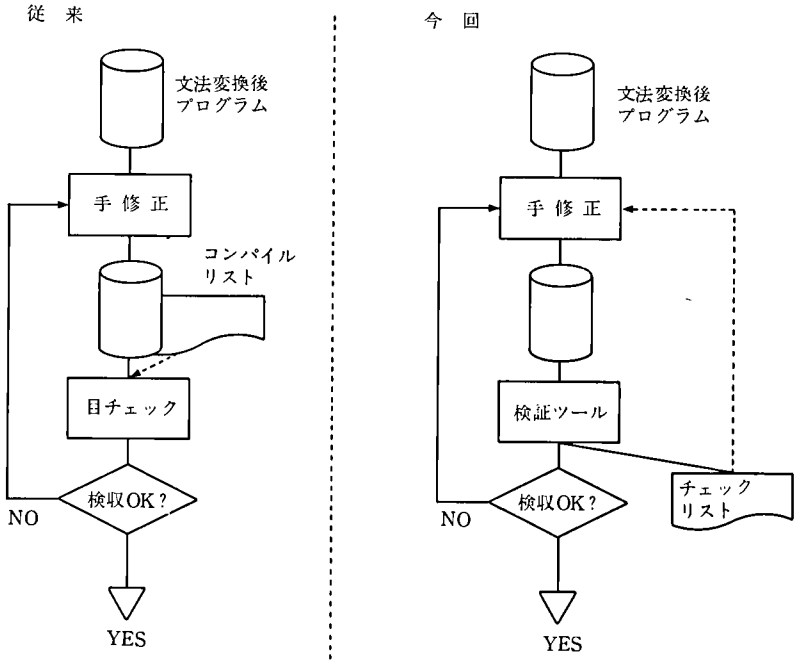


図 13　実施工程の比較図

Fig.13　Comparison between new methods

6. 検証ツールの使用例

実行に必要なものは，手修正前プログラムのファイル，手修正後プログラムのファイル，知識 DB ファイルの三つである．修正項目は図 14 に示すものとする．この時の手修正前・手修正後のプログラムが図 15 に示す状態だとすると，図 16 に示すリストが出力される．検収者は「検収者用チェックリスト」を見て，「知識 DB J 001 について，修正すべき箇所が 3 箇所あったが，全部は正しく修正されていなかった」ことを知る．修正者は「修正者用チェックリスト」を見て，

1)　200 ライン目の修正すべきでない箇所に修正がなされている，もしくは，知識 DB にない修正を行っている，

2)　250 ライン目の修正すべき箇所の修正内容がまちがっている，

3)　300 ライン目の修正すべき箇所に修正がなされていない，

ことを知る．その後，チェックリストのメッセージに従って，誤った手修正・漏れた手修正等を修正する．

7. 評 価 と 課 題

2.1 節で述べた問題に対処できるように AI の考えを取り入れて作ったことで検証ツールそのものの個別改造はなくなり，個別対応は各プロジェクトに合わせた知識 DB の組み合わせと新たな知識 DB の作成だけで済むようになった．この知識 DB の組み合わせはまったく自由であり，組み合わせに関して処理結果を気にする必要はまったくない．また，知識 DB の組み合わせや知識 DB の内容がどう変わろうとも，検

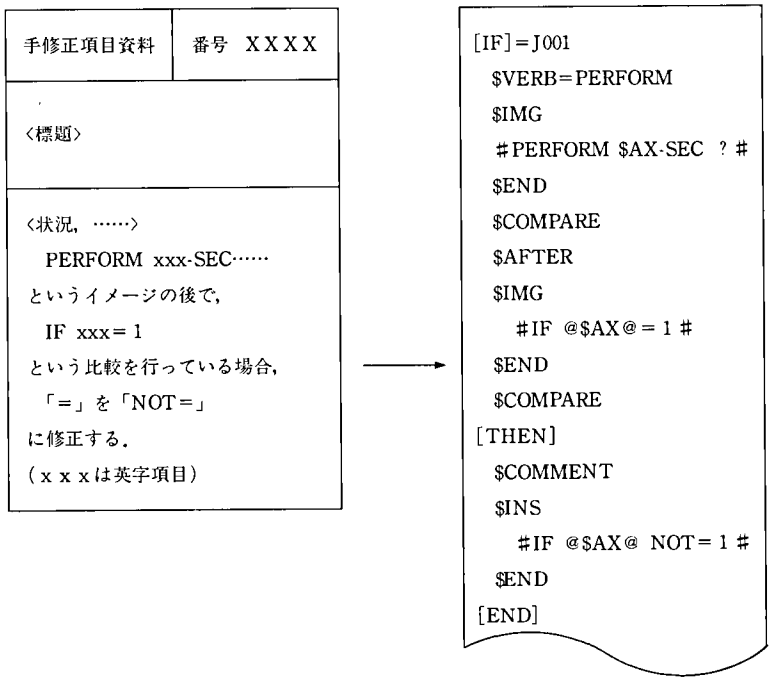


図14 移行作業の修正項目具体例

Fig.14 Example of amendment item

証ツールの本体に手を加えることは一切ない．この結果，個別対応作業は特定の担当者以外でも楽に行えるようになり，また作業に費やす時間も大幅に短縮されるようになった．

LAKETの記述で新規の知識DBを作ることができるといっても，中には記述できない手修正項目もあるので，ツールの個別対応作業をユーザに完全にまかせるには至っていない．ただし，表現できない項目があれば新コマンドを作成して対処している．表現できてもわかりにくい複雑な手修正項目をもっとわかりやすく表現できるようにしていくことが，これからの課題である．

AIの考えをCOBOLで実現するために，推論部を構文解析部分とそれ以外に分け，さまざまな情報をファイルに蓄え処理しやすい形にした手法は，開発工数面から考えて妥当だったと思われる．しかし，当初から予想はされていたことだが，処理時間は非常にかかるものとなった．今後はコーディングの見直しを図って，処理時間を短縮する必要があろう．

2.2節で述べた問題に対処できるように作ったプログラムの流れを追う機能については，ロジックテーブルを使った前述のしくみで実現できた．しかし，この処理を行うことによって極端に処理時間がかかる，ロジックテーブルの大きさが記憶領域の大きさに制限されるという問題が残っており，これらの解決が今後の課題である．

検証ツールの今後の可能性を次のように考えてみた．

1) 文法変換ツールにまで改良すれば，手修正作業・検証作業がなくなる．

2) 推論部はプログラムを解析・検証するので，システム移行だけではなく，日常

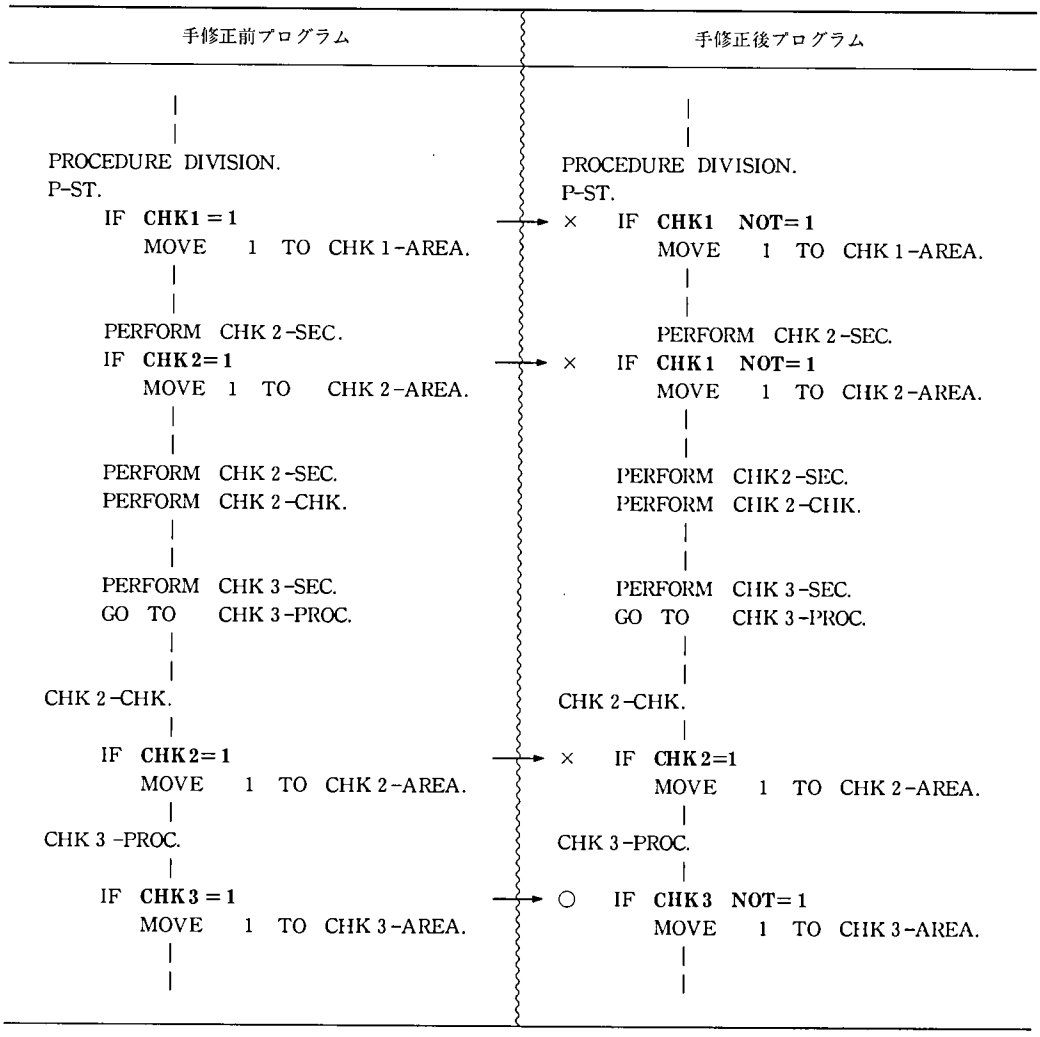


図 15　検証ツールの検収対象プログラムの修正例

Fig.15　Program without amendment and program with amendment

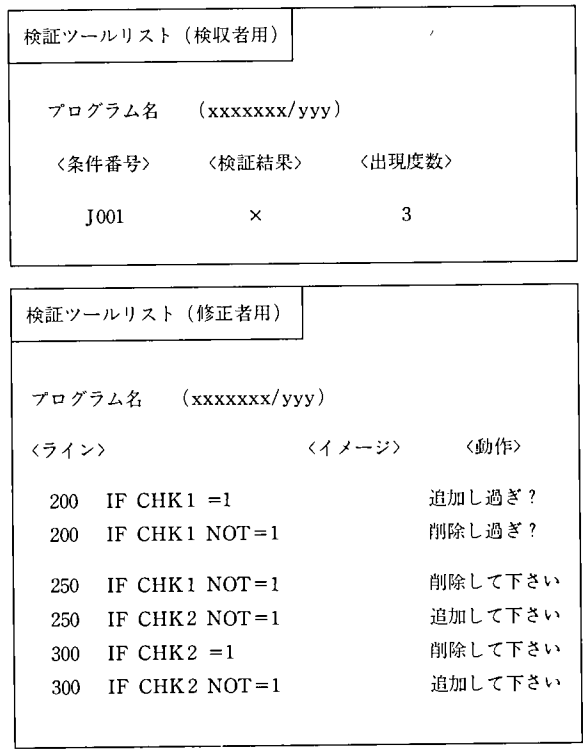
のプログラム保守作業・調査に応用できる.

3)　文法解析機能やプログラムの流れを追う機能だけを取り出せば，デバッグ時のプログラム解析に応用できる.

4)　現在の文法解析部分は ASCII COBOL 解析しかできないが，この部分を他言語の解析ができるものに代えれば，COBOL 以外の言語の移行作業にも使える.

8. お わ り に

特別な推論を必要としないシステムに AI のシステム構築方法を取り入れたこと自体は，とくに目新しいことではない．しかし環境的な制限からいわゆる AI 用言語・AI 用マシンを用いないで，COBOL 等の一般的な言語を用いて AI の概念のみを引き出したシステム構築は，新しい方法といえる．また，この方法によって，検証ツール(今


検証ツールリスト（検収者用）

プログラム名　（xxxxxxx/yyy）

〈条件番号〉　〈検証結果〉　〈出現度数〉

J001　×　3

検証ツールリスト（修正者用）

プログラム名　（xxxxxxx/yyy）

〈ライン〉　〈イメージ〉　〈動作〉

200　IF CHK1 =1　追加し過ぎ？

200　IF CHK1 NOT=1　削除し過ぎ？

250　IF CHK1 NOT=1　削除して下さい

250　IF CHK2 NOT=1　追加して下さい

300　IF CHK2 =1　削除して下さい

300　IF CHK2 NOT=1　追加して下さい

図16　検証ツールの出力リスト例

Fig.16　Kensyotool's output list

回新たに作成した移行ツール）に対する個別改造はなくなり，個別対応は知識DBの簡単な組み合わせだけで済むようになった．この利点は，一般的なツールにも大いに活かせるであろう．

検証ツールの用途は検収作業だけにとどまらない．出力リストは知識DBに従った修正状況も指示するので，検収作業前に検証ツールを実行すれば，手修正作業の大きな助けとなる．また，おおまかな知識DBを作成して準備工程で実行すれば，移行するシステムの傾向調査等に用いることもできる．

本来生産性を高める道具にすぎない移行ツール（移行用ツールの集合体：従来ツール＋検証ツール）に，保守・個別改造等の作業に工数がかかりすぎているという現状は見直されるべきであろう．本稿では，個別対応作業の容易化という目標に対して，従来の仕様とは異なったAIの考えを取り入れるという試みを紹介したが，これは解決策の一手段にすぎない．たとえば，上で述べたように，入力データを変えることで同じツールがさまざまな場面で使えるということは，複数のツールの個別対応を一つで済ますことになり，個別対応の容易化につながる．今後もこれらの試みに加え，用途の汎用性や使用法の統一性を目指したツール作りを行い，道具にかかりすぎている工数を減らしていきたい．

参考文献　［1］ OHP 編集部編，"エキスパートシステムの実務" オーム社，1987.
［2］ 玉木他 6 名，"プログラム変換"，共立出版，1987.
［3］ 白井・述井，"岩波講座『情報科学』"，Vol.22，岩波書店，1982.
［4］ "OS 1100 PLUS PRM"，Unisys Corporation.
［5］ GAS "1100 LEVEL 4 R 1 REFERENCE"，Unisys Corporation.

執筆者紹介　井　上　博　行　(Hiroyuki Inoue)
昭和 38 年生．61 年神戸大学文学部心理学科卒業．同年日本ユニシス (株) 入社．システム移行作業，移行ツール開発に従事．現在システム移行技術部所属．

Fortran-to-Cobol トランスレータの開発

The Development of a Fortran-to-Cobol Translator

瓜 生 和 史

要 約 Unisys シリーズ 2200•1100 の初期の頃からのユーザには，FORTRAN で事務計算システムを開発してきたところが少なくない．しかしながら今日では，事務計算分野は COBOL で記述することが当たり前のようになっている．

FORTRAN で書かれた既存のシステムを，なるべく少ない負荷で COBOL のシステムに変換できないか，というニーズが出現するのも当然であろう．今回ある大手ユーザ（A 社）より，FORTRAN プログラムを COBOL プログラムに自動的に書き換える言語変換ツールの提供を求められた．

当社ではこの新しい試みに対し，AI マシンおよび AI 技法を活用し，「FTC (Fortran-to-Cobol トランスレータ)」と呼ぶ言語変換ツールを KS-303 (Explorer II) 上の LISP 言語で開発した．また，あわせてシリーズ 2200•1100 と KS-303 が自動的に連携して変換作業を進める運用システムを構築し提供した．

この結果，A 社では現在までに約 840 本（変換前の FORTRAN ソース・ステップ数で約 172,000 ステップ）の大量変換を実現している．

本稿では，今回開発•提供された FTC トランスレータの概要およびそのシステムの運用について述べるとともに，客先での変換実作業を支援した立場から，変換実績の評価，今後の課題について言及する．

Abstract Among early Unisys Series 2200/1100 computer users, it is not a very few who have counted on the FORTRAN language for their development of business data processing systems. Nowadays, however, the use of the COBOL language has obviously become very common for business data processing applications, naturally causing user needs to emerge for a practical systems conversion at the least possible cost from FORTRAN-written programs to COBOL-based versions.

In response to a certain key user's recent request for Nihon Unisys's new offer of a language conversion tool which helps to automatically change FORTRAN programs into COBOL ones, a new challenge to Nihon Unisys, we have developed a language conversion tool called the "Fortran-to-Cobol (FTC) translator" with the use of the ISP language on the KS-303 (Explorer II) through the adoption of the AI method on AI equipment. And for the user's benefit, we have also built an operational system which enables the Series 2200/1100 and KS-303 to work together automatically to proceed with conversion processes.

Its use by the customer up to the present has resulted in the successful mass-conversion of nearly 840 programs (which compare in volume with some 172,000 program steps in terms of pre-converted FORTRAN sources).

Besides sketching the FTC translator now available for customer user and its operatations, this paper also refers to the evaluation of the conversion quality and other problems which remain yet to be solved in the light of our actual support for the customer program conversion.

1. はじめに

一般のユーザ・システムの開発において，プログラムを記述するために使用される言語は技術計算分野がFORTRAN，事務計算分野はCOBOLが主流になっている．しかしながら当社のシリーズ2200・1100の初期の頃からの大手オンライン・ユーザには事務計算にCOBOLではなくFORTRANを使用しているところがあり，しかも極めてシステムの規模が大きい（数百万ステップを越える）．

これらのユーザでは，現状のままでは絶えず業務に合わせて新システムの開発や機能の変更，追加や削除等，システムの保守をFORTRANで記述し続けていかなければならない．しかしながら，事務計算のための一般的なソフトウェア環境やシステムの開発要員の確保が今日ではCOBOLの方が容易であること等を考慮すると，将来的にユーザにとってシステムを維持していく上で，システムを記述するのに今まで通りFORTRANで行うか，新たにCOBOLにするかを検討する必要がある．

今回当社のユーザであるA社から現行システムをCOBOL中心のシステムに変更するため，FORTRANで記述したシステムをCOBOLに自動変換するトランスレータの開発を依頼された．AIマシンで変換を実現することになり，「FTC(Fortran-to-Cobolトランスレータ)」と呼ぶ言語変換ツールをKS-303（Explorer II）上のLISP言語で開発し，リリースを行った．

また，併せてシリーズ2200•1100とKS-303が自動的に連携して，以下に示す変換作業を進める運用システムを構築して提供した．このトランスレータ，運用システム，周辺ソフトウェアを総称してFTCシステムと呼ぶ．

① 変換対象

客先で保有している約2万本のFORTRAN（ASCII FORTRAN）プログラムのうち，まずバッチ系の先行変換部分を対象としている．

② 変換実績

当初，約100本の試行変換を行いその後，試行分の再確認を含み約840本の変換を実施している．(FORTRANプログラム　約172,000ステップ→ASCII COBOLまたは日本語COBOLプログラム約340,000ステップ)

2. FTCシステムの概要

今回客先に提供したFTCシステムは，シリーズ2200・1100のFORTRANプログラムからCOBOLプログラムへソースコード変換を行うトランスレータとそれを支援する周辺ソフトウェアから構成され，シリーズ2200•1100とKS-303を組み合わせることにより作動する．

FTCシステムを構成するソフトウェアは次の通りである．

1) KS-303上で作動するソフトウェア

- FTCトランスレータ……FORTRANプログラムからCOBOLプログラムへコード変換を行うFTCシステムの中核ソフトウェアである．内容については3章で述べる．
- ルール・コンパイラ……FTCトランスレータを実行するのに必要な付加情報を解析して，知識ベースを作成する．内容については4章で述べる．

・Explorer Kermit……シリーズ 2200・1100 とのファイル転送を行う.

・FTC daemon……FTCトランスレータとルール・コンパイラの実行管理を行う.

・1100 daemon……Explorer Kermit を使ってシリーズ 2200・1100 とのファイル転送を管理する.

2) シリーズ 2200・1100 上で作動するソフトウェア

・シリーズ 2200・1100 Kermit……KS-303 とのファイル転送を行う.

・Pickup……変換を実施するプログラムの変換情報を生成する.この変換情報は変換するプログラム・ソースや 3 章で述べる付加情報を内容とするものである.

Explorer Kermit とシリーズ 2200・1100 Kermit は当社が提供する標準ソフトウェアである.

FTC システムで使用するソフトウェア, ハードウェアの関連は図 1, 今回の客先における変換の流れ (実現形態) は図 2 の通りである.

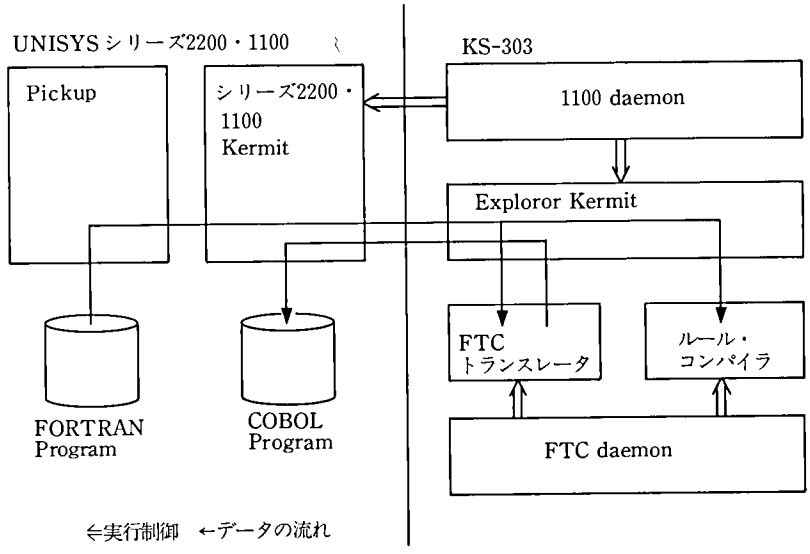


図 1 FTC システム概要

Fig. 1 The outline of FTC-system

3. FTC トランスレータ

3.1 FTC トランスレータの特徴

当社ではこれまでも, 同一の言語であればプログラムを異機種間でコンバージョンする技術(たとえば他社機種から当社機種へ)は確立している. しかし FORTRAN から COBOL への変換を考えた場合には, 言語機能間の相違のため単純に変換できないむずかしい部分もあり, また単純に変換したのでは実行可能かもしれないが, 機械的なコードの羅列となってしまい, 今後の保守に耐えうるものかどうか問題となる. FTC トランスレータでは変換を実現するためにエキスパート・システムの発想を採り入れたものとなっている.

3.1.1 個別変換内容の知識ベース化

従来の異機種間トランスレータの多くは, 標準版にはないユーザ固有の変換内容が

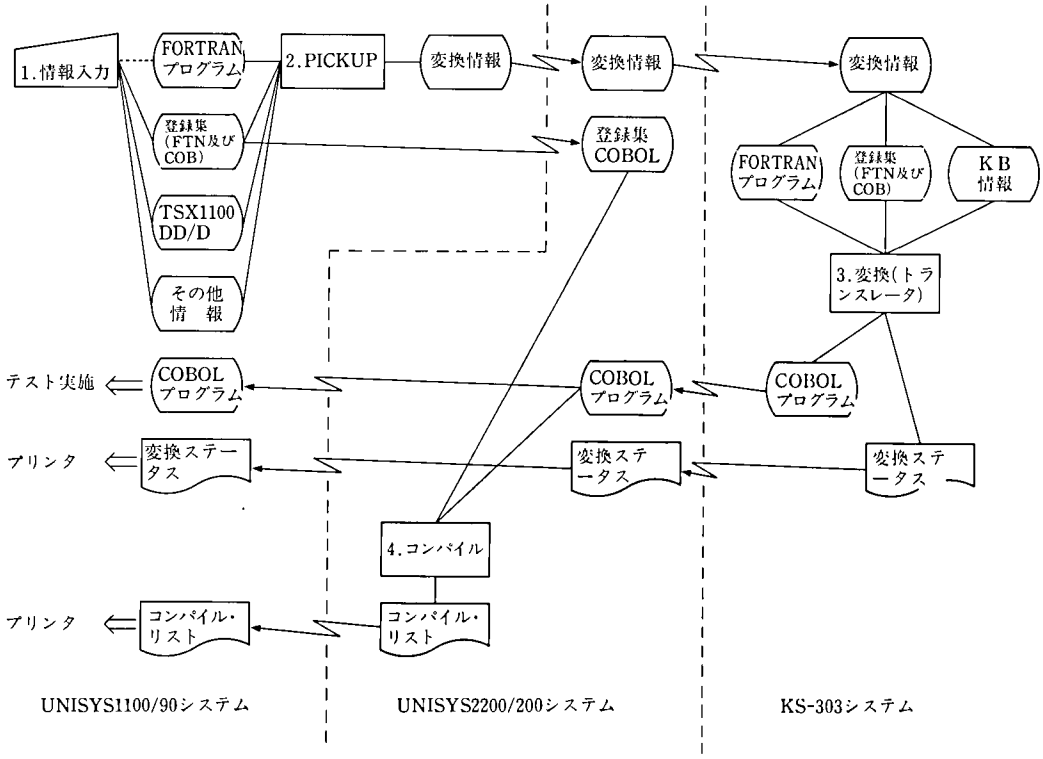


図 2　客先の変換の流れ

Fig. 2　The flow of user's translation

発生した場合，そのユーザ用にトランスレータを改造して個別改造版トランスレータを作成している．これでは次のような問題が発生してしまう．

- 個別改造版が多数発生
- 改造によるバグの発生
- 改造のための要員や時間が必要
- 標準版に保守（機能拡張やバグ対応）が入っても個別改造版に反映させることはできない

これらの問題を解決するために FTC トランスレータでは個別改造版を作成しないで，もしユーザ固有の変換内容があった場合には個別の変換を実行する内容を知識ベースとして自由に定義する．

3.1.2　生成コードの COBOL らしさの追求

FORTRAN ではファイル・レコードや副プログラムのレイアウトを表す場合に配列とすることが多く，この配列内のあるフィールドを参照する場合は配列要素で識別する．単純にそのまま変換したのではレイアウトを機械的に生成しなければならず，画一的なものができてしまい見づらいものとなってしまう．しかし COBOL では通常レイアウトを詳細に定義しデータ項目名で認識する．この COBOL のデータ記述の利点を活かすために，変換時に配列で表現したレイアウトに COBOL のデータ記述のレイアウトを与えることにより，配列位置で表現した配列内の各フィールドを対応する COBOL のデータ項目の名前に置き換える．

たとえば，次の記述があった場合

```
      INTEGER  I(2)
      J=BITS (I(1), 1, 18)
```

これをこのままCOBOLに変換すると

```
 01  I.
   03  I-1  OCCURS  2  TIMES.
       05  I-1-1  PIC  S1(36).
       05  I-1-2  REDEFINES  I-1-1.
       07  I-1-2-1  PIC  1(18).
       07  FILLER  PIC  X(2).
 01  J  PIC  S1(36).
       MOVE  I-1-2-1  (1)  TO  J.
```

となり，配列Iのデータ記述は複雑なものになってしまう．

もしこの配列Iに次のCOBOLのデータ記述を与えたとすると，

```
 01  I.
     03  RIRITU    PIC  1(18).
     03  ZANDAKA PIC  1(18).
     03  FILLER    PIC  X(4).
```

BITS (I(1), 1, 18)に当たるデータ項目を与えたデータ記述から決定することができ，生成するコードは

```
      MOVE  RIRITU  TO  J.
```

とすることができる．

このように与えられたデータ記述の対応するデータ項目を使ってCOBOLらしいプログラムを生成する．この生成コードのCOBOL項目名化は変換に必須ではないが，FTCトランスレータはそのまま言語変換するのではなく，COBOLの持つ言語機能の特徴を活かすコード生成を行って変換の質を上げる．

3.1.3 変換情報の抽出

ある文だけでは変換するための情報が不足し，他の文を見なければ変換できない場合，その他の文を抽出して情報を明確化しなければならない．たとえば，次のFORTRANプログラムがあり

```
      INTEGER  I(10)
      DO  20  K=2,5
                J=J+I(K)
  20  CONTINUE
```

配列Iに次のデータ記述を与えたとすると，

```
 01  I.
     03  TOTAL      PIC  S1(36).
     03  ZANDAKA   OCCURS  9  TIMES
                    PIC  S1(36).
```

この場合，単純にCOBOLコードを生成しようとしても配列要素I(K)が可変位置の配列要素のためデータ項目名を選択できない．しかし添え字Kはプログラムの制御

の流れを遡ってKの採り得る値をすべて調べると，Kの値は2から5までの値を採ることがわかる．レイアウトの中で2から5まで可変位置のデータ項目を捜すとZANDAKAを選択することができる．このため生成するプログラムは次のようになる．

```
        MOVE  2  TO  K.
    LOOP.
        IF  K >  5  THEN  GO  TO  SKIP.
        MOVE  K  TO  temp.
        SUBTRACT  1  FROM  temp.
        ADD  ZANDAKA  (temp)  TO  J.
        ADD  1  TO  K.
        GO  TO  LOOP.
    SKIP.
```

このようにCOBOLレイアウトが与えられた場合，ある文の変換情報を他の文から持ってくる．

3.2 付加情報

個別変換内容の外部定義化やCOBOLレイアウトの指定等，ユーザがFTCトランスレータに与える各種の付加情報はすべて知識ベース化して使用される．付加情報には以下のものがある．

- レイアウトに関する情報
- ルール
- その他の情報

3.2.1 レイアウトに関する情報

レイアウトに関する情報は生成コードのCOBOL化で使用するレコード・レイアウトとそのレイアウトを使用するファイル名や副プログラム名等である．

3.2.2 ルール

ルールは個別変換内容をルール命令を使用して指示する．

ルール命令は表1の通りである．

表1　ルール命令
Table 1　Rule command

ルール命令	機能
FTN	ルール対象の文のパターンを設定
FIND	内部テーブルの参照
REFER	内部テーブルの参照
CREATE	内部テーブルに値を格納
STORE	内部テーブルに値を格納
GENERATE	COBOL文を生成
DELETE	FORTRAN文を変換対象から除く
REPLACE	FORTRAN文をCOBOL文に置換
MODIFY	FORTRAN文をFORTRAN文に置換
IF	条件によって適用ルールの内容を変更
LISP関数呼び出し	ルールの中でLISP関数を直接呼ぶ
代入文	ルール変数に値を代入する

ルールには次のものがある.

1) 情報収集ルール……情報収集ルールはFORTRAN文から変換に必要な情報をFTCトランスレータの内部テーブルに抽出する方法を指示する. FTN文で情報収集の対象となるFORTRAN文を見つけ, CREATE文やSTORE文で情報を抽出する.

2) 準備ルール……準備ルールは, もとのFORTRAN文を変換しやすいFORTRAN文に書き換える方法を指示する. FTN文で書き換えの対象となるFORTRAN文を見つけ, MODIFY文で新しいFORTRAN文に書き換えたり, DELETE文で不要な文を変換対象から外す.

3) 変換ルール……変換ルールは, FORTRAN文からCOBOL文への書き換えを指示する. FTN文で書き換えの対象となるFORTRAN文を見つけ, REPLACE文でシステムによる変換を抑制して指定するCOBOL文に書き換える.

4) 出力ルール……出力ルールは, FORTRAN文には無いようなCOBOL文の生成を直接指示する. GENERATE文で挿入位置を示すセクションにCOBOL文を生成する. セクションにはファイル管理記述項・ファイル記述項・初期化実行コード・終了時実行コードがある.

3.2.3 その他の情報

その他の情報には副プログラム名の書き換え・ルール関数・LISP関数がある.

1) 副プログラム名の書き換えはFORTRANとCOBOLの両方のプログラムから同一の副プログラムを呼び出すことができないため, 両者にそれぞれ別の副プログラムを用意する場合に指示する.

2) ルール関数はマルチ・レイアウトの場合, どのレイアウトを選択するかの方法を記述する.

3) LISP関数はルールの中から呼び出すLISP関数をユーザ自身で記述できる. ルールで記述しきれない変換内容がある場合には, ルールの中から直接ユーザが記述したLISP関数を呼び出して変換を実行する.

3.3 FTCトランスレータの変換処理

FTCトランスレータの変換は, 次のような流れで実現している.

FORTRANソース・プログラム

構文解析	FORTRANプログラムの構文解析を行い, 内部表現に変換する.
‖	
名前付替	FORTRANプログラムに現れたサブルーチン名等を変更する.
↓	
FLOW ANALYSIS	ループの検出, 変数の取り得る値の情報を調べる.
↓	
情報収集ルール実行	情報収集ルールを実行する.
↓	
準備ルール実行	準備ルールを実行する.
↓	
FLOW ANALYSIS	準備ルールでFORTRAN文が書き換えられた場合に, 再度

FLOW ANALYSIS をやり直す．準備ルールの実行がない場合には実行されない．

↓

レコード・レイアウト解析　① 情報収集ルールで得られたファイルのレコード・レイアウトおよび手続きの引き数のレイアウトを変数に付与する．
② 各変数について参照フィールドがレイアウトのどの項目に一致するか調べる．

↓

変換ルール実行　FORTRAN 文を直接ルールに基づいて COBOL 文に書き換える．

↓

COBOL 文出力ルール実行　直接ルールに基づいて COBOL 文を生成する．

↓

COBOL 文出力　COBOL 文をファイルに出力する．

↓

COBOL ソース・プログラム

3.4 変　換　例

簡単なプログラムを変換して付加情報の記述例や付加情報を使用すると，どのように変換するか見てみよう．

図 3 のプログラムを変換するのに当たり，これに次の付加情報を与える．

1) 副プログラム名 FTNSUB を COBSUB に置き換える（図 4）．

2) 副プログラム COBSUB を CALL している文を探し STA で識別する．このとき引数は任意であり，？？AIPCT で識別する．STA からプログラムを遡って，STB にマッチする文を捜し，?INP 1 にマッチする変数を抽出する．

```
PROGRAM         TEST
INTEGER         PCT ( 10 )
BITS ( PCT , 19 , 18 ) = LOC ( IBUF )
CALL FTNSUB ( PCT )
RETURN
END
```

図 3　FORTRAN プログラムの例
Fig. 3　Example of Fortran program

```
KB "RULES"

RENAME
SUBPROGRAMS

( "FTNSUB" "COBSUB" )

ENDSUBPROGRAMS
ENDRENAMES

ENDKB
```

図 4　副プログラム名置き換え
Fig. 4　Rename of subprogram

```
KB "RULES"

RULES

(準備   MODIFY_COBSUB
  (FORTRAN STA "CALL COBSUB(??AIPCT)"
   FORTRAN STB "BITS(??AIPCT(6),19,18) = LOC(?INP1)" :KEY STA :BACKWARD)
=>(MODIFY  STA "CALL COBSUB(??AIPCT,INP1)"
   DELETE  STB))

ENDRULES

ENDKB
```

図 5　FORTRAN 文の置換（MODIFY）および削除（DELETE）
Fig. 5　Modify and delete Fortran program

MODIFY 文で STA を新しい FORTRAN 文に置換する.

DELETE 文で STB を削除する (図5).

3) 副プログラム COBSUB の引数のレイアウトを与える (図6).

```
KB "LAYOUT"

SUBPROGRAMS

SUBROUTINE    COBSUB ( PCT AREA )
01  PCT.
    03 PCTX       PIC X(20).
    03 PCT6-H1    PIC 9(5) COMP.
    03 PCT6-H2    PIC 9(5) COMP.
    03 PCTY       PIC X(16).
01  AREA.
    03 AREA-1     PIC 9(10) COMP.

ENDSUBPROGRAMS

ENDKB
```

図 6 副プログラム COBSUB の引数レイアウト

Fig. 6 Layout of subprogram "COBSUB" parameter

```
CONTROL                    DIVISION.
    COPY  CTL-DIV  OF  COB.
IDENTIFICATION             DIVISION.
PROGRAM-ID.                TEST.
AUTHOR.                    KS-301.
DATE-WRITTEN.              08/28/90 19:09:18.
ENVIRONMENT                DIVISION.
CONFIGURATION              SECTION.
    COPY  CONFIG-SEC  OF  COB.
DATA                       DIVISION.
WORKING-STORAGE            SECTION.
01  IBUF                      PIC  S1(36).
01  PCT.
    03  PCT--3-1              OCCURS  10.
        05  PCT--5-1.
            07  FILLER        PIC  X(2).
            07  PCT--7--2 PIC  S1(18).
01  CX-0                      PIC  S9(10)  COMP.
/
PROCEDURE                  DIVISION.
C00-MAIN                   SECTION.
C00-MAIN-ST.
    PERFORM  C10-INIT.
    PERFORM  C20-MAIN.
C00-MAIN-EX.
    STOP  RUN.
/
C10-INIT                   SECTION.
C10-INIT-ST.
    CALL  'FTNINI'.
C10-INIT-EX.
    EXIT.
C20-MAIN                   SECTION.
C20-MAIN-ST.
    CALL  'LOC'  USING  IBUF  CX-0.
    COMPUTE  PCT--7-2 OF PCT (6)  =  CX-0.
    CALL  'FTNSUB'  USING  PCT.
C20-MAIN-EX.
    EXIT.
```

図 7 変換後 COBOL プログラムの例 (付加情報なし)

Fig. 7 Example of Cobol program (non added-information)

```
CONTROL                    DIVISION.
    COPY  CTL-DIV  OF  COB.
IDENTIFICATION             DIVISION.
PROGRAM-ID.                TEST.
AUTHOR.                    KS-301.
DATE-WRITTEN.              08/28/90 19:02:58.
ENVIRONMENT                DIVISION.
CONFIGURATION              SECTION.
    COPY  CONFIG-SEC  OF  COB.
DATA                       DIVISION.
WORKING-STORAGE            SECTION.
01  PCT.                                    )
    03  PCTX      PIC  X(20).               |
    03  PCT6-H1   PIC  9(5)  COMP.          |(#1)
    03  PCT6-H2   PIC  9(5)  COMP.          |
    03  PCTY      PIC  X(16).               |
01  IBUF.                                   |
    03  AREA-1    PIC  9(10)  COMP.         )
/
PROCEDURE                  DIVISION.
C00-MAIN                   SECTION.
C00-MAIN-ST.
    PERFORM  C10-INIT.
    PERFORM  C20-MAIN.
C00-MAIN-EX.
    STOP  RUN.
/
C10-INIT                   SECTION.
C10-INIT-ST.
    CALL  'FTNINI'.
C10-INIT-EX.
    EXIT.
C20-MAIN                   SECTION.          (#2)
C20-MAIN-ST.
    CALL  'COBSUB'  USING  PCT  IBUF.  (#3)
C20-MAIN-EX.
    EXIT.
```

図 8　変換後 COBOL プログラム（付加情報付）

Fig. 8　Example of Cobol program (with added-information)

付加情報を与えずに変換した場合(図 7)と与えて変換した場合(図 8)を比べると次のようになる.

1)　データ項目名 PCT と IBUF に対して副プログラム COBSUB の引数のレイアウトを展開する (#1).

2)　FORTRAN 文"BITS(PCT(6)…"に対応する COBOL 文が生成されない (#2).

3)　副プログラム FTNSUB の CALL 文が COBSUB の CALL 文に置き換わる (#3).

4. 知識ベースの実現方法

4.1 知識ベースの中身

知識ベースは各種の知識の集合体であり，各知識はそれぞれ LISP で記述したプログラムである．このプログラムには大きく分けて二つある.

第 1 は付加情報を基にルール・コンパイラが生成する知識である．これは内部テーブルにデータを定義したりする命令文等の羅列で成り立つ LISP プログラムである.

第 2 はルールから直接呼び出すユーザ自身が記述した LISP プログラムであり，

LISP 関数定義命令（defun）で成り立っている．

4.2 ルール・コンパイラ

付加情報はユーザが作成するため（実際は Pickup が生成する変換情報に含まれる．ユーザは Pickup に対して付加情報の基となる情報を準備するだけでよい），人間が識別可能な形ではあるが FTC トランスレータには識別できない．このため付加情報を FTC トランスレータが認識可能な表現に変換する必要がある．この変換はルール・コンパイラと呼ぶ独立したソフトウェアによって行う（図 9）．

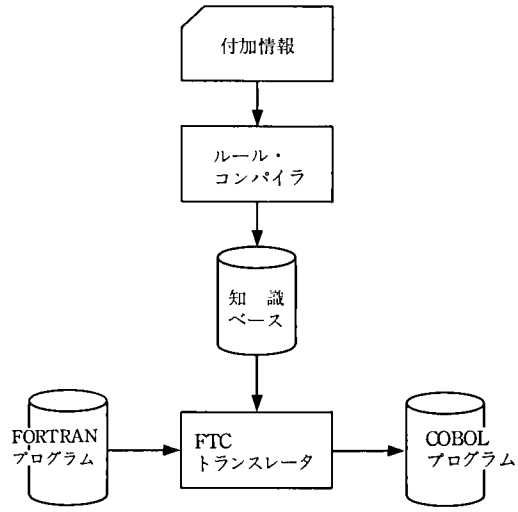


図 9　ルール・コンパイラと FTC トランスレータの関係

Fig. 9　The relation between Rule-compiler and FTC-translator

4.3 知識ベースの適用

変換ごとに知識ベースから必要な知識をメモリ上にロードすることにより，FTC トランスレータが知識を適用する環境ができる．

このロード実行により知識として記述されている LISP プログラムの命令を実行し，FTC トランスレータの内部テーブルにデータをセットしたり，ルールから呼び出す LISP プログラムを FTC トランスレータのプログラムの一部として採り込む．

5. FTC システムの運用

5.1 運用の省力化

KS-303 はシリーズ 2200・1100 と操作がまったく異なり，ユーザにとってオペレーションが問題となる．そのため FTC システムでは KS-303 側でのオペレーションの省力化を考慮している．省力化の結果としてユーザは KS-303 の立ち上げと停止だけを行えばよく，その他のオペレーションはシリーズ 2200・1100 上ですべて可能である．

省力化の具体的な実現方法は次の通りである．

1)　KS-303 を立ち上げると，FTC システムの 1100 daemon と FTC daemon が自

動的に起動を始め，両ソフトウェアが常駐してファイル転送から変換に至るすべてのFTCシステムを無人で絶えず管理する．

2) 1100 daemonは，ある一定時間ごとにKS-303側からシリーズ2200•1100をオープンし（本来はユーザがUTS端末からデマンドモードでキーインするコマンドを1100 daemon自身が生成しキーインする），ある特定のプログラムファイルに変換情報のエレメントが登録されているか調べ，KS-303にファイル転送を行う．

またFTCトランスレータで変換した結果があれば逆にシリーズ2200•1100にファイル転送を行い，コンパイルを実行したり変換結果をプリンタに出力する．

送受信するファイルがない場合にはある一定時間タスクをSLEEPさせる．

3) FTC daemonはある一定時間ごとに1100 daemonで受信したファイルがあるかチェックを行い，FTCトランスレータで変換を行う．変換するファイルがない場合にはある一定時間タスクをSLEEPさせる．

5.2 リカバリ機能

FTCシステムの中で，KS-303側は無人運転となるため，異常が発生した場合のリカバリ機能を考慮している．異常な状態となるのは次の通りである．

1) 各種ソフトウェアがなんらかの理由でエラー終了する．

2) 各種ハードウェアがストップする（シリーズ2200•1100またはKS-303のマシン停止，両マシンを接続する回線の切断等）．

リカバリ機能は，1100 daemonとFTC daemonの二つの運用ソフトウェアで同一の支援方法が採られており，異常状態によりそれぞれリカバリ対応を行う．

5.2.1 ソフトウェア異常によるリカバリ機能

ソフトウェア異常によるリカバリは，ソフトウェア異常を発生させるプログラムの変換を停止し次のプログラムの変換に移り，異常状態の詳細な情報をシリーズ2200•1100のプリンタに出力する．そしてFTCシステムは異常状態が発生する変換プログラムを除いて最後まで正常に実行する．

異常状態が発生した変換情報に基づきユーザが対応を行った後，再度変換を実施する．

5.2.2 ハードウェア異常によるリカバリ機能

ハードウェア異常によるリカバリは異常発生時の実行状態を無効とし，再度異常発生時の変換プログラムを最初から実行することで対応している．

1100 daemonとFTC daemonはそれぞれ独立して動作しており，それぞれの動作手順は図10の通りである．またソフトウェア間の実行順序は図11の通りである．

両ソフトウェアとも前段階からのファイルの有無で処理を行うかを決定し，正常に処理が終了した場合のみ前段階のファイルを削除する．ハードウェア異常が発生した場合には前段階のファイルは残されており，マシン再起動時に再度自動的に処理が行われる．

この機能によりユーザは常時マシンの停止を行っても，あらためてPickupから実行せずに処理を途中から再実行することができる．

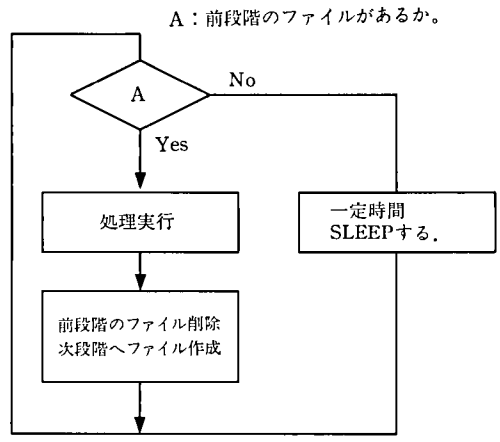


図 10　動作手順

Fig. 10　Process control

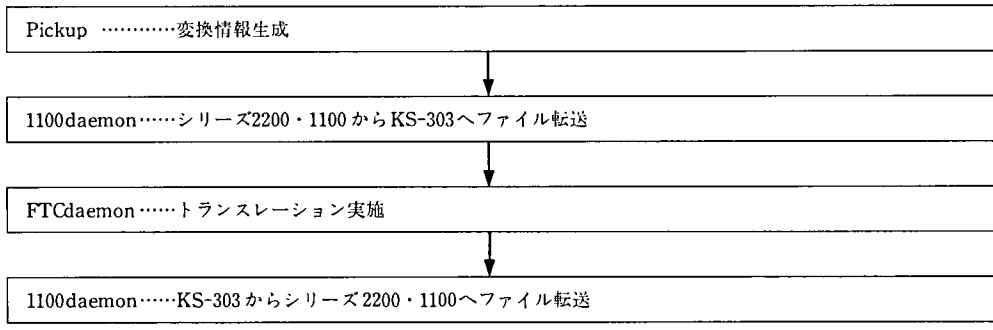


図 11　ソフトウェア間の実行順序

Fig. 11　The order of software

6. 変換実績と評価

6.1 変 換 実 績

冒頭の「1章　はじめに」で述べた通り，客先におけるこれまでの変換実績として

FTN プログラム　　約 840 本　172,000 ステップ

↓

ACOB(または NCOB)　同本数　340,500 ステップ

が実施された．ちなみに上記ステップ数は変換前の FORTRAN および変換後の COBOL プログラムのソースコード・ライン数（ただし登録集展開前）をカウントしている．

6.2 投入工数と変換生産性

本節での生産性算出の範囲としては，対象プログラムの選定，ルール登録，KB 情報収集，実変換作業およびテスト，COBOL での本番までの全工程の通しの工数を対象とする．

詳細は割愛するが客先工数と当社の変換技術支援工数を加えると概算で約51人月を費やしているので，当該ユーザにおける変換生産性は次の通りである．

・変換前 FORTRAN ソースベースの生産性

172,000 ステップ÷51 人月＝約 3,373 ステップ

・変換後 COBOL ソースベースの生産性

340,500 ステップ÷51 人月＝約 6,676 ステップ

一般に事務計算分野のバッチ業務の新規開発の場合，要求仕様作成から本番までの通しの生産性は，COBOL ベースの場合 600～1,000 ステップ（要求仕様が与えられている場合は本番まで 800～1,200 ステップ前後）といわれている．

今回の変換作業の要求仕様は既存のままであり，800～1,200 ステップの開発生産性と比べることになる．単純な比較はできないが，少なくとも新規開発よりもはるかに高い生産性でシステムの移行が可能であると見ることができる．また，移行中ないし移行後に必要部分の機能強化・改良を施したとしても，その程度にもよるが，最初から開発するよりはるかに少ない工数で済むことが予想される．要求仕様に立ち戻ってシステムを全面的に組み替える場合を除いては，極めて有効な手段であると考える．

6.3 ステップ数，メモリの増加率

1) ステップ数の増加率は，変換前の FORTRAN プログラムと変換後の COBOL プログラム各々の総ステップ数の対比で求めることができる．

340,500(COB)÷172,000(FTN)＝COBOL に変換後，約 1.98 倍

この理由は主として COBOL 言語の特性であるプログラムの文書化およびデータ定義部分の増加に起因しているので，プログラムの見やすさを考慮するとやむをえない増加と考えられる．

2) メモリの増加率は，今回は残念ながら大量サンプルでは測定できていないが，平成3年3月に実施した試行プログラム7本の測定値を表2（試行プログラム変換実績）に示す．これによると DMS 等を使用していない単純なバッチ・プログラムで平均 30％強の増加率となっている．

ステップ数との関係では小量サンプルなのでいちがいには言えないが，単純バッチ・プログラムの場合，ステップ数の増加ほどはメモリは増えていないことが推定される．この理由として，両言語のライブラリ等の相違に加え，通常 COBOL に比べ FTN の方がソースコードの1ラインで表現できる情報量（機能）が多いことが考えられる．

6.4 トランスレータによる自動変換率

この点についても前述のメモリ増加率と同様，大量サンプルでは測定できていないが，やはり前出の表2（試行プログラムの変換実績）に抽出調査を行った実績を示す．この表の“修正 STEP”の欄が，トランスレータによる自動変換後の COBOL プログラムに対し手修正を施したライン数を示す．

また同表の変換率は，“変換率＝(“ACOB STEP” － “修正 STEP”)÷“ACOB STEP”で計算され，変換後の COBOL プログラムに対する無修正ライン数の割合を示している．この結果，試行フェーズでは，変換率は平均 85％程度とみることができる．

その後はトランスレータの機能強化，ルール登録の充実等により，現段階では 90％

表2　試行プログラム変換実績
Table 2　Translation result of test program

プログラム名	MAIN /SUB	FTN STEP	ACOB STEP	修正 STEP	変換率	FTNメモリサイズ I-BANK/D-BANK	ACOBメモリサイズ I-BANK/D-BANK	メモリ増加率	プログラム概要
プログラム1	MAIN	132	135	15	88%	6967/9226	9138/10406	21%	TAPE と CARD 入力のマッチング
プログラム2	MAIN	226	536	99	81%	18124/23466	19028/21130	−4%	DMS DB 入力で対象データ抽出
プログラム3	MAIN	48	100	11	89%	8244/5792	11119/7522	33%	TAPE TO PRINT
プログラム4	MAIN	63	123	21	82%	14879/26634	15094/18980	−18%	DMS DB 入力で対象データ抽出
プログラム5	SUB	66	172	16	90%	148/21	384/292	400%	DMS DB よりレコード作成(計算・編集)
プログラム6	SUB	118	238	21	91%	244/139	337/196	39%	レコード出力
プログラム7	SUB	115	407	81	80%	99/475	206/564	34%	TAPE TO PRINT

以上クリアできているものと想定している．

6.5 変換時間

今回の客先ではマシン上の変換作業が自動運転のため，定量的なデータは測定できていない．抽出測定ではKS-303上のトランスレータの作動時間として，500ステップのFORTRANプログラムで2〜3分前後必要である．

7. 今後の課題

今回の客先での変換実績を踏まえ，今後に向けて次のような課題を挙げることができる．

1) FTCトランスレータの処理効率向上の必要性と汎用性．これは変換をKS-303上で実現しているため，ハードウェアの性能に依存してしまい，大型汎用機のような高速処理が期待できない．今後は汎用機（たとえばOS 1100上にトランスレータを移植）での変換を実現すれば，効率の向上に加え，多くのユーザで手軽に変換が可能となる．この点は今後のユーザのニーズにもよるが，技術的には可能と考えている．

2) 人間がコーディングしたものに比べて，見やすさの点で限界がある．この点は簡単には説明しきれない多くの検討課題を抱えている．言語の特性を活かす点については，前述の「3.1.2項の生成コードのCOBOLらしさの追求」の項でトランスレータで実現している一例を紹介したが，これもさらに見やすさを追求すれば付加情報登録等のユーザ負荷が増大する．また，変換後のCOBOLプログラムの整構造の実現についても今後の課題である．

3) 単純なコンバージョンと違って，プログラムごとに固有の付加情報（レイアウトやルール）等をユーザが準備するが，この作業に変換全工程の6割程度の負荷がかかる．この部分は生産性に大きく影響するので，できるだけ省力化が望まれる．

今後，これらの課題を解決してソフトウェアをさらに整備することにより，他のユーザにも広く適用することが可能となる．

8. おわりに

本稿は，FORTRAN から COBOL システムへの大量移行をプログラム再開発ではなく，コンバージョン（言語変換）で行った初めての試みについて，トランスレータおよび運用システムを中心に実績評価を踏まえて報告した．今回の報告が，言語変換に係わる方，および FORTRAN 中心のシステムから COBOL 中心のシステムへ再構築を検討している方の参考に供することができれば幸いである．

執筆者紹介　瓜　生　和　史（Kazushi Uryuu）
昭和 61 年慶応義塾大学商学部卒業．同年日本ユニシス（株）入社．システム移行作業および移行ツール開発に従事．
現在 システム技術本部 システム移行技術部に所属．

大規模分散処理環境下における開発および保守

Systems Development and Maintenance in the Large-scale Distributed Data Processing Environment

日 高 修 一, 寿 賀 徳 静

要 約 オフィス・コンピュータを利用しているユーザでも，ここ数年全国規模でエンドユーザにあたる関連部門および関連会社にコンピュータを設置し，システム・サービスの一環として導入展開から運用支援まで，独自の方法で援助する例が増えつつある．

こういった大規模分散型システムを抱えたユーザにとって，開発のむずかしさはもちろん，展開後の運用支援等で，大きな工数がかかる可能性が高く，これらの問題をいかに最小限にとどめるかが成功の可否を握っている．

本稿では，この種のシステムの実現を目指しているユーザの事例を通し，

1） 開発準備段階の構想
- 分散展開とセンタ側コンピュータにおける構成の配慮
- DDX-P 網を利用したネットワーク網の整備

2） システム設計段階での事前対策
- 開発工数および導入後の支援工数削減を目的とした運行管理システムの作成と標準化への取り組み

3） ネットワークによる遠隔保守の実現
- DDX-P 網活用によるリモート端末と AS-NET による遠隔保守

4） 現状の問題点とその取り組み
- トラブル対応体制と新規課題の取り組み

等の観点からその有効性と今後の課題について述べている．

Abstract Increasing in number these days are the cases where even small business computer users provide their related in-house departments and affiliated firms who have finished installing computers at their request with their nation-wide support ranging from initial computer installations to daily computer operations as part of their systems services activities. For those users equipped with such large-scale distributed data processing systems, a large number of processes are inevitably required for not only the pursuit of difficult systems development tasks but also for overall follow-on services from equipment installation through practical systems operation, and success entirely depends upon how to minimize the volume of the problems involved.

Citing as a case a certain user who has been working on building this type of system, this article discusses effective systems development and maintenance in the large-scale distributed data processing environment in terms of design concepts at the level of pre-development, considerations at the stage of systems designing, remote maintenance in computer networking, and current problem analysis and handling for future greater convenience.

