TECHNOLOGY REVIEW

UNISYS 技報

通巻 26

1990年8月発刊 Vol. 10 No. 2

特集：4GL

日本ユニシス

ISSN 0914-9996

経営資源の一つとして情報システムの果たす役割が大きくなっているが，現実にはほとんどの情報システム部門はバックログに代表される多くの課題を持っている．一方で多くの情報機器がオフィスの現場に導入されてきている．横山正敏は，**変革の時代と4GL**の中で，これらを効果的に活用するために，ネットワークを超えての情報アクセス手段とエンドユーザの情報リテラシーの高揚の重要性を指摘するとともに，ユニシスの4GLがAP開発言語から「共通の言葉」を基盤とした情報環境の構成要素にまでなってきていることを述べている．

利用者のニーズが基礎となって誕生した4GLは，なかでもシステム開発の生産性向上ツールとして広く支持を得てきた．一方，各社が提供しているCASEツールは，使い勝手/一貫性の面から統合化がさけばれ，ICASEの概念が登場した．川本光一の**日本ユニシスにおける4GL/CASEへの取り組み**は，当社4GLの設計工程・運用/保守工程への機能強化の予定とともに，4GL/CASEについて「統合環境」を提供するプロダクトとしての方向性について述べている．

MAPPERは4GLの中でも歴史が古く，実績も定評があるが，本来の特徴を生かしたエンドユーザ・オリエンテッドなシステムとしては，十分に活用しきれていない面があった．松木規子は，**MAPPER利用の現状と効果的な適用方法**の中で，MAPPER利用の現状の抱える問題点を明らかにした上で，今後より効果的にMAPPERを適用するためには，どのように考えていけば良いかについて考察を行っている．

エキスパートシステム開発が実用システム開発の段階を迎えた今日，従来の情報処理システムといかに融合するかが課題となっている．保科剛と川上峰子は，**MAPPERとエキスパートシステムの連動**の中で，化粧品の生産スケジューリングを事例に，MAPPERアプリケーションとエキスパートシステムの運動およびスケジューリング・エキスパートシステムについて述べている．

LINCの発展と開発規模の拡大に伴い，LINCに対する要望・意見も変化してきた．とくに開発が大規模化している現在は，開発環境のあり方についての要求が多くなっている．山科順一の**LINCにおける分散開発の現状と今後の課題**は，LINCで大規模システムを開発する場合に必要となる，分散開発とは何かを明らかにし，分散開発する場合の環境のあり方について提案している．

LINCはEVENT, COMPONENTという概念で，ビジネス活動をモデル化するオブジェクト指向性がある．福井務は，**LINCの持つオブジェクト指向性と今後の問題点**の中で，オブジェクト指向の持つ一般的概念を整理し，LINCにおけるオブジェクト指向性の評価を行い，さらに課題と今後の期待を述べている．

LINCによる大規模システム開発の一方法として，詳細設計段階のプログラム設計を省略してシステム開発を実施した．浅田昌紀は，**LINCによる一開発形態——概要設計即プログラミングの是非について**の中で，今回の開発方法はLINCを利用することにより，どのような仕様書があればよいのか，どのような開発体制があればよいのかという問に対する一材料を提供している．

葛飾区役所では，プライバシ保護優先とエンドユーザによる開発・運用を柱とした住民記録システムが，15か月/200人月の開発を終え，1987年10月1日本稼働した．森山勉は，**葛飾区役所における住民記録システムの開発事例**の中で，住民記録システムの業務機能，ソフトウェア開発方法論としてのSIM，およびLINC適用についての概要を紹介している．

戦略情報システム（SIS）を構築する企業が増えてきた．森正光と紙谷啓一郎の**MAPPERによる経営情報システム構築——西部ガス(株)における事例**は，データベースを全社的に統合し，このデータベースへのデータの集約から加工・検索までの機能を持つOAシステムの開発により，情報の戦略的活用が可能となった事例の紹介である．

特集「4GL」の発刊によせて

岩佐宏一

MAPPERが日本国内で販売開始されてから9年，LINCは8年が経過した．稼働しているシステムの数はMAPPERで800，LINCで600を超えている．この間に適用，開発されたアプリケーションはさまざまな業種にわたり，その領域も企業の基幹を構成する大規模なものから経営トップの意思決定支援システム等の非定型分野に至るまで広い範囲をカバーするものになっている．当社の提供するMAPPER, LINCを使用したソリューション・パッケージも年々その数を拡大しつつあり，また最近ではAI（人工知能）等の新しい技術と連動した使用法も実現されてきている．

一方でMAPPER, LINCはともに大幅な機能拡張を繰り返しながら現在に至っているが，これらの進化の源泉がより一層の「システム開発・保守の生産性向上」を求めるお客様の声にあることは論を待たないだろう．激化する企業間競争の中にあって，情報の戦略的活用はより重要性を増すであろうし，その実現を支える情報処理技術の高質化への要求も一段と強いものになっていくことは明らかである．これまでの歴史，実績から見ても，MAPPER, LINCが情報処理のインフラストラクチャの一翼を効果的に担っていくこと，開発・保守の生産性向上に寄与していくことへの役割と責任は大きいと言わねばならない．

1980年代の初頭から世に出た「4GL」という言葉は厳密な定義を与えられることなく現在に至っており，ソフトウェアの数は日本国内一般に知られているものだけでも50を超える．簡易言語と呼ばれるものから意思決定支援ソフトウェア，シミュレーション言語までまさに百花撩乱の趣であるが，それぞれにいくつかの問題を抱えながらもシステム開発の現場で一つのエポックを築いてきたことは否定できない．そして今，ここ数年来注目され，実用化の段階に入りつつある「CASE (Computer-Aided Software Engineering)」の台頭の中で改めてその有効性が問い直されている．

MAPPERとLINCは多くのお客様のシステム開発の上で大きな成果をあげてきたが，なお一層「最小限の努力で最大の効果を」というテーマに少しでも近づくためには検討，解決されなければならない課題も多い．

MAPPERとLINCについてその使い方の上で，重要な検討項目のうち今敢えて一つを挙げるとすれば，「COBOLの代わりのプログラミングツール」という考え方ではなく，システム開発の全工程の中でそれぞれのソフトウェアが持つ特徴をどう生かすかという点の認識であると思われる．単にプログラム作成工程の中でしか利用されないのであれば，本来のMAPPER, LINCの特徴の一部しか活用されていないと言える．すなわち，第四世代言語の「言語」

としての側面だけに焦点が合わされてはいないかが提起する問題の一つである．

現実に，お客様のシステム構築の上で解決されるべき課題は幅広く存在し，しかも相互に関連し合っている場合が多く，プログラム作成工程のみでの利用と言う観点では解決への期待に十分沿えないのではないだろうか．

また，言うまでもなく4GLは万能ではありえず，それ単独ですべてを解決することはできない．従来のさまざまな開発技法，ツールとの関連の中で生かされると考えるべきであろうし，さらにネットワークの発展，ダウンサイジング，マルチベンダ環境，3GL資産の継承等といった4GLをめぐる諸環境との結びつきへの考慮は不可欠なものといえる．今後のMAPPER, LINCの活用の有効性は，これらの視点から検証されていくべきと思われる．その意味からもMAPPER, LINCのより広い利用技術，経験の交流が今後ともとくに必要となるだろう．

1989年4月当社に組織された4GLセンター(本社，関西，中部)は，①MAPPERとLINCに関する利用技術情報の集約・統合・提供，②システム構築へのコンサルテーション，③セミナを通しての体験に関するサービスの実施，等を役割として活動を展開し，この一年で関連セミナを含め1,200名を超えるお客様に受講頂き，約100社のシステム構築の支援を実施してきた．さらに，多くの経験の文書化，提供にも力を注ぎつつある．また，システム間をつなぐデータのやりとり，より良い運用を目指すソフトウェア等の企画，開発，適用指導も行っている．

米国ユニシスも4GL関連プロダクトの更なる強化，拡充を表明しており，CASEとの連動，マン・マシンインタフェースの改善，実行効率の改良等の実現でMAPPER, LINCはこれまで以上にお客様のご期待に沿えるであろうと考えている．

本特集号ではMAPPER, LINCの利用技術の到達点を確認し，今後の発展方向を見据えるべく，適用事例，開発方法論，今後の新機能等，幅広い分野にわたるものとした．これからシステムの構築に関連する人たちにとって本特集号が何らかのお役にたてれば幸いである．

(4GLセンター 技術部　部長)

変革の時代と4GL

4GL in Changing World

横山正敏

要　約　揺れ動く世界，不透明で先の読めない時代にあって，経営資源の一つとして情報システムの果たす役割がますます大きくなっている．ところが現実には，ほとんどの情報システム部門はバックログの増大に代表される多くの課題を持っており，必ずしも欲しい人が必要とする情報をタイムリに得られる環境として提供できているわけではない．一方でコンピュータと通信の革新的進歩により，多くの情報機器がオフィスの現場に導入されてきている．これらを効果的に活用するために，秩序ある統制に基づくネットワークを超えての情報アクセス手段と，エンドユーザー人一人の情報リテラシーの高揚が重要になってくる．

このような時代に合わせるように，ユニシスの4GLは単なるAP開発言語から「共通の言葉」を基盤とした情報環境の構成要素にまでなってきている．

ある意味では新たな企業文化の創生にも一役買っているといえる．

Abstract We are facing to the changing world. As the age of uncertainty, the role of information processing system is constantly expanding as part of management resources. However the most of DP organization within enterprises has many problems represented by cumulative application backlog, they could not provide the environment where users can obtain necessary information in timely manner.

On the other hand, based on the innovative progress in computer and communication area, various data processing devices have been installed in offices for end users. In order to utilize these facilities effectively, information accessibility across the well-managed network system and empowerment of information literacy of individuals become so important.

In responding to these changing demands, Unisys 4GL is also expanding its functional coverage from those as a simple application development language to the one of the primary component providing information environment based on the "Common Language" among end users. It is probably correct that 4GL is contributing to re-create the enterprise culture through their utilization.

1. 90年代——世界の潮流

昨年から今年にかけて世界は大きく揺れ動いた．ペレストロイカを引き金にした東西の緊張緩和，ベルリンの壁の崩壊に象徴される東欧の民主化への雪崩現象，ゴルバチョフ大統領の誕生とリトアニアの独立問題等，一瞬でも眼を離すと，そこに変化が起こっている．まさに先が読めない不透明で変化の多い時代の始まりである．

いかに先が読めないかの一例を日経平均株価(東証225種)にあげよう．1990年1月3日付日経新聞紙上で株価展望のアンケートを行った．対象は，日本の一流企業現職の会長，社長，頭取など経営トップ20名であった．結果の一部を表1にまとめる．

表1で見る限り大半のトップは，今年の前半から後半に向けジリ高となり年末近くにピークがくると予想している．現実には，「2月の選挙で自民党が勝てば株価は安泰」と言われていたが，安定多数の議席を獲得したにもかかわらず，大方の予想を裏切っ

表 1　日経平均株価予想
Table 1　Estimated stock prices by top executives

氏名	① I氏	② I氏	③ K氏	④ S氏	⑤ D氏	⑥ H氏	⑦ I氏	⑧ O氏	⑨ K氏	⑩ S氏
高値(円)	44,000	46,000	41,000	42,000	45,000	45,000	43,000	48,000	42,000	43,000
時期	12月	12月	5月	12月	12月	12月	10月	12月	12月	12月
安値(円)	37,000	37,500	35,000	37,000	38,000	37,500	37,000	38,000	36,000	36,000
時期	3月	2月	11月	2月	1月	2月	2月	2月	2～3月	2月

氏名	⑪ T氏	⑫ H氏	⑬ I氏	⑭ N氏	⑮ M氏	⑯ I氏	⑰ S氏	⑱ T氏	⑲ N氏	⑳ M氏
高値(円)	43,000	48,100	44,000	43,000	41,000	45,000	45,000	42,000	42,000	44,500
時期	12月	12月	12月	12月	2月	10月	12月	10～12月	12月	12月
安値(円)	37,000	38,000	35,000	36,000	35,000	37,000	37,000	36,000	37,000	34,500
時期	2月	1月	5月	5月	10月	2月	2月	1～2月	1月	2月

て選挙直後から連日の下落が続き，4月2日には28,002円を記録した．本稿をまとめている時期(90年5月中旬)ではまだ，年初から5か月しか経過していないので高値の方はかなりの速度で戻す可能性もないではないが，こと安値に関しては何と20人とも大ハズレ!となってしまった．緊張を高めつつ構造協議に入った日米経済摩擦や円安，金利上昇等諸要因が錯綜しているとはいえ，各界のビジネス・リーダーの予想ですらこの結果である．

三菱総研の「90年代の世界全予測」によれば，90年代を象徴する第1のキーワードは「グローバル化」である．ここでグローバル化とは単なる国際化と異なり「資本の国境を越えての移動」と定義されている．現に経済においては国境がなくなりボーダレス・エコノミーが展開されている．ここで必要なのは地球規模での発想と経営の実践である．グローバリゼーションのためには基盤として交通手段の発達と情報通信のインフラストラクチュアの進歩が大いに貢献している．すでに世界中の拠点を結び24時間オペレーション(研究・製造・印刷等)を展開している企業も増加している．

さて第2のキーワードは「社会主義」であるとされている．ゴルバチョフ氏が大統領に就任しペレストロイカを推進しようとしているが，積年のウミを放出し民衆の生活改善にまで効果を及ぼすにはかなりの時間が必要との観測もあり，東欧の激変を見るにつけ，社会主義経済の崩壊の兆しが見え始めたとの意見もある．一方では社会主義経済の対局にある資本主義経済も物づくりを放棄し，財テクに走り，会社すら売買の対象とする「投機という熱病」に犯され，破局を迎えつつあるとの警鐘も鳴らされているという．

第3のキーワードは「ストック経済化」だという．1955年を基準として1987年の価格を見ると日本の消費者物価，アメリカの株価，国際石油価格等が4～5倍であるのに日本の株価は72倍，全国市街地価格は49倍で，まさに狂気のストック膨れ(バブル)であるという．破裂した時が恐ろしい．今回の株安，円安もバブルの破裂現象と見る人もいる．

第4のキーワード「アジア」は2000年の経済シェア予想(北米36％，日本を含むア

ジア34％，EC30％)からきている．

第5のキーワード「技術覇権」すなわち国家の覇権は「軍事力」から「経済力」へと移り，90年代は「技術力」を持つ国になるとの予測である．ここでとくにエレクトロニクスなど情報技術が強調されている．

90年代は大いなる変革の渦の中で企業間の「競争」と「協調」において，また社会生活の情報への依存度の高まりという側面において，コンピュータと通信を中心とした「技術」の果たす役割は限りなく大きなものになっていくであろう．

2. 変化に合わせた企業の対応

経済のグローバル化，社会の情報化，そして人々の価値観の多様化，これらのパラダイムの変化に対応して，企業は新しい尺度でのエクセレント・カンパニーを目指す上で国際的に通用する「経営理念」を創造する必要に迫られているという．また規制緩和や，垣根(業際)の不明確化にともない「見えない敵」をも意識した上で競争優位性を確保し維持し続けることが最も重要とされている．

前述のように経営環境が変化することにより競争の中味，相手も変化(進化)していることにも注意を払う必要がある．物不足時代，高度経済成長期と違い，安い物を大量に生産，販売し，市場シェアを支配する「量的指向」から，多様化する価値観に対応した高付加価値を持った差別化された商品・サービスを提供し，より高い利益を確保し，社会に貢献できるような「高付加価値指向」への転換が求められるようになってきている．経済のボーダレス化により，競争相手は昨日までのライバルのみとは限らず，昨日まで顧客であった企業が新規参入してきたり，新しい商品が突如，高機能・低価格の代替品として登場してくることがしばしば発生するようになった．

情報産業においても，鉄鋼メーカ等の新規参入やポータブルパソコン市場におけるノートパソコンの登場等は記憶に新しいところであろう．

「競争優位の戦略」を言い始めた経営学者のマイケル・ポーターによれば，競争優位を築くために，「業界内の競争業者」はもとより，「買い手」，「売り手」さらには「新規参入業者」，「代替品」からの差別化が必要であるという．

さらに，ハーバード・ビジネス・スクールのウォーレン・マクファーレン教授は地域特性，特定購買層の好みや購買パターンの分析に基づく，より細分化されたターゲット市場へのアプローチ，すなわち「マイクロ・マーケティング」の必要性を強調している．そのためには，幅広い品揃えとオプションの充実により選択肢を広げること，買いたいと思わせ，買った後でも良かったと思わせるようなきめ細かいサービスが必要である．人は単にハードウェア(モノ)の良否だけでなく，買う時の楽しみ，雰囲気，使い心地，自分の空間に持ち込んだ際の落ち着き具合等，自からの価値観に基づく選択をするようになっている．このようになってくると従来の「何が」，「どのくらい」売れたか，という平均値を浮き彫りにするようなデータ分析だけでは市場の変化に追随できない．ここにも情報の戦略的活用の意義が出てくる．

さらに顧客・購買層等の変化に加えて，職場においても女性の社会進出等，表層的現象面での推移だけではなく，「給与よりゆとり」という言葉で代表されるように社員自体の意識・価値観も変わってきている．

このような経営環境の変化に対応して，経営者の意識も変わり，経営戦略も変質してきている．

すなわち高度成長期においては，経済規模(マス)そのものが拡大していたから，「いつまでに，これとこれを，いくつ作れ!」等と号令していればよかった．ここでは，主に量をこなすために不足する労働力の定着化を図ることが課題であった．しかし，今日では，売れるものを探す，競争相手に差別をつけるといった目標自体の質的変化に対応することが重要になってきている．企業活動で言うと，戦術レベルから戦略レベルの競争になってきたといえよう．この競争は，企業の一部門の戦いではなく全社，さらには関連企業を含めたグループ同士の戦いにまで広がってきている．

ここでは企業自身の変身と組織の活性化，リストラクチャリングが求められている．企業変身のためには，「会社の寿命」等でも指摘されたように，本業自体を変えていくことが重要であるとされる．すなわち，「顧客分析」，「競合分析」，「業界分析」，「環境分析」等の外部分析と「人材」，「商品」，「サービス」，「技術力」，「アイデア力」，「財政上の資源と制約」等々自社の強み弱み等の自己分析とをベースに戦略の識別と選択が必要であるとされる．

さらに，選択肢のひとつとして，リソースの拡散化を防止する意味で何かをヤメル決断も必要になる．まさに筑波大学大学院寺本教授がいう，消費者ニーズに対応していく受け身の戦略「変化対応型」からニーズそのものを顕在化していくという積極的な戦略「変化挑戦型」への転換が求められている．したがって発想も積極的でなければならない．たとえば，グロール化に伴う時差等は，それをハンデと考えずにむしろこれを優位点ととらえ，地域を分散することによる 24 時間体制としての展開がすでに一部企業で始まっている．

そしてここでも重要なのは，人材であり，組織であり，仕事の仕組みである．人間に対する教育も含め，これらに対する変革(高質化)を短い期間で達成したとき競争優位性が生まれるという．この時間短縮のために情報をいかに活用するかが一つの大きなポイントであるとされる．このように適切な情報をタイムリに生み出す最も重要な手段として企業情報システムがある．極端な表現をすれば情報システムの破綻が経営の危機ということにもなりかねない時代にわれわれは生きている．

3. 情報システム部門の現状と課題

社会の情報化，ソフト化が進展し，工場の自動化，オフィス・オートメーションが進んだ今，SIS(戦略情報システム)がブームとなっている．

経営にとっては戦略が，そして戦略にとって情報が武器となろうとしている．経営者にとって，以前の，考え方と技術の歩調が合わず失敗に終わったと言われる MIS と違って，今回の SIS は著名な会社の成功事例と言われるものをつきつけられているだけ説得力があり，見逃すわけにはいかないものがある．

ところで，いざ SIS を構築しようとしたとき，何が SIS であり，具体的にどうやって，誰をリーダにし，どのような組織で構築させたらよいかとの迷いが経営者の頭をよぎるに違いない．それはなぜであろうか？一つは情報システムの構築であるので情報システム部門をはずすわけにはいかないという思いであり，戦略と言うからにはト

ップ自らやらなければならないと言い聞かせるからであり，とはいえ過去の経験則から強権的トップダウンのみでは動かない，根回しをベースとする日本企業の体質を知っているからこそ，企業全体に関係することでもあり，営業や企画部門も参加させないとまずいと配慮するからであろう．そして，まずなによりもSISとは何かと言うことが明確でないからであろう．このような状況から単純にCIO(Chief Information Officer—情報統括役員)を任命したから当社もSISを構築できる，といった思い過ごし(淡い期待—願望)を生じたりもする．

ところで，もし情報システム部門の長がSISを構築しろと命令を受けた場合，多くの人が悩んでしまうであろう．それは，SIS構築がそもそも難題であることもさることながら，情報システム部門がすでに多くの難問を抱えているからに他ならない．

その問題とは何かをレビューしてみることは，本稿のテーマ「4GL」とも関連し真の問題解決のためにも重要であると考えられる．

まず，システム部門が直面している課題を気がつくままに挙げてみよう．

1) バックログの増大
2) 開発システムの品質のバラツキ
3) 過大な保守作業(労力の80％とも言われる)
4) 陳腐化するシステム
5) 開発・保守コストの増大
6) ニーズの正確な把握が困難
7) 新製品・新技術への対応が困難
8) 要員不足および要員の高齢化

これらの課題についてもう少しその一般的背景をさぐってみよう．

バックログの増大(報告されている量はあくまで顕在化している部分——情報システム部門が認識している部分——のみであって，一説にはこの2～3倍は潜在化しているともいう)には，二つの大きな要因が挙げられる．まずシステム開発・改造しなければならない案件(アプリケーション)の増加である．

これにはFA，OA，LA，SA等といった広がりからも理解できるようにコンピュータ化したいと希望する部門や分野が企業内に(最近では企業間にも)大きく広がったこと，一方ではシステム開発の生産性および開発パワーが要求の増加度に比較して伸びなかったことである．他の部門から見れば多くの人員を抱えていると言われながらも，開発したシステムの品質上のバラツキが大きいこともあって要員の多くを既存システムに対する過大な保守作業のために消費し，新規要求はなかなか実現しない状況が続いている．複雑で巨大化したシステムを保守することは容易ではなく，新たな要求に対しても迅速な対応ができない結果，当然の帰結としてシステムは陳腐化してくる．マイクロエレクトロニクスに代表される情報処理技術の進歩が早い分だけ，一層システムの陳腐化が眼につくようになる．

さらにニーズの多様化に合わせ，その企業で扱っている製品や商品・サービスのライフサイクルが短命化の傾向にあるとともに，オプションなど顧客の選択肢が多くなったことが，保守の必要性をますます増大している．これでは正に悪循環で，最悪の場合，情報システムの可用性が商品提供の時期，方法等にインパクトを与えることも

ある．このようなことがあると，上層部からはずいぶん金を使っている割りに「ビジネス・チャンスを喪失した」，「戦略展開とペースが合わない」等と非難されることにもなる．

さらに新規開発業務は，外部リソースの活用を図る余地もあるが，保守作業となるとそれも困難な場合も多い．

また情報技術は日進月歩し，毎日のように種々の新製品が新聞紙上をにぎわせているが，このような状況の中で一早く研究し，既存システムの中に組み入れることはむずかしい．場合によってはマイクロ機器等は利用部門の方が良く勉強しており，それらの知識をベースにした新規要求等，情報システム部門では何が真のニーズなのか本質的に理解できないものも出現する．

また，一度情報システム部門に配属されると，知識・経験がものをいう職場だけに，長期にわたって開発・保守作業を続けることになり，他部門への転出機会が減少する結果，そのままの状態で年齢を重ねることになり，ローテーションによる組織活性化などの施策もとりにくくなる．このような恵まれない(？)環境にあって，手をこまねいているのであろうか．

多くの課題を抱えながらも情報システム部門は，その役割を変えながら復権を図っているのではなかろうか．

4. 情報システム部門の役割・使命の変化

'コンピュータ'の利用はもともとロケットの弾道計算や統計処理等から始まった．

'電子計算機'と和訳されたように最初は電子的に速く計算するための機械であった．商用コンピュータが登場し，一般の会社等で事務処理に利用されるようになってからも長い間，情報処理というよりデータ処理(数値処理)であったので，これを担当する部門も「電子計算課」とか「電算システム部」等と命名され，あるいは一般的呼称としての「EDP 部門」等が多く用いられた．それが約 10 年程前から「情報処理部門」，「情報システム部門」等と呼ばれるようになったのには理由がある．

実際に DP(データ処理)から情報処理へと取り扱う対象と処理内容が変わってきたからに他ならない．

ここで「データ」とは単に数字の羅列であり入力データ，出力データ等と使われる．「情報」は，この言葉のあらゆる使われ方を説明できる定義はないにしても，さらに対象領域が拡大し，インテリジェンシを持ったものとして扱われる．「情報資源」，「情報管理」等どのように使用され，'人'，'モノ'，'カネ'に加えて第 4 の経営資源としてその地位を確保している．

前述のように経営環境がグローバル化し，顧客の購買指向が多様化，個性化するとともに，業際を越えての新規参入等もあって競争が一段と激化している現在，情報処理に関しても，かつてのような日常の事務処理(あるイベントが発生したあとの後処理)の範囲に留っているわけにはいかないだろう．

現に多くの企業では後処理の進め方につても，さらに業務工程ごとの連結(統合)を深める方向での改善が実施されているし，イベントを予測する前処理へも向かっている．いわゆる事務処理の合理化，顧客管理や業務管理のツールといったいわば「戦術

レベルの支援」から，経営環境の変化に追随しながらの以下のような「企業戦略レベルの支援」が求められている．

1)　現場で起きている変化を現在進行形でとらえているデータベース
2)　階層，役割に合った形でアクセスできる仕組み
3)　さらに利用者の自的に合わせて自由に編集・加工できる機能
4)　利用者の柔軟な発想を引き出し(支援する)システム

これらは実現の方法と無関係にいわば，経営トップをはじめ企業内のあらゆる階層，部門で「欲しい時に目的に合った情報が簡単に取り出せ，加工できる」情報システムの構築が求められていると言えよう．しかし現実には前章で触れたような多くの課題を抱えているので，情報システム部門を「叱咤激励」するだけでは問題は解決しない．現場の利用者(いわゆるエンドユーザ)をも巻き込んだ形で，企業をあげての取り組みが必要であろう．

このような新しいパラダイムの中では，「エンドユーザ・コンピューティング」の考え方が非常に重要になる．本件に関しては後で大きなテーマとして議論したい．

このように第4の経営資源としての情報を戦略的に活用するためには，

1)　経営ビジョンに基づく中長期情報戦略の立案
2)　情報戦略を有効かつ確実にインプリメントできる組織体制・仕組みの確立
3)　仕組みを常にフレッシュな状態に保つメカニズム
4)　確立された仕組みを効果的に活用する企業風土の醸成
5)　社会へ浸透(貢献)する人間系を意識したシステムとすること

のアプローチが必要とされるている．

それでは情報戦略立案のポイントは何であろうか．ここでは「企業が競争上の優位性を獲得し，それを維持・拡大することを意識して構築された情報システム」というSIS(戦略情報システム)の定義が役立つと考える．自社の情報システムの現状を冷静な目で分析し改革することが始点かもしれない．その結果，情報システム自体を極端にスリム化した基幹システムと，多様な分散システムに分けて考える二極分化の発想が必要になるかもしれない．米国のSIS研究者として知られるマンハイム教授のいうように「SISの第一段階は先手必勝だからホームランを狙うこともできたが，第二段階に入った今ではみんなが追いかけてくるからホームランはむずかしい．むしろ連続してヒットを打つことを狙うべき」だとすれば情報システムの上での継続的差別化を十分に意識しながら進めるべきであろう．とくにこの分野では新しいテクノロジの実用化に伴い，常にシステムの修正や改良を加えなければ差別化の維持はむずかしいという宿命にあることは認識すべきであろう．

さらにここでは，通信ネットワークの発達を今後の見通しも含めて前提とする必要があろう．まして現代はビジネスの展開が単に一企業内にとどまらず関連企業も含めた「共生」を狙うべき時代だからこそなおさらである．いまや規模の大小を問わずコンピュータ・ネットワークの有効な利用が企業発展のカギを握っている．それから情報システムの投資が対売上高比で米国2.3％，日本1.4％という数字も何かの目安(ヒント)になろう．日経新聞が年頭に特集したように「合理化やコスト削減のためだけのコンピュータ利用は確実に過去のものとなりつつある．いまやコストをかけてもあら

ゆる組織階層で情報を有効に活用できるシステムづくりがネライになってきた．まさに『情報の"海"へ乗り出す大航海時代の始まり』だ」からこそ長期的投資計画に支えられた情報戦略の立案が必要になる．

つぎに組織・体制・仕組みの確立について考えてみよう．まず情報システム部門の役割を見直す必要があろう．ゴールの明確なサブシステムを設計・開発し，本番以降の運用・保守を行うといった専門家集団としての役割の中でのシステム開発の工程改善，効率化等は当然の役務として残るが，さらに経営戦略密着型のシステム展開や，エンドユーザを味方につける(ある範囲の構限を与え，管理負荷を分散する)ための教育・指導を含めインフォメーション・センタとして機能する等，全方位型の発想を基本とすべきである．

昨今，パソコン，ワープロの高機能，低価格化の影響もあって，従来からの「DP 部門が開発・提供し，エンドユーザは言われた通り使うだけ」という役割分担は崩れ始め，コンピュータの利用秩序が大きく揺れ始めた．もはや強圧的トップダウン型かつ束縛的管理は通用しないともいわれている．

だからといって全面的にエンドユーザ部門の自由裁量に任せられるかというと，現実にはパソコン等はその低価格化に誘導され部門決裁で導入できるためにいろいろな問題が発生しており，結果として大きな不都合が生じることがある．

現在，顕在化しているパソコン普及の問題点を思いつくままに挙げてみる．

1) 全社機種統一が困難なためバラバラに設置されている．
2) その結果ネットワーク化ができず情報の孤立化を助長する．
3) ソフトウェアも利用部門が独自の判断で購入する結果データ交換の道を塞ぐ．
4) 勝手に導入した機器，ソフトウェアに関しても困った時の'情シス'だのみで，予期せぬユーザ支援が増大する．
5) 始末の悪いことに業務内容寄りのアプリケーション・ソフト等，エンドユーザの方が詳しい場合も多く，指導的立場に立てない．
6) ディスケット管理の機会が多い割に不得手で，機密保護の必要なデータが机上に放置されたり，読めなくなって再入力等，大騒ぎをする．
7) 組織の中でもうまく使える人(えてして若者)とそうでもない人(えてして管理職)間で情報格差が生じる．

このような状況から見て，やはり情報システム部門の秩序ある統制は不可欠といえる．そのためには経営トップや情報システム部門長は，将来の情報システム，OA システムのビジョンを描きエンドユーザに方向を示すことが重要となろう．また実際の運用にあたっては，エンドユーザの力を借りるために，情報資源管理のためのツールの提供や環境の整備がポイントとなる．すなわちマンマシン・インタフェース(MMI)を十分に配慮した使い勝手の良いエンドユーザ・コンピューティング・ツールを情報の窓口として与えるとともに，その後方でデータベースやネットワークの整備を進めることが期待される．こうなってくるとエンドユーザの支援と管理に的を絞った専門組織の設置も必要となるかも知れない．そこでの強力な教育指導，啓蒙活動，サポートが情報マインドを持ったエンドユーザを育成し，ゆっくりとしかし確実に企業風土を変えていくことになろう．

いずれにしても情報システムの構築に際し，最終的に情報の恩恵を受ける人が誰か，また誰が顧客かの視点を持つことが重要であろう．とくに，えてして情報システムづくりの中で発生する可能性がある人間疎外に対しては，十分な考慮を払う必要がある．「コンピュータ・ルネッサンス」とも言われる今日，さらに人間系を意識すべきだと考える．

また一方で成功したSISの事例としてよく取り上げられるケースは，宅配便のヤマト運輸やアメリカン航空CRS「SABRE」等，単に自社の中でのクローズド・システムではなく広く社会に浸透・貢献している事実は興味深い．このような要素を持つシステムでないと長期的に存立・発展できないのではないかとも考えられる．

情報システム部門への期待とは，ひとことでいえば経営トップと共にこのような状況の中で，情報基盤を土台として新たな企業文化を創生するような諸活動のエンジン(推進役)を演じることであろう．

5. 期待されるテクノロジの進歩

ここで話題を転じてエンドユーザ・コンピューティングに関連する技術の進展を考えてみよう．いくら日進月歩の世界とはいえ，昨今の新製品ラッシュには目を見張るものがある．ワープロひとつとっても約10年前には大型冷蔵庫なみの箱に入っていた機能が現在のポータブル・ワープロで実現している．

この間価格も1/100程度になっている．これは技術進歩を物語るほんの一例で，今後ともかなりのスピードで進化し続けるであろう．あらゆる機器の基礎技術でもある半導体に目を移すと現在は1MbitのDRAMが主流だが，今年中には4Mbitのチップが量産に入り，16Mbit試作も始まっている．正にミクロンからサブミクロンに向けての競いである．これがある予測によると2000年には1Gbitチップも夢ではないという．

1Gbitという数センチ角のチップの中に，何と新聞なら4000ページ，電話程度の質の音声なら約4時間分入るという．電子手帳の中にぶ厚い辞書とテープレコーダが組み込める時代になる．

マイクロプロセッサ・チップも1984年に32bit機が誕生した時に2～3 MIPSであったものが1989年には約10 MIPS，1995年には100 MIPSにもなると予想されている．大きな記憶領域とものすごく速い処理能力が，ポケットに入るようになるかもしれない．

この結果コンピュータの小型化，低価格化，高性能化にますます拍車がかかることになる．大半の予想でパソコン導入は，大企業ではここ1～2年の内に確実に1人1台になると指摘されている．90年代後半には「1人3台」(会社・家庭・携帯用)との見方もあり一億総パソコン・ユーザ時代も近い．このような時代になると，現在の電話や複写機並みにオフィスの誰でもがパソコン(オフィス・ワークステーション)を何の不自由もなく使えるようになるだけでは不十分で，機器のインテリジェンシィが高い分だけ深い次元での活用度に個人差が出てくる可能性がある．

一方スーパーコンピュータの世界を見ても同じようなことが起きている．1968年の，CDC 6600が1 MFLOPS，1989年のCRAY 3が16 GFLOPSと20年間で16000

倍にまで性能強化した．従来資源探査や気象情報分析等一部の特別な用途での利用であったものが，価格性能費の良いミニスーパーコンピュータや，どこにでも置けるスーパー・ワークステーション化するとともに利用面でも民主化，大衆化し，ビジネスの分野でも多く使用されるようになってきた．今まで考えもしなかったことが〝できる〟時代になる．

もともとコンピュータと通信の技術基盤は相乗的に発展していく性格を持っているが，通信ネットワークに関しても，通信衛星等の情報経路の多様化・発達をベースとして移動通信を含めたISDNがますます進展し，安く大量に情報が伝搬できるようになる．現在最も多く導入されているOA機器である電話は，即時性と双方向性が持つ利点と弊害が共存している，電気通信の当初の目的は時間と空間の克服であって，このメリットを最大限に活用した電子メール，電子掲示板等もごく普通の通信手段として広く活用され始めている．とくにグローバル化を進める企業においては，今や国際電子メールの仕組みは不可欠の要素であるといえよう．また先進企業の一部では経営トップ用メールボックスを設営し，改善提案等，社員が自由に意見を上申する使われ方も始まってる．これは時間と空間に加え組織のハイアラーキをも克服しようとする一例である．

余談だがパソコン通信では多くの人に参加の機会があり，相手を意識せずにCRTの画面のみで会話することで，上司の顔色をうかがうこともなく物理的な距離を超えた自由な議論が展開し思わぬ良い結論を得ることがある反面，画面を通じてしか自分の意見を言えないため，フェース・ツー・フェースでの意思疎通の機会を活かせないといった弊害も出るとのこと．

現実に当社でもソフトウェア開発担当SEの中には，一日中一言も発せずに帰宅するケースも出始めており，「オフィス・ルネッサンス」の中にあって課題の一つと言える．

さらにはAIとともに地下鉄の運転制御や，エアコン，洗濯機等の家電製品でおなじみのファジー・コンピュータやノイマン機の苦手とする画像処理や音声認識等に応用されるニューロ・コンピュータ等もごく近い将来，オフィスでも身近なものになるであろう．経済のソフト化，量的指向から高付加価値指向等が進む中で商品やサービスの高度化・差別化を図っていくために経営資源の中で最も重要な“人”(知力提供型の新しい形の人材)をいかに育成していくかがポイントとなる．本章で述べたテクノロジはそのごく一端ではあるが，近未来において技術進歩の成果物は確実にオフィスでも利用可能となるであろう．

6. 企業人教育

今まで述べたように今後急速に一人一台時代に進むとした場合，そのような変化(進化)に合わせた人材育成が企業発展の鍵ともいえよう．

このような社会ではコンピュータ機器の高度利用は企業人必須の能力となり，すなわち社員全員がシステム・マインドをもって情報を戦略的に管理・活用できるかどうかが，円滑な組織活動とともに企業そのものの「強さ」や「柔軟さ」と密接に関係してくる．そのためにはキャリアパスの設定やスキル・インベントリの考え方も含めて

長期的，計画的な人材育成をプランし実行する必要がある．

とくに労働提供型から知力提供型へと求められる人の質が変化するのにつれて，幅広い知識と経験をベースにした創造力を身につけさせることが教育の使命となるであろう．

東京学芸大学岡本助教授がいわれているように「日本では，文部省がしっかりしたカリキュラムを決め，それにそった学習指導要領が決められている．こういう教育的条件の国では，自由な発想を尊重するという教育環境は作りにくい」とすれば学校教育には頼れず，企業内教育を通じ，従来の「平均的人間に対する知識の伝達」にとどまらず，もっと自主的な計画能力や問題解決能力を養成しておかないと，たとえ情報システムで武装しても活用の仕方を誤れば自ら暴発の犠牲にもなりなねない．昨今のブームともいえるワープロ，パソコンの導入は本当にそれらを使いこなすとすると，機種やソフトウェアの違いから幾通りもの操作を覚えなければならず，一般のエンドユーザにとってすべて習得した上でそれらを最も適切な形で使い分けること等とてもおぼつかない．現にほとんどのエンドユーザは，ワープロ，表計算，データベース等の講習を受けて実戦に復帰したとしても，どれも身につかないか，あるいは多少使いにくいことがあっても一つのソフトウェアの中でいろいろな使い方を工夫するようになる傾向がある．さらにさくたんのことを教えようとすると，内容が表面的なものにとどまるとともに教育期間も長期化し，この間前線では活躍できないマイナス要因もあり，いかに内容を絞って効果的に，表層的なレベルにとどまらず探層的なレベルにまで修得させるかがキーとなる．

このためにはエンドユーザが使うツールの数(種類)を限定し，より深く体験(問題解決のための行動)を通した実のある教育が期待される．

教育そのものは，本稿の主題ではないのでこれ以上の言及は避けるが，社内で共通に使えるツールが持てるということは，そのツールを通じてそれが言語になり，システムでの会話が始まり，かつて文明の発達の中で文字(言語)が果たした役割の大きさに匹敵するインパクトで企業文化の醸成に影響することになるであろう．

7. 人間系を意識したパラダイム

高度情報社会においては，長い企業の歴史を通じて培われてきた企業風土，企業文化が非常に重要な意味をもってくる．これが新しいパラダイムの中でオペレーションを展開した場合の加速要因にも阻害要因にも成り得るからである．情報システムの進展に伴い，必ず変えなければいけない部分も出てくるわけで，その時の対応の仕方で次のステップでの展開がまるで異なるものになることさえある．ある企業ではすでに，情報システムをベースに「地位の上下や経験の長短に関係なく，全員が平等の立場で知恵を出し合うという企業風土ができ上がりつつある」という．

従来のように，情報を組織階層の上位のものが大事(情報格差による権威づけ)にしまい込み，業務遂行に必要な最小限しか部下に伝えないとすれば，そのやり方をいかに電子的にしようと結果はあまり変わらないであろう．むしろこのような情報のフローを妨げるピラミッド型組織は崩壊し，業務処理上意味のある組織，人々の間でのよりスムーズな情報フロー(情報直結・共有・民主化)を促進するようなネットワーク型

組織へ変わらざるをえないであろう．

日本総合研究所の濱田主研は，情報システム戦略の展開における三つの課題として

1) カルチャーの変革

2) 長期的展望の必要性

3) 要員の育成とスキル管理

を挙げている．ここでは長期的なコンセプトを持ったシステムづくりを基本に，エンドユーザのシステム・マインド向上や，専門SEに要求される能力が急拡大しているのに対応する育成・管理の重要性とともに，どこにお金を使うかの投資水準決定方式の再検討の必要性も説いている．実際にはこの分野においては投資効果(投資に対するリターン)がなかなか目に見える形で現れないために，抑制しがちな傾向にある．しかし今日SISで成功している企業ですら，投資決定の段階で現在享受している成果を定量的に確信をもって予測し得た訳ではないと思う．ここにトップの英断と推進者の情熱が期待されることになる．

どのような情報システムを展開することにもせよ全員がそれを活用するとなれば，その使用環境がエンドユーザにとって“ここち良い”ものでなければならないだろう．それにはマンマシン・インタフェース(MMI)やオブジェクト指向操作，あるいはWYSIWYG(What You See Is What You Get：見たものそのまま)等，人とシステムの直接の接点での人間への歩み寄りは当然のこととして，さらに奥深い機能レベルでエンドユーザの期待に応じられるものでなければならない．どんなに簡単な操作で情報アクセスができるとしても，提供されるデータベースの内容が目的に合っていなければ何もならない．ところがやっかいなことに，具体的な要求となると企業によりあるいはその中の部門によりデータの内容，検索の方法，タイミング，編集・加工のやり方等，必ずしも共通しないことも多い．これらの個別要求にいちいち応えていては情報システム部門としてはバックログの解消どころか増殖になりかねないため，あるレベルからはエンドユーザ自らのAP開発，運用(エンドユーザ・コンピューティング—EUC)を進めることになる．

人間系という議論の中でいつでも忘れがちになることの一つが，情報システム部門の専門SEもまた人間であるという視点である．彼等はいま，3GLを中心に構築された巨大なシステムに日夜立ち向かっている．そこにOSがあり言語プロセッサがあり，DB/DCがある．従来あるシステムの開発が決定すると，いわゆる「分析」，「設計」「製造」(コーディング，テスト)の各工程を経て実際の運用に入るとともに「保守」工程へと進む．昨今4GL/CASE(Computer Aided Software Engineering)の進展とともにここで言う「製造」以下の下流工程(Lower CASE)はかなり自動化・効率化が図られてきた．これらの恩恵を受けて開発担当者は労働集約型単純作業からは解放され，より人間らしい創造的仕事ができるようになってきている．さらにUpper CASEのツールが拡充すれば上流工程の進め方もより人間系に近づくことになろう．