1. はじめに

ユニット工法では大手メーカの一社であるA社において，昭和63年4月より，全国23箇所の営業部，営業所，建設会社でのオフコン上位機種への移行作業，関連販売会社へのオフコン新規導入計画が開始され，現在も進行中である．

準備期間の約1年を含め，たび重なる打ち合わせ段階より販売会社システムの開発，導入展開に携わった経過を踏まえ，今後ますます増えていくであろうと予想される，オフコン（シリーズ8）における大規模分散処理型のシステム・サービスへの対応の一手法を紹介したい．

2. 新システムにおける運用構想

2.1 現行（旧）システムの見直し

A社は，昭和59年から同社業務体系に見合った独自の会計システムをシリーズ8（以後S/8と略する）DPS IV機にて開発し，各事業所に導入してきた．

システムは部門別に大きく，

① 営業所システム（直販部門）
② 営業部システム（関連販売会社の管理部門）
③ 建設本社システム（関連建設会社の管理部門）
④ 建設会社システム（関連建設会社）
⑤ 関連会社システム（住宅販売以外の関連会社）

に分けられる．そして，それぞれのシステムは，住宅契約・管理，入金管理，積算，受発注管理，原価管理，等のサブシステムから成り立っている（図1）．

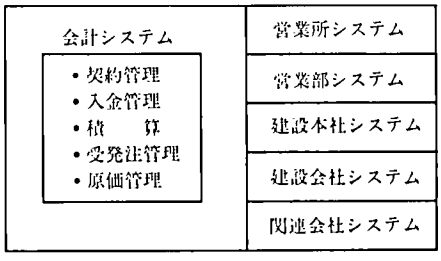


図1　システム概略
Fig. 1　Component of management system

また，各部門間のデータのやりとりはメールボックス・ファイルを用意し，センタのDPS IV機が，夕方各地からの送信電文を受信し仕訳し直して翌朝各地に配信するという運用を行ってきた．タイムラグが問題となる営業所から建設会社間のデータ転送は，各事業所間を公衆網による$UTRNS* ファイル転送で，直接つないで対応してきた（図2）．

しかし，近年の好景気を背景とした住宅需要の伸びに伴い，データ量が当初見積りの倍近くまで増加し，ディスクの容量不足から派生する障害が多くなってきたことや，公衆網による回線上の障害，処理速度の問題の解決のためにシステムの再検討を行う

* シリーズ8間のファイル転送ユーティリティ・プログラム

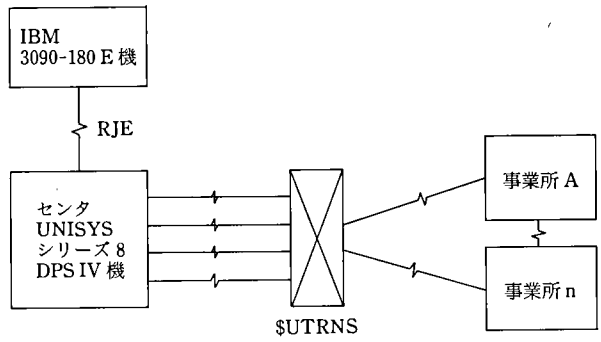


図 2 旧ネットワークシステム

Fig. 2 Outline of network (Old system)

とともに，全国の関連販売会社への本システムの導入計画に着手した．

2.2 新システム構想

新システム(図 3)では，前節で挙げた項目の検討を踏まえて，次の項目が盛り込まれた．

1) 処理速度の向上を目指して，現行システムのうち営業所システム，営業部システム，建設本社システム，建設会社システムを DPS IV機の上位機種である DPS 10 機へと移行する．

2) センタでの大量のデータ処理に対応するために，各地の S/8 とセンタの S/8 の

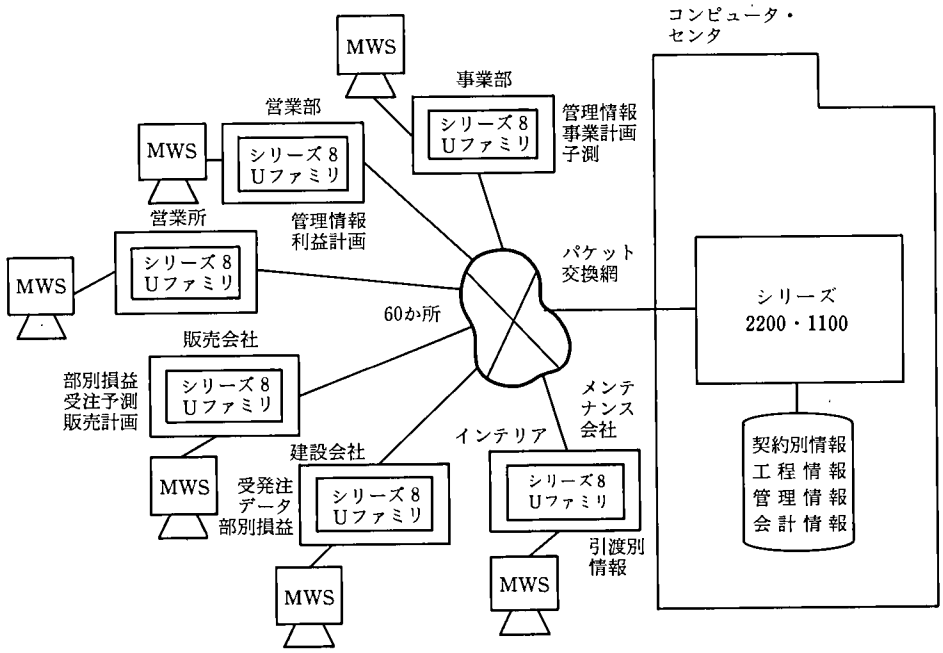


図 3 新システム構想

Fig. 3 New system outline

中継機として，またセンタ・コンピュータである IBM 3090-180 E システムへの窓口として UNISYS 2200/200 システムを導入する．

3) 回線上のデータの品質向上，回線トラブルの低減を目指して，各地の DPS 10 機と一部を除く DPS IV機をセンタ側 DPS 10 機，DPS IV機，2200/200 システムとパケット交換網（DDX-P）を介して接続する．

4) 各地から集信された営業情報のデータベースを 2200/200 システム上に構築し，UMML 11/UTS エミュレータ* にて各地の DPS 10 機に接続されるマルチワークステーション（以後 MWS と略す）を MAPPER（ユニシスの提供する 4 GL の一つ）端末とすることで情報検索/照会業務を行う．

5) これまでコンピュータ化されていなかった関連販売会社に対し，営業所システムの改訂版の開発導入展開と，一部のデータの営業部への転送を行う．

既存システムの改善と新規展開構想のもとで，昭和 62 年より DPS 10 機への移行作業と，DPS IV機での関連販売会社システム（以降「販社システム」と略する）の開発作業，MAPPER による営業情報検索システム開発作業を昭和 63 年春頃をめどに同時進行の形で進めることになった．

3. システム開発およびシステム移行時における配慮

3.1 作業工数の削減

これまで開発してきた 2000 本に及ぶプログラムと JCL を 1 本 1 本修正した場合，保守にかかる作業工数は計り知れず，テストにも同等の工数をとられることになる．しかし，作業要員の手配や開発期間には限りがあるため，それだけの工数をかけることは困難である．そこで，単純な修正・移行作業は機械化すべく，開発用ツールの作成を行うことにした．

開発されたツール群の主なものは次の通りである．

1) コピー JCL 作成プログラム……ここで言うコピー JCL とは，エントリ・プログラムを各端末で同時に起動できるようにトランザクション・ファイルを個別化した JCL を言う．機能として，任意の JCL を指定し自動的にトランザクションファイル名を端末台数分変更し，別個の JCL を作成するものである．

2) ファイルレコード件数パラメータ変更プログラム……全 JCL に対し，対象となるレコード件数パラメータの拡張/縮小を行う．

3) ユニット名の変更プログラム……任意の JCL 名を指定し，ファイルラベル名とそのユニット名の定義に基づいて，対象となるユニットパラメータの変換を行う．

4) 導入・展開モジュールの一括出力……その中で使用されるプログラムをインストール用ライブラリに格納する．

5) 使用項目チェックリスト……ファイル名とチェック対象項目の項目位置を定義し，指定 JCL に対し使用しているプログラム名をリストアップする．

これらのツール群を開発段階と導入展開時にそれぞれ使い分けることで要員不足を補い，かつ修正モジュールの精度を上げることに成功した．

---

* シリーズ 2200・1100 とシリーズ 8 のワークステーション（MWS）の間をハイレベル伝送制御手順（HDLC-LAPB）で接続し，MWS をホストのターミナルとするプログラム

3.2 DPS 10 移行時の工夫

現行 DPS IV機で稼働中のシステムを，DPS 10 に移行するにあたって検討された項目は，次の二点である．

1) DPS IV機とのオペレーション上の相違点をできるだけなくし，エンドユーザの操作上の負担を軽減させる．

2) できる限り保守が容易となるようファイル・システムを設計する．

1)については，とくに問題となったのが端末ごとにオルタネートプリンタを定義するという DPS IV機の機能が DPS 10 では提供されていないということであった*．この問題は端末ごとにプリンタを割り付けた定義ファイルを参照して，プログラムで割り付けるという手法をとることで解決した．また，コンソールからのアクセプト入力で簡単にプリンタの割り付けを変えられるようにした．

2)については，3.1 節の 1)で取り挙げたコピー JCL 作成の問題を解決するために，端末ごとにログインディレクトリを変え，端末台数分同一 JCL を持つという手法をとった(図 4)．これは端末台数が増えると，JCL サイズも増加するというデメリットはあるが JCL を一本化できるため，保守時や JCL の管理という点で非常に有効であった．また，OS 部，常駐マスタエリア，ワークエリアをそれぞれ別論理ボリュームに配置することでディスクアクセスタイムを減らし，ディスクの負担の軽減に努めた．その他，サブシステムごとに論理ボリュームを振り分け，ディスクトラブル時に全業務がストップしてしまうことを避ける工夫を盛り込んだ．

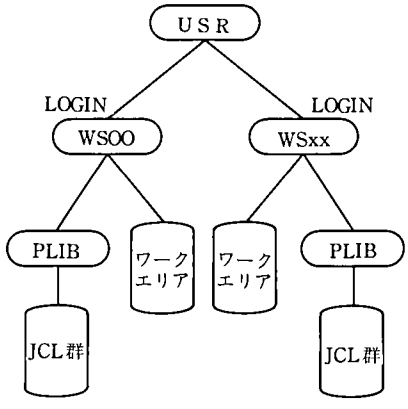


図 4 端末ごとにログイン先を定義する

Fig. 4 Basic pattern of file directory

3.3 運用効率の向上に対する検討

旧（移行前）システムの問題点の一つに処理時間の問題があった．毎日，バックアップや，マスタ再編成時のファイルコピーのために数時間が費やされる．このことを重要視し，処理時間のかかるバッチジョブは夜間に無人運転を行うようにした．またバックアップは，ディスク内のバックアップエリアに夜間のうちにコピーするようにした．この夜間走行ジョブによって，エンドユーザは無駄に数時間を費すことなく朝一番からコンピュータによる業務を開始できるようになった．

---

* 当時のバージョンは E 00 版である．この機能は，E 10 版より提供された．

しかし，夜間走行ジョブの対象となる業務は多く，10日ごと，15日ごと，一月に一度という間隔で走行させる必要のある業務もあり，夜間走行ジョブをスケジューリングする機能が必要となった．結果的に5.1節で述べる運行管理システムと組み合わせることで，スケジュール・ファイルを参照しながら必要な業務だけを実行し，最後に自動的にコンピュータの電源をオフする夜間走行ジョブを作成し，今回の新システムにおけるポイントの一つとした．

3.4 新メールボックスシステム

1) 2200/200システム導入の背景……エンドユーザの増加とそれに伴うデータ量の増加により，処理時間やディスク容量の面で，従来通りのセンタのDPS IV機をホストとしたメールボックス仕訳処理が困難となってきた．そのため，仕訳処理をDPS IV機より切り離し，2200/200システムでその処理を代行することにした．

2) 集配信データの種類……新メールボックスシステムにおけるデータの流れは図5に示す通りである．

① 決算データ：月次データとして各地S/8から集められたデータをIBM機へ送信する．

② 他事業所分負担会計データ：目次データとして各地S/8から2200/200システムに集められた後，対象となる事業所行きのメールボックスに格納して翌朝配信する．

③ 変更マスタ：月次データとして，変更となるマスタファイルのデータを，センタ側DPS IV・10機から各地のS/8に配信する．

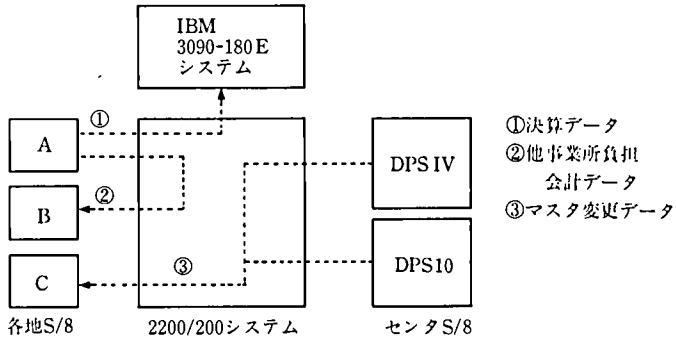


図5 新メールボックスにおけるデータの流れ

Fig. 5 Outline of data transfer by mail box system

3) 伝送媒体……A社既設の高速デジタル専用線を利用し，全銀協手順にてパケット伝送を行う．

3.5 センタ開発環境に対する検討

システムの開発・保守およびユーザ支援時の重要項目として，作業工数の削減，要員の確保と同時に作業環境の整備を挙げることができる．とくに今回はDPS 10への移行作業とDPS IV機での販社システムの開発作業が平行していることから，コンピュータの有効な活用方法が必要となった．また旧システムでは10数箇所だったエンドユーザが，販社システム導入後は一挙に5倍にふくれあがる．これらの状況に対する

保守体制を整えるためにも作業環境の整備は必須項目であった.

しかし，事務所内の床面積や，電源設備上の制約からDPS IV，DPS 10双方の端末拡張台数には制限があった．そのため，DPS IV～DPS 10間をHDLC(HIGHLEVEL DATA LINK CONTROL)による構内回線で接続し，M4374PKTシステムによるリモート・ワークステーション（以後RWSと略する）ならびにファイル転送を実現させた．これにより，各OS下の相互乗り入れ型のRWSとして必要に応じ，随時切り替えによる運用が可能となった．また，UTRNS (DPS IVは$UTRNS)によるファイル転送と，CL（COMMAND LANGUAGE）モードのJCLの組み合わせにより，外部入出力装置に頼らずに，DPS IV～DPS 10間で任意のソース・プログラム，JCLをやりとりできるようにした.

一方，DPS 10～2200/200システム間を既設の高速デジタル専用線にてHDLC接続し，DPS 10の全MWSをUMML 11/UTSエミュレータによるMAPPER端末とすることで，エンドユーザ向けMAPPER活用システムの開発にも一役を担った．その他，MWSはSuper-REPO* を試験的に導入しており，実にMWSは異なる四つのOSを，必要に応じて切り替えながら活用されていたわけである.

4. マシンの新規導入・展開の体制について

開発・移行作業が完了すると，次はいよいよマシンの導入・展開作業となる．しかし，DPS 10機で20数箇所，DPS IV機にいたっては約50箇所に順次手際よく導入していく必要があった．そのため，導入展開にあたっては標準化を最重要視した.

標準化された主な項目は次の通りである.

1) マシンのクラスやディスク構成は，導入するシステムやデータ量によって異なるため，DPS IV機ではJCL内で割り付けされているユニット名に，論理ボリューム名を用いることにした（表1)．これにより作業者は，新規導入ユーザのディスク構成に基づいて，システム生成時に適宜，物理ボリュームと論理ボリュームの割り付けを行ってやるだけでよくなった.

一方，DPS 10でも同様にファイル・システム構成，ディレクトリ名，パス名をシステムごとに統一した(表2)．その都度，JCLを修正しなくて済むことは，作業工数の削減に大きなメリットとなった.

2) SSV（SECONDARY SYSTEM VOLUME）も端末数，利用回線形態に応じてパターン化したものを各種準備している.

3) OSのバージョンについてはすべてDPS IVはE 30版，DPS 10はE00版で統一している．このことはバージョンの違いから発生するある種のベーシックなソフトウェア上のトラブルから開放されることになり，また保守という面からも有益である.

4) 導入マニュアルの作成も，導入展開の効率向上を考えるにあたって忘れてはならない課題の一つである．場合によっては導入経験のないものが作業する場合も考えられるため，マニュアルには細かなオペレーションやメッセージまで書き加えるようにし，誰でも問題なく作業ができるまで修正を重ねた.

---

* Super-REPO：データベース，表計算，英文/和文ワープロ，グラフ，送信機能を備えた統合ソフトウェア

表1　論理ボリューム割り付けの標準化
Table 1　Standardization of logical volume assignment

名　称	FILE LABEL名	FILE 編成	ISAM·KEY 位置	ISAM·KEY 長さ	レコード長	レコード件数 大	レコード件数 中	レコード件数 小	容量(セクタ) 大	容量(セクタ) 中	容量(セクタ) 小	容量(MB) 大	容量(MB) 中	容量(MB) 小
OS関係 (DK 00)									141023	141023	141023	36.1	36.1	36.1
マスタ (DK 01)									62014	62014	62014	15.5	15.5	15.5
部情報(入金を含む) (DK 02)									90750	65904	36293	22.7	16.5	9.1
出納・決算 (DK 03)									143941	118328	75663	36.9	30.3	19.4
原価(積算) (DK 04)									36273	27431	13868	9.3	7.0	3.6
原価(発注) (DK 05)									183229	146594	73334	45.8	36.7	18.3
管理情報 (DK 06)									29372	27519	27424	7.4	7.1	6.9
送受信 (DK 07)									21615	18070	15707	5.4	4.5	3.9
運行管理 (DK 08)									1521	1521	1521	0.4	0.4	0.4
ワーク (DK 09)														
BACK-UP (DK 10)														
給与 (DK 11)														

表2　マウントディレクトリ名の標準化
Table 2　Standardization of mount directory name

/LOAD	プログラム・エリア	/KJ	建設会社データ部
/USR	作業エリア	/KH	建設本社データ部
/EG	営業所データ部	/EB	営業部データ部

5.　ユーザ支援体制について

5.1　トラブル防止対策

今回の新システムの検討にあたり，以前より作成・活用されていたトラブル・レポートをもとに，トラブル追跡に費やされる工数の分析および工数の削減化を行った．コンピュータの使用経験のないユーザや，専門SEのいない各販社に対する大量展開に備え，トラブル追跡工数の大半を占めるオペレーションミスへの対策を考える必要があった．

オペレーション・ミスの大半は未然に防ぐことができる種類のものである．しかし，人間にはミスが付きものである．そこで人間の代わりに業務のチェックを行う運行管理システムの開発がスタートした．基本的な形態は，業務JCLの初めと終わりに運行

管理システムを付加し，その業務JCLが適正でないと判断された場合，業務処理をパスしてそのまま終らしてしまうというものである．

主な機能は以下の通りである．

1) DPS IV機の仕様制限によるジョブの異常終了がオペレータには意味が理解できないため，運行管理システムで平易なメッセージに置き換えオペレータに知らせるようにした．この機能はとくに同一端末におけるオンライン・プログラムの二重起動の防止等に役立っている．

2) 走行ログファイルを設け，ジョブの二重起動を防止した．これは日次，月次，期次のバッチ更新プログラムの二重起動の防止に有効な機能となった．

3) 必須ジョブの走行もれをスケジュール管理し，走行もれリストにてユーザに知らせるようにした．

4) センタ側との送受信で，ホストマシン側の未稼働に起因するジョブ異常終了を防ぐため，休日ファイルを設け，ジョブ起動を制限するようにした．

5) ジョブの相互の走行順番と排他関係を定め，不用意なオペレーションによるファイルロック状態の解除とデータ破壊を防止した．

なお，運行管理システムにて扱うスケジュール・ファイル，休日ファイルはセンタ側で一か月単位でS/8にて作成・保守されメールボックスに格納されて2200/200システム経由で各地に送信される．

5.2 ネットワークによる遠隔保守の実現

5.2.1 新ネットワーク網

新システムへの移行に伴って，ネットワークがDDX-Pで接続されたことはすでに述べた．ネットワーク系の概要を説明すると，まずセンタ側S/8は2200/200システムと高速デジタル専用線にてHDLC接続されている．また，それとは別に非常用として，DDX-P網による接続も行っている．DPS IV機はこれらの回線を通じてメールボックスの送受信を全銀協プロトコルにて行っている．DPS 10機はこれらの回線に2本の論理リンクの定義を行い，1本はDPS IV同様メールボックス送受信用とし，もう1本はUMML 11用ラインとして定義している．なお，DDX-P網による接続はPVC (PARMANENT VIRTUAL CIRCUIT) 接続されている．そしてエンドユーザ保守用として各地のDDS 10機とDPX-Pを介し，VC (VIRTUAL CALL) 接続されているが，メモリの関係上3ポートとしている．そしてそれぞれのポートはM 4374-RWS (リモートワークステーション) ライン，同一RRP (リモートプリンタ) ライン，同一ファイル転送ラインの3本の論理リンクが定義されている．同じくDPS IV機も各地 (一部を除く) のDPS IV機とDDX-PにてVC接続されているが，こちらは4ポート分提供している．また，DPS IV機はDDX-P未加入の一部販売会社の保守用に公衆回線も4本提供しており，AS-NET* を用いた保守を行っている．

一方，各エンドユーザは2200/200システム向けの全銀協ラインとUMML 11ラインをDDX-P (PVC) 接続で，センタS/8行きのM 4374ラインをDDX-P (VC) 接続にて，各1回線ずつ提供している (図6)．

---

* ユーザシステムのソフトウェア面の保守を，オンラインを利用してホストコンピュータにて直接行う機能を実現したシステムツール

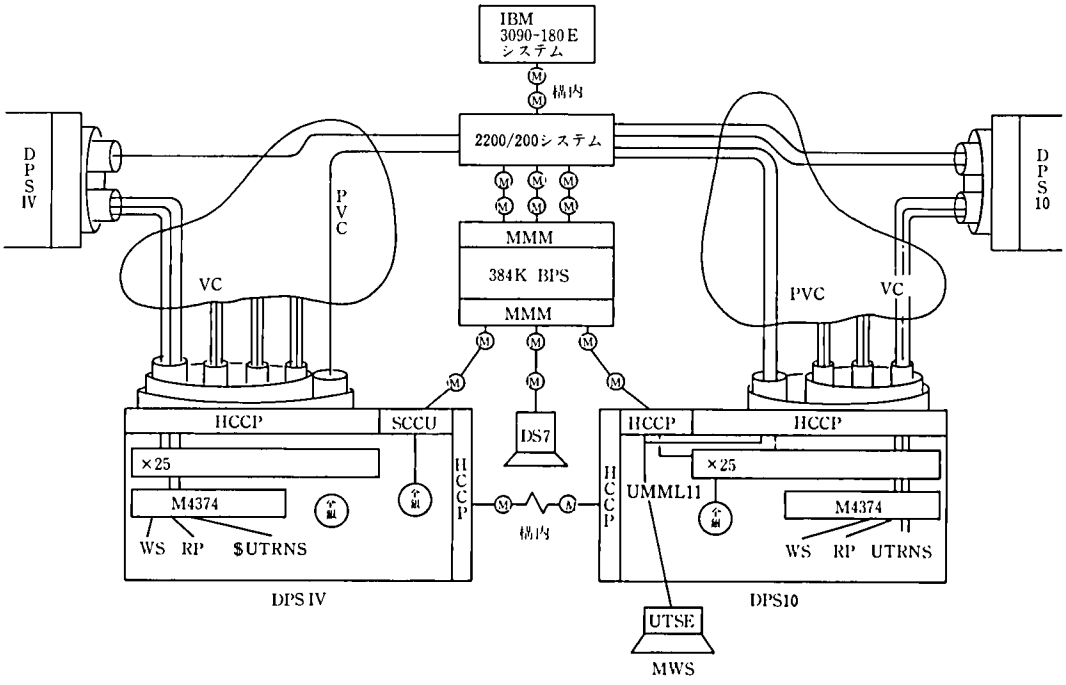


図 6　新ネットワークシステム

Fig. 6　New network configuration

5.2.2 遠 隔 保 守

こういったネットワーク環境を背景に，従来の公衆回線利用の AS-NET の活用に加え，RWS による保守サービスを開始したが，的確なトラブル状況の把握と迅速な保守作業にて作業効率が向上したことは言うまでもない．ファイル容量不足等の比較的軽度のトラブルであれば，電話のみの対応に比べて 1/4〜1/5 の作業時間で修復が可能となった．

RWS にしろ，AS-NET にしろ，オペレーションはエンドユーザに頼るしかなかった電話対応保守に比べ，修復の正確さという点で高く評価できる．

また，非常時には DPS IV機の端末を DPS 10 に接続し，DPS 10 の RWS とした後で，エンドユーザの DPS 10 機の RWS として，修復を行うという方法も非公式には行われている．

5.2.3 課 金 体 系

パケットサービスに対する課金体系は，先方負担（受信課金）を原則としているためエンドユーザにも自覚を促す結果となっている．

5.3 プログラム・JCL の保守

新システムへの移行を期に，プログラム・JCL をネットワークにのせて各地にリリースするシステムも考案された．

これは DPS IVの場合，リリース対象モジュールを転送用ライブラリにコピーし，ライブラリごとメールボックスに格納し，各地に送信するというものである．しかし，DPS 10 の場合はライブラリ自体が存在しないため，ライブラリの替わりとなるファ

イルの中に圧縮してからメールボックスに詰め込むという作業が必要であった.

また,メールボックスのエリアを節約するため,DPS 10 はソースファイルと JCL を送信し各地で受け取ったあと自動的にコンパイルするという手法をとった.

昭和 64 年初頭の元号対応,消費税対応といった,プログラム・JCL を全国に一斉にリリースしなければならない状況で,このネットワークはしばしばその有効性を発揮している.

しかし,モジュールの保守にはさまざまなスケールの場合が考えられるため,おおよその基準として下記の形態がとられている.

1) 一部限定されたモジュールの全国一斉リリース方法

緊急時には AS-NET によるモジュール送信.

事前対応時には 2200/200 システム経由で翌朝配信.

2) 数十本単位のモジュール展開

大がかりなモジュールのリリースが必要とされる時は,事前に十分な時間を見込めることと,現状の 2200/200 システムのディスク容量の不足からフロッピィやテープによるリリースが従来通り行われている.しかし,これは今後センタ側の機器を充実させることでいずれ解消されることとなろう.

また,DPS 10 ではセンタ内の DPS IV〜10 間のモジュール送受信機能と同様の機能を各地 DPS 10 との間で持っているため,特定の場所に少量のモジュールを即時に送りたい場合に有効となる.

5.4 メッセージ送信

朝のメールボックス配信処理が 2200/200 システム側のトラブル等で遅れる場合等,緊急を要する場合,各エンドユーザにいちいち電話している暇がない.このような時のために各地に一斉にメッセージを送るシステムが考案された.

概要は,フロッピィに保管されている基本の文書ファイルを画面より修正し,次にメッセージ送信先を指定することにより,プログラムが自動的に回線を割り当てて UTRNS のパラメータを作成し,その JCL を起動するというものである.ファイルを受け取ったエンドユーザはコンソールに指示されている JCL を起動すると,画面にセンタからのメッセージが表示されるという仕組みである.

一回のオペレーションで全国各地にメッセージが送れるという点で緊急時の連絡には非常に有効なツールとなる.

6. 新システムにおける課題

6.1 トラブル解決における問題点

6.1.1 ネットワーク系

ネットワーク系のトラブルは,目に見えない部分が多く,専門的な知識も要求されるため,業務上のトラブルと比べると,余計に時間がかかってしまうことが多い.A 社における導入作業時にも,いくつかのネットワーク系のトラブルが発生した.その経験よりリカバリを行う際のポイントをいくつか挙げ,今後に活かしていきたい.

1) ソフトウェアか,定義か……リリースされて間もないベーシック・ソフトウェ

アは，まだ不完全な場合がある．ソフトウェアのバグか，定義パラメータの設定ミスかの見極めも重要な要素である．そのため，今回の導入作業ではオンライン上の仕様の統一をはかり，定義上のミスからくるトラブルを一掃した．

2) ホストか，端末か……定義ミスの可能性が消えると，今度はホスト側ソフトウェアの問題か，端末側ソフトウェアなのか，またどの部分に問題があるのかという判断を強いられる．これは専門的な知識が必要となる場合もあるため，双方の担当SEが連絡を取り合って対処する必要がある．

3) 障害情報の採取手順の確認……業務稼働中のトラブルでは，障害情報の採取も迅速に行わねばならないため，手順書や障害情報採取JCL等を用意し，エンドユーザでも容易に情報採取できる環境を作っておく必要がある．

6.1.2 ハードウェア系

ハードウェア・トラブルにおいて最も危険なものがディスクのトラブルである．これは最悪の場合，ディスクの交換となるケースがある．現在は各地のユーザマシンのディスク交換後の修復作業はすべてセンタ側マシンよりRWSにて行っているが，作業項目が多いためオペレーション・ミスを起こす可能性がある．この種のミスの回避と，作業工数の削減のために，リカバリ用JCLの雛型を作成しておく必要がある．

6.1.3 ソフトウェア系

ベーシックソフトウェアの問題か，アプリケーション・プログラムが原因なのか見極めがむずかしく，リカバリが遅れる場合がある．また，業務中であったり，現象を再現させることがむずかしく，障害情報を取りにくい場合が多い．そのためオンライン・トラブル同様，障害情報採取手順書を作成し，ユーザ側に協力の依頼をしやすい環境を作る必要がある．

6.2 修正情報の適用

大規模なユーザになるほど，修正情報の適用は困難となる．修正情報の適用状況の違いから発生するトラブルを未然に防ぐためには，短期間のうちに全国のマシンに対して作業を行わねばならないが，一箇所あたり，1〜3日の工数がかかるため，実際には非常に困難である．修正情報の適用を円滑に行うためには，社内の支援体制を担当者レベルから，全社的な体制へと見直し，強化する必要がある．

6.3 OSのバージョンアップ

6.2節と同様，実際には困難な作業となるため，バージョンアップ作業JCL等を作成し，できる限りオペレーションの標準化を計れるような環境を作る必要がある．

7. おわりに

今回のさまざまな作業を振り返ってみると，他ユーザに比べて，マシン構成や回線利用形態等充実したものとなっており，アプリケーション部分でもさまざまな対策が施されている．

ユーザがますます多様化していく現状で，大規模な分散型のシステムは今後も増加していく傾向がある．このような状況において，ソフトウェア面からも，トラブル未然防止型の保守を前提とした標準的なシステムを組むことが，その後の作業工数を確実に減少の方向に導くポイントになると言える．

こういったアイデアは，ボリューム管理，データ管理等，よりコンピュータのベーシックな部分に関連する知識を身につけることにより，いっそう具現化していけるはずである．

最後に，本稿の執筆にあたり御協力をいただいた方々に感謝の意を表したい．

執筆者紹介　日　高　修　一（Shuichi Hidaka）

昭和33年生．57年職業訓練大学校電子科卒業，同年日本ユニシス(株)入社，オフィス・コンピュータ事業部配属後シリーズ8のシステム・エンジニアとしてシステム開発に従事．現在 九州支店システム2部所属．

寿　賀　徳　静（Tokusei Suga）

昭和36年生．61年同志社大学文学部社会学科卒業．同年日本ユニシス(株)入社，日本ユニシス情報システム(株)システム部配属，以後ユニシスシリーズ8でのシステム開発に従事．現在 関西支社ビジネスシステム1部所属．

通信ソフトウェア設計支援環境
——設計行為からの履歴情報獲得と修正支援機能

A Support Environment for Designing Communications Software
——Capturing of History Data from Design Activities and Modification Support Functionality

内田修市，平川　豊，門田充弘

要　約　大規模なソフトウェアの設計情報を構造化され電子化された情報として保存することは，システム設計論理の追跡性・理解性の向上のみならず，構造情報を利用した設計時・設計修正時の新たな計算機支援の可能性を有し重要である．

本稿では，まず設計情報とその間の関係を表現する設計履歴のモデルを紹介する．次に，信号シーケンス図作成ツール・SDL(Specification and Description Language)図作成ツールの操作から設計履歴情報を半自動的に獲得する手法を提案する．さらに，蓄積した設計履歴情報を活用した設計修正時の支援機能を提案する．

Abstract Preserving large-scale software design information in structured and computerized form is effective for the improved traceability and understandability of systems design logics as well as for a possibility of providing a new support environment when software designing and design modifications are conducted using such structured and computerized design information.

This paper first describes a design history model which represents design information and its inter-relationships, and then proposes a method by which to capture design history data in a semi-automatic way from design activities which include operations of a signal sequence editor and an SDL chart editor. In addition, another proposal refers to support functionality for design modifications with the use of earlier design history data.

1. はじめに

近年，より柔軟で高度なサービスへの高まりとともに，通信ソフトウェアは，大規模化・複雑化している．ソフトウェア設計過程では，各種の決定が行われ，ある一つの決定は，それ以降の一連の設計に影響を及ぼす．これらが積み重なり，設計情報間に複雑な関係が構成される．

しかし，現実のソフトウェア開発では，設計結果としてのドキュメントのみが残っており，そこに記述された内容の因果・参照・対応等の関係把握が困難である．このため，ドキュメントの論理的流れを把握することができず，システムを理解する上での大きな障害になっている．このような状況から，設計情報を理解しやすい形式で管理することが望まれている．

これらの問題を解決する技術として，ハイパーテキスト・ハイパーメディアの応用技術がある[1]~[3]．これらの技術は，複雑なシステムに埋め込まれた理論のつながりを計算機の支援により効率的に理解できる新たな情報管理技術としても位置づけられる．

本稿は，情報処理学会，ソフトウェア工学研究会資料67-3より学会の許可を得て転載したものである．

本研究は，設計行為より，設計情報と設計情報間の論理のつながり情報を獲得すること，およびそれらを後工程の支援に活用することを狙いとしている．

これを実現するには，以下の検討が必要となる．

1) 設計情報とその間の関係を表現するモデル
2) 設計者の設計行為より上記モデル形式の情報を獲得する方法
3) 蓄積された設計情報をもとにした設計支援・修正支援手法

これまでの研究[4]~[6]では，上記1)・3)に関して以下の提案を行っている．

・設計過程（履歴）を表現するモデル（情報成長モデル）の提案
・モデルの情報をもとに，設計時の状況に応じた支援方法の提案

本稿では，残された課題である上記2)の設計者の設計行為より設計履歴情報を半自動獲得する方法と3)の設計履歴を利用した修正時の支援方法について提案する．

2. 情報成長モデル(Information Growth Model)

支援の方法を述べる前に，その前提として提案済みの情報成長モデル(IGM)に関しての概要を説明する．

ソフトウェアの設計を目的システムについての情報を増加（成長）させる過程と見なす．各工程において設計者は，それまでの設計情報に新たな思考を加え，情報を付加する．IGMは，このような設計情報の成長過程を表現するためのモデルで，複数の情報boxとそれらのラベル，リレーションから構成される．

2.1 情報box

情報boxは設計情報のまとまりであり，同一の属性レコードを持った複数のレコードから構成される．情報boxには階層的なラベルが付加されているラベル付きboxとラベルなしboxがあり，ラベルにより各種の仕様書が階層的に分類される．ラベルなしboxは，設計情報間の関係を結び付ける中間情報を表現するものである．

2.2 リレーション

リレーションは，情報box間や情報box内データ間に対して定義され，かつ方向性を持っている．

リレーションは，非情報生成型リレーションと情報生成型リレーションの二種類に分類できる．

リレーションにより関係づけられる双方の情報に，明確なシンタックス関係が見い出せる場合は非情報生成型であり，抽出・連結・並べ変え・コピー・分割等の種類がある．

一方，リレーションで関係づけられた双方の情報に明確なシンタックス関係が見い出せない場合は情報生成型であり，抽象化・比較生成・詳細化・対応・変換等の種類がある．情報生成型リレーションは，人間の創造的活動履歴を表す．

2.3 情報成長モデルでの設計履歴表現例

具体例を図1に示す．図1は，知的電話機（相手の名前を呼べばつながる）の設計履歴をIGMにそって表現したものの一部である．

たとえば，box 1, 2, 4はラベル付きboxの例で，“/”により仕様書の階層を表現している．

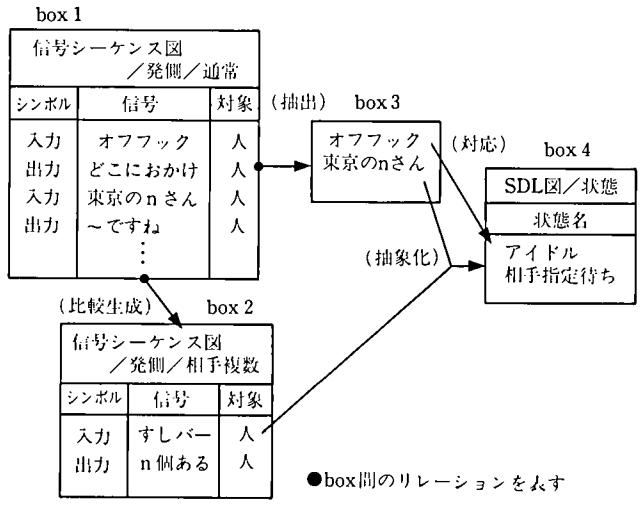


図 1　情報成長モデルでの設計履歴表現例

Fig. 1　Example of design history

box 1内の1レコード“入力　オフフック　人”は電話機に対して人からオフフック信号が入力されたことを示しており，box 1は電話機の通常通話のシーケンスを示している．

box 2は，box 1より比較生成されて作成されたことを示している．

box 3はbox 1より人からの入力信号のみを抽出したことを，またbox 4の状態レコードは，box 2・3の各々のレコード情報より，対応や抽象化により生成されたことを示している．

本稿では，リレーションに対して矢印の出発点となるデータまたはboxを「定義域」と呼び，到達点を「値域」と呼ぶ．リレーションは，複数の定義域（あるいは値域）を持ちうる．また，ある設計情報“x”と関係しているすべてのリレーションの定義域と値域の両者をさして，xの「関連域」と呼ぶ．

たとえば，図1の抽象化リレーションの定義域はbox 3“東京のｎさん”とbox 2“入力　すしバー　人”であり，値域はbox 4“相手指定待ち”である．また，box 3“東京のｎさん”は，抽出・抽象化リレーションと関連しており，その関連域は，box 1全体・box 2“入力　すしバー　人”・box 4“相手指定待ち”となる．

以上，例で示したように，IGMはわれわれの目に触れる「陽」の世界としての設計仕様書だけでなく，仕様書の影に隠れた思考履歴の一部や情報操作の一部，仕様書の中間情報等の「陰」の世界をも捉えたモデルである．

3. 設計履歴蓄積手法

本章では，信号シーケンス図作成ツール・SDL図作成ツールを用いた設計行為から半自動的にIGM形式の設計履歴を蓄積する方法について述べる．

図2は，設計者が支援環境上のツールを介して設計作業を行うことで，計算機が設計履歴を蓄積し，かつ履歴を利用して設計者の支援を行う形態を示している．

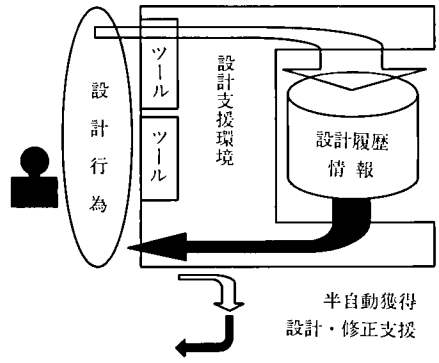


図 2 設計支援システム概念図

Fig. 2 Concept of design support system

これにより，ツール使用中に IGM を意識しない設計作業が可能となる．

手始めとして，主に通信ソフトの設計に用いる以下の作図ツール上での支援環境を検討した．

1) 信号シーケンス図作成

2) SDL 図の作成

以降では，上記の支援環境と設計履歴が蓄積される仕組みについて説明する．

3.1 信号シーケンス図作成

まず，設計者は自分の作成したい対象が，どのように動作すればよいかを明確にするために信号シーケンス図を作成する．

図 3 は電話機信号シーケンス図の一部である．

通常一つの設計対象システムにつき，設計者はケースごと（通話中のケース，突然受話器を置いたケース等）に複数の信号シーケンス図を作成する．本環境において，設計者は以下の制約を守ってシーケンス図の作成を行う．

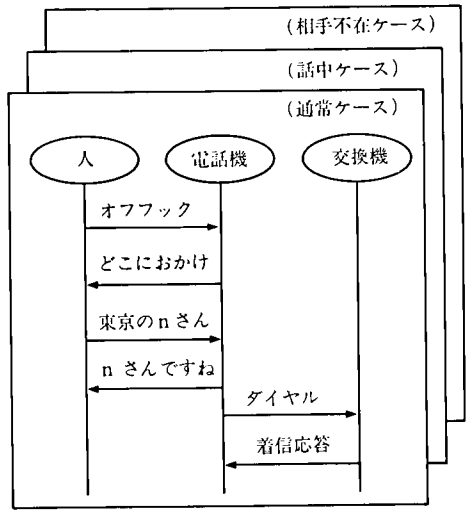


図 3 電話機信号シーケンス図

Fig. 3 Signal sequence of telephone

1)　複数の信号シーケンス図の中から一番類似している図を検索し，その差分だけを記述する．

2)　その際，検索された図より分岐するポイント，および合流するポイント（必ずしも開始ポイントで指示した同じ図とは限らない）を指定する．

このような制約上で信号シーケンス図を作成していくと，計算機内に図4 box 1～3のような設計履歴が構成される．

box 3は接続相手を指定しても，その電話番号が検索できないケースを，検索できたケース（電話機と人の通常の交信ケース）と比較しながら作成したことを示している．これらは，以下のような電話機の制御の流れを表している．

すなわち，人間が“オフフック”したら，電話機が“どこにおかけ”と尋ね，人間が“yさん”と答えると電話機が“該当なし”と返事をし，再び“どこにおかけ”と尋ねる動作である．

3.2 SDL図の作成

設計者は，対象システムに関しての動作を，わかりやすく体系だって把握するためSDL図を作成する．

SDL(Specification and Description Language)とは，CCITTで開発された電子交換用ソフトウェアの仕様記述言語で，グラフ表現，プログラム表現等が用意されている[7]．

3.2.1 信号シーケンス図から原始SDL図への変換

設計者がすべてのケースについての信号シーケンス図の作成終了を指示すると，計

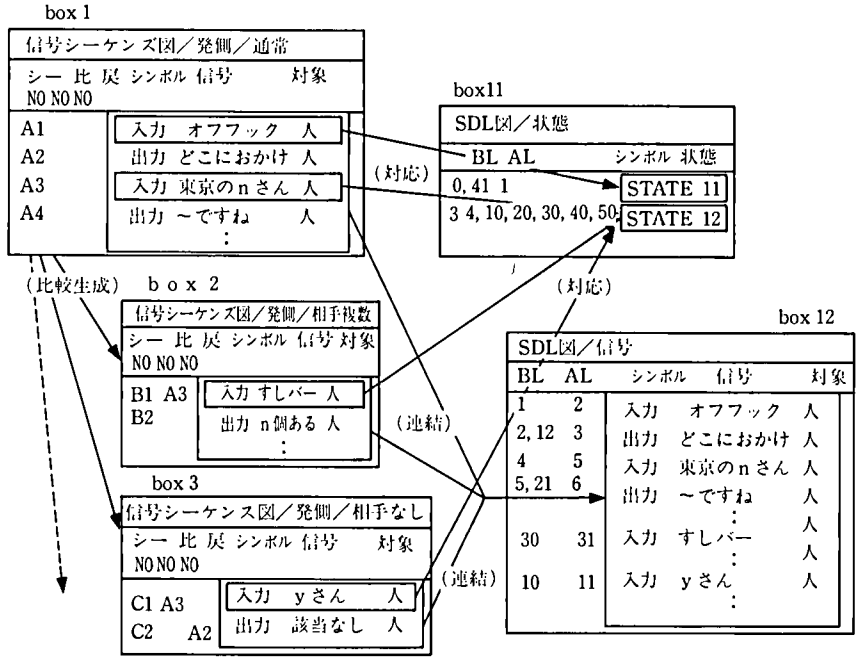


図 4　設計履歴(その1)

Fig. 4　Design history example 1

算機はどの対象に関してのSDL図を作成するか，設計者に尋ねてくる．ここで，設計者が「電話機」を指定すると，計算機は，これまでに作成したすべてのシーケンス図をもとに，図5のような電話機についての原始SDL図を作成する．

図はその一部であり，枝番号は以後の理解のために付記した．

なお，ここで作成される原始SDL図は，重複部分や不足部分が残っており，さらに修正・追加作業を行う必要がある．

原始SDL図の作成と同時に，計算機内では，図4 box 11，12のデータとその間のリレーションが構成される．

box 12はbox 1～3のレコードを連結し，入出力の枝番号を付け加えて作成したもので，“1　2　入力　オフフック　人”は入力シンボルのオフフックが一つの入力枝(1)と一つの出力枝(2)を持つことを示している．

box 11内の始めのレコードの状態番号11，12…は，box 1～3の入力シンボルに対応する所から，機械的に番号を割り振って作成されたものである．その時，比較生成リレーションの値域に相当するboxで，始めのレコードが入力シンボルの場合は，比較元と同じ番号を割り振る等の考慮を行っている．

3.2.2 原始SDL図の精錬

前節で作成された原始SDL図をもとに設計者は，SDL図の精錬作業を行う．ここでの精錬作業とは，シンボルの統合や不足部分の補填作業である．

この作業を支援するため，環境内にはSDLで使用するシンボルごとに，統合・挿入・変更・移動…等のコマンドが準備されている．設計者はこれらのコマンドをもとに計算機と会話を行いながら，最終的なSDL図に仕上げていく．

なお，この設計フェーズ内で設計者が新たな用語（新語）を定義したと認識した場合は，新語指定を行い，新語生成時の参照元を計算機に指定しなければならない仕組みになっている．これらの設計者による新語指定・参照元指定操作のことを，本稿では「新語処理」と呼ぶことにする．

以降では，精錬の三つの具体例に関しての操作環境と履歴の展開例を述べていく．

[例1　入力シンボル統合]

図5にSDLツール上での入力シンボル統合操作時の例を示す．設計者は，破線の丸で示された入力シンボルを統合するために以下の操作を行う．

まず，破線の丸で示した入力シンボルの統合処理を計算機に指示する．計算機は，統合後の入力シンボルのラベルの内容を設計者に尋ねてくる．そこで設計者は，“相手指定”を入力するとSDL図は，図6のように変化する．

この時蓄積される設計履歴は，図7のようになる．図7のbox 21では入力枝“100”と出力枝“101”が新たに作成されたことや，抽象化リレーションにより，“入力　相手指定　人”情報が生成されたことを示している．

box 41は，box 21生成の影響を受けて「STATE 12」の出力枝を変更したことを示している．

box 22もbox 21生成の影響を受けて，調整された入出力枝情報だけからなる，空データの条件シンボルが新たに作成されたことを示している．

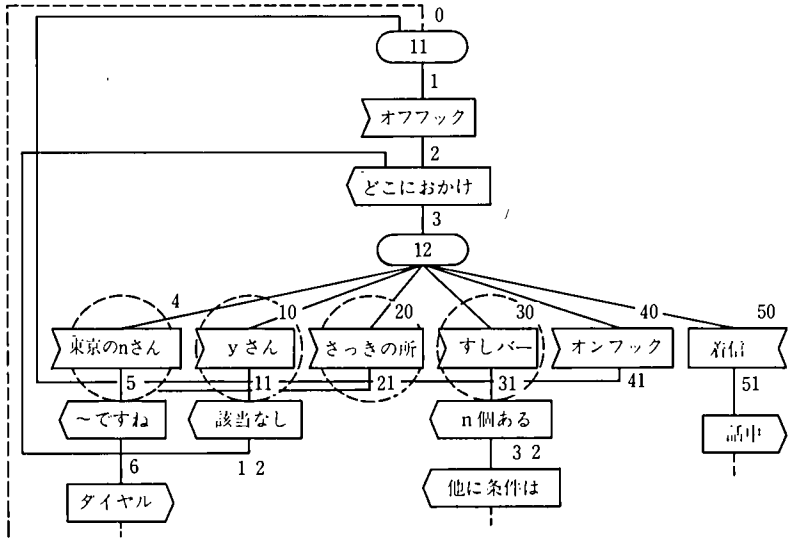


図 5　SDL図の一部(その1)

Fig. 5　SDL example 1

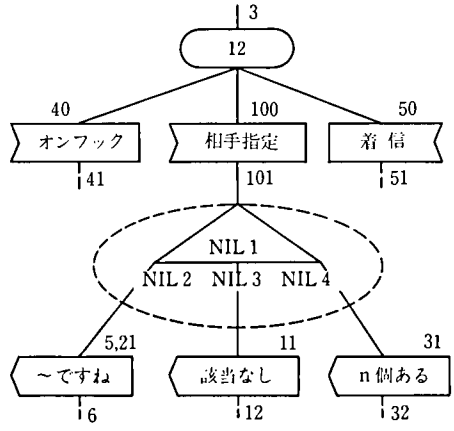


図 6　SDL図の一部(その2)

Fig. 6　SDL example 2

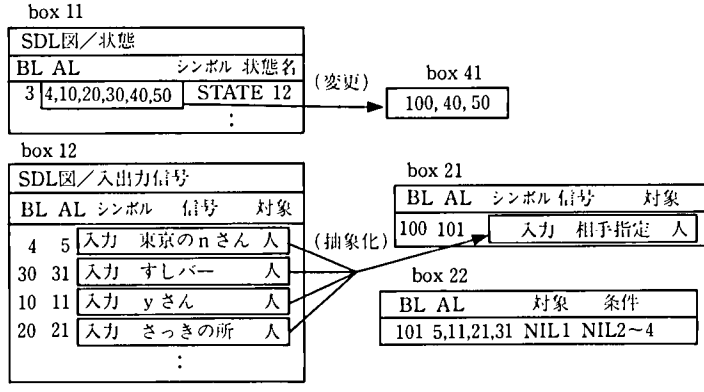


図 7　設計履歴(その2)

Fig. 7　Design history example 2

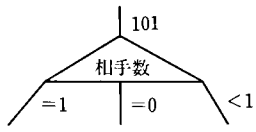


図 8 SDL図の一部(その3)
Fig. 8 SDL example 3

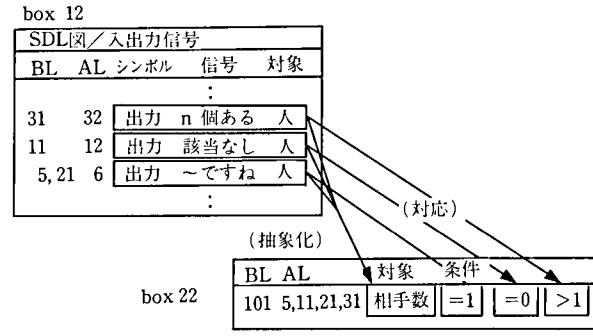


図 9 設計履歴(その3)
Fig. 9 Design history example 3

[例2 条件シンボル変更]

図6の破線の楕円で示した条件シンボルにラベルを設定する際の例を示す.以下に,操作例を記述する(図中では,便宜的に空ラベルを,NIL 1～4 を用いて識別している).

具体的には,NIL 1 を"相手数"に変更指示を行う.その際,文字"相手数"は,SDL上の他の部分から同じ文字をコピーするかタイプ入力して指定する.タイプ入力した場合は新語処理が駆動する.

この例では,タイプ入力し,SDL 図上の"～ですね"・"該当なし"・"n 個ある"部分を新語として入力した文字(相手数)の参照元として,指定した場合を示している.さらに,NIL 2～4 についても同様に,それぞれに"=1"・"=0"・">1"と変更する.

操作後の SDL 図および設計履歴を図8,図9に示す.

[例3 タスクシンボル挿入]

次に,設計者は入力シンボルの"相手指定"箇所と条件シンボルの"相手数"の間に,相手指定の情報から相手数や電話番号を検索するタスク記述挿入のために,ライン NO 101 を指定し,タスクの挿入指定をする.その後,計算機は,タスク記述に必要な項目(タスクの入力引数・出力引数・コメント)の入力指示を設計者に行う.

設計者は,タスク入力引数欄に"相手指定"を指定する.この際も,文字をタイプ入力した場合には,[例2]と同様に新語処理が駆動する.

以下,タスク出力引数欄も同様に"相手数"・"ダイヤル NO"・"相手名"の三つを指定する.さらに,コメント欄を入力した SDL 図を図 10 に示す.これに対応する履歴を図 11 に示す.ここでは新たに,box 23・24・42・43 が作成される.

以上のような枠組みで,グラフィック環境内に履歴作成作業の一部代替・設計履歴関係付けガイド機能を付加することにより,設計行為から IGM 形式の設計履歴の半自動的蓄積が可能となる.

次章では,この履歴を使用した修正支援環境に関して述べる.

4. 支 援 手 法

1章で述べたように,ソフトウェアの規模が拡大するにつれ,設計情報間の関係はさらに複雑化する.

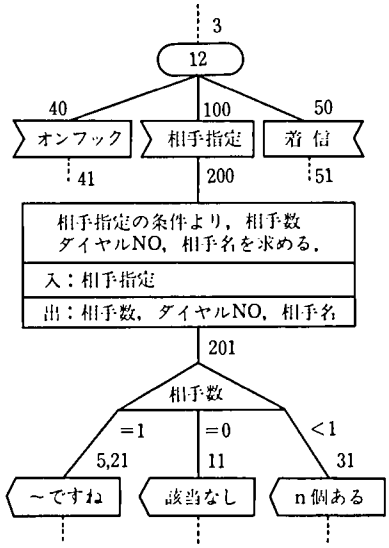


図 10　SDL図の一部(その4)

Fig. 10　SDL example 4

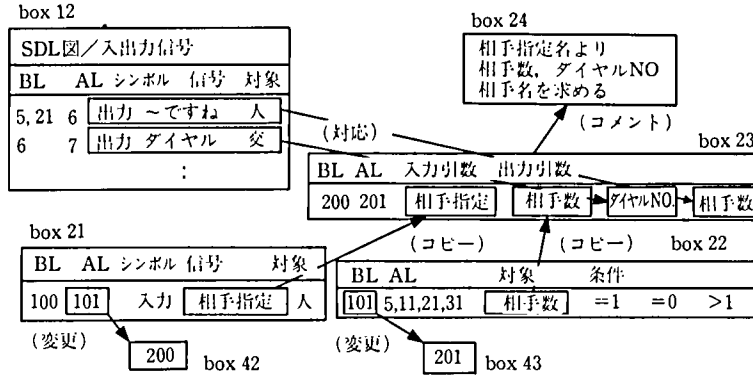


図 11　設計履歴(その4)

Fig. 11　Design history example 4

このような状況下，設計情報の修正時には，以下のような問題が発生する．

1)　他に必要な考慮部分の存在に気づかず修正作業を行い，そのためにまた別のトラブルが発生する．

2)　修正の繰り返しによる影響範囲が多岐にわたる場合，それを正しく把握できず，他で思わぬ不整合が生じる．

これらの問題は，以下の機能を計算機側に持たせることにより解決可能と考える．

1)　設計時の設計者と同じ考慮範囲を修正時にも表示させる機能

2)　修正による影響範囲の管理を行う機能

以下では，1)・2)の機能を総称して「修正支援機能」と呼ぶことにする．

計算機は，修正支援機能を以下の状況時に起動させる．

1)　設計者が修正支援機能の作動指示を行った場合

2) あるリレーションの定義域か値域に変化を及ぼすコマンド（設計情報の修正・削除・追加等のコマンドで，以下ではこれを総称して，「影響コマンド」と言うことにする）が実行され，修正者がその作動を希望した場合

4.1 設計者の考慮範囲表示機能

修正者が，影響コマンドを用いて設計情報の変更指示をした時に，計算機は，その修正指示された部分の関連域（本稿ではこれを「考慮範囲」と言うことにする）を表示することにより，修正支援を行う．

以下に，具体例を用いて支援の方法を説明する．

図12は前章までの履歴情報から，さらに設計が進行しbox 25～28が追加されている状態を示している．box 25はSDL図(box 12)の3箇所にわたる部分を抽象化したことを示している．これは，発信者からの一回目の相手指定だけでは，対応するダイヤルNOが捜し出せず，再度条件を入力し，接続する相手の対象を狭めている所に対応する．

box 26は，さらにその指定の内容から対応する条件部を作成したことを示している．また，box 27は接続相手を絞り込むために何度も入力された相手指定の情報が入っている条件入力テーブルから，該当する相手数，それに対応した複数のダイヤルNO，相手名，案内文を生成するタスクシンボルを挿入したことを示している．

box 28はbox 23で定めた機能の一部形式化を行っていることを示している．（なお，図12はその理解性を考慮し，表現に一部修正を加えている）

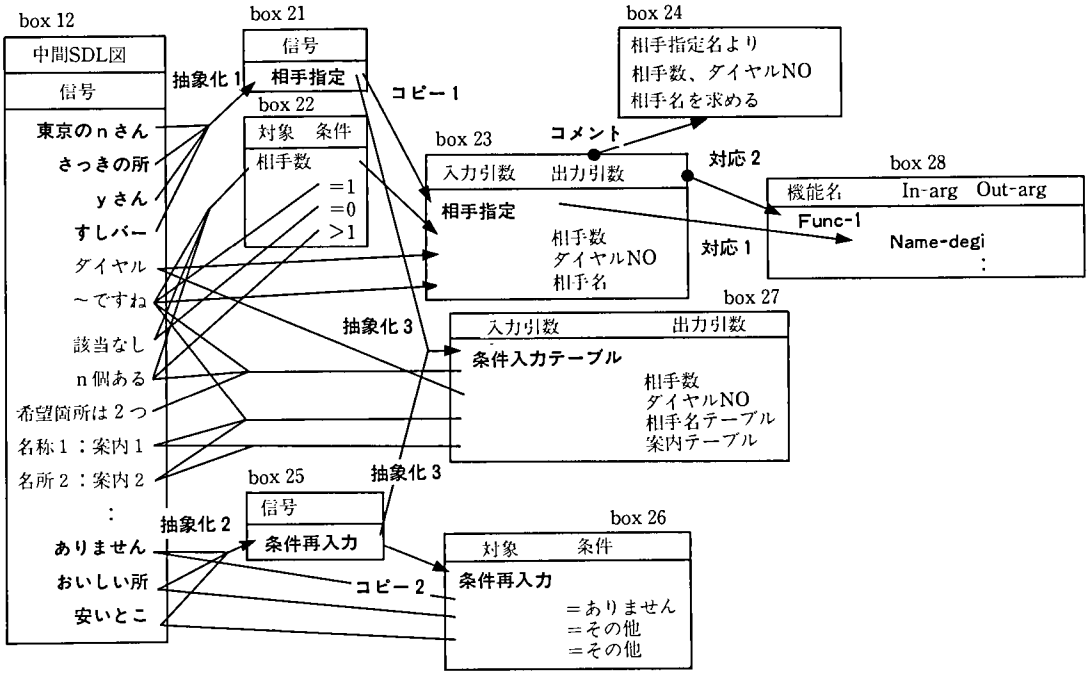


図 12 設計履歴(その5)

Fig. 12 Design history example 5

このような状態で，修正者がSDL作図環境上で，box 21の“相手指定”部分の修正指示を行ったとする．

box 21の“相手指定”部分は定義域でもあり，値域でもあるので，計算機は修正支援機能を作動させるか否かを修正者に問う．修正者が作動させる意志を計算機に示すと，計算機は修正支援機能を作動させ，別途ウインドゥをオープンし，表1上部に示した関連域の情報を表示する．

また，box 25の“条件再入力”修正時に表示される関連域を表1の下部に示す．

このように，設計時に作成されたリレーションを利用すると，修正時の状況ごとに，考慮すべき範囲を適宜表示することができる．これによって，修正時の検討漏れを防止することが可能となる．

表1　修正考慮範囲表
Table 1　Consideration scope

修正指示部		表示関連域		
box-No	設計情報	リレーション	box-No	設計情報
21	相手指定	抽象化1	12	東京のnさん
				さっきの所
				yさん
				すしバー
		抽象化3	27	条件入力テーブル
			25	条件再入力
		コピー1	23	相手指定
25	条件再入力	抽象化2	12	ありません
				おいしい所
				安いとこ
		抽象化3	27	条件入力テーブル
			21	相手指定
		コピー2	26	条件再入力

4.2　修正影響範囲の管理機能

修正者が影響コマンドを実行すると，計算機はこれに伴う影響範囲をリレーションから決定し，その情報を蓄積する．計算機はこの蓄積された情報を基に修正者とインタラクションを行いながら，修正者のガイドを行う．以下にそのガイド機能の詳細を述べる．4.1節と同じく図12を用い，この機能の説明を行う．

修正者がSDL作図環境上で，box 21の“相手指定”部分を“条件入力”に変更することを計算機に指示し，修正支援機能を作動させたとする．

このとき計算機は，修正指示されたデータと非生成型のリレーションで関連している定義域または値域がある場合，そのデータも一緒に自動修正し，その旨を修正者に示す．ここでは，box 21の“相手指定”とコピー1で関連しているbox 23の“相手指定”部分が該当する．

また，計算機は，影響コマンドおよび自動修正により変更されたデータ（修正源と呼ぶ）ごとに，影響部の情報を蓄積する．

影響部とは，修正源と生成型リレーションで関係している関連域のことである．

前述の変更例をもとに，修正源，影響部を表2に示した．表2では，二つの修正源

表2　修正影響範囲管理表
Table 2　Effect scope

修正源		影響部		
box-No	設計情報	リレーション	box-No	設計情報
21	相手指定	抽象化1	12	東京のnさん
				さっきの所
				yさん
				すしバー
		抽象化3	27	条件入力テーブル
			25	条件再入力
23	相手指定	対応1	28	Name-desi
		コメント	24	box全体
		対応2	28	Func-1

が発生し，それに対応して二つの影響部の情報が蓄積されている．

計算機は，設計者に表2の影響部の設計情報を関連するリレーション単位に順次表示する．修正者は表示された関連リレーションのまとまりごとに，修正の必要性を判断し，なければその旨を計算機に指示する．この際，新たな修正が加わった場合には，修正源・影響部の情報が計算機内に新たに蓄積される．修正がない場合には，計算機は，表示していたリレーションに関するすべての情報を影響部から削除する．

上記操作は，修正源ごとにその影響部の情報が空になるまで繰り返される．すべての修正源の影響部が空になれば，計算機は修正作業が完了したと見なす．

この例での範囲は小さいが，修正の繰り返しによる影響範囲が多岐にわたる場合でも，上記修正範囲管理機能により，修正作業による見落とし・不整合の減少が期待できる．

5. おわりに

設計者の設計行為から設計履歴を半自動的に蓄積する方法，およびその情報を利用しての修正支援方法について提案した．

今後の主な課題を以下に示す．

1) 設計作業進行にともない複雑化するリレーションを含めた情報管理の方法
2) 設計履歴を再利用する枠組の設定
3) 設計履歴情報獲得の全自動化
4) 作成中のプロトタイプを完成させ，システムを評価する

本研究に際し，設計支援環境の問題点を整理する上で葉原耕平国際電気通信基礎技術研究所副社長に貴重な御意見を頂いた．また，熱心な討論をして頂いた(株)ATR通信システム研究所山下部長，通信ソフトウェア研究室各位に感謝の意を表したい．

参考文献 [1] Communications of the ACM, Special Issue: Hypertext, Vol. 31, No. 7, July 1988.
[2] J. Conklin, "Hypertext: An Introduction and Survey", Computer, Vol. 20, No. 9, September 1987.
[3] C. Potts and G. Bruns, "Recording the Reasons for Design Decision", Proc. 10th

Int. Conf. Softw. Eng., 1988, pp. 418～427.
[4] 内田修市, 平川　豊, 門田充弘, "思考履歴を利用した設計支援環境の考察", 情報処理学会, 第37回全国大会, 4 L-6, 1988.
[5] Hirakawa, Uchida, Monden, "Communication Software Design Support Environment Based on The Infomation Growth Model", 1988 Joint Conf. on Commun. Net and swit. Sys., pp. 47～52 (信学技法, SSE88-124).
[6] Hirakawa, Uhida, Monden, "Hypermedia as a Communication Software Design Support System" Proceedings on Hypertext2, York, June 1989.
[7] CCITT Recommendations z. 100-z. 104, red book 1984.

執筆者紹介

内　田　修　市 (Shuichi Uchida)

1957年生. 1980年九州産業大学 経営学部産業経営学科卒業, 1980年日本ユニシス(株)入社, 1986年(株)ATR通信システム研究所 通信ソフトウェア研究室研究員(出向), 1989年日本ユニシス(株)関西支社教育部, CAIシステム開発, UNIX関連教育を担当, 情報処理学会会員.

平　川　　豊 (Yutaka Hirakawa)

1956年生. 1980年NTT武蔵野電気通信研究所入社, 1988年(株)ATR通信システム研究所　通信ソフトウェア研究室主任研究員(出向中), 通信ソフトウェアに関する基礎研究に従事, 電子情報通信学会, IEEE Computer 各会員.

門　田　充　弘 (Michihiro Monden)

1946年生. 1969年広島大学 工学部電気工学科卒業, 1969年NTT入社, 1986年(株)ATR通信システム研究所　通信ソフトウェア研究室室長(出向), 1989年NTT交換システム研究所, 電子交換機およびパケット交換機のソフトウェア開発に従事, 電子情報通信学会, 人工知能学会各会員.

フォールト修正文に基づくフォールト混入工程の分析

The Analysis of the Fault mingling Process Based on Fault Correction Statements

毛 利 幸 雄

要 約 本稿では，ジャクソン法（JSP法）に従ったソフトウェア開発を対象として，いわゆるバグに相当するフォールトを分析し，フォールトがプログラム中に混入した工程を求める分析手法を提案する．ここではJSP法の開発過程で開発者が行うフォールトの発見，除去の作業はフォールトリポートとして報告されるものと仮定する．提案する手法ではこのフォールトリポート中のフォールト修正文の詳細な分析を行い，フォールト修正の目的(修正理由)を明らかにする．次に，その理由に基づいて，フォールトの混入した工程を求める．

さらに本手法の有効性を確認する目的で，当社の新入社員研修において適用実験を行ったので，その結果についても報告する．

Abstract Focusing on JSP-compliant software development, this paper proposes a fault analysis procedure by which to trace the process where a fault has mingled in a program through analyzing the fault comparable to what is called a bug. What is assumed here is that fault discovery and deletion by a software developer in the course of JSP-compliant software development becomes known in the form of a fault report. The proposed procedure is suited to make a minute analysis of fault correction statements in the fault report in order to clarify the purpose of (or the reason for) fault correction. The next step is to trace the process, on the basis of the reason, in which the fault has mingled.

This paper also reports on the successful results obtained from the experimental application of the procedure to the training programs for new employees of Nihon Unisys, Ltd., which was carried out just to see its usefulness.

1. はじめに

最近，高信頼化システムへの期待と共に，ソフトウェアの品質保証技術への関心が高まってきている．ソフトウェア開発現場ではソフトウェア信頼度成長曲線が広く利用されている[3]．この曲線は，通常コーディング以降の単体テスト，結合テスト工程に適用され，プログラム中に混入したフォールトの総数，残存しているフォールトの総数の推定を行う．

一方，ソフトウェアの品質保証のためには，開発工程の初期段階でフォールトの発見，除去を行うことが重要である．とくに設計レビューの必要性が認識され，その実態について多くの報告がある[2][4][6][7]．

本研究の最終的な目標は，設計レビュー，コードレビューがフォールト発見，除去の立場から有効であることを実験的に示すことにある．ここでは，フォールトの発見，除去の具体的な作業内容を分析し，各フォールトが混入した開発工程とそれを発見，除去すべき工程（設計レビュー，コードレビュー，テスト等）を明示することを目指す．

ただし，本稿ではジャクソン法(JSP 法)[1]に従ったソフトウェア開発を前提とした議論を行うものとする．

開発者は，データ構造からプログラム構造を導く JSP 法の開発過程でフォールトリポートを作成するものとする．得られた主な成果は以下にまとめられる．

1) JSP 法に基づいたソフトウェア開発の各工程から正確なデータ収集を行うための作業記入用紙，フォールトリポートの形式，およびそれらの記入方法を定めた．
2) フォールトリポートに記入されたフォールト修正内容を分析し，フォールトが混入した開発工程を決定する手順を提案した．
3) 提案した分析方法を実際のソフトウェア開発に適用し，方法論としての妥当性を確認した．

IEEE Standard の定義では，ソフトウェア開発作業において人間が犯す誤りをエラー(error)，エラーがソフトウェア中に具体化したものをフォールト(fault)と呼ぶ．本稿中で使用しているフォールトとは，IEEE Standard に従うものとする．

2. JSP 法に従ったソフトウェア開発

2.1 開 発 過 程

JSP 法に従うソフトウェア開発ステップの概要を表 1 に示す．ただし，表 1 では構造の不一致[5]が発生する場合は除外している．フォールトを中心に見ると，本開発過程はフォールトの混入工程と混入したフォールトの発見，除去の工程に分けられる．

表 1　JSP 法に従う開発工程
Table 1　The JSP software development process

Step	フォールト混入工程	フォールト発見工程
S 1	仕様の理解	
S 2	入出力データ構造図の作成	
S 3	プログラム構造基本図の作成	
S 4	変数の列挙	
S 5	オペレーションの列挙	
S 6	オペレーションの割付	
S 7	プログラムの最適化	
S 8		設計レビュー
S 9	コーディング	
S 10		コードレビュー
S 11	テストデータの作成	
S 12		単体テスト
S 13		結合テスト

各 Step における主な作業内容は次の通りである．

Step S 1（プログラム仕様の理解）……仕様を読んで，要求される処理内容と使用するファイル情報を把握する．

Step S 2（入出力データ構造図作成）……仕様書で要求されている出力データと入力データの構造図を作成する．入力ファイルが複数ある場合には論理入力データ構造を作成し，一つにまとめる．

Step S 3（プログラム構造基本図作成）……入力データ構造と出力データ構造間の対応関係を見つけ，それをプログラム構造基本図として作成する．

Step S 4（変数の列挙と関連づけ）……入力レコードから出力レコードへと変換される過程で必要となる変数を列挙し，変数関連図にまとめる．

Step S 5（オペレーションの列挙）……仕様書の処理内容と変数関連図から，必要なオペレーションと条件式を列挙する．

Step S 6（オペレーションの割付）……プログラム構造基本図に，初期処理と後処理に対応する基本要素を付け加え，オペレーションと条件式を割付ける．

Step S 7（プログラム構造の最適化）……命令が割付けられなかった基本要素を除去する．可能であればいくつかの基本要素列を一つにまとめる．

Step S 8（設計レビュー）……求めたプログラム構造図に対し第三者がレビューを行う．

Step S 9（コーディング）……プログラム構造図に従い，COBOL 言語でコーディングする．

Step S 10（コードレビュー）……作成されたコードについて第三者がレビューを行う．

Step S 11（テストデータ作成）……仕様書の要求を満たすテストデータを作成する．

Step S 12（単体テスト）……テストデータに対するプログラムの出力結果が仕様書通りであることを検査，確認する．

Step S 13（結合テスト）……ジョブを構成する他のプログラムとの整合性を確認する．

2.2 開　発　例

ここでは図 1(a)に示す仕様書を用いて開発過程の Step S 2～S 7 の説明を行う．したがって，Step S 1 の結果が図 1(a)であると考える．まず Step S 2 で図 1(b)，(c)に示す入出力データ構造図を作成する．

次に，Step S 3 で図 2(a)のプログラム構造基本図を作成する．

次に，Step S 4 では今の場合，10 個の変数を求める．Step S 5 では，表 2 に示す 12 種類のオペレーションと表 3 の条件式を求める．引き続き，Step S 6 で図 2(a)の構造基本図に対し，表 2 と表 3 の結果を割付けて図 2(b)を得る．最後に，Step S 7 で最適化を行った結果，図 2(c)のプログラム構造図が求まる．

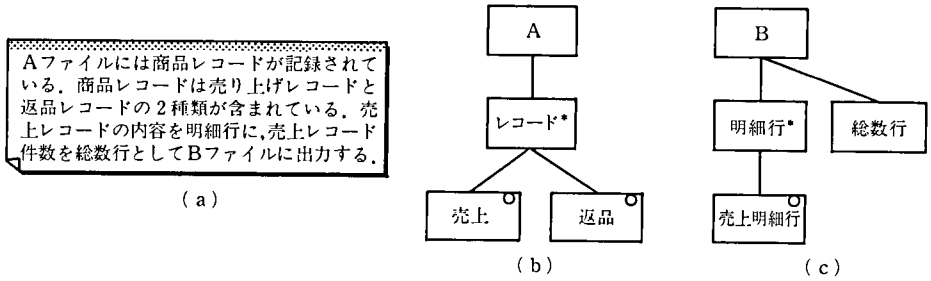


図 1　プログラム構造図作成のための情報

Fig. 1　Fig. 2 Application of Step S2

表 2　命令の分類

Table 2　Set of instruction

1	出力ファイルのオープン
2	総数行の編集
3	総数行出力
4	売上明細行の編集(a1, a2, a3)
5	売上明細行出力
6	出力ファイルのクローズ
7	売上レコード件数の累計
8	売上件数の 0 クリア
9	入力ファイルのオープン
10	入力レコードの読み込み
11	入力ファイルの終了処理
12	入力ファイルのクローズ

表 3　条件の分類

Table 3　Conditional expressions

C1	入力ファイルが終了するまで
S1	区分＝1
S2	区分≠1

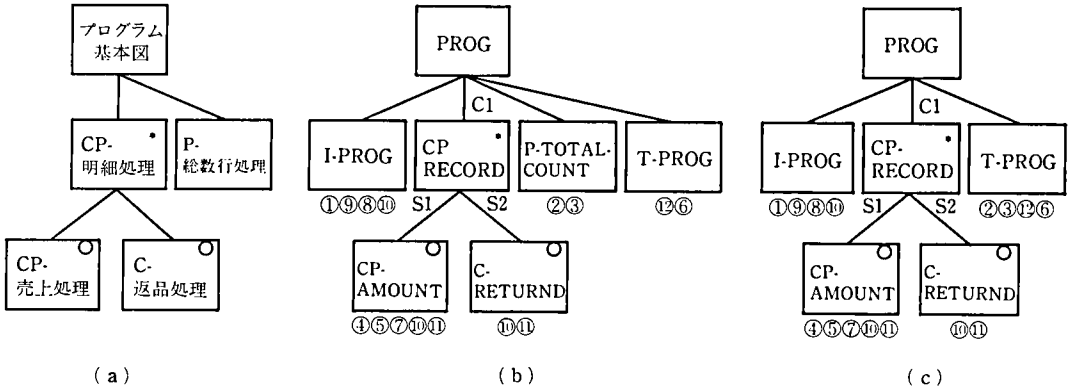


図 2　プログラム構造図の作成課程

Fig. 2　Application of Step S3, S6 and S7

2.3　プロダクトの依存関係

JSP 法に従ったソフトウェア開発においては，多くの種類のプロダクトが生成されるという特徴がある．これらのプロダクトの間には図 3 に示す依存関係が存在する．

2.1 節でも述べたように，JSP 法の場合，プログラムの構造と機能を仕様書からほぼ独立に求めた後，両者間の対応関係に基づいてプログラムを作成する．この流れは図

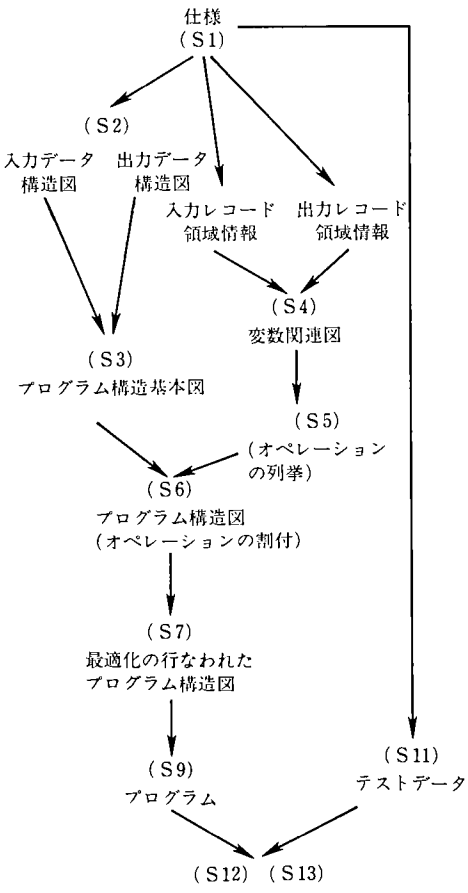


図 3　プロダクトの依存関係

Fig. 3　Dependencies among software products

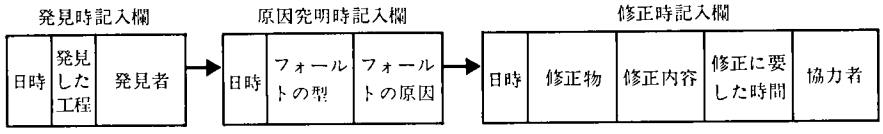


図 4　フォールトリポート

Fig. 4　Fault report

3からも明らかである．したがって構造と機能の両方が正しく，かつ対応関係も正しい場合にのみ正しいプログラムが構造図として作成できる．

3. フォールト情報の収集

フォールト情報の収集に当たっては作業記入用紙とフォールトリポート紙を用いた．作業記入用紙には作業時間とその作業内容を書く．

さらに，バグ分析のため，フォールトリポートを作成する．図4に示す流れにそっ

表4　フォールトの型
Table 4　Types of fault

1	入力データ構造図
2	出力データ構造図
3	プログラム構造図
4	オペレーション
5	データ定義エラー
6	入力処理エラー
7	初期化エラー
8	演算処理エラー
9	条件判定エラー
10	インタフェースエラー
11	出力処理エラー
12	構造部記述
13	改善・改良
14	テストデータ準備誤り

表5　フォールトの原因
Table 5　Cause of fault

1	仕様書の誤り
2	仕様書の読解・理解不足
3	JSPの記法誤り
4	プログラム構造図の作成誤り
5	コーディングエラー
6	ジョブ制御文の知識不足
7	言語文法知識・理解の不足

表6　修正物
Table 6　A list of products

1	仕様書
2	設計図
3	プログラム
4	テストデータ

て，フォールト修正内容を記入する．具体的にはフォールトの発見方法，フォールトのタイプ，フォールトの原因等を記入する．

フォールトのタイプは，表4に示すように1〜4の構造上の誤りと5〜13のコーディング上の誤りに大別される．以降では，表4の内容をtで，表5の内容(フォールトの原因)をrで，表6の内容（修正物）をcで参照する．

4. 提案する分析方法

ここでは，JSP法に従ったソフトウェア開発過程を前提とした分析方法を提案する．

4.1 基本方針

表7に分析方法の概要を示す．本方法はStep A 1〜A 4の四つのステップから構成される．まず，Step A 1ではフォールト修正文を表8に示す15種類に分類する．

次に，Step A 2では，修正物(c)とフォールトのタイプ(t)を用いて，修正理由を求める．具体的には，表9に示すように，構造，機能，対応関係の三つの観点から，フォールト修正理由を4種類に分類する．

Step A 3では表9の分類結果として得られた修正理由Rを利用して，フォールトの修正理由をさらに詳細に決定する．

最後にStep A 4では，Step A 3で求めた混入工程の求まった各フォールトに対し，そのフォールトを発見すべき工程を求める．

4.2 Step Al（C表現への変換）

仕様書は，今回の議論では誤りが無いものと仮定して表6から除外する．したがって，修正対象物は設計図，プログラム，テストデータの3種類である．さらに，JSP法を採用していることから，設計図は構造，条件と，割り付けられる命令の三つの構成要素を含む．プログラムについてはCOBOLを仮定しているので，本質的な構成要素はデータ定義部と手続き部である．手続き部については，設計図の構造部，条件，割付けられた命令の実現部分である．これらを整理することにより，表8に示す5種類

表7　提案する分析方法
Table 7　Fault analysis procedure FA

Step	フォールト分析内容
A1	フォールト修正文のC表現への変換
A2	修正理由の分類
A3	フォールト混入工程の決定
A4	フォールト発見工程の決定

表8　Step A1
Table 8　Fifteen actions(Step A1)

	追加	削除	変更
命令の修正	a1	d1	u1
条件の修正	a2	d2	u2
構造の修正	a3	d3	u3
データ定義の修正	a4	d4	u4
テストデータの修正	a5	d5	u5

表9　Step A2
Table 9　Four reasons(Step A2)

プログラム構造基本図の作成	オペレーションの列挙	オペレーションの割付	修正理由R
Y	Y	Y	A
		N	B
	N	—	C
N	—	—	D

の構成要素が得られる．

一方，修正の一般的な機能分類として，変更，追加，削除を採用する．以上により，最終的にフォールト修正文の修正内容に基づいて，表8に示す15種類の分類が得られる．これをC表現と呼び，記号Sで表現する．

4.3　Step A2（修正理由の決定）

図3に示すように，JSP法で中心となる作業は①構造の作成，②オペレーションの列挙，③オペレーションの割付け，の三つである．これらの各作業が正しく行われたか否かの観点から，フォールトに対する修正理由を分類する．その結果，表9に示す四つの分類が得られる．

表9の分類基準について述べる．まず条件式〔s∈ {a3, d3, u3} ∧t≠13〕が成立するとき，かつその時に限りプログラム構造基本図の作成をNとする．

次に，条件式〔c= {2} ∧s∈ {a1, a2, d1, d2, u1, u2, a4, d4}〕が成立するとき，かつその時に限りオペレーションの列挙をNとする．

最後に，条件式〔[c∈ {3, 4} ∧t∈ {5, 6, 7, 8, 9, 10, 11, 12, 14}]∨[c∈ {2, 3} ∧t= {13}]〕が成立するとき，かつその時に限りオペレーションの割り付けをYとする．

4.4　Step A3（フォールト混入工程の決定）

修正理由をさらに細分化すると同時に，エラー原因との対応を考慮して，フォールト混入工程を決定する．その決定の詳細を表10に示す．

4.5　Step A4（フォールト発見工程の決定）

JSP法では，プログラム設計工程の依存関係から明らかなように，前工程の出力が次の工程の入力として採用されている．したがって，フォールト混入工程を明らかにすれば，レビューを行ってフォールトを発見すべき工程を決定することができる．具体的には，表11に示すようなフォールトの検出が可能となる．

ここでの議論では，仕様書が正しいことは仮定されている．したがって，単体テスト終了時点で正しければ，仕様書が要求している通りのプログラムが完成しているも

のと考えられる．もし，準備するテストデータ以外で，単体テストまたは結合テストにおいて発見されるフォールトが存在するならば，仮定に反して仕様書に誤りが存在したことになる．

表10　Step A3
Table 10　Decision at Step A3

R	Rに関する条件	r (エラー原因)	P0 (混入工程)
A	t={14} c={4}		S11
A	t={13} c={3}		S9
A	t={13}		S7
B			S6
C	s∈S-{a4, d4}		S5
C	s∈ {a4, d4}		S4
D	t={3}	r∈{3, 4}	S3
D	t⊆{1, 2}	r∈{3, 4}	S2
D	t⊆{1, 2, 3}, E(d)={u3}	r={2}	S1

s= {a1, ⋯, a5, d1, ⋯, d5, u1, ⋯, u5}

表11　Step A4
Table 11　Final decision at Step A4

P0	P1 (発見工程)
S1	S8
S2	S8
S3	S8
S4	S8
S5	S8
S6	S8
S7	S8
S9	S10
S11	S12
—	S13

5. 適　用　例

典型的な二つの場合について，具体的な例を用いて分析方法の適用を説明する．なお，これらの例は，いずれも6章で述べる適用実験で収集した実際のものである．

5.1 構造に誤りがある場合（例1）

ここでは，フォールト修正文「資格レコードの処理ができるようにプログラム構造図を修正した」を考える．さらに発見工程がStep S 8の設計レビューであったと仮定する．またエラーの型はt= {3} で，修正物c= {2} で，エラー原因はr= {4} とする．

この事例に対して，提案する方法論を適用した結果について述べる．まずStep A 1では，修正対象が構造図そのものなので，C表現としてs=u 3を求める．次にStep A 2では，条件 [u 3∧c= {2} ∧t≠ {13}] が成立するので，表9より修正理由としてDを選ぶ．次にStep A 3では，表10より条件 [D∧t= {3} ∧r= {4}] が成立するので，Step S 3で混入したという結論を得る．最後にStep A 4では，表11よりStep S 8で発見されるべきであると決定する．

一方，プログラム構造基本図を調べてみたところ，開発者がプログラム構造基本図を作成する際に，入力データ構造図の資格項目ごとのレコード分割を抜かしていたことが判明した．したがって提案する分析法による分析結果と一致した．

5.2 命令の割り付けに誤りがある場合（例2）

ここでは，フォールト修正文「洩れなく初期処理に入力命令を記述する」を考える．さらに発見工程がStep S 12の単体テストであったと仮定する．またエラーの型はt= {6} で，修正物c= {2, 3} で，エラー原因はr= {4} とする．

この事例に対して提案する方法論を逢用した結果について述べる．まずStep A 1では命令の記述洩れであるので，C表現としてs=ulを求める．次にStep A 2では，条件 [ul∧c= {2, 3} ∧t≠ {13}] が成立するので，表9より修正理由としてBを選ぶ．

次にStep A 3では，表10より条件［B］が成立するのでStep S 6で混入したという結論を得る．最後にStep A 4では，表11よりStep S 8で発見されるべきであると決定する．

一方，プログラム構造基本図を調べてみたところ次のことがわかった．開発者が，プログラム構造図の初期処理の基本要素として，初期入力のための入力命令の割り付けを行っていなかった．したがって，提案する分析法によるフォールト混入工程の分析結果と一致した．

6. 適 用 実 験

提案した分析法（表7）の妥当性を確認する目的で，当社新人研修における1100 COBOLクラスの卒業演習の場を利用して適用実験を行った．

6.1 実 験 対 象

まず，開発チームについて説明する．開発チームは全体で24チームである．各チームは4人から5人で構成されている．データ収集期間は1989年8月9日から20日間である．

次に開発対象システムについて説明する．事務処理用バッチ・システムであり，表12に示す18個のモジュールから構成される．

作成すべきモジュールの各プログラマへの割当は各チームに任されている．表12の構造の複雑さは，各モジュールの持つ選択要素数と繰り返し要素数の合計値を示す．同表の処理部の平均ステップ数は，手続き部のコメント行と段落行を除いた行数の平均値を示す．同表の難易度判定は，構造の複雑さと平均ステップ数，ならびに仕様書

表12 プログラムの難易度
Table 12 Difficulties of programs

難易度	構造の複雑さ	処理部の平均ステップ数
A	11	81
A	9	83
A	9	69
A	8	63
A	6	52
A	5	49
B	8	88
B	8	82
B	6	30
B	5	53
B	5	48
B	3	34
C	3	26
C	2	26
C	1	26
C	1	16
C	1	14
C	1	13

で要求される処理内容の複雑さを加味して求めた値である．難易度は高いものから順にA，B，Cで表示した．

6.2 収集されたフォールト

各レビューにおいて収集されたフォールトを図5に示す．図中の無回答のデータはフォールトリポートに記入がなかったものを示す．これらの無回答データは分析対象から除外した．

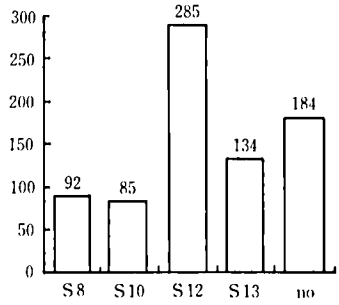


図 5　各発見工程で実際に発見されたフォールト
Fig. 5　Fault detection (experimental data)

6.3 分析法適用結果

まず，Step A 3での分析結果について述べる．Step A 3で求まる混入工程別のフォールト数を図6に示す．同図で横軸はフォールト混入工程を，縦軸は各工程で混入されたフォールト数を示している．

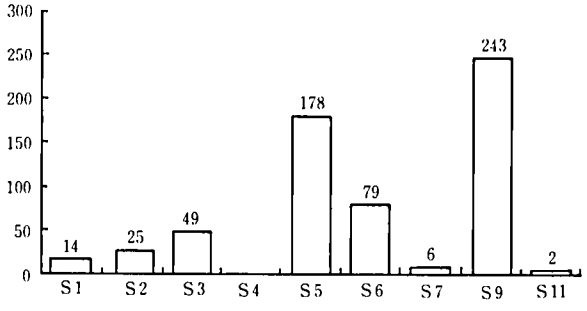


図 6　フォールト混入工程分析結果
Fig. 6　Fault introduction (analysis result)

今，フォールト混入工程Step S 5に注目すると，この工程で混入したフォールトの総数が178である．これらのフォールトは，表11によると，発見工程Step S 8の設計レビューで発見されるべきである．しかし，現実には18個だけがStep S 8で発見されている．残りについてはStep S 10で25個，Step S 12で97個，Step S 13で38個が発見されている．設計レビューが不十分であったことを示している．なお，Step S 4でフォールト数が0であるのは，変数関連図の作成が行われていなかったためと判明した．

次に，Step A 4 の分析結果について述べる．Step A 4 で求まる発見工程別のフォールト数を図 7 に示す．

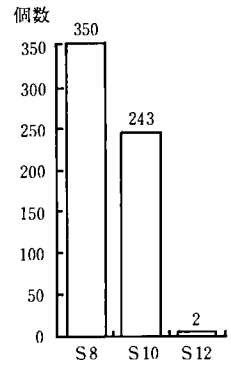


図 7　各発見工程において検出すべきフォールト数
Fig. 7　Fault detection (ideal case)

今回発生したフォールトについては，バッチシステムを対象にし，かつ仕様書が正しいことを前提にしている．したがって混入したフォールトはすべて単体テストまでに発見可能なため，今回設計対象としているようなバッチ処理システムにおいては，結合テストは確認のため実施するだけでよいことになる．

今回の分析結果から考えると，もし結合テストが必要なフォールトが存在すれば，その根本的原因はフォールト発見工程の作業内容の不備と，すべての前提となる仕様書の内容の不備にあると言える．

7. おわりに

本研究では，ソフトウェア開発現場の作業記録に基づいて，フォールトの混入工程を決定する試みについて述べた．今後の課題としては次の項目が挙げられる．

1) フォールトリポートから求まるフォールト修正文は，文法的に不完全な文となっていることが多い．今回の実験では，NUPS 法[5]の考え方を利用して各修正文の目的語と動詞を明らかにした．この前処理の部分を分析方法に組み込むことが今後の課題となる．

2) 今回の適用実験においては，開発者に対するインタビューによって分析結果の正しさを確認することを行っていない．そこで，適切な形式のインタビューを導入して分析方法の各ステップの詳細部分の改良を図ることも重要である．

3) フォールト発見工程の作業内容を細かく規定していないため，レビューは各チーム任せになっている．今回の分析結果に基づいて，各レビュー時に見逃されているフォールトの傾向を整理し，各レビュー内容を規定することは重要な今後の課題である．

4) 今回提案した分析法では 2.1 節で述べたように構造の不一致を除外している．今後の課題として，構造の不一致が発生するような場合にも対応できるように分析法を拡張することについて検討する予定である．

本実験の実施に当たり，ご協力いただいた当社 泉田主任に深謝したい．さらに，本研究をまとめるに当たり，ご指導いただいた大阪大学基礎工学部情報工学科 鳥居宏次教授，同 菊野亨教授ならびに，熱心なご討論をいただいた大阪大学基礎工学部情報工学科 松本健一氏，同大学院博士課程 楠本真二氏に深く謝意を表したい．

参考文献 [1] M. A. Jackson, "Principles of Program Design", Academic Press (1975)；鳥居宏次（訳）："構造的プログラム設計の原理"，日本コンピュータ協会（1980）．
[2] 楠本真二，高田義広，松本健一，菊野亨，鳥居宏次，"モデルに基づく大学，および企業におけるプログラム開発プロセスの評価"，ソフトウェア・シンポジウム'89, 1989, pp. 295～302.
[3] 松本健一，菊野亨，鳥居宏次，"S字型ソフトウェア信頼度成長モデルの大学環境における実験評価－推定精度の比較と習熟係数の決定－"，電子情報通信学会論文誌 D, J 73-D-i, 1990, pp. 175～182.
[4] 宮本勲，"ソフトウェアエンジニアリング：現状と展望"，TBS出版会 1982.
[5] 日本ユニシス教育部（編），"NUPS法による業務システム設計演習テキスト"，日本ユニシス(株) 1989.
[6] G. M. Weinberg and D. P. Freedman, "Reviews, walk-throughs, and inspections", IEEE Trans. Software Eng. Vol. SE-10 (1), 1984, pp. 68～72.
[7] J. Mckissick et al., "Software design inspection for preliminaly design", Proc. COMPSAC 84, pp. 518～519.

執筆者紹介　毛 利 幸 雄 (Yukio Mohri)

昭和26年生．49年青山学院大学理工学部物理学科卒業，同年日本ユニシス（株）入社，社内外に対する情報処理技術分野の教育に従事，現在人材開発本部 専門教育部システム教育課に所属．ソフトウェア技術者協会会員．

状態遷移に着目したプログラミング手法

A State Transition-oriented Programming Method

竹 内 征 勝

要 約 マイクロ・メインフレームの中心となるホストとのセッション接続を行うプログラムを開発してきた．そこではいかなる状況の下でも，ホストとセッション接続させるという目的を達成させるために，トレース法，アクション・マトリックス法等の方法を利用してきた．そして，最後に状態遷移図に基づくプログラミング手法に到達した．

ある状態で，ある刺激を与えた時に，別の状態に移行するというロジックを図式化した状態遷移図を，そのままプログラムで扱えるようにした方法である．

ホストとのセッション接続プログラムにこの手法を採用することによって，セッション接続の失敗の確率を大幅に下げることができた．

本手法はホスト・セッション接続以外にも，プログラムの実行コントロール，メニュー画面のコントロールに適用できる．また，プログラム・ロジックの一部を外部パラメタ化できるので，プログラム開発の生産性向上にも寄与できることを示した．

Abstract The author and his fellow workers have jointly developed a software package which serves for the connection of terminal equipment with a host computer, the core of a micro-mainframe link system. In order to attain the goal of making possible terminals' successful session connection with a host system in any situation, several methods including the message tracing method and the action matrix method have been employed, finally leading to the adoption of a programming method on the basis of state transition graphic formulas.

This method is so designed as to allow the program to manipulate transition diagrams as they are by formulating the logic that a certain state changes into another when a stimulus is given. The adoption of this method for the host-terminal connection program has contributed to a great reduction in the frequency of session connection failures.

In addition, it has been revealed that the method is also applicable for program execution control and a menu control system. Also, it helps to improve program development productivity because part of the program logic can be formed into a parameter set outside of the program.

1. は じ め に

システムの状態を示す方法として，状態遷移図を利用している例は数多くある．

本稿では，状態遷移図をパラメタの形で定義し，そのパラメタをプログラムで直接扱う方法について述べている．

状態遷移図をプログラムで扱おうとしたのは次の理由からである．

当社のオリジナル商品である APMENU(商品名は，U-AP MENU または PW-AP MENU)の中で端末とホストとのセッション接続を自動的に実現することが必要となり，APHSS(APMENU Host Session Software)と呼んでいるプログラムを開発した．APHSS プログラムでは，ホスト・セッション接続の確率を高めるために，トレー

ス法，アクション・マトリックス法等の方法を採用してきたが，最終的にたどり着いたのがこの状態遷移図によるプログラム手法である．

APMENUの中では，ホスト・セッション接続をはじめとして，プログラムの実行コントロール，メニューの表示・選択等で，この状態遷移図によるプログラム手法を採用している．

2. 状態遷移図に基づくプログラム手法

2.1 状態遷移図

状態遷移とは，あるシステムに複数の状態Q_lがあり，刺激X_kによって別の状態Q_jに移ることをいう．関数の形式で記述すると次のようになる．

$$Q_j = F(Q_l,\ X_k)$$

状態遷移図は，状態遷移のすべての場合を図の形式に表現したものである．表現の仕方には，マトリックス形式で表すものと，ここで論じようとしているネットワーク・グラフで表現するものがある．

図1では，Q_1からスタートし，順次刺激X_kを与えて，ゴールQ_gに向かうようになっている．

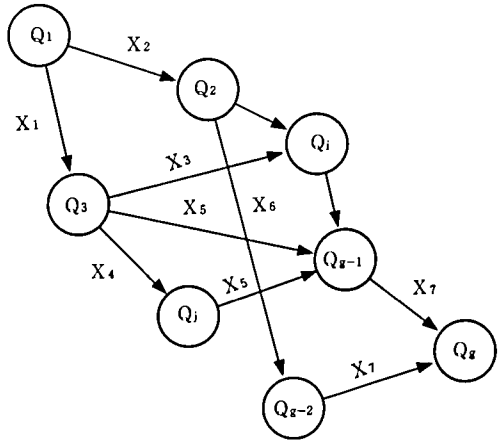


図 1　状態遷移図
Fig. 1　A state transition graph

たとえば，各プログラムの実行と実行ステータスの入手の連続を，状態遷移の進行として捉え，一つのプログラム実行を一つの状態と考える．各状態の機能としては，OSへのコマンド・ステートメントの送出，実行プログラムからのステータスの受信，受信ステータスによる次の状態への移行を含むようにする．図2でその具体例を示す．

2.2 状態遷移図の構成要素とコントロール

1)　構成要素……プログラム・コントロールの例でいうと次のようになる．

①　状態(ノード番号)

その状態を，他の状態と区別するための番号であり，次の状態への飛び先として使用される．

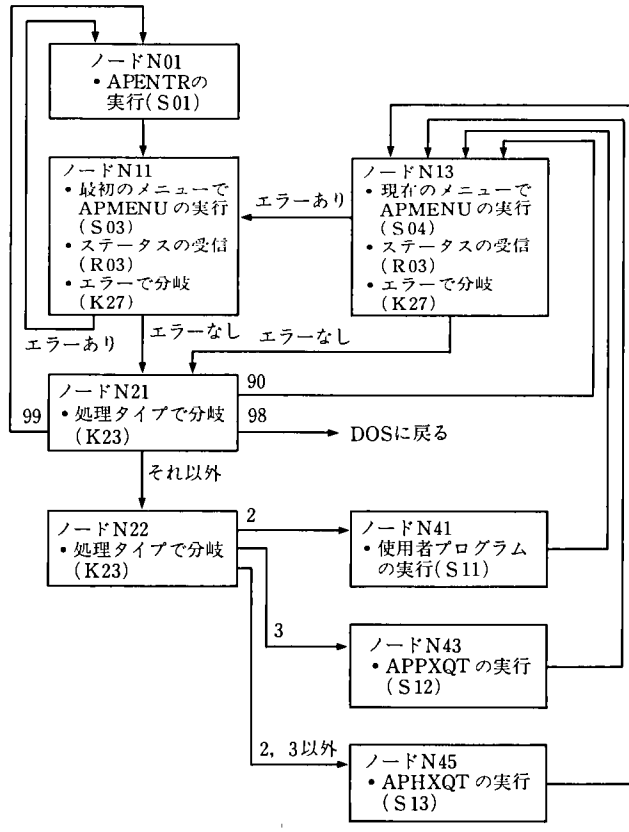


図 2 プログラム実行の状態遷移図例

Fig. 2 The state transition graph of program execution

② ゴールに近づくためのアクション

この場合は，プログラムを実行することがアクションになる．また，アクションの種類を送信電文の番号で指定している．

③ レスポンスの入手

この場合は，起動プログラムからの電文を受信することである．

④ レスポンスの種類による次の状態への移行

③のレスポンスを複数の定数と比較し，一致したら該当の状態番号に移行する．

これらの各構成要素は，その状態遷移図で扱うシステムの規模に応じて，定義数の大きさや複雑さが変わってくる．

2) 状態遷移図のコントロール……パラメタ化された構成要素を解釈し，実行するインタプリタによって，実際の状態遷移図がコントロールされる．インタプリタの実行手順は，次の四つの繰り返しとなる．たとえば，状態遷移ゲームの代表ともいえる双六を考えると次のようになる．スタートからゴールに向かって，馬(ピース)を置く場所が状態番号で示される各状態になる．

① 状態番号の入手

自分のピースが置かれている場所を確認する．

② その状態で指定されたアクションの実行

その場所が「1回休み」等でない場合はサイコロを振る.

③ レスポンスの入手

出たサイコロの目を見る.

④ 次の状態への移行，または完了

サイコロの目の数に従って，指定された状態番号の所に自分のピースを移動させる．移動先がゴールならば，その人は「上がり」となる.

2.3 状態遷移図法の形態

状態遷移図の各状態は，すべての状態が同程度の機能を含んだ等質なものである．しかし,状態遷移図を適用するプログラムの作り方によって,次のように分類できる.

1) 1ノード複数プロセス型……状態遷移図の本来の形式であり，各々のノードに，プロセスの実行，レスポンスの入手，レスポンスの内容ごとによる次のノードの指定の三つの要素を含んでいる.

2) 1ノード1プロセス型……各ノードの機能を単純化し，小回りが効くようにしたものである．プロセスの実行ノード，レスポンスの入手ノード等を別のノードとして定義しているものである．余り細分化すると，状態遷移と言うよりもコマンドになってしまうので注意が必要である.

3) メニュー・ノード型……画面に表示されるメニューの一つ一つを一つのノードとして捉えたものである．メニューには二つのタイプがある．一つは処理メニューと呼ばれるものであり，メニューを選択すると必ず処理が実行されるようになっている．たとえば，本の目次のようなものである．いくら目次の頁数が増えても，ある表題を選ぶと本文の頁が引けるようになっている．他方はメニューのメニューとも呼ぶべきものであり，大分類のメニュー，中分類のメニュー，小分類のメニュー等，メニュー自体がある構造を持っている．処理メニューでは，項目を選択するとプロセスが実行される．メニューのメニューでは項目を選択すると次のメニューが表示される．前者と後者を別のメニューにしている例もあるが，APMENU では同一メニューに前者と後者の機能を含めたものにしている.

2.4 定義の方法

各ノードに関連するプロセスを定義する場合に,そのノードに従属するものとして，内容を定義する方法(ここでは「従属型」と呼ぶ)と，同じ内容を他のノードからも参照することが可能なようにした方法(ここでは「独立型」と呼ぶ)とがある．3.1節で紹介する SMSG，RMSG と KEYW の各パラメタは「独立型」で定義している．各々の内容は独立して設定・変更できる．後述する 3.3 節のメニュー・ノードでは，メニュー自体の定義は「従属型」で定義しており，プロセス定義は「独立型」の定義になっている．「従属型」定義は一般的にノード定義の近傍で定義されるので，参照が容易なため理解しやすい．しかし，複数の場所で同一の内容を定義する場合には，同じ内容を重複して定義することを避けるためにも，「独立型」で定義することが望ましい.

3. APMENU での実装例

APMENU は多数のプロセスと制御用ファイルから構成されており，次の五つのモ

ジュールに機能分割されている.

- プログラム・コントロール・モジュール(apms. exe, appxqt. exe, aphxqt. exe 等)
- メニュー・コントロール・モジュール(apmenu. exe 等)
- ホスト接続モジュール(apphss. exe, aphss. exe 等)
- セキュリティ・モジュール(apentr. exe 等)
- 共通モジュール(apu03. exe 等)

そして APMENU では, モジュールの各場所で複数の状態遷移図を使用している.

図 3 で, APMENU の動きの概略を説明すると次のようになる. まず, 常駐プログラムの apms が実行される. apms によって apentr プログラムが起動され, 使用者名の入力待ちになる. 使用者名が入力されると, その使用者用のメニューが apmenu プログラムによって表示される.

メニュー項目の選択により, 次のメニューが表示されるか, その選択項目に定義されたプロセスが実行される. プロセスの種類によって, 単独のユーザ・プログラムが実行されたり, appxqt プログラム経由でユーザ・プログラムが起動され, あるいは, ホスト・アプリケーション起動のために aphxqt プログラムが実行される.

appxqt プログラム経由でユーザ・プログラムが起動された場合には, apu 03 プログラムがユーザ・プログラムのステータスをチェックし, 表示画面を保留したりする.

aphxqt プログラムでは, apphss プログラムがホストのセッション接続に必要な情

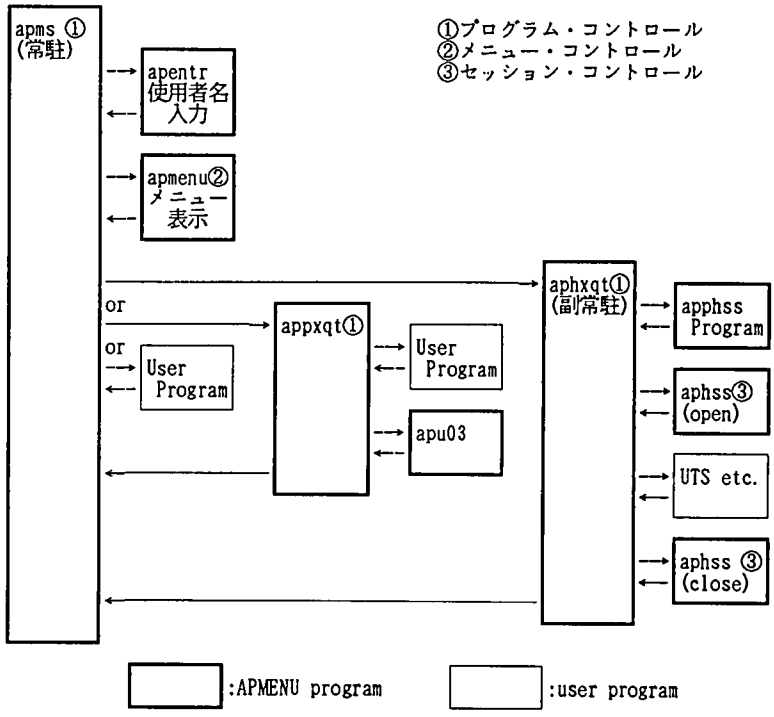


図 3 APMENU での状態遷移図の利用

Fig. 3 Various state transition graph methods are used in the APMENU system

報を各種ファイルから集めてキーワードに変換する．そのキーワードを使用して，aphssプログラムがホストへのセッション接続を行い，ホスト接続完了後にUTSエミュレータを起動する．そこで，人とホスト・アプリケーションが会話をする．会話の終了後は，aphssプログラムによってセッション切断が行われ，再度メニュー表示に戻る．

図3の①，②と③のところで状態遷移図に基づくコントロールが行われている．アクションを引き起こす刺激の種類によって分けると，次のようになる．

① 起動プログラムのエラーまたは指示等のステータスによって，次の行動を決定する．

② 利用者によるメニュー選択によって次の行動を決定する．

③ ホスト・システムからの受信電文によって次の行動を決定する．

3.1 1ノード多プロセス型(プログラム・コントロール)

APMENUでは，常駐プログラムが複数のプログラムをコントロールしており，そのコントロールにこの形式を使用している．

プログラム・コントロールの流れを状態遷移図の形で表現する理由は，人間や他システムとのインタフェースをもたない閉じたシステム内で，エラーが発生した場合等にも適切に作動するようにプログラムをコントロールできるロジックを実現させるためである．先の図2の状態遷移図をリスト1のパラメタで実現している．

リスト1での例をあげながら説明する．

[リスト1]　プログラム・コントロールのノード定義
list 1 : The Node Definition of Program Control

```
        |- ノード番号
        |   |- 比較が一致しない場合の，次のノード番号
        |   |   00 はAPMSから抜け出て，ＤＯＳに戻る
        |   |   |- 実行コマンドの番号
        |   |   |   |- 受け取るステータスの形式番号
        |   |   |   |   |- 比較するキーワードの番号
        |   |   |   |   |   |- 比較の数
NODE Nxx,Nxx,S01,R01,K00,00          ,
NODE OP,xxx,Nxx
     |  |   |- 電文が一致したときの飛び先のノード番号
     |  |- 比較する電文の番号
     |- オペレータ
        EQ: キーワード値 = 定数
        NE: キーワード値 = 定数

NODE N01,N11,S01,R02,K00,0  , execute APENTR and recieve status

NODE N11,N21,S03,R03,K27,1  , execute APMENU with orig. menu
NODE NE,000,N01             ,    status is not 0 then goto N01

NODE N13,N21,S04,R03,K27,1  , execute APMENU with curr. menu
NODE NE,000,N11             ,    error status is not 0 then goto N11

NODE N21,N22,S00,R00,K23,3  , compare with ptype(K23)
NODE EQ,090,N13             ,    ptype 90 then goto N13 for APMENU
NODE EQ,098,N00             ,    ptype 98 then goto N00 for DOS
NODE EQ,099,N01             ,    ptype 99 then goto N01

NODE N22,N45,S00,R00,K23,2  , compare with ptype
NODE EQ,002,N41             ,    ptype  2 then goto N41
NODE EQ,003,N43             ,    ptype  3 then goto N43

NODE N41,N13,S11,R00,K00,0  , execute user program

NODE N43,N13,S12,R00,K00,0  , execute APPXQT program

NODE N45,N13,S13,R04,K00,0  , execute APHXQT program
```

1) プログラム・コントロール処理の内容……プログラム・コントロールの大きな流れは次のようになっている．まず，apentr プログラムを実行し，いくつかの情報をキーワードの形でステータスを受け取る．そのキーワードに基づいて，apmenu プログラムを実行する．同じようにいくつかの情報をキーワードの形でステータスを受け取る．そのキーワードに基づいて，使用者プログラムを起動する．

2) APMENU のプログラム・コントロールでは，実用面から次の考慮を入れている．

① パラメタのソース・コードを直接実行するのでなく，一度中間ファイルに変換してから実行するようにしている．

② 各ノードのメモリ上の表現を固定長にするため，中継のノードを加えている．中継ノードを使用することによって，キーワード比較の数はいくつでも良くなっている(N 21，N 22 等)．

③ 共通変数としてキーワードを使用している．キーワードを使うことによって，あるプログラムから受け取った文字列を他のプログラムのコマンド・ステートメントに使用することができる．

3) ノード定義……APMENU の各プログラムの実行コントロールはリスト 1 のノード定義で行っている(リスト 1 を参照)．ノードの処理はノード N 01 から始まる．ノード N 01 で apentr プログラム実行のコマンド・ステートメント(S 01)を OS に対して送り，ステータスを R 02 の形式で受け取る．そしてノード N 11 に移る．

ノード N 11 では，メニューを表示するために apmenu プログラム(S 03)のコマンド・ステートメントを実行し，ステータスを R 03 の形式で受け取る．そして，ステータスの内容のうちエラー・ステータス(K 27)をチェックして，0 ならノード N 21 に移る．0 以外の時はノード N 01 に移り，再度 apentr プログラムを実行する．

ノード N 21 では，リターン・ステータスのうちの処理タイプ(K 23)の値によって飛び先のノード番号を決めている．

- K 23 が 90 なら，ノード N 13 に飛び，メニュー表示の続きを表示する．
- K 23 が 98 なら，OS に戻る．
- K 23 が 99 なら，ノード N 01 に飛び，再度 apentr プログラムを実行する．

ノード N 22 では，処理タイプのチェックを続ける．

- K 23 が 02 なら，ノード N 41 に飛ぶ．
- K 23 が 03 なら，ノード N 43 に飛び，appxqt プログラム(S 12)を実行する．
- K 23 がそれ以上なら，ノード N 45 に飛び，aphxqt プログラム(S 13)を実行する．

ノード N 41 では，使用者プログラム(S 11)を実行する．使用者プログラム終了後はノード N 13 に飛ぶ．

OS への送信電文は Sxx で指定し，受信電文の形式は Rxx で指定している．Sxx と Rxx の内容は，SMSG と RMSG 定義で設定している．

4) 起動プログラムの定義……送信電文は，SMSG パラメタにより，固定文字列と

キーワードで電文内容を定義している(リスト2を参照)．たとえば，S01はapentrプログラムの実行ステートメントの内容を定義しており，APENTRという固定部分とキーワード(K21：使用者名，K22：パスワード等)から構成されている．このパラメタを使って起動するプログラムはAPMENUのプログラムだけであり，使用者のプログラムはリスト7のプロセス定義を利用して起動する．

5) プログラム・ステータスの定義……起動プログラムから返されるステータスを定義する．これは，キーワードのみから構成されている(リスト2を参照)．

［リスト2］ 起動プログラムとプログラム・ステータスの定義
List 2 : The Command and Return Status Definition of a Program Execution

```
        |- 実行コマンドの番号
        |  |- キーワードの数
        |  |  |- 第1のキーワード番号
        |  |  |  |- 第2のキーワード番号
        |  |  |  |  |- ・・・・・・・・・
                        -- テキスト
SMSG S01,08,K12,13,14,15,21,22,18,19
SMSG APENTR 01,$12$,$13$,$14$,$15$,$21$,$22$,$18$,$19$

SMSG S03,11,K12,13,14,15,40,41,31,32,33,34,35
SMSG APMENU 01,$12$,$13$,$14$,$15$,$40$,$41$,$31$,$32$,$33$,$34$,$35$

SMSG S04,11,K12,13,14,15,29,30,31,32,33,34,35
SMSG APMENU 01,$12$,$13$,$14$,$15$,$29$,$30$,$31$,$32$,$33$,$34$,$35$

SMSG S11,01,K01
SMSG S01$

SMSG S12,07,K12,13,14,15,36,37,17
SMSG APPXQT 01,$12$,$13$,$14$,$15$,$36$,$37$,$17$

SMSG S13,07,K12,13,14,15,36,37,17
SMSG APHXQT 01,$12$,$13$,$14$,$15$,$36$,$37$,$17$


        |-受け取るステータスの形式番号
        |  |-キーワードの数
        |  |  |-第1のキーワード番号
        |  |  |  |-第2のキーワード番号
        |  |  |  |  |-・・・・・・・・・
RMSG R02,15,K27,12,13,14,15,26,40,41,31,32,33,34,35,21,22,
RMSG R03,18,K27,12,13,14,15,26,29,30,31,32,33,34,35,23,36,37,39,01,
RMSG R04,06,K27,12,13,14,15,26,
```