つまり実際の開発要求を実現するための方法として，情報システム部門が自らのリソースの枠の中で開発するものと，エンドユーザ・コピーティングに委ねるものとに分離されることになる(開発の二極化)．この際のいずれにしても重要なキーになるのが“ツール”である．すなわち，専門家向けのCASEツールとEUC向けの開発実行

環境となりうるツールである．どちらの場合にも4GLが役立つ．

8. 4GLの果たす役割と期待

これまで述べてきたように，経営環境がますます複雑化・多様化し，し烈な企業間競争が業際間に及ぼうとしている中で，いかに経営ニーズにマッチした戦略情報システムをタイムリに構築・維持していくかが，どの企業にとっても重要課題となっている．しかし現実には3章に挙げたように開発要員不足，既存システムの保守負荷増大，システム化ニーズの爆発的増加等，バックログは累積する一方で，需給ギャップはさらに広がろうとしている．ここで情報システム部門としてはどのような対応策があるだろうか．

8.1 情報システム部門の対応策

このような状況を打破するための対応策を思いつくままに列挙してみる．

1) 開発要員の増加
2) ソフトウェア開発工程の標準化の徹底と管理手法の確立
3) ソフトウェアの流通(民主化)・部品化等，再利用の徹底(重複開発の排除)
4) 業務処理パッケージ等の積極的導入
5) 4GL等，高水準言語の積極的導入
6) データベースの利用部門への開放
7) エンドユーザによるAP開発・運用の推進(EUC)
8) コンピュータ支援によるソフトウェア生産システム(CASE)の導入
9) ソフトウェアの柔軟構造化による適用範囲の拡大
10) 対象や必要とする技術に応じた要員構成の階層化と分業化

これらをつきつめると

開発量を減らす→ 3),4),6),9)
開発者を増やす→ 1),5),7)
開発効率を上げる→ 2),5),8),9),10)

ということになろう．実際には，これらの組み合わせによる総合的な対応が実施されている．

さらに時代の流れに応じ，要求内容も，バッチ処理からオンライン処理へ，単純ファイル構造からデータベースへと移り変わっており，開発にはより高度な知識と経験を必要とする領域が拡大していることが，要員不足に輪をかけている．4GLの役割はこれらの知識や経験を自身の中に閉じ込め，それを利用する人に要求される専門性を軽減もしくは排除することにある．結果として4GLを利用すれば経験の少ない情報システム部員でもかなり大きな基幹リアルタイムシステムが開発できたり，現業部門のエンドユーザが自らの必要とするアプリケーション開発を行えるようになる．

8.2 EUC推進における情報システム部門の役割

情報システム部門内での適用については，それなりの秩序と統制の中で推進されるのであまり問題は起こらないが，EUCを推進するとなると推進母体としてかなりの役割を果たすことになる．それらは以下にあげるように，環境の整備からコンサルテーション，データベース公開まで広範囲で前向きな活動である．

1) ツール((4 GL)の選定とエンドユーザ向け開発環境の整備
ここではツール(ソフトウェア)そのものが洗練された奥深いものであることが必須条件
2) 職務階層別の教育一単なる受講でなく，実技実習の場としての環境設定が必要
3) テクニカル・サポート
・候補アプリケーション(業務)の拾い出し
・初期の開発指導
・実体験を通じての習得
・困った時のよろずコンサルテーション
4) エンドユーザ部門の現状把握と組織横断的広報・啓蒙活動
・AP(部品)の収集と流通(経験度を加速度的に向上)
・ノウハウの収集，蓄積，改良，広報
・社内コンペティションの企画・実践
・マニュアルの整備
5) 基幹データベースの公開
・オープン・データベースの提供
・セキュリティ・システムの整備
6) システムのキャパシティ管理
7) 課金システムの正当化とフィードバック

4 GL は第一(機械語)，第二(アセンブラ言語)，第三世代言語(コンパイラ言語)があってその後に登場したことから命名されている．

4 GL 自体の定義として決定打はないが，その進化の度合いを見るにつれこれを単なるL：言語(Language)と考えることは危険である．というのもユニシスの 4 GL は進化(機能拡大)し続けている．

主なポイントを以下にあげる．

① 既存システム(3 GL 資産)との連結機能
② オープン・データベースとの連携
③ オープン・システムへの適用
④ ワークステーション時代に合わせた分散処理と可搬性
⑤ CASE ツールとの連携も含めたソフトウェア・ライフサイクル全工程への波及
⑥ 開発・実行システム分離
⑦ ソリューション・パッケージの拡充

このような進化の軌跡を顧みると，情報システムの中に 4 GL/E とも呼べる姿がある．ここで 4 GL/E(Environment)とは，4 GL をベースとしたアプリケーション開発・実行環境，すなわち企業文化を構成する一要素と考えるほうが正しいであろう．

変革の世に，「変化挑戦型」の情報システム構築を進めるに当たり，この認識は非常に重要であると思う．どんなに優れた情報システムで，その中で大量の情報から選別し自由にアクセスできる環境が実現されても，その情報を活用するのは，個人である．その人の立場，役職，経験等にかかわらず，有限の時間の中でこの情報をどのように

活用するかは，正にその個人のシステム・マインドをベースとした情報リテラシー(情報活用力)に依存する．一人ひとりのシステム・マインドを培養し，さらに磨きをかけることが，結果として組織対応力を増強し新しい時代の企業風土，文化を生み出すのではないだろうか．今や，経営トップの方針演説や規範，上位マネジメントの人間としての行動を通じて育む文化に加え，情報システム環境の整備を通じて創生する文化との併合が，新しい企業文化の源となると考える．4GLは単に非定型のスポット業務や開発のプロトタイピングへの適用から，企業文化を醸成するための基盤ツールの一つとして期待されはじめてきた，今はそんな時代である．

今パリで「グラン・プロジェ」の名のもとに大規模な都市改造計画が進められている．アルシュ・サミットで一躍有名になった「グラン・アルシュ」(新凱旋門)や，一時大変な物議をかもし出したルーブル美術館中庭のガラスのピラミッドもこの計画の一環である．グラン・アルシュはまったく新しい街づくりラ・デファンスの一角にあり，歴史への敬意をもって旧都心の都市軸の延長線上に建てられている．ガラスのピラミッドは美術館の入り口として明るさあふれる快適な地下空間を提供し，かつ各展示場へのアクセスを大幅に改善したという．これらの例のみでなく「グラン・プロジェ」で一貫している思想は古くても良いものは可能な限り残し，そこに最新の技術を適用して新しい利便性・合理性を追求していくことにあるという．

情報システムの改造にあたり，考え方の上で学ぶべき点は多い．東京のような無秩序な開発も，ブラジリアのような過去を捨てた新構築アプローチもそぐわない．

執筆者紹介　横　山　正　敏 (Masatoshi Yokoyama)

1961年東京都立港工業高校卒業．同年日本ユニシス(株)入社．当初CEとしてハードウェア機器の保守業務に従事，その後ハードウェアのフォールト・リカバリ機能開発（OSへの組み込み）作業担当を契機にソフトウェアの世界に入る．1978年に，当時から米国ユニシス社で有効活用されていたMAPPER 1100と出会い，この日本市場への導入，日本語対応付加およびその後の各種サポート業務を通じて4GLとの関係を深め今日に至る．現在，マイクロプロダクト本部OAソフトウェア部所属．

日本ユニシスにおける4GL/CASEへの取り組み

Approaches by Nihon Unisys to 4GL/CASE

川 本 光 一

要 約 約10年前にソフトウェア工学的というよりは，むしろ利用者のニーズが基礎となって誕生した4GLは，プログラムの開発工程を自動化する形態とデータベースを中心にエンドユーザがデータの編集，加工そしてレポーティングを直接行う形態とに分化し，なかでもシステム開発の生産性向上ツールとして広く支持を得てきた．

情報戦略が企業の優位性を左右する今日では，品質の良い，柔構造のSIS(Strategic Information System)開発を短期間で実施する要求が高まり，一定以上の経験を持ち，高い技術力のあるシステムエンジニア，プログラマは西歴2000年には90万人不足するというデータもある．

このような環境にあって各メインフレーム・ベンダ，ソフトウェア・ベンダが各社提供しているCASE(Computer-Aided Software Engineering)ツールは，システム開発工程の各部分を支援するツールが不統一に存在し，その使い勝手，一貫性等の面から統合化がさけばれICASE(Integrated CASE)の概念が登場した．

日本ユニシスの4GLは，システム開発工程の自動化・開発生産性向上の観点から，さらに設計工程や運用／保守工程への機能追加，強化が行われる予定である．

本稿では，4GL環境におけるICASE，すなわち4GL/CASEについて「統合環境」を提供するプロダクトとしての方向性について述べる．

Abstract The fourth-generation language(4GL), which made its debut some ten years ago on the basis of user requirements rather than software engineering, has branched out into two directions; that is, toward automated program development and toward the facility with which end-users can perform data editing, data processing and make reports through their direct operation of database directories. 4GL has widely been accepted especially as a high-productivity tool for systems develoment.

Today, an information strategy tends to determine business advantages and there have been strong demands for the short-term development of loose-structured high-quality strategic information systems (SIS). Some published data even suggest that both systems engineers and programmers who are fairly experienced and highly skillful will be 900,000 people short in number by the year 2000.

This background helped mainframe computer suppliers and software vendors to place computer-aided software engineering (CASE) tools on the market, but it has been found that the tools to support each process of systems development are all different from each other in ease of use and operational consistency. So cries for their integration have arisen, leading to the birth of integrated CASE (ICASE) concepts.

The 4GLs offered by Nihon Unisys are now planned for further enhancements in systems designing and operational management/systems maintenance as well as to added functions in automated systems development and productivity efficiency.

This paper discusses Nihon Unisys' approaches to ICASE in the 4GL world; i. e. for to the creation of 4GL products which provide the 'integrated 4GL/CASE environment.'

1. はじめに

今や，世界経済は地球規模で連動し，一時たりとも止まることのない活動が繰り広げられている．それにともない企業の活動も24時間，365日，息つく暇もないほどの性急さと，顧客ニーズの多様化への対応を強いられている．このことは「情報」を第四の経営資源として捉え，「情報」を活用するための「技術」の果たす役割がますます重要になっていくことを示している．そして，多様化に対応するためこの「技術」の中でもソフトウェアの占める位置はますます大きくなっていく．

さらに，社会が複雑化し，ニーズが多様化すればするほどこれに対応する情報処理システムの開発は際限なく，また短期間に開発することが要求される．

この要求を整理すると以下のようになる．

1) 新規システムの構築および現システムへの機能追加……データ処理を通して蓄えられたデータを「情報」として経営の判断に活用する「技術」である情報処理システムの構築，すなわち事務の合理化，販売や生産管理のシステムというような「戦術レベルのシステム」から，経営環境の変化を追随する「企業戦略レベルのシステム」の構築が求められている．さらに情報処理システム部門に対する要求も，必然的に企業の戦略レベルの要求となり，企業における部門の位置づけにも影響を及ぼしている．

2) システム保守作業の増大への対応……多様化・複雑化するニーズに柔軟に対応し競合していくためには，システム保守作業を頻繁に行い，システムの陳腐化を防止しなければならない．近年では情報処理システム部門の約80％の工数が保守作業に費され，システム開発の要員不足に一層の拍車をかけている．

3) システム開発の短納期化と品質の保証……経営環境が変化するサイクルが短くなり，ニーズも多様化している今日，それらに即応するための戦略的な情報システムの開発は短納期で一定の品質が保たれている必要がある．これが実現できていないこともあり，莫大なシステム開発要求をバックログとして抱えてしまっている．

2. 4GL/CASEへの期待と役割

前章で述べたように情報システムに対するニーズは，そのまま情報システム部門に対するニーズとして，その対応を迫られることになる．

一方，開発が要求されるシステムは自社内のオンライン・システム，他企業との接続システム，データベース間での連携処理システムへと，ますます複雑で高度な知識と経験を必要とする領域へ拡大している．このことも，スキルの高い要員の不足や開発されたシステムの品質の低下に拍車をかけている．

情報システム部門は情報戦略面からの優位性を保持し，情報システムに対する要求に対応していくための切り札として4GLを採用するに至った．

4GLを採用することによりシステムの開発工程において，生産性と使用の容易性を大幅に向上させることができるので保守への負荷軽減を可能にできる．また，もう一方では情報の活用をエンドユーザ自身により展開し，各々の階層における経営の意思決定に対する支援という業務をも可能としている．

当社では，前者への対応として主にLINC II (Logic and Information Network Compiler II)，後者については主にMAPPER(Maintaining, Preparing, and Producing Executive Report)を提供し，その効果を発揮している．

ソフトウェア開発支援という範ちゅうにおいては，一般的にCASE(Computer-Aided Software Engineering)ツールとして開発工程における要求定義から基本，詳細設計，プログラミング，テストおよび運用／保守に至る各工程の作業を支援するツールが，各ソフトウェアハウスやメインフレームベンダから提供されている．通産省もソフトウェア開発の自動化を目標にしたΣプロジェクトを発足させ，今年度から本番に入っている．

表1　ソフトウェア開発支援ツールのカバー範囲
Tablel The supported process of software development

ソフトウェア開発工程	下流工程 支援ツール (Lower CASE)	上流工程 支援ツール (Upper CASE)	一　貫 支援ツール ICASE	4GL	4GL/ CASE
要求定義		↑	↑		
基本設計	↑	↓	↓		↑
詳細設計	↓			↑	↓
プログラミング	↓		↓	↓	↓
テスト	↓		↓	↓	↓
運用／保守	↓		↓	↓	↓

表1に示すように，4GLとCASEの背景にある考え方は，システム開発の各工程で，人間の意思を伝達する方法を限りなく容易にし，最終的には自動化を目指すものである．

さらに，4GLの役割はシステム開発の知識やプログラミング技術を4 GL自身が補い4GLを使う人の専門性を減じ，業務知識を中心にシステム開発を進めることを可能にすることと，データベースをエンドユーザに開放し，エンドユーザ自身の責任のもとに推進されるEUC(End User Computing)に対する強力な支援システムとなることである．

3. 当社の4GL/CASEの方向性

4GL/CASEの位置づけに言及する前に，現在のコンピュータ運用の形態と処理の実行形態についてふれる．

従来の端末とホストによる集中した開発・保守および実行という形態だけではなく，近年のME(Micro Electronics)の急速な進歩とOSおよびユーザインタフェースの標準化傾向の中にあって，図1のような3階層の分散型の開発・保守および実行という形態がとられている．さらに，マルチベンダ化傾向の中で異機種接続とデータベースを中心としたマイクロとメインフレーム連動による実行形態に進みつつある．

すなわちコンピュータ利用が高度化するほど，4GL/CASEによって開発された戦略的情報システムは複雑化した環境のもとでの実行・運用が要求されることになる．

当社の4GL/CASEプロダクトは米国ユニシス社との連携のもとに，以下のような

基本的な考え方により提供される予定である．

1) 1990年代に向けて，4GL/CASEテクノロジ分野でマーケットリーダとしてプロダクトを提供する．

2) システム開発工程全般における支援ツールとして提供する．

3) すべてのプラットフォーム・システム上で使用可能とする(シリーズ2200，Aシリーズ，UNIX，PW²等)．

3.1 4GL/CASEの位置づけ

第2章の表1に示す通り4GL/CASEはICASEと同じ範囲の工程の作業を支援するものである．そのうち，LINCとMAPPERの位置づけは図2に示すように，それぞれの強味を生かした補完的なプロダクトである．

すなわち，LINCとMAPPERは情報戦略を統合した形で実現する柔軟な手段となり得る．

LINCは，主に情報システム部門を中心にエンドユーザが参加するシステム開発に適しており，業務の一線を支援する全社レベルのオンライン処理開発の支援に強みを

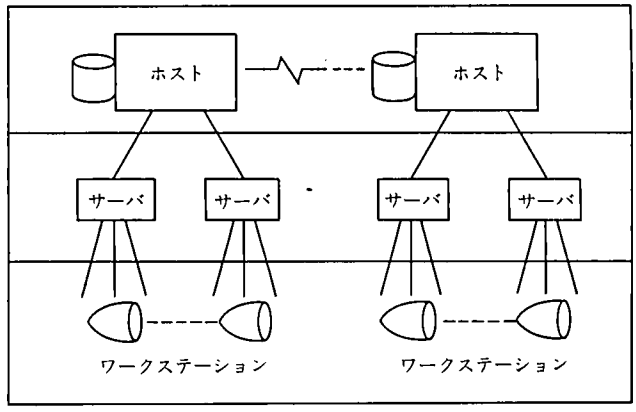


図1　三階層の運用形態

Fig. 1　Three layer's configurations

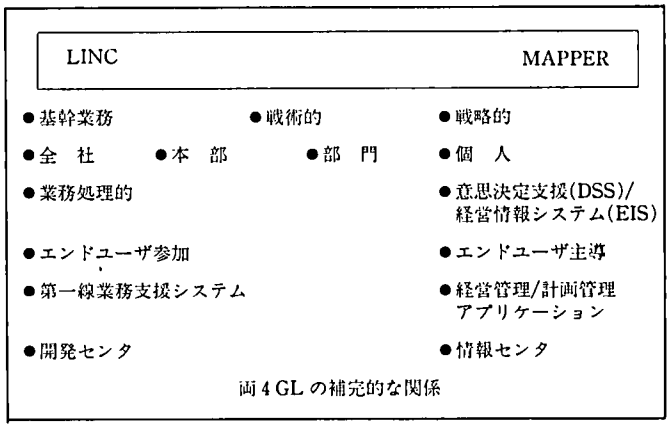


図2　LINC，MAPPERの位置付け

Fig. 2　Positionning of LINC and MAPPER

表 2　4GL/CASE のカバー範囲
Table 2　The supported process of software development by 4 GL/CASE

ソフトウェア開発工程	4GL/CASE	
	LINC	MAPPER
要求定義		
基本設計	②	⑥
詳細設計		
プログラミング	③	④ ⑤
テスト	①	
運用／保守		

① LINC II
② LDA：LINC Design Assitant(LINC システムの設計支援ツール)
③ LSA：LINC Systems Approach(LINC システム開発方法論)
④ MAPPER
⑤ MAPPER-KIT
⑥ SUM：Standard Universal Model(MAPPER システム開発方法論)

発揮する．一方，MAPPER はエンドユーザ主導のコンピュータ利用に適しており，戦略的なシステム，経営計画や管理的なアプリケーション，および部門レベルのアプリケーション開発支援に強みを発揮する．このように LINC と MAPPER は，適応形態に特徴を持っているが，システム開発における適用について，第 2 章の表 1 から 4 GL/CASE の部分を取り出し，ツール／技法の面からまとめると表 2 に示す 4 GL/CASE のカバー範囲となる．

3.2 LINC II の今後

LINC はプログラムを完全な形で生成し，システム開発工程におけるプログラミング・プロセスを自動化している．一方，開発方法論についても常にデザイン・アプローチを採用してきた．これらが一体となり，アプリケーション・システムは自動的に統合化され従来の環境に比べ，はるかに容易な保守環境を LINC II が支援してきた．

このように LINC II は，ツールの面からも方法論の面からも CASE ツールとして機能してきた．主に，LINC II は上流工程よりも下流工程にその効果を発揮してきた．

今後は，4 GL/CASE として(一般的には ICASE として)強化していくための第一段階として次のような機能の提供を計画している．

1) LINC Systems Approach(LSA)の提供……システムへの要求を分析・定義し，設計する工程を対象とする方法ではデータと処理機能について，ビジネスの実世界にあるがままの姿と同様，一緒に分析し，システムを構成するオブジェクトを定義する．

具体的にはコンポーネント，あるいはイベントと呼ばれるものである．LINC ではこれら個々のオブジェクトに対して，データに関する要求と処理機能に関する要求を一つのものとして記述する．

このように LINC のデザインは，より完成度の高いシステムを作り上げるため，異なった設計対象を効果的に結合し設計を進めていく．この後の工程で，

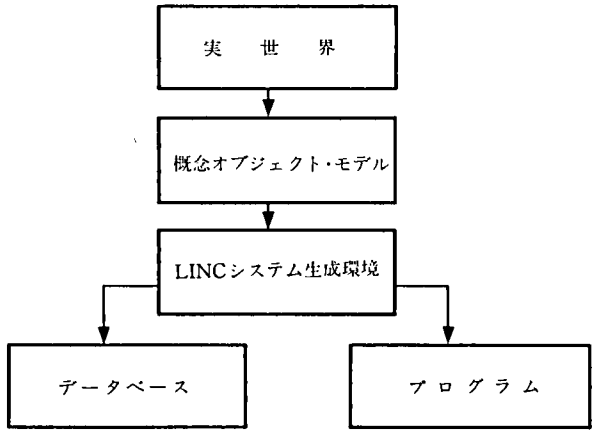


図3　LINCオブジェクト・モデリング
Fig. 3　LINC object modeling

LINC IIは直接データベースとプログラムを生成するため，常に完全な同期が保証される(図3)．

2)　LINC Design Assistant(LDA)の提供……LINC IIの利用者がLSAを活用することを支援するプロダクトである．すなわち要求定義段階でLINCオブジェクト・モデルをワークステーション上に図形表示形式で作ることができる．

開発者は，各々のオブジェクトに関して決定した情報をモデルに追加しLDAに取り込む．その後リポジトリ*に引き渡され，リポジトリを通してLINCシステムに直接引き渡されるので設計情報の重複入力は発生しない．この後，開発者はLINC II本体を使って開発を続けることができる．

またLDAにはリバース・エンジニアリングの機能がある．ユーザはこの機能を使用し，以前にLDAを使わずに開発したLINCシステムでもオブジェクト・モデルのダイアグラムを生成できる．したがって，オブジェクト・モデルの属性や関係を調査する上で有益であり，その後の保守の段階でもLDAは重要な役割を果たすことになる．

3)　LINC CASE Interface(LCI)の提供……現在多くのソフトウェア・ベンダ各社が提供している技法や上流工程CASEツールを使用しているユーザが，使い慣れたツールや技法を継続して使用したい場合，それらをLINC IIの環境に統合化するインタフェース機能である．

設計段階の後，LDAやそれ以外の上流工程CASEツールの情報をユーティリティによりLINC CASEインタフェースの標準形式に変換し，LINCリポジトリに取り込む．その後は，LINC IIの開発環境のもとシステム開発をすることによって，より完成した統合情報システムを生成することができるようにするものである．

3.3　MAPPERの今後

MAPPERはエンドユーザ・コンピューティングの旗頭として4GLの中でも特筆

* リポジトリ：ペインタ機能やエディタ機能を使用して入力された情報，生成に関する情報等のアプリケーションの設計や開発・保守に必要なすべての情報を管理しているデータベース．

すべき地位を築くと共に，MAPPER-KIT および SUM との連携によりそのシステム開発支援能力の高さの面からも利用者の多いシステムとなっている．

90 年代に向けて次に示すような機能強化を計画している．

1) 当社提供ハードウェアすべてへの MAPPER 移植……パソコンからメインフレームに至るユニシスの主要アーキテクチャの各システムで MAPPER の利用を可能とするものである．

すなわち，以下のような特長を生かした活用が計れるようになる．

① 基幹システム，部門システム，個人システムという 3 階層のラインアップが MAPPER だけで構築でする．

② 3 階層の MAPPER を用途に応じて使い分けることが可能となる．しかも，同一操作であるためエンドユーザにとって慣れや教育面の負担も軽くなる．

③ MAPPER アプリケーションの可搬性が増す．

④ 階層間を有機的に接続するためのネットワーキングが強力になる．

このように，広範囲のシステムに単一ソフトウェアを利用可能とすることで MAPPER ワールドにおけるどのような分散／集中形態における利用も可能となる．

2) Application Power Tool(APT)の提供……MAPPER のアプリケーション・システムを開発し管理するための一連のユーティリティ群であり，以下の機能を提供する．

① 端末の画面とメニューを生成するユーティリティの提供

② ディクショナリ，ディレクトリとともに，項目とアプリケーションのドキュメントを作成する機能

③ ディクショナリとディレクトリの作成・保守，およびデータベースのリフォーマット等を行う機能

3) MAPPER Relational Interface(MRI)の強化……MRI は開発部門とエンドユーザの両方での活用を前提に開発された機能で，次のような機能強化を計画しており，MAPPER とともにユニシスの主要なアーキテクチャに搭載していく．

① RDI(Relational Data Interface)

MAPPER ユーザは SQL コマンドを知らなくても，メニュー画面を通してリレーショナル・データベースを素早くアクセスすることができる．また，SQL コマンドを使い慣れたユーザであれば，SQL コマンドを入力したり，MAPPER ランとして事前登録した SQL を呼び出すことでリレーショナル・データベースを MAPPER から直接アクセスすることができる．

② ORACLE，DB 2 へのアクセス

今後リリースされる予定の UNIX* 用 MAPPER からは，U 6000/5000 シリーズ上の ORACLE** データベースやネットワークを通して他 UNIX 上の ORACLE データベースをアクセスする機能を，また IBM 社の DB 2*** に対

---

* UNIX：AT&T ベル研究所が開発し，AT&T がライセンスしている．

** ORACLE：オラクル社のリレーショナル・データベース．

*** DB 2：IBM 社のリレーショナル・データベース．

してはSNA* ネットワークを通じてアクセスする機能を提供していく．

4) MAPPERのCASEツールとの統合……エンドユーザ・コンピューティングプロダクトとしてのMAPPERに対し，CASEとしての特徴を付け加えると同時に，将来にわたってLINCと統合するための機能強化を図ることを考えている．すなわち，パーソナルコンピュータ上でMAPPERランのデザインを図形表示で行えるツールの提供やLINCのデータベースとMAPPERが連携するようになる機能の提供等である．

4. おわりに

約10年前，利用者のニーズから生まれた4GLは，利用者の厚い支援を受けてきた．さらに今日では，複雑で多様化した要求に対応すること，SEの慢性的な不足，短納期対応そして成果物の品質の確保等，4GLの要求に対し，それは「言語」の範囲を越えて要求定義から本番実行，保守等の環境の設定と運用管理までの広大な範囲を支援する「統合環境」の提供へと変化している．

すなわち，企業における情報戦略を具現化するためのインフラストラクチャとして，開発・実行および運用面での操作性の容易さ，一貫性とともに，アプリケーションの可搬性，接続性の良さ等を統合した環境として提供していく．そして4GLは，CASEツールと一体化した製品として90年代に対応するプロダクト群の中核に据える．

当社は長年，培われてきた技術力と経験を持っており，今日の4GLリーディング・カンパニーの地位を保ちつつ，4GL/CASEに関するリーディング・カンパニーとしてプロダクトを提供していきたい．

---

* SNA：IBM社のネットワーク・アーキテクチャ

参考文献　［1］ R. C. Goyette, Computer Aided Software Engineering(CASE), New Product Announcement, Unisys Corp., Oct., 1989.

執筆者紹介　川　本　光　一 (Kouichi Kawamoto)
1970年武蔵工業大学卒業．同年日本ユニシス(株)入社．主として製造工業のユーザ・アプリケーション・システムのサービスに従事．現在商品企画本部システム・プロダクト商品企画部に所属．

MAPPER利用の現状と効果的な適用方法

Current MAPPER Usage and Its More Effectual Application

松 木 規 子

要 約 MAPPERは，4GLの中でも歴史が長く，実績も定評もある．しかし，日本ではMAPPER本来の特徴を活かしたエンドユーザ・オリエンテッドなシステムとしては，十分に活用しきれていない面があった．

MAPPERの特徴を活かすためには，システムを従来より広く柔らかく捉える必要がある．ここで提案するシステムは，システムの中心にその業務の中核(アプリケーション・コア)があり，そのまわりを利用部門の担当者の仕事がとりまく形をしている．さらに，コアとその周辺にエンドユーザを支援するための情報提供ツールが配される．担当者の仕事に対しても各種の支援機能(部品ラン)が提供され，担当者による自由な組み合わせで柔軟なシステムの実現をめざすものとなっている．

今後は，このような考え方に基づき，適用技術やツールの整備・普及を図ることが重要と考えられる．

Abstract Among the existing fourth-generation languages (4GL), MAPPER has a longer history as a widely accepted and field-proven software product. In Japan, however, it has yet to be utilized as an end-user-oriented sysem to a satisfactory extent in building a system where MAPPER's own characteristics are fully adopted.

The most effective use of MAPPER requires us to regard a system as being much less limited and less tight. The system proposed here provides an environment where the application core exisits in the center of a system with its neighborhood surrouned by end-users who are responsible for their applications. In addition, information-providing tools to support end-users are placed between the core and the neighborhood (end-user applications). A wide range of support functions (parts runs) are also offered to end-users to enable them to create a flexible system in free combinations.

It would to be important that we try to further enhance application techniques and tools based on such concepts so as to make them accepted more widely in the future.

1. はじめに

コンピュータの商用利用も四半世紀を超え，情報システムを取り巻く環境の変化，つまり，業務の多様化・高度化，処理の分散化とともに，従来のシステム開発，運用体制が見直されつつある．それにともなって，積極的にエンドユーザの力を活用するエンドユーザ中心のシステムアプローチの考え方が提唱され，このための有効なツールとして4GLが期待されている．

MAPPERは1981年以来，数多くの客先のビジネスに関わり，現在も動き続けているが，この数年間でMAPPERをとりまくユーザニーズも大きく様変わりしてきている．

ここでMAPPER利用の現状を把握し，現状の抱える問題点を明らかにした上で，

今後より効果的に MAPPER を適用するためには，どのように考えていけば良いかについて考察を行う．

2. MAPPER 利用の現状と問題点

2.1 利用の現状

MAPPER は，当社の客先で業種・業態を問わず 850 システムと幅広く使用されている．これは，基本ソフトウェアをのぞくと唯一といって良いほどの広がりである．したがって，すべての MAPPER ユーザの状況を詳細に知る機会は少なかった．

このため，1988 年 9 月，主に MAPPER 1100 の客先のうち 132 社についてアンケート形式で現在の MAPPER 利用状況の調査を実施した．

この調査結果から MAPPER 利用状況を見たとき，特筆すべき事項として以下のことが挙げられた．

1) アプリケーション数，端末数が少ない……アプリケーション数，端末数が予想以上に低い値に片寄っており(図 1，図 2)，MAPPER 利用が特定業務，特定部門

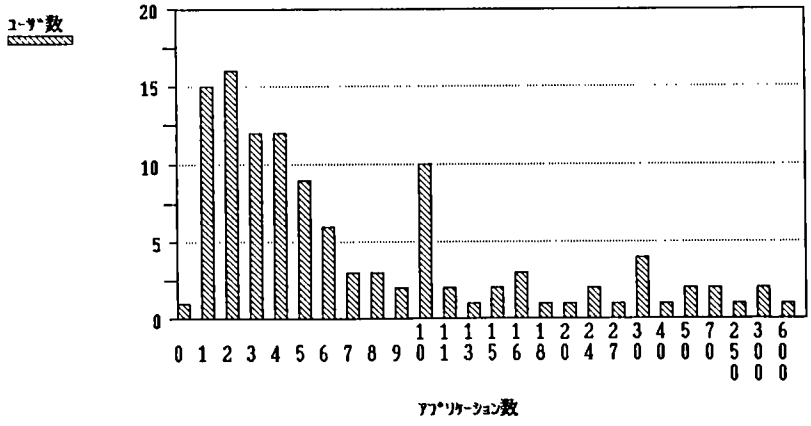


図 1　アプリケーション数の分布

Fig. 1　The distribution of the application systems

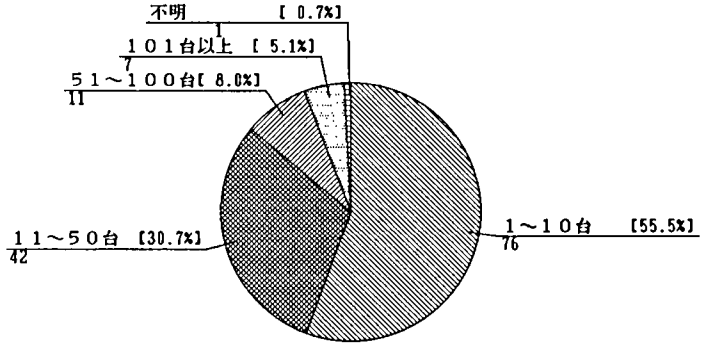


図 2　使用端末数の分布

Fig. 2　The distribution of the MAPPER terminals

のみで凍結してしまい，それ以上に拡大しない傾向が見られる．たとえばアプリケーション・パッケージにより，MAPPERが利用されていても，アプリケーション・パッケージに関連のない業務や，他部門に対してはMAPPERの利用拡大が行われるケースが少ないことがわかる(表1)．

表1　AP数，端末数の分析

Table 1　The analysis of the application systems and the MAPPER terminals

	採用理由に「APパッケージの適用」あり　39ユーザ		採用理由に「APパッケージの適用」なし	
AP数* ユーザ数	0, 1, 2	3以上	0, 1, 2	3以上
	10 (40%)	25 (60%)	22 (27.8%)	57 (72.2%)
端末数 ユーザ数	1〜3	4以上	1〜3	4以上
	9 (23%)	30 (77%)	19 (20%)	79 (80%)

*「AP数不明」のユーザを除く

2)　会話機能の使用率が低い……採用理由(図3)としてエンドユーザ・コンピューティング(E. U. C)を挙げているユーザの会話機能利用率を見ると，業務での会話機能の利用率が51%以上のユーザが13.4%あるが，逆に利用率5%未満のユーザも13.5%と，同程度の数値を示している(図4)．しかし，全体を見ると，会話機

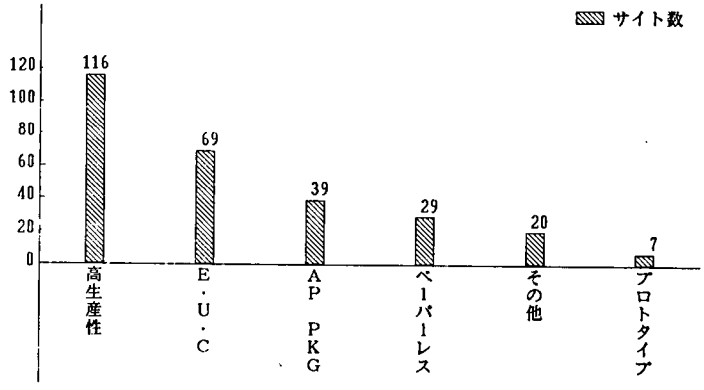


図3　採用理由

Fig. 3　The reason of the adoption

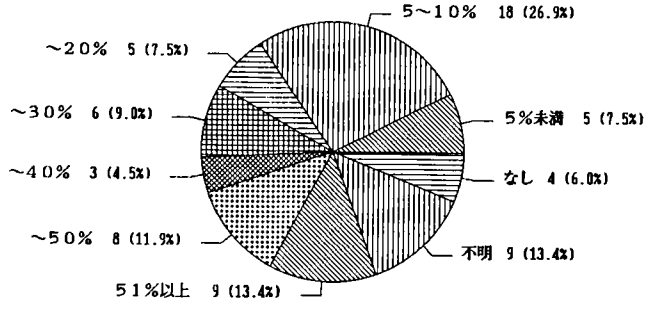


図4　「エンドユーザ・コンピューティング」を採用理由にあげているユーザの会話機能利用率

Fig. 4　The percentage of manual function use within the limit of "End-user computing"

能を使える人が5人以下のユーザが約半数を占めている(図5)．また，業務での会話機能の利用率が5％未満のユーザが3割を超える(図6)という結果が出ている．

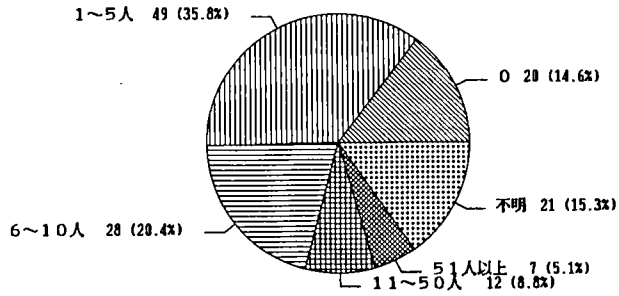


図 5　会話機能の使える人数

Fig. 5　The number of people who can use manual functions

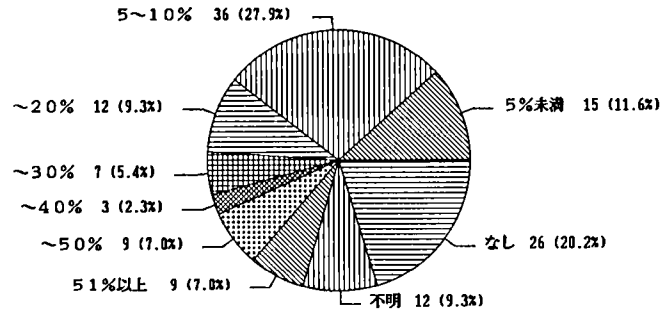


図 6　会話機能利用率

Fig. 6　The percentage of manual function use

3)　COBOL の代わりの高生産性開発ツールとして使用されている……採用理由に生産性の高さを挙げている客先が非常に多い(図3)ことや，開発形態が当社(NUL)受託，次いで客先の情報システム部門となっていることから(図7)，従来の COBOL によるシステム開発をそのまま MAPPER に置き換えて開発してい

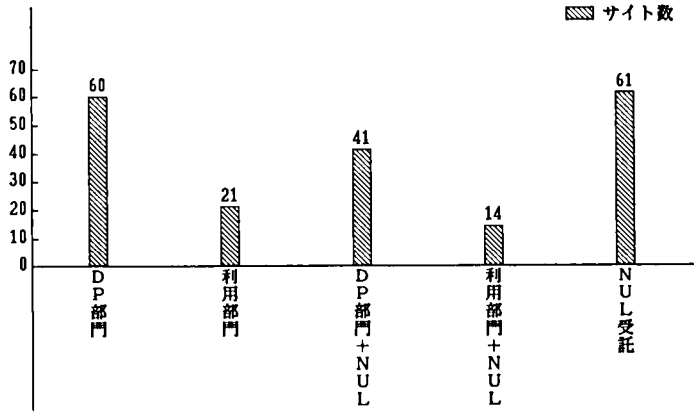


図 7　開発形態

Fig. 7　The section in charge of an implementation

ることがうかがえる．結果，利用部門の参画は極端に少なくなっている．

4) 実行効率が不十分とするユーザが多い……実行効率に対する評価を，採用理由にエンドユーザ・コンピューティングを挙げているか否かで分類してみると，表2のようになる．開発ツールとして捉えているユーザほど実行効率に関して問題視しているところが多い．実行時の効率は，対象業務の種類や性質・運用形態により，求められるものは一様ではないので，一般論として論じることはむずかしい．MAPPERを開発ツールとして利用しているところで，COBOL等3GLのリアルタイム・システムと同等の効率を要求しているケースがあり，多面的に利用されるMAPPERの苦しい一面を表している．

表2 実行効率が不十分と答えたユーザ
Table 2 The number of the users who answered "MAPPER' performance is inefficient"

実行効率が不十分と答えているユーザ	
採用理由に「EUC」あり	採用理由に「EUC」なし
67社中7社	70社中16社
10.4%	22.9%

EUC：End-User Computing

5) MAPPER適用はおおむね成功と思われている……MAPPERの適用そのものが失敗だったという回答は非常に少ない(図8)．これは，たとえば短納期を厳守するために実行効率をトレード・オフするといったような意見に代表されるように，あらかじめMAPPERの特徴のどの部分を活かすかという見極めがあり，その範囲内では十分に効果を発揮していることによると思われる．

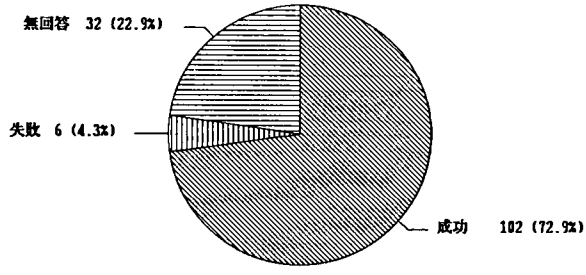


図8 MAPPER適用の成否
Fig. 8 The result of MAPPER utilization

6) その他の効果……利用部門自身がMAPPERを利用することによって
- 情報システム部門に無理な要求をしなくなった，
- 自分たちでいろいろ工夫するようになった，
- 情報システム部門の負荷が軽減された，

という効果があがっている．

2.2 利用拡大のための阻害要因

「MAPPERを効果的に十分活用する」という観点から，MAPPER利用の現状を見ると，主な問題点としては次の2点を挙げることができる．

まず第一に，MAPPERたる所以の会話機能があまり利用されていないということが挙げられる．この理由としては，次のようなことがアンケートに寄せられている．

1)　そもそも会話機能を利用するという発想が少ない
2)　ソフトウェアの機能に問題がある
3)　利用部門を巻き込む条件・環境が不足している

ソフトウェアやエンドユーザ側にある程度問題があるのは事実としても，最大の問題点は会話機能利用の発想が少ないという点である．これは，利用状況調査の結果にもあったように，MAPPERを単にCOBOLの代わりのプログラム言語としてしか見ていないところに起因していると思われる．

たとえば，当社が提供しているアプリケーション・パッケージでさえ，MAPPERが本来得意としているはずの非定型のアドホックな処理に柔軟に対応できるようにはなっておらず，最終的なエンドユーザにとっては，MAPPERでできていようがCOBOLでできていようがまったく同じというケースが多い．そのうえ，MAPPERの特徴をよく理解せず，COBOLの代わりにCOBOLと同じような考え方でファイル設計や処理設計をすれば，残念ながら効率問題が発生するのは当然のことと言えるかもしれない．

MAPPERが発表された当時は，コンピュータの利用範囲が拡大し，またOA化の波に乗ってエンドユーザからのシステム化要求が激増しつつあった．これに対し情報システム部門は大量のバックログを抱え3GLを唯一のツールとして，いわば孤軍奮闘していた．このような状況下で，MAPPERの高い生産性が注目され，COBOLに代わるバックログ解消の切り札的なツールとして利用されたのも無理からぬことであった．この結果，客先の業務システム化が進んだわけであるが，最近は情報システム内に蓄積されることになった各種の情報をさらに高度に活用することが求められている．このような要求に対して，単にCOBOLの代わりというだけでは，MAPPERの特徴を活かしたシステムとして不十分であろう．

MAPPERが世の中に出てきて以来言われ続けているMAPPER本来の使い方，つまり「利用部門の業務担当者自らが自らの手で処理する」という使い方を実践し成功をおさめている客先もあるが，現実には少数派である．なぜならば，従来この種の成功をおさめるためには，客先や客先担当のSEが，手探りの中から自分達だけでMAPPERの本質を十分理解し新たな発想に挑戦する，という困難な努力が求められたからである．しかも，いくつかのこの貴重な客先やSEの努力の結果が十分に体系化・明文化されているとは言い難いのが現状である．しかしMAPPERのように，非常に多面的な道具を多くの人が上手に使うためには，明確な使い方が誰にでもわかるように提示されていなければならない．この点MAPPER適用に対する基本的な考え方やノウハウ，ツールの体系化・民主化が強く求められる．

さらに，従来は「エンドユーザにフレンドリなMAPPERの会話機能を有効に使うシステムがMAPPERらしいシステムだ」という程度の指針しかなかったため，開発ツールとしての利用が大部分を占め，適用技術やノウハウもプログラム開発という点にかたよっている．今後はエンドユーザが自ら活用するための利用方法に関する技術力を高め，ノウハウを収集・蓄積し体系化していくことが求められる．

第二に，言葉の定義が不十分なために，ある種の誤解を含んだままMAPPERが適

用されているということが挙げられる.