6) キーワードの定義……ノード定義，SMSG定義，RMSG定義の中で参照される共通変数をKEYWパラメタで定義している．これらのキーワードは，ノード・パラメタの実行に先立って，外部から内容が設定される．リスト3のテキストは初期設定のキーワードである．

キーワードはAPMENUの各プログラム間で共通なパラメタであり，文字列と数値を定義できる．文字列のうち，キー番号1から10までは最大80桁，キー番号11以上は最大12桁である．80桁のキーワードは，ホストへの送信電文，受信電文，端末プログラムへの実行ステートメント，レスポンス，プログラムの存在するディレクトリ名，ウィンドゥのタイトル等に利用している．

3.2 1ノード1プロセス型(ホスト・セッション接続)

1ノード複数プロセス型の各ノードがもっているプロセスの実行，レスポンスの入手とレスポンスの比較を，各々別ノードで処理するようにしている．

ホスト・セッション接続で状態遷移図を使用する理由は，ホスト・セッション接続のためにホストに送る電文と，その時のホストの状態により返される受信電文と，次に送るべき電文との関係が複雑なため，それらの関係のみについて明確にするためで

[リスト3] キーワードの定義
List 3 : The Keyword Definition

```
        |-キーワード番号
        |  |- キーワードの文字のタイプ, A=ASCII,B=数値
        |  |  |-文字数
        |  |  |  |- テキスト
        |  |  |  |                    |- コメント
KEYW K01,A,80,APENTR etc              ,AP program execution command tail
                                      ,or host response text
KEYW K02,A,80,ERROR TEXT              ,AP progarm response
KEYW K03,A,80,¥APM¥APM                ,AP program directory
KEYW K04,A,80,APED                    ,ws program execution statement
KEYW K05,A,80,prog title              ,program title
KEYW K06,A,80,AOA     etc             ,host program execution statement
KEYW K07,A,80,$$OPEN etc              ,APHSS send text to host

KEYW K11,B,02,01                      ,function
KEYW K12,B,02,01                      ,category number(cat)
KEYW K13,B,02,01                      ,AP number(apno) for concurrent operation
KEYW K14,B,02,01                      ,user number(uno)
KEYW K15,L,04,1111                    ,option word(opt)
KEYW K16,A,12,ABCDEGHHIJKL            ,1st application parameter(K16)
KEYW K17,A,12,123456789012            ,2nd application parameter(K17)
KEYW K21,A,12,useruseruser            ,user ID(uid)
KEYW K22,A,06,passwd                  ,user password(upass)
KEYW K23,B,02,03                      ,process type(ptype)
KEYW K24,B,02,01                      ,host application type(aptype)
KEYW K25,B,02,01                      ,host number(hno)
KEYW K26,B,02,90                      ,return type(rttype)
KEYW K27,B,02,00                      ,return status(rstat)
KEYW K29,B,03,00                      ,menu 1st fno
KEYW K30,B,03,00                      ,menu 1st rno
KEYW K31,B,02,00                      ,menu selection no. of 1(s1)
KEYW K32,B,02,00                      ,menu selection no. of 2(s2)
KEYW K33,B,02,00                      ,menu selection no. of 3(s3)
KEYW K34,B,02,00                      ,menu selection no. of 4(s4)
KEYW K35,B,02,00                      ,menu selection no. of 5(s5)
KEYW K36,B,03,01                      ,prod fno
KEYW K37,B,03,01                      ,prod rno
KEYW K39,A,08,001                     ,communication line name(lln)
KEYW K40,B,02,00                      ,menu original fno
KEYW K41,B,02,00                      ,menu original rno
KEYW K65,A,06,DEM60                   ,host application ID
```

ある．このホスト・セッション接続に使用している方法は，相互関係の弱いシステムとシステムとのやりとりに利用できる．

ホスト・セッション接続での例(リスト4参照)をあげながら説明する．

1) セッション接続・切断の内容……行っていることは，ホスト・セッション接続とセッション切断である．セッション接続では次の電文を順次送って，ホストのアプリケーション・プログラムを起動している．

$$SON terminalID
$$OPEN applicationID
userID/password clearance-level
@RUN runID, accountID, project-code
@ADD statement

人とホストのアプリケーション・プログラムとの会話が終わった後は，次の電文を送ってセッションを切断している．

@FIN
$$CLOSE
$$SOFF

2) APMENUのホスト・セッション接続では，さらに次の考慮を入れている……比較電文に，ノーレスポンス(R 003)，SOE(R 001)とその他(R 002)の項目を追加している．

3) ノード定義……セッション・オープンのコントロールはノード11(N 11)から始

まる．ノード 11 はなにもしないでノード 23 に移る．ノード 23 で$$OPEN 電文(S 36)を送り，ノード 24 でホストからの電文を受信する．ノード 25，ノード 26 ではホストから返された電文と定型電文(R 015，R 016，R 001，R 003)とを比較し，一致した定型電文によって，次のノードを決めている．送信電文は Sxx で指定し，比較する受信電文は Rxxx で指定している．Sxx と Rxxx の内容は，次の SMSG と RMSG 定義で設定している．

［リスト４］　ホスト・セッション接続のノード定義
list 4 : The Node Definition of Host Connection

```
        |-ノード番号
        |  |-オペレータ
        |  |   U:無条件ジャンプ    (Nxxx)
        |  |   S:電文送信          (Sxxx)
        |  |   G:電文受信          (Gxxx)
        |  |   C:キーワード比較    (Kxxx)
        |  |        K001 : ホスト受信電文
        |  |        K045 : aptype
        |  |  |-オペレータ別のキーワードなどの番号
        |  |  |  |- 比較の数 (最大 2)
        |  |  |  |  |- コメント
        |  |  |  |  |
NODE N025,C,K01,2     , check response of $$OPEN
NODE E,R016,N041      ,    session path open to xxxxx
NODE E,R015,N035      ,    session path open (not match APNAME)
      |  |  |- 電文が一致したときの飛び先のノード番号
      |  |- 比較する電文の番号
      |- オペレータ


NODE N011,U,N023,0    ,*the start node of all session open  process

NODE N023,S,S36,0     ,*send $$OPEN

NODE N024,G,G001,0    ,*get a output of $$OPEN

NODE N025,C,K01,2     ,*check response of $$OPEN
NODE E,R016,N041      ,    session path open to xxxxx
NODE E,R015,N035      ,    session path open (not match APNAME)

NODE N026,C,K01,2     , check response
NODE E,R001,N041      ,    SOE
NODE E,R003,N066      ,    no-response

NODE N027,U,N024,0    ,    other

NODE N041,C,K24,2     ,*completion of $$OPEN and check aptype
NODE E,V001,N050      ,    MAPPER
NODE E,V002,N101      ,    demand
```

4)　送信電文の定義……送信電文は，SMSG パラメタにより，固定文字列とキーワードから電文内容を定義している(リスト 5 を参照)．たとえば，S 36 は$$OPEN 電文の内容を定義しており，$$OPEN という固定部分とキーワード(K 65：ホスト・アプリケーション ID)から構成されている．

5)　受信電文の定義……受信電文についても，RMSG パラメタにより，比較すべき固定文字列とキーワードで内容を定義している(リスト 5 を参照)．比較する文字列のうち実際に使うのは，指定された文字数だけである．たとえば，R 015 では，SESSION PATH OPEN TO…のうち，初めの 17 文字だけを比較の対象にしている．

6)　キーワードの定義……リスト 4 の例では受信電文(K 01)を使っており，リスト 5 の SMSG 文と RMSG 文ではホスト・アプリケーション ID(K 65)を使用している．

7)　ホスト・セッション・ノードを実行した時の送受信記録……APHSS プログラムで実際にホスト接続を行った時の送受信電文の例を図 4 に示す．

［リスト5］　送信電文と受信電文の定義
list 5 : The Message Definition Sent to and Received from Host

```
      |- 送信電文番号
      |  |- 文字数
      |  |  |- 送信電文の文字タイプ
      |  |  |  |- 送信電文に含まれるキーワードの数
      |  |  |  |  |- キーワード番号
      |  |  |  |  |                |- コメント
SMSG S36,12,A,01,K65              ,for application ID
SMSG $SOPEN $K65$


      |- 受信電文番号
      |  |- 比較電文の開始位置
      |  |  |- 比較する文字数
      |  |  |  |- 受信電文に含まれるキーワードの数
      |  |  |  |  |- 受信電文の文字タイプ, A=ASCII, J=漢字, H=16進
      |  |  |  |  |  |- 受信電文の内容
RMSG R001,01,00,00,A,     SOE
RMSG R002,01,00,00,A,     other message
RMSG R003,01,00,00,A,     no-response
RMSG R015,01,17,00,A,SESSION PATH OPEN
RMSG R016,01,20,01,A,SESSION PATH OPEN TO xxxxxx
RMSG K65
      |- キーワード番号
```

```
H C 90/11/23 09:22:35.01,76,1,8,0,:11n=UTS50-01<
H S 90/11/23 09:22:35.EV  OSS$$OPEN DEM60<
                     CT  01E              <
H R 90/11/23 09:22:37.EeEMEbCSESSION PATH OPEN TO: DEM60   EbCS<
                     C C C R                          C RE<
H R 90/11/23 09:22:39.EV6 SSENTER USERID/PASSWORD:EeEkEV6 S<
                     CT  IE                      C C CT  I<
H R 90/11/23 09:22:40.S<
                     E<
H S 90/11/23 09:22:40.EV  OSSZNQE9999/PASWD <
                     CT  01E              <
H R 90/11/23 09:22:40.EeEkEV6 S<
                     C C CT  I<
H R 90/11/23 09:22:41.EV6 SS*UNIVAC 1100 OPERATING SYSTEM  LEV. 41R3-5E27*V0(RSI)*  EeEkEV6 S<
                     CT  IE                                                      C C CT  I<
H R 90/11/23 09:22:42.EV6 SS*** THIS IS 1100/70#1-BABA NUL ***  EeEkEV6 S<
                     CT  IE                                     C C CT  I<
H R 90/11/23 09:22:42.EV6 SSCurrent session number:  25 EeEkEV6 S<
                     CT  IE                              C C CT  I<
H R 90/11/23 09:22:43.EV6 SSPrevious session was:  90 Dec 04    09:20:36EeEkEV6 S<
                     CT  IE                                            C C CT  I<
H R 90/11/23 09:22:43.S<
                     E<
H S 90/11/23 09:22:43.EV  OSS@RUN APM,ZY9999,APM<
                     CT  01E                    <
H R 90/11/23 09:22:44.EeEkEV6 S<
                     C C CT  I<
H R 90/11/23 09:22:45.EV6 SSDATE: 120490      TIME: 092222  EeEkEV6 S<
                     CT  IE                              C C CT  I<
H R 90/11/23 09:22:46.S<
                     E<
H S 90/11/23 09:22:46.EV  OSS@ADD,P FCS.START                                    <
                     CT  01E                                                     <
H D 90/11/23 09:22:46.0,76,1,8,0,90 L0      <
                                    F0      <
H Z 90/11/23 09:22:46.<

H C 90/11/23 09:22:59.02,76,1,8,0,:11n=UTS50-01<
H S 90/11/23 09:23:00.EV  OSS@FIN<
                     CT  01E    <
H R 90/11/23 09:23:01.EeEkEV6 S<
                     C C CT  I<
H S 90/11/23 09:23:01.EV  OSS$$CLOSE<
                     CT  01E       <
H R 90/11/23 09:23:02.EV6 SSSESSION PATH CLOSED EeEkEV6 SS<
                     CT  IE                      C C CT  IE<
H D 90/11/23 09:23:03.2,76,1,8,0,90 L0                              <
                                    F0                              <
H Z 90/11/23 09:23:03.<
```

(注)　H C はプログラムの開始とコマンド・テイル, H S は送信電文
H R は受信電文, H D はリターン・ステータス, H Z は処理終了

図 4　ホストセッションの送受信記録

Fig. 4　The logging data sent to and received from host

3.3 メニュー・ノード型

状態遷移図をメニューの表示・選択システムに適用したものである．一つのメニュー画面を一つのノードと見なしている．

ここで用いられているプログラミング手法は，使用者からの希望・条件等を受け入れながら目的の処理プログラムを実行させるようなマン・マシン・システムに適用できる．APMENU では，使用者が独自に設定するメニューと APMENU 自身がもっているメニュー・システムをこの方法で実現している．

APMENU の運用・管理のメニューを例にして説明する．

3.3.1 メニューの表示と状態遷移図

APMENU で最初に表示されるのは，図 5 の基本選択メニューである．図 5 で 80 を選択すると図 6 の運用・管理メニュー画面が表示され，図 6 で 40 を選択すると図 7 の自動立上げの設定メニューが表示される．図 7 で 90 を選択すると図 6 の表示に戻る．

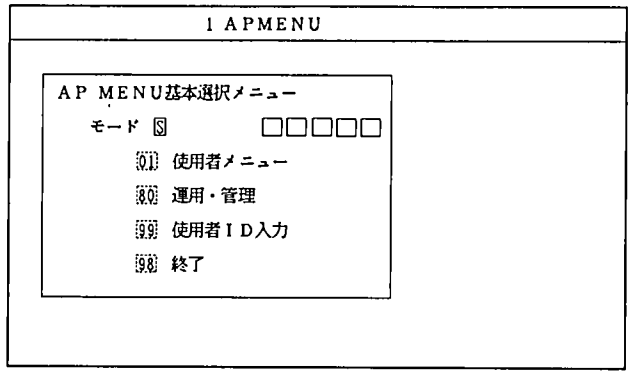


図 5　ノード M991-401 のメニュー画面

Fig. 5　The menu screen of node M 991-401

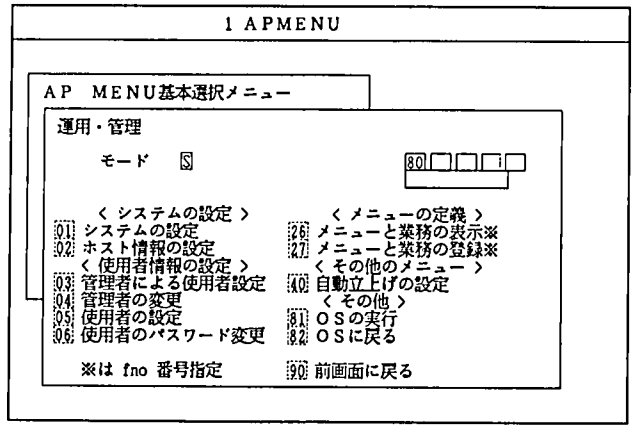


図 6　ノード M991-903 のメニュー画面

Fig. 6　The menu screen of node M 991-903

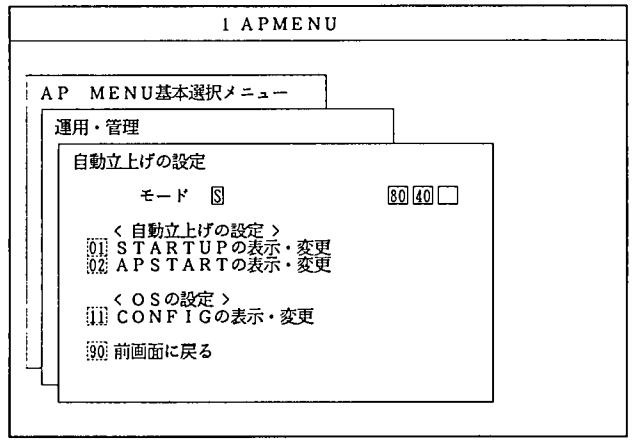


図 7 ノード M991-460 のメニュー画面

Fig. 7 The menu screen of node M 991-460

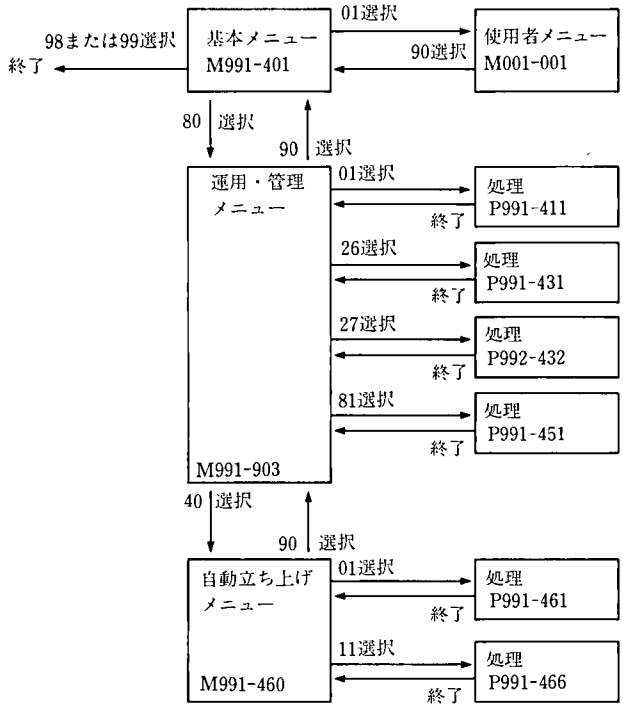


図 8 メニューの状態遷移図

Fig. 8 The state transition graph of the menu system

図 7 で 11 を選択すると，リスト 7 の P 991-466 のプロセスが実行される．実際にはエディタの aped プログラムが起動され，ルート・ディレクトリの CONFIG ファイルの内容が表示される．メニュー画面での選択は，選択番号を連続して入力(たとえば 8 0 4 0 1 1 Enter)してもよいし，スキャン・キーまたはマウスで で囲まれた部分を指定してもよい．これらを状態遷移図で表したものが図 8 である．

3.3.2 ノードの定義

一つのメニュー・ノードで，メニューの 1 画面分を定義している．図 5 のメニュー

[リスト6]　メニュー・ノードの定義
list 6 : The Menu Node Definition

```
        |-メニューのノード番号（ｆｎｏ）
        |   |-メニューのノード番号（ｒｎｏ）
        |   |   |-この画面に含まれる項目数
        |   |   |   |-選択番号入力欄の数
        |   |   |   |   |-業務パラメタ入力欄１部分の有無
        |   |   |   |   |   |-業務パラメタ入力欄入力２部分の有無
        |   |   |   |   |   |   |-選択１項目部分のカラム位置
        |   |   |   |   |   |   |   |-選択１項目部分の幅
        |   |   |   |   |   |   |   |   |
NODE Mxxx,xxx,02,Sn,x,n,lxx,ll

NODE ＡＰＭＥＮＵ基本選択メニュー          -------- 画面タイトル

      |-表示位置番号
      |   |-表示番号
      |   |   |-次のメニューまたは処理のノード番号（ｆｎｏ）
      |   |   |   | -次のメニューまたは処理のノード番号（ｒｎｏ）
      |   |   |   |   |
NODE 02,002,M101,202
NODE 部内ＯＡ業務                          -------- 項目タイトル

    ＜ＡＰ　ＭＥＮＵ基本選択メニュー＞
NODE M991,401 04,S5,0,0,111,20
NODE ＡＰ　ＭＥＮＵ基本選択メニュー
NODE 01,01 M001,001
NODE 使用者メニュー
NODE 03,80 M991,900
NODE 運用・管理
NODE 05,99 P000,099
NODE 使用者ＩＤ入力
NODE 07,98 P000,098
NODE 終了

    ＜運用・管理＞
NODE M991,900,S,18,S5,0,1,101,26
NODE 運用・管理
NODE 02,100,P00,000
NODE ＜ システムの設定 ＞
NODE 03,01,P991,411
NODE システムの設定
NODE 04,02,P991,412
NODE ホスト情報の設定
NODE 05,100,P00,000
NODE ＜ 使用者情報の設定 ＞
NODE 06,03,P991,413
NODE 管理者による使用者設定
NODE 07,04,P991,414
NODE 管理者の変更
NODE 08,05,P991,415
NODE 使用者の設定
NODE 09,06,P991,416
NODE 使用者パスワードの変更
NODE 12,100,P00,000
NODE ※は fno 番号指定
NODE 17,100,P00,000
NODE ＜ メニューの定義 ＞
NODE 18,26,P991,431
NODE メニューと業務の表示※
NODE 19,27,P991,432
NODE メニューと業務の登録※
NODE 20,100,P00,000
NODE ＜その他のメニュー＞
NODE 21,40,M991,460
NODE 自動立上げの設定
NODE 22,100,P00,000
NODE ＜ その他 ＞
NODE 23,81,P991,451
NODE ＯＳの実行
NODE 24,82,P991,452
NODE ＯＳに戻る
NODE 27,90,P000,090
NODE 前画面に戻る

    ＜自動立上げの設定＞
NODE M991,460,S,06,S3,0,0,101,36
NODE 自動立上げの設定
NODE 01,100,P00,000
NODE ＜ 自動立上げの設定 ＞
NODE 02,01,P991,461
NODE ＳＴＡＲＴＵＰの表示・変更
NODE 03,02,P991,462
NODE ＡＰＳＴＡＲＴの表示・変更
NODE 05,100,P00,000
NODE ＜ ＯＳの設定 ＞
NODE 06,11,M991,466
NODE ＣＯＮＦＩＧの表示・変更
NODE 08,90,P000,090
NODE 前画面に戻る
```

画面をリスト6のM 991-401のノード・パラメタで決定している．図6はM 991-900，図7はM 991-460で定義している．

先の二つのノード定義とは異なり，ノード定義の中にメニュー表示のためのパラメタを含めている．リスト6のM 991-401 NODEパラメタのところで□で囲んだパラメタが状態遷移図特有のものであり，それ以外はメニュー画面を表示するためのパラメタとなっている．これは先の定義の形式でいう「従属型」になっている．

例として，メニュー・ノード(M 991-401)について説明する．

ノード991-401では，ノード番号として991-401を持っている．実際には，991はファイルのIDで，401はレコードのIDである．ファイルのIDはファイル名の一部を構成しており，レコードのIDはレコードの検索キーになっている．

まず，□以外のパラメタを使ってメニューを表示する．使用者が指定した選択番号の内容によって，次の処理内容を決めている．選択番号が01ならばノード001-001のメニューを表示し，選択番号が80ならば991-900のメニューを表示する．選択番号が99ならば，ファイルIDが000なので処理タイプ99を実行して，使用者名入力画面に戻る．また，選択番号が98ならば同様にして処理タイプ98を実行してOSに戻る．

3.3.3 プロセスの定義

ここではメニュー項目の選択によって実行される処理について定義している．例として，プロセス991-466について説明する．このプロセスは「独立型」で定義している．参照されるためのIDとして，番号991-466を持っている．ノード定義と同じよう

[リスト7]　処理の定義
list 7 : The Process Definition

```
                |-処理番号（ｆｎｏ）
                |  |-処理番号（ｒｎｏ）
                |  |  |-パラメータ形式
                |  |  |  |-処理タイプ(ptype)
                |  |  |  |  |-電文中のキーの数
                |  |  |  |  |  |-第１キー番号
                |  |  |  |  |  |  |-第２キー番号
                |  |  |  |  |  |  |  |-第３キー番号
a:  PROC P000,000,F,P01,02,K15,K16,Kxx
b:  PROC program title                    ------ プログラム・タイトル
c:  PROC WS command statement             ------ プログラム起動コマンド


   ＜処理番号Ｐ９９１-４１１の内容＞
PROC P991,411,F,P02,03,K12,K21,K22
PROC システムの設定
PROC appset 1,$K12$,0,1,0,$K21$,$K22$,0,0

   ＜処理番号Ｐ９９１-４３１の内容＞
PROC P991,431,F,P02,03,K03,K12,K17
PROC メニューと業務の表示･変更
PROC aped -N $K03$APM$K12$$K17$.SAP

   ＜処理番号Ｐ９９１-４３２の内容＞
PROC P991,432,F,P02,02,K12,K17
PROC メニューと業務の登録
PROC apmdset APM$K12$$K17$.SAP

   ＜処理番号Ｐ９９１-４５１の内容＞
PROC P991,451,F,P02,00
PROC ＯＳの実行
PROC CMD

   ＜処理番号Ｐ９９１-４６１の内容＞
PROC P991,461,F,P02,00
PROC ＳＴＡＲＴＵＰの表示･変更
PROC aped -N ¥STARTUP.CMD

   ＜処理番号Ｐ９９１-４６６の内容＞
PROC P991,466,F,P02,00,
PROC ＣＯＮＦＩＧの表示･変更
PROC aped -N ¥CONFIG.SYS
```

に，991 はファイルの ID で，466 はレコードの ID である．リスト 7 の b パラメタのタイトルは，OS/2 のウインドゥ・モードで実行された時のウインドゥのタイトルになる．

実行プログラムは c パラメタでコマンド・テイルも含めて指定する．ホスト接続については，これ以外にホスト上で実行するプログラムのコマンド・ステートメントの指定パラメタが追加されている．

4. 今までの成果と今後の展開

1) プログラム・コントロールへの利用……今までは，APMENU の中だけで行っていたが，多数のプログラムを含んだシステムへの適用の検討を行っている．今後使用される分野は増えるが，機能の発展はすくない．

2) セッション・コントロールへの利用……この状態遷移図法を使用した，APHSS プログラムのホスト・セッション接続では，複数のハードウェア，OS，通信エミュレータと対象ホストへの適用を行ってきている．

それらを挙げると，次のようになる．

ハードウェア	OS	通信エミュレータ	通信プロトコル	対象ホスト
DS 7	CDOS	UTS	UTS	2200
DS 7	CDOS	UTS	パケット	2200
DS 7	CDOS	UTS	ハイレベル	2200
DS 7	CDOS	UTS	DS 7 WS	2200
DS 7	CDOS	UTS	M 345	2200
PW²	MS-DOS	UTS	UTS	2200
PW²	AX	UTS	UTS	2200
PW²	OS/2	UTS	UTS	2200
PW²	OS/2	ISINT	INT 1	2200
B 10	BTOS	UTS	UTS	2200
J-3100	MS-DOS	UTS	UTS	2200
if 386 AX	MS-DOS	3270	3270	2200
if 386 AX	MS-DOS	3270	3270	IBM

CDOS はデジタルリサーチ社，MS-DOS と OS/2 はマイクロソフト社，if 386 AX は沖電気(株)，J-3100は東芝(株)の登録商標である．
UNIX は UNIX System Laboratories, Inc. が開発しライセンスしている．

この分野では，今後対象ハードウェア，OS，通信エミュレータとホストの種類を増やすことになる．

3) メニュー・コントロールでの利用……メニュー・ノードのパラメタ内容を直接画面に表示するインタプリタを開発してきた．ウインドゥ・システムを持たない OS では，このインタプリタを独自に開発してきたが，OS/2 ではプレゼンテーション・マネージャを使ったインタプリタになっている．このインタプリタ部分を他のウインドゥ・システムへ移植することによって，同じメニュー・ノード・パラメタが使用できるメニュー・システムが実現できる．

プレゼンテーション・マネージャからMS-DOSのWindows*(3.0)への移植は計画中である．また，UNIXのX-Window**への移植も検討を行っている．

5. おわりに

このプログラミング手法はプログラム開発からの要請が先にあり，後から概念を整理し，汎用的なプログラミング手法としたものである．

初期の頃はホストのセッション接続の確率を高めるための努力が中心であった．次に求められたのは，ホスト・システムやネットワーク・システムのレベルアップと適用対象の拡大に対するプログラム適応能力の向上であった．それらを指向するうち，状態遷移図の観点からプログラムの仕組みを見直すようになり，プログラム・コントロールとメニュー表示コントロールにまで，範囲を拡大してきている．

また，実際に開発してきて言えることだが，プログラム・ロジックの大部分を外部パラメタ化しているので，別の処理プログラムに適用させても，プログラム本体の変更は少ない．そして，開発工数の短縮，保守のしやすさ等で効果を上げている．

メニュー・ノードの開発に際して行ってきたことだが，プロトタイピング的に先にノード・パラメタと処理パラメタを決めてしまい，後からそのインタプリタ部分を開発してきた．今後はメニュー・コントロールのパラメタに機能を追加し，OAシステム開発で大きな工数を占めているMMI(マン・マシン・インタフェース)システムの開発効率向上のためのプログラミング手法に発展させたい．

システム開発において，これらの手法がなんらかの参考になれば幸いである．

---

* WINDOWS：マイクロソフト社の登録商標である．
** X-Window：マサチューセッツ工科大学の登録商標である

執筆者紹介 竹内征勝 (Masakatsu Takeuchi)

昭和44年京都大学理学部宇宙物理学科卒業．同年日本ユニシス(株)入社．大型機の性能評価，評価ツールの開発に従事した後，パソコンを利用したOAシステムの開発に従事．現在オープンシステム推進本部OAソフトウェアー部に所属．日本技術士会，情報処理学会の各会員．

ソフトウェア開発現場におけるプログラミング教育

Programming Education at Front Software Development Sites

勝 田 祐 輔

要 約 本稿は，ソフトウェア開発の現場において実施したプログラミング教育の経験に基づき，その考え方および方法を整理したものである．主な主張は以下の通りである．

プログラミング教育においては，演習問題を中心に，わかりやすい良いスタイルのプログラムとはどういうものか，最初に書いたコードを何回も改良する学習の中で認識させていくということを根幹とする．そのためには，配慮の行き届いた教科書を選び，生徒の特質を見抜き，向上していくプロセスを生徒と共に経験する心構えと熱意が大切である．

開発現場における教育といえども，長期的な視野を持ち，絶えざる工夫・配慮が必要である．生徒が必要な資質を有していることは重要な要素である．そして，それをより伸ばすべく鼓舞すべきである．さらに，良い開発を行うための基本動作というべきものを，生徒に身に付けさせることを，この段階から意図すべきである．

Abstract This paper describes straightened-up concepts and methods of programming education on the basis of its actual implementation at front software development sites. The mainstay of programming education lies in letting trainees recognize what is a good, easy-to-understand style of program by means of learning through their repeated rounds of modifying their first-written program codes with the use of exercises for programming. Then, it is important that instructors or instructresses are eager and well prepared to share experience with the trainees in the progress of their growth in addition to having to select well-edited textbooks and to have a good knowledge of each trainee's quality and making.

Even on-the-spot programming education requires trainers to have far-reaching views in mind as well as to incessantly adopt new training ideas and schemes. And it is also essential that trainees are properly qualified for their training agenda and ought to be encouraged to further develop their aptitudes. Another necessary proposition is to have them learn to act according to what is called basic behavior for good software development efforts.

1. はじめに

われわれのソフトウェア開発現場において実施したプログラミング教育について，その方針と実践を述べ，あわせて良いプログラムについての意見も添える．

筆者の基本的な考えは，次のとおりである．

高品質のソフトウェアをつくるためには，周到な教育・訓練が必須である．また，そのような教育・訓練は，教育を受ける人（以下，生徒と称する）が，将来専門家を目指してさらに技術的に成長するための重要なステップである．

さて，生徒は順次配属ないし（外部ソフトウェア会社の5名を含めて）採用した延べ11名であった．大学で計算機科学を専攻したり，ソフトウェアの専門家（Computing Professional）を目指した教育[1]を受けたような人はいなかった．しかし，無条件に人を受け入れたのではない．強い向上心を抱いているか，ないしそれを喚起できそうか，ある程度の資質をもっているか，ないし資質をカバーする何か(たとえば熱意)

があって将来見込みがありそうか，これらを見るために，採用に当たっては，必ず面談し，面談の場で聞き出す内容についてもいろいろと工夫し，改良を加えたが，その内容はここでは省く．

現実の開発においては，プログラムの出来・不出来が堅牢性，性能，保守性等の品質や，開発期間，経費に直接に影響する．一例を挙げよう．

「清少納言知恵の板（図1）7片のピース全部を重ならないよう，また内部に空所のないよう並べて，凸多角形が何種類，並べ方が何通りあるかを調べるプログラムをつくれ．ただし，回転や裏返しで同じになるのは重複して数えないものとする」

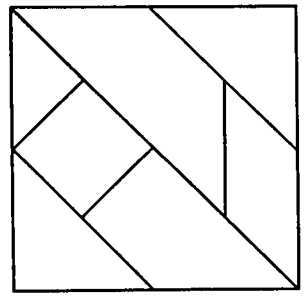


図 1　清少納言知恵の板
Fig. 1　Tangram of Seishonagon

これは，プログラミング技術の向上を促すために当社社内で毎年実施しているプログラミング・コンクールの課題である．応募登録者の約 1/3 が作品を提出した．残り 2/3 は，プログラムを完成できなかったと推定される．ここでは，ソフトウェア品質の一つの観点として，作品の実行時間に注目する．

16 種類の図形，並べ方は 531 通りと正解した作品の実行時間は，19 秒，数分，数時間，1 晩では答えが出なかったものまで，実にさまざまであった．その原因は，採用したコンピュータや処理系よりも，むしろアルゴリズムとデータ構造による計算量の差であった．この課題は極めてむずかしいというほどのものでもないかわり，やさし過ぎるというものでもないであろう．このプログラムを完成できる人にして，結果には大きな差があったわけである．

この結果をどう考えるか．われわれの実システム開発において，ひょっとしてこれに類似の事例が，あるいは，もっとひどい例があるのではないだろうか．

「大素人集団がソフトウェアをつくっている」というような言葉をどこかで耳にしたことがある．心当たりはないだろうか．

開発の現場という，予算や期間，その他種々制約の多い環境においても，そこで本当に必要な教育についてよく考え，方法を工夫し，実施すべきではないだろうか．その価値は十分あるのではないか．

以上のような反省から，われわれの実践したプログラミング教育について述べる．われわれの開発においては，主として C 言語を使ったことから，教育も C に即したものであった．

2. 教育事例と反省 —— 一つの簡単な場合

C言語を生み育ててきた人達による基本的な著作[2]の中に，次のような演習問題がある．

「入力文字列を，その中に二つ以上の空白が続けば，それらを一つの空白に置き換えながら，出力する複写プログラムを書きなさい」

〔解答例1〕

```
#include  "stdio.h"

main()                          /*  E1-7 */
{
    int  c;
    c=getchar();
    while (c!=EOF)
    {
        putchar(c);
        if (c==' ')
            while ((c = getchar()) == ' ')
                ;
        else
            c=getchar();
        }
    }
```

解答例1は，以前社内の「C言語研究会」で発表されたものである．発表者はシステム職10年以上の人であった．

このプログラムはどうやら正しく動くようである．でも，次のような疑問が起きる．

「これはやさしい演習問題である．コードを読み終えると同時に，その正しさが確信できるようなすっきりしたコードにならないのだろうか．getchar () が3回も使われる必然性はあるのだろうか．」

この演習問題は，良いコードとはどんなものであるかをまず考える上で，適当な題材であると思えたので，先徒たちの演習においては，必ず含めることにした．どの生徒も数回程度の書き直しによって，正しく動くコードには到達できた．しかし，できたコードの皆が皆，明快なコードというわけにはいかなかった．

空白文字の制御のために，オン・オフ・スイッチを使ったもの，空白文字を数えるカウンタを使ったもの，前掲のコードに似たものとさまざまであった．次はそれらの中の一例である．

〔解答例2〕

```
#include  <stdio.h>
main()
{
    int  c,pc;
    pc=1;
    while((c=getchar())!=EOF)
        if(c!=' '||c==' '&&pc!=' ')
            putchar(pc=c);
}
```

解答例2では，入力命令も出力命令も1個ずつである．コードの意味もわかりやすい．だが，まだ少し気になる点がある．まず，pc＝1が自己記述的 (self-descriptive) でない，つまり「pc (previous character) に空白でない文字を初期値としてセットす

る」が，素直に表現できそうである．さらに，次の点にも改良の余地がある．

```
(c!=' '||c==' '&&pc!=' ')
```

このif文の中の式は，次のif文と同値である．

```
(c != ' ' || pc != ' ')
```

この方は，余分な比較をしないから，より簡潔であり計算量も少ない．もう一つ，字が詰まりすぎていて，コードが視覚的に見にくいことが指摘できる．空白文字を適所に挿入してもっと見やすくできる．さらに，空白行を有効に活用すべきであろう．視覚的に見やすくすることは，錯覚によるミスの誘発を予防し，コードをわかりやすくする上で重要な要素である．

次に示すプログラムは，これらの点を改良した今一つの解答例である．

〔解答例3〕

```
#include  <stdio.h>
#define   NONSPACE_CHAR  'a'

main()
{
    int  c, pc;          /* pc: previous char */

    pc = NONSPACE_CHAR;

    while ((c = getchar()) != EOF)
        if (c != ' ' || pc != ' ')
            putchar(pc = c);
}
```

解答例3は，「入力した文字（c）が空白でないか，直前に入力した文字が空白でないならcを出力する」ことを自然に表している．

このように，たとえ数ラインのプログラムであっても，よく吟味すればさらに改良できる点が多いわけである．このようなプログラムの改良に対する多くの視点を演習を通じて，生徒に体得させることをプログラミング教育の考えの根幹とする．

3. 教育の方針と留意点

以上のような考えの上に順次参加してきた要員に対してその都度教育を実施していったが，前の生徒の教育で得られた経験を，後の生徒の教育に活かすように，工夫・改良を加えた．以下はそれらの方法・考え方を整理したものである．

1) 演習中心……プログラミングは実践・演習によって体得できる一面を持っている．通り一ぺんの理解では，実際にプログラムは組めないものである．狙いをはっきりさせて，その時点の生徒のレベルに相応しい演習問題を，所要時間も勘案し，選んで与えることが肝要である．

2) プログラムには改良が必要……実行結果が正しいだけでは良しとしない．前章で示したように，各演習問題をさまざまな観点から，それ以上改良の余地がないコードに到達するまで，生徒に改良を重ねさせること．本当に満足できる状態になり，生徒が必要な理解を得たなら次の演習問題へ進む．単にプログラミング言語の使い方を教えるだけが目的ではない．単に正しく動けば良いというだけではない．他にも重要な品質特性があるからだ[3]．さらに良いコードにならないか生

徒に考えさせる．生徒を鼓舞し，適切な示唆を与えてみる．簡単な演習問題でも改良を重ねて，教育の実際ではたとえば改訂版が9になることもある．

コードの改良における技術的な観点として留意した事項を，若干の注釈を加えて以下に列記する．

① 制御構造が簡単か．

単に構造化プログラムの3構造のことを言っているだけではない．if文の入れ子，case文の使い分け，ループ制御に関すること等いろいろある．整然としていないコード，言わばゴチャゴチャしたコードには，しばしば誤りが内在している．もっと良いコードが存在するはずという確信を抱いてほしい．そのようなセンスを持ってほしい．すっきりしたコードに書き直すことによって，誤りもなくなり，効率も良くなったという経験は貴重である．

② 良いアルゴリズムを採用しているか．(時間・空間) 計算量は適切か．

明らかに余分な計算量を伴って正しい結果を出して良しとしている例を挙げる．前出教科書[2]の演習問題である．

「整数nをbビット右に回転する関数rightrot (n, b) を書きなさい」

〔解答例1〕

```
rightrot(n, b)
unsigned  n, b;
{
    for (; 0 < b; --b)
    {   if (n & 0x0001)
            n = n >> 1 | 0x8000;
        else
            n >>= 1;
    }
    return (n);
}
```

アルゴリズムをよく検討して，改良する余地は本当にないのか．吟味する習慣をつけたいものである．次のコードは解答例1に対する改良版である．計算量は激減している．

〔解答例2〕

```
#define  WORD_LEN  16

rightrot(n, b)
unsigned  n, b;
{
    if (b > WORD_LEN)
        b %= WORD_LEN;
    return (n >> b | n << (WORD_LEN - b));
}
```

③ 変数や関数の名前は十分に吟味しているか．

名前自体が説明する重要性は，もっと強調されてよい．次に例を挙げる．

if (flag1 == 1) ……改良の余地あり

if (inword == YES) ……意味をわかりやすくした例

関数の名前や変数の名前の吟味を怠って，注釈によってわからせるようなことは，初期の段階でしっかり戒めておくべきである．

④ 適度の大きさの関数になっているか．

多くの機能を持った1個の大きな関数にしないで，ある単位に細分し複数の関数にして構造化する．

たとえば，一つの目安として，24ライン（1CRTディスプレイに入る程度）大きくても60ライン（プリント・シート1ページに入る程度）であると，見通しがよい．これに反して，そのような意識なしにずらずらと書かれた1個400ラインの関数では，解読に2日もかかったりする．まとまりのある適度なサイズの関数に分け，適切な関数名が付けられることは絶対に必要である．

図2の例を見てみよう．

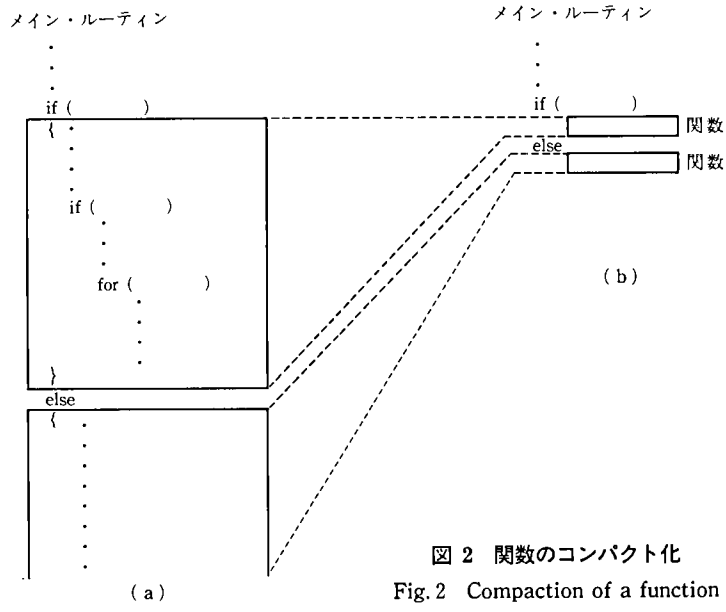


図 2　関数のコンパクト化

Fig. 2　Compaction of a function

メイン・ルーチンの中に，あるまとまった処理のコードが多数混入している(図2(a))のを，その処理の内容を明快に表す名前の関数にして，図2(b)のように緊密にした場合，このメイン・ルーチンの見通しは劇的に良くなる．また，単機能の小さい関数はコードの再利用上も有利である．

⑤　外部変数を多用しすぎてないか．

外部変数を多用することは，プログラムの解読性に悪影響を及ぼすだけでなく，関数流用の妨げになることも強調すべき点である．

⑥　移植性のない表現になってないか．

このようなことを，常に意識してコードを書けということである．

⑦　空白や空白行等を用いて，コードを見やすくしているか．

この他にも，字下げ，大文字/小文字の使い分け等がある．先にも触れたように，コードが視覚的に見にくいと，錯覚等によるエラーを誘発しやすい．開発現場では，これらの点は大切である．なお，可能ならこの辺りまで十分に配慮が行き届いた教科書を選びたいものである．

⑧　テストの限界と効用について実地に考えさせる．

テストそのものは品質を上げるものではない．それはプログラムがある範囲

で，たとえば，あるテスト・ケースで正しい，ないし正しくないことを示すものである．実用に供するある程度の複雑度を有するプログラムでは，あらゆる組み合わせのケース数は通常厖大であり，すべてテストしてしまうことは，現実にはほとんど不可能である．何万ケースのテストをしようが，依然として未テスト部分が残る．未テスト部分については，リリース後ないし本番稼働後，あるいは3カ月後，あるいは半年後，あるいは1年後，…と不具合が発生する可能性がある．4年，5年と使われて，実際に稼働する範囲では問題のない，いわゆる「枯れたソフトウェア」となり得る．しかしながら，できることなら，そんなに長い年月待たないで不具合のないソフトウェアを実現したいわけである．

この点について，多くの実験や考察が行われてきた．最も見込みのありそうな方法がCleanroom Software Development Methodであるとの報告[11]もある．この方法における最も重要な技術の一つは「読むこと」である[12]．このことは，次のことを連想させる．数学の定理は，実験や測定をいくら繰り返しても，全ケースを照査しない限り，正しいとは断言できない．定理は証明を「読むこと」によって正しいか，正しくないかが確認される．

プログラミングにおいても生徒には，「読むこと」の重要性を教える．コードをろくに確認もせず，すぐコンピュータを使ってテストすることに頼りがちになるのを戒める．

一方，初心者教育におけるテストの効用もある．論理の組み立てにおいて，生徒が錯覚したり，例外ケースを見落とすことは起こりがちである．生徒がテストをするとき，思考の未熟さに気付くことはよくある．たとえば，再帰関数（ある関数の実行の途中で再びそれと同じ関数呼び出しが行われるような関数）の演習では，メイン・ルーチンからの1回目の呼び出しではうまくいくが，2回目以降はNGというケースがしばしばあった．

どういうテストをするべきかをよく考えないで，慢然と粗末なテスト・プログラムを書いたケース，テスト結果を「眺めた」だけで細部をよく確認しなかったケース，簡単な修正だからと再テストしなかったために別の誤りを組み込んだままにしたケース等，生徒は実際にさまざまな経験をした．

そのような経験をし，都度考え，次回はより武装するといったことを数多く重ねることを通じて，思考の深さと幅が改善されていくものと思う．

このように，10ライン～20ライン程度のプログラムであっても，これ以上は良くなり得ないコードになるまで，何回でも改良する．プログラム・リストと実行結果は改良途中のものも含めてすべてファイルに保存させる．したがって，最初に作成したコード（版1）と，最終的に到達したコード（版n）が，どう違うか，その悪さ・良さを生徒自身がはっきりと確認することができる．つくったのはほんの小さなプログラムではあるけれど，それはその課題に対する最高のプログラムであるという充実感を生徒は味わうことができる．誇らしくも思うであろう．

「わたしは，わたしたちのデッサンを額縁に入れさせる．きれいなガラスで覆って，人が手で触れないようにし，いつもそうした状態に置かれてあるのを眺

めて，それぞれ自分の描いたものを粗末にしないように心がける．20回も30回も繰り返し描いたもののデッサンの一つ一つを見れば，描いた者の進歩の跡がわかるような順序に部屋のまわりに並べてかける[8]」

それと同時に，コードをよく吟味することの意義について，そして良いスタイルのプログラムについての理解を深めるということが期待できる．

3) 良い教科書を選ぶこと……教科書の中の材料がよく吟味して選ばれていて，周到な説明，たとえば生徒が誤りを犯しやすい箇所において注意を喚起するような何らかの配慮に加えて，進度に応じた適切な演習問題が要所に配置されていること等が必要である．

さらに，単に言語の使い方を示すに留まらず，良いスタイルのプログラムや有用なアルゴリズムを合わせて教えてくれるような教科書であればなお望ましい．生徒が陥りやすい誤りや，理解が不十分であることを繰返し鋭く教えてくれるような実例や演習問題が盛り込まれているような教科書[7]を併用するのも有意義である．たとえばCに関する本は今や何十種類と出版されているが，上記のような観点から選ぼうとすると，さてどれだけ残るだろうか．

われわれのところでは，生徒の興味に応じ，アルゴリズムや計算量について解説している成書[4]や論文，雑誌の記事，その他プログラミングやソフトウェアの作法について論じている本[5][6]等を生徒に紹介するようにしている．そのようなものがあると気付かせること，さらにそうしたものを探し求めること，良書を買い求めるという良い習慣を付けることも，最初は教えられるものであると思う．なお，それらの本が，実際に職場の中の自発的な夜の勉強会でのテキストとして使われたことを付記しておく．

4) マンツーマン方式……プログラミングの経験のあるなしにかかわらず，マンツーマンで，C言語プログラミングの基礎は1か月で終えるように設定した．経験のある人が必ずしも進み方が良いとか，成績が良いということはない．端末に向かってどんどんコードを入れていくやり方に長年慣れてきた人の方が，むしろ結果は良くない．たとえば，変数の名前の付け方等，まったく無造作であった．単に動けばいい；計算量を考え，コードを吟味し，良いスタイルのプログラムをつくること等考えたこともないかの如きもあり，考えさせられるところである．

生徒の経歴，資質，興味，熱意，性格等を勘案して，速度，助言の仕方，採り上げる演習問題等を加減する．ある場合は，より挑戦的な問題を課し，関連する他の本も薦める．ある場合は，先の章へ進んでからもう一度，前の章の問題をやってもらう，等々．開発現場は，いろいろな人で構成されているのであるから，開発の進め方もさることながら，教育・訓練についてもそれぞれの違いに留意し，その人をよく観察し工夫しながら進めることが大切であると思う．プログラム改良のための助言も，生徒がよく考えることの手助けとなる程度に留めるべきだ．一つ一つ良いコードへ，ある時は苦吟し，時には楽しみながら，近づいていくプロセスを共に経験する．

「わたしは生徒に1人で楽しませようとは思わない．たえずかれと楽しみを分かつことによって，それをさらに楽しいものにしてやりたい[8]」

後になって振返ってみて，短かかったけれど有意義であったと懐かしむような価値ある期間にしたいものと願ってきた．

5) 標準化規約は，初めは示さない……われわれのところでは標準化規約をつくっている．他の人が作成したコードを保守したり，利用したりする上で，標準に基づいているかどうかは要点である．しかし，これは皆を枠にはめるものではなくて，最も良いと思えるやり方を皆で倣おうという趣旨のものである．さらに良いのがあればそれをどしどし標準に採り入れていきましょうと宣言しているものである．

その標準化資料は，生徒が一連のカリキュラムを終了してからでないと示さないことにしている．その段階になると，生徒もその標準の個々の項目についてより理解し，納得することができるからである．生徒自身が考えてみても，やはり同じその標準に行き着くであろうという内容なら，容易に賛成するであろうと考えるからである．進歩が十分でない期間に示してしまうと，自分でよく考えるという良い習慣をつけるのを阻害してしまうように思う．それに，単に従えと言われてその通りにするのと，その価値をよく認識して従うのとでは，その差は小さくないと思う．

6) 部品をつくって組み立てさせる……10 ライン程度の小さなプログラムを，部品を意識してつくらせる．教科書[2]の演習は次のものである．

「入力単語を発生頻度の降順に分類し，単語の発生頻度を印書するプログラムを書け．頻度は各単語の前に印字すること」

この演習は，二分木を使って解くのであるが，たかだか 7 個の関数（しかも各関数は 5〜20 ライン，平均 10 ライン程度の大きさ）に分けて解くことができる．このように比較的むずかしそうな問題でも小さく分割して，やさしい関数の集まりとして解くことができる．生徒はこの事実に目を見張る．小さな関数の集まりとしてつくることの意義と重要性を，この演習を通じて教える．

すなわち，問題を単機能の小さな関数に分割することにより，

① 手に負えないようなむずかしい問題でも，解く対象を個々の小さな機能とすることによりずっとやさしくすることができる．これは普遍性を持つ一つの方法論である．同じことを，職場内での LISP の勉強会でも参加メンバが味わっている．

② 修正が容易である．小さい方が解読も一般的に言って容易であるし，単機能の関数の修正は絡みが少なく，より容易である．

③ 部品として流用しやすい．

開発現場としては，生産性，品質の面から部品化は非常に重要である．

7) 辞書を備えさせる……変数や関数の名前を，最適なものにするよう各自机上に辞書を備え，推敲の手助けとする．

なお，初期教育 1 か月で生徒がつくったプログラムの例を付録に掲載する．筆者がどういうことを意図してきたかが，より具体的に理解していただけると思う．

4. 次の教育はなにか

次の1か月間は，教育終了後アサインを予定しているアイテムを題材に，必要な関数やプロトタイプをつくることによって学習する．開発の分野に従属した部分(application dependent な部分) で必要となる次のうちのいくつか，すなわちファイル，画面やキーボードアクセス，グラフィックス機能，通信回線制御，タスク制御等の基本的な部分の習得に当てる．実際のプロダクトの開発において，この期間中に作ったものを活用できる可能性は十分にあり，そのような題材を選ぶが，主な目的はやはり教育である．

気をつけるべきは，せっかく前の1か月でみっちりやって身に付けたつもりの美しいコードを書く習慣が，元に戻ったり損なわれたりしてしまわないように注意することである．そのようなことは，現実に繰り返して起こったことであった．この過程で，本当に身に付くのかといったところである．

5. OJT

前章までのカリキュラムを終了すると，実際の開発に参加させる．最初は小さなやさしい関数をつくることから始める．この段階からは，いわゆる OJT である．

とくにコードレビューの場が良い OJT の場になっていると言える．なお，勉強のためにコードレビューに参加する工数は，そのプロダクトのコストにはカウントしない．能力向上のためのコストにする．

コードレビューの場で，演習ではない本物のプロダクト開発の中で，良いコードと悪いコードについて考えさせることができ，さらにコードレビューの意義をも認識させ，今後の開発における基本動作として身に付けさせるということが狙いである．たいした教育もせず，コードレビューもせずというようなことでは，それによる歪みは，開発過程でか，本番稼働の中でか，それとも保守過程においてか，どこかで大きな付けとなって，しっかりと払わせられるやも知れない．ところで，コードレビューを効率よく実施する上で，コードがわかりやすい読みやすいコードであるかは，非常に重要な点である．「わかりやすさ」は，単に保守段階で重要なばかりでなく，開発中においてもその意味は大きい[9][10]．

コードレビューを効率よく実施する上でも，プログラミングの周到な教育は重要である．

6. おわりに

開発現場のようにいろいろと制約のある環境下では，Computing Professionals 教育のような内容は望むべくもないし，不適当である．しかしながら，ここにおける教育でも，付焼刃的なものでなく，長期的な視野と配慮は必要である．基礎として必要な教育は，社内・社外の区別なく開発に参画する人すべてを対象とすべきである．

教育的効果という観点から，教育を受ける者の資質および熱意，もう一つ，まじめさが極めて重要である．目覚ましい進歩を見せている生徒も出てきている．生徒が終始，熱心に取り組んだ結果だと思う．高いモラルを保持することの効果を表していると思う．

さらに開発における基本動作とも言うべきものを，担当者に実地に体験させ，その意義を認識させることもプログラミング教育と併行して考慮すべきであり，それは有効である．そうしたことのために，種々の方法を考え出し，実施し，その結果をもとにさらにフィードバックして，工夫を加味しながら方法の改善に結び付けることが，月並みではあるが大切である．わかっている，知っているだけでは，何の役にも立たない．このことは，ソフトウェア開発の方法論として提唱されている種々の有効な手法についても言えることである．地道に試行することが大切である．

良いプログラムとはどういうものかについて，当社研究開発部 山崎主席研究員から多くのご教示をいただいたことに感謝の意を表したい．本稿は，それらを私なりに理解し，考察して行った実践をベースとしている．

[付　録]

```
/* C programから、すべてのcommentを取り除く。commentのnestは許さない。*/

#include <stdio.h>

#define YES 1
#define NO  0
#define DUMMY '¥0'
#define IN_NORMAL   0
#define MAY_COMMENT 1
#define IN_COMMENT  2
#define IN_STRING   3
#define IN_CHAR     4

#define comment_bgn(LAST, C) ((LAST) == '/' && (C) == '*')
#define comment_end(LAST, C) ((LAST) == '*' && (C) == '/')
#define string_ind(LAST, C)  ((LAST) != '¥¥' && (C) == '¥"')
#define char_ind(LAST, C)    ((LAST) != '¥¥' && (C) == '¥'')

main()
{
    int c;
    int last;               /* 一つ前の文字              */
    int status;

    status = IN_NORMAL;
    last = DUMMY;
    while ((c = getchar()) != EOF)
    {   switch(status)
        {   case IN_NORMAL:
                status = chk_status(last, c);
                if (status != MAY_COMMENT)
                    putchar(c);
                break;
            case MAY_COMMENT:
                status = chk_status(last, c);
                if (status != IN_COMMENT)
                {   putchar(last);
                    putchar(c);
                }
                break;
            case IN_COMMENT:
                if (comment_end(last, c))
                    status = IN_NORMAL;
                break;
            case IN_STRING:
                putchar(c);
                if (string_ind(last, c))
                    status = IN_NORMAL;
                break;
            case IN_CHAR:
                putchar(c);
                if (char_ind(last, c))
```

```
                    status = IN_NORMAL;
                break;
            default:
                break;
        }
        last = c;
    }
}

chk_status(last, c)
int last, c;
{
    if (string_ind(last, c))
        return(IN_STRING);


    else if (char_ind(last, c))
        return(IN_CHAR);
    else if (c == '/')
        return(MAY_COMMENT);
    else if (comment_bgn(last, c))
        return(IN_COMMENT);
    else
        return(IN_NORMAL);
}
```


参考文献 [1] D. L. Parnas, Education for Computing Professionals, IEEE Comp. 23, 1, Jan. 1990, pp. 17～22.

[2] B. W. Kernighan, D. M. Ritchie, The C Programming Language, Prentice-Hall. 1978, (石田晴久訳, 共立出版 (株), 1981).