一つには「簡単」という言葉がある.これは,MAPPERを説明する際必ず使われる,「誰でも簡単にすぐ使えます」というキャッチフレーズである.この一文は,正しく言えば,「業務に精通した利用部門の人であれば誰でも,簡単に理解し実行できる会話機能を利用して,すぐ業務の一部を実行することに使えます.」ということであろう.ところが,これが明確になっていないために限りなく拡大解釈されて,新人でも素人でも周到な計画や準備もなしで,簡単にシステム開発ができると思われてしまった結果,

1) MAPPERは簡単だと思って採用したのに,予想以上の工数が掛ってしまった,

2) 開発時にMAPPERを熱知している人が必要なのは問題だ,

といった声が上がるようになってしまったのである.

確かに従来の3GLに比べれば,MAPPERの機能の習得は容易である.しかし,利用部門が自らの仕事にMAPPERの会話機能や一部ランを適用するということと,品質や効率等を十分考慮したアプリケーション・システムを開発する時の道具としてMAPPERを利用するということは,必要な知識・技術の修得の面でも本質的に異なるのである.

次に問題になるのは「エンドユーザ」という言葉である.「エンドユーザ・コンピューティングといってもエンドユーザは定型的な仕事しかしていないので,会話機能を使う必要がない.」といった意見がアンケートに寄せられている.しかし,これはエンドユーザのごく一部の人だけを見た狭い理解であろう.

MAPPERのエンドユーザというのは,MAPPERでできたシステムの端末を操作するだけの人ではなく,各種の情報をいろいろな角度から利用し,その時々に合わせた判断を要求される人達,MAPPERの会話機能を自らの仕事に活かせる人達であると解釈すべきである.

もう一つは「会話利用」と言うことのとらえ方の問題である.会話利用と言うと何が何でも会話だけと思ってしまう人が多い.アンケートにも,次のような声があがっている.

1) 会話だけでできるユーザの業務はない.

2) 会話だけでやるにはユーザの仕事は複雑すぎる.

しかし,MAPPERで実現されるシステムの世界はすべて会話か,逆にすべてランといったものではないのである.従来会話機能とランをいかに上手に組み合わせて活用するかといったことはあまり真剣に考えられていなかったといえる.

このような現状を踏まえ,MAPPERの今後のさらなる利用拡大を図るためには,曖昧だった言葉を明確に定義し,さらにあまり議論されてこなかったMAPPER適用のコンセプトを構築することが重要である.ここで,できる限りMAPPERの原点に立ち返り,これまでのMAPPER適用の経験を活かし,システム構築に際してMAPPERの良さを最大限に活用するためのコンセプト,技術,ツールを考えてみる.

3. MAPPER活用システム

3.1 基本的な考え方

まず,従来のコンピュータ・システムについて考えてみる.

一つの業務を，縦に仕事の流れを意識して図式化すると図9のように捉えることができる．ある仕事をするために必要な準備作業(G)があり，そこから一定のルーチンワーク(P)がつながり，ルーチンワークの結果に対して確認や判断，分析，事後処理(U)が発生する．従来のコンピュータ・システムは(P)の部分を情報システムとして捉え，コンピュータ上で実現してきた．そのために(G)と(P)の間にはコンピュータに対する情報提供の仕事(I)が発生し，(P)と(U)の間には，コンピュータからの情報提供(O)が行われる．この業務そのものは，担当者の作業としては(G)と(U)が並行して行われたり混在しているものである．

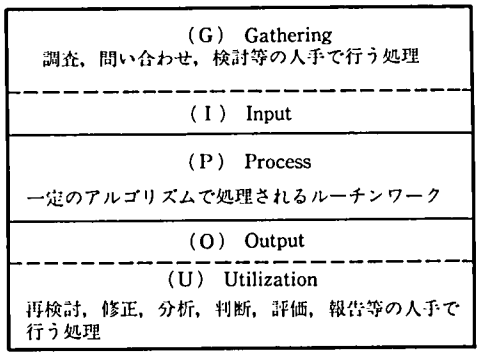


図 9　業務の図式化

Fig. 9　The diagram of the application

これを図式化すると図10のように表すことができる．そして多くの業務では(G)や(U)の部分はかなり変化するものであり，図10での外側の輪は時間の経過や世の中の変化にともない，少しずつ半径が大きくなったり楕円になったり歪んだりしながら形を変えていくものと思われる．

これに対し，従来のコンピュータ・システムというものは，ある時点で輪を固定し，図9の形に再定義し構築していく．このため，時間が経ち業務の形が変化し，必要とする情報や作業が増減すると徐々に使い勝手が悪くなり，ある時点以降，保守，改修に追われることが多い．とくに，エンドユーザのコンピュータに対する情報提供要求はその変化の度合が速く，したがって(O)の部分の厚みがどんどん増し，この工数が情報システム部門を圧迫することになる．

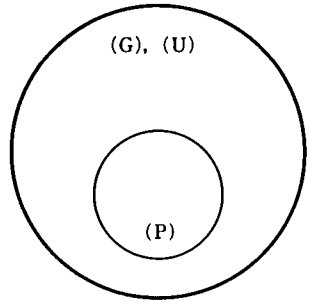


図10　担当者から見た業務の図式化

Fig. 10　The diagram of the application which is viewed by end-user

また，従来の情報システムでは(G)や(U)の中で発生するさまざまな情報はその範ちゅうに取り込めずにいた．MAPPERの出現により(G)や(U)の中の定型的な部分はシステム化されたが，エンドユーザに固定的な情報を提供するという従来のシステムと同じ考え方で作られているため，エンドユーザの要求の変化への対応が絶えず求められる．また，非定型の仕事はコンピュータにのせづらいということで，相変わらず切り貼りをしてコピーをとるといったことが繰り返されているのが実情ではなかろうか．

次にMAPPERを活用したシステムをどのようにとらえるかについて述べる．MAPPERでシステム構築を考える場合，情報システムを従来よりも柔らかく広く捉えるということがポイントとなる．図9のほとんどすべての部分を情報システムとして捉えるのである．(P)のみでなく(G)から(U)までを広く，外側の輪の変化をある程度吸収することができるくらい柔らかく捉えたシステムを絵で表すと図11になる．

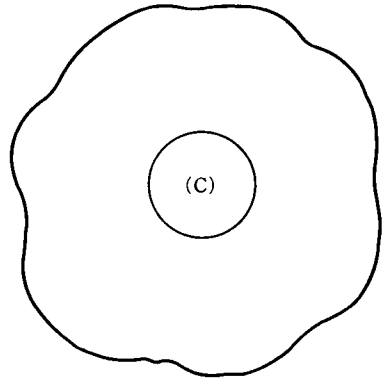


図11　広く柔らかく捉えた情報システム

Fig. 11　Comprehend flexible the information system

この中心(C)の部分は，この業務の幹となるべき処理，時間の経過とともに業務の輪の形がぶれたとしてもずれることのない軸であり，業務の中核＝アプリケーション・コア(application-core)と呼ぶ．そしてこのコアの周辺に流動的な非定型の処理がコアを取り巻く形で成立している．コアはその業務の基幹ともいえるものであり，コア自身は必ずしもMAPPERで構築される必要はない．業務の特質や，処理量，データ量や，その会社の方針によって3GLやLINC等で構築されることも十分考えられる．周辺部分のダイナミックな変更に対しては，MAPPERの会話機能を効果的に適用することによって初めて対応可能になる．したがって，MAPPERを活用するシステムには会話機能は必須である．しかし，このコアの周辺部のシステム化に効果的に会話機能を適用させるためには，エンドユーザを支援するためのいくつかの仕組みが必要である．ここで言う仕組みとは，エンドユーザの支援についての考え方や，利用環境を整えるための仕掛，エンドユーザに対する教育ツールや教育システム等のことである．従来はこの仕組みが明確でなかったために，図11に記したシステムがなかなか現実のものにならなかったといえる．

ここでは，上記の利用環境を整えるための仕掛について考えてみる．エンドユーザ

が MAPPER を活用して実現する柔軟なシステムを可能にするための仕掛の最低限の要件としては，

1) エンドユーザにとって有益な情報が，エンドユーザが理解しやすい形のレポートになっていること，
2) エンドユーザにとって必要な情報がどこにあるかわかること，
3) エンドユーザが安心して使えるような環境が設定されていること(これは同時にシステムやコンピュータを管理する側にとっても重要なことである，
4) アプリケーションの性格上，コアで使用するレポートがエンドユーザにとってなじみにくいものであるとき，またはコアそのものが MAPPER 以外で構築されているとき，コアで作られた情報をエンドユーザにわかりやすい形に加工して提供する機能があること，

等が挙げられる．これらの仕掛をコア・インタフェースと名付け，図 11 に反映したものが図 12 である．

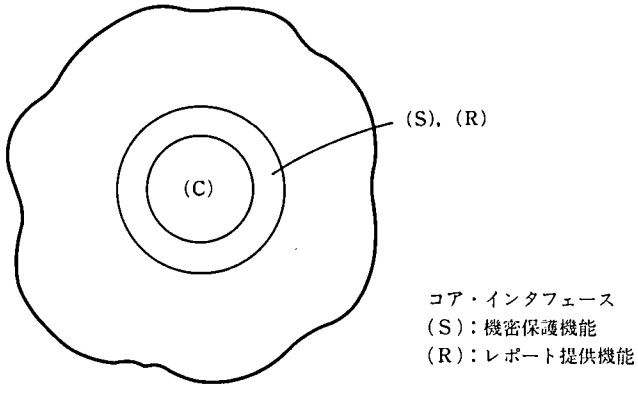
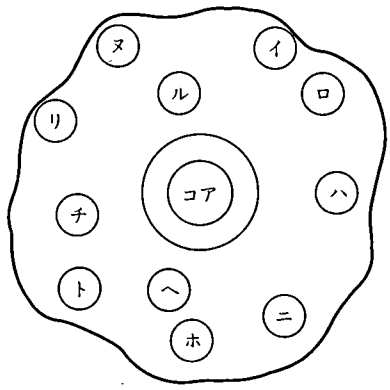


図 12　図 11 にコア・インタフェースを付け加えた図
Fig. 12　Fig. 11+Core interface

コア・インタフェースは機密保護機能(S)とレポート提供機能(R)から構成される．

これでエンドユーザに安全な環境と使いたい情報が提供される．しかし，前述したようにこの周辺部にも定型的な処理や煩雑な作業は存在するので，必要な情報が提供されたからといってその後のすべての処理を会話機能で実現するというのは現実的ではない．

そこでランが必要になるわけであるが，ここで重要なのは，このランをメニューを選択したりデータを入力するだけで最終的な結果が得られるといった完成品とするのではなく，エンドユーザによって入出力や機能が幾通りかに組み合わせることができるような，完成品に対して部品に当たるランとして作成していくことである．部品ランを組み合わせることによって，周辺部の定型的な処理に対応することができ，さらに個々の部品として非定型の処理からも随時利用できる．部品ランは汎用部品(複数業務で共通に利用できるもの)と業務部品(特定業務でのみ利用されるもの)の 2 種類に大別される．このような部品をちりばめることによって MAPPER 活用システムの全体像が完成するのである(図 13)．



イ：二次元表編集（汎用）
ロ：縦横変換（汎用）
ハ：罫線印書（汎用）
ニ：簡易フォーマット変換（汎用）
ホ：標準偏差計算（業務）
ヘ：個人情報表示（業務）
ト：請求伝票出力（業務）
チ：エラー・データ出力（業務）
リ：　　：
ヌ：　　：
ル：　　：

図13　MAPPERシステムの全体像

Fig. 13　The whole image of the MAPPER application system

3.2　MAPPER活用システムの構築モデル

次に3.1節で述べた考え方をある業務に当てはめてみる．ここではMAPPERによるシステム化の中で最も数が多い人事情報システムを取り上げる．

システム全体の柔軟性を高めるために，人事情報システムのコアを可能な限り小さく捉えると，人事データベースの維持につきる．人事情報は各種の業務から発生するが，それらすべてをコアとして捉えるのでなく，人事データベースの更新部分だけをコアとして定義すると図14となる．このコアで更新・保守されたデータがエンドユーザの行う各種業務に情報として提供されたり，逆に業務の中から更新情報が発生しコアに引き渡されるのである．そのためには，各種業務から発生する情報を漏れなく収集する機能，決められたタイミングで矛盾なくデータを更新する機能，データの正当性を検証する機能が必要になる．

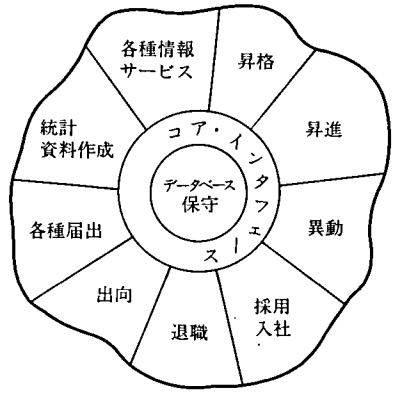


図14　人事情報システム

Fig. 14　The information system of personnel affairs

データを保全するために必要とされる機密保護の機能と，エンドユーザが自分の仕事に必要な情報をわかりやすい形で提供するレポート提供の機能が，コアの周りを取り巻く形になる．とくに人事情報のように，1件のデータの項目が非常に多いものには，MAPPER の行長の制約から，必要な情報を必要な時にレポートの形式に編集するというようなレポート提供の機能が必須である．この機能はそれだけで人事部門の他部門に対する情報提供サービスのほとんどの部分を満足することができると考えられる．

図 14 の一番外側の部分が人事部門の日常業務になる．たとえば退職業務はレポート提供機能にてコアより退職予定者の情報を得て，これに対し各種処理を実行し，必要情報をコアへ戻すといった流れになる．もう少し複雑なものの例として，異動業務をとりあげてみる．

異動の業務の流れを表すと大略図 15 のようになるが，ここでわかるように最終的な

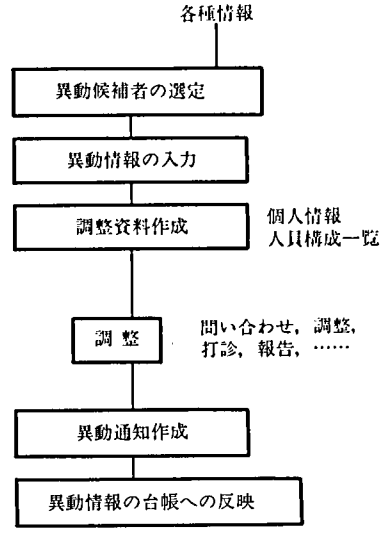


図 15　異動業務の流れ

Fig. 15　The process flow of personnel changes

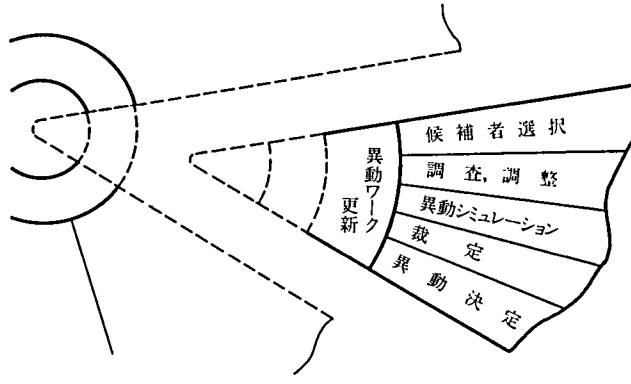


図 16　異動業務

Fig. 16　Personnel changes

情報が確定するまでに何度も更新情報が変更される可能性が大きい．変更情報は発生する度にそれをコアへ引き渡すのでなく，異動処理用の一時的なレポートに蓄積するのがよい．この一時的なレポートの作成および更新，そしてコアへの引渡しを異動業務の局所的なコア(仮想コア)と見ることができる(図16)

この仮想コアの周辺では会話機能，汎用部品(各業務共通に使用できるもの)，業務用部品(その業務でのみ使用できるもの)で仕事の流れを実現していく．たとえば異動決定の処理の中には，最終決定事項の確認(確認用資料の会話機能での作成)，決定事項の再修正や一部他部所への事前連絡(会話機能でのデータ修正)，連絡用資料の作成・出力(紙，FAX等)，辞令発布(辞令編集出力処理(ラン))といった処理があり，この中で会話機能のみでは対応がむずかしいものに対してラン(部品ラン)が作られる(図17)．

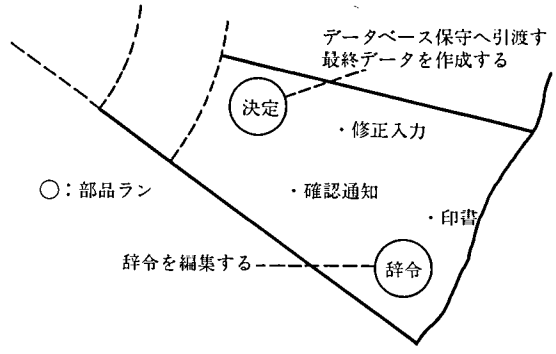


図17　異動決定

Fig. 17　The decision of personnel changes

このようにエンドユーザの仕事全体を視野に入れ，コアとコア以外に分け，仕事の適正に応じて会話機能かランに振り分けていくというのが基本的な考え方となる．

3.3　MAPPER活用システムの構築のポイント

このようなMAPPER活用システムの構築を目指すとき，情報システム部門には新たな技術や役割が求められる．

その構築にとってまず第一に重要なポイントは，情報システムのコアをいかに見極めるかということである．コアの見極め自体は，その業務の本質を理解し将来への見通しを立てうるエンドユーザが主体となり，情報システム部門が技術的な見地から支援を行う形になる．MAPPERの特徴を活かして柔軟なシステムを実現するためには，このコアを時間が経過して業務の形が変化してもぶれない軸の部分として必要最小限にとどめることがポイントとなるが，コアとしてとらえるべき範囲の適正化についての考え方，方法に関して今後の実践とノウハウの蓄積が必要である．

この他にも，MAPPER活用システム構築のポイントとして，以下の四つをあげることができる．

1)　レポートをエンドユーザに提供するインタフェース機能の設計……エンドユーザに対してどのようなレポート提供インタフェースを持つかで，そのシステムの使い勝手の善し悪しがある程度決まってしまう．この重要なインタフェース機能

を設計する上でのポイントとなるのは次の2点である．

一つは必要な情報がどこにあるか，またレポート上の項目がどういう意味を持っているか，といったことがエンドユーザにスムーズに提供されるということである．どこに何があるかをエンドユーザが覚えていないといけないということでは使いづらいものになってしまうし，またエンドユーザにとってレポート上に存在する項目がどういう意味を持っているかを正確に理解することがデータ活用のために不可欠である．このような情報をエンドユーザに提供するには，データの説明書のような一覧表やレポートの所在を示すインデックス・レポートを持つといったことが考えられる．

従来はアプリケーションの機能の一部としてこういったことが実現されているが，ツールとしてこのような機能を提供しているものにMAPPERKITがあげられる．とくにインデックス・レポートは保守が適宜行われていないとすぐ意味を失ってしまうが，MAPPERKITではインデックス・レポートの保守機能もMAPPERKIT自身が行うのでエンドユーザはとくに意識する必要はない．なお，レベル33以降MAPPER本体もこのような方向に機能強化が図られている．

もう一つのポイントは，コアからエンドユーザに対してレポートを提供する機能である．このレポートの提供の仕方には2通りが考えられる．一つはあらかじめレポートの形式を決め定期的にエンドユーザに与える方式，一つは必要な都度エンドユーザが操作することで必要な情報を集める方式である．情報パターンには定型的なものと非定型なものが混在するので両方の方式が提供されることが望ましい．

またコアがMAPPERで構築されているか，MAPPER以外で構築されているかは，仕掛の作り方が異なるだけでエンドユーザから見たレポート提供機能としては同様の方式に分けられる．

既存のツールの中で，この機能を提供しているものとしては次のものがある．

・コアのデータベースがMAPPERの場合
　MAPPERKIT
　MAPPER　ユーティリティ
・コアのデータベースがMAPPER以外の場合
　MAPPER/IDBKIT
　MRI機能

2)　エンドユーザに理解しやすい形のレポートの設計……従来のレポート設計は，ともすればデータベース設計としてコンピュータにとって処理しやすい形が追求されてきた．しかし，今後はこれに加えてエンドユーザつまり人間にとって理解しやすい，使いやすいレポートとはどのようなものかという視点が必須になってくる．これに対応するために，エンドユーザ自身が試行錯誤的に設計できるような，レポート設計支援ツールといったものの提供が効果的であろう．

3)　部品の持ち方……部品ランで柔軟なシステムを構築するためには，ユーザの要求を一連の処理を実現するプログラムの固まりとして定義するのではなく，その処理が業務の中で本当にどのような役割かをつかみ，いくつかの機能としてとら

え，さらに業務全体を見ながら洗い出した機能の統合化を考えていく．このためには，どの処理をどういう部品ととらえるかを明確にし，ユーザの要求を部品として再構築していく技術が必要になる．さらに開発時には，ランの実行方法やユーザ・インタフェースのあり方，リザルトの使い方等に十分工夫の余地があり，このあたりをどのように使いこなすかという技術や，エンドユーザの仕事・スキルに合わせて使えるように部品を作る技術等，高度な技術が必要になる．

部品ランは汎用部品と業務用部品がある．突詰めて言えば，汎用部品はMAPPERそのものの機能として存在すべきものと言えるが，MAPPERの機能と同じインタフェースを持つユーザ部品を作れるところがMAPPERの良いところでもあり，その特徴を活かして機能の充実を図るのが現実的な方法であろう．汎用部品として考えられるものとしては，

- リザルト保持機能(SV)
- レポート，キー割れページ替え，ヘッド追加印書機能(PRINT)
- 罫線印書機能(RPRNT)
- 項目縦横変換機能(RFORM)
- 2次元集計表編集機能(XYHYO)
- データ重複チェック機能
- 年数計算機能
- 繰越計算機能
- 標準偏差計算機能

等が考えられる(図13)．

汎用部品は会話機能と会話機能の間でエンドユーザから呼ばれるものなので，入力は画面に表示されているレポートまたはリザルトとし，操作内容の指示方法は会話操作と違和感のないマスク画面形式が望ましい．

4) 教育・支援……いくら環境やツールが準備されていても，エンドユーザ自身が使う気がしなければMAPPER活用は実現しない．とくに担当業務の中で非定型の仕事の占める割合の多いエンドユーザがMAPPERにとってのエンドユーザということになるが，この人達にMAPPERの良さを理解してもらい，使ってもらうための技術が当社のSEや客先の情報システム部門，また利用部門のキーマン等に必要となってくる．また，エンドユーザに対する継続的な教育・啓蒙や，コアの部分の提供，部品の提供・作成指導他，全般にわたる技術指導・支援等，情報センタとしての情報システム部門の役割は非常に重要なものである．

3.4 マニュアルの方向

MAPPERを有効に活用するシステムを作るためには，既存のマニュアルにあるソフトウェアの機能の記述だけでは不十分であり，以下に示すような内容を持つ適用ガイドが必要である．

- システム構築に際してのMAPPER適用の範囲やポイント，効果的システムの構築を支援する環境や体制のあり方
- MAPPER適用範囲に対する効果的・効率的なシステム設計のノウハウや留意点
- MAPPERによる効率的システム開発のための基礎知識や手順，ノウハウ，ツール

・エンドユーザが積極的に情報を処理するための基礎知識，手順，ノウハウ

・MAPPER システム運用に関するノウハウ，留意点，ツール

これまでもガイドの類がいくつも作られているが，客先に埋もれていたり，あるいは保守されず現在では陳腐化してしまったものも少なくない．今後は当社の正式なマニュアルとして適用ガイドを位置づけ，有効な情報を適宜社内外に提供することが重要であり，一部執筆を開始している．また客先の情報システム部門においても，エンドユーザに対する教育・支援の立場から，客先の実情にあったマニュアル/ドキュメントを作成し，エンドユーザに提供することも重要である．

4. おわりに

本稿では，MAPPER のより一層の利用拡大のために，「MAPPER 活用システム」の考え方，構築のポイント等を考察した．この考え方自体は決して目新しいものではないが，従来不明確な部分が多く，実務に適用するには脆弱な面もあった．今回の考察は，今までの MAPPER 適用経験を基に，MAPPER 活用システムの明確なイメージ作りを目指したものであり，現実的な適用技術については個々の客先の現状やそれぞれの情報システムの目的に合わせて対応できるように，今後より一層の検討を行うとともに，以下の 2 点の早急な実施が必要であると考える．

1) MAPPER 適用の基本的考え方の確立と普及……2.1 節で述べたような MAPPER 活用システムに対する基本的考え方を検討・確立したのち，教育やセミナを通じて普及させていく必要がある．

2) 適用技術およびツールの整備・普及……普及のベースとなるのは，2.4 節で述べたマニュアルである．この各種適用ガイドとともにツールを整備し，それと並行して普及のための教育の実施が望まれる．また実際の適用に当たっては，組織的な支援が不可欠であろう．

適用技術の中には，従来あまり重きを置かれなかったり，顧みられなかった分野もあり，この部分については今後さらにノウハウの蓄積のための作業が必要である．

執筆者紹介　松　木　規　子 (Noriko Matsuki)

1956 年生．80 年早稲田大学第一文学部卒業．同年日本ユニシス(株)入社．81 年 10 月より MAPPER システム適用サポートに従事．現在システム技術本部 4 GL 技術部 MAPPER 技術課に所属．

MAPPERとエキスパートシステムの連動

The Linkage of a MAPPER Application and an Expert System

保科　剛, 川上峰子

要　約　エキスパートシステム開発が実用システム開発の段階を迎えた今日，従来の情報処理システムの中にエキスパートシステムをどう位置付けるか，いかにエキスパートシステムと従来のシステムを融合するかが課題となっている．

本稿では化粧品の生産スケジューリングを事例に，MAPPERアプリケーションとエキスパートシステムの連動およびスケジューリング・エキスパートシステムについて述べる．

Abstract　Nowadays, when the development of an expert system is beginning to mean that of a practical application system, a major subject of current discussion has focused on how an expert system can more appropriately be positioned among conventional data processing systems and on how a knowledge-based system is better integrated with traditional systems.

Taking cosmetics production scheduling as an example, this paper describes how a MAPPER application is linked to an expert system and additionally discusses a scheduling expert system.

1. はじめに

これまでエキスパートシステムの開発は，試行錯誤をともなう研究開発として行われることが多かった．開発の目的は主としてエキスパートシステムで何ができるか，どうすればよいのかに向けられ，従来のシステムとの連携を考慮したハイブリッドなシステム開発にはあまり目が向けられていなかった．開発の主体が従来のシステムでは情報処理部門であるのに対し，エキスパートシステムでは研究開発部門であったことも，その原因の一つと考えられる．

エキスパートシステムの開発が実用システム開発の段階を迎えた現在，その開発の主体は研究開発部門から情報処理部門に移りつつある．既存のソフトウェア資産を抱える情報処理部門では，エキスパートシステムをどのように位置付けるか，いかにエキスパートシステムを従来のシステムと融合するかが課題となっている．

従来のシステムとエキスパートシステムの融合には，さまざまな形態が考えられる．しかし，一般的にシステム全体でエキスパートシステムを適用できる割合が1割程度と考えられていることを考慮すると，その連動形態としてはエキスパートシステムをサブシステムとして従来のシステムの付加機能と位置付けることが実際的である．

本稿では従来のシステムがMAPPERを用いて開発されている場合を前提に，エキスパートシステムとの融合を具体的な事例，化粧品の生産計画を用いて考察する．MAPPERを候補としたのは，すでにMAPPERによるビジネス分野でのアプリケーションが数多くあり，MAPPERがデータベースやユーザインタフェースとして定評があるからである．

MAPPERとエキスパートシステムの連動を示す事例として，化粧品会社をモデルとした．その化粧品会社では，すでにMAPPERによる販売管理システムが利用され

ていることを仮定する．生産部門で生産スケジュールを作成する時は，この販売管理システムを通して注文を集計し，熟練のスケジュールの専門家がさまざまな制約条件を満たし，かつ効率のよいスケジュールを作成する．既存の販売管理システムと新しく開発する生産スケジューリング・エキスパートシステムの連動が，本稿で考察する事例である．この事例の概要について，第2章で述べる．第3章では，エキスパートシステムをどのように融合したかについて述べる．第4章では，販売管理システムから得る生産依頼表，生産部門での製造機械の制約を満たす最適なスケジュールを作成するエキスパートシステムのしくみについて述べ，第5章では，開発したシステムの評価と今後の課題について述べる．

2. 問　　題

今回，事例として取り上げたのは，化粧品の生産スケジューリング・システムである．既存のMAPPERアプリケーションの情報を利用して，スケジュールの専門家が作成するようなスケジュールを，エキスパートシステムで作成しようとするものである．まず，既存のMAPPERアプリケーションである販売管理システムについて概要を説明する．

モデルとした化粧品会社では，販売計画部門と生産部門がある(図1)．生産部門では，販売計画部門より提案された製品の生産依頼表を集計し，生産スケジュールを作成している．この生産スケジュールをもとに，工場では生産ラインを動かしている．

販売計画部門では，すでにMAPPERを利用した販売管理システムを活用している．このシステムではこの会社が販売する全製品の情報をマスタ・ファイルに登録しており，販売計画を立てる際に利用している．この販売管理システムに対し，個々の製品の販売計画(製品の色・数量・納期・優先度等)を入力することにより，システムは表1のような当月の製品生産依頼表を自動的に作成する．販売計画部門は，作成した生産依頼表を生産部門へ提出する．

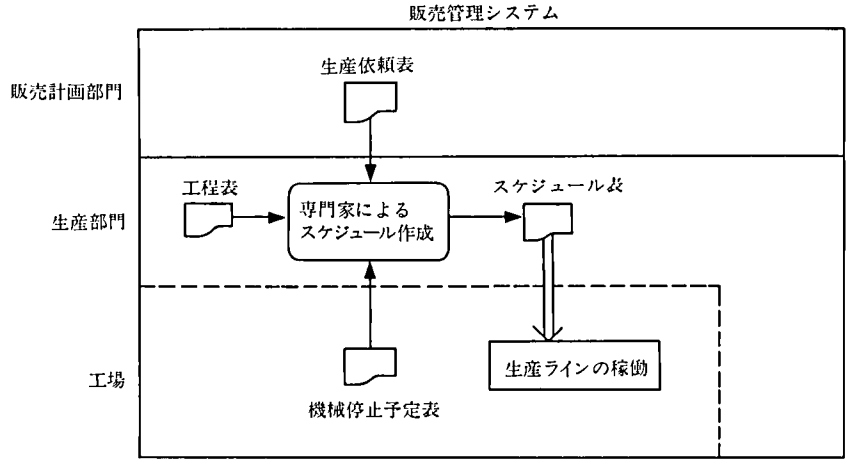


図1　化粧品会社の業務形態

Fig. 1　Process flow of scheduling

また，このシステムには各製品の工程情報を記録した「工程表」が登録してあり，生産部門が管理している．化粧品の製造工程は，「原料加工１」，「原料加工２」，「バルク製造」，「充填仕上」からなる．これらの工程，およびその処理順序は，基本的にはどの製品にも共通である．しかし，各工程の処理に要する日数は，製品ごとに異なっている．これを記録したものが表２に示す「工程表」である．

工場が管理している情報に，「機械停止予定表」がある(表３)．各製造機械には，メンテナンス等のため稼働停止期間が存在する．これを記録したものが「機械停止予定表」で，既存システムで保持している．この表には，機械の機能(どの工程を処理する機械か)，機械の容量等も記録してある．

さて，生産部門では当月の生産スケジュールを作成するわけであるが，この処理はMAPPERではシステム化が困難な部分である．なぜなら，販売計画部門から提出され

表 1　生産依頼表

Table 1　Order form of products

```
LINE♦·1     FMT♦·  RL♦·-     SHFT♦·    HLD CHRS♦·   HLD LN♦·   ♦ 5B24          ♦
.DATE   10 JUL 90  14:33:36  RID    5B    19 JUN 90  AIDEMO
.   【 生産依頼表 】
*製  品:                      : 色  :      :      :      :優: 開始 :
*コード:    製   品   名      :コード:納  期:数  量:計画日:先:可能日:
*=======.====================.======.======.======.======.==.======.
·LS-999·キャンディペン        ·RE0200·900830· 20000·900527· 7·900701
·LS-200·ウォータルージュ      ·PI0001·900830·  2500·900601· 3·900701
·LS-101·バイオ口紅            ·OR0020·900810·  3500·900601· 6·900701
·LS-100·ベールリップ          ·WI1200·900815· 15000·900602· 5·900701
·LS-001·ＢＡリップ            ·PI0201·900815·  4500·900602· 6·900701
·LP-300·アブナールリップ      ·OR1200·900820·  5000·900605· 4·900701
·LP-350·ミスティーリップ      ·PI0201·900901·  1200·900605· 5·900701
                        ..... END REPORT .....
```

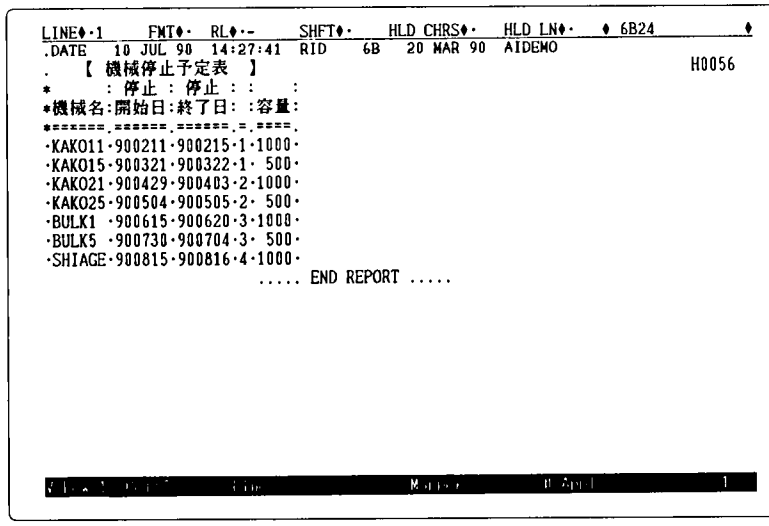
表 2　工程表

Table 2　Process table

```
LINE♦·1     FMT♦·  RL♦·      SHFT♦·    HLD CHRS♦·   HLD LN♦·   ♦ 4B24          ♦
.DATE              10:38:07  RID    4B    20 MAR 90  AIDEMO
.   【 工  程  表 】                                              H0056
*製  品:加 :加 :ﾊﾝﾄ:仕 :                  :
*コード:工1:工2:ク  :上 :   製  品  名      :
*=======.===.===.===.===.====================.
·LP-001· 02· 03· 03· 01·シャイムリップ
·LP-002· 03· 03· 03· 01·リップカラー
·LP-003· 08· 08· 04· 01·チェンジリップ
·LP-100· 03· 03· 03· 01·レディ９０
·LP-101· 02· 03· 03· 01·アイ・リップ
·LP-150· 10· 10· 04· 01·ダーク・キス
·LP-200· 03· 03· 04· 01·フィットリップ
·LP-220· 03· 03· 03· 01·フィットネスミニ
·LP-300· 07· 08· 04· 01·アブナールリップ
·LP-350· 03· 03· 04· 01·ミスティーリップ
·LP-500· 08· 06· 04· 01·エルクシルリップ
·LS-001· 03· 03· 04· 01·ＢＡリップ
·LS-100· 03· 03· 04· 01·ベールリップ
·LS-101· 03· 03· 03· 01·バイオ口紅
·LS-111· 03· 03· 04· 01·カーネリップ
·LS-123· 04· 04· 04· 01·ミニミニップ
·LS-200· 04· 03· 04· 01·ウォータルージュ
·LS-210· 04· 03· 04· 01·ピンク・カラー
```

表 3　機械停止予定表

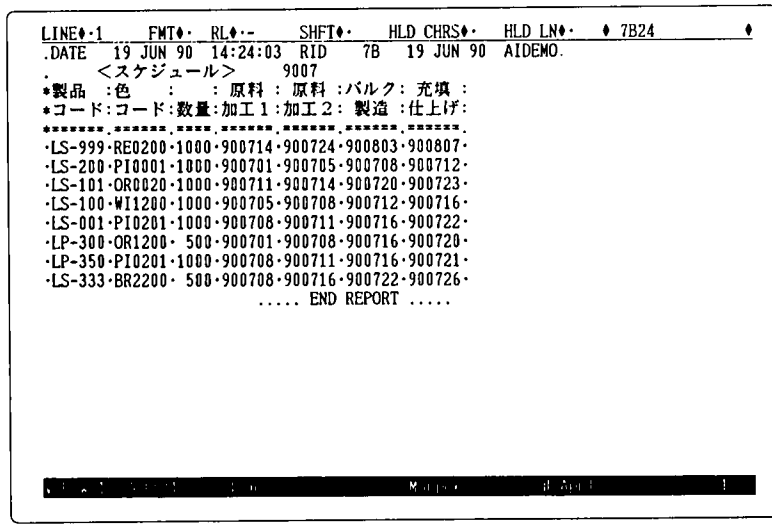
Table 3　Schedule of machine suspension



```
LINE♦·1     FMT♦·  RL♦·-     SHFT♦·    HLD CHRS♦·   HLD LN♦·   ♦ 6B24          ♦
.DATE   10 JUL 90  14:27:41  RID    6B   20 MAR 90  AIDEMO
.    【 機械停止予定表 】                                                  H0056
*       : 停止 : 停止 : :     :
*機械名:開始日:終了日: :容量:
*=======.======.======.=.====.
·KAKO11·900211·900215·1·1000·
·KAKO15·900321·900322·1· 500·
·KAKO21·900429·900403·2·1000·
·KAKO25·900504·900505·2· 500·
·BULK1 ·900615·900620·3·1000·
·BULK5 ·900730·900704·3· 500·
·SHIAGE·900815·900816·4·1000·
                        ..... END REPORT .....
```

表 4　スケジュール表

Table 4　Schedule



```
LINE♦·1     FMT♦·  RL♦·-     SHFT♦·    HLD CHRS♦·   HLD LN♦·   ♦ 7B24          ♦
.DATE   19 JUN 90  14:24:03  RID    7B   19 JUN 90  AIDEMO.
.      <スケジュール>      9007
*製品  :色     :     : 原料 : 原料 :バルク: 充填 :
*コード:コード:数量:加工１:加工２: 製造 :仕上げ:
*=======.======.====.======.======.======.======.
·LS-999·RE0200·1000·900714·900724·900803·900807·
·LS-200·PI0001·1000·900701·900705·900708·900712·
·LS-101·OR0020·1000·900711·900714·900720·900723·
·LS-100·WI1200·1000·900705·900708·900712·900716·
·LS-001·PI0201·1000·900708·900711·900716·900722·
·LP-300·OR1200· 500·900701·900708·900716·900720·
·LP-350·PI0201·1000·900708·900711·900716·900721·
·LS-333·BR2200· 500·900708·900716·900722·900726·
                        ..... END REPORT .....
```

る「生産依頼表」と工場から上がってくる「機械停止予定表」の制約条件を満たしつつ，最適のスケジュールを得たいからである．製品は納期に遅れては困るし，停止中の機械に工程を処理させようとしても無理である．しかも化粧品の製造には独特の規則がある．たとえば，色／香料の種類によって製品を製造する順番が変ってくる．まとめて処理できそうな工程がいくつかあれば同時に製造する，等である．そこで，このような化粧品製造に関する知識を持った専門家が，過去の経験を生かし，「工程表」，「機械停止予定表」，「生産依頼表」を照合して，最適な生産スケジュールを立案する．このスケジューリングには，相当の熟練を要し，時間もかかる．このスケジューリング部分にエキスパートシステムを用いれば，かなり有効であると考えられる．しかも，既存の販売管理システムと連動することができればこれまでの情報資源を活用できるし，また人手を介在することなく販売管理から生産計画まで一貫して処理することが

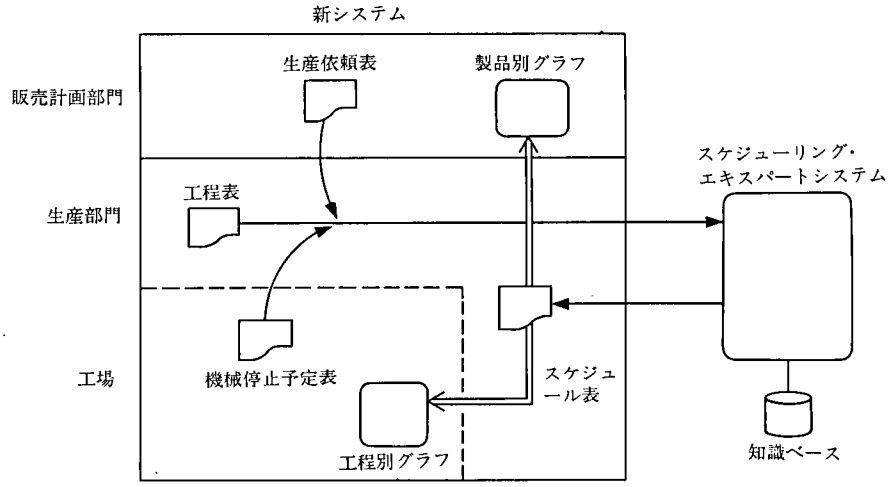


図 2　新システムの概要

Fig. 2　New scheduling system

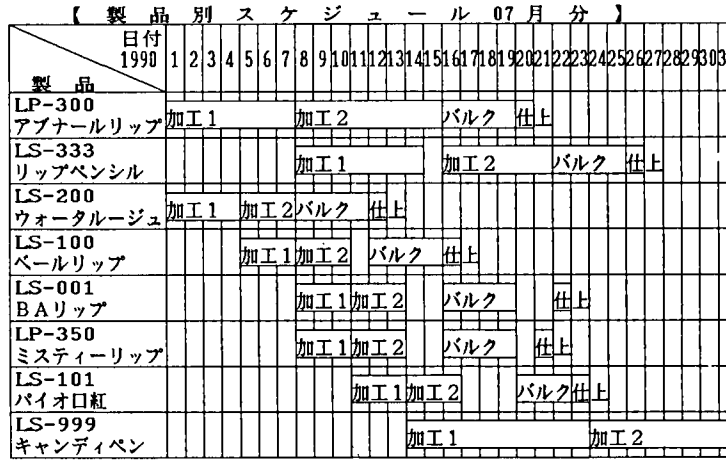


図 3　製品別スケジュール・グラフ

Fig. 3　Gant chart of production

可能である．

新システムの概要を図2のように考えた．生産計画部門にサブシステムとしてスケジューリング・エキスパートシステムを据える．とはいうものの，エキスパートシステムは実際にはMAPPERによる既存システムの内部に構成するのではなく，既存システムとエキスパートシステムの連携方法により，あたかもエキスパートシステムをサブシステムのように利用する．生産計画部門は販売計画部門から提出された「生産依頼表」と「工程表」を照合し，さらに工場から上がってきた「機械停止予定表」をエキスパートシステムに送りスケジューリングさせる．エキスパートシステムは，スケジューリングに必要な専門知識(製品の納期や優先度，色／香料・生産量・機械の停止期間の考慮，さらにどの工程とどの工程がまとまるかの判断)を知識ベースに蓄積しており，化粧品生産スケジューリングの専門家が作成するのと同等の効率の良い稼働スケジュールを作成する．作成されたスケジュール表(表4)はグラフに表示すると非