[3] B. W. Boehm, J. R. Brown, M. Lipow, Quantitative Evaluation of Software Quality, Proc. IEEE/ACM 2nd Int. Conf. Software Eng., Oct. 1976, pp. 592～605.

[4] A. V. Aho, J. E. Hopcroft, J. D. Ulman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974, (野崎, 他訳, (株) サイエンス社, 1977).

[5] B. W. Kernighan and P. J. Plauger, The Elements of Programming Style, (木村泉訳, 「プログラム書法」, 共立出版 (株), 1976).

[6] B. W. Kernighan and P. J. Plauger, Software Tools, (木村泉訳, 「ソフトウェア作法」, 共立出版 (株), 1981).

[7] A. R. Feuer, Cパズルブック, 村井純訳, アスキー出版局.

[8] J. J. ルソー, 「エミール」, 今野一雄訳, 岩波文庫, 1962.

[9] 勝田祐輔, 全ラインコード・レビューの定着化, SEA MAIL：Newsletter from Software Engineers Association, Vol. 5, No. 5, 1990, pp. 18～22.

[10] Y. Katsuda, The Significance of executing codereview for the entire source code, Proceedings of The Third International Workshop on Software Quality Improvement, Jan. 1991.

[11] D. L. Parnas, A. J. van Shouwen, S. P. Kuan, Evaluation of Safety-Critical Software, Communication of the ACM Vol. 33 No. 6 June 1990, pp. 636～648.

[12] V. R. Basili, R. Selby, Cleanroom Software Development：An Empirical Evaluation, IEEE Trans. On Software Eng., Dec. 1987, pp. 1027～1037.

執筆者紹介 勝 田 祐 輔 (Yusuke Katsuda)

昭和35年三重県立名張高等学校普通科卒業. 43年日本ユニシス(株)入社. OUK 9400 OS担当. 46年大阪支店. 西日本地区フィールド技術支援, ユーザ・システム開発, OS 1100運用管理システムCOSTAR, DS 7/PW²のソフトウェア・プロダクト開発等に従事. 現在, 関西支社システム技術二部長. 技術士 (情報処理).

意思決定支援ソフトウェアFACILE1100の特徴と今後の方向

The Characteristics of FACILE 1100 Decision Support (DSS) Software and Its Future Developments

古田　茂

要　約　約9年前に開発された意思決定支援ソフトウェアFACILE 1100の開発の経緯をたどりながら，人間の考えに近い発想を持ったFACILE 1100の機能と特徴を説明する．

FACILE 1100は既存ソフトウェア群を有機的に連動させることにより，既存ソフトウェアに快適なマンマシン・インタフェースとグラフィック・インタフェースを提供するとともに，システム全体系の統合を計ることができる．

システム全体系の統合の結果として，さらに一歩進んだ意思決定支援システムを構築することができる．

Abstract　While keeping an eye on how and why the FACILE 1100 (FACILities for End users) Decision Support System (DSS) software product was developed about nine years ago, this paper explains the functions and characteristics of the FACILE 1100 designed to operate on the concepts and thoughts close to those of human beings.

Through its organic linkage of existing repertoires of software, the FACILE 1100 serves to provide a comfortable man-machine interface and user-friendly graphics interface for existing software so as to make it possible to build a total integrated system. Such total system integration brings about the successful creation of a new decision support system which is one more step advanced.

1. はじめに

FACILE 1100 (Facility For End Users) は，単純な問題から複雑な問題まで使える言語であり，統合プログラミング環境を持った言語として使用され成功してきた．FACILE言語によるプログラミングは，コンパイルとマップの作業をすることなく直ちに結果を見ながら，少しずつプログラムを組み立てていく．プログラムが完成するはるか前から部分部分を実行し，方針，構造，アルゴリズム等を検討しながら，プログラミングをしていく．プログラミングにおける綿密なシステム設計，プログラム設計等の伝統的な手法を必要とせず，伝統的な手法では達成できないような複雑，あいまいな問題にFACILE 1100は強力な能力を発揮する．

FACILE 1100は3次元データベース（キューブ）という考え方をもとにしている言語である．データの構造は項目，レコードの平面的要素に，奥行き（時間）を加えた3次元で表され，人間の考え方に近い構造と言える．人間のデータのとらえ方をそのままプログラミングできるFACILE 1100が，いかに強力な言語となっているかが理解できる．

本稿では，FACILE 1100の生い立ち，およびその特徴を述べるとともに将来の方向についても触れる．

2. FACILE1100 開発の歴史

市場流通ソフトウェアは，保守と次の時代の要求分析とを絶え間なく行っており，時代の流れを敏感にとらえ，時代の要求する機能を実現し成長していく．

成長した後に，外見（機能比較表等）で判断しようとすると見分けがつかないソフトウェアが多数ある．しかし，生まれ，育ちを見ることによって真の姿を捉えることができ，ソフトウェアの適用を正しく行うことができる．

本章では FACILE 1100 の真の姿を理解してもらうために，FACILE 1100 の開発の歴史を振り返りながら，FACILE 1100 がどのような目的で生まれ，またどのように成長してきたかを説明する．

2.1 開 発 の 歴 史

1) 1976 年：会話型によるシステム開発用ソフトウェアとして AID（An Interactive Data Processor）を開発．

AID は旧ユニバック総合研究所で作成されたもので，FACILE 1100 にも引き継がれている非常に重要な機能の一つである既存ソフトウェア群を有機的に連動させるということを目的の一つとしている．

「有機的に連動させる」を簡単に表現すると，「他の実行可能プログラムをサブルーチンを呼び出すように実行させることができる．」ということである．呼び出された実行可能プログラムが終了すると，呼び出しを行ったプログラムは呼び出しを行った命令の次の命令から，前の環境が引き継がれて実行が再開される（図 1）．

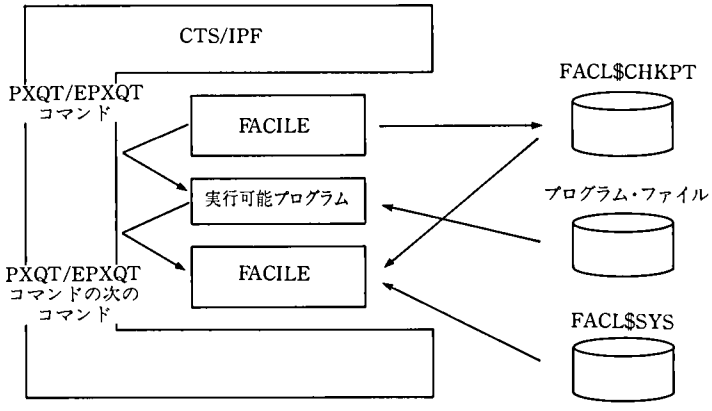


図 1 有機的連動

Fig. 1 Organic linkage

この機能は，既存ソフトウェアで全画面会話機能，グラフ出力機能等を持たないものに，それ自体にはなんら手を加えずに全画面会話機能，グラフ出力機能等を簡単に追加実現できることを意味している．たとえば，数理計画システムのソフトウェア AMPLE（Advanced Mathematical Programming Language Environment）に，簡単に全画面会話機能，グラフ出力機能を実現することができる．

2) 1982 年：AID ユーザの要望に応える型で新時代ソフトウェア（FACILE 1100）を開発．

新時代の端末として，UTS 50 漢字カラー・グラフィック・ターミナルが製造，販売され始め，AIDにZTS50の機能を反映する形でFACILE1100が開発された．

3) 1983 年：FACILE 1100 の最初のリリースで，①漢字の使用，②全画面会話，③グラフ出力，④構造化プログラム等の機能を実現した．

FACILE 1100 は，既存ソフトウェア群を有機的に連動させるということを目的の一つとしているが，FACILE 1100 自体も次のような既存ソフトウェアの有機的結合体として作成された．

全画面会話　DPS 1100（Display Processing System 1100）
グラフ出力　BGL 1100（Business Graphic Library 1100）
テキスト・エディタ　CTS 1100/IPF 1100
開発言語　PLUS 1100（Program Language for UNISYS System 1100）

また，FACILE 1100 の構造化プログラム機能は，プログラム設計言語と言えるほどで「設計者が自らプログラムする言語」を実現している．

4) 1984 年：プログラム制御コマンド（サブルーチン関連）処理の強化を行った．

5) 1984 年～85 年：大規模データ処理コマンド(EXTRACT コマンド)，簡易処理系の追加，条件式の追加を行った．

簡易処理としては，全画面会話指示によるデータ表示，グラフ作成，経営科学手法の適用，作表を可能としている．また簡易機能を使用して行っている作業は，作業履歴が FACILE 1100 のコマンドとして保存され，プログラムまたはプログラムの一部として使用することができる．結果を見ながら少しずつプログラムを組み立てていくということから，さらに一歩進めたプログラムの開発形態を実現した．

また，特記すべき機能として条件式がある．条件式は手順的な選択構造を簡単に表現できるもので，人間の思いつく形の選択構造をそのまま次式のように表現できる．

[条件 1 →式 1 ：条件 2 →式 2 ：条件 n →式 n：その他]

6) 1985 年～86 年：この頃すでに意思決定支援システムと言うものが叫ばれており，FACILE1100も意思決定支援システム構築用第四世代言語として使われ始めた．

意思決定支援システムと言っても，年に一回経営者が行う意思決定から，現場の担当者が時々刻々行う意思決定まで多岐にわたる．FACILE 1100 は，現場の担当者が時々刻々行う意思決定に使われることが多く，オンライン・リアルタイム・システムと同等の即応性が求められた．

また，データ処理は 3 次元データベース（キューブ）を，データ加工コマンド（リレーショナル・オペレーション）を複数個組み合わせて処理するため，キューブ（オムニバス・エレメント）をデータベース（ファイル）から探し出す入出力が余分にかかる．

FACILE 1100 のレベル 6 R 1 まで効率改善は，すべて外部記憶装置との入出力をいかに少なくするかと言うことであった．外部記憶装置との入出力回数を減らすことは，CPU タイムを減らすこととは比較にならないほどの効果を期待できるためである．外部記憶装置との入出力を少なくするためにはさらに CPU タイム

を消費することになるが，CPU のスピードアップは外部記憶装置のスピードアップとは，比較にならないほど進んでおり，CPU に負荷を負わせても外部記憶装置との入出力を減らすこととした．

7) 1987 年～88 年：CF，CIR-X のリリースを行った．

CF（Client Facility）は，システム部門がバックログ解消ツールとして使用する言語と，エンドユーザ部門がアプリケーションを開発するために使用する言語とは，異なるという考え方から作成された．

CIR-X は，FACILE 1100 のシステム部門向けバックログ解消ツールとしての方向をさらに明確にしたもので，会話型日本語文字情報検索システム CIR-J (Conversational Information Retrieval System-Japanese) のデータベースを FACILE 1100 で扱えるようにした．本機能は FACILE 1100 の既存ソフトウェア群を有機的に連動させるという原点に戻って開発したものである．

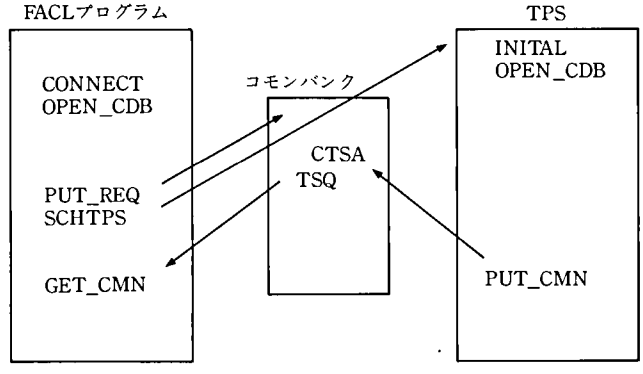
CIR-X が，FACILE 1100 の次のレベルへの架け橋であった．

8) 1989 年～90 年：会話処理系の充実，TIP（Transaction Interface Package）プログラム・インタフェースのリリース，RDMS インタフェース 1 R 1，2 R 1 のリリース，少量データ処理効率の改善，プログラム呼び出し効率の改善を行った．

会話処理系の充実は，FACILE 1100 の全画面会話機能を受け持っている DPS 1100 を最新レベルの 3 R 1 にしたものである．DPS 1100 の最新レベルは会話画面を作成する部分が大幅に改善されており，会話画面の作成がさらに容易になっている．

TIP プログラム・インタフェースは，既存ソフトウェア群を有機的に連動させる機能をデマンド・バッチの処理モードの領域から，さらに TIP の処理モードの領域まで拡張したものである（図 2）．CIR-X では，FACILE 1100 の処理効率を維持しながらデータベース・システムの機能を組み込む最善の方法として，コモンバンクを使った TIP プログラム・インタフェースを開発した．

RDMS インタフェースは，TIP プログラム・インタフェースを使って開発され



CTSA：Clear Test and Set and Active
T S Q：Test and Set Queuing

図 2　TIP プログラム・インタフェース

Fig. 2　TIP program interface

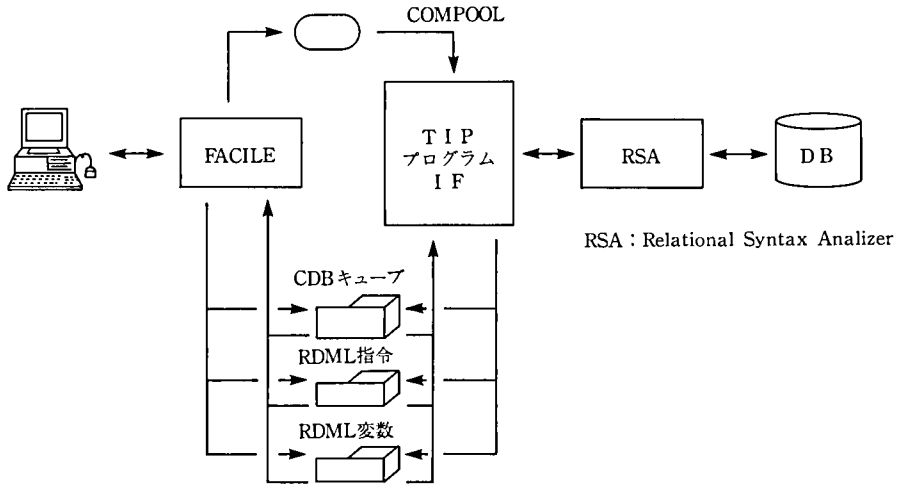


図3　RDMS インタフェース
Fig. 3　RDMS interface

たものである（図3）．SQL（Structured Query Language）をキューブで与え，RDMS のレコードをキューブで得るものである．

少量データ処理効率改善とプログラム呼び出し効率改善は，データとプログラムの検索方法を OS 1100 の標準ライブラリを直接使うように変更し，さらにメモリに記憶する量を大幅に増やした．

とくに，データ処理部分はコードを書き直し CPU タイムの改善も計っている．

以上が，FACILE 1100 の開発の歴史である．

現在の最新レベルである 6R2 はレベル 1R1 と比較すると，処理効率で3倍，機能で4倍程度になっている．

また，平成2年度末での導入状況は 35 ユーザ 51 セットである．

3. FACILE1100 の機能

FACILE 1100 は3次元で表されるデータベースを中核とし，「プログラミング」「デバッグ」「テキスト・エディタ」「データ入力・修正」「データ検索・加工」「統計・分

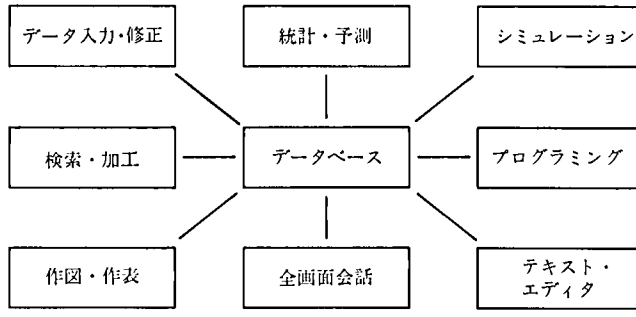


図4　FACILE1100 の機能概要
Fig. 4　Function chart of FACILE1100

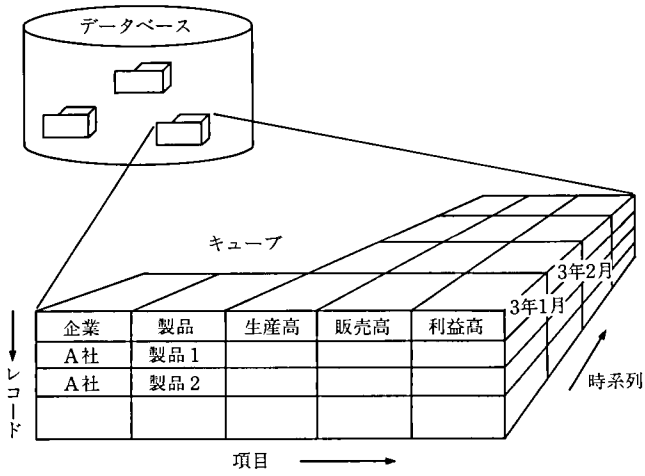


図5 FACILE1100のデータベース（キューブ）

Fig. 5 Database of FACILE1100 (CUBE)

析・予測」「シミュレーション」「全画面会話」「作図・作表」等の統合プログラミング環境を持つ言語である（図4，5）.

3.1 プログラミング機能

FACILE 1100言語によるプログラミングは，明確に機能分化，単純化されたコマンドを組み合わせて行われる．コンパイルとマップの作業をすることなく直ちに結果を見ながら，少しずつプログラムを組み立てていく．プログラミングの過程はログが取られており，取られているログを制御構造の中に埋め込んでプログラムを完成させる．制御構造を含んだプログラムは，コンパイル後制御構造の部分のテストがされ単体テストが終了する（図6）.

コンパイルされたプログラムは，マップすることなしに他のFACILE 1100言語で作られたプログラムから呼び出すことができる．この機能は，FACILE 1100言語で作られたプログラムに保守の容易性をもたらしている．

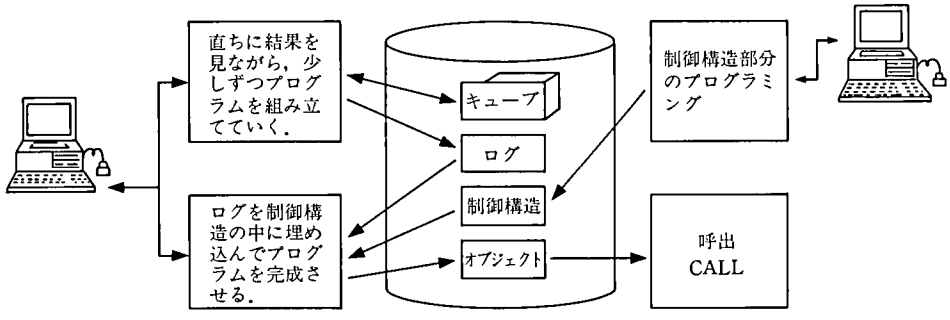


図6 FACILE1100のプログラミング

Fig. 6 Programming of FACILE1100

3.2 デバッグ機能

デバッグ・ツールとして実行トレース，キューブ変遷トレースの機能を持つ.

実行トレース機能は，実行されているコマンド名を出力する機能に加え，実行時のIO回数，CPUタイム等を出力する機能を持っており効率を測定することが可能となっている．

FACILE 1100では，非常に簡単にプログラム作成が可能なためプログラムの構造によっては，効率の悪いものも簡単にできてしまう．このような時にも実行トレースが役に立ち改善すべき点が明確に判断できるようになっている．

キューブ変遷トレースは，直ちに結果を見ながら，少しずつプログラムを組み立てていくために非常に重要な機能である．簡易データ表示機能を使うと，FACILE 1100との会話画面がコマンド入力部分と，キューブ表示部分の二つのウインドゥに分割され，コマンド入力部分に入力されたコマンドの結果が，キューブ表示ウインドゥに表示される（図7）．

```
キューブ名                レコード   項目   時系列    期数   ホールド   表示

COMMAND?

キューブ名   FACL•DEMO.GYOKAI
レコード数 56  項目数 14  時系列  Y'76'..Y'82'          現在の時系列   Y'76'
COMPANY GOODS  PRODUCT    SALES    PROFIT  P_RATIO  S_RATIO PF_RATIO  P_SHARE

KA      H1       16822    31689     3178   18.174   18.286   18.920   39.090
KA      H2       10974    21642     2256   11.856   12.488   13.431   32.207
KA      H3       13751    24029     2004   14.856   13.866   11.931   35.421
KA      H4       10505    19443     1682   11.349   11.220   10.014   36.571
KA      H5        8492    15893     1351    9.174    9.171    8.043   30.358
```

図7　簡易データ表示画面

Fig. 7　Display image of easy data display

3.3　テキスト・エディタ機能

FACILE 1100は専用のテキスト•エディタを持っていない．FACILE 1100のテキスト•エディタは，CTSまたはIPFという標準ソフトウェアを使用する．CTS/IPFというコマンドでFACILE 1100専用のテキスト・エディタであるかのように使用し，CTS/IPFの処理を終了するとCTS/IPFを呼び出したコマンドの次のコマンドから実行が再開される．新たな知識を学ぶことなく，普段使っているテキスト・エディタを使うことができる．

3.4　データ入力・修正機能

FACILE 1100は既存データを処理の対象とすることを基本としている．そのために次のような機能が用意されている．

- MAPPERデータベース入力
- DMSデータベース入力
- RDMSデータベース入力
- COBOL（SDF）ファイル入力
- カードイメージ・ファイル（SDF）入力

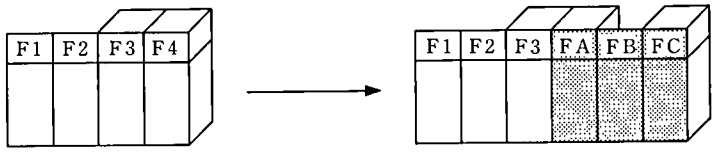
・端末全画面入力・修正

COBOL，カードイメージ・ファイルについてはSDF（System Data File Format）をキューブに変換する基本機能を使って行う．他のデータについてもSDFファイルを経由してキューブに変換できる．

3.5 データ検索・加工機能

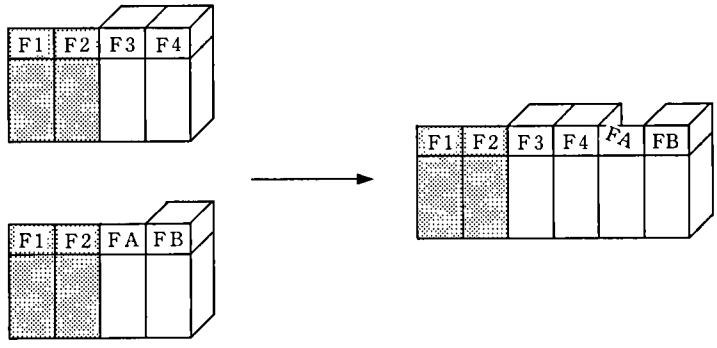
既存ソフトウェア群を有機的に連動させる機能の次に，特徴的な機能としてあげられるものがデータ検索・加工機能である．

データ検索・加工機能はリレーショナル・オペレーションを基本としたものとなっており，キューブとキューブの合併等も実現されている（図8，9）．



TRANSFORM CUBE1 CUBE2 ‥F3,^
FA*/I, FB/I, FC*/I
キューブのデータ構造を変更し，項目間の演算を行う。

図8 キューブの変化（TRANSFORM）
Fig. 8 TRANSFORM command



JOIN CUBE1, CUBE2 CUBE3
F1, F2
二つのキューブを，指定キーで対応するレコード同士を合併する．

図9 キューブの変化（JOIN）
Fig. 9 JOIN command

代表的な機能として次のようなものがある．

・検索・加工	・加　工	・その他
構造変更	演　算	キューブの更新
分　割	整　列	時系列のレコード方向への展開
合　併	集　計	レコードの時系列化
	累計，番号付け	会話型処理

3.6 統計・分析・予測・シミュレーション機能

基本統計・相関分析，回帰分析，季節調整，主成分分析，シミュレーションの機能

を備えており，結果を FACILE 1100 の作図・作表機能等につなげることができる．他の統計・分析・予測機能を必要とする場合は，既存ソフトウェア群を有機的に連動させる機能を使って相当するソフトウェア (MASCOT，SUFICS 等) を拡張機能として使用できる．

3.7 全画面会話機能

全画面会話機能は画面の定義とプログラムの独立性を特徴としている．

画面定義は，端末の画面に必要な様式を描くことによって行われる．実際に使用する画面の様式を端末に向かって定義するためイメージしているものと，実際にできたものとのズレがない．

プログラムでは定義された画面を番号で対応付けして使用する．データのやりとりは画面番号と対で指定する変数で行い，画面の様式はプログラムの中には現れない．

したがって，データのやり取りに関係のない画面の様式を変えても，プログラムは変更する必要がなく，データのやり取りの順序の変更もプログラムには影響を及ぼさない（図 10)．

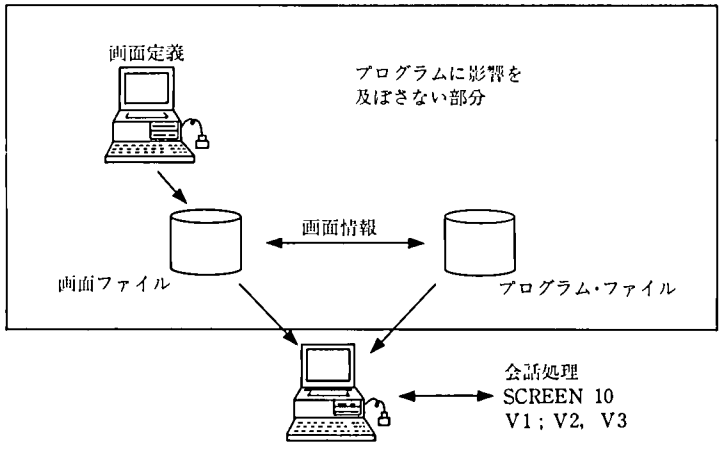


図 10　画面定義と会話処理

Fig.10　Display definition and interactive process

3.8 作 図 機 能

FACILE 1100 の作図機能はキューブの内容を見やすい形で表示する．出力情報をいったんファイルに保存し，別のコマンドで表示・出力する．同じものの出力・表示は出力情報を作り直すことなく可能である．

全画面会話機能で以下に示すグラフを作成することも可能で，通常のグラフはこの機能を用いて作成する．

棒グラフ
積み重ね棒グラフ
折れ線グラフ
層グラフ
ヒストグラム
積み重ねヒストグラム

散布図
円グラフ
レーダ・チャート
帯グラフ
地図グラフ
組み合わせグラフ

会話を通して作成した手順はFACILE 1100のプログラムとして生成され，内部的に実行される．生成されたプログラムは他のプログラムに組み込むことが可能で，必要ならばテキスト・エディタを使って追加指定を行うこともできる（図11）．

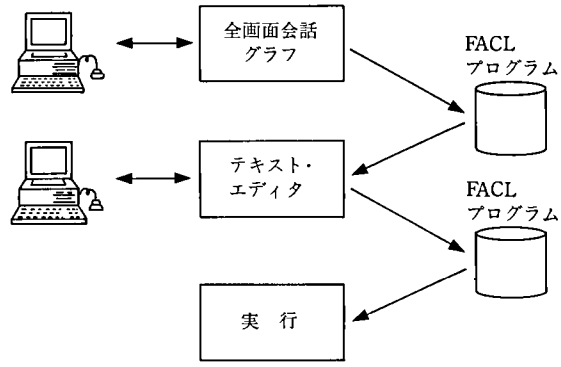


図11　作図の手順

Fig.11　Procedure of drawing a graph

3.9 作 表 機 能

帳票の作成・表示，全画面会話機能等，作図機能と同じ機能を持っている．

FACILE 1100の帳票は，項目の内容による縦・横展開として捉えられる．項目内容のキー割れによる集計計算の指定が可能で，明細集計まで行うことができる．

4. FACILE1100の今後の方向

FACILE 1100は次の時代の要求分析を絶え間なく行っており，時代の要求する機能を次々と実現している．

現在作業を進めている項目の代表的なものは，次のようなものである．

1) FACILE1100のTIPプログラム化……FACILE1100の処理モードはデマンドである．デマンドはプログラム開発には非常に有効な処理モードであるが，単位時間内に大量のデータを処理するためには資源的な環境が整っていなければならない．FACILE 1100は，処理対象の広さから基幹に近い業務に使用されることが多い．そのため，端末台数が多い処理，または応答性が要求されるものにも耐えられるように，FACILE 1100の処理モードをTIP化しようとしている．

開発はデマンドで，処理はTIPでと言うことを実現しようとしている．

2) BGL/TIP……BGL/TIPはFACILE 1100のTIPプログラム化の第一段階として作業されているもので，作図機能の出力部分をTIPに対応しようとするもの

である．

今までは，端末の性能・回線スピード等の制限からTIPの処理モードでグラフを端末に出力すること等は考えられなかったが，ここにきて端末の性能・回線スピードの改善が目ざましく，環境が整ってきたため実現できる状態になった．

この機能はFACILE 1100のTIPプログラム化の一貫であったが，BGL/TIPに対する要望がFACILE 1100のTIPプログラム化以上に高まっており，開発の優先順位に影響を及ぼし始めている．

5. FACILE1100の将来構想

UA（Unisys Architecture）に基づいた，①異機種間の相互運用性の向上，②ユーザ・インタフェースの共通性の向上，③開発生産性向上を中心にワークステーションFACILEの開発を考えている（図12）．

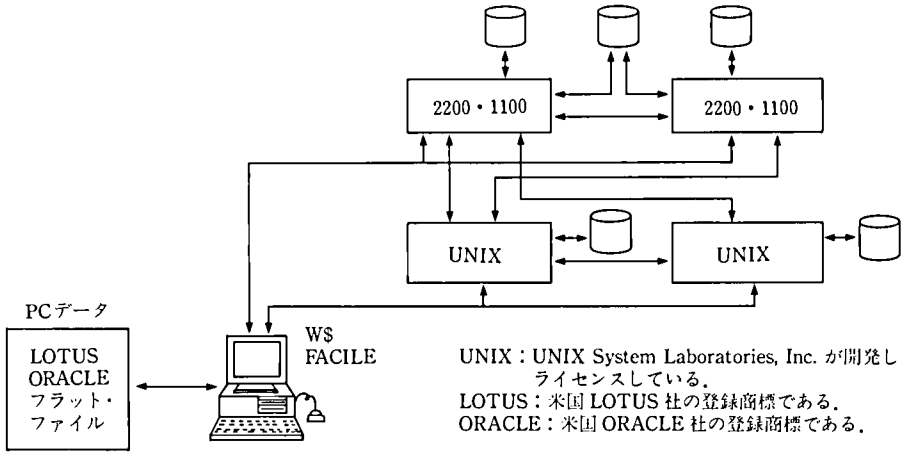


図12　ワークステーションFACILE

Fig12　Workstation FACILE

6. おわりに

第四世代言語（4 GL）の利用が急速にのびており，スポット業務の対応，開発時のデバッグツールとしての位置づけであったものが，今では企業の基幹業務の開発言語として，COBOLに代わるものになりつつある．

一方では，現場へのサービス向上のためのツールとしての役割もかなりの企業で成功している．

だが，4 GLを導入したすべての企業が，ねらい通りの効果をあげているわけではなく，導入したが使われていない，一部の人にしか使われていない，特定の人しか使用していない，という企業も数多くある．成功した企業では何らかの工夫により効果をあげている．

4 GLにも多種多様なものがあり，それぞれに得意とするところが異なる．導入時には機能の検討，効果予測等を行うが，導入後には大きな問題がなければ定期的なチェックがなされていないのが現状である．

導入した4GLが当初の期待通りの効果を発揮しているかどうかを知り，改善できる点は何かを把握し，次の開発に活かしていくことにより，さらに生産性向上につながる．

FACILE 1100においてもその適用を通して，生産性向上につながる機能を追加改良している．

執筆者紹介 古　田　　茂 (Shigeru Furuta)

昭和44年早稲田大学理工学部数学科卒業．同年日本ユニシス(株)入社．製造業のSEサービスに従事した後，FACILE 1100の開発に携わる．現在システム技術本部 応用ソフトウェア部に所属．

ショートモノポール・アンテナを用いた間接ESD測定

Indirect ESD Measurements Using Short Monopole Antenna

本田昌實

要　約　低電圧に帯電した金属物体間の間接静電気放電（ESD）によって生じた電磁界をミリメートルオーダの非常に小さなショートモノポール・アンテナを用いて測定した．その測定結果から，今までとらえられなかったインパルス界の特異な性格，すなわち超広帯域性と空間における単極性が明らかとなった．

Abstract The measurement of electromagnetic fields generated by indirect electrostatic discharge (ESD) between low-voltage-charged metal objects with the use of very short monopole antennas have clarified the so far unknown specific characteristics of the impulse field which features the ultra wide band and unipolarity.

1. 間接ESDによるEMI

オフィスや工場において，帯電する可能性のある金属物体の例としては表1に示すものがあり，帯電電圧が約3kV未満の場合，これらに人が触れても衝撃を感じることはない．

ところが，このような相対的に低電圧に帯電した金属物体と他の金属物体が何らかの理由により，接近・衝突して放電を引き起こした場合，これら金属物体の近傍に位置しているディジタル・エレクトロニクスにおいて非常にシビアな障害が発生してしまうことがある[1]．

常識と異なり，相対的に低電圧の間接ESD[2]に起因する電磁干渉（EMI）の威力は，

表1　金属物体の帯電の例

Table 1　Example of charged metal objects

帯電場所	帯　電　物
オフィス	・OA用椅子，スチールパイプ椅子 ・ファイルキャビネット，ロッカー ・事務机上の金属製筆記具 ・スチール製の引出し ・OAフロア（アルミダイキャスト） ・電気掃除機 ・アルミトランク，金属ボックス
工　場	・運搬台車 ・ベルトコンベア上の金属部品 ・ベルトコンベアのガイドローラ ・作業台上の“ワーク” ・工具，機械部品 ・無人搬送車上の金属部品 ・非接地のロボット（アーム部分）

Translated and reprinted, with permission, from Proceedings of the 1990 International Symposium on Electromagnetic compatibility Washington, DC, Aug. 21-23, 1990.

高電圧に帯電した人体から装置表面に対する直撃的なESDよりもはるかに強力であることを経験的に，また数多くの間接ESD実験からも得ている．つまり（電気）エネルギの大小とEMI作用は比例しないのである[3][4]．

間接ESDによって生じた妨害電流が，犠牲装置にまったく注入されないにもかかわらず“なぜ間接ESDは直接ESDよりも威力が強いのか．”という疑問が生じるのは当然であろう[5]．

低電圧の間接ESDによるEMI作用の理由を定常的な伝導電流の存在を大前提とした“人体モデル（HBM）”[6]や“家具モデル”[6]を使って説明することはできない．

したがって，間接ESDを調べるためには，波源（放電点）近傍においてこれによって生じた電磁界を高忠実度にとらえる技術が必然的に要求される．

2. ショートモノポール・アンテナ

現実の間接ESDに起因するEMI事例は，電磁界を放射する波源としての金属物体と妨害を被むる装置間の距離が約1m未満で発生していることを数多く経験している[7][8]．

非常に狭隘で限定された状況にあるこのような空間に対し，既存の広帯域アンテナ（たとえば対数周期アンテナあるいはバイコニカルアンテナ等）を持ち込むことはできない．

このため，コネクタを含む全体の構造がミリメートルオーダの寸法を有する超小型の受信アンテナを新たに必要とした．また，過去に行なった一連の間接ESD実験から，この過渡電磁界を観測するにはできる限りの広帯域周波数特性も必須である．

これらの諸要求に基づき数種類のショートモノポール・アンテナを製作し，波源近傍において過渡電磁界をとらえる“プローブ”として用いることにした．

写真1はエレメント（中心導体）長が5～25 mmのショートモノポール・アンテナである．これらのベースプレート（グランドプレーン）の大きさは何れも直径10 mmであり，この部分にSMA(F)コネクタが付属している．

このアンテナに接続する伝送線路として，今回の測定ではDC～18.5 GHzの帯域幅

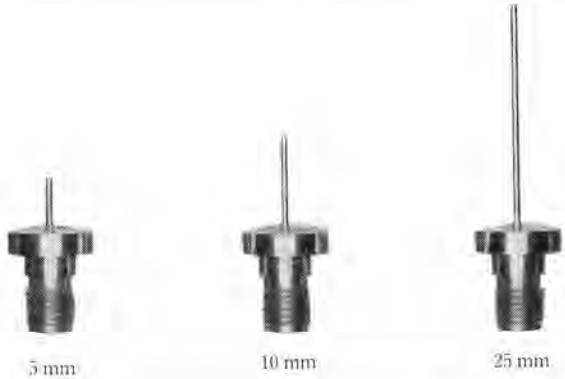


写真 1　ショートモノポール・アンテナの例（SMA コネクタ付）
Photo. 1　Short monopole antennas

を有するマイクロ波同軸ケーブル（潤工社製 DGM 024 シリーズ：Semi Rigid Cable UT-141 と同等の性能）を用いた．この同軸ケーブルは，Double Shielded 構造となっており，両端に SMA(M) コネクタが付属している．

3. 運動金属物体の ESD 測定

相対運動をともなう金属物体間で発生する ESD を調べるため，球状の金属物体を用いて以下に示す実験を行なった．

3.1 運動の方向とインパルス極性

真ちゅう球（直径 $\phi=12.7$ mm，静電容量 $C\fallingdotseq 0.7$ pF）2 個（a），（b）をスチロール樹脂製の名刺箱（94 mm×58 mm×21 mm，厚み 1 mm）に入れ，これを傾けることにより（a），（b）が回転しながら移動し，概略 500 V 程度帯電し，箱の角で双方が衝突して 1 回の ESD が発生する．

長さが 50 mm のショートモノポール・アンテナを垂直に固定し，ここから約 60 mm の位置で（a），（b）間の ESD が発生するよう名刺箱を操作する（図 1）．

このアンテナからは，長さ 1.5 m のマイクロ波同軸ケーブル（DGM 024-1500）で帯域幅 1.0 GHz のリアルタイムオシロスコープ（TEK 7104/7 A 29/7 B 15：50 Ω）に接続した．

実験の結果，真ちゅう球（a），（b）がこのアンテナに放射方向に接近しつつある時の ESD では写真 2（a）に示すように，リファレンス・グランドレベルから一瞬だけネガティブ⊖方向に行く一つの孤立したインパルスが認められた（振幅は 1.5 V くらい）．

しかし，同一測定条件にもかかわらず真ちゅう球（a），（b）がこのアンテナから（放射方向に）遠ざかる時の ESD では，写真 2（b）に示すように一瞬だけポジティブ⊕方向に行く一つの孤立したインパルスが認められた（振幅は 1.5 V 以上）．

さらにまた重要なことは，このインパルス極性を決定するのは単に運動の方向だけではなく真ちゅう球と接するプラスチックの種類（正確には摩擦帯列上の位置）にも

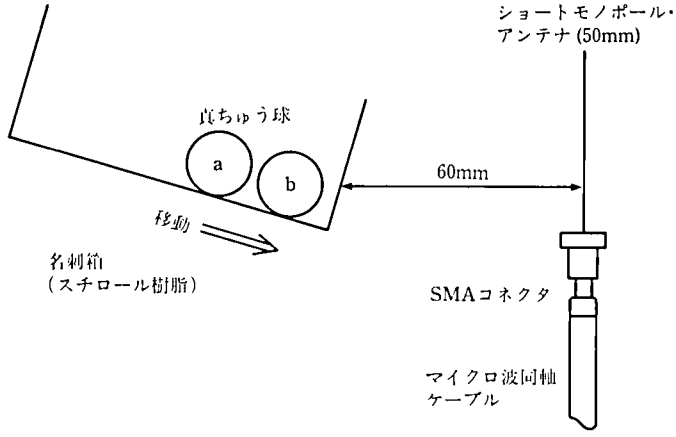
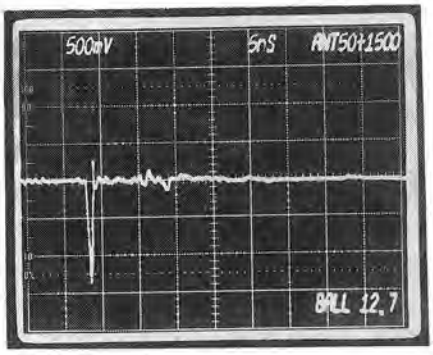


図 1　真ちゅう球（2 個）を用いた ESD インパルスの発生とその測定

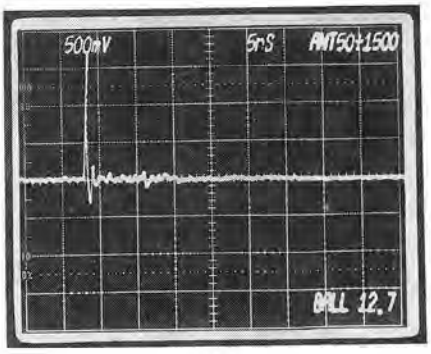
Fig. 1　ESD impulse generation using brass balls

依存することがわかった．たとえば，アクリル樹脂を用いた場合，得られるインパルス極性はスチロール樹脂のそれとまったく逆である．すなわち，アンテナに接近しつつある時の ESD では⊕インパルスが，アンテナから遠ざかりつつある時の ESD では



(a)

(Negative-going) H=5 ns/Div
V=500 mv/Div

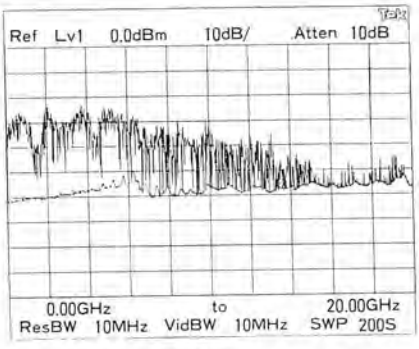


(b)

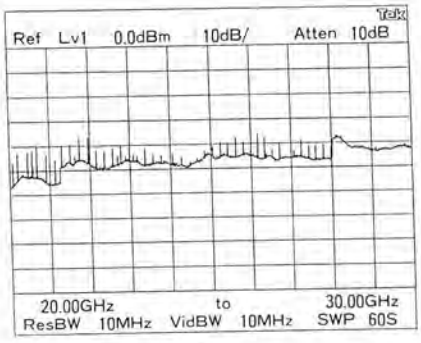
(Positive-going) H=5 ns/Div
V=500 mv/Div

写真 2　運動金属物体によって生じた単極インパルス

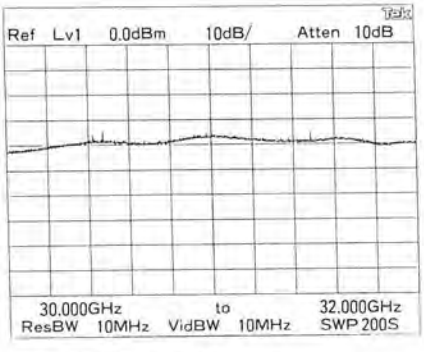
Photo. 2　Unipolar impluse from moving metal objects



(100 Hz～20 GHz)
H=2 GHz/Div, V=10 dBm/Div



(20 GHz～30 GHz)
H=1 GHz/Div, V=10 dBm/Div



(30 GHz～32 GHz)
H=200 MHz/Div, V=10 dBm/Div

図 2　超広域スペクトラム

Fig. 2　Ultra wide-band spectra

㊀インパルスが認められる．なお，この現象は真ちゅう球のかわりに同一寸法の鉄球を用いても変わらない．

3.2 運動の方向とスペクトラム分布

長さが25 mmのショートモノポール・アンテナを，33 GHzまで同軸入力で測定ができるスペクトラムアナライザ(TEK 2782)の入力コネクタ部(K-Connector)に“K-APC 3.5 Adapter”を介して直接接続した．同軸ケーブルは使用していない．

3.1節の実験と同じ真ちゅう球を用い，同様の方法によってESDを発生させ，一つの孤立インパルスに含まれるスペクトラム分布を調べた．

このアンテナ先端から10 mmくらいのところで発生するESDのスペクトラム分布は，真ちゅう球の運動の方向を問わず測定下限の100 Hzから上は17 GHzあたりまで認められた．約17 GHzあたりから上のスペクトラムは，この位置(約10 mm)では真ちゅう球(a)，(b)がアンテナから放射方向に遠ざかる時のみ出現することが多い(図2)．

アンテナ先端から距離約1 mmまで波源・真ちゅう球(a)側を近づけた時のESDでは，32 GHzまでのスペクトラムを含むことを確認した．この値は今回の測定で使用したスペクトラム・アナライザの測定上限(33 GHz)に非常に接近している．

4. 固定した金属物体のESD測定

アスペクト比の大きい低電圧に帯電した金属物体間のESDによるインパルス界を調べるため，以下の実験を行った．

4.1 孤立単極インパルス

垂直に保持した長さ150 mmの2本の銅パイプ(直径ϕ=12.7 mm，静電容量C=2.2 pF)(A)，(B)の先端部(ϕ=12.7 mmの真ちゅう球を半田付)を60 μmの間隙を保持して対向させ，銅パイプ(A)に50 GΩの直列抵抗を介して+1.2 kVを印加し，1秒間に数回程度放電させた．

銅パイプ(A)から水平距離20 mmの場所に，長さ25 mmのショートモノポール・アンテナのエレメントを上に向け銅パイプと平行に配置した．このアンテナに長さ1 mのマイクロ波同軸ケーブル(DGM 024-1000)を用いて帯域幅1.0 GHzのリアルタイム・オシロスコープ(TEK 7104/7 A 29/7 B 15：50 Ω)を接続した(図3)．

この条件下での放電に際しては写真3(a)に示すごとくリファレンス・グランドレベルから一瞬だけネガティブ㊀方向に行く一つの孤立したインパルスが認められた．振幅は2.5 V以上あった．

銅パイプ(A)に−1.2 kVを印加し同様に放電させた場合は，写真3(b)に示すごとく一瞬だけポジティブ⊕方向にいく一つの孤立したインパルスが認められた．振幅はやはり2.5 V以上あった．

このアンテナを銅パイプ(B)側の同じ位置に移動し同じエレメント方向にセットした場合，観測されるインパルス極性は銅パイプ(A)側で得られるものと反対になる．

この放電/測定条件(電圧V=1.2 kV，間隙g=60 μm，距離d=20 mm)におけるインパルスの立上り/立下り時間は約350 psecと測定されたが，この値は使用したリアルタイムオシロスコープ(TEK 7104)が有する固有立上り時間に等しい．

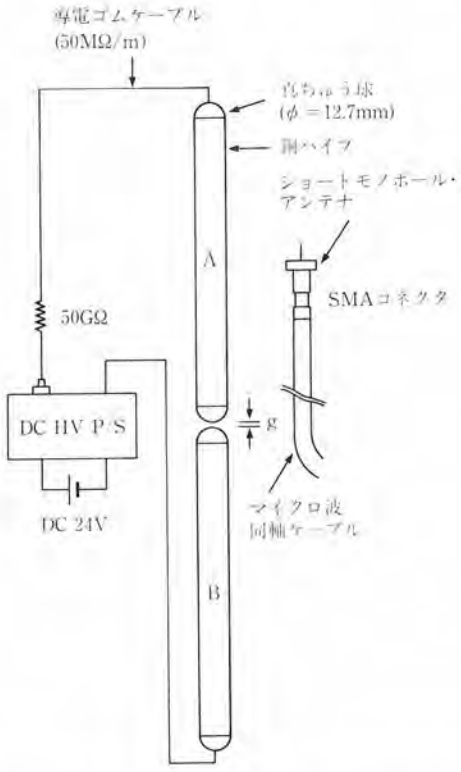
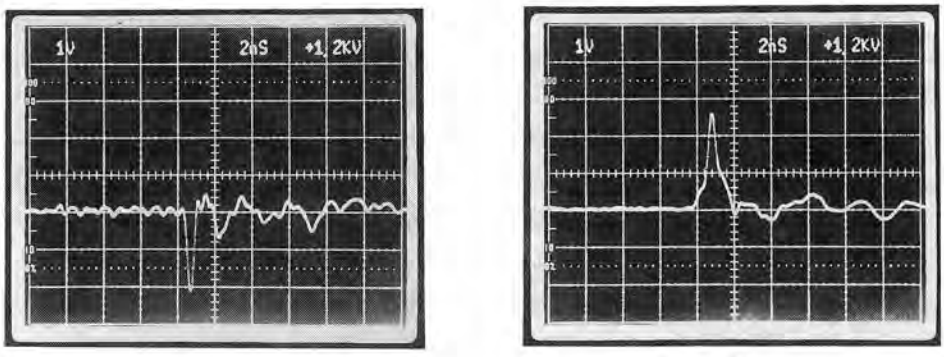


図 3　固定金属物体を用いた ESD インパルスの発生とその測定

Fig. 3　ESD measurement on fixed metal objects (copper pipes)



(a)

(Negative-going) = H = 2 ns/Div
V = 1 V/Div

(b)

(Positive-going) H = 2 ns/Div
V = 1 V/Div

写真 3　固定金属物体によって生じた単極インパルス

Photo. 3　Unipolar impluse from fixed metal objects

4.2　インパルス立上り時間

長さ 200 mm の 2 本の銅パイプ(直径 ϕ=12.7 mm, 静電容量 C=3 pF)(A), (B) の先端部(ϕ=12.7 mm の真ちゅう球を半田付)を 30 μm の間隙を保持して対向させ,

銅パイプ(A)に+550 V を印加し 1 秒間に 10 回程放電させた.

銅パイプ(A)の先端にある真ちゅう球から水平距離 10 mm の位置に長さ 5 mm のショートモノポール・アンテナをエレメントを下に向けてセットし，全長が 4.5 m のマイクロ波同軸ケーブル(伝搬遅れ時間：約 17 ns)と 20 dB 減衰器(HP 33340 C：DC～26.5 GHz) を介して，帯域幅 34 GHz のサンプリング・オシロスコープ (HP 54123 T)に接続した(写真 4)．このオシロスコープは 16 nsec のディレータイムを必要とする.

この放電/測定条件ではインパルスの立上り時間は約 20 psec，インパルス幅は約 40 psec が得られた (図 4).

これらの値は，今回用いた測定系の中で最も狭い帯域幅を有するマイクロ波同軸ケーブル (DGM 024：DC～18.5 GHz) のインパルス伝達特性に等しい.

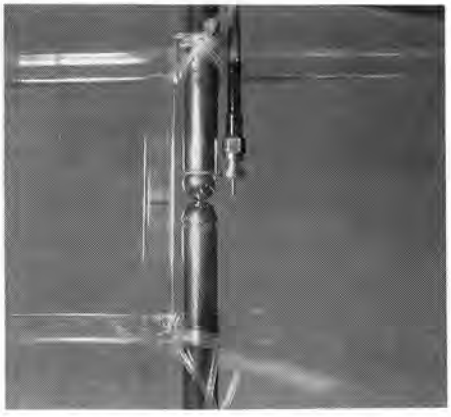
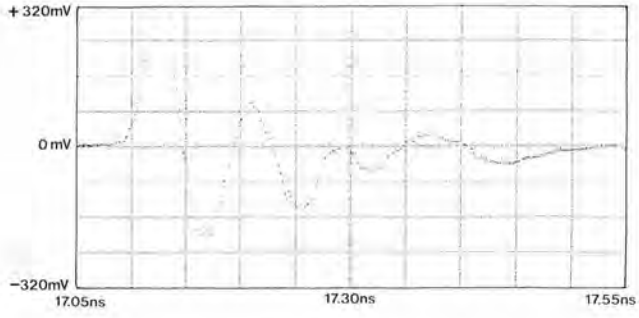


写真 4　ショートモノポール・アンテナ (5mm) と銅パイプとの位置関係

Photo. 4　Relative positions of copper pipes and a short monopole antenna



H＝50 psec/Div
V＝80 mV/Div

図 4　ショートモノポール・アンテナに誘起した電圧波形

Fig. 4　Voltage waveform on a short monopole antenna

5. 実験のまとめと検討

5.1 単極インパルスの存在

固定金属物体間でのESD，運動物体間でのESDを問わずこれらの近傍の空間には一瞬ではあるが一つの孤立した“positive going”または“negative going”の単極性界(Unipolar Impulse Field) が発生することがある．

この単極性インパルス界が観測されるには，以下の条件があげられる．

1) 帯電電圧が約3 kV未満と相対的に低く，したがって放電間隙が数100 μm以下で一回限りのスパーク放電が発生するとき．
2) 放電を引き起こす金属物体の寸法（形状）と同程度の距離以下の空間内に受信アンテナが置かれているとき．
3) 受信アンテナの構成が一つの金属物体の寸法と同程度か，またはそれ以下のエレメント長さを有するショートモノポール・アンテナであるとき．

5.2 超広帯域幅インパルスの存在

電圧が相対的に低く，しかも放電間隙が時間とともに減少しつつスパーク放電に至るESDでは，非常に立上りのするどいしかも強力なインパルス界が発生することがわかった．

時間領域および周波数領域の測定結果から波源でのESD事象は，その始まりから消滅までの全時間経過が“**ピコセカンドオーダ**”で推移していることを示している．

ミリメートル波を完全に含むこの超広帯域インパルスの存在は，間接ESDによるEMI作用を理解する上で一つの重要な手がかりを与えてくれるであろう[9]．

5.3 従来理論の限界

低電圧の金属物体間で発生する間接ESD事象を従来理論で説明することはできない．とくに時間領域の測定データから言えることは，すべての系の長さ・距離・空間の広がりと**光速 (C : 3×10^8 m/s) によるこれらの部分の進行速度とは十分比較し得る状態になっている**ということである．

たとえば，インパルス幅 $t=40$ psec (4.2節参照) の時間内に長さ200 mmの銅パイプ上の電荷はどのくらい移動できるであろうか．このように極めて短時間では仮りに**光速で動いたとしても，その値は $ct=12$ mmであり，これは銅パイプ全長の10分の1にも満たない長さである．そしてこの銅パイプから10～20 mmの距離にショートモノポール・アンテナが位置していることから，この空間を雰囲気（電界）が伝搬する時間も先のインパルス幅 t と比較し得る値になっている．**さらに，このショートモノポール・アンテナのエレメント長は5～25 mmであり，これもまた先の長さや（空間）距離に対し比較し得る値になっているのである．

この事実は，金属物体間におけるESD現象とこれに付随するEMI作用を“無限一媒質近似”および“無限一継続時間近似”で取り扱うことはできないことを示しており，今後より精度の高い実験と単極インパルス電磁界に対する新たな解析手法を必要とするであろう．

6. 結　　論

長さがミリメートルオーダのショートモノポール・アンテナを使用したことにより，

今までわからなかった間接ESDによるインパルス界の超広帯域性および空間における単極性が明らかとなった．

ショートモノポール・アンテナは超広帯域インパルス現象をとらえるのに最適な“プローブ”であり，ピコセカンドオーダの時間領域測定テクニックに必須の存在となるであろう．

参考文献　［1］ 本田昌實・川村雄克，"ESDの特徴と計算数に対する影響（その1)"，電子通信学会技術研究報告(環境電磁工学)，EMCJ 83-75 VOL. 83, No. 209, 1983 pp. 25～30.
［2］ M・ Honda "A New Threat-EMI Effect by Indirect ESD on Electronic Equipment"，IEEE Transactions, Industry Applications, VOL IA-25, Sept/Oct. 1989. pp. 939～944.
［3］ M. Honda and T. Kawamura, "EMI Characteristics of ESD in a Small Air Gap"，EOS/ESD Symposium Proceedings, EOS-6 1984, pp. 124～130.
［4］ M. Honda and Y. Ogura, " Electrostatic Spark Discharges"，EOS/ESD Symposium Proceedings, EOS-7, 1985, pp. 149～154.
［5］ 本田昌實，“ESDの特徴と計算機に対する影響（その4)”，電子通信学会技術研究報告(環境電磁工学)，EMCJ 86-26 VOL. 86, No. 107, 1986, pp. 17～22.
［6］ IEC Standard：Pub. 801-2 Draft 4 (1989)，EMC For Industrial Process Measurement and Control Equipment Part 2「Electrostatic Discharge Requirements」
［7］ M. Honda and Y. Nakamura, "Energy Dissipation in ESD and Its Distance Effects"，EOS/ESD Symposium Proceedings, EOS-9, 1987, pp. 96～103.
［8］ 本田昌實，“ESDの特徴と計算機に対する影響（その6)”，電子通信学会技術研究報告（環境電磁工学)，EMCJ 88-1 VOL. 88, No. 5, 1988, pp. 1～4.
［9］ M Honda, "Evaluation of System EMI Immunity Using Indirect ESD Testing"，EOS/ESD Symposium Proceedings, EOS-10, 1988, pp. 185～189.