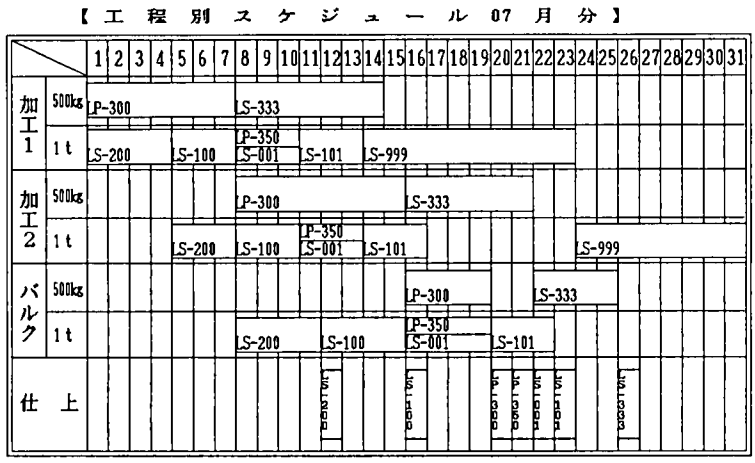


図 4　工程別(機械別)スケジュール・グラフ

Fig. 4　Gant chart of machine usage

常に見やすくなる．販売計画部門では，どの製品がいつから製造され，いつ仕上がるのかを常に把握したい．そこで，図3のような「製品別スケジュール・グラフ」を表示させる．工場では，どの機械が現在稼働中か否か，稼働中であればどの製品を製造しているのかを常に知る必要がある．図4のような「工程別(機械別)スケジュール・グラフ」が有用である．

3. MAPPERとエキスパートシステムの連携

MAPPERとエキスパートシステムの連携として考えられる方法はいくつかあるが，今回はMAPPERのアプリケーションからエキスパートシステムを呼び出す形態とした．詳細をMAPPERのラン(業務プログラム)とエキスパートシステムに分けて説明する．なお説明上，UNIX* システムでの実現を前提とした．

まずMAPPERのランでは，エキスパートシステムで必要なデータ(「生産依頼表」と「工程表」をマージしたもの，および「機械停止予定表」)を，UNIXファイルに変換しておく．このとき，@FIL(ファィル)命令を使用する．以下に示すMAPPERラン中の②では，「生産依頼表」と「工程表」のマージ結果を，/mapper/kes/job-data. txtというファイルに変換し，③では「機械停止予定表」を同じディレクトリ下のmachine. txtというファイルに変換している．

```
①　@LDV V1A4=24, V2A1=B, …
　　⋮
②　@FIL, V 1, V 2, -2, N, N　/mapper/kes/job-data. txt.
③　@FIL,  V 1, V 2, -3, N, N　/mapper/kes/machine. txt.
④　@UNX,, mapkes　-w-c　/mapper/kes/job. sh.
⑤　@RET, V 1, V 2, N, N　/mapper/kes/map. in RNM-1.
　　⋮
```

次に，@UNX(UNIX)命令を使用して，エキスパートシステムを起動する．@UNX

* UNIX：AT＆Tベル研究所が開発し，AT＆Tがライセンスしている．

命令はランからUNIXコマンドを実行する命令で，UNIXシステム上の機能やプログラムをMAPPERランの中に容易に組み込めるものである．④のファイル/mapper/kes/job.shは，エキスパートシステムを実行するためのシェル・スクリプトである．ログイン名には，このシェル・ファイルの実行権を持つもの(今回のシステムではmapkes)を指定する．エキスパートシステムが終了すると，自動的に制御がMAPPERのランに戻るようになっている．しかし，オプション-Wを付加することにより，エキスパートシステムが出力する画面を保持することも可能である．エキスパートシステムの推論結果を確認する時は便利である．この場合，ファンクション・キーを押下すると制御がMAPPERのランに戻る．

最後に，エキスパートシステムの実行結果を受け取るわけであるが，これには@RET(リトリーブ)命令を使用する．今回のシステムでは，エキスパートシステムが/mapper/kes/map.inというファイルに実行結果を出力するので，⑤でこのファイルをMAPPERレポートにリトリーブしている．

以上が，MAPPERのランからエキスパートシステムに連携するために必要な操作である．

エキスパートシステムは，MAPPERのランが@FILした二つのファイル(/mapper/kes/job-data.txtと/mapper/kes/machine.txt)を読み込んで，MAPPERのランが@RETするファイル(/mapper/kes/map. in)へ実行結果を出力するように作成する．シェル・スクリプトは，次に示すような形でMAPPERのランが参照するファイル(/mapper/kes/job.sh)に記述しておく．

```
① #!/bin/sh
② ( cd/mapper/kes ; ./demo )
```

以上が，エキスパートシステムでMAPPERとの連携のために必要な操作である．

MAPPERとエキスパートシステムの連携は，MAPPERのランからエキスパートシステムを呼び出す形態とした．これは，MAPPERとエキスパートシステムを連動させたシステムを開発する際に，多くの場合すでにMAPPERのアプリケーションが存在している場合が多いと考えたからである．この連携形態ではMAPPERのアプリケーションに数行の修正を加えるのみで，エキスパートシステムを呼び出すことが可能になる．すなわち，エキスパートシステムを既存システムのサブシステムとして位置付けることができるわけである．

エキスパートシステムでこのような販売管理システムを実現しようとしても，使い勝手の悪いシステムになる恐れがある．エキスパートシステムでは，データの照合や抽出やグラフ表示等は不向きな処理だからである．しかし，MAPPERのランや会話機能をユーザインタフェースとして使用することにより，これらの処理が容易に行え，エキスパートシステムは本来の推論処理に専念することができる．このように，明確にモジュール化されていれば，エキスパートシステムの部分を他のMAPPERアプリケーションへ接続するということも容易に行える．もちろん，エキスパートシステム部分を単独で作動させることも，この連携形態だと可能である．

システムの利用者から見れば，ユーザインタフェースにMAPPERを使用している

ので，従来のMAPPERアプリケーションと同等の使用感でエキスパートシステムが操作できるのは非常に便利である．また同様の方法により，エキスパートシステムが持っている知識ベースを，MAPPERのレポート上で作成／修正し，エキスパートシステムに落とし込むことも可能であり，エキスパートシステム自体の活用をさらに容易にすることができる．

4. スケジューリング・エキスパートシステム

何種類かの仕事を何台かの機械で処理する工場を考える．各仕事がいくつかの工程からなっており，それらの工程の実行順序あるいは仕事の間に先行関係が与えられていた時に，各仕事をどのような順番で処理すれば総所要時間が最小となるであろうか．また，仕事の間の先行関係および各工程を処理するのに要する資源が与えられているとき，それらの資源制約を満たしつつ，総費用または総所要時間を最小にするにはどうすればよいだろうか．

このような問題をスケジュール問題と呼んでいる．化粧品の生産スケジューリングをスケジュール問題の観点から調べると，以下のような特徴がある．

1) すべての仕事が同一の四つの工程から構成され，またその実行順序が同一である．
   - 原料加工1
   - 原料加工2
   - バルク製造
   - 充填仕上
2) 各工程は複数台ある同種の機械のいずれか一つで実行される．
3) 各機械は割り当てる同種の工程がある一定の条件を満たすならば，複数個の工程を同時に実行することができる．
4) 各機械には稼働停止期間がある．

スケジュール問題の解法は，問題を組み合わせ最適化問題として定式化し，整数計画法や分枝限定法あるいは，これらの近似解法といった数理計画手法に基づく解法が中心である．この一つに，総所要時間最小化スケジュール問題の解法として，部分アクティブ・スケジュールを列挙木の頂点とした分枝限定法が知られている[1]．

ここでアクティブ・スケジュールとは，実行可能なスケジュールで，どの工程も順序関係を破らずに前へつめることができないようなスケジュールのことである．だが，この解法では，工程の数・機械の数が大きくなると，組み合わせ数が大きくなりすぎ，現実的な時間内で解を得ることが困難である．そこで列挙木の頂点での枝の選択に，実際のスケジューリングを行っている専門家の知識を利用することで枝刈りを行うことにした．

エキスパートシステムは，「問題領域の専門家の知識を利用して，十分に複雑な問題を，専門家と同等のレベルで解決するアプリケーション・プログラム」である．通常，専門家が問題解決に用いる知識を一定の形式で表現し蓄積した知識ベースと，この知識ベース内の知識を利用して外部から与えられたデータあるいは事実を解釈し，結論を導き，その結論の根拠等を説明する推論機能から構成される．

エキスパートシステムには，次のような特徴がある．

1) 知識ベースと推論機構が分離されているので，どの知識が，どんな状況で，どのように利用されているかを把握することができる．

2) モジュール性に富む表現に基づく知識ベースに知識を追加することにより，システムの性能を段階的に向上させることができる．すなわち，最初は小規模な知識ベースからスタートし，評価・修正・追加を繰り返して，大規模かつ能力の高いエキスパートシステムへと発展させることができる．

3) 知識の表現や推論のメカニズムは，問題解決担当者である専門家にも十分わかりやすいので，専門家自身によるエキスパートシステムの構築が可能である．これにより，プログラムの生産性・信頼性を高めることができる．

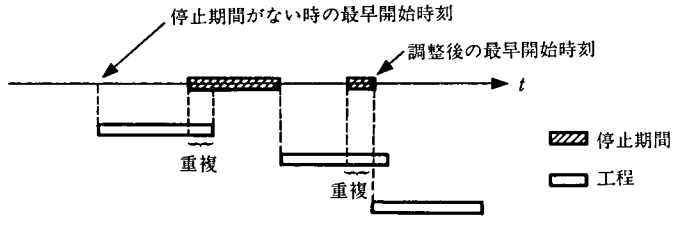
4) 推論に用いられた知識の連鎖を記憶しておくことにより，推論過程に関する説明機能を持っている．この機能により，エキスパートシステムの利用者である非専門家は，結論に至った専門家の経験的知識を知ることができる．また，専門家は知識ベースの評価に，この機能を利用することができる．

次にスケジューリング・エキスパートシステムの処理の概略を示す．

$\sum_t$：すでにスケジュールされた t 個の工程を含む部分スケジュール．

S_t^m：$\sum_t$ が与えられた時に機械 m で次にスケジュール可能な工程の集合．次にスケジュール可能な工程とは，その先行工程がスケジュールされた状態にある工程のことである．たとえば，ある仕事の原料加工 2 までがスケジュールされた状態にあった場合は，バルク製造が次にスケジュール可能な工程となる．

e_j^m：工程 $j \in S_t^m$ の最早開始時刻(工程 j の処理が対応する機械において開始可能となる最早時刻)．ただし機械の停止期間を考慮し，e_j^m を開始時刻として工程 j を割り当てたとき，停止期間と重ならないように調整する．



最早開始時刻の調整

f_j^m：工程 $j \in S_t^m$ の最早終了時刻(工程 j の処理が対応する機械において最も早く終了しうる時刻)．ただし，e_j^m と同様に停止時間を考慮して調整する．

---

step 1：$t:=0$, $\sum_0=\phi$, $S_0^m:=${機械 m で実行可能なすべての原料加工 1}

step 2：$\cup_m S_t^m=\phi$ ならば終了.

1. $f^m:=\min j\in s_t^m f_j^m$
2. $f^*=\min_m f^m$
3. $m^*:=\text{minargm} f^m$ を処理する機械}

step 3：$e_j^{m*}<f^*$ なる工程 $j\in S_t^{m*}$ に対してルールを適用し，実行する工程の集合 J^* を求める.

1. 工程 $j\in J^*$ の開始時刻を設定する.
2. $\sum_{t+1}:=\sum_t\cup J^*$

step 4：新しい部分スケジュール $\sum_{t+1}$ に対して

1. $S_{t+1}^m:=S_t^m\backslash J^*$
2. $S_{t+1}^m:=S_{t+1}^m\cup${機械 m で実行可能な $j\in J^*$ の後続工程}
3. $t:=t+1$

step 3で $e_j^{m*}<f^*$ なる工程 $j\in S_t^{m*}$ に対しルールを適用し工程を選択しているので，このアルゴリズムから生成されるスケジュール $\sum_t$ はアクティブスケジュールとなる．ルールによって枝刈りを行っているので最適解が得られる保証はないが，すくなくともアクティブスケジュールが得られる.

知識ベースには，step 2で求めた機械 m^* に割り当て可能な工程の候補，すなわち $e_j^{m*}<f^*$ なる工程 $j\in S_t^{m*}$ から，工程を選ぶためのルールが蓄積されている．納期・優先度・生産量・停止期間等を総合的に判断し，同時に実行できる工程があればそれらをまとめるルールである．しかしそれ以外の部分，たとえば部分スケジュールの更新，次に割り当てを行う機械の決定，その機械に割り当てる工程の候補の抽出等はルールとして記述する必要はない．この部分は推論機構中に組み込んだ.

5. おわりに

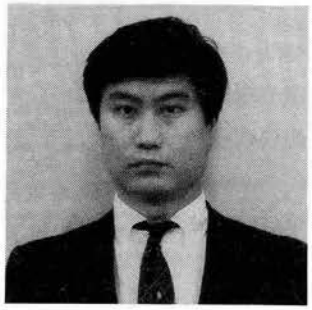
化粧品会社の生産スケジューリングをモデルとして，MAPPERとエキスパートシステムによるハイブリッドシステムを開発した.

システムの利用者から見ると，インタフェースが一般のMAPPERアプリケーションと同様であり，エキスパートシステムを意識することなく使うことができる．たとえば，スケジューリングのための諸条件(注文の優先度・機械の停止日等)の変更も，通常のMAPPERレポート操作で行うことができる．また知識ベースの開発者から見ると，スケジューリングの基準を与える知識ベースの変更を，通常のMAPPERの会話機能で行える．つまりMAPPERとエキスパートシステムのハイブリッドシステムを，MAPPERを利用することでユーザインタフェースが統一された密結合型のシステムとして実現することができた.

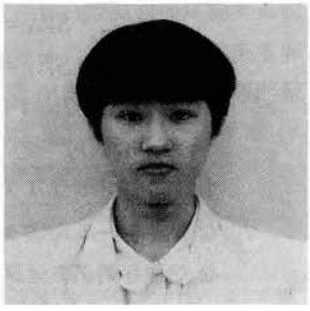
今後の現場での適用を通じてMAPPERとエキスパートシステムの連動技術を確立していきたい.

参考文献 ［1］ 今野浩・鈴木久敏，「整数計画法と組合せ最適化」，日科技連出版社，1982.
［2］ Adding the power of knowledge to MAPPER applications with KES™ II software, Unisys Corp., UP-15134, 1989.

執筆者紹介 保 科 剛 (Tsuyoshi Hoshina)
昭和32年生．56年東京理科大学応用数学科卒業．同年日本ユニシス(株)入社．人工知能，数理計画分野のシステム開発に従事．現在知識システム部所属．日本オペレーションズ・リサーチ学会会員．



川 上 峰 子 (Mineko Kawakami)
昭和39年生．62年筑波大学第三学群情報学類卒業．同年日本ユニシス(株)入社．統合オンライン・ソフトウェアの開発を経て，汎用機上の Common Lisp の開発に従事．現在知識システム部 開発二課に所属．



LINCにおける分散開発の現状と今後の課題

Current Distributed Systems Development by LINC II and Suggestions for the Future

山 科 順 一

要 約 1983年にLINCが日本に初めて登場したころは，COBOLコンパイラのようにLINC定義言語（LDL）で記述されたソースコード・ファイルを入力として，業務システムを生成するバッチ型であった．この頃のシステム開発は開発チームも数人で，開発するシステムの規模も比較的小さなものであった．

1985年になると，対話形式で開発を行うLINCが登場し，業務システムの開発が容易になってきた．それに伴い開発規模も大きくなり，現在ではLINC定義言語で10万ステップを超える規模のシステムが多く見られるようになってきた．

このようなLINCの発展と開発規模の拡大に伴って，LINCに対する要望や意見も変化してきた．登場した初期のころは，プログラミングに関するものが多かったが，今日では開発環境，開発体制，開発方法に関する事柄が多く見られるようになってきた．とくに開発規模が大規模化している現在は，開発環境のあり方についての要求が多くなっている．

本稿では，LINCで大規模システムを開発する場合に必要となる，分散開発とは何かを明らかにし，分散開発する場合の環境のあり方について提案する．

Abstract When released for the first time in Japan in 1983, LINC came as a software product for batch-processing use, designed to help generate application systems, like the COBOL compiler, from source-code files written in the LINC definition language (LDL). In those days, systems development teams consisted of a very limited number of members and the size of the systems to be developed was also relatively small.

In 1985, the interactive LINC compiler (LINC II) was announced and made it possible to develop application systems in a conversational mode. In parallel to this progress, systems development has gradually become larger and larger in size, with the result that there are a lot of application systems large enough to exceed 100,000 LDL steps at present.

With the enhancement of LINC and the expansion in size of applications development, there have also been changes in customer requirements and desires for LINC. While those were forcused on programming-related problems in the early days, they are now related in great numbers to the environments, resources and methods required for systems development.

Especially in the current trends toward the enlarged scale of systems development, more and more requirements tend to go around how development environments should really be.

This paper, therefore, clarifies the distributed systems development which helps create large-scale systems by use of LINC and proposes how environments involved with distributed systems development should be established.

1. はじめに

日本でLINCが初めて登場したのは1983年の始めのことである．この頃のLINC

はLINC Iと呼ばれるバッチ型のタイプであり，COBOL言語で記述されたプログラムと同様に，LINC定義言語(以下，LDLと呼ぶ)のソースコード・ファイルをLINCコンパイラにかけて翻訳する形式であった．当時のシステム開発の特徴としては，開発メンバが1人から2人程度で，開発プロジェクト自体も自社開発が主体であり，規模もLDLでせいぜい5千ステップから1万ステップであった．

1985年になるとLINC IIと呼ばれる対話型が登場した．この対話型LINCの特徴は，これまでのようにLDLのソースコード・ファイルを作るのではなく，LINCと対話形式で開発を行う点にある．これによりLDL入力時にシンタックスが検査され，入力が終了すればエラーのない状態のLDL記述ができあがる．

このころになると次第に開発規模が大きくなり，LDLで2万ステップから3万ステップくらいのものが多くなり，開発プロジェクトも外注化の割合が高くなり，開発メンバも増えてきた．この開発規模の拡大に伴って，LINCで記述された業務処理システムの生成（以下，単にシステム生成と呼ぶ）時間も増大してきた．

そこで，開発環境を改善した対話型が1986年に登場した．このころから，LINCで大規模開発と呼ぶ，LDLで10万ステップを超えるものが出てくるようになり，現在では20万ステップを超えるシステムも稼働し始めている．さらに，LDLで50万ステップを超えるシステムの構築も行われるようになり，開発プロジェクトの外注化率と開発メンバの増加が飛躍的に進んでいる．

このような大規模なシステム開発が盛んになってきた現在，LINC適用上の論点も単なるプログラミングから，開発環境や開発体制，プロジェクトの管理，さらにLINCにおけるシステム開発の方法や技法に移ってきた．

とくに，大規模開発を行う場合の開発環境のあり方に関する要求が多く寄せられるようになってきた．そこで，LINCでシステム開発する場合の必須条件となってきた分散開発環境のあり方について，検討する必要が出てきた．

本稿では，

1) LINCでシステム開発する場合，「分散開発」とはどういう形態を指すのか，
2) 「分散開発」がなぜ必要になってきたのか，
3) 「分散開発」する場合，注意すべき点は何か，
4) どうすれば「分散開発」が円滑に進められるか，

を取り上げる．

2. LINCの分散開発

本章では，本稿における分散開発とはどういう開発形態なのかを定義し，LINCにおける開発環境はどうなっているのかを説明する．

2.1 分 散 開 発

一口に分散と言っても，開発するシステムは分割せずに開発チームだけをいくつかに分ける組織の分散．開発するシステムをいくつかに分割して，開発チーム自体をそれぞれの分割したシステムに対応づける開発環境の分散．ワークステーションに開発機能の一部を移動した機能の分散．さらに，これらを組み合わせた形態が考えられる．そこで，本稿で対象とする分散開発とは何かを定義したい．

一言で表せば「システムをある論理的繋がりのある機能単位に分割し，開発環境をそれぞれ分割した単位に割り当てて開発を行い，最後にそれぞれを一つの環境下で一つのシステムに統合するような開発形態」であり，開発環境の分散の一例である．

なお本稿では，システムを分割した後の各開発単位をモジュールと呼ぶので，一般に使われているモジュールとは区別している．したがって，とくに説明のない場合は，分割された各開発単位を指すことにする．

2.2 LINC の開発環境

2.2.1 システム開発時の LINC の基本的環境

図1に示すように，開発時の環境は縦割りになっている．つまり，LINC コンパイラで共通なデータ辞書であるグローバル辞書と，業務システムの二つに大きく分けることができる．業務システムはさらに，業務システム固有のデータ辞書であるローカル辞書とシステム記述（LDL で記述された業務システムのこと．以下，単にシステム記述と呼ぶ）からなり，業務システムは互いに独立で他に影響を与えない環境を実現している．

2.2.2 分散開発時の LINC の環境

開発環境を分散する場合，大きく分けて二つの方法がある．

一つは同一コンパイラ上で行う場合であり，もう一つは同一コンピュータ上あるいは，複数コンピュータ上に複数のコンパイラを持つ場合である．

1) 同一コンパイラ上の場合……同一コンパイラ上で分散開発環境を作る場合は，図1とほぼ同じ関係になる．違うのは，統合システムを作るためにシステム・オーディット・ユーティリティ（以下，SAU と呼ぶ）を使って，ローカル辞書とシステム記述を併合するという点である．

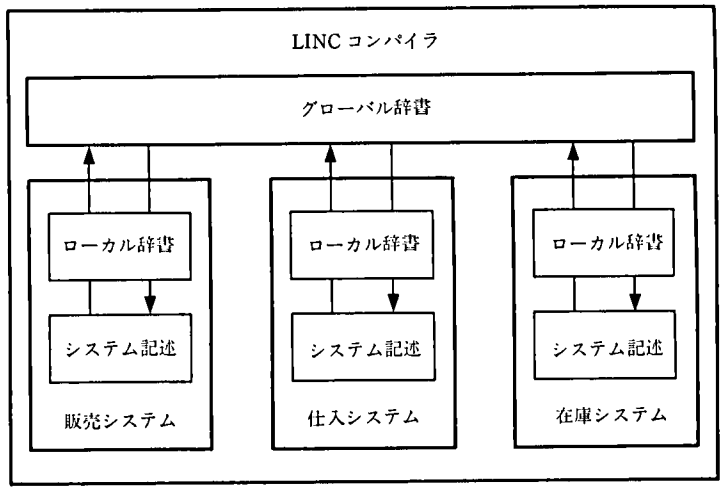


図1　基本的な開発環境の例

Fig. 1　Basic development environment

図2の例は，販売在庫システムを販売モジュール，在庫モジュールという機能単位に分割し，それぞれを別々に開発しておいて，最後に販売在庫システムとして統合することを表している．

この開発環境は，比較的ハードウェア能力が大きい場合や，システムの分割数が少ない場合に向いている．

2) 複数コンパイラあるいは，別コンピュータ上の場合……図3は複数のコンパイ

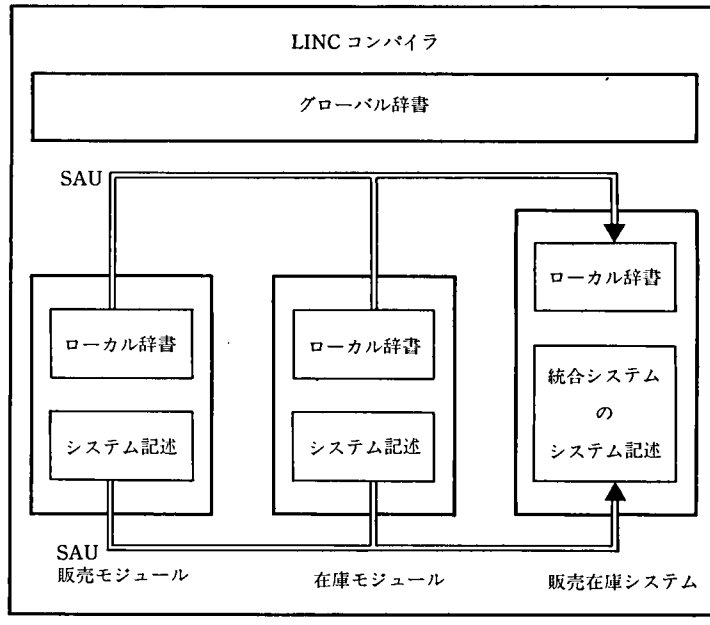


図2 分散開発時の開発環境の例

Fig. 2 Development environment in distributed development

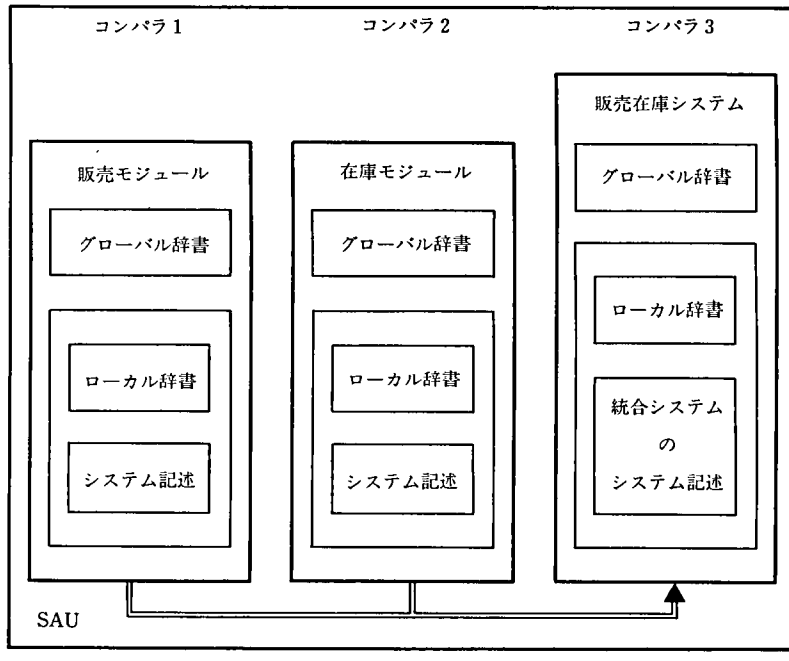


図3 複数開発環境の例

Fig. 3 Development environment on multiple LINC compilers

ラに開発環境を分けた例である．基本的には同一コンパイラの場合と同じであるが，この開発環境は同一コンパイラ上で開発する場合に比べて，パフォーマンス低下が少ないというメリットがあり，比較的ハードウェア能力が小さい場合や，システムの分割数が多い場合に向いている．

3. LINCにおける分散開発の現状と問題点

本章では，LINCを適用した分散開発の現状と問題点を明らかにしたい．

3.1 分散開発の現状

LINCで分散開発が行われるようになってきた要因として，以下の理由をあげることができる．

1) 開発規模の増大
2) 開発形態の変化
3) 容易な分散開発環境の構築

対話型になって大規模開発への適用が容易になってきた反面，システム生成時間も増大してきた．そのため，開発時に必要なハードウェア規模と実行時に必要なハードウェア規模のギャップが大きくなってきた．そのための対処として，LINCの処理効率の向上が図られてきたが，LINCへの機能追加も同時に進められているため，現実には処理効率の向上が相殺されてしまっている．そこで，開発のための環境ができるだけ実行時の環境で済むような開発の仕方が必要になっている．

また，初期のころは開発を自社で行う場合が多かったが，① 開発規模が大きくなるにつれ，委託開発するケースが増えてきたこと，② 業務に精通した開発者がシステムを構築していたのが，分業化され多くのプログラム専門のメンバが開発をするようになったこと，③ 同時に複数のシステムを開発するようになったこと，等により，他の開発環境に影響を及ぼさないような開発環境づくりが必要になっている．

さらに，高性能で低廉なハードウェアの登場によって開発専門の環境が作りやすくなっている．

これらのことにより，LINCでシステム開発を行う場合の分散開発の必要性が日増しに高くなっている．

3.2 分散開発を行う場合の問題点

分散開発を実施する際のキーワードに従って，問題点を明らかにしたい．

3.2.1 分割の設計

開発段階に入り，実際に分散開発の運用を行おうとすると，各モジュール間のファイル共有を最小限にできない場合が多い．これには二つの原因が考えられる．

第一は，分散開発を意識していないために一つのファイルがいくつもの処理と関連づけられた設計をしてしまうことである．第二は，プログラム設計段階でのファイル構造の検討の甘さである．例として，固定項目と変動項目を別ファイルに分けるのか，それとも一つのファイルにするのかを挙げることができる．

このため，ファイルの共有やデータの相互関係が生まれやすいシステムになり，結果としてモジュール分割が困難であったり，分割してもファイルの共有が多くなり，システムの変更に伴う保守作業が増大することになる．

3.2.2 プログラミングの標準化

LINC で分散開発を行う場合は，データ名の付け方が重要な意味を持ってくる．基準の設定はされているが，実施が不十分であったり基準自体が曖昧なケースが見られる．

各モジュールの開発中に問題点が発見されることは少なく，統合時に発見される場合が多い．結果として統合時に不整合を起こし，統合作業をやり直さなければならない場合もある．

3.2.3 開発時の運用ルール

プロジェクト・マネジメントの範ちゅうであるが，開発中の運用ルールが曖昧なケースやルールの実施が不十分なケースが見られる．とくにデータ辞書の運用と仕様変更に伴う開発チーム間の保守作業の調整が正しく行われていない．

最も重要なことは正しく実施することである．正しく実施することを怠ると，統合時の不整合を起こしやすくなることと，各開発チーム間の仕様変更の調整がうまくいかず，開発スケジュールが遅れることになる．

4. LINC における分散開発への提案

本章では，分散開発を行う場合に最も重要である分散の考え方について，一つの提案をする．

4.1 プログラム設計段階での分散の検討

プログラム設計段階での論理設計・物理設計の両面で，プログラミングの分散，モジュールテストの分散，統合の順序，実行時の分散を十分に検討することと，開発時の分散と実行時の分散は別物である，という認識に立つ必要がある．

なぜ開発時の分散と実行時の分散を区別するのかと言うと，開発時の分散は静的なプログラムの動きから捉えるが，実行時の分散は動的（プログラムが実際に動く環境を対象とする）なプログラムの動きから捉えるためである．

ところで，LINC にはサブシステムと呼ぶ分割の機能が提供されている．この機能を開発時に使用する場合は開発時サブシステムと呼び，実行時に使用する場合は実行時サブシステムと呼ぶ．

開発時サブシステムは，関連の強いもの同士を同じサブシステムにすることで，変更の影響を最小限にできる．また実行時サブシステムは，処理の複雑なものと単純なものを分けることにより，処理時間の平均化やレスポンスタイムの重要な処理が他の処理に影響されることを防げる．

この LINC が提供しているサブシステムを適用する場合でも，開発時と実行時は区別して考える必要がある．

4.1.1 プログラミングの分散

ここでの基本方針は，モジュール同士の関連性をできるだけ小さくすることにある．具体的には，ファイルの競合ができるだけ少ない分割単位にすることと，そのモジュール内ですべての関連する処理が完結することである．

4.1.2 モジュール・テストの分散

ここでの基本方針は，各モジュールのテスト環境を独立にすることである．言い換えると，あるモジュールの実行結果が，別のモジュールの実行結果に影響を与えない

環境が作れることである．

具体的には，それぞれ関連のないモジュール同士には同じテストデータを割り当て，関連のあるモジュール同士には，別々のテストデータを割り当てることにより実現できる．したがって，この状態を作りやすい分割を行うことである．

4.1.3 統合の順序

分散開発の最終段階は，できあがった各モジュールを一つのシステムにまとめあげることである．

基本方針として二つの選択肢が考えられる．一つは，同じファイルを使用するモジュールから統合していく方法，もう一つは使用するファイルの重なりが少ないモジュールから統合していく方法である．

前者は不整合が発生しやすく統合作業に遅れを生じやすいが，後者は比較的不整合が生じ難い．したがって，後者の方法は統合の割合を早い時期に高めることができるので，重要なモジュールが早い時期に統合できれば統合システムの完成度が高くなる．

4.1.4 実行時の分散

ここでの基本方針は，実行時の処理効率を上げることである．

処理効率を上げるポイントとしては，次の三つの事柄を挙げることができる．

1) 使用するファイルの共有を減らす，
2) 複雑な処理を分ける，
3) 処理時間が重要なものを分ける，

これらを考慮して，実行時サブシステムの振り分けを検討する必要がある．

4.2 システム分割の指針

一口にシステムを分割すると言っても，その切り口はさまざまであり，それによってシステム開発の効率あるいは生産性に影響を与える場合がある．ここでは，システムの分割の考え方を示す．

1) ボリュームで分割……それぞれの分割単位が均等になるように，ステップ規模で分割を行う．

   図4のケースは，単に開発チーム間の負荷を平均化するために，ステップ規模で分割したものである．この方式の短所は，第一に開発負荷をステップ規模で測れると錯覚していること，つまりプログラミングの難易度を考慮していない点である．第二に各モジュールの独立性が低くなる点である．

2) 難易度で分割……それぞれの分割単位の難易度が均等になるように規模を分割する．

   図5のケースは，開発チーム間の負荷を平均化するために，作業工数で分割したものである．この方式の長所は，難易度で分割しているため，各開発チームの作業工数の配分が平均化する点である．しかし，モジュールの独立性が低い点ではステップ規模で分割する場合と同じである．

3) 機能単位に分割……事務処理の場合，システムがある機能単位に分かれており，その単位で分割を行うことができる．ただし，それぞれのモジュールがファイルの共有という点で独立であれば問題ないが，共通な部分や業務処理の流れに関係する場合，分割に注意が必要である．

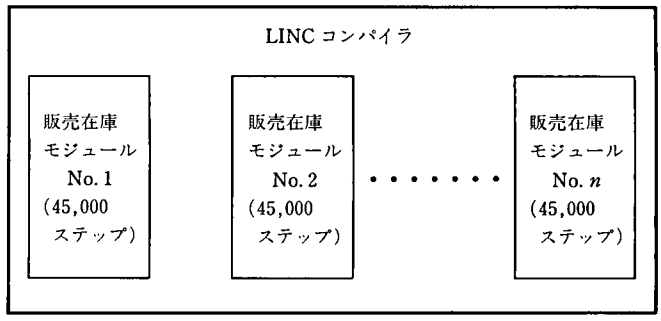


図4　規模による分割

Fig. 4　Decomposition by program steps

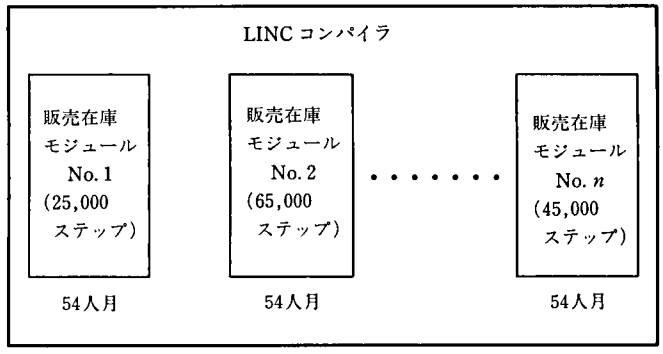


図5　開発工数による分割

Fig. 5　Decomposition by man-month

図6のように，ある販売管理システムを受注・発注・入庫・出庫に分割するようなケースがこれに当たる．

「ボリュームで分割」する場合に比べ，望ましく（論理的に正しそうに）見える．しかし，重要なことはファイルの共有と言う点で独立性を高くすることにある．

一般に言われる構造化分析を取り入れて設計した場合，おそらくこのケースのような形になると思われる．

この場合忘れてならないことは，構造化分析はシステムの論理的な関係（情報の流れ・関連）を表すことに優れているが，それがそのままプログラムの物理的な関係になるわけではないということである．

図6を例にとると，発注モジュールには発注情報として発注ファイルがあったとする．さらにこの発注ファイルは，受注モジュールでは納期情報として参照され，入庫モジュールでは発注情報の消し込みが行われるとする．このとき発注ファイルの変更が入庫モジュールで必要になったとすると，その影響は受注，発注の二つのモジュールにも及ぶことになる．これが開発中のシステムであれば，開発チーム間で作業スケジュールの調整を行うことと，変更が正しく実施されたことを確認することが必要になり，開発効率を低下させることになる．

このような場合，構造化分析の結果（たとえばDFD）にプログラムとファイルの関係を加えて，モジュール間の影響度が最小になるように検討することが必要

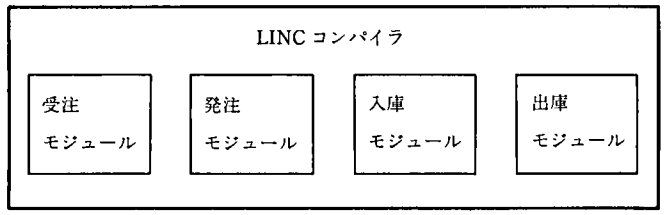


図6　機能単位に分割

Fig. 6　Decomposition by application function

である．

4)　使用業務部門で分割……機能で見れば同じ受注でも，業務部門により内容はさまざまな場合がある．そこで3)で説明したような機能による横割りではなく組織で縦にシステムを分割する．

たとえば，商社のシステムで原料を扱う部門や製品を扱う部門があるような場合，それぞれの部門単位に分割をする．

図7の例は機能単位の分割の改善例であり，変更影響度の減少や一つの機能の中にいくつもの類似の処理を取り込むことによる複雑さを減少することを狙ったものである．

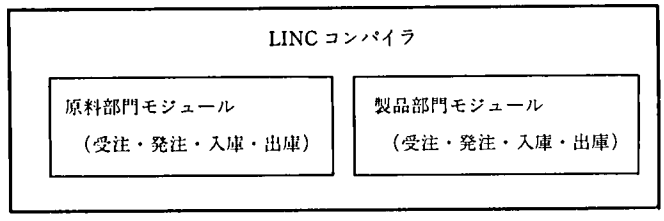


図7　使用業務部門で分割

Fig. 7　Decomposition by sectional application function

業務機能に着目すると，両部門とも同じ業務機能があり影響度が大きいように見える．しかし，原料部門と製品部門で受注ファイルが必要だったとしても，原料と製品では捉え方が違うため，ファイルとしては別の方が都合の良い場合がある．この場合には，同じ受注と言う機能であっても，原料部門と製品部門で関係がないため，他のモジュールの変更とは無関係になり，独立性が高くなる．

5)　サブシステムで分割……LINCでは開発効率や実行システムの効率向上のために，サブシステムと言う分割手段が提供されている．実際の開発単位をこのサブシステムに対応させたものである．

図8の例は，開発の分割を実行時の分割に合わせたもので，分割の検討が一回で済むので一挙両得な考え方である．

しかし，開発時の効率か実行時の効率かどちらかに偏ることになるので，開発中は開発時のサブシステム分割に，統合後は実行時のサブシステム分割に，改めて振り分け直すと効果的である．

6)　冗長度で調整……1)～5)に加え，システム自体の関連性が強い場合，分割をすると共通部分が多くなり各モジュールごとに重複する部分が多くなる．重複部分

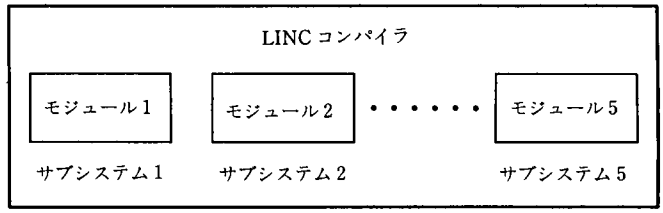


図8　サブシステムで分割

Fig. 8　Decomposition by LINC subsystems

が多くなると各モジュールの影響が強くなり，変更による調整作業が増えて開発生産性が低下する．そこで，図9に示すように重複部分が最小になるように分割の調整を行う必要がある．

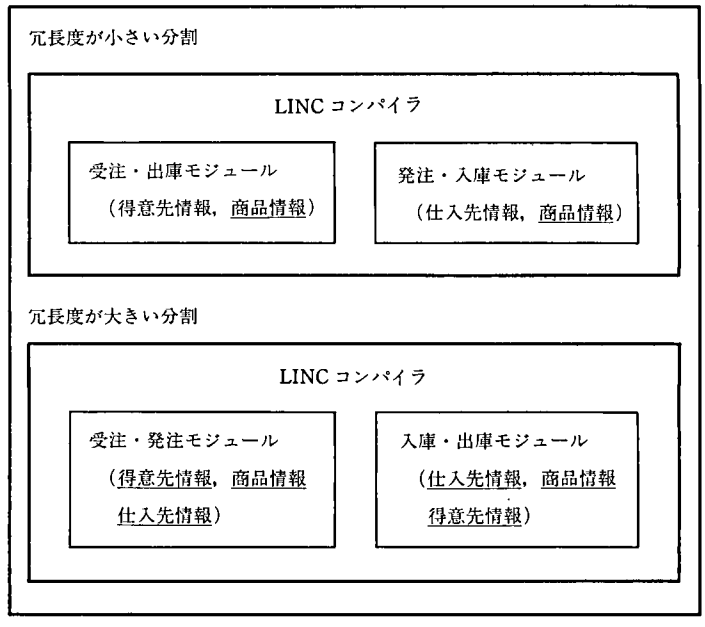


図9　冗長度による調整

Fig. 9　Adjustment of redundancy

「使用業務部門で分割」でも述べたとおり，同じファイルであっても別ファイルとした方が有利な場合と，データベース構造上同一ファイルの方が有利な場合もある．ここで言う調整は主に後者の場合に必要になる．

注意してほしいことは，単に重なりが減ればよいのではなく，いかに変更時の影響を最小にするかが目的だと言うことである．

以上，いくつか分割の考え方を示してきたが，これが絶対の考え方だというものはない．ここで強調したいことは，目的を持ってシステムの分割を行うことと，さまざまな角度から分割の検討を行ってほしいということである．

5. 今後の課題

LINC を適用して分散開発を行うに当たり，今後どのようなことが望まれるのかを

明らかにしたい．

5.1 LINCにおける開発方法論の確立

分散開発ばかりでなく，LINCを適用してシステム開発をする場合の基本的手順とそれぞれの段階で何をすべきなのかを明確にすることである．LINCでは「LSA (LINC Systems Approach)」が計画されており，早い時期での適用が望まれる．

5.2 LINCにおける分析・設計技法の確立

構造化分析，構造化設計に代表されるような分析・設計技法がLINCでは確立されていない．LINC独自の技法が確立できれば望ましいが，少なくとも既存の技法がLINCでのシステム開発に適用できることが必要である．そのための適用技術の確立が望まれる．

5.3 システム開発支援ツールの提供

方法論，技法の適用を進めるために必要なのが，開発支援ツールである．とくにシステム開発の上流工程を支援する「UPPER CASE (Computer Aided Software Engineering)」やプログラミングの標準化や部品化による生産性向上ツールの提供である．これらのものは，「LDA(LINC Design Assistant)」，「LDK (LINC Developper's Kit)」として計画されているが，開発方法論や技法と結び付いた形で提供されることが望まれる．

5.4 開発機能の分散

現在のLINC開発環境は，あくまでもホスト・コンピュータ上で稼働するように考えられている．その限りにおいて，根本的な開発負荷の分散は図れない．したがって，ワークステーションを利用した開発機能の分散が望まれる．

6. おわりに

LINCで分散開発を行う場合の問題点について述べてきたが，分散開発が本格化してきたのはここ2年程のことである．結果的にはまだ十分な開発環境が整っているとは言えない．しかし，今後大規模なシステム開発が増えていくこと，分散開発が要請されていることを考えると，「LINCに対して的確な機能要求を行っていくこと」，現在作成を目指している「LINC適用ガイド」の必要性，「開発方法論・開発技法の確立」を強く感じるものである．

執筆者紹介　山　科　順　一 (Junichi Yamashina)

昭和30年生まれ．53年専修大学経営学部情報管理学科卒業．同年日本ユニシス(株)入社，卸売を主体とする流通ユーザを担当し，64年よりLINCの利用技術の普及を職務とする．現在，システム技術本部4GL技術部所属．

LINCの持つオブジェクト指向性と今後の問題点

LINC's Object-oriented Approach and Its Problems Requiring Future Solutions

福井　務

要　約　近年オブジェクト指向が話題になっている．オブジェクト指向は，プログラミング，データベース，および人工知能の分野において，それぞれが持っていた問題点を解決するために発生し体系化したと考えられる．オブジェクト指向では，“オブジェクト”，“属性”，“インスタンス”等の概念が整理されており，これらを使用し実世界や情報をモデル化する．オブジェクト指向を取り巻くソフトウェアとしては，“意味データモデル”，“オブジェクト指向プログラミング”，“オブジェクト指向データベース”等がある．

LINCはEVENT，COMPONENTという概念で，ビジネス活動をモデル化するオブジェクト指向性があり，単位活動業務，リソース維持管理活動，ビジネスルール，活動の統合等を中心にモデル化する．また，ビジネスの成り立ちを明確にモデリングでき，モデリングされたものを直接取り扱うことができる．

しかしながら，LINCの概念の有効利用，および機能上でのオブジェクト指向においては課題も多い．

Abstract　What is called an object-oriented approach has recently become a popular topic of discussion. This approach is considered to have emerged (and have been formed in systematic order) to give solutions to respective problems pertinent to the fields like programming, database and artificial intelligence.

The object-oriented approach has such concepts established as 'objects', 'attributes', 'instances', etc. which are used for enabling what really exists, including information, to be modeled.

The software repertoire for this approach includes 'semantic database', 'object-oriented programming', and 'object-oriented database'.

LINC provides an object-oriented approach which helps model business activities under the concepts of 'EVENT' and 'COMPONENT'. Using these concepts makes it possible to model each activity unit, resource retention management, business rules and integrated activities after analyzing business actions.

LINC serves to put business frameworks into precise models and is capable of directly handling the models. However, LINC also has problems yet to be solved regarding more effective use of the LINC concepts and its object-oriented approach functionality.

1. はじめに

オブジェクト指向という言葉が，さまざまな分野でしきりに使われるようになってきたが，オブジェクト指向とはどのようなものか，なかなか捕えずらいものである．

当社のAシリーズ汎用コンピュータには“意味データモデル”をインプリメントしたSIM（Semantic Information Manager）があり，オブジェクト指向との係わり合いを整理しようとしている．

LINCはSIMより5年ほど早く提供され，オブジェクト指向との係わり合いに関す

る議論はあまりされていないが，LSA（LINC Systems Approach）の中には，オブジェクトモデルという言葉が出てくる．

このような環境下において，本稿第2章では，オブジェクト指向の持つ一般的概念を整理し，オブジェクト指向の概念および特質を明確にする．

また，神戸大学・田中克巳氏のオブジェクト指向関連レポート等に基づき，意味データモデルを含め，オブジェクト指向を取り巻くソフトウェアの特徴を整理する．

この後第3章より，LINCにおけるオブジェクト指向性を評価していき，続く第4章で，課題と今後の期待をまとめる．

2. オブジェクト指向の一般的概念

2.1 オブジェクト指向の出現

近年計算機のあらゆる分野で，オブジェクト指向が話題になっている．Smalltalk-80あるいはC++に代表されるようなオブジェクト指向プログラミングにとどまらず，オペレーティングシステム，シミュレーションモデル，計算機モデル等の基礎的な分野より，データベース，知識表現，情報システム設計方法論等，応用分野に至るまで広範囲に及ぶ物事の考え方を支配するものとして発達しつつある．

しかし，その実体や応用はなかなか捉えにくいものがある．オブジェクト指向の起源や背景は統一的に把握することが困難であり，このため次の三つのことがらより整理する．

2.1.1 プログラミングからの流れ

プログラミングにおけるオブジェクト指向とは，“モジュール”や“抽象データ”あるいは“部品化”“抽象化”というソフトウェア工学の背後にある概念の発展形態として位置づけることができる．

Smalltalk-80に代表されるオブジェクト指向プログラミング言語の主な特徴は，

① データと手続きのカプセル化（オブジェクト＝データ＋手続き）による抽象データ型の支援，

② メッセージ・パッシングによる計算メカニズムと情報隠蔽機能，

③ 差分プログラミング・部品化を可能にするオブジェクトの汎化階層に基づいた継承（インヘリタンス）機能，

等といわれ，手続きを重要視する“振る舞い的オブジェクト指向”といわれている．

2.1.2 データベース技術からの流れ

データベース技術からの流れとしては，定型的な処理によって，データを蓄積し利用するための“データ蓄積”と“データ利用”を表現するデータモデルがある．

これは，プログラミング言語やシステムが処理や手続きといった動的な問題解決のアプローチであることに対し，静的なアプローチであった．

この中で，実世界に存在するデータの持つ意味を考え，ありのままにこれを表現し，データを蓄積し，データの持つ意味に従い処理を行う“意味データモデル”，あるいは意味データモデルに“振る舞い”の要素を付加し，さらに動的なモデリングを目指す“オブジェクト指向データベース”へと変化しつつある．

1) 意味データモデル(SDM)……意味データモデルは，実世界の情報構造を直接データベース構造に反映するためのさまざまなスキーマ構成要素を有するデータモデルであり，エンティティ関係モデル，関数型データモデル，IFO モデル等があり，SDM に基づく商用 DBMS として当社の SIM がある．

これら意味データモデルの主な特徴は，

① 属性値と実体（entity：オブジェクト指向ではインスタンスといった）・関係（relationship）の区別，

② 複合オブジェクトの直接表現能力，

③ 実体間の意味的関連を表現するための豊富なスキーマ構成要素（汎化・集約等），

④ 永続的なデータ（永続的オブジェクト，persistent object）の支援，

等といわれており，データ構造を重視する“構造的オブジェクト指向”であるといわれる．

2) オブジェクト指向データベース……上述の意味データモデル，オブジェクト指向プログラミング言語が源流となり，関西では神戸大学(田中克巳氏)，大阪電気通信大学（永田元康氏）等，多くの大学，研究所で研究が成されている．

この主な特徴は，

① オブジェクト識別性（object identity），

② 複合オブジェクトの支援，

③ データと手続きのカプセル化，

④ 階層型（type hierarchy）の支援，

⑤ 永続的オブジェクト（persistent object）の格納・管理機能，

であり，従来の意味でのデータベース管理システムというよりも，プログラミング言語と融合した新しい位置づけでのソフトウェアということができる[6]．

2.1.3 人工知能の流れ

人工知能研究での知識表現においては，知識自体を知的作業のためのデータとして蓄積することが必要であり，この知識表現はデータベース分野におけるデータモデルと非常に似た点がある．このためオブジェクト指向へと体系化していった．

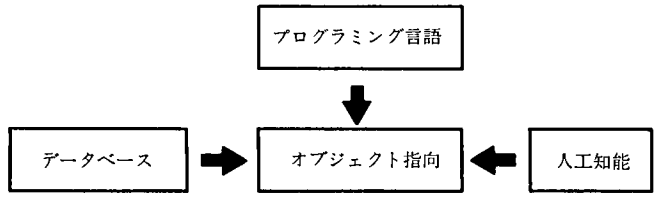


図1 オブジェクト指向の起源

Fig. 1 Origin of objrect-oriented

これら三つの事柄は同時発生的に提起されてきた．

またこれらは，情報処理として包含されるさまざまな応用分野における解決すべき問題をクリヤするため，必然性を持ち，オブジェクト指向へと向かっていったことが

重要な事実であると思われる（図1）.

たとえば，システムのライフサイクルにおける分析・設計・開発・テストにおいてもオブジェクト指向性が要求される.

最近話題が多い上流CASE (Computer Aided Software Engineering) においては，下流工程とのつなぎをどうするかが一つのキーポイントである．システムのライフサイクルを全体にわたって一つのモデルで押し通すことができることの意義は明白で，オブジェクト指向はこのようなモデルの最有力候補の一つである[4]．ワークステーション，パソコン等で稼働する上流CASEツールにおいては，単なるツールだけでは効果は出ず，これを支援する設計方法論が必要不可欠であるといえる.

2.2 オブジェクト指向の必要性の背景（なぜ概念が重要であるのか）

情報処理システムを開発するとき，「ユーザのニーズに従い，開発・修正が容易なシステムを築くのはどうしてこんなにむずかしいのか」と誰しも困難さを経験する．これらに対しては次に述べる事柄が問題点であるといわれている.

1) 専門分野以外の仕事……情報処理システムが専門である開発者が，自分の専門以外であるアプリケーションに取り組む場合，早急に業務に順応することを迫られることになる．こうした時の問題点は，突然多くの事柄を学ぶ必要性があることに対し，学習のための特別な時間は用意されず，学習過程を助ける体制・方法も用意されないことである.

システムの基礎となるプログラムは業務専門分野の知識に基づくべきであり，誤認識は致命的エラーとなる.

2) 多種多様な専門用語……販売物流システムを考えた場合，輸送部門におけるOrder（順番）と，販売部門におけるOrder（注文）とは明らかに違ったものである．また製造部門におけるリードタイム（発注—入荷）と，EDP部門におけるリードタイム（File Read Time）もそうである.

このように同じような用語群が持つ異なった概念が一つの組織に存在している中で，概念と用語を整理し，真にシステム開発を行うためには，知的な度量を傾け，考えうる差異を調べ，これらを解決することが求められる.

しかし，システムの要求仕様を記述する際には，誰しも用語に対する厳密な定義に確信が持てず，混乱は避けられないと思われる.

3) 情報の変更……開発の着手時より実稼働を迎えるまでには，さまざまな情報が変更される．ここでシステム開発者に必要とされることは，ソフトウェア開発にどのように影響を及ぼすか判断することのできる能力である.

4) 曖昧さ……専門外の分野であればあるほど，自分は何を知っており，何を知らないかを認識することは困難なことであり，また何を知ろうとしているのか整理できないこともある．システム開発のスペシャリストとしての立場であれば教養が知識の欠陥をさらしにくくし，質問することに不安を抱いてしまう．ここでは，集められた各々の情報の個々の事実が観察でき，現在の認識を提示できるような方法が必要となる.

5) 誤った専門知識……多岐多様なシステム開発においては広範に冠たる専門知識が必要とされ，たとえスペシャリストといえども完璧に対応することは不可能で

ある．ここで重要となることは，情報と事実とのチェック機構である．

6) 詳細の定義……全体の状況認識が確立されていない状態において，一つの問題解決に対し満足のいく定義を確立することは非常に困難なことである．ここで必要とすることは，一連の概念群についての諸定義をいくつも候補として並べ，それら定義の持つ意味の関連を検討することのできる方法である[1]．

2.3 オブジェクト指向での実世界のモデル化

オブジェクト指向に基づき，実世界をモデル化することにより2.2節で述べたようなプロジェクトの失敗を避けることができる．

実世界をモデル化する手順としては，まず実世界に存在する情報をモデル化する．情報モデルとは，ある問題領域にある事柄を定義し表現しようとするものである．そしてこの表現は，問題となっている用語と概念群を，記述定義するのに適した機構と図形表現によって構成され，高度に構造化させ表現する．

モデルは考察対象となる実世界に内在する事象を，識別し，区分し，抽象化されるが，問題の中での類似の事象ないし実体を“オブジェクト”として識別し抽象化する．またオブジェクトの持つ特徴は“属性”として抽象化し，その中での個々の事象を“インスタンス”として表現する．さらにオブジェクト間のはっきりと確認できる結び付きは，“関係”（relationship）として抽象化する．

情報モデルの理念は，“オブジェクトを識別し，これらのオブジェクトの属性を明確にし，オブジェクト間の関係を整理する”ことであり，正確に首尾一貫してこれを行う方法である[1]．

抽象化においては，厳密に定義され，明確にされたルールに従い，行うことが重要な点である．

情報モデルが定義されても，実世界をモデリングしたことにはならない．あるオブジェクトのインスタンスは，一種の“ライフサイクル”をたどることになる．たとえば販売物流システムにおいて，顧客に注文は，図2のように状態が変化する振る舞いをも定義する必要がある．

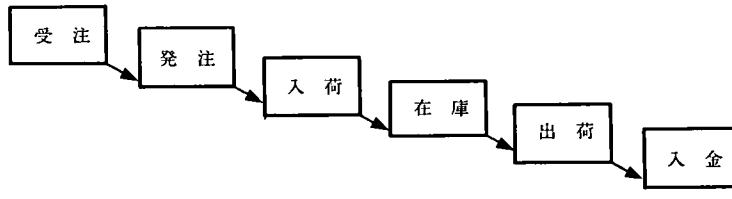


図2 インスタンスのライフサイクル
Fig. 2 life cycle of instances

ソフトウェアが持つオブジェクト指向性を評価する場合，振る舞いの定義を含め，実世界をどのようにモデル化し，モデル化された実世界をどのようにソフトウェア上にインプリメントするかが大きなポイントとなる（図3）．

LINCは，ビジネスという実世界をモデル化するために開発されたソフトウェアであり，モデル化されたものから直接コンピュータ上に稼働するソフトウェアとしてマッピングすることができる．

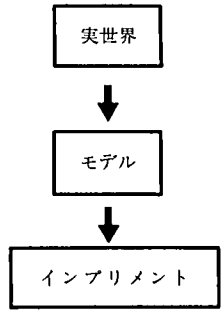


図3　インプリメントの手順
Fig. 3　Flow of implement

3. LINCにおけるオブジェクト指向性

LINCは1982年より出荷されたが，その当時はオブジェクト指向といわれるような考え方は一般的でなく，少なからずとも日本においてはCOMPONENT，EVENT，PROFILEといった物が何であるかを模索しながら対応してきた．

1987年頃よりULDC（UNISYS Linc Development Center）より，LINC本来の業務開発コンセプトとしてD & I (Design & Implementation) が出され，LSA (Linc Systems Approach) として出荷当時より言おうとしていた事柄が整理されようとしている．

前章で述べたように，オブジェクト指向という言葉自体に“こういうもの”とした定義がはっきりと確立しているわけではないが，一般的にいわれているオブジェクト指向としての特質を，

- 構造の直接表現
- データと処理のカプセル化
- 複合オブジェクト
- 継承を表現する

としてみた場合，LINCにおいては継承を表現する構造はないが，不完全ながらデータと処理のカプセル化の機能があり，COMPONENT，EVENT等でビジネス活動を抽象化する手段を持っている．これらの機能により，トップダウンに物事を整理しようとする考え方はオブジェクト指向であるといえる．

ビジネス上のさまざまな事柄をオブジェクトとして捉え，オブジェクトごとに入出力画面，処理ロジック，データベース構造の三つをカプセル化し，これによりビジネスを直接表現する．

ただし現在の導入開発適用例を見ると従来の開発ステップによるボトムアップとしての事例が多数ある．筆者はこの原因は，データベースエンジンであるDMS II (Database Management System II) の持つ機能等を意識し過ぎたことより発生するものと考える．ともあれ，オブジェクト指向的な思考は，処理しようとする対象物が複雑であればあるほど必要であり，今後これらに対応するソフトウェアにおいては必須であると考えられる．

本章ではLINCが持つビジネス活動の直接表現，つまりLINCによるビジネスのモデリングを考え，どのようなものであるかを分析する．

3.1 LINCビジネスモデリング

LINCは企業というものを前提にモデリングを行う．企業は営利・利潤を追求するための活動を行っているが，通常組織だって行われる．物を仕入れて販売を完了するまでの活動を考えると，

物を仕入れる➡ストックする➡注文を受ける
➡在庫を引き当てる➡出荷する➡請求する‥‥

といったことになるが，このような作業だけではビジネスとしては成り立たず，次のような事柄があって初めて作業ができるのである．

何を仕入れるか検討する．
↓
仕入れの条件を決める．
↓
どこから仕入れるか検討する．
↓
仕入先を管理する．
↓
入荷する商品を管理する．

販売の条件を決める．
↓
どのようなマーケット戦略を
↓　行うか検討する．
販売先を管理する．
↓
販売すべき商品を管理する．

これらは，現業部門・企画部門・管理部門等に別れビジネスとして活動する．このように企業組織は分業化されるが，全体として統一された企業目標に対して，同期した活動にするためのルールが定められる．

これらがそろって初めて企業として活動できるのであるが，上の事柄は以下の要素に整理することができる．

1) 単位業務活動……企業は組織化され分業となり，各機能単位の業務活動となる．たとえば，仕入部門は物を仕入れる活動を行い，受注部門は注文を受ける活動を行う．
2) リソース維持管理活動……仕入部門や受注部門が活動するための，仕入先管理や販売先管理等ビジネス資源の管理活動を行う．
3) ビジネスルール……企業の目標を達成するために，それぞれのビジネス活動においての公式・非公式に日常行われているビジネス慣習．
4) 活動の統合……単位業務活動は一貫した流れ作業の中にあり，単位業務が終了するか，行っている最中に他の単位業務に対しコミュニケーションを取り活動の統合を計る．

ここで重要なことは，“単位業務より出されたメッセージが最終的にどのようになったか”ということより，むしろ“必ずメッセージを渡す”という責任を負っていることである．つまり，単位業務自身は流れを意識せず，独立した物としてモデリングが可能ということである．

LINCでは，この事実を重要視し，上記1)～4)を直接表現するよう対応付けされている．

3.2 ビジネスモデルのソフトウェア構造へのマッピング

3.2.1 ビジネス活動を直接表現するEVENT, COMPONENT

LINCが持つ要素として,

EVENT

COMPONENT

がある．これらの定義はインタフェース仕様（ISPEC）と呼ばれ，ビジネス活動を表現する概念である．

1) EVENT……ビジネス活動の入荷・出荷・請求・入金等の活動はEVENTとして定義する．たとえば，EVENT；“出荷業務”とした場合出荷業務を行うために必要なすべての事柄が表現される．

●出荷作業を行う際に必要となるデータ

何を 何個 何処から 何処へ

●活動を行う際のマンマシンインタフェース

面画およびキーボードアクション

等

●出荷作業そのもの

取り扱い可能なものなのか 取引可能な相手なのか

等の確認作業

確認された後の企業内業務慣習 出荷登録作業

等

2) COMPONENT……出荷作業を行うためには“取り扱い可能なもの”が何であるのか，また“取引可能な相手”はどこかを維持管理する活動が必要となるが，これらはCOMPONENTという概念で表現し定義する．

COMPONENTは，EVENT活動の環境を支える活動である，ということができ，COMPONENTには“取り扱い可能なもの”，“取引可能な相手”を維持管理するすべての事柄が含まれる．

すべての事柄とは，

●維持管理を行う際に必要となるデータ

たとえば“取り扱い可能なもの”の場合

ものの名前 ものの値段

“取引可能な相手”の場合

取引相手の名前 取引相手の住所

等

●維持を行う際のマンマシンインタフェース

画面およびキーボードアクション　等

●維持する作業そのもの

二重登録はないか
変更削除はできるか　等の確認作業

確認された後の企業内業務慣習
登録変更作業　等

である．これらをCOMPONENT；“取扱品”，COMPONENT；“取引先”とし定義し表現する．

3) EVENTとCOMPONENTの関係……EVENT；“出荷業務”のモデリングにおいては，COMPONENT；“取扱品”および“取引先”により維持管理されているデータは，プロセスを定義することなく自動的に関係付けられ，すべて参照可能な状態になっていることを前提に作業することができる．

このように，EVENTおよびCOMPONENTはそれぞれの活動を表現し，さらに両者の関係付けをも表現する．関係付けのためのプロセスを廃絶し，ビジネスの観点よりこの両者を関係付け，ビジネス活動を直接表現しようとするものである．

図4に出荷活動とCOMPONENT環境をLSAでのビジネス構造表現手法“オブジェクトモデル”で表現する．

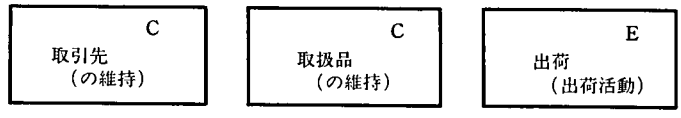


図4　オブジェクトモデル図(1)

Fig. 4　Object model (1)

EVENTおよびCOMPONENT各々の関係が表現されていないように見える．これは，「各々の活動はそれぞれ独立したものである」ということを強調したものであり，「定義そのものも独立し表現すべきである」というところからくる．

3.2.2　ビジネスルールを定義するLOGIC

EVENT，COMPONENTはビジネスの大枠を直接表現したが，各企業が持つ活動慣習の詳細は表現しきれない．このため，LOGICで対応しようとするが，ここでは詳細活動慣習をいかに表現しているかを見る．

1) データ表現……EVENT；“出荷業務”を支えるデータは，出荷作業に必要となるデータと作業環境を支える“取り扱い可能なもの”“取引可能な相手”を維持管理するためのデータからなるが，これらのデータ表現においては，それぞれの所在が明らかとなる工夫がなされている．

たとえば

取扱品　～　を　取引先　～　へ　～　個　出荷する。

とした場合，取り扱い可能なものの名前は

〈取り扱い可能なもの〉.〈名前〉

として表現され，同じ名前であっても取引可能な相手の名前は，

〈取引可能な相手〉.〈名前〉

として識別できる．ビジネス慣習の定義において「このデータはこの活動で発生したもの」ということが，開発者の個性に左右されることなく完全に明確になり，第三者への業務表現力としても高く評価される．次に示す“処理表現”と合わせ，構造の直接表現ではなく，処理の直接表現として評価できる．

2) 処理表現……EVENTおよびCOMPONENTにおいては活動にともなうデータに対して

参照する　蓄積する　変更する　転記する

等，活動のおおむねが表現される．残された各企業特有のビジネス慣習は処理手続きとして定義することになるが，重要なことは，これらを単位業務として定義したEVENTあるいはCOMPONENTの中に閉じ込めることができる点である．このことにより，単位業務に集中してモデリングしマッピングすることが可能となる．

3.2.3 活動の統合

1) PROFILEによる活動結果の参照……単位業務の活動結果として発生したデータは，データベースに蓄積されるが，蓄積されたデータを他の活動等により目的に応じて参照しなければならない場合，PROFILEという概念を使用する．

これは，配送活動より発生した出荷情報を“出荷状況”として参照したり，荷受け活動より発生した入荷情報を“入荷状況”として参照し，さらに出荷情報と入荷情報の総合した結果より“在庫状況”として参照すること等を可能とする（図5）．

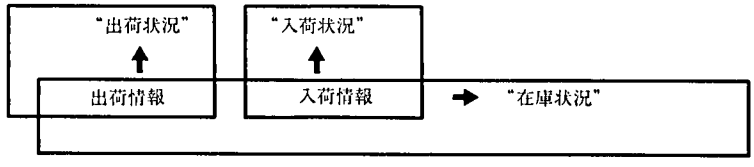


図5　活動結果の参照

Fig. 5　Reference of activity results

2) FLAGによる活動情報の変遷制御……複数の単位業務の活動結果として，ある一つの情報は終結し実績として残される．たとえばある受注情報は出荷情報に姿を変え，請求情報，入金情報へと変遷していきその生涯を終える（図6）．

LINCは，これらの対応をFLAGという処理概念で解決しようとしている．

LINCの業務開発コンセプトでは，最初に発生した情報はデータベースに格納するが，変遷していく過程では新たに別の情報として，データベースに格納したりせず，データベース内にユニークに存在する値を，必要とする各単位活動が利

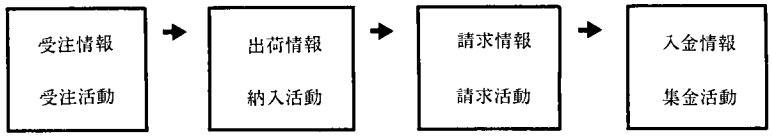


図6 活動の変遷
Fig. 6 Transition of activity

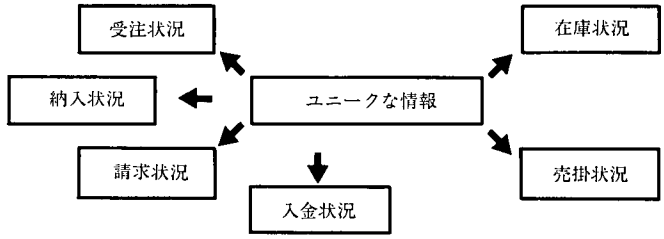


図7 ユニークな情報のビュー
Fig. 7 View of unique information

用できる構造にしている（図7）.

各単位活動ごとに自分に関係するデータが覗けるビュー（PROFILE）を設定し，このビューを通し処理を展開するが，情報が変化し他の単位業務のビューへ情報を渡すとき単位業務内のロジックで“FLAG（状態の変更）”という行為を施す.

これにより生のデータを各単位業務が利用できることとなる.

3) AUTO. ENTRY による他の単位活動の誘導……さらに各単位業務の複雑度合いが増し，情報を他の単位活動に対してのビューに渡すだけでは自分の業務が全うしきれない場合，他の単位活動を誘導実行させ自分の業務を全うさせる機能がある.

たとえば，「得意先別に単価を設定する活動があり，ある層の顧客については受注活動時に単価を再設定しなければならない」といったような場合，受注活動時に得意先単価維持活動を誘導しなければならない.

このような場合，受注活動ロジック内部で

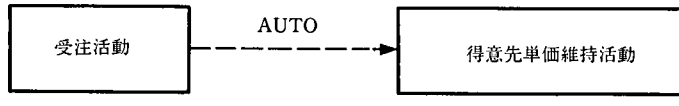


図8 オブジェクトモデル図(2)
Fig. 8 Object model (2)

AUTO. ENTRY：〈得意先単価維持活動〉

とし，複合した単位活動（オブジェクト）に対処することができる（図8）.

3.3 LINC ビジネスモデリングの有効性

従来ウォータフォール型による開発フェーズでは，次のようであった.

業務分析を行う.
↓
要求定義を行う.
↓
設計を行う.
↓
コーディングを行う.
↓
テストを行う.
↓
導入する.

この作業においては，以下の事柄が問題となる.

① 分析結果を要求した側が理解できるよう定義できえたか.
② 分析結果を開発する側が理解できるよう定義できえたか.
③ テストの結果問題が発生した場合再度，業務分析を行うのか.

これらに対しオブジェクト指向分析アプローチでは，実世界の成り立ちをコンピュータ上で組み上げ，作動させるために次の手順を踏む.

実世界の成り立ちを理解する.
↓
関連するプロジェクトの各メンバが相互理解し
確認できるよう，概念を明確にする.
↓
明確化された概念を使用しモデル化する.
↓
モデリングされたものを表記する.
↓
表記されたものをソフトウェア構造に変換する
ための作業を行う.

この作業では実世界がそのままモデル化できるため，利用者と開発側とが同一のレベルで確認したものをソフトウェア化することができる.

さらにLINCではより強力にこれらに対処できるよう，実世界の範囲を“ビジネス”と限定し，ビジネス活動に立脚した中で，非常に明確なモデリング手法を提供している．これは，EVENT, COMPONENT, PROFILE等の概念を理解することにより現実化する.

この概念により組み上げられたモデルをシステム化するための，作業に対応したシステムがあらかじめ用意されているため，モデリングよりソフトウェア生産まで一貫した体制を敷くことができる.

実世界の成り立ちを理解する.
↓
関連するプロジェクトの各メンバがLINCの概念を理解する.
↓
EVENT, COMPONENT等により，モデル化する.
↓
モデル化されたものを直接LINCで定義する.

このように，ビジネス分野におけるシステム“LINC”は，正にオブジェクト指向システムとして位置づけることができる．

この事実により，以下の二つの効果が得られる．

① ビジネスの実世界をそのまま表現し，モデル化することができる．このため，目的および問題点も明確にされ，これらに明確に対応したシステムが構築できる．

② モデル化されたものから実際に稼働するソフトウェアまで，一貫した概念で押し通すことができるため，ソフトウェア開発工程が一元化され，開発生産性および品質が向上する．

また，これらの効果はSIS（Strategic Information System）の構築の基礎になりえる事柄である．

4. 課題と今後の期待

オブジェクト指向システム分析アプローチでのLINCは非常に評価できるものと思われる．しかし半面，旧態依然としたプロジェクトしか組めず，LINCを有効利用しえなかったケースも多く存在する．

また，オブジェクト指向プログラミング，意味データモデル，オブジェクト指向データベース等，現在研究開発されているソフトウェアとしての観点では課題は多い．

本章ではこれらの問題点を明確にし整理する．

4.1 LINC概念の有効利用

EVENT, COMPONENTにおいて，多くのプロジェクトではEVENTの概念が無視されている．これには二つの要因があると思われる．

一つにはDBエンジン上での物理的な構造から来る問題がある．EVENTを概念どおり使用した場合“レスポンス”，“パフォーマンス”，“運用”，“運営”において支援しきれていない部分があることである．この部分は今後の改善に期待したい．

残る一つはSE，つまりソフトウェアを開発する側に問題があることである．通常，開発者はエキスパートであり，エキスパートとは特殊な技能と経験を基本として作業を行う．このため新しい手法，考え方等に対しては保守的であり，旧態依然としたアプローチになりがちである．

しかし，ハードウェアの進化に対応してきたように，ソフトウェア開発における手法，考え方等の進化にも対応する必要があると思われる．

これらに対しては教育等の，考え方を切り替える何がしかのパワーが必要である．

4.2 機能上でのオブジェクト指向性

1) インヘリタンス……現在LINCでは継承（インヘリタンス）という考え方はまったくない．しかし，あるビジネス構造を分析すると，いずれかのビジネス活動が別のビジネス活動に継承されているということがある．

たとえば“入荷活動”は「物を倉庫に入れる」という“倉入活動”があり，“返品活動”においても「物を倉庫に入れる」“倉入活動”がある．こうした場合，“入荷活動”は“倉入活動”を継承し，“返品活動”も“倉入活動”を継承する（図9）．

LINCにおいてこのような「継承している」という事実を表現する機能がほし

図9　ビジネス活動における継承

Fig. 9　Inheritance of business activity

い．“複雑な業務”というものは“単純な業務”の組み合わせから発生していることが多い．“複雑な業務”に地域性等が加わり，さらに複雑となる．

2)　メッセージパッシング……継承とは別に“単純な業務”を定義し，これらの基本機能を有機的に結び付け“複雑な業務”を定義していきたい．

たとえば「ISPEC“出荷業務”は，ISPEC“単価再計算作業”とISPEC“在庫調整作業”も含む」とした場合，図10のように構造化される．

また，これらは図11のように拡張される．

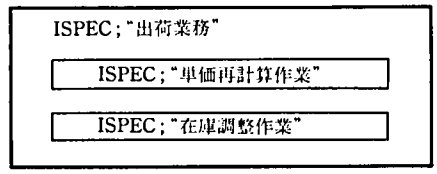


図10　ISPECの構造化

Fig. 10　Structured ISPEC

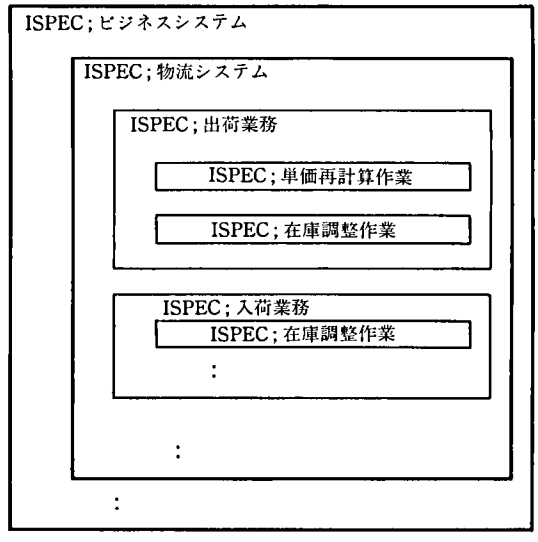


図11　構造化の拡張

Fig. 11　Extended structured ISPEC

ビジネスを直接表現する難易度は，いかに複雑なビジネス慣習を持っているか，によるものと思われる．こうした場合，構造化されたビジネス業務そのものを直

接表現できれば，大きな効果が期待できる．

図11のように構造化された業務を有機的に結び付けるためにメッセージパッシングを使用する．LINCにおけるメッセージパッシングの具体的解決案としては“AUTO. ENTRY”の機能拡張に期待したい．

現在，AUTO文は，次に示すように単なるレコードの書き込みとして使用されているケースがほとんどある．

Auto. entry；〈EVENT名他〉
Auto；〈移送するデータ〉　〈移送されるデータ〉
Auto；〈移送するデータ〉　〈移送されるデータ〉
Auto；〈移送するデータ〉　〈移送されるデータ〉
Auto；Write ←レコードの作成

このような使用形態であれば第三世代言語と全く変わらないことになる．

しかし，COMPONENTにおいては，筆者が主張したい要素が含まれる．

下の例に示すようにCOMPONENTに含まれる〈作業指示欄〉に追加・変更・削除のいずれかの指示を与えることによりCOMPONENTの持つ概ねの活動が実施される．

これは独立したCOMPONENTの活動をプロセスとして稼働させることになる（メッセージパッシング）．

Auto. entry；〈COMPONENT〉
Auto；〈追加指示〉 or
〈変更指示〉 or 〈作業指示欄〉
〈削除指示〉
Auto；〈移送するデータ〉 〈移送されるデータ〉
Auto；〈移送するデータ〉 〈移送されるデータ〉
Auto；Write ← **COMPONENTが持つ活動**

しかしながら概ねの活動でありビジネス慣習を含んだCOMPONENTの活動すべてではない．

筆者は現在のAUTO文の機能は不十分であり，AUTO. ENTRYされたISPECそのものの活動を行うべきであると考える．

さらに，ISPECそのものの活動には，活動してよいかどうかの検証も含まれるが，辞書項目に指定したVALUEの範囲もAUTO文の対象としたい．

この機能によりデータベースに格納されるデータは，辞書に示した値を保証し，かつISPECで定義された“業務内容も保証”することとなる．

これらが実現されたとき“ISPECの構造化”で直接表現された業務が有機的に結び付く．

出荷業務を例にすると，EVENT；“出荷業務”のビジネス慣習として，

AUTO. ENTRY；“単価再計算作業”
AUTO. ENTRY；“在庫調整作業”

と定義することにより出荷業務発生時に正確・確実に単価計算作業と在庫調整作業が成されることとなる．

ISPECモデルによる表現を図12に示す.

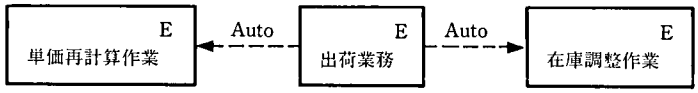


図12　オブジェクトモデル図(3)
Fig. 12　Object model (3)

このことはISPECにビジネス活動の直接表現を求めるだけでなく，処理そのものに抽象化された直接表現を求めることができる.

これが発展するとISPECの利用はより構造化される.

3)　カプセル化……3.2節で述べたように，LINCではデータベース構造とこれに対する処理がカプセル化されているが，さらにこれを拡張し，データ項目そのものにも処理をカプセル化したい．これは，よりビジネス指向仕様言語であることを望むからである．具体的なインプリメントの例としては，前述のメッセージパッシングとからめ，LDLの例として次のようなことが考えられる.

COMPONENT；基本給計算メソッド
　DATA；職級
　DATA；職階
　DATA；基本給
　MULTIPLY；(100000) 職級
　DIVIDE；職階 職級 GIVING；基本給
　RECALL；

COMPONENT；給与台帳
　DATA；社員コード
　DATA；職級
　DATA；職階
　DATA；基本給　AUTO. ENTRY；基本給計算メソッド

　給与台帳は，社員コード・職級・職階・基本給よりなる.
　基本給そのものには値を持たず，“COMPONENT；基本給計算メソッド”へ，職級・職階の値と共にメッセージパッシングした結果として参照される.

5. おわりに

以上，筆者なりにLINCの持つオブジェクト指向的な部分を取り上げ評価してみたが，現実のLINCを用いた導入開発ではオブジェクト指向的であるか，そうでないかはほとんど重要視されないかと思う．しかしソフトウェアが持つコンセプトを十分理解し適用することは重要なことである．と同時に，本来どうあるべきであるかということを考察することも重要であると考える．これらの考察を通し，LINCはオブジェクト指向でモデリングすべきであると考える.

現在，LINCのコンセプトに則った上流CASEツールLDA (LINC Design Assistant) の提供も検討されており，またSIS (Strategic Information System) 等も絡み，今後ますますLINCにおけるオブジェクト指向の論議は盛んになると思われる.

今回問題点として挙げた事柄は，現在のソフトウェア技術で実現可能範囲で述べたつもりである.

参考文献
［1］ S. シュレイヤ，S. J. メラー，オブジェクト指向システム分析，啓学出版.
［2］ P. Coad, E. Yourdon, Object-Oriented Analysis, Yourdon Press.
［3］ 酒井博敬，堀内一，オブジェクト指向入門，オーム社.
［4］ 春木良且，オブジェクト指向への招待，啓学出版.
［5］ T. DeMarco，構造化分析とシステム仕様，日経 BP.
［6］ 田中克巳，オブジェクト指向データベース・システム その背景と概念，bit Vol. 20, No.6, pp.83～90.
［7］ 田中克巳，オブジェクト指向データベース・システム 研究・開発の動向，bit Vol. 20, No. 7, pp.51～61.
［8］ オブジェクト指向を軸にソフトウェア革命が始まる，日経コンピュータ 1989，5月 pp.66～134.
［9］ 山下淳一，LINC 開発技法「LSA」，Systems No.212，pp.50～54.

執筆者紹介 福 井 務（Tsutomu Fukui）

昭和 27 年生. 50 年甲南大学理学部経営理学科卒業, 52 年日本ユニシス(株)入社. 流通業担当 SE サービスに従事した後, 58 年より LINC 中心にシステムサービスを行い，現在に至る. 関西支社システム技術 1 部に所属.

LINCによる一開発形態 ——概要設計即プログラミングの是非について

One Pattern of Systems Development by LINC ——The Validity of Systems Development with No Program Specifications Gives

浅田昌紀

要 約 LINCによる大規模システム開発の一方法として，ソフトウェア開発サイクルのうち，詳細設計段階のプログラム設計を省略してシステム開発を実施した．

この開発は，開発期間・工数・費用等の顧客側の制約条件により，リスクを覚悟の上で行ったものであるが，次の効果があった．

1) COBOLによる一般的な開発方法に比べ，3～4倍の生産性が得られた．

2) エンドユーザから見て，余分なプログラム仕様書等の資料を省略できた．

無事システム開発を完了し，納入することができたが反省点もある．概要設計書と，開発者の良否が直接プログラムの品質に反映される事実は今後の課題である．

今回の開発方法はLINCを利用することにより，どのような仕様書があればよいのか，どのような開発体制があればよいのかという問に対する一材料を提供する．

Abstract To establish one way to develop a large-scale application system by LINC, we have implemented a systems development without writing any program specifications usually required in the detailed design phase of a software development cycle.

With all anticipated risks, this way had to be adopted because of the customer's restricted requirements such as the limited period of development, available manpower and size of costs involved. As a result, the following effects have been found:

1) Three to four times higher productivity than conventional COBOL-based development methods

2) Successful deletion of extra documents like program specifications which have nothing to do with end-users

There was not any specific trouble in our efforts to develop the system and install it at the customer, but some points to review we also detected.

The fact still remains to be considered that both the quality of a conceptual design and the level of developers' skill are directly reflected in the quality level of the program product.

The author's experience this time may provide a hint for the questions as to what specifications and development resources are required when LINC is used as a systems development tool.