執筆者紹介　本　田　昌　實（Masamitsu Honda）

昭和18年生，40年北海道大学　工教　電気科卒業，同年日本ユニシス(株)入社，大型計算機の保守に従事．その後，稼働環境，とくに計算機に与える電磁妨害，静電気障害対策の調査・研究に従事．システムプロダクト本部をへて，現在　人材開発本部　専門教育二部所属，EMI/ESD．

静電気研究懇談会会員，静電気学会会員，電子情報通信学会会員，IEEE会員，米国電子戦学会(AOC)会員，米国EOS/ESD協会会員．

光磁気ディスクの互換性阻害要因の究明

Interchangeability Test of the Magneto-Optical Disk Cartridges

大石完一

要　約　国際標準化作業が一段落した130 mm書換型の光磁気ディスクが一斉に登場し出した．しかし，本来この光ディスクは，媒体が交換できる大容量記憶装置であるが，1社の駆動装置には1種の媒体しか使えないのが現状となっている．互換性は，経済的にも，またユーザにとって万一の時のセカンドソースとしても，不利益を生じることから改善の声が高まってきている．しかし，光ディスクが新しいテクノロジに基づいた産業であることもあり，互換性阻害の原因が把握されていないのが現実である．

通商産業省工業技術院の委託を受けて，光ディスク標準化に関する調査研究を目的に設立されている「光ディスク標準化委員会」に互換性に関する作業部会の設立を提案し，互換性の現状把握とその阻害要因の究明を行った．

現在製品レベルにある，媒体17社，駆動装置8社の参加を得て，ISO規格の互換パラメータの測定，実機でのデータ読取りエラー率等をラウンドロビン測定法で行った．これらのデータ分析検討の結果，記録するためのレーザ・パワーの絶対値の捕え方に各社の差があり，この差が互換性を損ねている主原因となっていることを突止めた．

この対策として，標準駆動装置を設け，これとの相対比較をすることによって，問題解決が比較的簡単にできることを提案した．

Abstract　A round-robin test has been carried out by the Japanese Optical Disk Standardization Committee to clarify specific problems of the data interchangeability between media and drives based on the parameters defined by the ISO/IEC DIS 10089. A total of 17 media manufacturers and 8 drive manufacturers have participated in this program.

Consequently, the reference drive is required to calibrate and get the most appropriate write power on the media, and to keep the interchangeability among drives.

1. はじめに

高度情報化社会の一層の進展に伴って，情報そのものの量が増大するだけではなく，コードデータ・文書・画像等のイメージ情報，音声情報等が多様化し，これらが混在していく傾向にある．これに伴い，記憶装置の用途も大型システムのみならず，公共・企業・家庭・個人向けと多様化し，それぞれの用途にマッチした製品開発の要望が増大している．光ディスクは，これらの要望を満たし得る記憶装置として大きな発展と市場性が期待されている．すなわち，光ディスクは，面記録密度が磁気記録媒体に比べて1桁高い，数ミリの空隙をとって非接触記録・再生・消去ができる，長期にわたって高い信頼性が保てる，フロッピディスクのように容易に媒体交換できる，といった特徴があり，直径130 mmの媒体にGB（ギガバイト）に近い情報が記録できる．光ディスクは，130 mm追記形が国際標準化された[1]のに続いて，130 mm書換え可能形である光磁気方式も技術的内容の合意がとれ，最終的な編集作業を残して国際規格

(案) ISO/DIS 10089[2]が完成した．これに伴い，市場にはISO準拠をうたう製品が相次いで出荷され出した．

情報交換媒体である以上，情報の記録・再生を含めて媒体の互換性の確保がユーザ/メーカ双方にとって必要条件となる[3]．初期の内に，互換性確保の手を打たないと手遅れになりかねない．本来ISO規格は，互換性を確保するために設けたものである．光ディスクは，新しいテクノロジに基づいた製品で，互換性の確認が十分にできていない．光ディスクの規格は規格が先行した関係もあり，物理特性の絶対値で互換パラメータを規定しているので，これらの互換パラメータの物理特性が実際のデータの読み書きの信号特性と相関していることの確認が必要となる．また現実的には，1社の駆動装置には特定な1社の媒体しか使えないという状況にあり，普及を阻害している要因の一つにもなっている．

通商産業省工業技術院の委託を受けて，光ディスク標準化に関する調査研究を目的に設立されている「光ディスク標準化委員会」に互換性検討に関する作業部会の設立を提案し，筆者が主査となって，最初に130 mm追記形光ディスクの互換性に関する確認試験を実施した．その結果については，媒体自身の互換性を論じる以前に光学的測定技術の互換性の確立が重要課題であることを指摘した[3]~[5]．

次いで昨年度，130 mm書換え可能形である光磁気方式の光ディスクの互換性に関する確認試験をISO規定項目に従った測定および実機でのデータ記録再生との相関性の確認測定を行った．その結果，問題点は残るものの，ISO規格項目の互換性は実質的には満たされるようになった．しかし，ISO規定の互換パラメータと実機のデータエラー率との相関特性は，明確にならなかった．そこで，媒体と駆動装置との互換性を阻害している要因を明らかにするために，さらに両者のトラッキング特性，コントロールトラックに記載した記録条件と駆動装置が判断した値との偏差，物理特性と信号特性との相関等を実機上で測定調査した．その結果，媒体間，駆動装置間に記録するためのレーザパワーの絶対値の捕え方に差があり，その差が互換性を損ねている主原因となっていることを突き止めた．

本稿は，これら試験の概要および光磁気ディスクの互換性阻害の原因解析並びに対策について考察したものである．

2. 互換性試験の概要

ISO/DIS 10089で物理量の絶対値によって規定されている互換パラメータの測定，実機上での媒体の機械的互換性の測定およびデータ読取りエラー率の測定等，実機での基礎的な互換性試験，さらに互換性阻害要因を探る測定等をラウンドロビンテスト法[6][7]によって行った．ISO規定の互換パラメータの測定および実機による基礎的な互換性測定は，測定期間の関係があり，同一種類の媒体を測定者の数用意し並列に測定した．この場合，同種の媒体間の特性差による各測定結果の誤差を防ぐために，あらかじめ媒体ごとに媒体提供者による測定データを取付け，事後に補正できるようにした．互換性阻害要因を探る測定では，本来のラウンドロビン測定法である一つの媒体を全測定者に順次回して測定する方法を採った．

なお，これらの試験は，ISO規格準拠をうたって開発している製品のほぼ全数であ

る媒体17社，駆動装置8社の協力を得て実施した（表5参照）．

2.1 互換性パラメータの測定

ISO/DIS 10089規定項目のうち，表1に示す媒体の物理的・光学的特性の測定を行った．これらは，実際のデータ記録再生の信号レベルを左右する媒体固有の物理的特性であり，互換性を規定する上で重要な確認項目となる．また，1987年に行った追記形光ディスクの測定で，測定の再現性が問題となった項目でもある[3][4]．ただし，前回問題となったベースライン反射率や信号特性測定のバラツキの原因となった測定器の校正の問題[4]に関しては，仮の基準反射率校正板の配布，レーザスポット形状の校正基準を作成する等，事前に測定器の校正措置[5]をとって測定するよう改善した．

測定は，各項目ともに，ディスクのデータ記録領域最内周・中周・最外周の3か所で行い，ディスクの記録場所による差異も見られるようにした．

2.2 実機による基礎的な互換性測定

実機による基礎的な測定は，表2に示すように，各種媒体の駆動装置への装着性，データ記録再生の基本となる駆動装置との機械的動作の互換性およびデータ読取りエラー率（ビット誤り訂正前の）等の測定を行った．エラー率は，ディスクのデータ記

表1 互換パラメータの測定項目と規定

Table 1 Principal parameters and specifications for interchangeability test for physical parameters

測定項目	定義	規定
ベースライン反射率	I_0	10～34%
トラッキング関連信号特性		
・トラッククロス信号	$(I_1+I_2)_{MAX}/I_0$	0.70～1.00
・プッシュプル信号	$(\|I_1-I_2\|)/I_0$	0.40～0.65
・トラッククロス信号変調度	$[(I_1+I_2)_{MAX}-(I_1+I_2)_{min}]/I_0$	0.20～0.60
マーク関連信号		
・セクタマーク信号	$\|I_{SM}\|/I_0$	≧0.50
・VFO信号	$\|I_{VFO}\|/I_0$	≧0.25
狭帯域信号対雑音比 C/N	CW	≧45 dB
消去効果	CE/CW	≦−40 dB
クロストーク	隣接両トラックへの漏洩信号レベル	≦−26 dB

表2 実機による基礎的な互換性測定項目

Table 2 Principal parameters for interchangeability test for media with drives

①動作の互換基本特性	
・ドライブにカートリッジが挿入できたか．	Yes/No
・ディスクが回転したか．	Yes/No
・サーボトラックのポジショニングとフォーカスロックができたか．	Yes/No
・コントロールトラックのPEPが読めたか．	Yes/No
・ヘッダ一部が読めたか(シークが正確に行えたか)．	Yes/No
②データ読取りエラー率の測定(ディスク3か所，各1000トラック測定)	
・ID部のバイトエラー率	
・データ領域にドライブ固有記録条件で記録再生した時のバイトエラー率 (4192_Hパターン)	
・データ領域に媒体固有の記録条件で記録再生した時のバイトエラー率 (4192_Hパターン)	

録領域の最内周・中周・最外周の3か所各々1000トラックの測定から算出し，ディスク上の汚れや欠陥箇所による影響を除去するよう考慮した．記録データはディスク上で最悪ビットパターン配列となる$(4192)_H$を用いた．

2.3 互換性阻害要因を探る測定

各種媒体間または各種駆動装置相互間で互換性を損ねている実際の要因を探る目的から，トラッキング特性および信号特性について，①駆動装置を固定した時の各種媒体間の差異，②媒体を固定した時の各種駆動装置間での差異の測定を行った．

駆動装置を固定した時の測定については，特定の駆動装置一台を選定し，この機器での全種の媒体の特性比較を行った．

トラッキング特性については，媒体の溝構造と駆動装置のトラッキング追従性との関連を調べるために，5000回のランダムアクセスに要した時間を各種媒体/駆動装置間で比較してトラッキングの安定性の評価を行った．

信号特性については，記録パワーを1 mW刻みで変化（5～10 mWの範囲で）させたとき，次の各特性の変化度合いを測定し，記録パワーの依存性およびデータ読取りエラー率と物理的信号特性との相関を調べた．

①バイトエラー率，②C/N，③信号振幅，④分解能，⑤消去率

特定の駆動装置は，2.2節のバイトエラー率測定で，バイトエラー率が10^{-4}以下となった媒体の数が多くあり，かつ媒体のコントロールトラックに対する制御機能のある装置1台を選定した．

次いで，特定の駆動装置で測定した媒体の中から感度の異なる5種の媒体を選び，これを全駆動装置に回覧して機器間の差異，特定の駆動装置で記録した信号を他の駆動装置で消去した時の消去率等，機器相互間の互換性について測定した．

3. 互換パラメータの測定結果と評価

互換パラメータの代表的な測定結果を図1～3に示す．いずれも規格値を逸脱しているものが若干見受けられる．これらは，“媒体D”・“媒体K”・“測定者4”等のように，特定の媒体あるいは測定者またはその組み合わせによって発生している．これらは，彼等が使用している測定器の基準値に異常があったものと推定される．すなわち，異常媒体については，彼が測定した申告値は，図1に見られるように正常な値を示していることから伺える．同様に，異常測定者についても，図2(b)に見られるように，彼が測定した媒体は他の測定者に比べて全体に低い値を示している．

これらの異常データを除外すれば，マクロ的に見て測定者間・媒体相互間ともに，かなり揃ってきており，追記形のラウンドロビンテストで「媒体自身の互換性を論じる以前に光学的測定技術の互換性確立が重要」と指摘し，その改善を計ったこと[3]~[5]の効果が現れたものと思う．

しかし，大勢的には規格範囲に収まっているものの，全体の測定のバラツキは，まだ±10%を超えている．とくにベースライン反射率については，事前に反射率の基準校正板を回し，測定器を校正する処置を採ったにもかかわらず，図1に示すように，まだ±10%以上のバラツキが生じている．ベースライン反射率は，信号特性の分母を形成する駆動装置との互換性の基本となるもので，厳密な測定値が求められる．今後

の基準反射板の整備，測定器校正方法の標準化等によって測定のバラツキを減少させる対策が今後の課題となる．

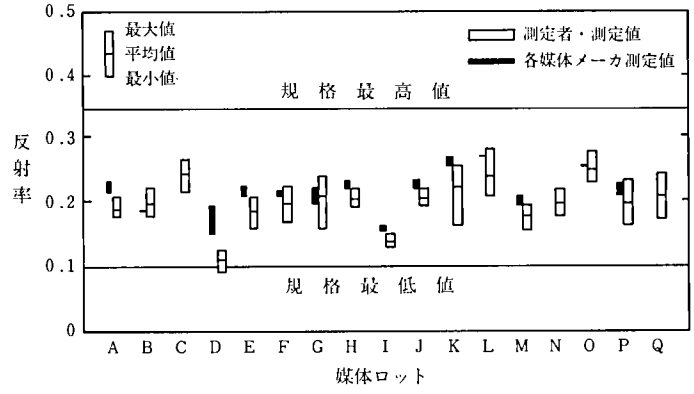


図 1　ベースライン反射率測定の媒体のばらつき

Fig. 1　Baseline reflectance

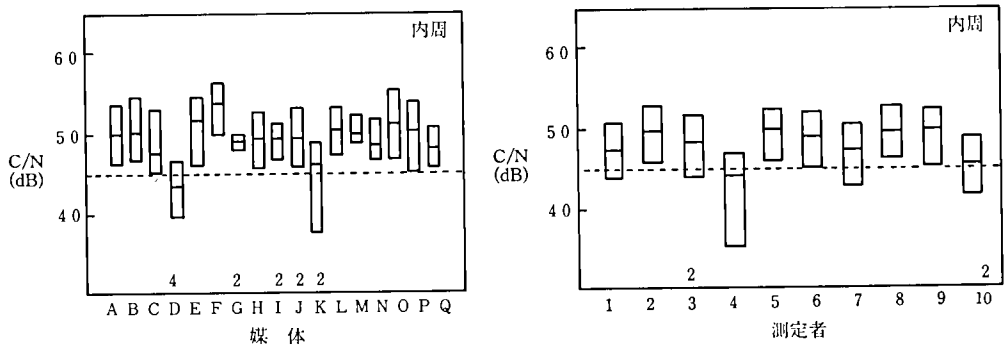


図 2　C/N 測定の媒体間および測定者間のばらつき

Fig. 2　C/N ratio

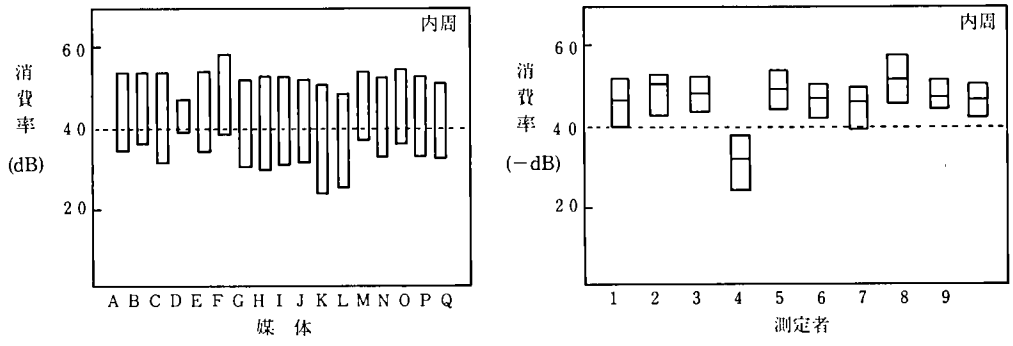


図 3　消去率測定の媒体間および測定者間のばらつき

Fig. 3　Ease of erasure

表 3　実機での基本動作測定結果

Table 3　Test result of mechanical interchangeability for media with drives

○：可能　×：不可　－：不可または未測定

(a) カートリッジの挿入

媒体種類	11	12	13	14	15
A	○	○	○	○	○
B	○	○	○	○	○
C	○	○	○	○	○
D	○	○	×	○	○
E	○	○	○	○	○
F	○	○	○	○	○
G	○	○	○	○	○
H	○	○	○	○	○
I	○	○	○	○	○
J	○	○	○	○	○
K	○	○	○	○	○
L	○	○	○	○	○
M	○	○	○	○	○
N	○	○	○	○	○
O	○	○	○	○	○
P	○	○	○	○	○
Q	○	○	○	○	○

(b) ディスクの回転

媒体種類	11	12	13	14	15
A	○	○	○	○	○
B	○	○	○	○	○
C	○	○	○	○	○
D	○	○	－	○	○
E	○	○	○	○	○
F	○	○	○	○	○
G	○	○	○	○	○
H	○	○	○	○	○
I	○	○	○	○	○
J	○	○	○	○	○
K	○	○	○	○	○
L	○	○	○	○	○
M	○	○	○	○	○
N	○	○	○	○	○
O	○	○	○	○	○
P	○	○	○	○	○
Q	○	○	○	○	○

(c) サーボの確保

媒体種類	11	12	13	14	15
A	○	○	○	○	○
B	○	○	○	○	○
C	○	○	○	○	○
D	×	○	－	○	○
E	○	○	○	○	○
F	○	○	○	○	○
G	○	○	○	○	○
H	○	○	○	○	○
I	○	○	○	○	○
J	○	○	○	○	○
K	○	○	○	○	○
L	○	○	○	○	○
M	○	○	○	○	○
N	○	○	○	○	○
O	○	○	○	○	○
P	○	○	○	○	○
Q	○	○	○	○	○

(d) SFP(コントロールトラック)の読み出し

媒体種類	11	12	13	14	15
A	－	－	○	○	○
B	－	－	○	○	○
C	－	－	○	○	○
D	－	－	－	○	×
E	－	－	○	○	○
F	－	－	○	○	○
G	－	－	○	○	○
H	－	－	○	○	○
I	－	－	○	○	○
J	－	－	○	○	○
K	－	－	○	○	○
L	－	－	○	○	○
M	－	－	○	○	○
N	－	－	○	○	○
O	－	－	○	○	○
P	－	－	○	○	○
Q	－	－	○	○	○

(e) ヘッダ部の読み出し

媒体種類	11	12	13	14	15
A	○	○	○	○	○
B	○	○	○	○	○
C	○	○	○	×	○
D		○		○	○
E	○	○	○	○	○
F	○	○	○	×	○
G	○	○	○	○	○
H	○	○	○	○	○
I	○	○	○	○	○
J	○	○	○	○	○
K	○	○	○	○	○
L	○	○	○	○	○
M	○	○	○	○	○
N	○	○	○	○	○
O	○	○	○	×	○
P	○	○	○	○	○
Q	○	○	○	○	○

4.　実機による基礎的な互換性測定結果と評価

媒体と実機との基本的動作特性の互換測定結果では，表3に示すように，特定な媒体(D)が大半の駆動装置で受入れられなかったケースを除いて機械的互換はとれている．媒体(D)は，ベースライン反射率が規格下限値を割込んでいたために大半の機器でフォーカス・ロックができなかったことによるものと推察する．

また，ディスク固有の最適記録条件等の制御信号が記録してあるコントロール・トラックの制御情報部(SFP)の読取りについては，これらの機能を持たない装置が2機種あった(装置11，12)．これは，この機能を装備することによって高価になることを嫌った措置と思えるが，現状では媒体の種類（特性に対する）が多く，ISOでもSFP情報の記録を義務付けている．ISO規格準拠製品とうたって販売している以上，多くの媒体との互換性をとる必要があり，改善を要する．

データ領域の読取りエラー率の測定結果では，図4に示すように，駆動装置の特性の違いがハッキリ現れている．ISO規定の互換パラメータとの相関は，明確になっていないが，ベースライン反射率・C/N・消去・クロストークとの相関を見てみると，図4で，バイト誤り率10^{-12}に訂正のきく限界とされている，ディスク自体のバイトエラー率が10^{-4}以上となった媒体については，表4に示すように，例外もあるが，主要なディスクの物理的互換パラメータのいずれかが，規定下限値を下回っている．しかし，互換パラメータの特性と実際のデータ記録再生特性との互換性確認ができるデータとはなっていない．

5.　互換性阻害要因を探る測定の結果と評価

トラッキングの安定性については，一定のランダムアクセス回数実施時の所要時間

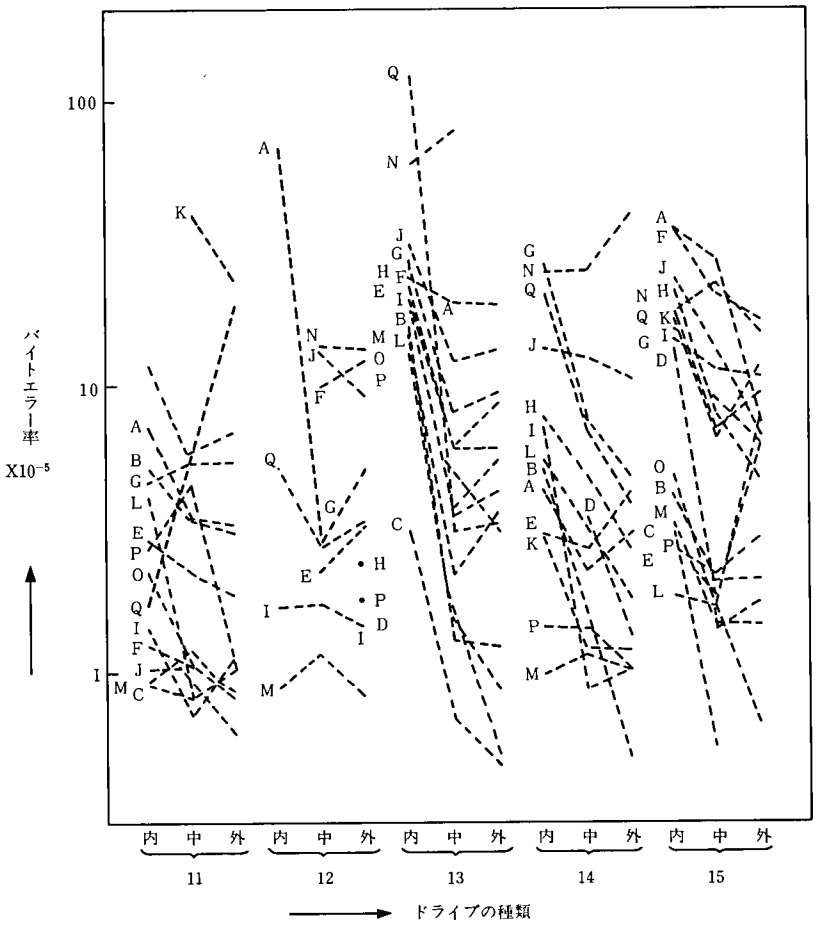


図 4 駆動装置別データ読取りエラー率の比較

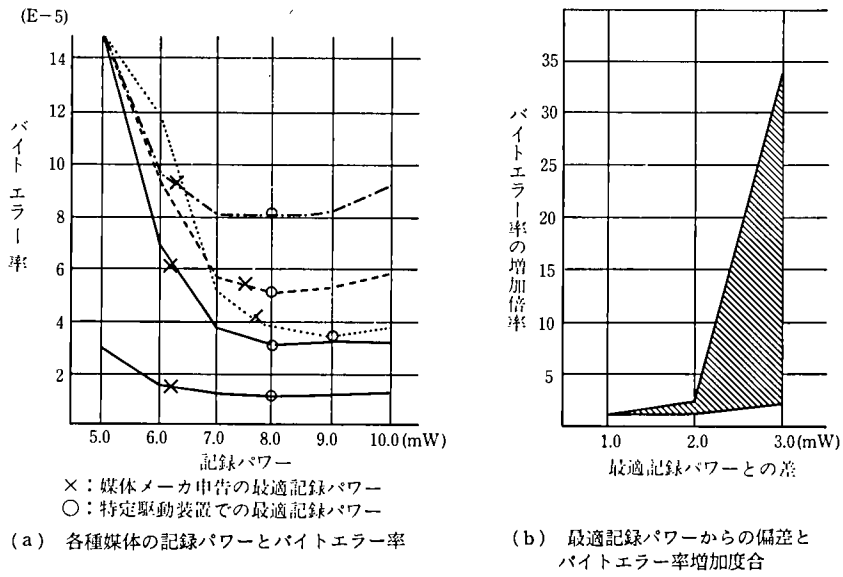
Fig. 4 Read error rate of interchangeability test for media with drives

表 4 互換パラメータとデータ読取りエラー率との相関

Table 4 Relationship between physical parameters and read error rate

	媒体															
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
ベースライン反射率				X								X		X			X
C/N, 3.7 MHz 再生				X								X					X
C/N, 3.7 MHz 記録再生				X							X	X					
消去率				X													
クロストーク				X			X				X			X			

□：バイトエラー率が10^{-4}を超えた媒体，X：規格下限値を割込んだ媒体



（a）　各種媒体の記録パワーとバイトエラー率

（b）　最適記録パワーからの偏差とバイトエラー率増加度合

図 5　記録パワーとデータ読取りエラー率との関係

Fig. 5　Relationship between write power and read error rate

の比較を行った結果，媒体間/駆動装置間共に 5%以内にあった．媒体の溝形状には，大別して，丸・三角・四角等があり，各社異なっているようであるが，グルーブ形状の違い等による各種駆動装置のトラッキング動作への影響はなく，トラッキング特性が互換性阻害要因とはなっていなかった．

信号特性は，記録パワーに大きく左右されている．すなわち，図 5 (a) に示すように，データ読取りエラー率は，記録する時のレーザパワーの値に大きく影響されている．ここで問題は，各媒体メーカが示している最適記録条件（図中の“×”印の値，媒体のコントロールトラックに記載されている）と特定の駆動装置が示した媒体ごとの最適記録条件（図中の“○”印の値）との間には，最大 2 mW 前後の差が生じている．図 5 (b) に示すように，記録パワーが 2 mW ずれると，データ読取りエラー率が 2〜3 倍以上の差が生じ，場合によってはビット誤り訂正の効く範囲[8]を逸脱することになる．記録レーザパワーの標準化が互換性をとる上で重要課題となる．

データ読取りエラー率と ISO/DIS 10089 規定の物理的信号特性との関連性については，駆動装置を固定して，その装置での媒体最適記録条件で評価すると，図 6〜9 に示すように，よく相関がとれている．前章の測定で両者の相関が明確とならなかった原因は，記録条件が一致していなかったためと判明した．

「1 社の駆動装置には特定の 1 社の媒体しか使えない」というのは，多少オーバとしても，互換性が損なわれている主因が記録パワーにあることが確認された．この原因は，次の事柄が考えられる．

1)　記録パワーの絶対値の捕え方に差が生じている．

2)　最適記録パワーが媒体ごとに異なるのに多くの駆動装置の書込みパワーが固定記録となっている．

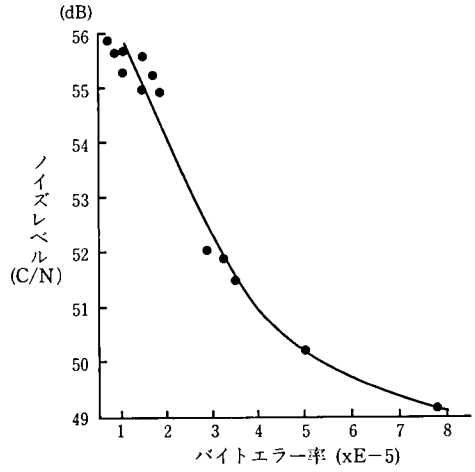


図 6　データ読取りエラー率と C/N

Fig. 6　Relationship between read error rate and C/N ratio

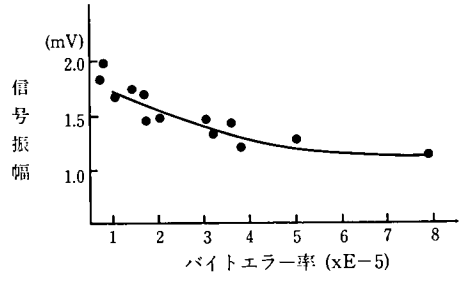


図 7　データ読取りエラー率と信号振幅

Fig. 7　Relationship between read error rate and signal amplitude

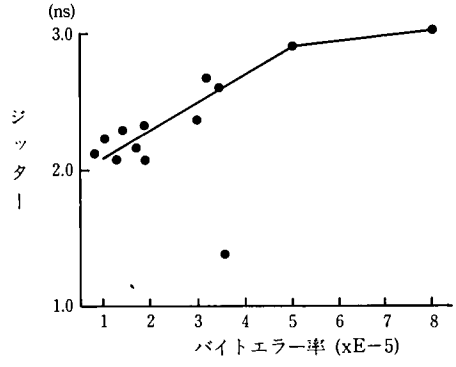


図 8　データ読取りエラー率とジッター

Fig. 8　Relationship between read error rate and jitters

3)　コントロールトラック SFP に記載した 1800 rpm での最適記録条件に対して，駆動装置の回転数が異なる場合，駆動装置に合った最適記録条件に換算する換算の仕方に差が生じている．

記録パワーは，レーザパワーの密度によって決まるが，このパワー密度は媒体の記

録膜面上に結ばれるレーザスポット径やスポット形状の違い，レーザーパワーの測定誤差，測定方法の違いによってバラツク可能性がある[8]．媒体メーカ各社間での最適記録パワーの絶対値の基準/評価方法等に差が生じているものと考える．

対策として次の方策が考えられる．

① 物理的測定法およびレーザパワーの基準値の決め方の標準化

② 標準駆動装置を設け，この機器によって媒体の最適記録パワーを校正する．

①の方法は，測定器または駆動装置の光学的ヘッドであるレンズのフレア，波長と対物レンズの開口数比(λ/NA)，波面収差等，変数が多く絶対値規定がむずかしい．標準ヘッドを決めたとしても，その取付けチルト，媒体と装置との取付け状態に左右される性質があり困難を伴う．

②の標準駆動装置による校正（相対比較）方法は，基準となる駆動装置1台を決めれば良いので比較的簡単な解決方法となる．

ISO/DIS 10089の規格では，媒体にコントロールトラックを設け，媒体ごとの最適記録パワー等の記録条件を記録するように規定し，駆動装置側で記録条件を合わせ込む方式が採られている．したがって，ISO規格準拠をうたって販売している駆動装置には，記録条件制御機能が現状では必要条件となる．しかし，最適記録パワー条件を特定の駆動装置で見ると，図5（a）に見られるように，大半の媒体は8 mWにあり，媒体間の差も前後1 mWの範囲にあるので統一することは可能と思われる．統一されれば，駆動装置の記録条件制御機構や媒体のコントロールトラック機構を取外すことが可能となり，両者の価格低減化に大きく貢献できる．

一方，規格では，媒体のコントロールトラックSFPに，1800 rpmでの最適記録条件を記載するよう規定しているが，他の回転数での最適記録条件の換算方法の基準化がされていない．現在，市販駆動装置の回転数は，1800 rpmよりも2400 rpmの方が多く，将来もっと高速な駆動装置が出現することが予想される．したがって，互換性をとるためには，回転数に合った最適記録条件の換算方法やSFPへの記載方法を標準化しておく必要がある．

6. 互換性対策

光磁気ディスクの互換性を阻害している原因については，前章までに明らかにした通りである．互換性を損なっている原因が判明しても，最終的には各メーカが対応しなければ解決したことにならない．媒体/装置メーカは，互換性をとれるようにしたい意向を本質的に持っている．しかし実行論になると，各社のノウハウ，市場に出ている等から，特性を統一させることはむずかしい問題が多々生じてくる．

媒体/装置メーカ各社の意向を集約すると，標準駆動装置を設定し，校正できる機関の設立を望んでいる．標準駆動装置を設けることによって，標準駆動装置で測定した媒体を介して自社の媒体/駆動装置の諸互換特性の標準との偏差を知ることができるので，互換性が確保できる範囲内に自社製品をコントロールできる．

一方，標準駆動装置の設置・測定サービス等の運営は，中立機関で行う必要がある．そこで互換性確保の一環として，通商産業省工業技術院に国際的標準のサービス機関の設立を要請した．通商産業省工業技術院としても，日本が国際的に貢献しなければ

ならないという立場と相応し，早期に設立することで前向きな検討が進められている．

標準センター（仮称）の設立までに，標準駆動装置の選定，標準特性となるアイテムの選定，サービス方法，標準駆動装置の標準アイテムの維持管理運営方法等を決める必要があり，今後の検討課題となる．基本的には今回の試験データが生きるものと確信する．

その他，前記のようにコントロールトラックSFPへの最適記録条件の記載方法および回転数が異なる場合の最適記録条件の換算方法の標準化等が今後の課題となる．

7. ま と め

光磁気ディスクの互換性の現状についてラウンドロビン測定を通して調査検討した．

1) ISO/DIS 10089規定の互換パラメータの測定結果
   - ① 特定の媒体，測定者を除けばすべての媒体は規格範囲内に収まっている．
   - ② 規格を逸脱した媒体は，媒体自身の問題以前に使用した測定器の校正に問題がある．
   - ③ ±10%を超える測定のばらつきがあり，機器の校正方法の改善が必要である．

2) 互換性を損ねている原因の調査結果
   - ① 記録レーザパワーが最適値から2 mWずれると，データ読取りエラー率は2〜3倍以上増加する．
   - ② 媒体/駆動装置相互の最適記録条件となるレーザパワーの絶対値が各社間で2 mW前後あり，これが互換性を損ねている主原因となっていることが判明した．
   - ③ 記録レーザパワーの基準を採れば，ISO規定の互換パラメータと実機でのデータ読取りエラー率とは相関関係にあることがわかった．
   - ④ 媒体回転数が異なった時の最適記録条件の算出方法に差がある．

3) 対策の検討
   - ① 標準駆動装置による相対比較法が，比較的簡単な解決方法であることを示唆した．
   - ② 媒体のコントロールトラックSFPへの最適記録条件の記載方法の標準化が必要．
   - ③ 回転数が異なる場合の最適記録条件の換算の標準化が必要．

8. お わ り に

このプロジェクトは，石井泰弘（三洋電機），石橋稔（ヘキスト），沖野芳弘（松下電器），小林政信（沖電気），小川紘一（富士通），山田文明（日本IBM）の諸氏が参加して行われた．また，測定に当たっては，表5の企業の絶大な協力があった．諸氏に感謝すると共に，情報処理用大容量記憶媒体としての光ディスク産業の発展に，この結果が寄与できれば幸いである．

表5　光磁気ディスク互換性試験参画企業
Table 5　Attendance for the round-robin test

媒体提供	ドライブによる測定	互換パラメータ測定
TDK	三菱電機	出光石油化学
住友化学工業	松下電器産業	オリンパス光学
住友スリーエム	シャープ	京セラ
三菱樹脂	日立製作所	東ソ
旭化成	東芝	ナカミチ
出光石油化学	日本電気	ニコン
東芝	オリンパス光学	三菱化成
シャープ	ソニー	三菱樹脂
三井石油化学		三菱電機
富士電機総研		NTT
ヘキストジャパン		
ダイセル化学		
松下電器産業		
三菱化成		
クラレ		
セイコウエプソン		
PDO		

参考文献　[1] ISO 9171-1990 Information technogy-130mm write once optical disk cartridges for information interchange.
[2] ISO/DIS 10089 Information technogy-130mm rewritable optical disk cartridges for information interchange.
[3] 光ディスク標準化に関する調査研究III，1988，3月，光産業技術振興協会.
[4] 大石，追記形光ディスクの互換性評価，UNISYS 技報第24号 Vol.9-4)，1990 Feb.
[5] 大石，山田，Interchangeability test of optical disk cartridges, Sept. 26, 1989 International Symposium on Optical Memory 1989, Technical digest 26D-1.
[6] Japanese information paper for the round robin measurements for cartridge tape on ISO 9661, ISO/JTC1/SC11 N 940, 5-1989.
[7] 光ディスク標準化に関する調査研究 V，1990年3月，光産業技術振興協会.

執筆者紹介　大　石　完　一 (Kan-ichi Ohishi)
昭和30年日本大学第1工学部電気工学科卒業．昭和33年日本ユニシス(株)入社．主として周辺機器の信頼性評価等に従事．また，ISO/IEC JTC 1/SC 11 委員長，同 SC 23 委員および磁気テープ，フレキシブルディスク，光ディスク JIS 化等の標準化専門委員会の委員長，主査等として標準化活動に従事．現在，システムプロダクト本部ハードウェアプロダクト4部に所属．フロッピーディスクのおはなし（日本規格協会），ハードウェア入門（共立出版）等の著書がある．電子情報通信学会会員，情報処理学会会員．

戦略情報システム構築における人間軸の世界

The Human Domain in Strategic Information Systems Implementation

小　坂　　武

要　約　戦略情報システム（SIS）は市場に新しい価値を提供するために構築される．これはビジネスの再構築を必要とするだけでなく，タスク，役割，権限を変える組織変革を伴う．このような経営革新は組織の深い部分におよび，人と組織までも変革の対象とする．

SISを構築する方法論が各社で開発され提供されている．これらの方法論は戦略立案からシステム化計画までを対象とした技術的側面（技術軸）を中心に扱う．これらはいずれも構築上，「人と組織」（人間軸）への対応が重要であると言及しながらも，それを暗黙知の世界として具体化していない．

本稿はSISを構築する上で必要となるこの人間軸の世界を取り上げ，シャインのUCR（解凍・移動・再凍結）モデルとベイトソンの人間の認識モデルを基にして，新しい情報技術を導入する上で必要となる変革のための方法論を構築する．

Abstract　Strategic Information Systems (SIS) are constructed to provide a new value to a market. They require the restructuring of business with organizational changes in tasks, roles and authorities. This management innovation penetrates the depth of an organization and makes us perceive that an organization and its members are the target of change.

Systems consulting firms have developed to provide methodologies for SIS planning. These methodologies are mainly of technical domain (technical axis) spreading from strategy formulation to systems planning. They have often referred to the importance of consideration for an organization and its members (human axis) in SIS implementation but left it in the world of the tacit dimension without any concrete results.

This paper discusses the domain of human axis in SIS implementation and formulates a methodology required when a new information technology is introduced, using Schein's UCR (Unfreezing, Changing, Refreezing) model and Bateson's human perception model.

1. はじめに

戦略情報システム（SIS）の構築のためには，従来の情報システム計画（情報システム構築のための中長期計画）にない新しい考え方が必要である．価値活動（ポーター，1985）の連鎖に情報技術を適用すること*がSISの構築である以上，既存の組織を構成している業務の体系を新しい価値活動の連鎖に変革しなければならない．しかも，それ自体が新しい経営戦略の出発点となる．同時に，このような経営の革新は企業組織の深い部分に及び，人と組織の価値観までも変革の対象となる．すなわち新しい情報システム計画においては，人間的な側面と技術的な側面を持つ企業活動の両面から統合的にとらえなければならない．

* 価値活動用の合理化をするのが従来の情報システムであったのに対し，価値活動群を融合化・統合化するのがSISの基本である．前者の例は生産管理システムであり，後者の例がCIMである．

そのために，本稿では，人と組織の変革を「人間軸」とし，価値活動の連鎖の再構成とそれによる新しい経営戦略の立案を「技術軸」として考えていくことにする．組織が人間的活動に依存しないのならば，すなわち，組織が社会性をもたず合理性だけで成り立っているのであれば，情報システム計画活動は技術軸だけですることになる．しかし，組織が人間的側面を持つ以上，合理性だけの計画は組織の中で実行されない．

本稿においては人間軸の検討を行う．SIS が企業革新を意味するということは，その構築が企業の人と組織にとって極めて大きなインパクトを与えるということであり，人間軸の検討は極めて重要である．しかも従来の情報システム計画では，人間軸についての考察がまったくと言ってよいほどなされていなかった．

2. 人間軸で扱う問題

2.1 組織の社会的慣性

SIS を構築するための人間軸での活動目的は，このシステムが定着して効果を発揮するように，組織のメンバーが新しい考え方と価値観で行動するようにすることである．もともと組織には慣性力があり，新しい規則や仕組みを作ってもなかなか使われず，従来のままの活動がいぜんとして続くものである．Keen (1981) は情報システムの導入におけるこの問題に触れ，どのように技術的に努力しても，何も組織に起こらないことを「社会的慣性」と呼んだ．

情報システム計画で対処しなければならない社会的慣性の最も大きなものは，情報システム計画を外部のコンサルタントに依頼し，その活動に内部の人間がタッチしなかった時に発生する．外部コンサルタントは，内部の人間の参加がないために，限られた情報を基礎に最大限の純技術的な方法を適用する．そして，合理的でエレガントな計画を作り，報告書にまとめる．その報告書を受け取った情報システム担当者がいざ社内で実行に移そうとすると，組織内からさまざまな抵抗を受け，前へ進めなくなってしまう．これは，報告書をそのまま書棚にしまい込んだ場合と同等の結果になる．すなわち，コンサルテーション費用の発生以外は実質的に何も組織に起こらない．

この問題は，SIS の構築の場合に極めて重大になる．従来の組織の持つ社会的慣性をそのままにして SIS を構築しようとすると，「何も起こらなかった」ではすまされない大きな混乱が組織の中に発生する．つまり，従来の組織の慣性を支えてきた仕事の体系は，新しい価値活動の連鎖を作るという名目の下に切り裂かれ，まとまりを失った業務が旧い価値観のまま遂行されて，収拾のつかない混乱に至る．そして，新しい価値活動の連鎖がどのような価値の創造になるかがまったく受け入れられていないために，新しい経営戦略としての統合性が発揮されず，以前よりもはるかにレベルの低い経営活動となる．これは，莫大なムダ投資による重度の後遺症である．

SIS の構築における人間軸の役割は，従来からの組織の慣性を変換して，新しいシステムに必要な新しい組織の慣性を作り出すことである．仮りに，外部コンサルタントに計画を依頼して企業革新が可能になるのであれば，すでにどの企業でも革新に成功していなくてはいけないはずである．組織の深い部分に関わる人間軸での変革活動を外部コンサルタントだけで行っていくのは，不可能と言わざるをえない．企業内部の人間との協調活動が不可欠と言えよう．

2.2 抵 抗 の 型

人間軸で扱わねばならない組織の社会的慣性は，現実には，新しい情報システムを構築しようとする時の「変化に対する抵抗」という形をとって組織内に現れる．変化を否定しようとするさまざまな抵抗活動が現れたり，あるいはまったく変化を無視した従来通りの活動が続いたりする．また，もっともらしいタテマエ論を展開して変化を骨抜きにすることも行われる．

Markus (1983) は，情報システムの計画や開発の活動に関連して発生する抵抗には三つの型があることを示している．それは原因にもとづいて分類したものである．

第1の抵抗の型は，個人が原因となっている抵抗である．ある種の性格や価値観を持っている人は，導入される変化がどのようなものであれ，その個人にとっては受入れられないものとして抵抗を示す．たとえば，人事情報システムの開発を提案すると，社内の人材に精通しているある人事スタッフから抵抗を受ける．その抵抗は，彼個人の持っている人事情報が，情報システムの導入によって彼固有のものでなくなることと関係している．この種の抵抗は，抵抗する個人を変えたり，あるいは逆に情報システムの設計にその人の参加を求めることで対応が可能となる．このような抵抗は，社会的慣性の現れよりも，むしろ個人的慣性の現れの色彩が強い．

第2の抵抗の型は，導入するシステムそのものが技術的に使いにくいために，その使用に抵抗するというものである．システムの使いにくさは，いろいろな技術的問題から発生する．プログラムに欠陥のあるシステム，人間工学的に設計されていないシステム，ユーザ・フレンドリでないシステム等である．いずれにしても，新しいシステムを与えられた人は，自分なりの慣れたやり方でなく，そのシステムの要求する使いにくいやり方をしなければならず，抵抗する．この抵抗も，本質的には技術的な問題からの抵抗であり，社会的慣性の現れと言うべきではないであろう．

第3の抵抗の型は，従来の組織を基礎にする行動様式と新しい情報システムによって引き起こされる活動のし方の相互作用から発生する．つまり，人の働き方とシステムの特性との間の相互作用のために，従来の組織における社会的関係が変化することから抵抗が発生する．これこそが社会的慣性の現れとしても最も大きな抵抗となる．たとえば，分権化した事業部制の組織構造においてデータを中央統制する情報システムを導入すると，事業部間の業績競争の基礎をなす数字が，いつでも，誰にでも見えるようになってしまうために，抵抗を受ける．あるいは，企業と顧客との間にオンライン情報ネットワークを導入すると，顧客に接する重要部門であった営業部門と他の部門の間の力関係が変わるため，権力を失う人からは抵抗を受け，新たに権力を獲得する人には受け入れられる．

この種の抵抗が現れるのは，システムが使用される社会的状況とシステムの技術的設計面との相互作用から組織内に新たな社会関係が発生するからであり，政治闘争にまで発展する場合がある．この問題は，個人を変えたり技術的な問題を解決するだけでは解消することにならない問題であり，従来組織での社会的慣性が新しい情報システムに対する抵抗として現れたものである．

3. 「意味の体系」としての組織の文化と政治

3.1 組織文化が社会的慣性を与える

新しい情報システムを構築しようとする時に，従来組織の持つ社会的慣性が変化への抵抗として現れてくる問題は，情報システムに限らず何らかの新しい技術が組織内に導入されると必ず発生する問題である．このような社会的慣性は，組織内の行動様式や考え方がある一定のパターンで安定的に共有されているために，変化を受けた時にその安定を保とうとして現れる．組織が組織として安定的な活動を保っているのは，その組織に独特の「ものの見方」や「仕事のやり方」が維持されているからであり，それが組織に共有されている思考や行動様式となっている．これが「組織の文化」であり，組織内の人々の行動や考え方に目に見えない形で影響を与えている．ちょうど，未開部族の文化が部族内のメンバの行動規範や価値観の基礎となるようなトーテムやタブーを持っているのと同じである．

組織が安定的であればあるほど，その安定を維持する強力な組織の文化が存在している．そこに新しい情報システムが導入されると，組織の文化が安定を維持しようとするために社会的慣性となり，変化に対する大きな抵抗力となって出現する．

しかし，組織文化を組織メンバの行動や価値観を規定するものとして想定しても，組織文化そのものが単独に存在しているわけではない．組織文化は必ず人々の行動や価値観という形を通して現れてくるものであり，文化だけが人を離れて存在するのではない．つまり，組織文化が目に見えないものとして人々を律すると言うことは，組織内のさまざまなものが複合的に影響し合った結果，人々の心に共有される「意味の体系」としての共通認識となっているということである．

3.2 組織文化の複合的特徴

SIS の構築が従来の組織の社会的慣性を変革することである以上，それを導入しようとしている組織自体がどのような文化を持っているかを理解しなければならない．しかも組織文化が，組織のさまざまな要因を人々が複合的に認識した結果としての共有化された「意味の体系」であるため，その理解は，組織の要因を複合的でシステム的な関連のあるものとして捉えることから始めなければならない．

この点の理解を促進してくれる考え方が，組織の要因として「タスク」，「技術」，「構造」，そして「人間」をとりあげる考え方である（図 1）．

すべての組織は一つのタスク（課題・課業），あるいは一連のタスク，を達成するた

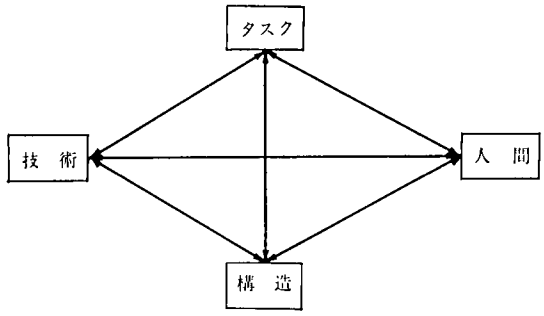


図 1　組織の要因

Fig. 1　Factors of organization

めに存在しており，その流れの中で相互作用を持つ人間がその組織を構成している．さらに，そのタスクを遂行するのに必要な技術と，人とタスクの関係を規定する組織の構造からなっている（バーグ，1987）．

組織文化の理解にとって重要な点は，これら四つの組織の要因が相互に関連しており，互に影響，調整し合っているということである．一つの要因の変化が他の複数の要因の変化の原因となると同時に，その変化が2次的原因となって他の要因に逆に変化を与えるという双方向的で複雑なシステム関係にある．この複合的な関係を，図1ではダイヤモンドの形で表わしている．したがって，組織の文化とは，これらの要因の双方向的な影響関係を通じて人間が認知する「行動のし方」であり，「価値観」ということになる．

SISの構築とは，すなわち，組織の構成要素の大きな部分である「技術」を変化させることである．本来ならば，技術の変化は他の組織の構成要素の変化と密接な相互関係を持たなければならない．新しい技術が導入されるための新しいタスクが設定されるべきであろうし，そのための組織構造や人の育成・配置が，それら要因間の新しい影響関係のもとに形成されるべきであろう．人の行動様式や価値観という観点からすると，そのような要因間の複合的で双方向的な新しい関係を人々が認識して初めて新しい行動様式や価値観が形成され，新しい組織文化が作られることになる．

しかしただ単に情報システムのみが導入され，しかも他の要素がそれに関連する相互関係から意味のある変化を起こさないとすると，新しい組織文化は形成されない．つまり，組織メンバの心の中にある既存の「意味の体系」にとっては，新しい情報システムは何ら本質的な変化ではなく，安定を乱す「異物」としてしか認識されない．このような場合，一時的に小さな変化はあるものの，全体としては何ら変わらないという組織のホメオスタシス（恒常性）行動が現れることになる．異物を排除しようとする拒否反応が起こるのである．それが社会的慣性であり，変化への抵抗と呼ばれるものとなる．

3.3 組織の政治的側面

情報システムの開発は，組織の文化と密接な関連を持つと同時に，組織の政治的なプロセスとも重要な関係を持っている（Keen, 1981）．ここで言う政治的プロセスとは，新しい情報システムを構築するかどうかの意思決定における政治的プロセスや，新しい情報システムの稼働によって引き起こされる権力関係の変化による政治的プロセスである．そこには，組織内の価値のバランスの変化や利害の変更による人々の力学的な動きがあり，組織メンバの持つ「意味の体系」の変化にともなう力関係の変更が起きる．組織におけるこのような政治的側面からも変化への抵抗が発生し，組織文化から発生する抵抗と同様に，SISの構築にとって非常に重要となる．

この重要性をあえて指摘する理由は，もともと情報は意思決定過程のほんの一部分に過ぎないにもかかわらず，コンピュータ専門家は情報システムが意思決定の中心的役割を果たすと思い込んでいる場合が多いからである．彼らにとって，技術的に美しくエレガントに情報を提供できるシステムは，当然，組織が必要としており，経営の意思決定の大前提であるかのようなものになっている．しかし，その意思決定は，多くの場合ドロドロとした政治的色彩を帯びている．

情報システムの構築が組織の中の政治的プロセスを引き起こす最大の理由は，情報が知的商品であるばかりでなく，政治的資源ともなるからである．新しい情報システムによる情報流通のやり直しは，すでに何度も触れているように，特定のグループの利害に影響を与える．そもそも各組織単位は，自部門の情報を統制する権限があることから，その影響力と自治を得ていると考えられる．したがって，全社的観点からSISが必要であると唱えられても，その権限を簡単に明け渡すことはない．もちろん，このような政治的な思惑からする変化への抵抗は，表面的には，まったく別のタテマエ論的な理由づけをもって現れるかもしれない．

さらに，このような政治的影響の大きい情報システムの構築の意思決定プロセスは，自ずと政治的プロセスとなる．政治科学においては，組織はお互いに衝突する優先順位，目的，価値を持つアクタ（行動者）からなると見なしている（Linstone, 1984）．技術的に見てどんなに優れた情報システムであっても，その構築と稼働が組織内の権力関係を変化させる潜在的影響力を持っていると認識されると，意思決定場面でのアクタたちは，技術的側面だけでなく政治的側面から新しい情報システムについて評価し，その決定に抵抗を示すことになる．

3.4 「意味の体系」としての組織が変革の対象

以上の議論から，情報技術を組織に導入して変革を起こすとき，優れた技術を導入するだけでなく，その組織に共有されている「意味の体系」としての組織の文化と政治への影響を考慮することが，変革を成功させるために必要であることが明らかとなった．新しい技術が導入されるとき，組織の中には抵抗が生まれる．その抵抗は組織文化からの社会的慣性であり，組織内の政治的関係である．このような意味体系の変更にともなう変化への抵抗に対処することが人間軸の変革活動の最も大きな課題である．

つまり，この変革は，抵抗したり政治的行動をする物理的な個人を対象としていると考えるべきではない．変革の対象は，個人としての組織メンバというよりも，システム，すなわち組織を構成している要素の相互関連性であり，それについて人々が心の中に持っている共通認識としての「意味の体系」である．すでに述べたように，組織はさまざまな要因が複合的で双方向的な影響関係を持っているシステムであるから，個人を変革の対象とするだけではまったく不十分である．システム（組織）の内部の一つの側面が変革されると，その付随結果としてシステムの他の側面も影響を受け，それがさらに2次的影響となって送り返される．したがって，新しい情報システム計画は，特定の個人を扱ったり，特定の制度だけを扱ったりするのではなく，組織全体をトータル・システムとして考えるアプローチをとらなければならない．しかもそのトータル・システムは，物理的に存在しているものとしての重要性と同時に，それが人々の心の中に認識された「意味の体系」となっていることの重要性に注目しなければならない．

個人を変革することが組織の変革につながらないことは，社会心理学的な研究からも明らかになっている．たとえば，小人数のグループで心を開いた話合いをする感受性訓練が個人の考え方を変えられることを示す証拠は多く，また場合によってはその小グループ自身の変革にも効果があることが研究されている．しかし，この手法によ

る個人の変容が組織を変革する方法になるという証拠はほとんどない（キャンベルとダンネット，1968）．人々に共有された「意味の体系」としての組織の文化や規範が変わらないままである限り，個人にそれを捨てるように求めても，抵抗するのみである．

4. 組織文化と政治の基本要素

人間軸の活動の目的は組織内で共有されている「意味の体系」の変革にあることが明らかになったので，その具体的な活動内容を次に検討するが，その前に，組織の「意味の体系」の現れである文化や政治についてもう少し説明を加えておくことにする．組織の文化や政治という用語は，日常的に使用される割には内容がよく理解されていないものの一つであり，それらを具体的に理解しておくことが人間軸からの情報システム計画を考える上で必要である．組織の文化と政治に関する見方は，社会心理学の概念が使用でき，その大きな要素として，ここでは，規範，価値観，パワー（力）をとりあげて概観しておく（バーグ，1987）．

4.1 規　　範

規範とは，人々が準拠する行動の基準もしくはルールである．ある集団のメンバが変化に対して激しく抵抗する場合，その集団は特定の，あるいは一連の規範を持っており，メンバはそれに準拠しようとする．組織がまとまりのある集団として存続していくためには，そのメンバが特定の規範に準拠している必要がある．しかし，規範そのものが人々の自由を制約しすぎ，自立を妨げるものであると，その規範に準拠することからは創造的な活動は生まれない．また，生産性や効果的な業績という観点からは，あまり意味のない規範が組織の中で守られている場合もある．

規範には，目に見える明示的なものと目に見えない黙示的なものがある．組織の中で目に見えやすい規範の例として，服装，家具の配置，勤務時間，組織図といったものに関連するルールとしての規範がある．このような明示的な規範は公式組織を反映している場合が多い．一方，組織の中を一見しただけでは見えない黙示的な規範の例としては，会議の時の発言のし方とか，決裁を求めるために当然やるべきとして考えられている根回しの順序等があげられる．黙示的な規範は，それに準拠していることにまわりの人が気づかない場合が多く，非公式組織を反映している．規範を氷山にたとえれば，明示的規範は海面に出ている氷山の一角で，黙示的規範は海面下にある大きなものである．すなわち，組織内の人々の社会生活をなすものが非公式組織であり，それを成り立たせている黙示的な規範の方がはるかに規模が大きい．

社会心理学者たち（レビン，フライシュマン，キャンベル/ダンネット）の研究によると，メンバの個々人に集団の規範に合わない新しい行動をするよう求めると変化に対する抵抗が強く出るが，その規範自体が変われば抵抗はなくなるという．たとえば，有給休暇がたくさんあっても，日本企業では一般従業員が率先して取得することがやりにくい．しかし，トップが率先して休暇を取れば，その下の管理者たちも取りやすくなり，一般従業員も休みやすくなる．これは，組織の黙示的規範を変えることで人々の行動を変えようとする例である．また，規範を変えようとする場合，新しい規範がどうあるべきかについて従業員自身が決定を下せる度合が高いと，成功しやすくなることも確かめられている．これは，人間は自分達が直接影響を受ける意思決定に参画