1. はじめに

1986年から87年にかけて，4GLの一つであるLINCを使用し，プログラム仕様書をつくらずにシステム開発を行った．

このような開発を行った経緯は，開発期間・工数・費用等の制約条件が顧客側にあり，リスクを覚悟の上で最善の方法と判断されたためである．

すなわち，顧客において新商品発売にともなうシステムの開発が急遽必要になった

が，将来の拡張性を考えた時パッケージソフトウェア等は見送られ，自主開発の方針がとられた．ところが，われわれにとっても顧客にとっても，この開発のための十分な業務知識がなく，最終的にノウハウを持つ第三者の業務基本設計書を購入し，詳細設計以降の作業を当社が請け負う，という状況となった．

これは，業務基本設計書(概要設計書)を起点にして，LINC を使用することにより，COBOL 等の 3 GL による従来の開発工程と比べ，どこまで開発工程を軽減できるかと言うテーマに対する一つの回答になると考える．

2. LINC による一開発形態の報告

2.1 開発システムの概要と背景

86 年当時，生命保険業界では情報系システムの整備・拡充が必要となってきており，新商品発売にともなう資産運用実績の公開の必要から，資産運用トータル管理システムの開発が急務となっていた．

中堅生保である T 生命も，大手生命保険会社に遅れをとることなく新商品発売に向けてシステムの開発を検討していた．

しかし T 生命では，開発にかかれるシステム要員が限られており，かつエンドユーザ部門にもシステム開発のための専門要員が出せない，という事情があった．さらにパッケージソフトウェアでは将来の拡張性や，自社向けの機能変更に不安があった．

そこで検討されたのが業務ノウハウを持つ大手証券会社系のソフトウェア会社が販売していた業務基本設計書(概要設計書)の購入である．これは数社の開発実績を持つものであった．この設計書を基に開発しても期間が足りず，必要最小要件として，国内株式・国内公社債・短期金融・またこれらをとりまとめるファンド管理までを，第一次システムとして開発することで，実現可能かどうかが検討された．

この概要設計書で，COBOL による従来工法でシステム構築を行うとすると，詳細設計，プログラミング等々といった工程が必要である．販売元であるソフトウェア会社による見積りでは，COBOL で 20～30 万ステップ，開発に要する工数としては 150～200 人月であった．これでは，期間・費用の面から極めて実現が困難と予想された．

一方，LINC を利用することにより，概要設計書から即プログラミングが可能であるとすると，このシステム開発が新商品発売までに実現できる可能性があった(図 1)．

そこで，万一の場合の開発優先順等をエンドユーザに了解の上，LINC による開発を前提に以下の検討に入った．この時点で残された期間は，すでに 6 か月になっていた．

2.2 開発方法の検討

2.2.1 概要設計書の位置づけ

最初に，概要設計・詳細設計という用語についての考え方を明らかにしておく．

本稿では，当社のソフトウェア開発方法論である「SIM」(Software Implementation Methodology)に準拠して記述している．

ここで，必要な資料を SIM ハンドブックより引用しておく．

1) ソフトウェア導入のライフサイクル……SIM では，ソフトウェア導入のライフサイクルを図 2 のようにとらえ，各段階では「利用者の要求どおりであるか否か

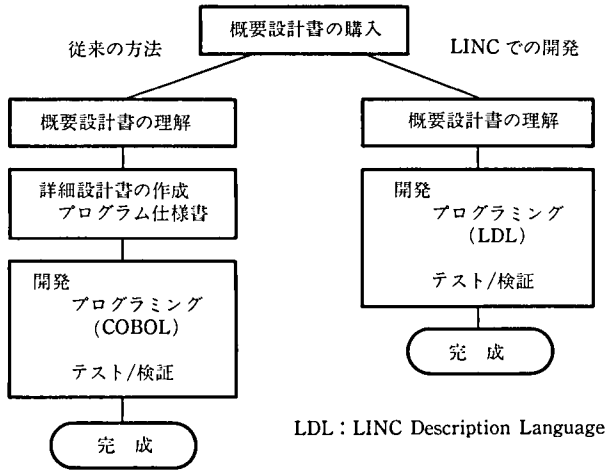


図 1　開発方法の比較

Fig. 1　Comparison between two development methods

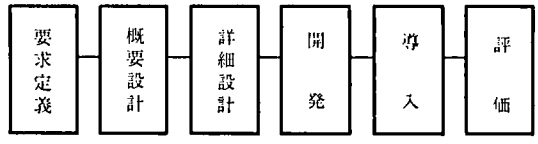


図 2　ソフトウェア・ライフサイクルの段階[1]

Fig. 2　Phases of software implementation life cycle

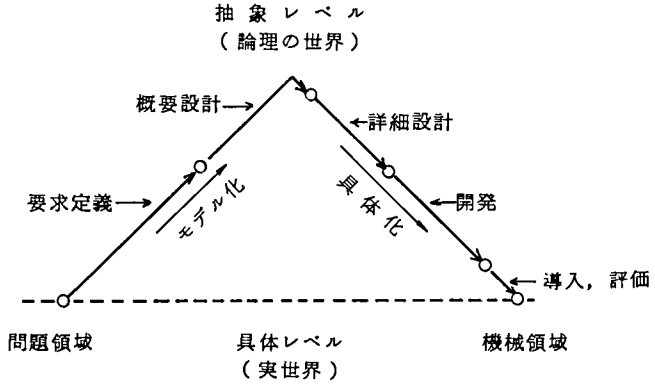


図 3　ソフトウェア・ライフサイクルの概念図[1]

Fig. 3　The concept of software implementation life cycle

の評価を行い，利用者の承認を得た上で次の段階に進むこととしている．

また各段階の概念図は図 3 で表され，「ソフトウェア・ライフサイクルは，問題領域を抽象化していくモデル化と，モデルを機械にマッピングしていく具体化の二つの大きな過程を段階化したものである」としている．

したがって，モデル化の成果物が概要設計書であり，具体化の成果物が導入されたコンピュータシステムである，という考え方である．

2)　概要設計……作業目標は，「要求定義で明らかにされた要求と制約に基づき，利

用者から見た機能の完全な記述と，データ処理面から見た機構上の概要の記述を行う」である．各作業の概要と流れは図4で示される．

3) 詳細設計……作業目標は,「概要設計で明らかにされた解決策としての構能および処理フローをより詳細にし，プログラミング可能な仕様書とする」各作業概要と流れは図5に示される．

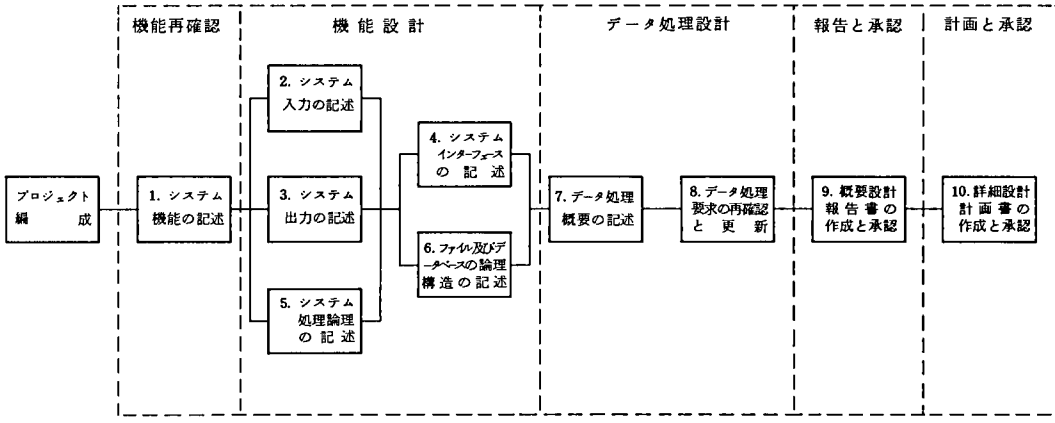


図 4 概要設計の作業概要[1]

Fig. 4 Task charts of conceptual design phase

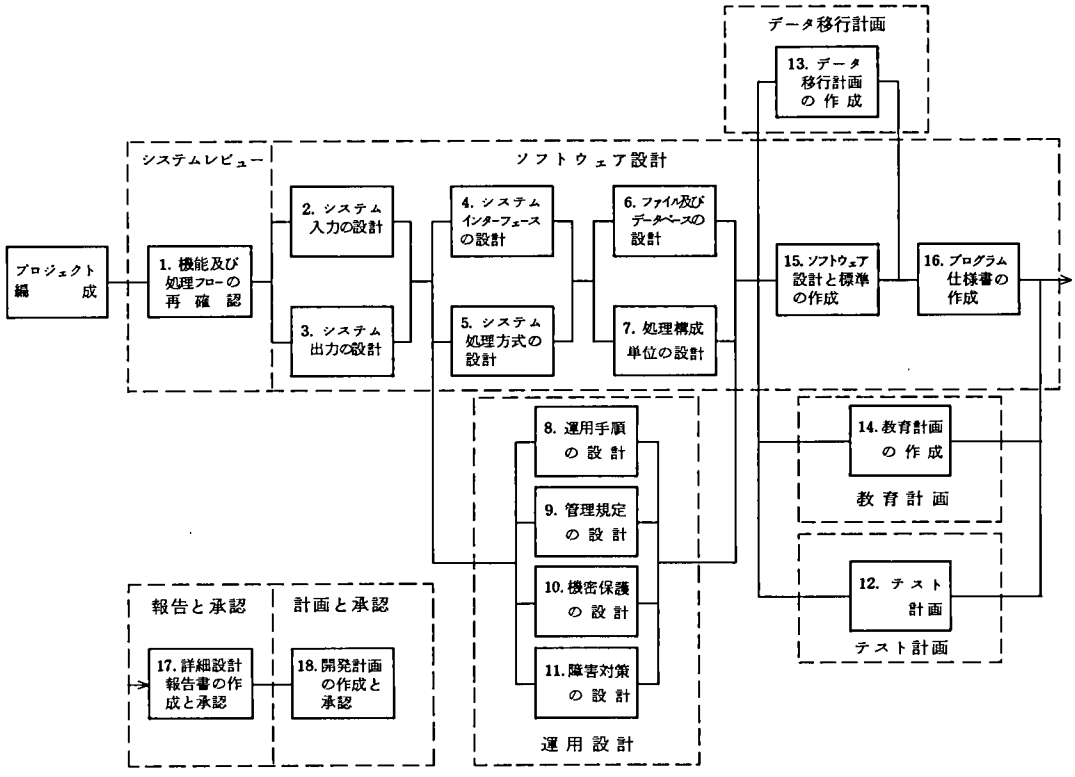


図 5 詳細設計の作業概要[1]

Fig. 5 Task charts of detail design phase

2.2.2 使用した業務基本設計書(概要設計書)

前述の立場に基づき，今回使用した業務基本設計書を簡単に紹介・評価する．

1) サブシステムの構成

① ファンド編
② 国内株式編
③ 公社債編
④ 短期金融編
⑤ 時価・レート編

2) 各サブシステムの記述の構成

各サブシステムは，次のような項目で構成されている．

① システムの目的
② 基本的な考え方
③ システムの概要
④ 業務処理の内容(データエントリ，データベース，レポーティング，その他)
⑤ プログラムの構成
⑥ 入出力レイアウト
⑦ その他

図6(a，b)は使用した設計書の記述程度を示すサンプルである．これ以外にも概念的な説明や，概要の記述，画面や帳票の一覧表が含まれるがここでは割愛する．

3) 業務基本設計書の内容と評価……SIMの基準から比べ，不足と思われる点は，とくに認められなかった．一言でいうと，使用する機種や言語に依存する記述は一切ないが，業務上の記述は十分になされていた．逆にLINCでは不要あるいは意味のない資料もあった．

一部説明不足，曖昧さ，不正確な点もあったが，LINCを使用した開発のもとでは以下の観点で使用可能との判断を下した．

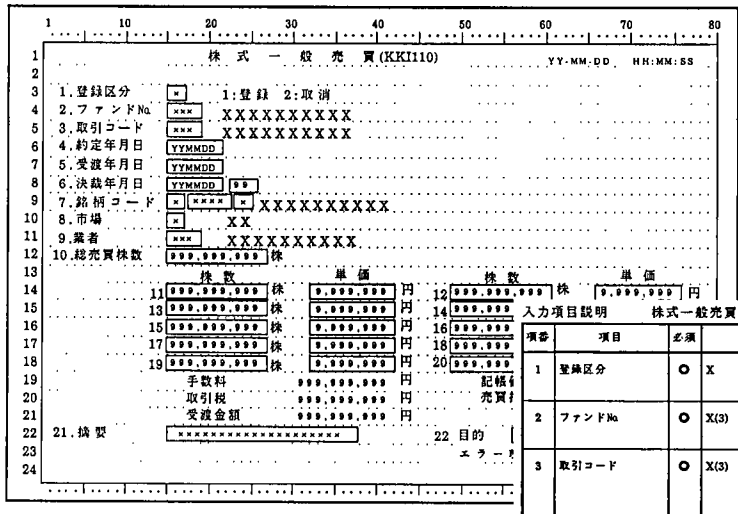
① 記述が標準化されている．
② 入力画面，入力項目の記述は正確かつ理解し得る．
③ 出力帳表，出力項目の記述は正確かつ理解し得る．
④ データベースの設計は，LINCを使う上においても妥当である．
⑤ エンドユーザが内容を理解でき，その範囲で誤りはない．
⑥ その他一見してわかる致命的な矛盾，誤りがない．

2.3 開発の実際

2.3.1 詳細設計工程

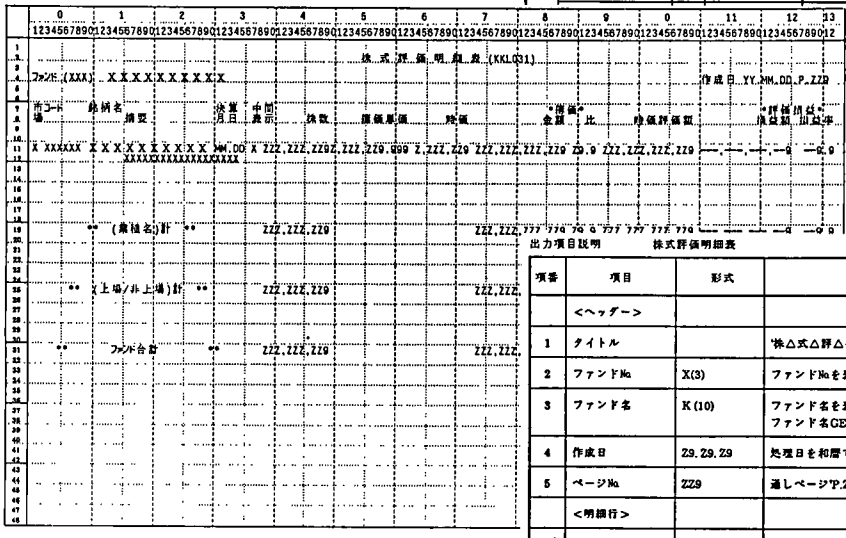
以上の判断により，概要設計は完成したものとみなした．

詳細設計の作業項目は大きく分類すると，ソフトウェア設計，運用設計，データ移行，教育計画，テスト計画，である．本稿ではソフトウェアの開発が成り立つかどうか，生産性はどうなのかということが主題であるため，このうち必要と思われる「ソフトウェア設計」，「運用設計」，「テスト計画」の関連部分について，SIMの詳細設計における作業項目と合わせて解説する．



入力項目説明　　株式一般売買・その他売買

項番	項目	必須	形式	内容
1	登録区分	○	X	1：登録 2：取消
2	ファンドNo.	○	X(3)	コード表参照 コード表よりファンド名(漢字)を表示
3	取引コード	○	X(3)	売買区分＋取引の種別を表す (コード表別紙参照) コード表より取引名(漢字)を表示
4	約定年月日	○	9(6)	YYMMDDで入力
5	受渡年月日	○	9(6)	YYMMDDで入力
6	決裁年月日	○	9(6)	案件決裁日をYYMMDDで入力



出力項目説明　　株式評価明細表

項番	項目	形式	内容
	<ヘッダー>		
1	タイトル		'株△式△評△価△明△細△表△(KKL031)'を表示
2	ファンドNo.	X(3)	ファンドNo.を表示
3	ファンド名	K(10)	ファンド名を表示 ファンド名GETルーチンを使用してGET
4	作成日	Z9.Z9.Z9	処理日を和暦で表示
5	ページNo.	ZZ9	通しページ'P.ZZ9'
	<明細行>		
6*	市場	X	評価用の時価を採用した市場コード
7	銘柄コード	X(6)	銘柄コードと新旧区分
8	銘柄名	K(10)	銘柄名と新旧表示
9	決算月日	Z9.Z9	決算月・日
10	摘要	X(20)	銘柄属性DB上の摘要を表示
11	中間表示	X(1)	年2回決算会社の場合'*'表示
12	株数	ZZZ,ZZZ,ZZ9	当日現在の保有残株数 単位:株
13	簿価単価	Z,ZZZ,ZZ9.99	当日現在の簿価単価 単位:円

日(配当金の起算日)

ド＋新旧区分
漢字)を表示

売却株数(単位:1株)

図 6(a)　概要仕様書の抜粋

Fig. 6　Examples of conceptual design document

(1) 新旧合併処理

既計上取引残高DBを先頭レコードより順次リードしていき、直近計上日レコードとなった時、銘柄属性レコードの次回決算日を参照。

次回決算日の翌営業日＝処理日なら、以下の処理を行なう。

満足しない時は、当処理はスキップして、次の処理チェック下同じ。)

|レコード(注)の項目更新

日　＝　処理日をセット

1新株保有残株数　＝　ゼロをセット

2新株保有残株数　＝　ゼロをセット

|残高DBの更新

計上日のレコード;
†上日(DBのKEY項目)≦処理日で、　計上日,受渡日,管理
ı(DBのKEY項目)が最も大きいレコード…(以下同じ。)

2.5.2　株式取引残高DBの更新

(概要)

株式取引残高DB(既計上および未計上DB)を先頭から、1レコードづつシーケンシャルに入力し、取引レコード単位で下記の処理を行なう。

① 新旧合併、既計上DBの更新

… 銘柄属性DBの次回決算日を参照し、処理日＝次回決算日の翌営業日のとき、既計上DBよりゲットした直近計上日の保有残高に対して、新旧合併処理をする。

② 未計上DBから既計上DBへの移し換え

… 当日計上データを既計上DBに移す。

③ 権利取得データの自動作成・既計上DBへの追加

… 直近計上日の保有残高より、銘柄属性DBの権利情報を参照して、

・新株割当(権利)

・無償増資

・未収配当金 … 取得データを作成・追加する。

④ 当日保有残高の時価評価・既計上DBの更新

… 直近計上日の保有残高について時価評価する。

⑤ 月末保有残高レコードの作成、既計上DBへの追加

… 処理日＝月末営業日のとき、直近計上日の保有

ファイル項目説明　　株式取引残高ＤＢ

項番	項目	形式	内容
17	約定日	P(8)	入力項目 ・購入・売却データのとき　約定日 ・権利取得データのとき　権利取得日 (上記以外のときは、ゼロをセットする)
18	資本稼動日	P(8)	入力項目
19	株数	P(10)	入力項目 (注)未収配当金、受渡配当金のときは、セットしない……ゼロとする
20	単価	P(6V2)	入力項目 入力なし時は、約定金額÷株数 (注)未収配当金、受取配当金のときで、単価・特別配の入力がある場合は、　普通配＋特別配　とし1株配合計単価をセットする
21	約定金額	P(12)	株数×単価 単価入力なし時は 売却のとき、 受渡金額＋手数料＋取引税 買付のとき 受渡金額－手数料 (注)未収配当金、受取配当金のときは税込配当金額をセット
22	手数料	P(10)	入力項目

図 6(b)　概要仕様書の抜粋

Fig. 6　Examples of conceptual design document

以下作業項目の説明はSIMの引用である.

[ソフトウェア設計に関して]

1) システム入力の設計

作業目標：概要設計で記述したすべてのシステム入力について，その詳細を完全に記述する.

2) システム出力の設計

作業目標：概要設計で記述したすべてのシステム出力について，その詳細を完全に記述する.

この二つの作業は不要であった.

概要設計書の入出力に関する記述の精度が高かったこともあるが，LINCの開発では，画面や帳表のレイアウトは，根本的に誤りでない限り容易に変更可能である．また入力チェックの仕様等も多少の変更は，エンドユーザに操作してもらった上で確定した方が確実である．業務上，何を入力し，何を出力するかということがはっきりしていれば十分である.

3) システムインタフェースの設計

作業目標：概要設計で記述したすべてのインタフェースについて，その詳細を完全に記述する.

業務に固有な共通サブルーチンを使用する上でのパラメータを設計した.

COBOLの開発の場合，他にもオンライン・プログラムを画面単位に分割し，親プログラムや画面間の技術的なインタフェース等の設計が必要になったりする.

LINCの場合は，画面あるいはデータベースをISPEC(Interface SPEC)という単位で記述していくが，これらのとりまとめや技術的なインタフェースは，LINCコンパイラが自動的に行う.

4) システム処理方式の設計

作業目標：概要設計において記述した機能構成およびルールを実現するための処理方式を設計する，と共に設計指針の作成を行う.

ここでは，以下の作業を実施した.

① ISPECまたはREPORT(帳表またはバッチ更新プログラムのスペック)の割当基準の検討

② 画面操作の標準化とメニュ画面の仕様作成

③ 共通性の高いコードテーブルの仕様作成

④ 帳表作成パターンの指示

⑤ カレンダ管理および日数計算の仕組みの設計

5) ファイルおよびデータベースの設計

作業目標：データベースおよびファイルの最適な物理構造の設計とその説明文書の作成.

LINCは原則的にデータベースを使用する．また物理属性はコンパイラが最適化して決定する．ここでは，ファイルサイズの決定およびプロファイル(データベースのアクセスキー)の設計を行った．また通常のファイルレイアウトも設計書に含まれてはいたが，LINCでは意味がないため，参考程度にしか使用しなかった.

6) 処理構成単位の設計

作業目標：処理構成単位を個々のプログラムにまとめ，処理効率目標の確認を行う．

ISPECおよびREPORTの一覧表の作成を行った．

LINCの場合，画面プログラムの構成単位はISPECであり，帳票プログラムの構成単位はREPORTとなり，極めて明快である．ただしバッチ更新については，どのように設計書の機能をREPORTにおいて分担するのかを検討する必要があった．これは業務基本設計書の立場上，どのような構造のプログラムをつくるかということに関し言及していなかったためである．

7) ソフトウェア設計と標準の作成

作業目標：プログラムを最低要素まで分解し，その構造を設計すると共に標準化を行う．

とくに作成しなかった．ISPECまたはREPORTが最低要素とみなし，それ以下はプログラマに任せた．

8) プログラム仕様書の作成

作業目標：プログラム作成およびテストに必要なすべての情報を提供する仕様書を作成する．

新たには一切作成しなかった．

設計書をコピーし，ISPECまたはREPORTの担当者に必要箇所をまとめて配布することで済ませた．この作業は通常，詳細設計工数の大半を占める作業である．この作業を行わずにプログラムが作成でき，かつ導入後の保守ができるのか，というのが重要なポイントである．

ただし，以下の理由によりかなりの確信を持つに至った．

プログラム作成上，LINCには次の機能がある．

① プログラムはISPECにより自動的に構造化される．

② 通常高度なプログラミング知識を必要とする，メイン処理ロジック，データベースの排他制御，リカバリ等のコーディングは不要(LINCコンパイラが自動生成する)

③ 画面や帳表はスクリーンペイント方式で可視的に作成できる．

④ 処理ロジックは入力時リアルタイムに構文チェックが行われ，コーディングミスは論理的なものに限定される．

⑤ 辞書機能により，データ名，属性，日本語名称を統一管理できる．

⑥ LINCコンパイラが辞書とデータベース，画面を関連し管理する．

⑦ LINCコンパイラのデータベースには，種々の設計情報が内蔵されており，ドキュメント機能により最新情報をいつでも作成または照会できる．

⑧ ロジックのリストは構造化し出力される．

さらに今回使用した設計書の特徴として，業務上の機能が十分記述されていたことにより，ISPECまたはREPORTへの対応がつければそれをプログラミングすることはさしてむずかしいことではないと考えた．

[運用設計に関して]

9) 機密保護規定の設計

作業目標：機密保護規定がシステム設計に正しく含まれているか確認する．

これをアプリケーション的に対処するとなると，ソフトウェア設計の範ちゅうであるが，システムソフトウェアの機能で十分対処できることから，とくに検討を行わなかった．

10) 障害対策の決定

作業目標：システム設計に適切なバックアップおよびリカバリ機能が含まれているかを確認する．

LINCではリカバリ機能が自動的に備わっている．ことさら検討することをしなかった．

[テスト計画に関して]

11) テスト計画の作成

作業目標：各レベルのテストプランの作成

- 単体テスト
- サブシステムテスト
- システムテスト
- 検収テスト

基本的に次の方針をとった．

① 単体テストは，顧客より提供されたテストデータをベースに，プログラマが概要仕様書に記述されている各要件をクリアすることとする．

② サブシステムテストは，同じく顧客よりデータをベースにサブシステム責任者の裁量で行う．

③ システムテストは，それらのデータをもとにスルーランする．

④ 検収テストは，顧客および概要設計書作成者により実施する．

後にいくつかの反省点を生ずるが，LINCならば多少のトラブルでも即応できる，という確信もあり，これ以上の計画は立てなかった．

以上，作業項目としてはある程度少しずつ実施したが，プログラム仕様書の作成を行わないため，作業工数を大幅に省略することができた．

基本設計書からプログラム開発までに作成した主たる成果物をまとめると，以下のとおりである．

① ISPEC/REPORT一覧表

② 設計書にとくに記述がないが必要となった仕様

コードファイル・データベース

メニュ画面

③ 共通サブルーチンの仕様

2.3.2 開発グループの組織

開発組織について簡単にふれておく．プログラミング・グループは都合により，図7のように3グループによって行われた．

担当プログラミングは，グループごとのプログラミングスキル，LINC経験の有無，作業場所(コミュニケーションが容易か否か)によって決定した．

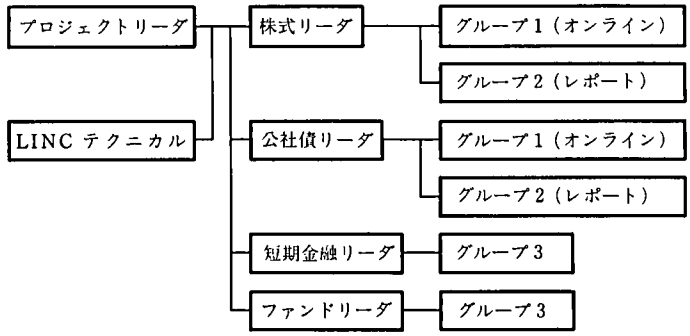


図 7　開発組織図

Fig. 7　The team of the development

各グループの特徴を記しておく．

1)　グループ1……当社のプログラム開発部門より4～8名(途中交代あり)．ただし入社年度は1～3年で，LINCの経験は全員なし．社内のためコミュニケーションが容易であることから，4サブシステムのうち2システムのオンライン部分を担当した．

2)　グループ2……外部ソフトウェア会社である．多少のLINC経験を持ち，M/Cリソースはあるものの，過密状態で開発には難があった．グループ1の担当したザブシステムの帳表およびバッチ更新部分を担当した．

3)　グループ3……外部ソフトウェア会社である．LINCの十分な経験と，強力なM/Cリソースを持つ．残りの2サブシステムを担当した．(このサブシステムは，納期どおり完成しなければならなかった)

外部ソフトウェア会社は，いずれも窓口となるSEのほかに4～10名程度(交代あり)のプログラマにより構成された．

2.3.3　開発の結果

テスト工程で予想以上の苦労をしたが，結果的に納期を守ることができ，所定の機能を提供することができた．

約60画面のオンラインプログラム，および約80帳表，約50本のバッチ更新プログラムを，約半年の間に50人月の工数で完成した．概要設計者による見積り(他社機開発事例に基づく)では，COBOL換算20～30万ステップであり約150～200人月であったことを考えると，約3～4倍の生産性を達成できたことになる．

2.4　問題点と今後の課題

今回の開発はほぼ完了し，短期間に所定の機能を提供することができた．しかし若干の問題も発生した．その中より今回の開発の特殊性によると思われるものを二つ紹介する．

1)　テスト工数の見積り……テストはLINCにおいても重要な意味を持つ．構文上のエラーはリアルタイムにチェックできていても，業務上の誤りまではチェックできない．

また次の理由により，テスト工程のむずかしさがある．

表1 開発実績
Table 1 The results of the development

	LDL ステップ			生成（個数）			
	ISPC	REPT	合計	画面	ファイル	帳表	更新
国内株式	19	28	29,084	14	6	21	7
公社債	21	43	38,726	14	8	30	13
短期金融	6	11	4,916	6	1	6	5
ファンド	10	31	12,367	9	5	10	21
その他	21	13	8,556	19	9	11	2
合 計	77	126	93,649	62	29	78	48

総開発工数 50 人月　　'87.10 末現在

① 生産性が高いため，一定期間に作り出される機能が多い．このため，従来全工程の 1～2 割程度テスト期間が必要なシステムの場合，3～4 割程度のテスト期間が必要となる(ここでいうテストとは，手直しおよび再テストを含む)．

言い換えると，プログラム作成工程が短縮されるが，テスト工程まで短縮される訳ではない．

② 当時 LINC ではプロトタイピング工法を強くセールスしていた．これはある程度の仕様の変更を認める開発とも言える．今回概要設計レベルで大きな変更は発生しなかったが，小さな変更要望は積極的に認めた．

この結果単体テストで行った細部のテストが無駄になることもあった．

2) プログラムのバグ……今回の開発では，単体テストの仕上がり程度がグループによりかなりばらつきが出るという事態が発生した．

数の上ではプログラマの基本的スキルに依存するものが多かった．また，LINC の経験の有無は，LINC を効果的に使用しているかという点では差があったが，バグの発生にはさほど関連がなかった．むしろいかに業務を理解していたかに関連していたように思われる．この意味で，業務を含めたプログラムのスキルを考慮し，開発組織を構成することが必要だったと考えている．

単体テスト終了(申告ベース)後発生したバグを分類すると次のようになった．

① 単なる不注意
② LDL 構文の誤解
③ プログラマの基本的なスキルの不足
④ 一般常識の欠如
⑤ 業務知識の不足
⑥ 概要仕様書の不備

この中から，プログラム設計を省略することにより発生したと思われる，特徴的なものをいくつか解説する．

もちろん，プログラムも設計者も完璧ということは有り得ないので，非難するつもりではなく，通常であれば詳細設計時に問題点を発見し吸収されるはずが，表面化してしまったケースと言える．

a) プログラマの基本的なスキル不足：演算のワークエリアの定義がいい加

減なため，減算後符号をロストしたり，乗算でオーバーフローを起こすケース．

b) 一般常識の欠如：概要仕様書上「当月発生した～のレコードを選択し～する．」とある場合に，前後関係では当然「年月は当月か」ときいてほしいところが「月は当月か」というコーディングをするケース．この種のバグは，プログラムリストを見るか，翌年になるまで発見されない可能性がある．

c) 業務知識の不足：データベースに「移動額面」と「額面残高」という意味の項目があったとする．データベースの定義(名称)では，前者が「額面」後者が「残額面」となっていたとする．

一方二つの帳表「売買明細表」と「保有残一覧表」があり，ともに「額面を表示する」と指示してあったとする．

プログラマは何の疑いもなく，どちらも「額面」を表示してしまう．ところが設計者は保有残一覧表の場合，「残額面」に決まっていると考える．

d) 設計書の問題

概要計書の処理内容で次のような記述があったとする．

i) ～のとき～する．

ii) ～のとき～する．

⎫

v) ～のとき～する．

今回プログラマはこのとおり素直にコーディングした．

ところが，よくこの条件を理解すると，i)～v)は必要な条件を網羅的に記述しているが処理順については考慮しておらず，i)とii)の条件の記述はv)の後でないと正しくは機能しないというケースがあった．

このケースは，設計者も責められない．今回このような開発を行うとは考えてもいなかったはずである．

3. おわりに

LINCを利用することにより，詳細設計段階のプログラム設計等を，ほとんど省略することができた．この結果従来工法に比べ，かなりの生産性向上が得られた．またエンドユーザにとって不要かつ膨大な設計資料を作成しなくてすむ，という点において意義があった．

ただし，概要設計および開発者の良否が，直接プログラムの品質に反映される危険性がある点等，注意が必要である．

このシステムはその後，同様にして，第二次開発として外国株式，外貨債・外貨預金・投資信託サブシステム等を加え，さらに諸制度変更等の機能変更や強化を行い今日に至っている．LDLの規模としては約24万ステップになっている．当初の約2倍以上になっており，単純にCOBOL換算すると約50～70万ステップの規模となっている．

このことは本稿で述べた開発方法において，開発のみならず保守も可能であることを実証していると考える．

参考文献 ［1］ SIM(Software Implementation Methodology)ハンドブック，日本ユニシス(株)，1987.3.

執筆者紹介 浅 田 昌 紀(Masaki Asada)

昭和51年早稲田大学理工学部数学科卒業．55年日本ユニシス(株)入社．流通業，倉庫業を経て，生命保険業のSEサービスに従事．現在日本ユニシス(株)金融システム第一本部金融システム三部所属．

葛飾区役所における住民記録システムの開発事例

The Development of a Resident Record System at the Katsushika Ward Office

森　山　　勉

要　約　地方自治体における事務処理は，外部計算センタ委託から汎用コンピュータの自己導入に変化しつつある．その中で住民に対するプライバシ保護が叫ばれている．葛飾区役所は，自己導入に当たって行政史上初めてともいえる「個人情報保護条例」を制定した．プライバシ保護優先と専門家組織（電算室組織）を持たず，エンドユーザによる開発・運用を柱とした住民記録システムが，15か月に渡り約200人月の開発を終え，1987年10月1日に無事本稼働した．LINCシステムとしては大規模なものとなった．

LINCの大規模システム開発事例紹介として，この住民記録システムが持つ業務機能とソフトウェア開発方法論としての「SIM」[2]適用と，第4世代言語「LINC」適用についての概要を記述する．

abstract　Amid the transition of data processing tasks for local self-governing bodies from contracted computing by external data processing centers to their own installation of general-purpose computers, there have been cries arising for the protection of residents' privacy information. On the occasion of its own installation of computer systems, the Katsushika ward office has enacted 'the ordinance for individual information protection' which is unprecedented in the Japanese history of administration.

With greater care given to the protection of privacy information and with no computer specialists organized into a team (with no dedicated computing function established), the ward offiice's resident record system developed and designed for operation exclusively by end-users started its successful operation on October 1, 1987 after the development had cost some 200 man-months over a period of fiteen months. The sysem is one of the largest LINC applications even produced.

In an attempt to show a sample of large-scale systems development, this report describes the operational functions provided by the resident record system, and also summerizes the application of the 'SIM' software development methodology and that of the 'LINC' fourth-generation language.

1. はじめに

本稿で紹介するシステム開発事例は，地方自治体における住民記録システムである．このシステム開発の特徴は，以下のとおりである．

1) 住民のプライバシ情報を取り扱うことにより，セキュリティ対策が必須であった．

2) 開発すべき最終成果物の納期，開発期間から見て極めて厳しい状況が予想された．

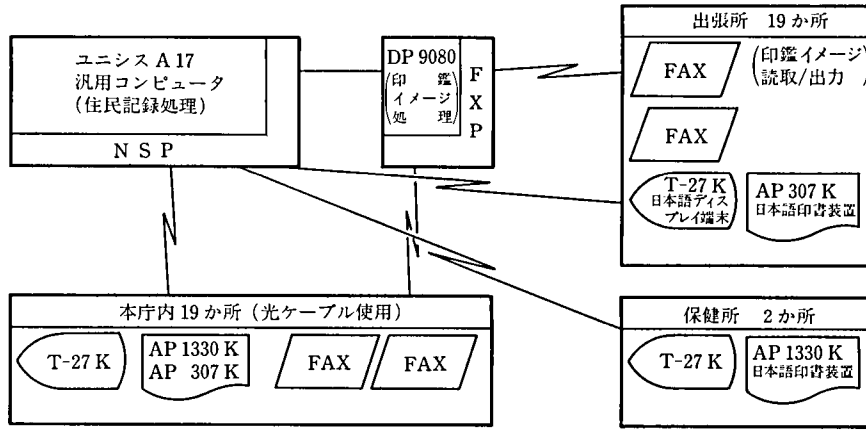
3) 本システム開発は，エンドユーザである主管課(実際に業務を司る部署)，すなわち，システムの適用を受ける者とシステム開発担当者（日本ユニシス）の共同開発である．

今回のシステム開発は，上記3項目に留意し，当社のソフトウェア開発方法論「SIM[2]」および第4世代言語「LINC」を適用することにより，納期どおりに1987年10月1日に本稼働した．

2. 葛飾区役所・住民記録システム概要

2.1 住民情報系における住民記録システム

葛飾区役所では，「個人情報保護条例」を遵守した住民サービス・行政事務の効率向上を目的とした，住民情報系システムの実現が必要となった．今回の住民記録システムは，地方自治体の基幹システムであり，住民税・国民健康保険・選挙管理・福祉事業等の自治体業務と密接な関連を持った住民情報（プライバシ）を取り扱う業務である．また，住民情報の中にも印鑑登録業務におけるイメージ処理を扱うために，印鑑の印影を高精度に処理するサブシステムの導入も必要とした（図1）．



NSP: Network Support Processor
FXP: Facsimile Exchanger Processor

図1 葛飾区役所システム構成図

Fig. 1 The system configuration of Katsushika ward public office

本稿では，住民記録システムとして住民基本台帳業務・印鑑登録業務・住記照会業務の開発について記述する．

2.2 住民記録システムの業務機能

本システムは，住民の異動(転入・転居・転出等)から住民票発行および印鑑登録・印鑑証明書発行を，出張所19か所を含む20か所のどこからでも可能にしたシステムであり，住民のプライバシ保護を優先とした行政史上初めてのシステムである．

次に，住民記録システムを構成する住民基本台帳業務，印鑑登録業務，住記照会業務の各々についてその業務内容を紹介する．

2.2.1 住民基本台帳業務

1) 異動処理……住民の転入・転居・転出等の異動記録となる住民台帳を一元管理する．
2) 証明書発行……住民票・記載事項証明等の公的証明書発行をする．
3) 通知処理……自治体間での転入・転出・出生等の通知を処理する．

4)　異動連絡……住民異動処理結果を他業務に，承認範囲内で連絡票を送付する．

5)　統計……人口動態・事務月報等の統計処理の効率・充実を計る．

6)　その他……住民情報照会のサービス業務

2.2.2 印鑑登録業務

1)　印鑑登録……登録時の印影イメージ処理をDP 9080（ミニコンピュータ）にて，データ圧縮も含め処理する．文字情報はホストにて管理する．

2)　印鑑証明書……ホストの文字情報とDP 9080の圧縮された印影イメージを証明書として発行する．

3)　その他……印鑑の廃止・亡失および住民異動に伴う履歴管理をする．

2.2.3 住記照会業務

住民記録システム以外の他業務に必要とされる住民情報の照会システム．プライバシ条例を遵守するために住民記録から分割した独立のシステムである．

2.3 システム機能要件

個人情報保護の遵守，システムの安定稼働，大量トランザクション処理の実現のため，本システムは以下の機能を有している．

2.3.1 機密保護機能

機密保護を実現するために以下の機能を利用した．すなわち，COMS*の提供するユーザコード/ステーション/ウィンドゥ定義，LINCが提供するオペレータコードの権限/ステーションの権限およびHUB**の機能である（図2）．

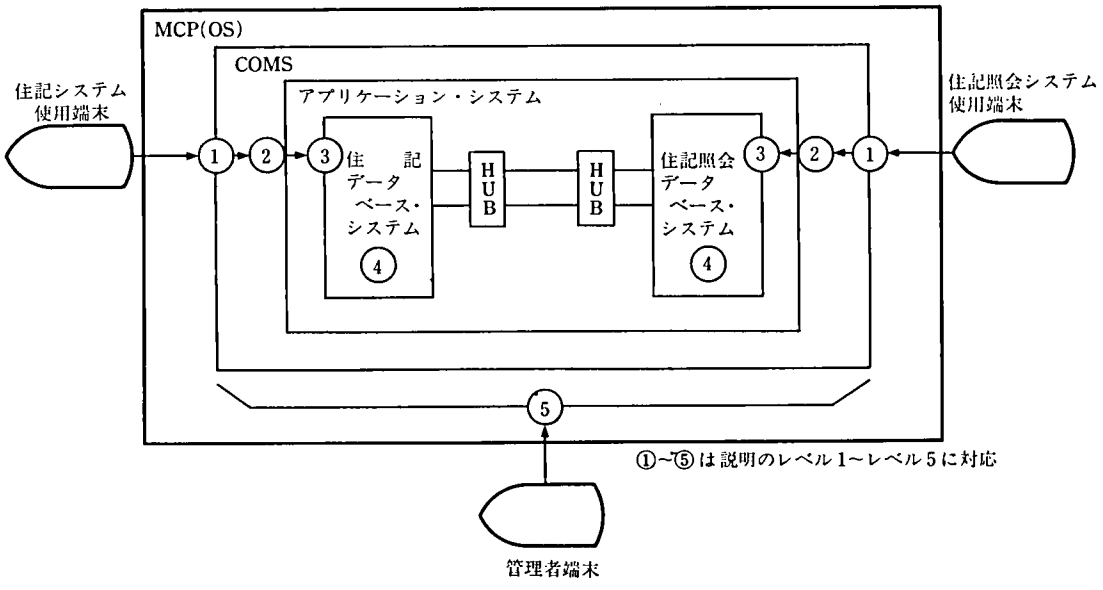


図 2　葛飾区役所・機密保護システム概要図

Fig. 2　An outline of security systems of Katsushika ward public office

* COMS (Communication Management System)：オンライン・トランザクションを行先経路別にコントロールおよびオンライン・システム環境を設定するシステム・ソフトウェア．

** HUB：LINCが提供するプログラムで，複数LINCシステム間でのコミュニケーションを提供する．

図2の機密保護の説明は，以下のとおりである．

① レベル1……承認されたユーザコード，パスワードをチェックする．

② レベル2……機密保護対象システムへのチェックをする．

③ レベル3……ユーザコード，パスワード，端末名の機密関連チェックをする．

④ レベル4……許可された使用者のアプリケーション・システムによるチェックをする．

⑤ レベル5……システム全般にわたる機密保護管理を行う．

2.3.2 システムの安全性（障害対策）機能

本システムは，ホットスタンバイシステムではなく，CPUの二重化/ディスク(ミラーディスク)/代替NSPにより，システムの安全性，データベースの保護を実現した．また，ディスク障害が発生してもミラーディスク(OSによるディスクの二重化)の採用により，使用者へ影響を及ぼさない機能を実現し，ディスク以外の障害についても30分以内の回復を目指した．

2.3.3 大量トランザクション処理機能と他社インタフェース

1日（8時間）の入力トランザクション2万件を効率よく処理し，印鑑イメージ処理システム（他社機）とのインタフェースを実現するために，非LINCシステム間コミュニケーション(LINC 12 USER)を採用した．このLINC 12 USERを使用した他社機とのインタフェースおよびリアルタイム出力帳票処理においては，トランザクションのトラフィック量が高いSMCS*の採用により効率向上を計った（図3）．

2.4 住民記録システムにおけるLINC拡張機能の採用

住民記録システムは，自治体特有の外字処理と印鑑の印影イメージ処理が必須要件である．LINCの拡張機能（非LINCシステム間コミュニケーション）を活用した外

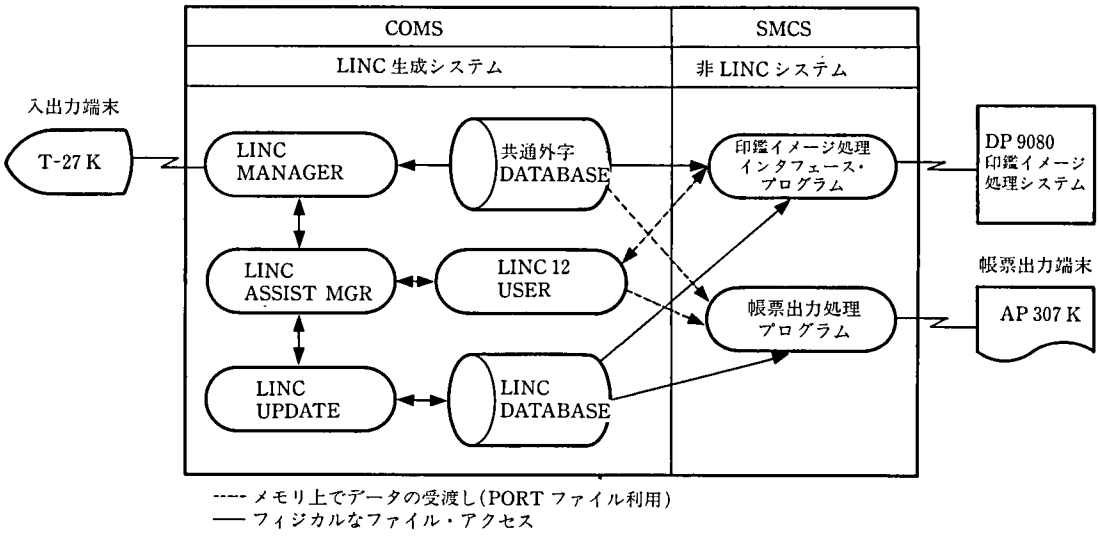


図3 住民記録システム・ソフトウェア構成図

Fig. 3 The software configuration of resident system

* SMCS (Simple MCS)：DIAGNOSTIC MCSを母体として多くのユーザで利用されているメッセージ・コントロール・システム

字処理および他社機(DP 9080 イメージ処理システム)とのインタフェースにより，これを実現した．

非 LINC システム間コミュニケーションにより実現した処理は次の二つである．

1) 発生した外字をデータベース化し，各システムで共有可能とする．外字処理(漢字フォント・ロード）を共有ライブラリとした．

2) 印鑑登録，印鑑証明書等のホスト側（文字情報処理）と DP 9080 側（印影イメージ処理）の同期処理（利用者には，同一システムに思える）を可能とする．

以上述べてきたように，LINC による生成システムと非 LINC システムを効率的にコミュニケーションさせることにより，住民記録システム全体のスループットを向上させることができた．

3. システム開発経緯

3.1 システム開発スケジュール

本システムは 1986 年 7 月システム開発に着手し，1987 年 10 月本稼働を開始した．開発期間は 15 か月であった．担当したプロジェクト・メンバ全員が，業務経験・知識が皆無であることから，作業見積・プロジェクト体制については，当社の地方自治体における LINC ユーザである某自治体のシステムを参考にすることとした．しかし，当該ユーザも印鑑登録業務はシステム化されておらず，人口規模・組織環境においても大きな差があった(人口は 1 万人弱に対して 41 万人強，業務処理は 1 か所に対して，20 か所)．実績ベースにおいては，当初予定の 100 人月を遙かに越える 200 人月以上のシステム規模となり，LINC による開発規模としては，大規模なシステムとなった．

当初の 3 か月は，41 万人分を越えるデータ移行と開発要員の確保が困難（とくに，LINC の経験者不足）であったことから，実際の開発着手が遅れ，開発期間は 12 か月足らずであった．

一方，ユーザの専門家も不在（汎用コンピュータの自己導入が初めて）で，電算組織がないこともあり，エンドユーザとの間で直接要求定義から始めなければならなかった．仕様設計段階では，業務知識は有しているがコンピュータ知識の少ないユーザとコンピュータ知識は有するが，業務に対しては未経験の当社担当者の共同作業となった．仕様確定の遅れ，要員確保，協力会社要員の教育・管理等の解決すべき問題が山積みであった．しかしながら，これらの問題は SIM/LINC によるプロトタイプの実施等により解決を計り，図 4 のスケジュール通り開発作業を終了した．

3.2 プロジェクト体制

今回のシステム開発を実施する上で，図 4 の分類に準じた三つのチーム編成を行った．

1) 第 1 チーム（オンライン・メインプログラム開発)……エンドユーザが直接関与する業務処理モジュールであり，ユーザとの共同開発で進められた．また要求定義から詳細設計段階で，SIM および LINC でのプロトタイプが有効に活用された．

2) 第 2 チーム（非 LINC システム処理：印鑑イメージ処理，外字および端末インタフェース)……印鑑登録業務の印影イメージ処理を実現するため，東芝の DP 9080 とのインタフェース・システムを導入した．またユーザの要求に必要な新

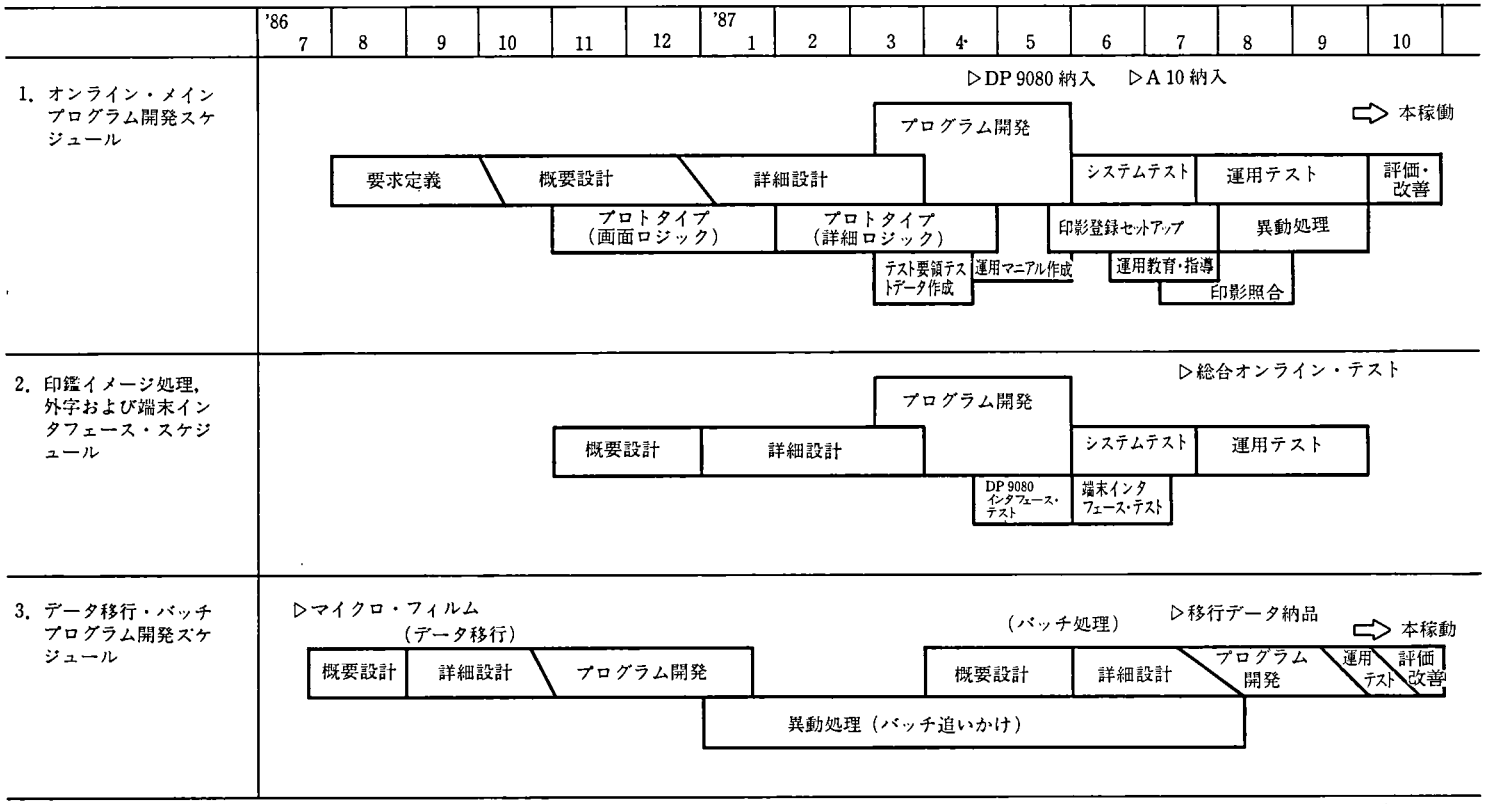


図 4　住民記録システム・開発スケジュール

Fig. 4　Development schedule of resident system

規端末として，24ドット・スクリーンのT-27K，単票の自動挿入・排出を実現したAP 307Kのインタフェース処理とシステム全般にわたる外字処理を担当した．

3) 第3チーム（住民情報（住民基本台帳）のデータ移行とバッチプログラム開発）……約415,000人の台帳をマイクロフィルム経由でデータベース化するとともに，本稼働直前までの住民異動処理および発生した外字作成を担当した．また，日次・月次・年次・適次のバッチプログラム開発を担当した．

以上のプロジェクト体制の中で，第1チームが中心となって開発を進めた．

3.3 システム開発規模

住民記録システムの開発規模をステップ数，作業工数およびLINCのCOBOL換算で表したのが，表1である．

表1 住民記録システム開発規模
Table 1 A scale of the application systems

作業フェーズ	ステップ数	COBOL換算ステップ	工数(人月)	ステップ/人月
オンライン・メイン処理	LINC・LDL 135,000	337,500	98	3,444
印鑑・端末インタフェース	COBOL・ALGOL 47,000	47,000	41	1.146
データ移行バッチ処理	LINC・LDL 5,000	12,500	3	4,167
	COBOL 132,000	132,000	60	2,200
計	319,000	529,000	202	(平均) 2,619

(LINCのCOBOL換算定数は，生成比率の2.5倍を使用した)

この結果を説明すると次の通りである．

1) 開発ステップ数に対する工数については，機能仕様の追加・変更と仕様の曖昧さによる追加作業が含まれている．とくに，オンライン・メインプログラムの開発では，上記要因以上にLINC言語・機能についての開発担当者（協力会社も含めた）の知識不足が多大な工数増加を招いた．

2) 今回のシステム開発におけるLINCの生産性について考察する．表1の要求分析から本稼働までの人月工数によるとCOBOLの生産性は，2,200ステップ/人月となる一方，LINCのCOBOL換算による生産性は3,500ステップ/人月となり，1.5倍以上の効果があった．今回のLINC開発担当者の知識不足を加味すると，実際上の生産性は2.0～2.5倍望めたと考える．今後LINCの開発に当たっては，LINCの機能および採用するバージョン内容とCOBOL・データベース知識を十分に把握することにより生産性の向上が期待できる．

4. プロジェクト管理

今回のシステム開発で発生した問題点・苦慮した点は，次のとおりである．これらは受託システム開発において，一般的に共通する点と今回のユーザ独自の条件もある．

共通事項としては，

1) 開発システムが大規模で複雑である，
2) 開発期間が短い，
3) 他メーカの開発システムとのインタフェース，
4) 開発要員の確保とスキル（とくに業務知識），
5) ユーザと開発担当者間での共通認識不足，

また，今回の独自条件としては，

1) ユーザ担当者の EDP 専門知識が不足であった(電算室組織がないために，エンドユーザとの共同開発)，
2) 自治体特有の外字処理対応が必要であった，
3) 新端末（T-27 K，AP 307 K）が初導入であった，

があげられている．

これらの問題点は，SIM/LINC によるプロトタイプ導入により解決をして，システム開発を完了させた．

とくに，ユーザとの要求定義（分析）・設計においては，約 20 名のエンドユーザとエンドユーザ代表各 7 名の集団と複雑なターゲット・ドキュメント（要求仕様書等）について協議し，すべてを納得してもらうという困難な仕事であった．

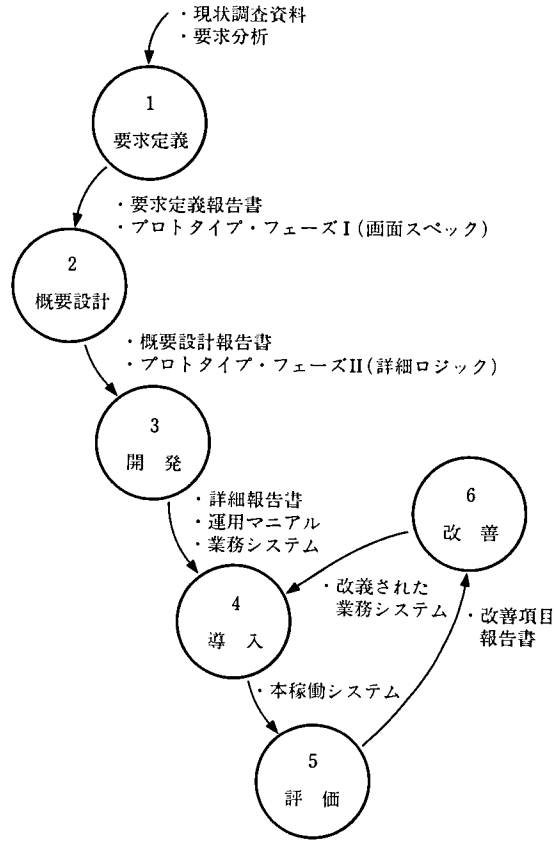


図 5　プロジェクト推進モデル（開発フレームワーク）

Fig. 5　Phase network (implementation flame work)

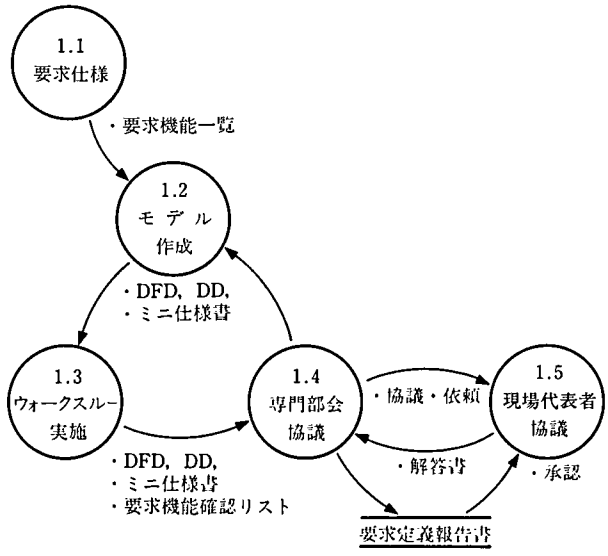


図 6　要求定義段階の推進モデル

Fig. 6　Phase network (requirements definition phase)

4.1 SIM 手法の導入

要求定義（分析)・設計における人間関係は複雑であり（とくに，エンドユーザと共同で行った場合)，ときには相反する利害関係にまで発展することもある．分析は，あまりにも不明確な仕事であるために，満足感をもたらさない．複雑なシステムは妥協しなければならことがあまりにも多く，ターゲット・ドキュメントの協議に参画した当事者は自分達の譲歩を後悔し，システム・アナリストが彼らを合意に導いた功績を忘れてしまうのである．

こうした欲求不満と複雑な人間関係に悩み，かつ専門知識不足のエンドユーザ担当者を合意に導く手段として「SIM」を導入した．

1)　当社の「SIM」で設定しているSILC[2](ソフトウェア開発のライフ・サイクル)を図5のような内容に改訂して実施した．

2)　SIMによる開発作業の成果物としては，要求定義報告書と概要設計報告書が評価すべきものである．とくに，DFD[1](データフロー・ダイアグラム)，DD[1](データ・ディクショナリ)，ミニ仕様書[1]はエンドユーザを含めて合意を得るための構造化ウォークスルーにおける理解を深める資料であり，次の行程への橋渡しを効果的に行えるものとなった（図6)．

以上のように，要求定義段階におけるプロトタイプングの特徴は，LINCが提供するISPEC*(DFDによる業務画面)・データ辞書(DDで定義された)等を含めて，EDPの専門知識不足のエンドユーザとのコミュニケーション・ギャップの克服と要求仕様の正当性の確認を可能とした．

---

*　ISPEC：ビジネス・モデルとLINCの領域をインタフェースする処理単位（主にオンライン処理系）

また，概要設計段階においては，詳細ロジックのプロトタイプを実施することにより，ユーザとの開発実行承認を得られることができた．

4.2 LINC導入の評価ポイント

以下に今回の事例の評価ポイントを記述する．

1) オンライン（ネットワーク）およびデータベースに関する知識と言語が，開発担当者の技術として必要としなかった．
2) プロトタイピングにより「システム仕様」ではなく，「実物のシステム」での検証・評価が，システム分析段階より実施できた．成果物としてのシステムが完成するまで，エンドユーザが関与できないという問題を除去することを実現した．
3) LINCが提供するLINCLOGと解析ユーティリティにより，監査記録の蓄積と適時な監査リストの作成を可能にし，豊富なLINCシステム指令により業務システム運用の安定稼働とパフォーマンス維持が実現できた．

5. おわりに

ソフトウェア開発の上流部分（要求定義・概要設計）に適用した「SIM」と上流部分から下流部分（プログラム開発）で適用した「LINC」によって，COBOL換算で50万ステップを超す大規模システム“住民記録システム”が完成した．本稼働後若干のトラブルと改善事項はあったが，1987年10月1日より約2年半にわたり安定稼働をしている．この住民記録システムの稼働後，ホスト・マシンのレベルアップ（A 10からA 17）が行われ，他の住民情報系の大規模システム（住民記録システムより大規模なものでも3システム）が開発を終了し本稼働している．現在，大・小規模合わせて20を超すシステムが稼働している．

今回のソフトウェア開発方法論「SIM」について，内容を開発担当者およびユーザ環境において使いやすいように，かなり修正（成果物を最低必須項目の範囲に絞り，表現内容もわかりやすく）を施した．また，「LINC」により作成された“ISPEC”の中から共通利用できるものを抽出・整理し，システム標準とした．

参考文献 [1] T. Demarco, 構造化分析とシステム仕様，日経マグロウヒル社，1987.
[2] Software Implemenation Methodology 解説書，日本ユニシス(株).
[3] 平山道彦，LINC適用によるソフトウェア開発事例，UNISYS技報 No. 20, 1989,pp. 42～56.

執筆者紹介 森 山 勉（Tsutomn Moriyama）
昭和23年生．47年東京電気大学電子工学科卒業．49年1月日本ユニシス(株)入社．流通業および公共関連システムの設計・開発を担当．現在，社会公共システム本部公共システム二部に所属．

MAPPERによる経営情報システム構築 ——西部ガス(株)における事例

The Creation of a Management Information system by Using MAPPER ——How It was Developed at Saibu Gas Corp.

森　正光, 紙谷　啓一郎

要　約　1970年代中頃米国で生まれた新しい経営手法「戦略情報システム(Strategic Information System)」は，日本でもシステムを構築する企業が増えてきた．しかし，戦略情報システムの構築はまだ始まったばかりで，どのようなシステムを構築すれば良いか模索している企業も多い．

本稿は，データベースを全社的に統合し，このデータベースへのデータの集約から加工・検索までの機能を持つOAシステムの開発により，情報の戦略的な活用が可能となった事例の紹介である．

Abstract　a new business management method——a strategic information system (SIS) which started to be heard of in the United States in the mid-1970 s —— has also become popular among Japanese businesses. In Japan, however, they have just started their efforts to build what is called strategic information systems, and many are still in doubt as to what systems to create.

This article describes a sample case where it has become possible to make strategic use of information by developing an office automation system which has its database integrated on a company-wide basis to allow top-and middle-management to collect, process and retrieve required data through the help of MAPPER.

1. はじめに

最近，日本においても戦略情報システム(SIS)を構築する企業が増え，同業他社を大きくリードするような成果を上げた事例も数多く報告されるようになった．従来の「ひと，もの，カネ」という経営資源に「情報」という新しい資源が加わり，情報の収集から加工，そして利用をいかに効率的にできるかが今後の経営を左右する重要な課題になってきているからだ．

西部ガス(株)においても，新本社ビル建設に合わせて構築した全社的なOAシステム(「西部ガス新本社OAシステム」)により経営層から担当部門まで情報の戦略的な活用が可能となった．

西部ガス新本社OAシステムは，全社のデータを集約した総合データベースを中心とし，意思決定支援を目的とした経営情報システムや中間管理層の企画・立案支援を目的とした計画管理システム等，情報の戦略的な利用を支援する11のサブシステムから構成されている．

システム構築には，4GL(MAPPER)を全面的に採用し，データベース構築や他ソフトウェアとの融合等，4GLの特性を随所に生かした仕組みを造り上げた．

本稿では，このシステムを事例として紹介する．

2. 西部ガスにおけるデータベースの戦略的活用

2.1 総合データベース構想

西部ガスでは，従来から業務のコンピュータ化を強力に推進してきたが，業務の拡大によりデータの重複，アプリケーションの重複，さらには全社的なコードの不統一により情報収集時に混乱が発生する等，利用部門が最新の情報を迅速に入手しにくい状況になりつつあった．そこで，西部ガスでは電算部門(西部ガス情報システム株式会社)を中心として全社的なシステムの見直しを計ることとなり，既存のデータベースの調査・分析から着手した．

その結果，コードはもちろんファイルの形態までさまざまな形で存在することが確認され，当時の状況では最新の情報を迅速に収集しそれを戦略的に活用する環境が形成されてないことが判明した．

一方，西部ガスは将来予測される環境(天然ガスへの転換，エネルギー分野における電力との競合等)や価値観の変化に柔軟にかつ迅速に対応するために，情報を全社的に集約し管理する必要性にせまられていた．

ここで，新たに全社的なデータベースを再構築することになり，これを西部ガス総合データベースと呼び，次のように定義した(図1(a，b))．

(西部ガス総合データベース)

「情報を企業戦略の中核に据え，企業経営，組織運営に役立つもの」

このデータベースは，西部ガスの目指すSIS構築の上で基幹としての役割を担うものとして位置づけられた．

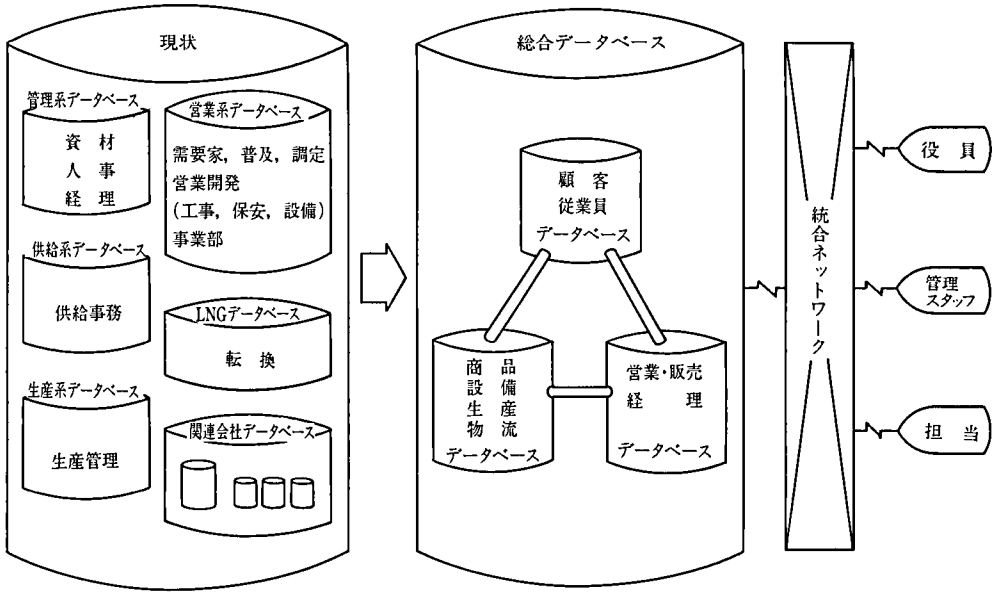


図 1(a) 総合データベース概念図

Fig. 1(a) Total database concept

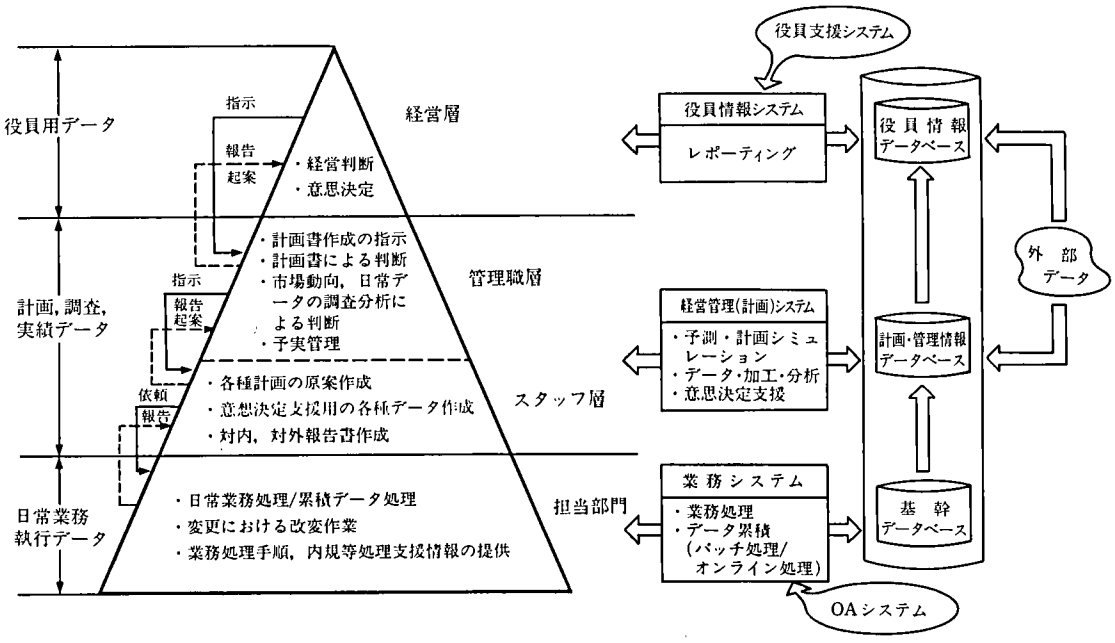


図 1(b)　利用者層から見た総合データベース

Fig. 1(b)　Total database system viewed by user

2.2 SIS インフラストラクチャとしてのデータベース

SIS における総合データベースとはどのような形態のものであろう．前提になる条件を整理してみると次のようになる．

ホストコンピュータを中心にいろいろなデータが発生と同時に収集・蓄積され，他のデータと組み合わされ，社内のどの部署でも自由に利用できるといったものである．つまり，従来の各種業務単独のデータベースの統合化・一元管理化を計り，重複データ・分散データを整理・統合した全社的なデータベースを指す．

総合データベースを構築し，情報の戦略的活用のためにはデータベース本体および管理ソフトウェアにおのずから要件が出てくる．これら要件をまとめると，①マンマシン・インタフェース(エンドユーザの操作性)，②現有資産の有効活用，③データ管理機能，④システム運用の容易性，等である(図 2)．

これらの要件を満足するためには，「エンドユーザが気楽に使えるマンマシン・インタフェース」が重要となる．つまり，SIS の情報データベースとしてリレーショナル型のデータベースを中核に据え，エンドユーザが使いやすい環境を作り上げることが必要である．

そこで，SIS 構築のためにはエンドユーザの操作性を重視したソフトウェア(4 GL)を使用することが必須となってくる．なかでも，MAPPER はリレーショナル型のデータベース構造を持ち，エンドユーザ・コンピューティングの面で優れていることから，今回の SIS 構築で全面的に採用されることとなった．

2.3 MAPPER/IDBKIT の開発と適用

MAPPER による総合データベース構築のポイントは次の 2 点であった．

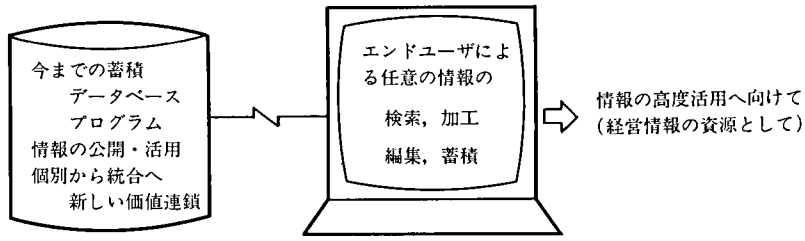


図 2　システムの要件

Fig. 2　Systems requirements

1)　現有資産の有効活用

- 現有システムへの影響がないこと
- 既存データ定義の有効活用
- 既存データベースの変更を必要としないこと

2)　データベース管理

- 基幹業務システムと連動しつつ独自のデータベース管理を行う
- 基幹業務と重複したデータを持たない
- エンドユーザが情報をうまく使いこなせる環境を準備する

今回の大規模な SIS 構築のためには，MAPPER の持つ機能だけでなく，上記問題点にも対処できるより強力な支援ツールが必要であった．

そこで，今回システム開発に当たった西部ガス情報システム(株)と当社は共同によ

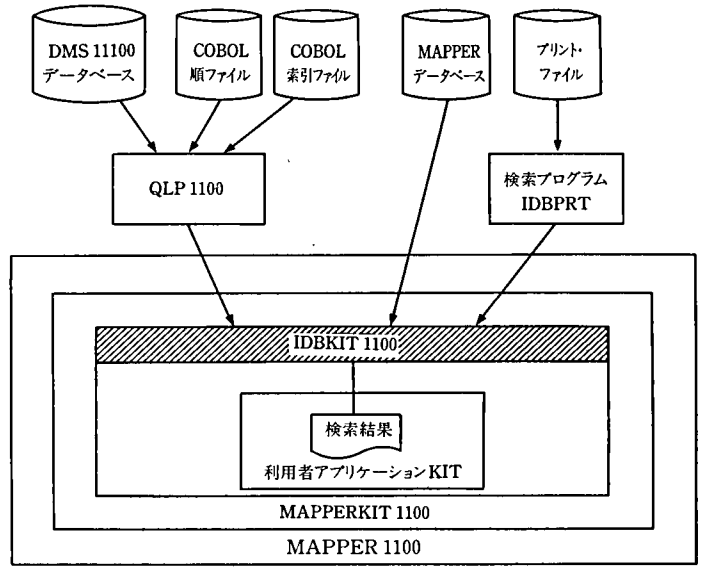


図 3　MAPPER/IDBKIT の概念図

Fig. 3　MAPPER/IDBKIT concept

り総合データベース構築支援システム(MAPPER/IDBKIT)を開発し適用した(図3).

これにより，戦略的データベースを4 GL(MAPPER)で構築し支援する体制ができ，生きた情報をより速く収集し加工して情報を戦略的に利用する環境ができた.

3. 西部ガス新本社OAシステムの概要

3.1 第1次開発

総合データベースを利用し，経営層から担当部門まで全社的に情報を高度活用するためには，戦略的な支援システムが必要となってくる．今回，西部ガスでは新本社インテリジェントビル建設に合わせて全社的なOAシステムを構築した.

システムは11のサブシステムから構成され，その中でも計画管理システムと経営情報システムが西部ガス新本社OAシステムにおける戦略情報システムの中核となり，全社的に最新の情報を提供している(図4).

3.1.1 計画管理システム

本システム最大の特徴は，個別情報である既存業務データベースを変更することなく部門(業務)を横断して有機的に計画管理データベース(MAPPERデータベース)へ結合し，利用者に公開することにある.

そこで，ファイルの属性にとらわれず，横断的なファイル・アクセスが自由にでき

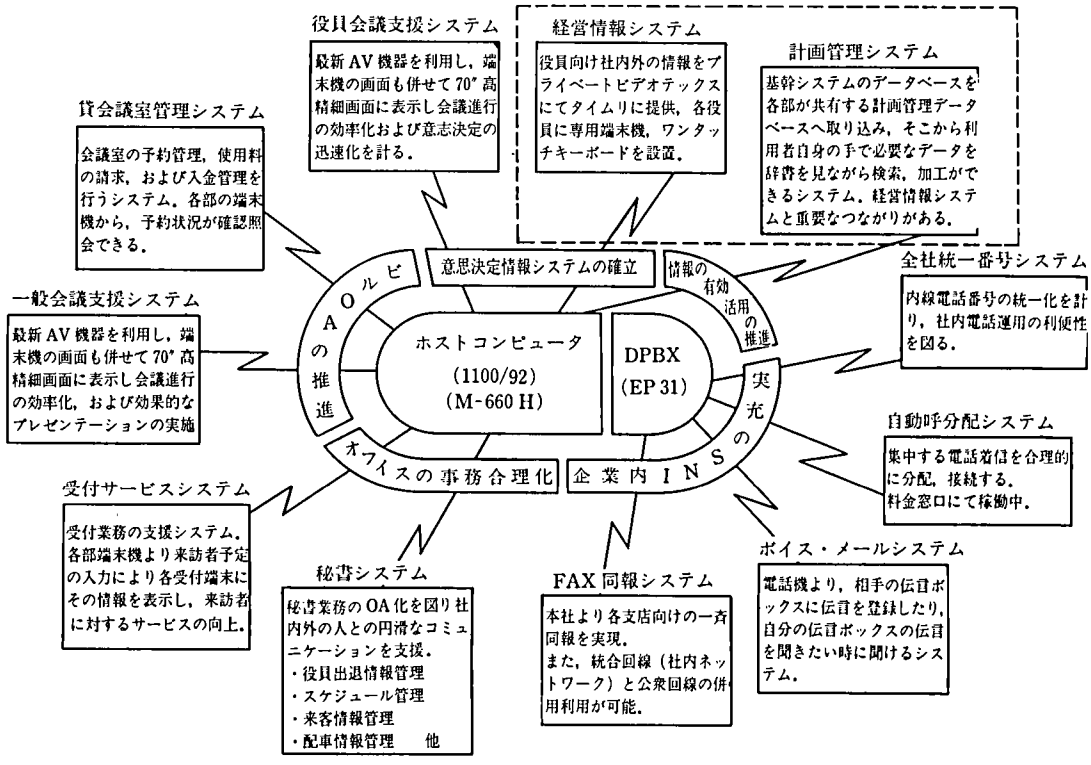


図4　西部ガス新本社OAシステム概要

Fig. 4　Saibu Gas OA System for new headquarter

るツール(MAPPER/IDBKIT)を用いることにより，横断的なファイル・アクセスはもちろんのことユーザ・フレンドリなデータ辞書や各種ユーティリティ群を組み合わせて各種データを一元的に取り扱うことが可能となった．

計画管理データベースは大別して2通りに分類される．一つは利用者に開放する計画立案用の作業環境と，もう一つは経営情報システムへデータを渡すための累積データである．これらのデータは，既存業務データベースから集約・圧縮して取り込まれて累積されている(図5)．

言い換えれば，計画管理データベースが西部ガスにおける全社的なデータの中心であり，戦略的データベースとなっている．

3.1.2 経営情報システム

経営情報システム構築に当たっては，SISにふさわしいハードウェアの採用とさまざまなソフトウェア面の工夫をこらしている(図6)．

今回採用したハードウェアは，

① 商用TV兼用15，21インチ・カラーディスプレイ

② スクリーンキーボード

③ 端末用リモート電源装置

④ ホスト接続の役員出退パネル

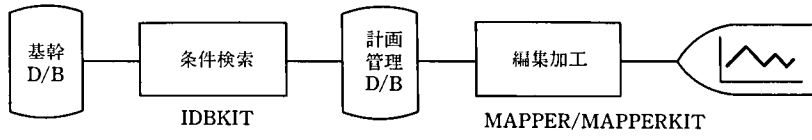


図5 計画管理システムの概念

Fig. 5 Planning management system concept

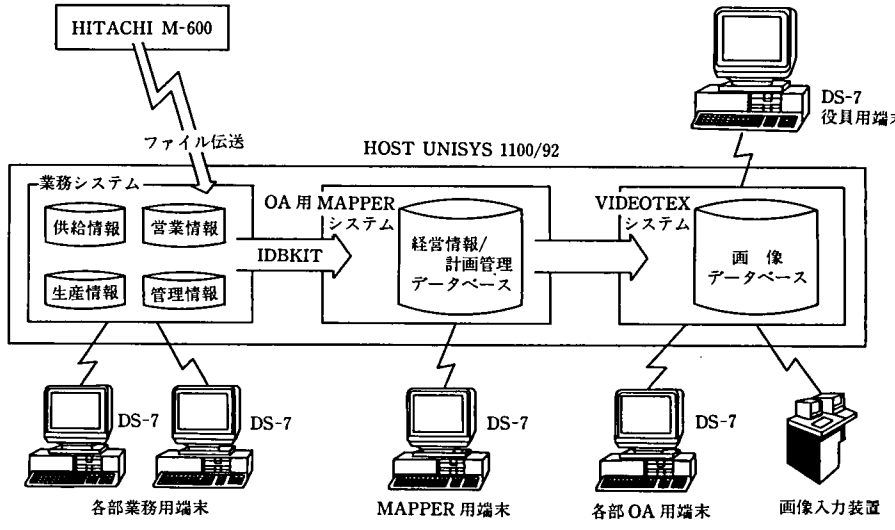


図6 経営情報システム構成図

Fig. 6 Management information system architecture

⑤　端末機と接続の会議室用70インチ高精細スクリーン

等である．

一方ソフトウェアでは，NAPLPS方式*のビデオテックスを採用してデータを画像化することにより，見てわかりやすい情報を全社的に提供している．

以下に経営情報システムの特徴を記述する．

1)　表現力豊かな画像……専門の画像制作業者による枠画面(以下下絵と記す)とプログラムから出力されたデータを合成して表示される画像は，NAPLPS方式の特性が十分生かされ，美しく表現力豊かな画面を経営層に提供している．

2)　画像の自動生成……既存業務データベースが業務アプリケーションで更新されるたびに，データを自動的に経営情報データベースに反映させる仕組みを開発した．ここでは，COSTARスケジュール機能を使用し，更新された業務データを計画管理データベースに取り込むMAPPERランを起動する．さらに，データをビデオテックス画像化するためのMAPPERランにより，経営情報データベースにデータを書き込んでいる．これにより，人手による画像作成の手間とコストの削減を計った．

3)　TLTSUDの採用(MAPPERによるメディア変換)……MAPPERデータをビデオテックス画像に変換するためには，データベース構造の違いから何らかの変換ツールが必要となる．そこで，コネクト・バッチ形式のビデオテックス画像作成ユーティリティ(TLTSUD)に着目し，計画管理データベース内の画像作成ランからこのユーティリティの実行制御文(JCL)を生成し，バッチ起動した．

4)　汎用グラフルーチンの開発……計画管理MAPPER内の画像作成ランは，作成する画像数に近い本数必要であり，1本ずつ個別に開発したのでは生産性が悪い．そこで，グラフ・数表等の種類別に汎用サブルーチンを開発し，システム開発の負荷軽減と生産性および品質の向上を計った．代表的なサブルーチンは，折れ線グラフルーチン，階段グラフルーチン，単純棒グラフルーチン，積み上げ棒グラフルーチン，3次元棒グラフルーチン，数表作成ルーチン，等である．

5)　M/C負荷の軽減と操作性・レスポンスの向上……ビデオテックス画像の検索を直接検索方式にすることで，役員用端末に接続されたワンタッチ・スクリーン・キーボードよりボタン一つで検索したい画像を即座に表示することが可能となった．また検索の際，ユーザ・アプリケーションでデータ処理することもないため，マシン負荷が少なく，レスポンスも向上している．

以上の特徴をまとめて，業務データベースから経営情報システムの画像が作成されるまでの自動化された流れは，図7に示すとおりとなる．

3.2　新本社OAシステム・第2次開発

開発した計画管理システム・経営情報システムは実績値をベースとし，データの推移や予算・前年実績との対比が情報の中心となった．

今後，情報を高度に充実・発展させ，側面から長期経営計画を支援するためには，各種予測・シミュレーションにより将来予測を行うケースも出てくる．

---

*NAPLPS方式：北米ビデオテックス通信規約

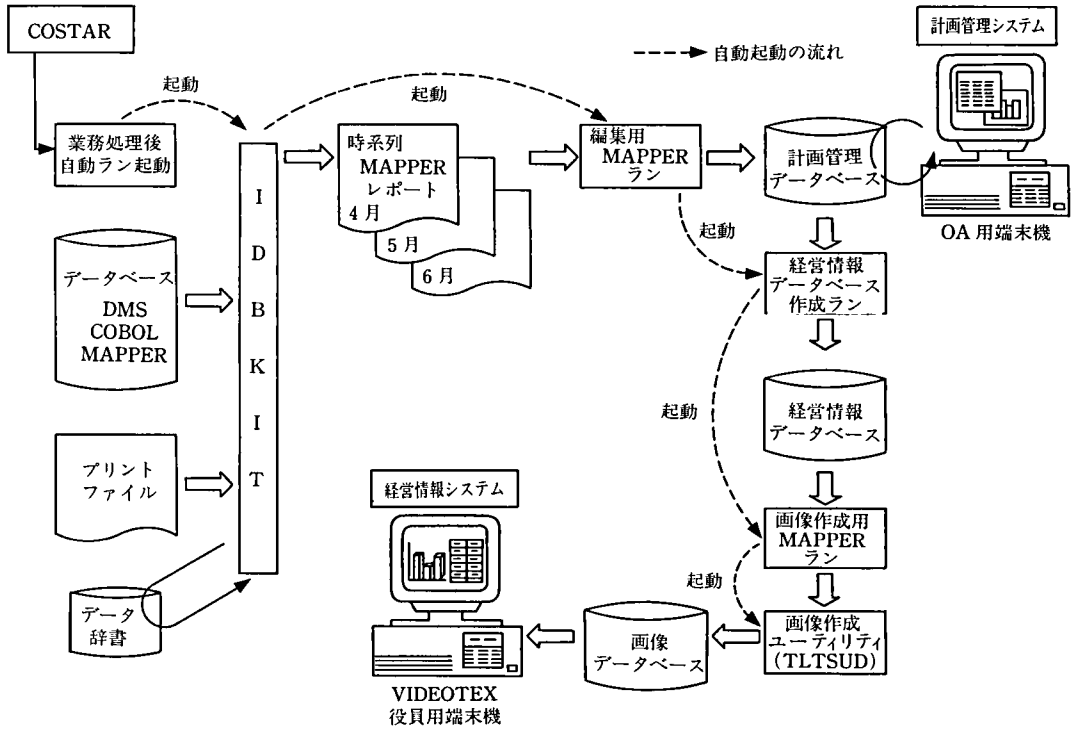


図 7　データの自動取り込みから画像作成されるまでの流れ

Fig. 7　Data stream from automatic data input to graphic output

第 2 次開発では，計画管理システムのデータ加工機能の強化や経営情報システムの充実を目的としたツール開発を行った(図 8).