していれば，その決定を実施するコミットメントも高くなるという傾向と関係している．参画的なマネジメントの重要性が，規範を変えることにおいても当てはまると言えよう．

4.2 価　値　観

価値観とは，特定の行動様式について，あるいはあるものの究極的な状態について，これとは逆の，もしくは相いれない行動様式や究極的状態よりは，個人的にも社会的にも好ましいとして，持ち続けている信念である．規範が行動を準拠させるべきルールであるのに対して，価値観は，行動や考え方の善悪，良否を判断する基準としての信念となる．具体例をあげてみよう．「ここでのやり方は前例に従うことである．」という発言は規範について言っている．一方，「このやり方はあのやり方より好ましい．なぜなら目標を順序よく達成させることができるし，したがってずっと効果的な方法ということになる．」という発言は価値観を反映している．人間の価値観は何が良く何が悪いかという信念から生まれる．

組織の持つ価値観や信念は，ときには神話あるいは儀式という形式を取ることがある．ディールとケネディ（1987）はこの点について次のように強調している．理念を形成し，英雄を作り，儀礼と儀式を定めることによって独自性を養った会社には強い文化がある．これらの会社には，製品だけではない次代に伝える理念と信念があり，利益を追うだけでなく，語るべき物語がある．管理者も従業員もその行動と考え方を見習うことのできる英雄がいる．要するに，これらの企業は人間の組織であって，組織メンバの日常生活に価値観と意味を与えているのである．会社を成功させる哲学としての価値理念は，社員全員に，共通の目的で結ばれているという意識と日々の行動の基準とを与える．

人々はしばしば，企業生活の日常的な儀礼的な面に注目することを忘れ，あるいはその有益さを見落としている．そして，「手順は略して，ずばり問題の本質に入ろう」等と言う．しかし，そうすることによって，彼らは他の人たちをその手順から排除し，彼らを阻害するばかりでなく，彼らの貢献による価値をも失っている．ディールとケネディは，強い文化，すなわち，強い価値観を持つ企業は優れた業績を達成できることを多くの事例から提示し，英雄や，儀礼，儀式，神話を持つ強い文化を構築することが必要であることを示している．

4.3 パ　ワ　ー

SISの構築は組織の変革を意図しているのであって，現状のパワー関係の変革となる．パワーとは，他者に影響を与える能力・力量のことである．また，パワーを持っている人間は，他者が望むいろいろな資源をもっている人間でもある．組織の変革によって，このような意味における現状のパワー関係を組み替え，新しいパワー関係を作り出さねばならない．

管理者のパワーは職位からくるものであり，合法的なパワーである．それは，組織のルール(規範)，方針，慣習，およびそこに付与されている権限によって，合法的なものとなっている．逆に言うと，管理者のパワーの及ぶ領域は，せいぜいその職位に関連する規範や価値観に部下が従う程度の範囲となる．一方，集団や組織の規範に準拠するのは，部下よりむしろ管理者としてのリーダの方であり，皮肉なことに，こう

した規範を最大限変えることのできるのも，またリーダである．管理者のパワーの源泉が限定されたものであればあるほど，管理者は参画的な方法に依存して変革をもたらさなければならない．

カンター（1984）は，組織内におけるパワーの主要な源泉として活動と人脈をあげている．活動では次の三つのタイプのものが鍵となっている．その第1は非凡な活動で，新しい地位・役割について第一人者となること，第2は目だつ行動で，人の目につくことの重要性をいう．第3は組織上の当面の切迫した問題に関係した活動である．これらの活動に成功していくことで，その人は組織の中でパワーを持てるようになる．

さらに，人脈を通じて他の人からパワーが提供されることもある．第1に重要な人脈は後援者である．ここで言う後援者とは，自分の後楯となって闘ってくれる人，緊急時に組織階層のバイパスを可能にしてくれる人，そして自分が重要であることを示してくれる人である．すなわち，組織の上層部に重要な後援者を持つことによりパワーを持てるようになる．人脈の第2は同僚との人脈である．どんな組織のパワー源泉にとっても同僚からの承認・支持は必要である．第3は部下との人脈である．管理者は，部下の潜在的な力を引き出さなければ，自分の業務を遂行することもできないし，昇進に必要な堅固な基盤を築くこともできない．

カンターは組織内でのパワーのなさ（パワーレスネス）についても次のように述べている．人は，公式の役目によって命令を出す権限を持っていても，非公式な政治的影響力や，資源への接近，後援者，あるいは動ける見込み等に欠けていると，その組織ではパワーがないと見なされる．パワーレスの人は，ルールに対して執着し，縄張り意識を持つことでパワーを持っている気分になろうとする．

5. 人間軸の活動プロセス：「意味の体系」の再編成

以上三つの章にわたって，新しい情報システム計画における人間軸での活動対象とすべきものを述べた．その本質は，組織メンバに共有されている「意味の体系」を再編成することである．つまり，人間軸での活動は，SISが経営戦略の要として活用されるのに必要な新しい「意味の体系」として組織の規範，価値観，パワーを構築していくことであり，既存の規範，価値観，パワーを再編成することである．

ここで重要なのは，人間軸の活動の中心が，アウトプットそのものよりも，この活動において既存の組織の文化や政治と対決していく過程，つまりプロセスにあるということである．組織が深刻な危機的状況にあり，組織メンバがそのことに十分気づいているのであれば，人間軸の活動は単純なものになるであろう．しかし，組織メンバの多くがそのことに気づいていないのであれば，変革を組織内に導入するための人間軸での活動はより複雑なものとなろう．このことは，人間軸での活動プロセスが，変革を成功させる重要な鍵となることを示している．最終的に必要な「意味の体系」を組織内に作り上げるために，それぞれの状況にあった手を次々と打っていくのが人間軸の活動プロセスとなる．

5.1 解凍・移動・再凍結のモデル

それでは，人間軸においてはどのような活動プロセスがとられるべきなのであろうか．その具体的なプロセスを考察するに当たり，本稿では，Schein（1987）の提唱す

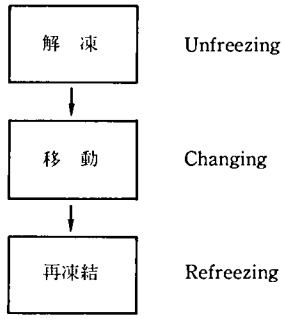


図2　解凍・移動・再凍結のモデル

Fig. 2　Unfreezing・changing・refreezing model

るモデルを基礎にすることにする．このモデルは，組織に変化を導入していくプロセスを示す最も基本的なフレームワークで，「解凍」，「移動」，そして「再凍結」の三つのステップからなっている（図2）．

SISの構築が組織にとっての変革であり，その計画活動のプロセスは，やはり基本的にこの三つのステップを通ることになる．ここではまず，各ステップを次のように説明し，このモデルの重要性を述べる．

「解凍」とは，その組織のメンバに一定の行動をとらせている力を変え，彼らを動機づけて変化への準備をさせることである．言い方を変えれば，安定的な均衡を崩すことであり，変化への圧力を高め，あるいは変化への脅威のいくつかを取り除くことにより達成できる．SISの構築においては，それを受入れる準備体制を組織の中に作り上げることが解凍となる．つまり，できるだけ多くの組織メンバがそのシステムを構築することの意味を理解し，その目的のためにこれから組織の中でどのような活動が展開されるべきであるかを理解していくようにする．

「移動」は，理解した変化の方向に向かって，必要な新しい行動や考え方を学習していくプロセスである．このステップで実際の変化が導入され，新しい行動や考え方が実施に移される．SISの構築について言えば，計画活動が実際に行われてSISが構想され，システム体系として策定されるのがこのステップである．

「再凍結」とは，新しく導入された変化を定着させるステップである．すなわち，変化した行動や考え方を，変化せずに継続している行動や考え方と再統合する．このステップによって，導入された変化は変化でなくなり，新しい安定した統合体としての組織となる．SISの構築においては，この再凍結ステップで，すでに策定された情報システム計画が組織内で承認され，公式的な認知が形成される．そして，SIS計画を実施に移すことで，物理的な情報システムとして形をなしてくるのである．

5.2　「意味の体系」を組み替えるプロセス

解凍・移動・再凍結のモデルの本質的な有用性は，人間軸での活動の目的が組織で共有されている「意味の体系」の変革にあることを思い起こすと，一層明らかとなる．SISの構築による人間軸での変革の対象は，複合的で双方向的な関係を持つ組織要因であり，それについて人々が心に抱いている共通認識としての意味の体系である．それが組織の文化や政治の基礎になっており，規範，価値観，パワーという具体的な形

をとる．このような意味の体系がすでに人々の心の中に確立され共有されているならば，新しく導入される情報技術は，その意味の体系の中において，それこそ意味を持たなければならない．

そのために必要なステップはまず解凍である．解凍によって既存の意味の体系が再検討され，新しい意味の体系の必要性が理解され，その受入れ準備がなされる．移動のステップに入ると，すでにある受入れ体制の上に新しい技術が導入される．解凍によって既存の意味の体系が柔らかくなっているために，導入される新しい技術が異物ではなく，新しい意味の体系に必要なものとして認識される．そしてこの新しい技術が人々の心の中で共有化され，新しい意味の体系として再構成される再凍結のステップとなる．すなわち，この解凍，移動，再凍結の3ステップとは，人々に共有されている意味の体系を組み替えていくプロセスである．

新しい技術を導入するために必要なこのような意味の体系の組み替えプロセスは，なぜ，この三つのステップを通る必要があるのだろうか．Schein の言葉に従えば，まず解凍しなければ移動できず，移動してからでなければ再凍結できない．しかし，この説明のしかたは「氷」に例えているだけであって，何ら人間の持つ意味理解や認識のしかたの本質的メカニズムに基づいて説明しているわけではない．解凍・移動・再凍結のモデルの有効性を一層高めるために，このモデルが人間の意味理解や認識の構造から言って極めて理にかなっていることを知っておく必要がある．この点を次の章で説明する．

6. メタ認識による創造的破壊

組織を変革する解凍，移動，再凍結の3ステップは，組織が自分自身に対して行う「創造的破壊」のプロセスであり，今まで保ってきたものを壊し，新しいものに創造的に変換していくプロセスである．それを，人々の行動様式や価値観を組み替えていく解凍・移動・再凍結の3ステップとして表現したものであり，本質は，人が心の中で安定的に保とうとする「意味の体系」を新しく創造的なものに組み替えるプロセスである．

意味体系の組み替えのプロセスが，なぜ「解凍・移動・再凍結」と比喩できる3ステップをたどるのかを理解するために，ベイトソン（1982）の人間の認識構造に関する考え方を用いることにする．その中心をなす概念は「メタ認識」であり，従来の意味の体系をメタ認識することが新しい意味の体系の必要性の認識になり，それを創造していくことになる（メタとは「超越する」，「変化する」という意味の接頭辞である）．すなわち，人々のメタ認識を起こしていくプロセスこそが解凍，移動，再凍結を行っていくプロセスなのである．言い換えれば，継続的な創造的破壊を行える組織であるためには，組織メンバにメタ認識が必要である．

6.1 人間の脳における情報認識

メタ認識がどのようなものであるかを説明するために，その基礎となる人間の脳における情報認識の構造をまず述べる．クック（1988）によると，人間の脳における情報処理過程では，物事，つまり事象（モノ，出来事，あるいはメッセージ）を認識する時は，2種類の情報をもとにしている．第1は，意識が焦点を当てている「事象」そ

のものに関する情報であり，第2はその事象が配置されている「文脈(コンテクスト)」に関する情報である．

たとえば，ある情報システムについて焦点を当てると，そのシステムの物理的な規模や技術的な特性，あるいはアウトプットの仕様が「事象情報」となる．一方，なぜその情報システムがその職場で必要とされるのか，あるいは今後の経営にどのように役立ち，今の仕事とどのように結び付いていくのかに関する意味や価値についての情報が，「文脈情報」となる．

すなわち，人間が事象を認識する時には，事象そのものの情報の認識だけでなく，その事象の存在する理由についての文脈情報をも同時に認識する．この二つの種類の情報を統合的に認識することによって，人間はその事象を存在感のある現実的な事象として認識し，その意味をとらえることができる．そのとき我々は，全体像を把握したと言い，一段高い所から本質を見ることができたと言う．SISの構築において，その有効活用に必要な新しい組織の意味体系が必要であると述べてきているのも，このような情報認識のメカニズムが人間の頭の中にあるからである．物理的な情報システムそのものの事象情報だけでなく，そのシステムの存在価値が文脈情報として組織の中に理解され，新しい意味の体系としての組織の文化が形成されねばならない．SISの事象情報と文脈情報の両方が統合的に理解されて，初めて組織の中に有機的に存在する情報システムとなる．

ちなみにクックによれば，ある事象に焦点を当てて事象そのものの情報を認識するのは分析脳と言われている左脳であり，事象が置かれている文脈に関する情報を認識するのが感情の脳と呼ばれている右脳である．さらにこれらの情報を統合する機能をはたすのが二つの脳を連結している脳梁である．したがって，左脳だけによる事象の分析的な認識が行われても，背景となる事情や存在価値がまったく理解されないために，その人にとって，その事象は実在感のない宙に浮いているものにすぎなくなる．また，右脳だけによる背景や文脈の認識が行われても，焦点の定まらない雰囲気的な感情や不安感の意識にしかならない．このような二つの脳による情報認識の機能分担を脳梁が統合することによって，初めてその事象の意味や価値が認識され，実在感をもってその事象に対する行動がとられるようになる．

6.2 メタ認識のむずかしさ

ベイトソンは，以上のような事象情報と文脈情報の統合的な認識を「メタ認識」と呼んだ．日常的な言葉でよく言う「全体像を把握した」，「物事の本質を理解した」，あるいは「一段高い所から理解した」という認識のし方がこれに相当する．ベイトソンはメタ認識の概念を導くために二つの下位概念，「デジタル認識」と「リレーショナル認識」を用いた．デジタル認識は，個々に存在する事象（出来事やメッセージ）をそのまま個々にとらえて認識することである．リレーショナル認識は，それらの間にある関係性や意味を認識してそれらの事象からかもしだされる文脈（コンテクスト）をとらえることである．このデジタル認識とリレーショナル認識とを同時に統合的に機能させることで得られるのがメタ認識である．

この概念は，1970年代にベイトソンによって提案されたものであるが，すでに述べたことから明らかなように，その後に現れたクックによる脳の情報認識構造の説明に

極めてよく符号する．つまり，左脳による事象情報の認識がデジタル認識であり，右脳の文脈情報の認識がリレーショナル認識である．そして，脳梁を介した事象と文脈の統合的な認識がメタ認識となる．

しかし，真の意味でメタ認識することは人間にとって非常にむずかしいことに注意しなければならない．通常，人間は一つ一つの事象をデジタル的に認識し，その経験から，それら多くの事象の背後にある文脈や意味を抽出して認識しようとする．それがリレーショナル認識である．ところが，一度この「意味の体系」が頭の中に出来上がると，次の新しい事象の認識からは，すでに作り上げた意味の体系に合うようにリレーショナル認識が行われるようになる．自分にとって都合のよいように解釈してしまう．これが人間の認識活動に一貫性と安定性を与える．しかしその結果，自分にとって新しい事象が起きても，自分の意味の体系からそれる事象はエラーとして処理してしまい，認識の視野に入れることがむずかしくなる．しかも，新しい事象のうち，自分の意味の体系にとって都合の良い部分は，理解できたとして心の中に取り入れる．

本来ならば，新しい事象が起き，それが自分の持つ理解の文脈に合わないとき，新しい事象の配列を試みて既存の文脈の再検討をしなければならない．その結果，新しい文脈を頭の中に再構成することに成功して，その事象を1段高い所から理解できるようになる．これが真の意味でのメタ認識となる．デジタル認識とリレーショナル認識を絶えず新しく行い，メタ認識していかなければならない．しかし，一貫性と安定性を保とうとする人間の認識のメカニズムがこれを困難にしている．

人の情報認識におけるこのような一貫性と安定性を求める特性は，おそらく文脈情報を外界から直接入手するのではなく，主にデジタル認識で得た情報をリレーショナル認識で加工，構成することによって文脈情報を作っているからであろう．つまり，文脈情報はあくまでも人が自分の頭の中に作り上げる情報であり，それが現実世界での文脈を正確にとらえているかどうかは別の問題となる．リレーショナル認識においては，文脈として一貫性を持つ情報に構成することがもともとの機能なのである．だからこそ文脈としてスジが通るのであり，ひとたび形成されるとその安定を維持しようとする機能が働く．したがって，自分が認識した外界の事象とその文脈のパターンが，現実世界を正確に反映しているかどうかのチェックをしていくメタ認識が極めて重要となり，またそのむずかしさもある．

6.3 メタ認識のプロセス

それでは，組織の変革ステップがなぜ解凍，移動，再凍結の3ステップを通らねばならないのかを，メタ認識のプロセスとして考えてみることにしよう．すでに述べたように，組織の構成要素は複合的で双方向的な関係にあり，それらとの相互関係を通して組織メンバは組織文化を「意味の体系」として認識している．この意味の体系は，とりもなおさず人々が共有している文脈情報であり，価値判断の体系である．日常的な個々の仕事や技術が「事象」としてデジタル認識されると，その仕事や技術が必要とされる事情や背景は，すでに存在している文脈情報の中でリレーショナル認識され，その存在価値を認めることができる．

しかし，SISのようにまったく新しい技術が組織の中に導入されると，それをきっかけにして他の組織の構成要素との相互関係が変化し，今までなかったような仕事や，

仕事と仕事の関係が現れ，人々にとって経験したことのない新しい事象が頻発するようになる．ここで求められるのがメタ認識である．既存の文脈に当てはまらない事象であるならば，文脈そのものを再検討して新しいリレーショナル認識をしていく必要がある．ところが，自分の持つ環境理解の文脈に一貫性と安定性を保とうとする傾向があるために，メタ認識がなかなか進みにくい．

そこでとられる第1のステップが解凍である．すなわち，人々がすでに持っている文脈情報（つまり既存の意味の体系）を変える「必要」があり，「新しい方向」へ進まなければならないことを理解するような新しい事象の経験を進めるのである．しかし現状はその必要性を満たすものが具体的な形で存在しているわけではなく，方向性を示す先に着いてみて初めて物理的に実現するのである．したがって，このような「必要性と方向性の情報」は極めてイメージ的な文脈情報なのであり，それを認識させられるような事象を数多く工夫して人々に経験させなければならない．そのような事象情報のデジタル認識を重ねて，「必要性と方向性」という文脈情報をリレーショナル認識してもらう．

この働きを狙う新しい事象は，新しいSISのもとでの仕事そのものをやらせることではない．新しい仕事を無理やり経験させて，結果的にその必要性と方向性をリレーショナル認識させるという方法も可能ではあろうが，変化に対する強い抵抗を引き起こすことになりかねない．あるいは，その必要性と方向性を言葉にして話して聞かせたり，文書にして読ませたりするだけでは人々の頭の中に文脈情報として形成されない．聞いたり読んだりする事象は，単にデジタル認識すべき一つの事象でしかない．すなわち，変化の必要性と進むべき方向性というコンセプトとしての文脈情報を人々の心の中に形成できるように，数多くのいろいろな事象を工夫して経験させ，リレーショナル認識させることが解凍のステップとなる．

必要性と方向性の文脈が人々の心の中に形成されたならば，初めて具体的なSIS計画が策定される移動のステップに入る．すなわち，変化が必要であり進むべき方向性があるという文脈の中での新しい情報システムの具体的な計画活動となり，導入される変革に抵抗するという社会的慣性は現れてこない．

再凍結のステップに入ると，策定されたSIS計画が実施に移される．このステップで初めて，計画された情報システムの具体像が組織内に浸透する．そのために，計画されたその特定の情報システムを稼働させれば経営的に実効が上がり，戦略的に役に立つという「実効性」の文脈情報があらかじめ必要になる．計画として具体化されてもまだ物理的なシステムとして組み上げられていないから，それが実働した時の価値を実働する前に認識しておいてもらわねばならない．それが「実効性」の文脈情報として共通認識されなければならないところのものである．

このステップにおいては，移動のステップにおいて計画された具体的な情報システムの機能やアウトプット等の仕様に関する事象情報が，計画実行による実体的なさまざまな事象の経験で裏打ちされる．このような事象のデジタル認識から，計画されたそのSISの経営的な価値や実効性が新しい文脈情報としてリレーショナル認識される．それによって，そのSISの有効活用を目指す新しい意味の体系としての組織の文化が形成される．

以上のような解凍・移動・再凍結の三つのステップを通ることで事象情報と文脈情報を交互に発生させ，組織を変革するメタ認識を形成する．このメタ認識のプロセスによって，組織内の人々に共有されている「意味の体系」としての組織の文化を再編成し，新しいSISを有効活用していけるものにする．

7. おわりに

本稿は，SISの構築における人間軸の方法論の諸条件を明らかにした．一般に，方法はそれぞれの企業の状況に合わせて開発するとよいと言われる．しかし，標準的な方法は個別企業に適合した方法を開発する上で有用な参考となる．そこで，本稿で明らかにした諸条件を満たすSIS構築のための情報システム計画の方法を併せて開発したので，参考に付録として採り上げておく．

本稿は，慶応大学大学院 高木晴夫助教授と筆者の「SIS経営革新を支える情報技術」（日本経済新聞社，1990年2月）の第4章「人と組織への接近」と第5章「人間軸の展開－3つのステップ」の内容をほぼ再掲したものである．

参考文献 [1] C. Argyris, "Double Loop Learning in Organizations", Harvard Business Review, 1977, pp. 115～125.
[2] C. Argyris, "Management Information Systems: The Challenge to Rationality and Emothonality", Management Science, Vol. 17, No. 6, Feb. 1971. pp. 278～292.
[3] アリー・P・ドゥハウス，〝組織の学習能力向上のための計画策定〟，ダイヤモンド・ハーバード・ビジネス，Jun.-Jul. 1988, pp. 11～24.
[4] G. ベイトソン著，佐藤良明訳，「精神と自然－生きた世界の認識論－」，思索社，1982.
[5] G・ベイトソン著，佐伯泰樹・佐藤良明・高橋和久訳，「精神の生態学」（上），思索社，1986.
[6] G・ベイトソン著，佐藤良明・高橋和久訳，「精神の生態学」（下），思索社，1987.
[7] J. L. ボウワー，〝技術型組織と政治型組織　その特徴とジレンマ〟，ダイヤモンド・ハーバード・ビジネス，Oct.-Nov. 1983.
[8] W. W. バーグ著，吉田哲子訳，「組織開発教科書」，1987年9月，プレジデント社.
[9] N. D. クック著，久保田競・桜井芳雄・大石高生・山下晶子訳，〝ブレイン・コード〟，紀伊國屋書店，1988.
[10] S. M. Davis, "Managing Corporate Culture", Ballinger Publising Company, 1984.
[11] T. ディール/A. ケネディ，「シンボリック・マネジャー」，新潮社，1987.
[12] M. A. Diamond, "Resistance to Change: A Psychoanalytic Critique of Argyris and Schon's Contributions to Organization Theory and Intervention", Journal of Management Studies, Vol. 23, No. 5, Sep. 1986, pp. 543～562,
[13] EDP Analyzer," Manage the Impact of Systems on people", Vol. 23, No. 5, May 1985, pp. 1～12.
[14] V. Grover, A. L. Lederer and R. Sabherwal, "Recognizing the Politics of MIS", Information & Management, Vol. 14, 1988, pp. 145～156.
[15] J. C. Henderson, J. G. Sifonis, "The value of Strategic IS Planning: Understanding Consistency, Validity, and IS Markets", MIS Quarterly, June 1988, pp. 187～199.
[16] R. M. カンター，（長谷川慶太郎訳），「ザ・チェンジ・マスターズ」，二見書房，1984.
[17] P. G. Keen, "Information Systems and Organizational Change", Communications of the ACM, Vol. 24, No. 1, January 1981, pp. 24～33.
[18] P. G. W. Keen and E. M. Gerson, "The Politics of Software Sysems Design", Datamation, November 1977, pp. 80～84.
[19] W. R. King, "Evaluating Strategic Planning Systems", Strategic Management Journal, Vol. 4, 1983, pp. 263～277.
[20] W. R. King and D. I. Cleland, "A New Method for Strategic Systems Planning",

Business Horizons, August 1975, pp. 55～64.
[21] D. A. Kolb and A. L. Frohman, "An Organization Development Approach to Consulting", Sloan Management Review, Fall 1970, pp. 51～65.
[22] L. B. Kurke and H. E. Aldrich, "Mintzberg was Right!: A Replication and Extension of The Nature of Managerial Work", Management Science, Vol. 29, Ne. 8, August 1983, pp. 975～984.
[23] A. L. Lederer, A. L. Mendelow, "Convincing Top Management of the Strategic Potential of Information Systems", MIS Quarterly, Dec. 1988, pp. 525～534.
[24] A. L. Lederer, A. L. Mendelow, "Information Systems Planning: Top Management Takes Control", Business Horizons, May-Jun. 1988, pp. 73～78.
[25] H. A. Linstone, "Multiple Perspectives for Decision Making", North-Holland, 1984.
[26] M. L. Markus, "Power, Politics, and MIS Implementation", Communications of the ACM, Vol. 26, No. 6, June 1983.
[27] H. Mintzberg, "Planning on the Left Side and Managing on The Right," Harvard Business Review, July-August 1976 (邦訳, H. ミンツバーグ, "計画は脳の左で, 経営は脳の右で", ダイヤモンド・ハーバード・ビジネス, Feb.-Mar. 1983 pp. 65～75.
[28] R. A. ニスベット著, 堅田剛訳, "歴史とメタファー", 紀伊國屋書店, 1987.
[29] 野中郁次郎, "知的創造経営への提言", ダイヤモンド・ハーバード・ビジネス, Feb.-Mar. 1989, pp. 4～13.
[30] 野中郁次郎, 網倉久永, "企業はいかにして新たな視点を獲得しうるか", ダイヤモンド・ハーバード・ビジネス, Dec.-Jan. 1988, pp. 43～51.
[31] G. ピンチョー著, 清水紀彦訳, 「イントラプルナー社内企業家」, 講談社, 1985.
[32] M. ポラニー著, 佐藤敬三訳, "暗黙知の次元", 紀伊國屋書店, 1980.
[33] M. E. ポーター著, 土岐・中辻・小野寺訳, 「競争優位の戦略」, ダイヤモンド社, 1985.
[34] M. E. ポーター, V. E. ミラー, "進展する情報技術を競争優位にどう取り込むか", ダイヤモンド・ハーバード・ビジネス, Oct.-Nov. 1985, pp. 4～16.
[35] J. F. Rockart and M. S. Scott Morton, "Implication of Changes in Information Technology for Corporate Strategy", Interfaces, Vol. 14, No. 1, January-February 1984, pp. 84～95.
[36] G. Salaway, "An Organizional Learning Approach to Information Systems Development", MIS Quarterly, Jun. 1987, pp. 245～264.
[37] E. H. Schein, "Process Consultation", Addison-Wesley, 1987.
[38] N. M. Tichy, "Managing Strategic Change, Technical, Political and Cultural Dynamics", John Wiley & Sons, 1983.
[39] 山田善靖, "マネジメントシステムと情報技術", オペレーションズ・リサーチ, 1984年11月.

[付　録] 戦略情報システム構築のための人間軸の方法論

1. はじめに

戦略情報システム (SIS) の構築に必要な人間軸での活動は, 既存の組織の規範, 価値観, パワーを再編成することであり, 新しい情報システムが経営戦略の要として活用されるのに必要な新しい組織の文化を作り上げることである. この活動の本質は, 組織メンバに共有されている「意味の体系」を作り変えることであり, そのプロセスは, 解凍, 移動, 再凍結の三つのステップを通らねばならない.

ここで, これらのステップをどのような活動で構成していけばよいかの方法論を検討する. この方法論は, 三つのステップのそれぞれにおいてどのような活動を構成していけばよいかの基準や考え方を示すものであり, 各ステップごとに活動を例示してその考え方を議論している. それはあくまでも方法論についてであって, どのような活動をどのように構成していけばよいかの考え方や方向性を与えるものである. つまり, 例示する活動は直接そのまま個々の企業で実行に移すべきとするものではなく, それぞれの企業で具体的な活動を創意

工夫していくためのガイドラインとなるものである．

ひとくちにSISといっても，事業の形態や業種の特性，あるいはその組織の既存の文化や歴史的な背景に応じて，さまざまなバリエーションのSISが工夫されねばならない．したがって，そのための人間軸での活動が具体的にどのような形をなすべきかについても，ここで展開する方法論の枠組みの中で，SISを構築する個々の企業の状況に応じた工夫の余地が大きく残されている．ここにCIO（チーフ・インフォメーション・オフィサ）やSIS担当者の創造的活動が求められる．

2. 解凍・移動・再凍結の方法論

人間が持っている情報認識の能力の観点からすると，人間軸における解凍・移動・再凍結のプロセスは，SISに関する事象情報と文脈情報を交互に作り上げていくメタ認識のプロセスである．人間軸の方法論の基本をなすのがこの考え方であるので，もう一度振り返って，方法論として明確に述べておくことにしよう．そのエッセンスは，SISの「必要性と方向性」および「実効性」という文脈情報を，具体的な情報システム計画活動を行うステップをはさんで，2段階的に形成することである．これらが解凍，移動，再凍結の三つのステップとなる（図1）．

人間が環境におけるさまざまなことを現実的に認識するためには，多くの事象の情報をそれぞれデジタル認識し，そこから事象間の関係性（すなわち文脈）をリレーショナル認識していかなければならない．これらの事象情報と文脈情報を統合的にメタ認識してはじめて意味のあるものとしてその事象が把握される．

2.1 「必要性と方向性」の文脈形成

SISを新しく構築する場合，構築することの必要性や価値，あるいは構築していく方向性が組織メンバに理解されていることが前提となる．求められるSISの具体像がまだ明確でない時点であるにもかかわらず，それを構築する必要性と方向性の理解を組織の中に作り上げることをまず行わなければならない．これが解凍のステップとなる．ここで重要なことは，「SISを構築しなければならない」という必要性の情報と，「これからの進展と成果のイメージ」としての方向性の情報は極めて文脈的でコンセプト的な情報であり，リレーショナル認識が必要になるということである．

たしかに，文章等を用いてこの必要性と方向性を述べることはできるが，それを読むだけでそれらの理解がなされることはまずない．いろいろな事情やさまざまな状況を想定し，SISを構築する必要性と方向性を考えなければならないことをたくさん経験して，はじめて「必要だ」という認識が形成され，「この方向へ行くべきだ」という理解になる．これはコンセプトとして文脈情報的に認識されるものであり，そのためには多くの事象をデジタル認識して，そこからの文脈情報としてリレーショナル認識しなければならない．

したがって，解凍のステップで行う活動がどのようなものであるべきかというと，必要性と方向性という文脈情報を認識させるための数多くの事象情報を発生する活動ということになる．これは直接的にSISを計画する活動ではなく，結果的にそのシステムの必要性と方向性をリレーショナル認識できるような多くのデジタル認識を可能にする活動でなければならない．これが第1段階の文脈情報の形成である．

次の移動のステップにおいては，すでに形成された必要性と方向性の文脈を背景として実際の情報システム計画活動が行われることになる．そこでは，SISが新しく必要とされてお

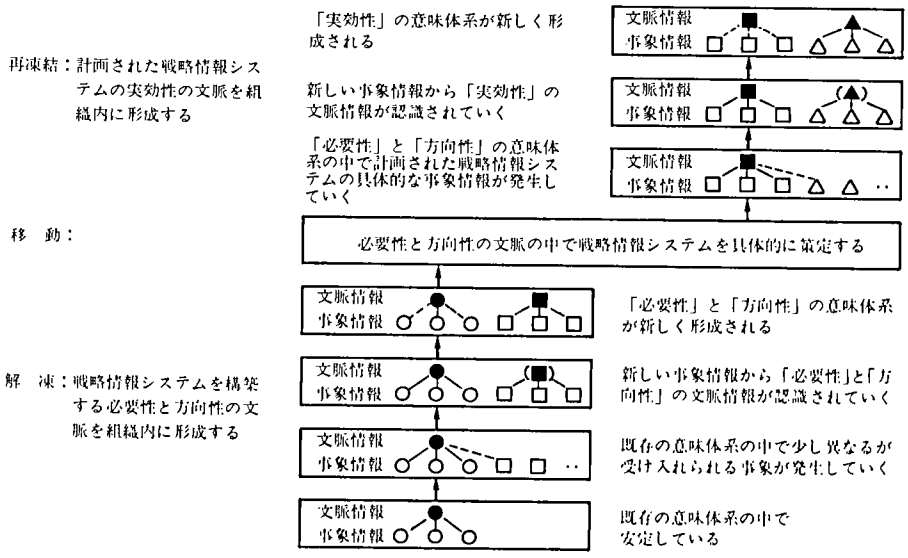


図1　解凍・移動・再凍結の方法論
Fig. 1　A methodology based on unfreezing・changing・refreezing model

り，そのための方向があるという文脈が組織の中ですでに共通認識されているために，変化への抵抗という社会的慣性が出現するのではなく，変革のための計画活動を推進するという慣性力が生れてくる．このような共通認識された変化の文脈の中で，SIS 計画が具体的な形として策定され，情報システムがどのような形をなすかの具体像が形成される．

2.2　「実効性」の文脈形成

再凍結のステップに入ると，計画された具体的な SIS の構築に向けてその計画の実施に移り，実施のために必要な数多くの活動が行われる．この段階において必要なことは，計画された「特定の SIS」の稼働が経営戦略的な効果と価値を生むという実効性の文脈情報を組織内に形成することである．「その SIS は実際に役に立つ」という理解の文脈がシステムの稼働以前に組織内に形成されて，はじめて実施活動が成功裏に行われる．これが第 2 段階の文脈情報の形成である．

解凍ステップでの第 1 段階の文脈情報の形成は，「新しく SIS を構築することの必要性と方向性」の認識であったが，この再凍結ステップでの第 2 段階の文脈情報の形成は，むしろ「計画された特定の SIS の実効性」の認識に向けられている．すなわち，特定されたとはいっても，まだ情報システムとして物理的に形をなしていない段階で，組織全体のさまざまな部署でその SIS を活用していくことが経営戦略的に具体的な実利をもたらすという共通認識を作り上げることである．移動のステップで数多く行われた具体的な計画活動の事象情報を基礎に，再凍結ステップでさまざまな活動を行い，その SIS の仕様やアウトプット，そしてその稼働による戦略的効果等のデジタル認識が進められる．その結果，その SIS の実効性がリレーショナル認識され，計画実施に必要な文脈が組織内に形成される．

3.　情報システム計画の関係組織と活動展開

3.1　関　係　組　織

次に人間軸における情報システム計画の活動を解凍・移動・再凍結の方法論に従って具体

的にどのように展開していけばよいかを述べることになるが，その前に，情報システム計画活動を担う関係組織を明らかにしておく．SISの構築に限らず，どのような組織変革であっても，変革活動を実際に担当するのは一つの部署や一人の人間ではなく，いくつかの関係する部署や人の集りである．そのような複数の関係組織の共同作業によって変革活動が遂行されていく．SISについて言えば，計画活動に関係するさまざまな組織が情報システム計画活動を実行し，またその影響を受けていくことになる．

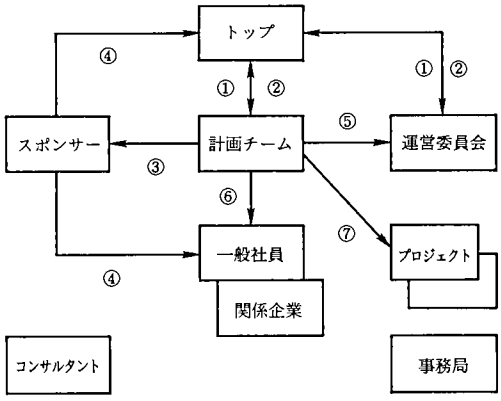


図2 関 連 組 織
Fig. 2 Related units

図2が情報システム計画で関係すべき組織の一般的な形を示している．SISを構築しようとする企業の個々の状況に応じて，構成される具体的な関係組織は多少異なるであろうが，図2はその一般的な形を示そうとするものである．

関係組織のそれぞれの詳細は各活動ステップの中で取り上げるが，おおよその内容は次のようになる．

「トップ」は，実質的に情報システム計画を開始し，その全体の進捗をモニターするグループないし個人である．「計画チーム」は，情報システムそのものを作ることを目的に作られる一時的で特別なプロジェクト・チームである．企業組織が大きい場合には，組織内から複数の人々が選抜されて計画チームが構成される．「運営委員会」は，情報システムに関連する意思決定のために設けられた部門横断的な組織である．計画チームの活動で公式的な承認が必要になった場合，この運営委員会に意思決定を求めることになる．情報システムが集中的に管理されていたMIS（経営情報システム）時代までならば，このような部門横断的な委員会は必要なかったであろうが，情報システムに関する権限が広く各部に分散されるEUC（エンド・ユーザ・コンピューティング）時代以降は，それが必要となっている．これらの計画チームと運営委員会にトップからSISの構築についての意向が伝えられ，それとのすり合わせが行われる．さらに，計画チームと運営委員会の庶務を行うために「事務局」が設けられる．

「スポンサー」は，組織内の政治闘争から計画チームの活動を防衛することを目的として設定される個人をさす．情報システム計画の意向がスポンサーに伝えられて，組織内の政治への対応が行われる．計画活動を進める上でのトップのパワーが組織内の政治闘争を押えるのに十分な場合は，必ずしもスポンサーを必要とするわけではない．この場合は，トップ自身がスポンサーとなっている．

「プロジェクト」は，計画チームから依頼される問題，業務機能，役割，情報体系等のより詳細な分析や設計を行う臨時の組織である．また，策定された情報システム計画に基づく後続活動として，情報システムや実体システム（ビジネス・システム）を設計・構築するために設けられる臨時の組織でもある．メンバには計画チームの人間が一部入ることが多い．

「コンサルタント」は，コンサルテーションを担当する外部あるいは内部の人間で，その人の持つ専門知識やコンサルテーション・スキルが求められる．

「一般社員」は，上記以外の人々であり，計画活動に直接関与するというよりも，むしろその影響を受ける人々である．それらは企業組織の各部門を構成している人々であり，異なった価値観や目的，優先順位を持ついくつかの連合（一人または複数の人間からなる派閥）から構成される場合がある．「関連企業」は一般社員と同様に，SIS によって影響を受ける企業である．

3.2 解凍・移動・再凍結の活動展開

以上のような一般形で考えられる関係組織の活動とその結びつきは，解凍・移動・再凍結の方法論に従うとどのような具体的な活動として展開できるであろうか．表 1 に示すのがその活動展開である．これは，三つのそれぞれのステップをどのような活動で構成していくと，方法論で狙っている文脈情報の 2 段階的形成がなされていくかを示している．

解凍のステップでは，「新しく SIS を構築する必要性と方向性」の文脈情報を関係組織間で共有することが目的である．それに必要なさまざまな事象情報を発生するために，具体的にどのような活動が各関係組織で行われるべきであるかを示している．解凍活動の目的はあくまでも必要性と方向性の文脈を構築することであり，一つの活動だけでその目的を達成す

表 1　解凍・移動・再凍結の活動展開

Table 1　Development of unfreezing·changing·refreezing activities

解　凍	トップのコミットメントを創り出す． 組織の問題を個人の問題に翻訳する． テーマを発見する． 外部コンサルタントを導入する． 内部コンサルタントを選出する． コーチング 診断する． スポンサーを見つける． 運営委員会を設置する． 計画チームを編成する． 日程を決定する． 計画活動を広報する．
移　動	チームワークづくり プロセス・コンサルテーション
再凍結	トップへ報告する． 運営委員会へ報告する． 中間発表する． 組織を編成する/プロジェクトを結成する． 委譲する． 発表する．

ることは不可能であることに注意すべきである．そのためには多くの事象情報を発生させることが必須であって，一つの活動だけに特化すべきではない．

移動のステップでは，形成された変化の必要性と方向性の文脈の中で，具体的な情報シス

テム計画の作成に必要な各関係組織の活動を示している．再凍結のステップは，「計画された特定のSISの実効性」という文脈情報を関係組織間で共有することが目的である．そのための事象情報を作る各関係組織間での活動を示している．このステップも解凍のステップと同様に一つの活動に集中しすぎることなく，様々の活動で多くの事象情報を発生させて実効性の文脈情報を形成すべきである．

ここで再度強調しておきたいのは，表1の各種活動は，解凍・移動・再凍結の方法論をどのような形で活動展開できるかの理解を促進するための例示であるということである．そのような種類の活動を展開することによって，方法論で狙っているような事象情報の提供と文脈情報の形成が可能になるという意味である．もちろん，これらをそのまま個々の企業で活動に適用することも可能であろうが，むしろ個々の企業の現状に適応した活動展開が図られるべきであり，そこでの創意によって新たな活動を工夫していくべきである．表1はそのための考え方の基礎となるものである．さらに，表1に示す各ステップでの活動は，必ずしも示された順序で行う必要はない．ある活動と他の活動の同時並行的に行ったり，あるいは不足であれば，必要に応じて一部もどって行うべきである．

4. 解　　凍

4.1 トップのコミットメントを創り出す

SISの計画活動は企業組織の全体と関連し，また，それに影響を与える大きな革新活動であるため，当初からトップの理解と支援が必要となる．それなくして，そもそもSISの構築を進めることは困難である．トップが，自社の組織とそこでの事業展開の問題を正しく認識しており，将来のあるべきビジョンをもってSISの構築を自ら推進していこうとする場合は，容易にトップのコミットメントを得ることができよう．すでに，トップの意識は解凍されており，変化に向けて柔らかくなっているのである．

むしろ問題は，トップがそのような意識を持っていない場合である．組織の中に問題が隠されていて，トップに聞き心地の良い公式的な情報しか伝わらないために，問題の情報がトップに届かないことが多い．これは組織の中でトップに接触できるのは，トップを取り巻く限られた権力者だけだからである．トップに悪い情報を伝えるということは彼らの権力構造を壊すことになりかねない．このような状態だと，トップは真の問題を認識しておらず組織革新を狙うSIS構築の必要性を感じない．ましてや組織革新にどのような方向性を持たせるかの意識もない．

硬直化した組織においてトップがやらなければならないことは，自分が認識している組織の状態は現実の問題を正確に反映していないというメタ認識を持つことである．そのためには，自らが自分の意識を解凍する必要があり，すでに述べたメタ認識のメカニズムからすると，同じようなトラブルが何度も発生し，現状の対応策はどれも役に立たないような事象を多く経験することである．同時にトップ自らが組織の階段を降りて行って，現場の問題状況を皮膚で実感しなければならない．また，自分の認識とは異なる多くの情報を公式・非公式なルートで常に自分に伝わるような仕組みを作っておくことも必要である．

メタ認識がまだないトップである場合は，SISの必要性に気づいた人間が，トップにメタ認識を起こさせる努力をしなければならない．トップ自身が現状認識に疑問を抱くような事象情報を組織内の人間がトップに多く与えるということでは，その方法の原理も同じである．たとえば，その人間が組織内の生の問題をトップに伝える役目を果たさなければならな

いであろうし，似たような問題で失敗した他社の事例や，あるいは解決に成功した事例を多く提示して，リレーショナル認識を図る必要がある．

4.2 組織の問題を個人の問題に翻訳する

組織メンバが変革の必要性を感じていないとき，トップや一部の人々が変革の必要を説いても，多くの人々は変革への熟成がないため一般に反応することはない．それは組織にたとえ問題が存在しても，自分自身に関係する問題として捉えていないからである．このような場合，現在の組織の悪化した状態を翻訳して組織メンバにわかるように伝える工夫が求められる．とくに，外部環境と接触していない企業内の人々に対しては，変革の必要性が彼ら自身にも関係していることを感知してもらわねばならない．しかし，変革の必要性そのものを伝えても，彼らはそれを受け入れることがないことは前にも触れた．

変革への組織の熟成を作り出すために，現状の組織の深刻さを組織メンバにわかるような形で，トップが組織に伝えていくことができる．たとえば，市場の求める製品を開発していないために収益性が悪化しているとき，ショック療法的に管理者のボーナス・カットや昇給の一時停止をして収益性の低下を実感させることもそのような事象の一つとして考えられる．すなわち，「組織の問題」を「個人の問題」に翻訳することにより，組織メンバに熟成を，言い換えれば必要性と方向性の文脈を作り出す．

4.3 テーマを発見する

SIS の計画活動を始める時のテーマは，一般社員にとって馴染みやすい言葉を使う必要がある．それによって，SIS 構築という組織全体の変革活動が自分にも関係するのだという認識を持たせるのである．逆に馴染みのない概念を使ったテーマにすると，専門家がやっているもので自分には関係がないという意識を持ってしまう．

たとえば，「情報システム計画の立案」とか「システム化計画の策定」といった抽象的なテーマであると，一般社員はその活動の具体的なイメージを把握できず，むしろ特定の専門家グループのための活動と捉えてしまう．このようなテーマでは，一般社員からの広い賛同が得られない．また，「その会社全体の革新」といった大きくかつ抽象的なものにしてはならない．問題があるといっても，一般社員は，人材，技術，商品，資金，あるいは制度などに問題があると思っているのであって，会社のすべてが問題になっているとは思っていない．すなわち，一般社員にとって理解しやすい一つのまとまりを持った問題を最初のテーマとして設定することが好ましい．

たとえば初期段階で，次のような形で一般社員自身に関係する具体的なテーマを設定し，SIS の構築必要性と方向性を一般社員にも考えさせていくことが可能である．

1) 傷ついているシステム……多くの人達から問題と認識され，その改善なくしては他の活動が適切に行い得ないと思われているシステムを採り上げる．たとえば，仕事の横の連係を阻害している「つぎはぎだらけの情報システムの再構築」．
2) 新しいシステム……まったく新しい方法で仕事を設計することが必要なシステムを採り上げる．たとえば，情報技術の発達で仕事の効率と効果を飛躍的に向上させられる「受注と出荷のリニューアル」．
3) 個別活動の運営方針……風化して一貫性がなくなってしまった従来の情報システムの開発活動に将来の方向性を与える．たとえば，今後の資源配分と優先順位を決定するための「今後の情報システムの構想作り」．
4) スタッフの教育……後日，情報システム計画の本番を行うに当たって，支援が必要な

人材を教育する．たとえば，情報システム計画活動の内部コンサルタントや事務局になると予想される人に対する「情報システム計画活動の演習」．

4.4 外部コンサルタントを導入する

大規模な変革には，多くの場合，外部のコンサルタントが導入される．外部コンサルタントの内部コンサルタントに対する優位性は，彼らがいくつもの組織や会社でコンサルテーションしているため，組織の問題に広く対応できる能力，スキル，経験を持っていることである．したがって，彼らは内部コンサルタントよりも客観的であると見なされる（Tichy, 1983）．

外部コンサルタントを導入することにより，トップの信念の強さや取り組み姿勢を一般社員に示すことができ，組織に対する解凍活動として用いることができる．また，専門的なあるいは高額なコンサルタントを使えば使うほど，トップのコミットメントが大きいと社員から理解され，「改革は動きだした，戻れない」との雰囲気を会社内に醸し出すことができる．

4.5 内部コンサルタントを選出する

コンサルタントは外部と内部を混在させることが理想的であり，一つのプロジェクトに外部と内部のコンサルタントのチームを作り，同時に働く方法を編み出すとよい（Tichy, 1983）．外部コンサルタントだけでプロジェクトをスタートした場合，できる限り早い段階で内部コンサルタントを任命すべきである．

内部コンサルタントとして，特定の分野で非常に高いスキルを持つ人間を選ぶことができ，外部のコンサルタントに較べて安い費用で優れたサービスを期待できる．また，彼らに求められる力は，人から意見を導出する技術，それをまとめる技術，組織の政治を理解する力であり，必ずしも情報システム計画のための技術を持っている必要はない．彼らはその会社のメンバであるので，多くの社員に気安く受け入れてもらえ，より接触しやすい．固い古い組織を解凍して柔らかくしていくために，人と組織に働きかけていく彼らの力が極めて重要になる．

内部コンサルタントとしての仕事は，計画活動の運営状況についてデータを集め，またチーム作りのセッションや各種のトレーニング・プログラム等の部分的な介入の実施を援助し，またいろいろな追加データを分析する．つまり，開始された運動の弾みを継続するように努めるのが内部コンサルタントの役割になる．さらに，彼らは組織内の特定の政治的な連合に属してないことが望ましいが，もしそのような人材の選出が不可能であるならば，複数の人間を選出すべきである．なお，庶務を行うために事務局は別に設ける．

4.6 コーチング

コーチングとは，内部コンサルタントや計画チームのメンバになりそうな人に対して外部コンサルタントが行う教育的な活動である．これは陰のコンサルテーションとも呼ばれ，外部コンサルタントが内部コンサルタントを舞台裏からコーチして，提案したり援助したり，あるいは重要な活動に先だっての実習セッションをもったりする（バーグ，1987）．

コーチングによって，外部コンサルタントと内部コンサルタント，あるいは計画メンバとの関係が密になり，スムーズなコミュニケーションができるようになる．そしてSIS構築の具体的な課題や問題点について検討する機会を多く持つことで，SIS構築の必要性と方向性の共通認識を深めることができる．そればかりか，変革活動のプロセスの理解を可能にし，内部コンサルタントらの自助的な行動を誘発してスムーズに活動が進むようにする．とくに，内部コンサルタントがこの種の変革活動について不慣れな場合は，活動の全体の流れを

理解させる上でコーチングが重要となる.

4.7 診 断 す る

計画的変革を推進していく上で,内部および外部のコンサルタントがその組織の文化や風土を把握しておくことが重要である. 解凍のステップにおける活動の目的が「SIS を構築する必要性と方向性」という文脈を組織内に形成することであるから, 現在の組織の文化に受入れられ,なおかつ必要性と方向性の文脈情報をリレーショナル認識してもらえるような事象を多く工夫しなければならない.既存の文化では受入れられない事象活動をいくらやっても, 狙っている文脈情報は認識してもらえない. そのために組織文化の診断が重要となり, 既存の組織文化に関する情報を集め, その価値観や規範, パワーの関係を理解しておくことが前提となる.

このような診断は, 敏感な感覚を持ったコンサルタントであれば, 変革活動を始めた時から直観的に非公式に行っている.あるいはもっと体系的に,アンケート分析やインタビュー, 各種経営文書の分析等によって行われる.

公式的な診断の多くは, 直接組織のメンバにコンサルタントが面会して情報が収集される. このような活動は組織メンバに対し, 組織の文化・政治を再考させるきっかけとなる.

このような診断によって, 組織の信念や戦略が人々の中でどのように理解され, 彼らの規範や価値観としてどのような形になっているかを分析することができる. 同時に, 組織全体の計画, 統制, コミュニケーション, あるいは人事等のシステムを見ることにより, それらと戦略との関係が理解できる. 診断から得られる情報を用いて, この組織の中で変革を行っていくに当たって必要となる戦術を形成することができる. また, 診断結果をトップに提示し, 変革で最も留意すべき点についての合意を得ることができる.

4.8 スポンサーを見つける

スポンサーは変革活動に必要な資源を確保してくれ,変革に脅威を感じている人々が持ち込む苦情のホコ先をかわしてくれる人物である. しかし, スポンサーは後援者 (メンター) とは異なる. 後援者は被後援者の出世や昇進も心配してくれるのに対し, スポンサーは変革の目的や方向性, それに伴う組織内の大きな活動を経営幹部や一般社員に説明する.

このような役割を期待されるスポンサーの任命は,人間軸における解凍活動の重要な一部をなす. それは組織内の既存の政治の絡みを解くための新しい手を打つことの表明となり, 新しい価値観を作る一歩であることをまわりに示すことになる. 多くの場合スポンサーは, SIS 構築の必要性や方向性を組織内に明示するシンボル的な人物となる.計画チームや内部コンサルタントは頻繁にスポンサーに接触し, 常に進捗を報告する必要がある.

スポンサーはその組織の最高の権限を持つ責任者 (すなわちトップ) であるのが理想的である. しかし, そのような人物がスポンサーとして得られない場合, あるいは得られたとしてもトップが細かな動きができないためにその代理のスポンサーが必要となる場合は, ピンチョー (1985) が主張するスポンサー像が参考になる.

第 1 に資源不足を解決してくれる人である. たとえば, 活動や人事や予算を決定したり承認を取り付けてくれる人であり, 有能なメンバを計画チームに連れて来る力のある人である. また, 情報システム計画の後続活動に資金を投資する際に, それを集めることのできる人である.

第 2 に地位が高く政治的な力を持っている人である.このようなスポンサーがついていれば, 攻撃を仕掛けようとする者も思いとどまるであろうし, 手ごわい相手に対しても防御し

てくれる．また，スポンサーの地位の高さや彼に対する尊敬が計画チームの地位の高さにもつながる．

第3に年齢が55～60歳位で，企業の重要な意思決定者や一般社員から尊敬されていて，彼らと接触するパイプを持っている人物である．また，現在の地位以上に出世しようという気を持っていない方が好ましく，個人的な野心(この情報システム計画活動で評価を上げようとする野心)からスポンサーになるのではなく，企業のために貢献したいという望みからスポンサーになる人の方が好ましい．

さらに，以上のようなスポンサー像に加えて，SIS時代におけるスポンサーは，情報技術についても理解し，トップ・マネジメント集団の中でも大きな発言力を持っていなければならない．今日，そのような人物はCIOとも呼ばれることがある．

4.9 運営委員会を設置する

運営委員会の最も良い例えは，企業の重役会である．運営委員会は，情報技術の全般的な計画と管理のために常設されている場合があるが，その役割はとくにSISの構築の場合に重要となる．歴史的に見ると，運営委員会が全社を横断する形で設置されるようになった理由は，情報技術が企業の競争優位を構築する戦略的な意味を持つようになり，情報システムの開発と管理の問題が全社的な重要性を持つようになったからである．コンピュータに関する権限が情報システム部長に集中していたMIS(経営情報システム)時代までは，運営委員会は必要なかった．

情報技術が分散化したEUC（エンド・ユーザ・コンピューティング）時代以後，その権限が組織全体に分散し，それを再度まとめるための手段として運営委員会が用いられるようになった．情報システム部門から利用部門側に移り，分散化しつつあった情報処理の計画・管理の権限を再度統合する形となっている．そのため，運営委員会はプロジェクト組織ではあるが，基本的に継続して存在すべきものである．運営委員会の本質的な機能は，情報システム開発の方向設定，開発プロジェクトの選別と資源割当，コンピュータの効果的な使用のための組織設計，人事，監査の五つであり(Nolan, 1982)，SISの構築と，とくに関係するのが情報システム開発の方向設定の機能である．

このような意味での運営委員会を結成し，トップや計画チームがそこにSISの構築の意向を伝えることは，現状の組織における情報システムに関連するすべての組織部門にその意向を伝え，それに関する全社レベルでの共通認識を推進することになる．つまり，運営委員会でSIS構築に関する議論をすることで，委員会のメンバたちはSIS構築の必要性と方向性の認識を深める事象をたくさん経験でき，構築のための背景や文脈を認識することになる．

SISの計画活動では，全社的な意思決定が求められるため，その承認機関として運営委員会が必要となる．運営委員会は情報システム計画を最終的に承認するため，トップや一般社員に受け入れられる人材で構成する必要がある．

4.10 計画チームを編成する

SISの実際の計画活動を担当するのが運営委員会からの委任で編成される計画チームである．計画チームは基本的にその場限りのプロジェクト組織であるが，その編成をすることで，情報システム計画の策定の意識が組織の中で大きく前進する．運営委員会を通じての人選によって，計画活動開始の具体的イメージが関係組織間に形成され，計画チームの人々の動きが計画活動の具体的な動きを代表するようになる．SIS構築の必要性を受けて具体的な

方向を目指した活動を担当する組織の設置となる．

したがって，計画チームの人選は非常に重要な意味を持っている．この人事で，一般社員の代表と認知されるようなSIS構築担当者を選ばねばならず，それによって新しい組織の動きが行われようとしているメッセージを一般社員に伝えなければならない．このような意味を持つ計画チームの人選には，次のようなガイドラインが考えられる．

1)　意見を持つ人……分析能力よりもむしろ意志，信念を持っている人間を集める．分析能力はコンサルタントによって提供されうる．

2)　現場で最も活発に仕事をしている人……現場で最も活発に働いている人間は問題を最も良くつかんでおり，かつその解決を望んでいる．彼はその職場の代表として認知されやすく，そのような人を選ぶことはプロジェクトが重要であることを回りの人々に知らせるのに役立つ．いわゆる「窓際族」のような人間をもってくると，このような文脈を作ることはできない．

3)　異質な意見を持つ人……異質な人を混ぜ合わせると多様な意見構成が可能となる．まわりの一般社員の多様性を反映しやすくなり，計画チームに参加できなかった一般社員の意見も代表しやすくなる．また，複数の連合から人選すると既存の政治色を薄められ，多くの人の賛同が得られる．

4)　トップ層の意向を知っている人……SISの計画活動は，トップの意向を具体的に実現する過程であるから，トップがそれについて具体的なイメージを持っている場合は，その意向をわかっている人を入れる．あるいは，トップが変革ニーズのみを持つ場合は，その変革ニーズを共有している人を入れる．

5)　いわゆるキーマンとよばれる人……現場において強い発言力を持ち，尊敬を集めているいわゆるキーマンを入れ，計画や実施における主要な役割を演じてもらう．これは，彼の忠告や意見を求めるためというよりも，彼のバックアップを求めるのである．

4.11　日程を決定する

SISの計画活動は全社に影響を及ぼすものであるので，当然，組織内の政治の対象となる．トップが意向を表明し，それを受けた形で運営委員会や計画チームを編成することは，SIS構築が必要であり，その方向へ向けて動き出したというメッセージを組織内に伝えることである．したがって，それに続けて，SIS構築の目的と具体的な方向が何であるかを一般社員に考えてもらい，組織内の政治的な連合が変化に対応できるように時間的な枠組みを与える必要がある．期日の設定は，人々に変化への準備を始めさせる間接的メッセージとなる．

その発表においては，計画の日程や目的，およびその範囲等を示して，政治日程が組めるように必要な情報を提供する必要がある．もしその種の情報が決定されず，発表もされないとすると，政治的な対応が行われないまま時間が経過し，必要性と方向性の文脈が組織内に形成されない．そして結果的に，策定された計画だけが事務的に孤立的に発表され，既存の連合や新しい連合のコミットメントを得られないまま宙に浮いてしまうことになる．組織文化の変革プロセスにおいては，アクションのないエンドレスな議論になりがちであり，そのため，行動を起こせるような明確な役割，特定の目標，そして期日を設けることが極めて重要となる（Davis, 1984）．

計画活動においては，期日の発表だけでなく，期間の長さも重要である．技術軸の計画活動だけを採り上げたとき，人々は短期間で行うことを理想のように考えていることが多い．前にも述べたが，技術軸の活動は人間軸の活動に織り合わされて実行されねばならない．情

報システム計画案だけが短期間でできてしまうことは，それが回りに受け入れられず，宙に浮いてしまうことを意味する．すなわち，SIS の計画活動においては，短期間に行うという効率主義は他の損失を招くことになりかねないことに注意しなければならない．

4.12 計画活動を広報する

計画日程の発表ばかりでなく，情報システム計画活動の進行状況についても，機会あるごとに各種のメディアで組織内に伝えていくことが極めて重要である．計画活動の進行を追いながらさまざまなことを伝えていくことで，一般社員をカヤの外において活動を進めているのではないという文脈を組織内に作っていかなければならない．同時に，SIS 構築の必要性と方向性についての理解を促進する具体的な情報を与えていくことにもなる．人間には，活動の状況が逐次知らされていれば，その問題へのコミットメントを高くするという性質がある．進行中の情報を入手し，かつ自分の代表が計画活動を行っているとすれば，それに対する発言の権利と自由度を保っているように認識する．逆に情報が得られないと，意図的に排除されているという認識を持ち，形を変えた抵抗を示すようになる．

このような広報活動は，儀式的な活動の形式をとる場合がある．とくにトップがそのような活動を行う時にはその色彩が強くなり，一般社員を排除して計画活動を進めているのではないというメッセージを伝える．そしてそれ以上に，この計画活動が会社にとって極めて重要であることの表明となり，全社的な協力を要請し，むしろ公然とそれを強制することにすらなる．

5. 移　　動

移動のステップは，解凍によって必要性と方向性の文脈が形成された組織の中で，SIS の計画を具体的に策定していく段階である．ここで注意すべき点は，解凍のステップが明確に終了してから移動のステップに入るというものではないということである．概念的にこれら二つのステップに分けておく方が理解しやすいが，実際の活動では二つのステップとしてくっきり分離できるものではなく，多少の重なりを持って進行するものである．移動の活動を進めつつ解凍の活動をさらに行っていくこともあり得るし，行きつ戻りつ解凍と移動を進めていく場合もある．

しかも，具体的な情報システム計画を作る作業は極めて技術的なもので，その内容は「技術軸」のテーマとなる．市場が求める価値を創造する新しい価値活動の連鎖としての業務を再編成し，それを古い情報システムを再構築した SIS に組み込んでいかなければならない．このような技術軸での具体的作業を担当するのが計画チームであり，内外のコンサルタントである．本章では，文脈情報を形成する活動としての人間軸の活動ではなく，彼らの効果的なグループ活動にとって必要な人間的な配慮としての活動について述べる．

5.1 チームワークづくり

計画チームは情報システム計画を作るという共通の目標を持ち，その達成のために，グループメンバ全員の協調的な相互活動ができるようなチームワークが必要となる．効果的なチームワークを作るためには，メンバ相互の良好なコミュニケーションのメカニズムを開発し，対立や衝突を建設的に解決する方法を学習しなければならない．チームメンバの一人一人が対人関係や集団のダイナミックスを分析し，変革するスキルを開発できるようにコンサルタントが援助する必要がある．

一般に，効果的なチームワークを作るために次の四つのステップを踏む必要がある（バー

グ，1987)．

1) チームの明確な共通目標を設定する．

2) その目標に基づきチームメンバの役割を設定する．

3) 目標と役割が確立されたならば，チームはコミュニケーション，問題解決，意思決定，対立管理等のプロセスを開発する．

4) チームの作業環境（たとえば，部屋の配置や作業時間帯）によってもたらされるメンバ間の問題を扱う能力を開発する．

これら四つのステップはこの順で行う必要があり，一般に番号の若いステップが不十分であると，後のステップでのコミュニケーションの質が低下する．目標があいまいなままグループ活動が進行して問題が生じると，その解決に必要な共通理解の基盤がないため，問題解決のコミュニケーションが進展しなくなってしまうからである．

なお，職場から離れた場所（オフサイト）で合宿して行うと，チームワーク作りの活動が効果的になる場合が多い．日常の職場環境から離れて活動することで既存の規範，価値観，パワーから解放され，自由な発想が促進され，メンバ同士の共通理解が深まる．

5.2 プロセス・コンサルテーション

このようなチームワークづくりを促進する方法として，プロセス・コンサルテーションの方法が使用できる（バーグ，1987)．つまり，計画チームが行うグループ活動の「手続き」と「維持」のプロセスについて援助するものである．手続きのプロセスは，グループの役割分担，リーダーシップのとり方，意思決定の方法，問題解決の進め方等である．維持のプロセスは，グループをチームとして束ねてその凝縮性を高め，分裂しないように維持する活動である．

「手続きのプロセス」を援助するために，コンサルタントは目標設定の会議をデザインしたり，あるいはどのような意思決定の様式に従って決定を下すかをアドバイスする．職場を離れて合宿する方法も手続きのプロセスを援助する一つの方法である．

一方，「維持のプロセス」を援助するために，コミュニケーションが開放的で，相互信頼のあるものになるように，メンバが参加できる機会を多く作ったり，問題点の整理や締めくくりをして，意見の相違に対する緊張を和らげる方法を採ったりする．また，組織に必要な新しい活動を描くために，組織内部の問題に原因を求めるのではなく，外部環境の要請から考えていく方法（オープン・システムズ・プランニング）も用いられる．この方法は意見の相違に対する不必要な緊張を和らげるのに役立つ．

このようなプロセス・コンサルテーションを施すことにより，普段は高い垣根があるために共同で作業することのない人々の間に，質の高いコミュニケーションを起こすことが可能となる．

6. 再　凍　結

移動のステップで策定されたSISの具体的な計画が，再凍結のステップにおいて実施に移される．この計画は特定の情報システムを構築するという極めて具体的なものである．そのように特定化されたものは解凍のステップにおいては存在しなかった．

再凍結のステップでは，具体化された計画を物理的な存在に変換するのに必要な下地を準備しなければならない．下地の準備としては，策定された計画の実施に必要な組織の文脈として，「計画されたそのSISには実効性がある」という共通認識を組織内に形成する．これ

を行っていく方法の原理は，解凍のステップの場合とまったく同じである．実効性という文脈情報のリレーショナル認識を促進するために，組織内に多くの活動を展開してSISに実効性が期待できるという事象情報を人々にたくさん与えていく．

移動と再凍結のステップの分け方も，概念的な理解のしやすさのためのものであって，解凍の場合と同じように多少の重なり合いをもっているのが実際である．移動のステップでの具体的な（技術的な）計画活動を進めながら，再凍結の活動がスタートしていく．

6.1 トップへ報告する

SISの計画ができたなら(場合によってはそのドラフトができた段階で)，トップへ報告する．計画チームとして最終的な意思決定を行う前に，その方向性がトップの考えと合っているかどうかを確認し，トップが抱く変革のイメージと計画チームが作成したものとの間の最終的な調整をする．トップが具体的なイメージを持っていない場合は，その内容を誰よりも早く彼に報告することが必要である．

実効性の文脈を組織内に形成しなくてはならない再凍結のステップで，トップへの報告は極めて重要な活動となる．それは，計画の実施のために彼のサポートが不可欠であり，彼がその情報システムの実効性を熟考し，変革の具体的イメージを自分のものとする時間的な余裕を与える必要があるからである．しかも，トップのサポートが得られるまではこれ以上先に進むべきではない．トップのサポートのない計画案は，一般社員の間で真剣に採り上げられることがないからである．

また，トップへ報告することによって，計画に対するコミットメントを取りつけることができる．とくに，経営資源の配分に関する彼の同意を取りつけておくことは，一般社員に変革へのトップの取り組み姿勢をアピールする上で極めて重要である．

6.2 運営委員会へ報告する

運営委員会は，計画チームによって提案された情報システム計画を公式に承認する組織である．運営委員会への報告では，計画された特定の情報システムの経営戦略的な実効性についてさまざまな側面からの情報が提供され，そこでの検討は組織を代表した者による検討となる．計画案に問題がなければ，会社の各部門を代表する者としての意思決定がなされ，実効性を組織で承認した形としてアピールされる．同時に情報システム計画の実施が，組織全体の課題として承認されることになる．

運営委員会でのこのような検討と承認のプロセスは，そのメンバが，計画案から新しい文化と政治の可能性を読み取り，それに対処できるようにするための情報を入手していくプロセスである．それらの情報を基礎に，従来のパワー関係を変更して新しいパワー関係を作っていく．またメンバ同志が意見を相互に表明することを通して，運営委員会としてのコミットメントを形成し，意思決定を受け入れる文脈が形成される．

6.3 中間発表をする

一般社員にとって，新しい情報システムがどのようなものになるかについて絶えず報告を受けていることは，彼らがその情報システムの実効性を知り，それに対するコミットメントを持つようになるために重要である．

「自分たちは情報を受けている」という認識を与えることが必要であり，情報システム計画の進行状況を彼らに中間発表していかなければならない．その形式としては，社内報や発表会という場合もあるし，トップの参加を得た儀式的な形にすることもできる．

一方，一般社員は，これらの公式情報と人的ネットワークで伝わってくる非公式情報をも

とに，そのSISが稼働した後の新しい組織内のパワーの形を各自が想定できるようになる．彼らは，そのような想定にもとづいて，新しい連合に対応しようとする行動を取り始めるであろう．

6.4 組織を編成する/プロジェクトを結成する

公式に承認されたものとしての計画案を一般社員に発表する前に，具体的なシステム開発のための組織やプロジェクトを最低一つは結成しておくことが重要である．それは計画発表と同時に新しい担当部門やシステム開発プロジェクトとして発足させるものであり，会社の方針としてのSISの構築を明示するための一種のシンボルとなる．

同時に，SISが稼働した時の新しい組織を一部分でもよいから発足させ，その責任者とメンバを発表する．この新組織と人事異動は，SISの稼働による実効性を会社が期待していることを示すシンボルとして機能する．このようなシンボルがなければ，計画発表をしても，一つのお話として組織の中で受け流されるであろう．奥村 (1986) は，変革の最後に組織構造と人事の変革を伴わせることで「新しい意味の体系」を組織内に共有させられるとしており，組織構造は人々に安定した関係パターンを作り上げ，人事は人々に変革の完了を伝達する．

とくに，発足させる開発組織やプロジェクトには，企業内で注目を浴びている人を割り当てるべきである．重要人物を割り当てることは，会社の方針の重要性を示すメッセージとなる．

6.5 委 譲 す る

SISの構築で狙う組織の変革が大きい場合は，計画チームだけで，新しく求められる組織構造，業務機能のやり方，社内外コミュニケーションのやり方，評価方式や報酬制度を詳細化していくことがむずかしい場合がある．そのような場合は，より実務に近い下部組織やその分野に詳しい外部組織に権限を委譲し，より多くの人を巻き込んで，組織の構造や制度の細部の検討をすることになる．この種の作業を組織の下位層にまで及んで行うことで，SISの実効性に関する情報を一般社員にまで広めることができる．これを怠ると，実現可能性や具体性が一般社員に見えてこないことになる．

6.6 発 表 す る

再凍結のステップのピークは，意思決定したSISの計画を一般社員に発表することである．しかし，一般社員は計画活動に最も遠い位置にあり，変革への認識は他の人間に較べて遅れやすい．このような一般社員に，計画されたSISの稼働による経営的な実効性を認識させるには，ただ計画案を発表するだけでは不十分である．SISの構築によって会社が組織の変革に本気で取り組んでいるとの印象を強くする必要があり，計画発表を次のように配慮して行う必要がある．

まず，発表を重要な儀式とする．SISの構築は組織文化の変革であることを組織全体に伝えるために，トップやスポンサーを動員し，会社が生まれ変わるために重要な儀式として発表を演出する．次に，開発プロジェクトを同時に発足させることである．この点はすでに述べたが，計画と実行を同時に発表することにより，会社の取り組み姿勢の強さを示す．これによって，後戻りできないというメッセージを一般社員に与えることとなる．さらに，これもすでに指摘したことだが，シンボリックな人事を発表して，開発組織やプロジェクトのリーダやメンバに極めて重要な人材を配置する．このことにより，組織内の政治が変わることを明示する．

また，後続プロジェクトの発足や新しい組織の発表と同時に，使命を終えた計画チームの解散を発表し，変革が実際のものになったことを一般社員に伝えていく．

執筆者紹介　小　坂　　武（Takeshi Kosaka）

昭和45年上智大学理工学部電気電子工学科卒業．46年日本ユニシス㈱入社．MIS，DSSの開発，構築支援を行う．61年より，SIS構築のための戦略コンサルティングに携わる．この間，慶應義塾大学大学院経営管理研究科（ビジネス・スクール）に派遣され，経営学修士（MBA）終了．著書・論文に「意思決定支援システム」（竹内書店新社，昭和58年），「SIS—経営革新を支える情報技術」（日本経済新聞社，平成2年），"An Effective Architecture for Decision Support Systems", Information & Management (North-Holland, 1982)他がある．日本経営協会より著書「意思決定支援システム」で経営科学文献賞を受賞．現在，愛知学院大学経営学部助教授．特種情報処理技術者，オペレーションズ・リサーチ学会会員．

世界最大規模の超大型汎用コンピュータ UNISYS 2200/900 シリーズ

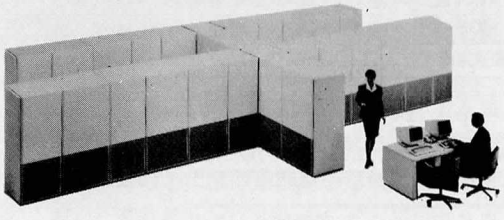
日本ユニシスは，90年代の戦略的企業ネットワーク情報システムに求められるさまざまな要件を包括的に満たしたインフォメーション・ハブを担う新世代メインフレームとして，世界最大規模の超大型汎用コンピュータ「UNISYS 2200/900 シリーズ」(14モデル) を世界に先駆けて発表した．

ユニシスは90年代の情報処理環境を「インフォメーション・ネットワーク」と考えている．インフォメーション・ネットワークとは，〝異機種システムを含むマルチシステム環境における巨大なシステム資源をあたかも単一のシステムであるかのように自在に活用できる〟新しい世界である．過日発表のUA（ユニシス・アーキテクチャ）では，その中枢を担っていくメインフレームを「インフォメーション・ハブ」と位置付けた．

ユニシスは90年代のメインフレームに求められる役割を，①大規模な基幹システムを支える高信頼性と十分な処理能力，②より高度なレベルでの分散処理を統合化し管理する能力，③企業戦略を支え環境に柔軟に対応しうる使いやすさ，を追求したシステムと考え，新世代メインフレームの役割を次に示す八つの機能（インフォメーション・ハブ）に集約した．

1) 大規模トランザクション処理
2) 限りないシステムの成長性
3) 無停止連続処理
4) オープンな相互接続性/相互運用性
5) 先進的なデータベース・システム
6) 高生産性アプリケーション開発/実行環境
7) システムの無人運転/統合管理
8) 高度なセキュリティ

UNISYS 2200/900 シリーズは，90年代のメインフレーム・コンセプト「インフォメーション・ハブ」にもとづいて開発されたシステムであり，今後もこのコンセプトのもとに，積極的な製品の開発・提供が行われる．



UNISYS 2200/900 シリーズモデル 9848

1. 拡張処理アーキテクチャ「2200/XPA」

拡張処理アーキテクチャ「2200/XPA (eXtended Processing Architecture)」は，インフォメーション・ハブ・コンセプトを実現するためのアーキテクチャであり，「XCPA」，「XTPA」，「XIOA」の三つのサブ・アーキテクチャから構成される（図1）．

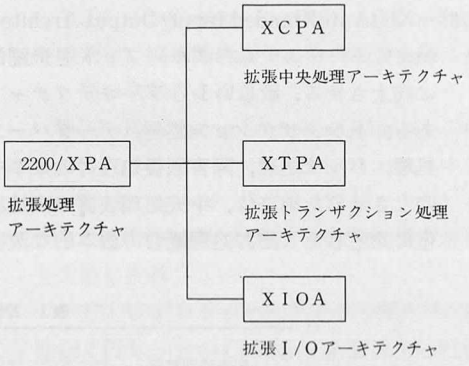


図1 2200/XPA を支えるサブ・アーキテクチャ

1) XCPA (eXtended Central Processing Architecture) ……拡張マルチ・プロセッサ・システム（プロセッサ8台のマルチ・プロセッサ・システム）と巨大アドレス空間，大容量記憶装置，高度なセキュリティ，無人運転，集中管理機能等を支援する中央処理系のサブ・アーキテクチャである．

XCPAのもとに開発された「2200/900」シリーズでは，巨大なシステム空間72P(ペタ*)バイト(当初560T(テラ*)バイトから提供開始)，276G(ギガ*)バイトのプログラム・アドレス空間，そして2G(ギガ)バイトの大

* ギガは10の9乗，テラは10の12乗，ペタは10の15乗

容量記憶を実現している．

これにより8台までの高速な拡張マルチ・プロセッサと連動して，データベースや使用頻度の高いファイル，またソート作業領域をサイバー空間に展開したり，会話型処理やバッチ処理の作業領域として利用することで，処理効率を飛躍的に高めている．

さらに入出力処理能力も大幅に向上し，最大96のI/Oプロセッサにより480チャネルが使用可能である．

2) XTPA (eXtended Transaction Processing Architecture) ……ユニシス独自の実績のあるXTPAがさらに充実して，最大32台のプロセッサ群がシングル・システムとして稼働し本格的な無停止運転が可能となる．

2200/900シリーズ内で25倍以上の拡張性があり，『インフォメーション・ハブ』に要求される能力に応じて，柔軟にシステムの拡張ができる．

3) XIOA (eXtended Input/Output Architecture) ……システムのスループットを飛躍的に向上させる，新しいI/Oアーキテクチャである．トランザクション処理，データベース処理，バッチ処理，障害回復処理等の効率を向上させるためには，中央処理装置の高性能化に対応した入出力処理能力の根本的な改善が必要である．

XIOAは，これに応えるアーキテクチャであり，基本設計思想は複数ホスト環境を前提としたインテリジェントI/O機構の実現にある．

第一段階として，現在磁気ディスク装置に対して行われている入出力処理を，中央処理装置の一部に取り込み，これによって中央処理装置と磁気ディスク装置の機械的なアクセス・メカニズムが切り離され，入出力処理時間が大幅に短縮される．

2. 2200/900シリーズのプロダクト概要

1) システム概要……システム能力は，2200/600シリーズの2倍以上で，最小構成のモデル9111から最大構成のモデル9848まで14モデルあり，業務の拡大に対応し設置場所において柔軟に拡張ができる（表1，2）．

システム操作，システム制御，保守診断機能を統合した「SCF (System Control Facility)」と無人化運転を支援する「SFCP (System Facility Control Processor)」が標準機能として提供される．

「SCF」は，オペレーティング・システムとハードウェアの独立性を図る「HIS (Hardware Indipendent Software)」機能の思想を

表1 2200/900シリーズの性能

中央処理装置 (IP)	命令数		264種
	IP数		1～8
	キャッシュ・メモリ容量		64Kバイト/IP
ストレージ制御装置 (SC)	台　数		1～4
	キャッシュ・メモリ容量		1024Kバイト/SC
主記憶装置 (MSU)	台　数		1～8
	容　量		256Mバイト～2Gバイト
	増設単位		256Mバイト
I/Oチャネル	最大チャネル数		480チャネル
	最大IOP数		96IOP
	ブロック多重チャネル (BMC)	チャネル数/IOP	4
		転送速度	3.1Mバイト/秒
	ワード・チャネル (WDC)	チャネル数/IOP	8
		転送速度	3.7Mバイト/秒
	IOP数/ICC*		12
システム・コンソール			PW²800

*ICC当たりのワード・チャネルIOPは最大3個

表2　2200/900シリーズのシステム構成

システム・モデル	中央処理装置(IP)	ストレージ制御装置(SC)	入出力		主記憶容量(Mバイト)	オペレータコンソール	システムサポートプロセッサ	プロセッサ冷却装置	チラー装置(オプション)	電源供給装置(オプション)
			入出力装置(ICC)	入出力プロセッサ(IOP)						
モデル9111	1	1	1	12	256～512	2～11	1	1	1	1
モデル9211	2	1	1	12	256～512	2～11	1	1	1	1
モデル9212	2	1	2	24	256～512	2～11	1	1	1	1
モデル9222	2	2	2	24	512～1024	2～11	1	2	2	1
モデル9322	3	2	2	24	512～1024	2～11	1	2	2	1
モデル9323	3	2	3	36	512～1024	2～11	1	2	2	1
モデル9333	3	3	3	36	768～1536	2～11	1	3	3	1
モデル9422	4	2	2	24	512～1024	2～11	1	2	2	1
モデル9424	4	2	4	48	512～1024	2～11	1	2	2	1
モデル9444	4	4	4	48	1024～2048	2～11	1	4	4	2
モデル9633	6	3	3	36	768～1536	2～11	1	3	3	2
モデル9636	6	3	6	72	768～1536	2～11	1	3	3	2
モデル9844	8	4	4	48	1024～2048	2～11	1	4	4	2
モデル9848	8	4	8	96	1024～2048	2～11	1	4	4	2

実現し，システム構成に変更があってもシステムの再生成を必要としない．

また基本モードと拡張モードの両モードが提供されるので，現行のユーザ資産は，そのまま2200/900シリーズで使用できる．

さらにXTPAの導入で，32台のプロセッサがデータベースを共用しつつ，シングル・システムとして運用できる．

これによりシステムの拡張性は，25倍以上にも拡大し，いかなる業務の増加にも余裕をもって最小の投資で対応が可能である．

こうした複数のシステムをシングル・ポイントで操作可能にする機能「SPO (Single Point Operation)」を提供する．

2)　2200/XPAを支える最新のテクノロジ……プロセッサ部分の論理素子には，1万ゲート・レベルの超高速ECL(Emitter Coupled Logic)と，10個の高速SRAM(Static RAM)を両面実装したMAM(Multi-Array Module)を採用し，実装密度を飛躍的に増大させている．これにより素子間の伝播時間を大幅に短縮した．

さらに50層からなる高密度プリント基盤DPCB (Dense Printed Circuit Board) を開発し，この上に高速論理素子を最大210個搭載した高性能シングル・ボード・プロセッサを実現した．

この高密度プリント基盤を効率よく冷却するILDC (Individual Logic Direct Cooling) を開発し，高速論理素子の接合温度を低く抑え，高い信頼性を保証している．

PCC (Processing Complex Cabinet) の中には，2台のプロセッサ (IP)，1台のストレージ・コントローラ (SC)，512 M (メガ) バイトの主記憶装置 (MSU) を収納し設置面積を大幅に改善している．

一方I/Oプロセッサとチャネル部には，高性能4万ゲートのCMOSを全面採用し性能向上と大幅な経済性の改善を図った．

ICC (I/O Complex Cabinet) には，最大12台のI/Oプロセッサと48チャネルが収納でき，省エネ・省スペースを図っている．

3)　高信頼技術の採用……「インフォメーション・ハブ」に要求される高信頼性に応えるため，以下のような技術を採用している．

- 障害の拡大を防ぎ，リアルタイムに障害を検出し，同時に障害個所を特定する障害検出機能の充実
- 間欠障害を自動的に回復する，自動訂正，自動再試行機能
- 固定障害部分を自動的に切り離し運転を継続させる自動縮退機能
- オペレーティング・システムとSCFが連動して障害の重要度を判断し，最適な回

復方法を指示，実行する自動回復機能
・IPの重大障害の場合に障害の状態を診断し，障害直前の正常な状態を他のIPに復元し，システムを止めずに動作させるトランスプラント機能
・オペレーティング・システムのサブ・システム化により，ソフトウェアの障害を局所化し回復する機能

3. 2200/XPAを支援するオペレーティング・システム

2200/900シリーズは，付加価値を高めて豊富に蓄積されてきたシリーズ2200・1100のソフトウェア資産を継承しつつ，2200/XPAアーキテクチャのもとで，今後の「インフォメーション・ハブ」に求められるより大規模なトランザクション処理/データベース処理，無停止連続処理支援のためにオペレーティング・システムを，以下に示すように一段と強化・拡充した．

1) 記憶管理の飛躍的拡張
・2G（ギガ）バイトの大記憶容量，276G（ギガ）バイトのプログラム・アドレス空間，72P(ペタ)バイトのサイバー空間，バンクド・ページング機構の採用

2) 処理効率の向上
・メモリ・ファイル機能により，使用頻度の高いファイルをサイバー空間に展開し，処理効率を向上
・ソートの作業領域として巨大アドレス空間を使用し，バッチ処理や会話型処理の高速化を実現

3) 大規模スケーリングの実現
・8台までの密結合マルチ・プロセッサ，480個のI/Oチャネル，32台までのクロースリ・カップリング・マルチ・プロセッサの支援

4) セキュリティ機能の強化
・データやプログラムを不正なアクセスから保護するリング/ドメインによる管理．ドメインの数を従来の500倍の65,000個まで拡張し，超大規模システムに対応
・米国NCSC（National Computer Security Center）から，ビジネス用では最高レベルの認定を受け，高度なセキュリティを実現

5) 信頼性の向上
・データ・ストラクチャごとの専用バンク化とドメイン保護機能による障害の局所化と障害モジュールの動的置換
・オペレーティング・システム自身の機能のサブシステム化により，障害の局所化と動的置換，SCFとの連動による障害回復機能の向上

6) XTPAの拡張
・32台のプロセッサを支援
・汎用データベース「UDS（Universal Data System）」の提供によるネットワーク型データベースやリレーショナル型データベースの使用可等，利用範囲の向上

7) HIS（Hardware Indipendent Software）コンセプトの拡充
・ハードウェアとソフトウェアの独立性を高め，ハードウェアの変更によるソフトウェアの変更を極小化

4. インフォメーション・ハブを実現するソフトウェア・プロダクト

1) 容易な大規模トランザクション処理の構築
「XIS」……「XIS（eXtended Information System）」は，オンライン・トランザクション処理システムのプラットフォーム構築を，開発から運用・保守に至るまで総合的に支援するソフトウェア群である．

XISの特徴は以下の通りである．
・XTPAとの連携
・多様な処理形態の提供
・高水準アプリケーション・インタフェースの実現
・統合開発環境支援システム（IDES）との連携
・統合運用管理システム（IOF）との連携

2) 完全なノンストップ・システム「XTC-TIP/UDS」……XTPAは，大規模システムにおける完全な無停止連続処理を実現する．ホット・スタンバイ・システムのように，本番系/待機系に分けることなく，すべてのホストが本番系であり，万一，どれかのホストに障害が発生した場合でも，切り替え時間ゼロで他のホストが処理を肩代わりするため，処理は寸断することなく継続する．

同機能を支援するハードウェアが，RLP

(Record Lock Processor) であり，ソフトウェアが「XTC-TIP/UDS」である．

現在すでに16台までのプロセッサがXTC-TIPのもとで稼働可能であるが，2200/900シリーズでは，32台までのプロセッサをシングル・システムとして運用できる．適用可能なデータベースは，TIPファイルに加えUDSデータベースへと，さらに拡大する．

RLPには，TMR(Triple Modular Redundancy＝三重化設計）と呼ばれる超高信頼フォールト・トレラント技術が導入されている．

3) オープンな相互接続性/相互運用性「UNIDSS」……「UNIDSS」は，ユニシス・アーキテクチャの分散処理サービス（DSS）を支援するソフトウェアで，OSI基本参照モデルの上位層，すなわちセッション層からアプリケーション層に対応する分散システム・サービスを提供する．

「UNIDSS」のもとでは，DCA, OSI, SNA, TCP/IPを支援し，それぞれホスト/部門サーバ/ワークステーション相互間でファイル転送，ジョブ転送，トランザクション転送，プログラム間通信等を可能としている．

① DCA支援

シリーズ2200・1100相互間，シリーズ2200•1100とビジネスUNIXシステム「Uシリーズ」間でのファイル転送，ジョブ転送，プログラム間通信を行う「DDP/PPC」，「DDP/FJT」を提供する．

統合オンラインシステム「XIS」の環境下ではシリーズ2200•1100間のトランザクション転送，BOSS/ICCPゲートウェイ，分散データアクセス，プログラム間通信を行う「ASCOT 1100」を提供する．

さらにリレーショナル・データベースをアクセスするための「ASCOT 1100/RDAF」を提供する．

② OSI支援

マルチベンダ環境下で，OSI準拠の次のような分散処理機能を提供する．

OSI-FTAM……ファイル転送機能
OSI-MHS……電子メール機能，「MAIL 1100」と結合が可能
OSI-CSF……トランザクション処理
OSI-OCSF……トランザクション処理プロトコルの構築支援機能

さらに分散トランザクション処理を支援する「OSI-DTP」，ディレクトリ（CCITT X.500）機能を提供する「OSI-DIR」を今後提供する予定．

③ SNA支援

ユニシス・ホストとSNAホストの間で，双方のソフト/データ資産の相互活用を図るSNA*ゲートウェイ機能，「SNA LU 6.2」機能を提供する．

④ TCP/IP支援

イーサネット**を介して接続された異機種，マルチ・ベンダ環境下での相互運用性を実現する．

DDN 1100……ファイル転送機能，プログラム間通信機能
SQL＊Star***……SQLインタフェースに基づく分散リレーショナル・データベースの構築機能（計画中）
NFS 1100……シリーズ2200•1100と「Uシリーズ」の間で分散ファイル・アクセス機能（計画中）

4) 高生産性アプリケーション開発・実行環境「4 GL/CASE」，「IDES」……CASE（Computer Aided Software Engineering）ツールは，システム開発の各工程で人間の意思をシステムに取り入れる方法を限りなく容易にし，最終的にはシステム開発の自動化を目指している．

とくに昨今は，開発の上流工程を支援する上流CASEの必要性が叫ばれ，下流工程との整合性のある統合CASEへと発展しつつある．

①リポジトリ

CASEツールを使ってアプリケーションを作成する場合，運用・保守までも含めたライフサイクル全般を視野に入れ，開発・運用・保守に関するすべての情報をデータ・ディクショナリに一元的に管理する必要性が生じる．

データ・ディクショナリは，単なるデータに関する辞書という役割を越え，アプリ

---

* SNAは米国IBM社の登録商標である．
** イーサネットは米国XEROX社の登録商標である．
*** SQL＊StarはORACLE社の登録商標である．

ケーション構築のための総合的な情報貯蔵庫（リポジトリ）に進化していく．

② CASEツールとリポジトリ

上流CASEツールは，ワークステーション上で要求分析・定義や基本設計をグラフィカルに支援するツールである．

作成された設計情報をリポジトリに入力できるようになれば，そこからプログラムを自動生成したり，設計・運用に関する文書を出力させることができる．

すなわちフォワード・エンジニアリングを可能とする．また，逆にリポジトリ内の開発情報を上流CASEに取り込み，設計情報をグラフィカルに再現するリバース・エンジニアリングも可能となる．

すでに3GLのCASEツールとして提供してきた「TSX 1100 II」に代わり，より高度のCASEテクノロジを採用した分散開発環境を提供する新しいプロダクト(IDES)を提供する．

4GLの世界でも，リポジトリを中心に上流CASEを充実させ，従来のLINC, MAPPERと併せ，統合CASEとしての提供を計画中である．

5) 先進的なデータベース・システム「UDS」……UDSは，一つのプログラムから複数のデータ・モデルのデータベースを同時に使用できる汎用データベース・システムである．

このUDSの大規模トランザクション処理/データベース処理，さらにアプリケーション開発支援等へのより強力な対応を図る一方，分散リレーショナル・データベース機能を一段と強化した．

① リポジトリの採用「UREP」

アプリケーション開発関連情報の一元管理，DMS, RDMS, SFS等の複数のデータベースに関する情報の一元管理，開発ツール間での自由なデータの受け渡し，ファイルの物理的属性の動的変更等を支援するため，従来のデータ・ディクショナリの機能を一層発展させた．

② XTPAへの適用「XTC-TIP/UDS」

XTPA環境のもとでUDSが利用できる．これにより従来のTIPファイルをはじめ，DMS, RDMS等さまざまなデータベースの混在した大規模ノンストップ・システムが構築できる．

③ 分散リレーショナル・データベースの実現

統合オンライン支援システムXISの環境下でDCAのネットワークを使用した分散データベース機能，SQL * Starの環境下でTCP/IPのネットワークを使用したRDMSによる分散データベース機能，MAPPERのリレーショナル・データベース・インタフェース（MRI）を使用したUDSとMAPPERによる分散データベース機能を提供する．

6) 容易な統合運用・管理システム「IOF」……システムの巨大化・分散化そして多様化・複雑化に伴い，容易なシステム管理，運用管理の要求は，ますます増大しつつある．

「2200/900シリーズ」ではこうした要求に応えるために，総合運用システム「IOF（Integrated Operation Facility）」を提供する．

「IOF」は，システム監視制御，自動・無人運転，ワーク・フロー制御，システム・リソース管理，システムの運用管理に求められるさまざまな機能を一つの体系に統合，さらにシングル・システムからネットワークされた複数システムの運用管理までを一貫して支援するシステムである．

「IOF」の特徴は以下の通りである．

- 統合化された汎用の運用管理
- 一貫性をもった容易な操作
- 効率のよい集中監視制御と自動・無人運転の提供
- 新しい自動化機器への対応

7) 統合運用管理を支える新しいシステム・コントロール機能「SPO」……「2200/900シリーズ」では「2200/XPA」のもとに統合運用管理を支援するさまざまな機能を提供する「SCF」と「SPO（Single Point Operation）」を用意する．

SCFでは，ハードウェア・リソース管理のためにオペレーティング・システムの必要とする情報をすべて収集，蓄積する．

この情報は，システムの初期化時にオペレーティング・システムに渡され，ハードウェアの機器構成に変更があった場合でもシステ

ムの再生成が不要となる．

さらにLANを介して複数のシステムを集中監視制御する「SPO」を提供する．

すなわちLAN上の各SCFから可能な操作は，「SPO」からもすべて行える．大型のスクリーン上でグラフィカルに監視制御が可能となる．

統合開発環境支援システム (IDES) Integrated Development Environment support System

IDESは，データ・ディクショナリを中心とする開発環境の構築を支援するとともに，一連の開発工程（図1）の作業を支援するツール群によって，システム開発・保守の生産性と品質の向上を実現するシリーズ2200・1100用の3GL CASEツールである．

1. IDESの特徴

1) 階層化分散開発環境の推進……UA（ユニシス・アーキテクチャ）のフレームワークに準じたメインフレーム，サーバ，ワークステーションで構成される3階層に分散した開発環境（図2）によりホスト負荷分散が可能となり，安定した応答とホストの運用時間帯にとらわれない柔軟な開発環境が実現できる．

また大規模なシステム開発においては，開発センタの分散化により要員を1か所に集中させることなく，柔軟な要員配置が可能となる．

2) ユーザ・インタフェースの統一と高度化……ワークステーションの活用による安定した応答と統一されたユーザ・インタフェースによる，思考の中断の少ない高い操作性の実現によって効率の良い開発が行える．

3) データ・ディクショナリによる情報の統合化……データ・ディクショナリにより，一連の開発工程（分析/基本設計/詳細設計/プログラミング/テスト/運用・保守）の情報を一元管理し，工程間の連携を強化している．

また実行環境（XIS），運用環境（IOF）との情報の共有化により，システムのライフサイクル全般に亘る開発・保守の効率化が可能となる．

2. 機能概要

IDESは3階層に分散した開発環境の構築を支援するために，以下に示す機能を持っている（図3，表1）．

1) 設計支援……システムの入出力を構成する下記項目を一元的に定義することを支援し，データ・ディクショナリを構築する．

—データ項目定義
—レコード定義
—ファイル定義
—帳票定義・フォーム定義
—画面・帳票定義
—データベース定義
—システム構成定義

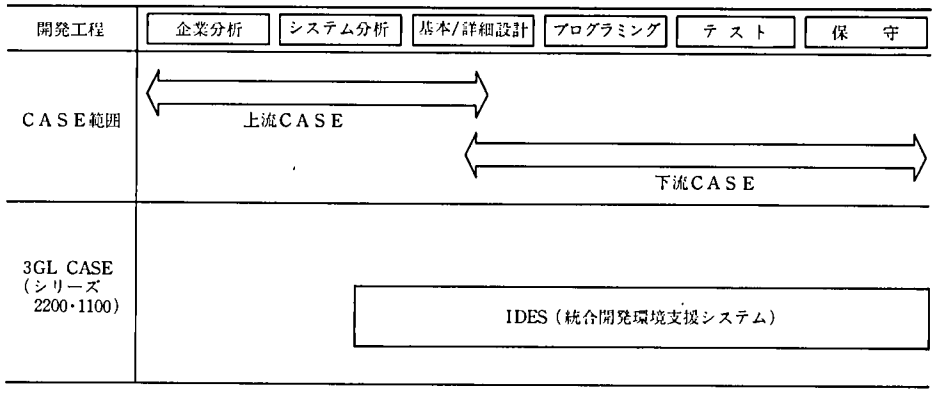


図1　サポート開発工程

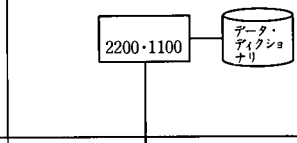
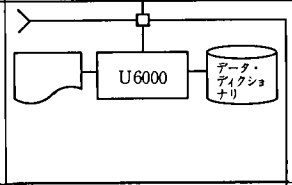
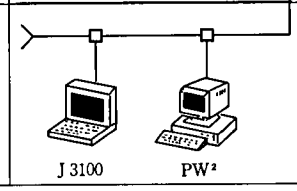
	システム構成	環　境	機能分担	機能概要
メインフレーム		□ 実行環境 □ 結合テスト環境 □ 単体テスト環境	□ 全体管理 □ リポジトリ	□ 設計支援 □ プログラム作成支援 □ 部品化・再利用 □ 単体テスト支援 □ 結合テスト支援 □ プログラム管理支援 □ プロジェクト管理 □ データ・ディクショナリ
サーバ		□ 検査環境 □ 開発環境	□ 部門管理 □ プリント・サーバ □ ファイル・サーバ □ プロセス・サーバ	□ 文書管理 □ 設計支援 □ プログラム作成支援 □ 分析設計支援
ワークステーション		□ 操作環境	□ メニュ制御 □ ウインドゥ制御 □ 入／出力制御 □ 編集	□ 文書管理 □ 設計支援 □ プログラム作成支援

図2　開発環境と機能分担

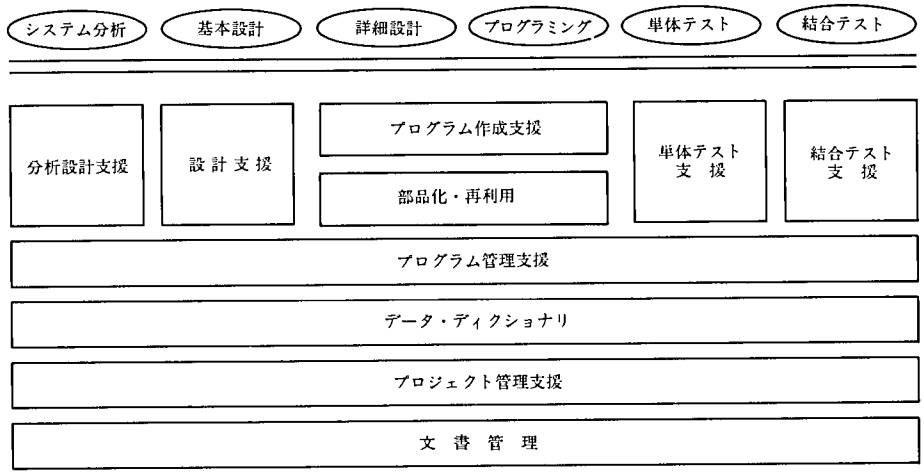


図3　開発工程別機能概要

データ・ディクショナリに定義された情報から定義体（登録集）を生成しメインフレームへアップロードする機能，設計情報をメインフレーム/サーバ間で相互にアップロード/ダウンロードする機能，設計文書を作成する機能等がある．

これらの支援機能によってシステム設計および各種定義の整合性が維持され，登録集の自動生成，項目の変更に対する関連項目の自動変更等システムの品質向上と開発・保守の生産性の向上をはかることができる．

2)　プログラム作成支援……仕様書をベースとしたプログラミング（仕様書の作成）とソースコードの作成を支援する．仕様書には実行単位としてのプログラム仕様書と翻訳単位としてのモジュール仕様書があり，この二つの仕様書体系は画面のパネルを通して容易に編集することができる．

木構造図エディタを使用して作成するプログラム構造図にコード化のための詳細説明（制御条件，実行命令，作業領域定義）を追加定義したものがモジュール仕様書となり，こ

表1 機 能 一 覧

機　能		機　能	
文書管理	文書管理	単体テスト支援	ファイル仕様編集 テストケース仕様編集 ファイル作成/抽出/ 印書/比較/加工 モジュール仕様編集 モジュール・テスト テストデータ仕様編集 モジュール・テスト テストケース仕様編集 モジュール・テスト 実行
設計支援	データ項目定義 レコード定義 ファイル定義 データベース定義 バッチ帳票定義 画面帳票定義 帳票フォーム定義(PW²) システム構成定義 ドキュメント出力 定義体生成 定義体アップロード 設計情報アップロード 設計情報ダウンロード	結合テスト支援(＊)	ファイル・マニュピレータ ラン走行ネットワーク編集 運用パラメタ & JCL 生成
プログラム作成支援	仕様書作成 仕様書検索 仕様書再生 仕様書印書(U6000, J3100) 仕様書印書(2200) ソース・コード生成 構文検査 仕様書アップロード 仕様書ダウンロード DD情報アップロード DD情報ダウンロード	プログラム管理支援	開発作業管理 デバッグ支援 プログラム更新 多重変更管理 リリース管理
		プロジェクト管理	進捗管理 経費管理 他ツール連携
		分析設計支援(＊)	モデル作成支援他
		データ・ディクショナリ	ホスト DD(2200版)
部品化・再利用	ソースコード展開 定義処理	(＊印　計画)	

れからモジュールのソースコードが生成される．

プログラム作成支援のもたらす標準化された高品質なソースコードの自動生成，仕様書/設計情報/プログラムの相互関連性の一元管理，類似仕様の活用等によって開発・保守の生産性向上ができる．

3) 部品化・再利用……プログラムの作成において，プログラムのひな型（スケルトン）およびソースコード部品（マクロ）の定義を支援する．またこれらのスケルトン，およびマクロをソースコードとしてプログラムへ自動展開することを支援する．

この部品化・再利用の機能によって，仕様の標準化，ソースコードの標準化，プログラム作成の省力化，テスト負荷の軽減が可能となる．

4) 単体テスト支援……単体テスト支援は，テストデータ仕様の作成，テストファイルの作成/加工，スタブ/ドライバ機能によるモジュール・テストの支援，テストデータ/ファイル/テスト環境情報の一元管理を行う．

またテスト結果の検証を容易にするために，ファイルの内容を印書する機能，二つのファイルの内容を比較する機能がある．

これらの機能によってテストファイルの作成/再生負荷の軽減，対話型デバッグ支援によるデバッグの効率化を図ることができる．

5) 結合テスト支援……結合テスト支援機能では，テスト用のデータベースやオンライン・ファイルを容易に作成/変更/印書する機能，ランやタスクの進行をネットワーク図を使用してテストケースとして定義する機能，プログラムやファイルの定義およびラン・ネットワーク図等から実行 JCL や運用管理用のパラメタを作成する機能等を予定している．

これらの機能によって，テストデータ，ファイル，テストケース，実行 JCL，運用管理用パラメタ等の作成負荷の軽減が可能となる．

6) プログラム管理支援……プログラム管理支援は，ソフトウェアのコードとその状態（開

発/修正の開始および終了等)，ソフトウェアの構成とそのバージョン，ソフトウェアの変更履歴，同一プログラムに対する多重変更，リリースの予定と実績，リリース・ファイルの作成等を一元的に管理する．

これらの機能によって，システムの整合性が維持され，システムの現状，変更履歴，修正コード内容，リリース状況を正確に把握することができる．

7) プロジェクト管理……システムの開発および保守作業を設定したスケジュールとコストの範囲内で実施できるように支援する．ツールは，進捗管理とコスト管理の二つのサブシステムから成り，それぞれ次の機能をもっている．

(進捗管理)

—作業計画の登録(成果物,日程,担当者)
—作業実績の登録（進捗状況，消化率)
—関連情報の登録（協力会社，作業グループの属性)
—内容照会（グラフ表示)
—報告書出力

(コスト管理)

—要員計画の登録（作業，役割，能力，月別計画)
—工数，経費の実績登録
—関連情報の登録（協力会社や経費費目の属性)
—内容照会（グラフ表示)
—報告書出力

これらの機能によって，作業の進捗状況およびコストの消化状況を容易に把握でき，管理者による迅速かつ的確な判断/アクションが可能になる．さらに，作業の機械化によりリーダあるいは管理者による管理負荷の軽減がはかれる．

8) 文書管理……開発，保守工程で作成されるワープロ文書をサーバ上で一元管理するために下記の機能が提供される．

—文書の転送（WS ←→サーバ)
—文書の検索
—文書の登録/更新/削除
—セキュリティ・レベルによる文書の検索と更新の保護
—メールの送受信

また文書は,章/節レベルで分割管理することができる．

9) 分析設計支援……分析設計支援機能は，システム化領域を分析しモデル化することにより，情報システムとして開発する機能を定義するための支援機能の提供を予定している．

10) データ・ディクショナリ……システムが扱うデータの定義，システムを構成するソフトウェアの定義，システム開発に関わる要員の情報，システム開発の状態等，あらゆる情報を一元的に記録しておくデータベースで，オブジェクト指向の拡張 ER (Entity Relationship)モデルで表現されている．これらのデータは，IDES の設計支援，プログラム作成支援，部品化・再利用，テスト支援，プログラム管理支援の基礎データであり，また出力データでもある．

たとえば，設計仕様は自動的に記録され，この情報から登録集や定義体，さらにプログラムの一部生成も行う．またテストの時には，プログラムに関する情報やテスト方法に関する情報の登録によってテスト作業の一部が機械化される．

このようにシステム情報を一元的に管理しているデータ・ディクショナリによって，複雑に関係しあうシステム情報を整合性を持った情報として活用することが可能となり，開発から保守に至るシステム・ライフサイクル全般に亘る効率化が可能となる．

掲載論文梗概

古田茂の**意思決定支援ソフトウェアFACILE 1100の特徴と今後の方向**は，約9年前に開発された意思決定支援ソフトウェアFACILE 1100の開発の経緯をたどりながら，人間の考えに近い発想を持った同ソフトウェアの機能と特徴を説明するとともに，将来の方向についても触れている．

帯電電圧が約3kV未満の低電圧の場合，帯電した金属物体に触れても衝撃を感じることはない．本田昌實は，**ショートモノポール・アンテナを用いた間接ESD測定**の中で，低電圧に帯電した金属物体間の間接静電気放電（ESD）によって生じた電磁界をミリメートルオーダのショートモノポール・アンテナを用いて測定し，測定結果から，今までとらえられなかったインパルス界の特異な性格，すなわち超広帯域性と空間における単極性が明らかになったことを報告している．

本来光ディスクは，媒体が交換できる大容量記憶装置であるが，1社の駆動装置には1種の媒体しか使えないのが現状となっている．本稿の筆者は工業技術院の委託により設立されている光ディスク標準化委員会に互換性検討に関する作業部会の設立を提案し，該部会の主査として互換性の現状把握とその阻害要因の究明を行い，その原因を突き止めた．大石完一の**光磁気ディスクの互換性阻害要因の究明**は，光磁気ディスクの互換性に関する各種試験の概要および互換性阻害の原因解析並びに対策について考察したものである．

戦略情報システム（SIS）は市場に新しい価値を提供するために構築される．これはビジネスの再構築を必要とするだけでなく，タスク，役割，権限を変える組織変革を伴う．このような経営革新は組織の深い部分に及び，人と組織までも変革の対象とする．小坂武は，**戦略情報システム構築における人間軸の世界**の中で，SISを構築する上で必要となる人間軸の世界を取り上げ，シャインのUCRモデルとベイトソンの人間の認識モデルを基にして，新しい情報技術を導入する上で必要となる変革のための方法論を構築している．

掲載論文梗概

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 11　No. 1　(No. 29)

発　行　日　平成3年5月31日
編　集　人　柳　生　孝　昭
発　行　人　富　田　和　夫
発　行　所　日本ユニシス株式会社
東京都港区赤坂2-17-51 〒107
TEL(03)3585-4111（大代表）
印　刷　所　三美印刷株式会社

UNISYS

スケールもコンセプトも、21世紀の情報中枢となるために。

ユニシスは、始めます。情報と社会の新しい未来を。その大きな一歩として、いまメインフレームのかつてないコンセプト「インフォメーション・ハブ」を提案します。グローバルなビジネスの進展と、激変する経済環境。戦略的企業経営の中枢を担う情報処理システムもいっそう高度化、複雑化し、その情報量とシステム資源は飛躍的に増え続けています。21世紀へ向けてのさまざまな課題に応えるために、ユニシスが目指しているのが、メインフレームからワークステーションまで、あらゆるベンダーのあらゆるステージのシステムを統合し、情報資源を共有する「インフォメーション・ネットワーク」の実現。「インフォメーション・ハブ」は、その中枢を担うための先進の機能を備えた、メインフレームのまったく新しい姿です。そしてこのコンセプトのもとに開発された最上位のメインフレームとして、世界最大規模の巨大システム空間を備えた超大型汎用コンピュータ「UNISYS 2200/900シリーズ」を、発表します。

インフォメーション・ハブの条件とは　ユニシスのテクノロジが答えます

○次世代システム・アーキテクチャ「2200/XPA」により
32台のプロセッサを結合、72ペタ・バイトの巨大システム空間を実現します

新開発の拡張処理アーキテクチャ「2200 XPA」を採用。8台のプロセッサを備えた最上位モデルを最大4システムまで接続することにより、最大32台のプロセッサからなるシステム構成が可能に。システムの限りない成長性をお約束します。また、世界最大規模の72ペタ・バイトという巨大システム空間、2ギガ・バイトの大記憶容量、276ギガ・バイトのプログラム・アドレス空間を実現。さらに、入出力機構の高速化により、大規模トランザクション処理やデータベース処理の高速化を飛躍的に高めます。また、バッチやデマンド処理のスピードアップなど、「2200 XPA」はさまざまな革新的機能をもたらしました。

○24時間、365日連続稼働も可能に　無停止連続処理の時代が始まります

新開発テクノロジにより、完全なノンストップ・システムを実現。障害が発生しても切替え動作不要のまま、処理を継続。ホット・スタンバイ・システムを超えた、新しいシステムの信頼性を手にしました。

○相互接続・相互運用性でオープン・ネットワーク化を加速します

ユニシスのネットワーク・アーキテクチャ「DCA」はもちろん、OSI、SNA™、TCP/IPなど各種ネットワーク・アーキテクチャをサポート。オープンな相互接続性・相互運用性を提供します。

○ユニシス独自のテクノロジの数々が、21世紀へ向けてさらに磨かれました

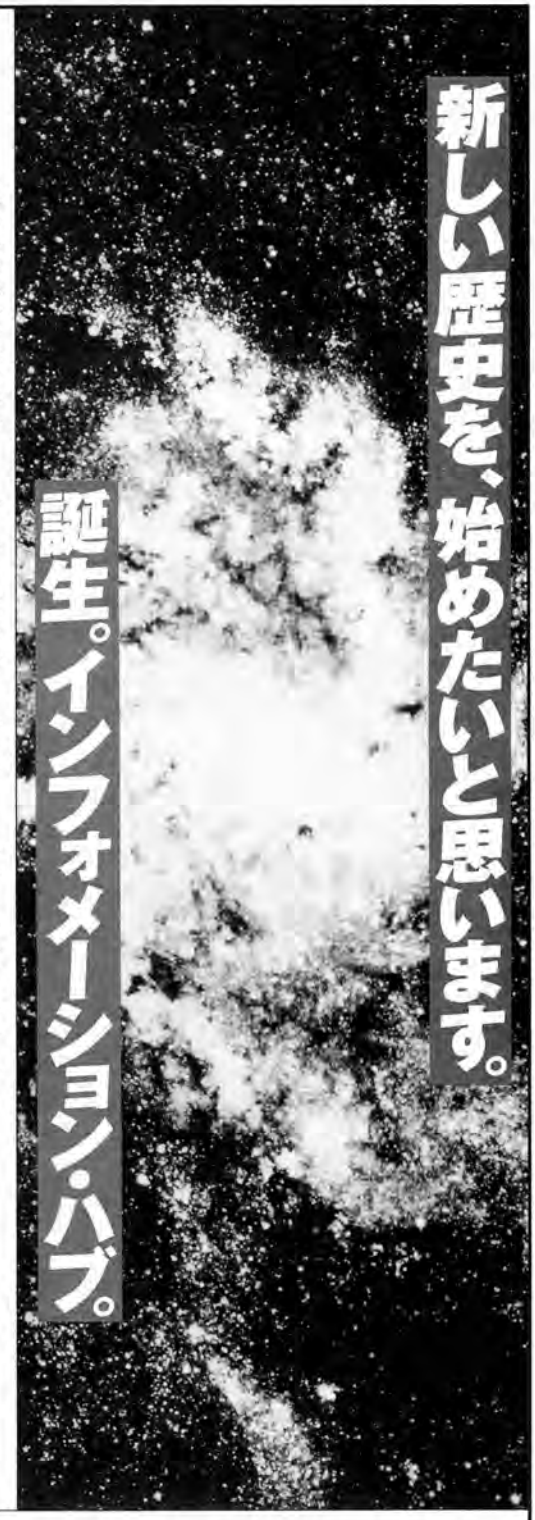
ひとつのプログラムから複数のデータベースを同時使用できるUDSを提供するなど、先進的なデータベース・システム。より強化された4GL（第四世代言語）環境でのMAPPER、LINC II、4GL CASEを提供する、高生産アプリケーション開発・実行環境。またシステムの無人運転・統合管理の支援、米国NCSCの認定を受けた高度なセキュリティ。そして高密度多層基板（DPCB）に超高速ECL VLSIを搭載した高性能シングル・ボード・プロセッサの実現……など、ユニシスならではの技術力が、インフォメーション・ハブ実現のためにさらに磨き抜かれました。

XPA：eXtended Processing Architecture
ペタ：1,000兆

SNA™：SNAはIBM社の登録商標です
NCSC：National Computer Security Center

超大型汎用コンピュータ

UNISYS 2200/900 シリーズ



日本ユニシス株式会社　本社 東京都港区赤坂2-17-51 〒107 電話03-3585-4111(大代表)