3.2.1 MAPPER による予測ツール

予測・シミュレーション機能を持つソフトウェアは多いが，1100 シリーズでは機能の豊富さから，SUFICS(Super Financial Integrated Control System)が代表的である．計画管理システムにおいても SUFICS を採用することにより，利用者に予測・シミュレーションができる環境を用意することになった．

しかし，MAPPER 会話操作を中心としたデータ公開の状況で，使用形態もデータベース構造も異なるソフトウェアを新たに利用者に教育し，使用させるのは利用者側の負担が大きい．

そこで，MAPPER 側に SUFICS とのインタフェースをとるツールを開発し，利用者は SUFICS に関する知識がなくとも予測・シミュレーションが行えるようになった(図 9).

予測ツールの特徴は，次のとおりである．

1)　計画管理システム(MAPPER)の延長上に位置するツールである．

2)　利用者の操作性を重視し，MAPPER ランと会話機能のみで処理するツールである．

3)　計画管理システム上の MAPPER データをツール用に FORMAT 変換することなく，そのままの形で利用できる．

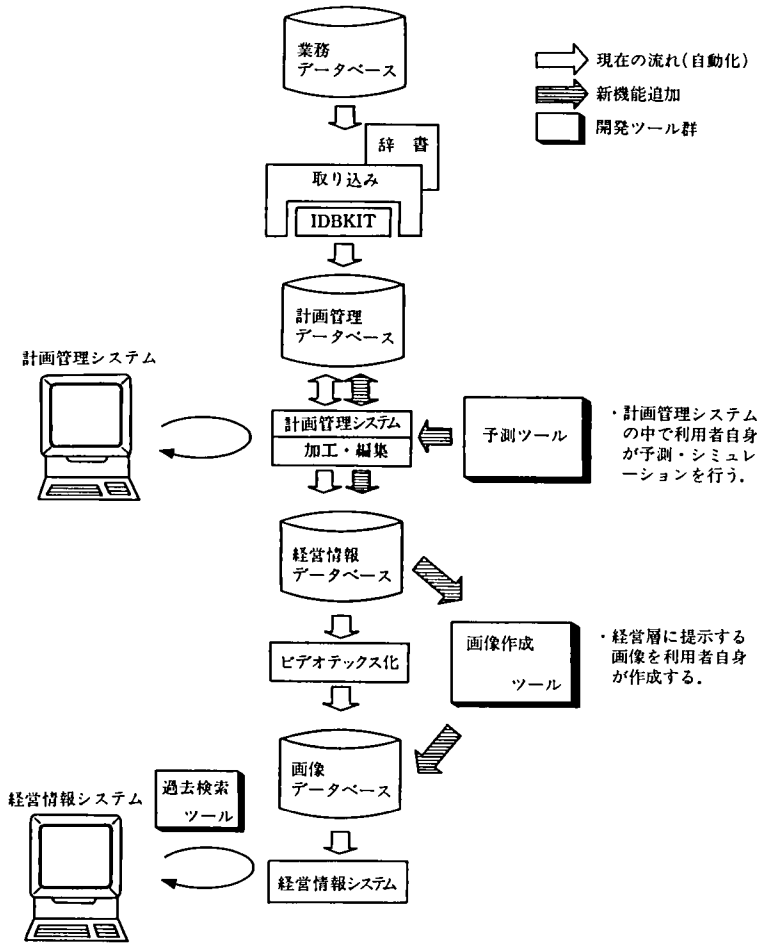


図 8　経営情報システム第2次計画の仕組み

Fig. 8　2nd phase concept for management information system

4)　予測・シミュレーションの基礎知識をガイド機能で支援し，初心者でも利用できる.

5)　きめ細かいセキュリティ機能とリカバリィ機能を持つ.

6)　グラフにより実績値と推定値の比較ができる.

3.2.2　経営情報システムにおけるツール

第2次開発では，予測ツール以外に経営情報システム内にツールを開発しシステムの充実を図った.

1)　画像作成ツール……MAPPERデータを利用して簡易にビデオテックス画像(経営情報データ)を作成するためのツールである．このツールにより，利用者自身が加工した非定型のMAPPERデータをビデオテックス画像化して，経営層に直接情報を提示することが可能となった(図10).

2)　過去検索ツール……経営情報システムで累積しているビデオテックス画像の中で，過去に作成された画像を簡単な操作で呼び出せるツールである．このツール

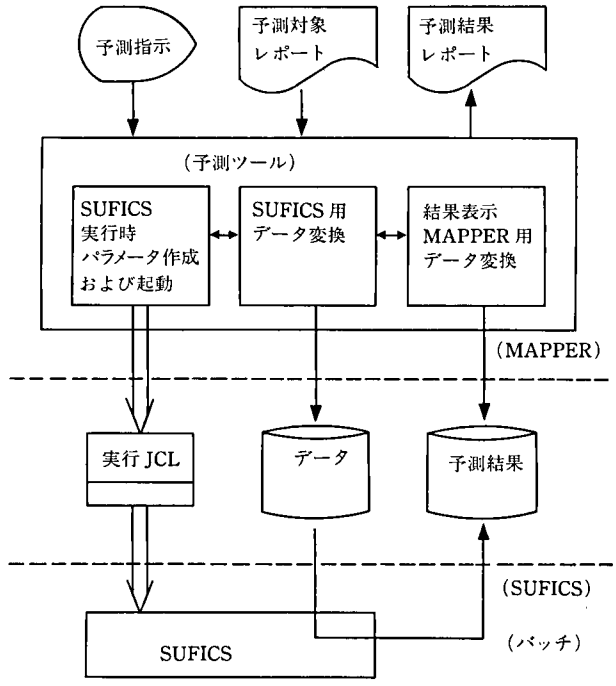


図 9　予測ツール処理の流れ

Fig. 9　Forecast tool stream

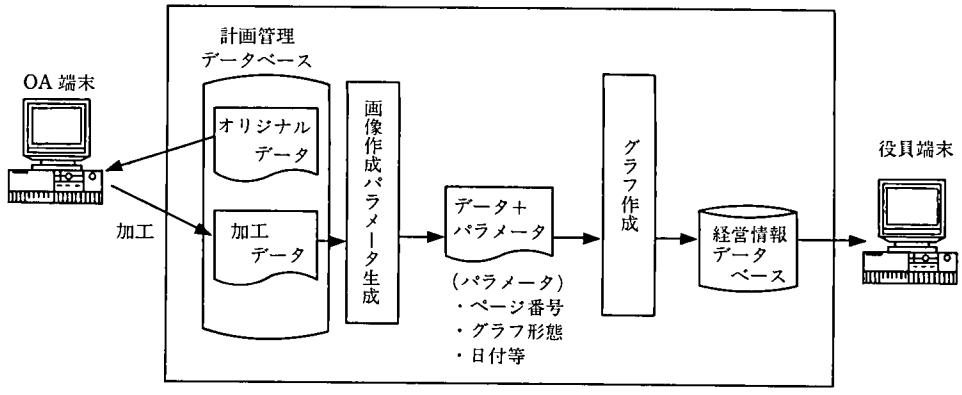


図10　画像作成ツール

Fig. 10　Graphic generation tool

により，それまで不可能であった最新画像と過去画像の比較が可能となった(図11)．

4. 他ソフトウェアとの融合

構築したシステムではMAPPERデータベースを中心としてMAPPERと他ソフトウェアとの融合を図り，利用者が気楽に使えるマンマシン・インタフェースを作り

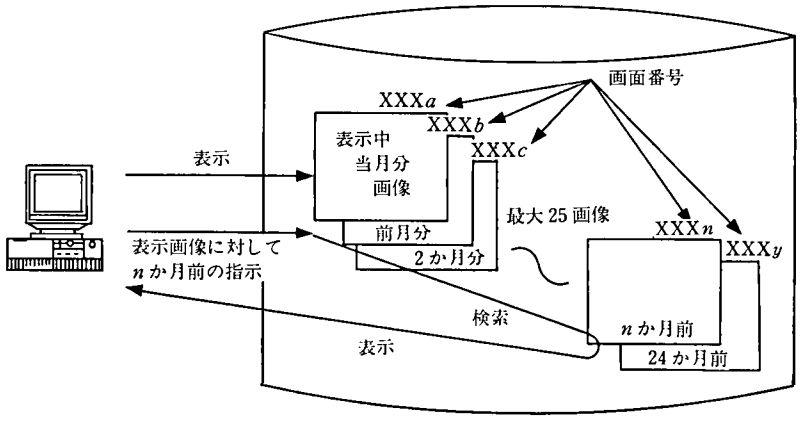


図11　過去画像検索ツール
Fig. 11　Search tool stored graphic data

上げている．

以下に，MAPPERと融合を図っているソフトウェアを形態別にあげる．

1)　業務アプリケーション
- COBOLインタフェース(PCIOS, DMS, AIS/UDS, 他HOST)
- FACILE(Facility for end users)インタフェース
- 別MAPPERインタフェース

2)　予測ツール
- SUFICSインタフェース

3)　経営情報
- VIDEOTEXインタフェース

5. 期待される効果

今回開発された計画管理・経営情報システムは，経費節約的な定量効果より，利用者のシステム活用方法に左右される部分が大きく，次の定性効果を期待している．

1)　提供情報の立体化・高度化……平面的情報から立体的情報(異次元・各次元との比較対比，グラフ化，色彩化)の提供による情報の高度化・明確化が図れる．

2)　経営情報・分析の迅速化……経営層を含め情報が直結することによって，多面的な分析が可能になると共に，より正確な意思決定が迅速に行える．

3)　情報の共有化と収集情報の相互チェック……従来部門内に止まっていた情報を各部門が共有することによって，情報がより正確に，より高度なものになる．

4)　スタッフ部門の本来業務への回帰……スタッフの手作業による報告書作成作業が自動化されることによって，スタッフ本来の企画・分析業務がより充実する．また，計画管理データベースの提供によって，スタッフはMAPPERを利用して情報を自由に加工できるようになり，より高度な企画・分析が可能となる．

6. おわりに

本システムの開発により，約700種類の経営情報画像と数百万件の計画管理データを最新の状態で提供できるまでになった．これにより，担当部門から経営層まで一台の端末で社内のあらゆる情報を検索・加工が可能となった．今後は，社外データを含めたデータの充実と急激に変化する企業環境に柔軟に対応できる統合戦略情報システムの構築をめざしていきたいと考えている．

今回のシステムは，大規模な戦略的情報システムを4 GL(MAPPER)主体に開発した事例の紹介である．

本稿が，他企業における大規模なSIS構築に何らかのお役に立てれば幸いである．

執筆者紹介 森　　正　光 (Masamitsu Mori)

昭和30年生．50年九州電子計算機専門学校卒業．同年西部ガス(株)入社．61年10月電算部門が西部ガス情報システム(株)として分離独立と同時に出向，現在に至る．

紙谷　啓一郎 (Keiichiroh Kamiya)

昭和26年生．51年武蔵工業大学経営工学科卒．同年日本ユニシス(株)入社．人社以来，社会公共関連担当としてSEサービスに従事．平成1年4月九州ソフト・エンジニアリング設立と同時に出向．現在に至る．

南王運送(株)におけるプロトタイピングによるシステム開発事例

A System Development by Prototyping at Nanoh Co., Ltd.

小 川 裕 彦

要 約 従来，出退勤に関するシステムはバッチ処理がほとんどで，リアルタイム処理を行っている例は少なく，給与計算のためのデータエントリ・システムとして，位置づけられている例がほとんどである．

本稿の事例は，「人の動き」をタイムリに捉え，有効に人材を活用していくことを目的に，一つの独立したシステムとして出退勤処理システムを位置付け，LINC を開発支援ツールとしてオンライン・リアルタイムシステムを構築した事例である．

また，将来発生するであろうさまざまなビジネス環境の変化や要求の変更に柔軟に対応していくため，プロトタイピングによる開発に挑戦をしたものである．

Abstract Traditionally, employee attendance management has mostly been computerized as a batch-processing application with the exception of only a few systems based on real-time processing; that is, the application is for the most part regarded as a data-entry system for pay-roll calculations.

With the aim of capturing 'the transition of human resources' in a timely manner and making effective use of them, Nanoh Co., Ltd. whose main line of business is cargo transportation by truck has established its employee attendance management system as independent in order to newly build an on-line real-time system with the help of LINC serving as a system development support tool.

This report focuses on how the challenging development by prototyping has been undertaken so the system can respond flexibly to a variety of changes in business environments as well as to user requirements that will occur in the future.

1. はじめに

現在，物流業界を取り巻く環境は大きく変貌を遂げようとしている．とくに運送業界においては，今まで周りの変化に対して保守的であったため，ここにきてのギャップが他の業界に比べて大きいように思われる．

その一つに労働問題があげられる．キケン，キタナイ，キツイ，「3K」と言われ，労働者の業界離れは深刻である．この慢性的労働不足に，さらに労働時間の短縮という課題にいかに対処していくかを迫られている．

このような業界の現状は，南王運送(以下当社)においても同様である．とくに人事管理，細かいレベルの労務管理，多様な勤務体制に応じた効率的な出退勤管理を行う必要があった．しかし，利用者のターゲットを絞り込んだシステムではないため，利用部門の要求が不明確であり，かつ多岐にわたっての統一性がなかった．さらに新規システムの開発でもあるため，最終的な成果がはっきりしなかった．

しかし本システム開発の目的としては，以下のような項目が挙げられていた．

1) 特殊な勤務体制に対応する

(365日，24時間営業，時間外シフト，フレックスタイム，深夜，宿泊等)

2) 時間外労働時間の管理と要因の管理をする.

3) 配員管理をする.

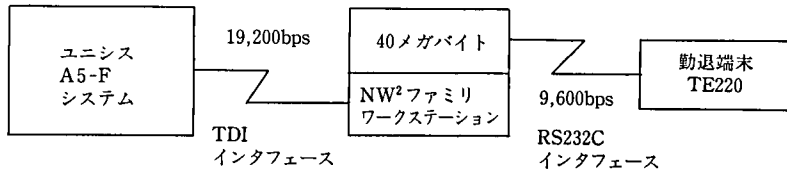
以上に対応するようなシステムを目指し，要求を少しずつ引き出すような形でのプロトタイピング手法による開発に着手した.

4GLの使用については，当社ではLINCにより全システムを構築しているため，採用するに当たってはとくに論議されることはなかったが，このシステムでは要求が明確でないにもかかわらず，短期間で高品質のものを作らなければならなかったため，開発方法についてプロトタイピングを用いることで生産性の向上を狙った.

2. システム概要

2.1 システムの構成とデータの流れ

本システムは，ホストコンピュータのユニシスA5システムにおいて，LINC IIリリース13で構築したオンライン・リアルタイムシステムとデータベース，およびNW²ファミリ・ワークステーション，勤退端末TE 220（精工舎）を使用した入力システム部分より成る（図1）.



TDI: Two-wire Direct Interface

図1 システム構成

Fig. 1 System configuration

データの受け渡しは，以下に示すように入力端末とデータベース間で行うことにより，リアルタイムに人の動きを捉えている.

1) 社員は各人のIDカードを使用し，TE 220より出退時にデータを入力する.

2) 入力されたデータは，一時TE 220内のメモリ上に蓄積される.

3) NW²より1分間隔でTE 220のデータを読み取り，NW²のディスクへ格納すると同時にA5へデータを転送する.

4) A5側では受け取ったデータによりデータベースを更新する.

以下，A5の内部処理および開発について論じる.

2.2 運　　用

入力のトリガとして，月間スケジュール表，週間スケジュール表，出退勤データ，訂正済出退勤状況表があり，出力として出退勤状況表がある（図2）.

入力された出退勤時間をもとにシフトマスタ，週間スケジュール，月間スケジュールで出勤状況（たとえば遅刻・欠勤）を判定し，残業時間を計算する．月間スケジュールでは，休暇等の大まかな予定を決め，週間スケジュールでは，月間スケジュールの変換，細かい時間の設定ができる（図3）.

スケジュールの持つ意味は，一つには出勤状況を判定するためである．たとえば，

第三事業部情報管理課　　＊＊＊ 出 退 勤 状 況 表 ＊＊＊　　90 年 5 月 27 日 (日)　　PAGE- 1
13 時 12 分

---- 社 員 ----	本日 (5/27) 出勤状況 実績	予定	実績	予定	-- 実 績 -- 出勤	退出	-- 予 定 -- 出勤	退出	前 日 (5/26) 情 報 --- 勤 怠 状 況 --- -- 実 績 --	-- 予 定 --	外出	再入	残業	深夜
000094 安倍隆夫	:00	:00	週休	週休	8:26	:21	8:30	20:30			:00	:00	4:30	2:21
000043 有川由美	:00	:00	週休	週休	8:20	18:21	8:30	17:30			:00	:00	:51	00
010189 石塚英雄	:00	:00	週休	週休	8:15	17:32	8:30	17:30			:00	:00	00	00
010782 伊丹正行	:00	:00	週休	週休	:00	:00	:00	:00	公休	公休	:00	:00	00	00
*020133 島崎順二	8:51	8:30	遅刻	休代	8:30	19:26	8:30	19:00			:00	:00	1:56	00
*040193 田中昭夫	8:12	8:30		休手	8:16	21:17	8:30	21:30			:00	:00	3:47	00
*400106 浪川新一	8:29	8:30		休代	:00	:00	8:30	17:30	㊡		:00	:00	00	00
*500070 林田一朗	:00	:00	週休	週休	10:32	20:56	10:30	21:30	遅刻	時差	:00	:00	1:26	00

図 2　出退勤状況表

Fig. 2　Time schedule

302410

第三事業部情報管理課

週 間 勤 務 予 定 表

社員名 社員コード	5/27 日 出勤	 退出	28 月 出勤	 退出	29 火 出勤	 退出	30 水 出勤	 退出	31 木 出勤	 退出	6/ 1 金 出勤	 退出	2 土 出勤	 退出
安倍誠夫 000094	週休		定時	残業	定時	残業	定時		定時		定時	残業	半休	
有川由美 000043	週休		定時		定時		定時		定時		定時		定時	
石塚英雄 010189	週休		定時		定時		定時		定時		定時		定時	
伊丹正行 010782	週休		定時		定時		定時		定時		定時		定時	
島崎順二 020133	休代		定時		振休		定時		定時		公休		当直	
田中昭夫 040193	休手		定時		定時	当直	時差		定時	当直	定時		定時	当直
浜川新一 400106	休代		定時		定時		振休		定時		直行	直帰	定時	
林田一朗 500070	週休		定時		定時		定時		定時		定時		定時	

定時出 00 | 定時退 01 | 早出 10 | 時差 11 | 直行 12 | 直帰 13 ＊月間勤務予定表は前月15日、週間勤務予定表は前週金曜までに提出のこと
残業 14 | 当直 15 | 夜勤 16 | 休出手 20 | 休出代 21 | 出張 25
週休 30 | 公休 35 | 有休 40 | 半休 41 | 振休 42

南王運送株式会社

図 3 週間勤務予定表

Fig. 3 Weekly time schedule

9：00 に出勤予定の者が 9：30 に出社すれば，これは遅刻である．では，7：00 に出社した場合はどうなるか．これらはすべて社内のルールに従って処理されるわけであるが，個人によって，日によって出勤形態が変化するため，これを判断するための基準ともなる．

もう一つの意味は，コンピュータシステムとは直接関係ないが，最も重要なことの一つで，管理・監督する者がスケジュールをきめ細かく管理することにより，組織の効率的な配員管理が可能となることである．

3. システムの特徴（LINC について）

3.1 コンポーネントとイベント

当システムでは，マスタ（固定情報）関係をコンポーネントとして，トランザクション（活動情報）をイベントして定義した．

従来よく採られている入力データを明細情報としてメモコンポーネントに落とし，加工した情報を変動情報としてマスタ（コンポーネント）に持つコンポーネント中心設計（以下コンポーネント型）の方法がある（図 4）．しかし，当システムにおいては，コンポーネント型の方法では情報の加工範囲が限定され，変化していく情報要求に速やかに対応していくことができない．

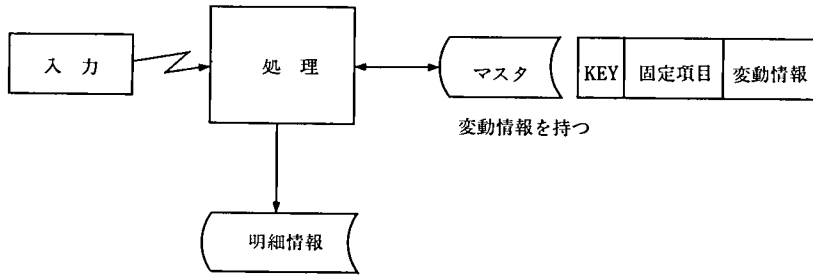


図 4　コンポーネント中心設計
Fig. 4　Component oriented-design

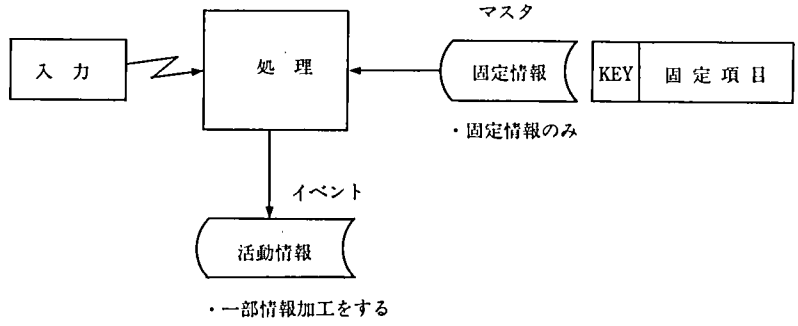


図 5　イベント中心設計
Fig. 5　Event oriented-design

そこで，今回の開発においては，フレキシブルな情報加工のできるイベント中心設計（以下イベント型）を採用することにしたのである（図5）．この形態の場合，コンポーネントには一切変動情報を持たず，一部加工した情報を明細情報に付け，活動情報としてイベントに定義する．情報の出力は，すべてイベントの活動情報を再加工することにより，その時の状況に最も適した情報を出力する．

イベント型を採用した主な理由は，以下のとおりである．

1)　LINC の自動ロジックを極力利用する．
2)　データベース内での入力データを一元的に管理したい．
3)　コンポーネント型では，出力データの加工範囲が限定されるため，マスタに累計したデータを極力持ちたくなかった．
4)　要求仕様が固まっていなかったため，コンポーネント型のようにデータを特定できなかった（コンポーネント型では，目的がはっきりしたシステムの場合はよいが，情報の視点が不特定の場合，情報として出力できるものと，できないものが発生する）．
5)　更新先をシンプルにしたい．
6)　情報をビジネスの活動結果によるものと，固定情報に分類したい．

3.2　ドキュメント

もう一つの特徴は，基本的にドキュメント類を作成していないことである．なぜなら，あらかじめ仕様ができなかったために，ドキュメント類を作成したとしても，保

概要：週間スケジュール画面より入力し、各社員の週間スケジューの
検索及び修正を行う。

各項目を入力し、エラーチェック後エラーが無ければＥＶＥＮＴの
書き換えを行う。エラーがある場合は、自画面をＲＥＣＡＬＬする。

ＫＥＹ：ＷＩＨＣＤ－ＰＲＯ
社員コード
年月日

エラーチェック：
１．ＥＶＥＮＴの存在チェック

２．社員コードのマスターチェック
ＳＴＡＦＦ－ＰＲＯ

３．人事社員マスターのチェック
ＧＬＧ：ＣＡＰＩＳ－ＳＴＡＦＦ－ＦＩＮＤ

４．有給休暇の日数のチェック
ＧＬＧ：ＹＵＫＹＵ－ＫＹＯＫＡ

５．各時間及び日付により妥当性のチェックをする。

更新：
修正処理
ＥＶＥＮＴの予定勤怠区分・予定出勤時間・予定退出時間・
所定出勤時間・所定退出時間・所定退出日及び予定退出日
の書き換えを行う。
又、各項目の変更により残業時間や有給の使用日数が
変わる時には、残業時間及び社員マスターの有給使用日数
の書き換えを行う。

検索：
ＧＬＢ．ＷＯＲＫより受け取った日付と社員コードにより
該当するレコードを表示する。

図 6　TEACH 画面定義例
Fig. 6　Example of TEACH definition

守に膨大な時間を費やす恐れがあると考えたからである．ドキュメント作成の理由は，思考する過程のメモであったり，日本語化されたものの保存である．よくコンピュータによる管理帳票の削減であるとか，画面検索によるペーパーレス化ということを聞く割には，コンピュータにたずさわる人は紙が基本的に好きなようである．確かに紙に書かれたものがないと何となく不安感は残るものの，大抵のものが紙に書かなくてもよかったり，また保存の必要がなく，まして保存媒体が紙であるために，われわれは多くの苦労を背負ってないだろうか．

以上のような理由で，文書類を極力減らすことにし，ペーパーレス化を試行した．

一つは TEACH 機能の利用で，基本的なオペレーション・マニュアルは，すべてこの画面に収めるようにした(図 6)．オペレータはいちいちマニュアルを探すことなく，自分が今実行中の内容について，処理を中断することなく操作方法を調べることができる．もう一点は仕様書（機能条件書）のペーパーレス化であり，これも LINC の基本機能の一つである TEXT 画面を利用し，保守効率を上げたものである．度重なる仕様，条件の変更をいちいち紙に書いて，訂正をしていたのでは開発作業よりも，文書化作業の方が作業量が大きくなりかねない（図 7）．

4. プロトタイピングによる開発

4.1 なぜプロトタイピングか

プロトタイピングは実物のひな形を試作，または実物と同じ物を試作することであ

```
+- --*---1----*----2----*----3----*----4----*----5----*----6----*----7----*----8+
!       この画面は、社員マスターの保守画面です。                                  !
!                                                                                 !
!       1．登録には、                                                             !
!           a、社員コード、を入れないと登録できません。                           !
!                                                                                 !
*           b、振休の残を登録して下さい。                                         *
!                                                                                 !
!           c、通常の場合は、有給の日数は入力する必要はありません。               !
!                                                                                 !
!       2．残業計算区分には、                                                     !
*           "0"と"1"しか入力出来ません。                                          *
!           "1"を入力した場合"賃金支給通知書"に残業計算されません。               !
!                                                                                 !
!       3．変更、削除の場合は、社員コードを入れ一度画面を呼び出した後、           !
!           処理を行って下さい。                                                  !
*                                                                                 *
!       4．処理区分に「－」を入力した場合マスター保守メニューへ戻ります。        !
!                                                                                 !
!       元の画面に戻る場合は、［ Back Page ］ｋｅｙを押して［ Page 1 ］に         !
!       戻って下さい。                                                            !
*                                                                                 *
!                                                                                 !
!                                                                                 !
!                                                                                 !
+- --*---1----*----2----*----3----*----4----*----5----*----6----*----7----*----8+
```

図 7　TEXT 画面利用例

Fig. 7　Example of TEACH Screen

る．プロトタイプが実システムに近いほどプロトタイプの精度は高まるが，半面，試作に掛かるコストは増加する．システムの生産性を「システムの規模/開発運用コスト」と定義すれば，開発に当たっての開発コスト，すなわち開発に当たる人件費をいかに下げるかが開発の生産性に大きな影響を及ぼすことになる．

システムが陳腐化するとよく言われるが，それは開発をする側の人間がエンドユーザに対して，常に受け身であるからである．要求を受けて，開発し，開発が終わった時点から陳腐化が始まることは当然である．なぜならば，われわれを取り巻くビジネス環境は絶え間なく変化しているからであり，それに対して受け身であれば，どんなシステムであれ，時間の流れから取り残され陳腐化という烙印を押されるのである．陳腐化を避けるためには時の流れに乗ること，すなわち要求を受けるのではなく，提案をし，システムを成長させていかなければならない．

しかし，現実には開発セクションは膨大なバックログを抱え，ビジネス環境に注意を払うことは極めて不可能に近い．従来の開発方法論では，開発の下流工程において生産性を高めることが行われてきたが，これではプログラミングレベルの生産性は上がっても，開発プロジェクトの生産性や企業全体の生産性の向上にはならないのではないか．

このような漠然とした問題点を抱え，以下，当社におけるプロトタイピングによる開発を実施した具体的理由に置き換えてみる．

1)　システムの要求が固まっていなかったので，ヒヤリング等によって調査したが，検討することがほとんどできなかったため，プロトタイピングによって要求を明確にしなくてはならなかった．

2)　エンドユーザを説得するための文書を書いている時間的余裕がなく，プロトタイプを見せてレビューしたほうが早く，また相手に理解させやすいと考えた．

3)　システムは，開発開始時点のビジネス環境や概念で既定され，環境の変化とと

もに陳腐化させるものでなく，成長を繰り返すべきものであるから，設計に当たり，フレキシブルなシステムを作りたかった．

4) 当システムの開発に割ける担当者が2名だけで，一人はLINC IIリリース10の経験が2年あるものの，2人ともLINC IIリリース13を使用することは，今回が初めてであったため，スクラップ&ビルドを繰り返すことにより，品質の向上を図りたかった．

5) 従来の工法では，繰り返し検証をする時間が十分に割けなかったので，各開発フェーズで品質の確保をしたかった．

6) 要求仕様が明確でないため，システム全体の検証をエンドユーザが行うことが不可能に近く，また開発側においても，ビジネスのこれからの要求に応えられる設計ができるかどうか判断できなかった．

7) 他事業部でも関連性のあるシステムを同時に開発中であったため，当システムの仕様が固まらないので，相互にシステム上の整合性を各フェーズで検証，確認を取りながら開発を進めたかった（図8）．

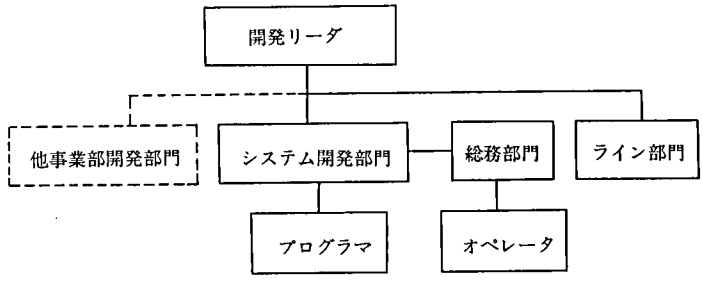


図 8 開発体制

Fig. 8 Organization of development

4.2 開 発 経 緯

開発は，①調査・分析，②設計・開発・評価，③総合テスト（変更），④並行処理，⑤機能の追加の5段階に分けて，平成1年5月から12月にかけて実施した（図9）．さらに開発のポイントとなる設計・開発・評価においては，データベースの作成，業

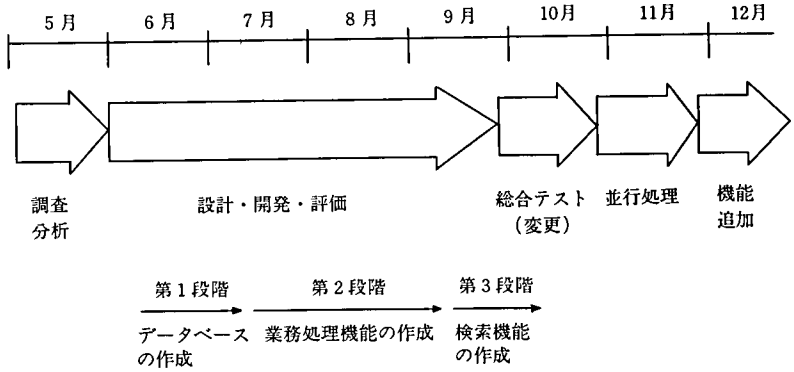


図 9 開発スケジュール

Fig. 9 Schedule of development

務処理機能の作成，検索機能の作成の3段階に分けた．

4.2.1 調査・分析

5月よりとくにエンドユーザの要求を確認することなく，出退勤システムの検討に入った．開発の骨子は前にも述べたように，特殊な勤務体制に対応し，時間外の管理と要因の追求を行うことで，配員管理を的確に実施できるようにすることである．

なぜ開発に当たり，エンドユーザの要求を聞くことから始めなかったのか．本来，開発に当たっての常套手段は，まずヒヤリングからである．しかし，エンドユーザの要求はよくマクロ的問題よりも，身近なミクロ的問題に興味の中心が行ってしまいがちである．ミクロ的問題も大切ではあるが，システム全体の設計の時点では，本来の目標を見失わせる大きな要因の一つとなる可能性が，今回の開発ではあると考えたからであり，そのような問題こそ開発過程において修正・変更していくことでエンドユーザの要望に応えていけばよいのである．

検討した内容は，データベースのアウトラインと開発担当者へのスケジュール説明，システムの概略説明，実現手段の検討である．また実現手段を検討説明しながら，LINC IIリリース13の機能の修得，最適開発方法の検討をも行った．分析に関しては，

1) 必要なビジネス活動
2) ビジネス活動に必要な情報
3) 活動間の情報の流れ
4) 活動のトリガ
5) 活動を行う手順

を明確にすることに主眼点を置いて行った．さらに，必要最低限のアウトプットを決定した．

4.2.2 設計・開発・評価

1) データベースの作成(第1段階)……まず始めに概略の分析を基に業務処理に必要な固定情報（コンポーネント），活動情報（イベント）を定義した．データディクショナリに項目を定義することにより，今後変更が起きても，データの整合性が保証されるように配慮した．続いて画面，ロジックと作成し，固定情報の作成を終了した．ただし，活動情報については業務処理機能の開発であり，この時点ではデータの定義に留め画面のロジックの作成は行っていない．また，開発および変更の整合性を保ち時間を短縮するため，グローバルロジック(共通ロジック)

表1 開発システムの規模
Table 1 Size of development system

	スペック数	ロジック数 (LDLステップ)	平均ロジック数 (LDLステップ)
グローバルロジック	25	1,026	41
コンポーネント	9	1,828	203
メモコンポーネント	4	973	243
イベント	21	32,405	1,543
レポート	30	20,409	680
合　計	64	55,615	

をコーディングし，生産性を高めるようにした（表1）．

ここまでの開発においては，データベースの構造や他データベースとの整合性に注意を払い設計に当たった．コンポーネントの開発に要した時間は3日間である．グローバルロジックの利用とマスタの基本的な画面の動きは，作成済みのものをコピーすることにより生産性を上げることに成功した．また，マスタ処理自体が複雑な処理を伴わないため，初めてLINCを使用する者にとって受け入れやすかったことが，短期間での開発を可能にしたと考えられる．

個々のISPECは，開発が終了すると同時に検証に移り，とくにこの段階では業務自体がまだ固まっていないので，機能的な面での検証に留まる．そして開発部門の中で，全体像を話し合いながらデータベースを練り上げていった．ここまでは，開発部門外でのレビューは行っておらず，プロジェクトのミーティングで開発部門の主旨説明をし，エンドユーザからの脈絡のない要望事項を聞き，必要と思われる内容をデータベースへ付加するにとどめた．なぜならば，設計の段階においてビジネスの本質のみをシステムに展開し，エンドユーザの持つ個人的な事柄や，本人が最も大切だと思っている部門間の政治的な事柄等をコンピュータシステムと明確に分離して取り組みたかったからである．

2) 業務処理機能の作成(第2段階)……データベースの作成に一区切りをつけたところで，業務処理機能の設計・開発に入った．当段階での主たる実施内容は，実際業務の考察と，それのイベントへの展開である．業務について分析をしながら開発し，エンドユーザとレビューをしながら，分析•開発•評価を繰り返した（図10）．

1週間に1回エンドユーザを交じてえのレビューを行い，業務内容，開発のコ

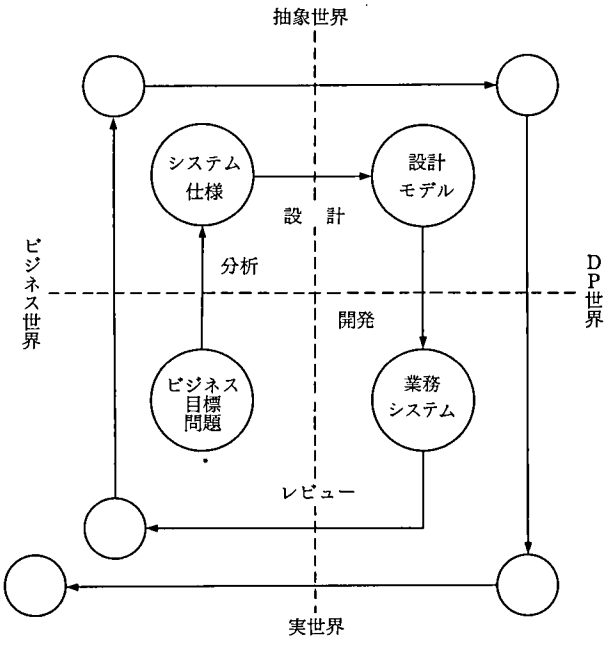


図 10　プロトタイピングのアプローチ

Fig. 10　Prototyping approach

ンセプトに対しての検証をすることにより，実システムに近づけた．さらに週に数回は開発部門内において内部のレビューを行い，プログラミングレベルでの検証を実施し，精度を高めるように努めた．

このように成長型のプロトタイピングによる開発を進めてきたので，利用部門の参加が比較的円滑に行われた．さらに，繰り返し行うレビューでの検証を最終決定とせず，システム構築上の暫定的合意事項としたことが，システムを硬直化させたり，エンドユーザに不信感を与えることがなかった．

また実システムを視覚に訴えることにより，エンドユーザの改善要求が具体的になってきた（表2）．

表2　業務処理機能のレビュー結果
Table 2　Results of functional review on application

レビュー	1回目	2回目	3回目	4回目	5回目	6回目
改善要求（件数）	3	1	7	12	8	3
変更プログラム（本数）	1	3	3	5	5	2
プログラムバグ（件数）	0	1	3	3	1	0

しかし，この段階で注意しなくてはならないことは，システムに慣れてきたエンドユーザが，対象とするプロトタイプの範囲を越えた考えを持ってきてしまうことである．いくら成長型のプロトタイプといえども，開発の範囲を明確にしておかないとエンドユーザの要求に翻弄され，開発目標のコンセプトまでも見失なう恐れがあるからである．

3)　検索機能の作成(第3段階)……第1段階のデータベースの作成，第2段階での業務処理機能の作成では，開発部門主導のプロトタイピングであったが，第3段階の検索機能の作成においては，エンドユーザの要求を反映するように設計とレビューを実施した．またレビューにおいて要求を満たしきれない部分は，第1段階，第2段階への改善・変更作業をし，検証作業を行った（表3）．

表3　検索機能のレビュー結果
Table 3　Results of functional review on inquiry

	プログラム数	変更本数	変更回数	DB変更	処理変更
検索画面	12	7	15	3	2
帳　票	25	12	13	0	1

第2段階と比較して，複雑な処理機能より検索機能の方が理解しやすいこともあるが，エンドユーザの参加意識が高まり改善・変更要求がより多く具体的になってくる．また第2段階までのシステム機能への変更が少ないことは，プロトタイプがある程度の精度まで達したと考えられる．

4)　総合テストでは一部変更を実施したものの，基本的事項に対する改善・変更はなく，一部検索機能の追加とプログラムのバグに因る修正が発生したにとどまっ

た．その後20日間の現業務との並行処理を行い，本稼働を実施した．本稼働後は，1か月に1回プロジェクトミーティングを開き，機能の追加・変更を行いシステムを拡張している．

5. 効　　果

本稿の事例では，当社の特殊な勤務体制のために，初めに述べたような理由で，新たな試みとして，LINCを使用したプロトタイピングによる開発を実施した．24時間営業体制ということで最も問題となったことは，どの時点で一日を区切るかということであった．時間の流れを区切ることなく，一連のデータのコードとして時間を捕えることにより発想を転換していった．これも分析時点で仕様書に向かっていたのでは，考えられなかったことである．また，開発の初期段階において，エンドユーザに対しヒヤリング等を行わなかったことは，開発サイド主導型の自社開発であったためであり，エンドユーザの要求に対して応えていくのではなく，エンドユーザに管理させるためのシステムであったからである．

したがって，実際にエンドユーザが開発のコンセプトを理解したのは，システムの開発に入り，プロトタイプのモデルを繰り返しレビューしてからであった．

当システムのように，あらかじめ仕様を決定することが困難な場合，プロトタイピングは思考錯誤を繰り返しながらシステムを開発する上で非常に有効であり，以下の効果があったと思われる．

1) システムの変更や拡張が簡単にできる．
2) 開発中に繰り返しレビューを実施でき，エンドユーザに見せることができる．
3) 利用者の要求を満足させるシステム構築ができる．
4) 利用者にシステムのコンセプトを徹底できる．
5) 確認のための仕様書等のドキュメントが簡略化できる．
6) 本システム運用への円滑な移行が可能である．
7) エンドユーザ教育の時間が短縮できる．
8) システムの全体像をあらかじめ把握することができる．
9) プロトタイプシステムと実システムとを意識して分ける必要がない．
10) 開発時間が大幅に短縮できる（とくに設計段階の所要時間とエンドユーザとの内容確認，結果の検討にかかる時間）．
11) 開発に携わる人数が小規模のほうが効率が上がる．

しかしながら，LINCを使ってのプロトタイピングによる開発に慣れていないこともあり，逆に当初の予想ほど効果が出ない面もあった．

以下のような，反省点・注意点が挙げられる．

1) プログラム仕様書は，TEXT機能で対応できたが，開発がある段階まで進行した時点で，システム概観を表すドキュメント類を作成しておかなければならない．
2) LINCを使用して繰り返し生成を行うため，開発用のマシンにある程度の余裕が必要と思われる．サブシステムを使用しなかったため，とくに生成時間がかかった（開発環境については測定していない）．
3) 分析段階において，LINCを分析ツールとしてダミーモデルを作成して使用し

てみたが，かえって効率が悪かった．LINC は，開発(設計)支援のツールであり，基本的な部分に関しては既存の分析方法論に従って行うことが望ましい．

4) レビューの度に機能を追加していくので，1 ISPEC の処理が複雑になりステップ数も増加した．システムの内部を疑似的にモジュール構造化し，モジュールの強度を大きくし，モジュール間の結合度を小さくする努力が必要である．

また，以下の事項はプロトタイピング実施上の注意点として一般的に述べられていることではあるが，今回の開発を通して再認識させられた事項でもある．

① プロトタイプの開発範囲を明確にしておかないと，開発が長引く恐れがある．

② 自社内で開発できない場合，コミュニケーションに十分注意を払わなくてはならない．

③ システムの規模が大きい場合，拡張や変更を繰り返すうちに本来の開発目標を見失いやすい．

以上を総合的に評価すると，LINC の持つ高生産性を考慮しても，かなりの生産性をプロトタイピングによって上げることができた．LINC の場合，言語の容易性もあるが，変更に伴うデータベース内の整合性の保証が繰り返し行われるシステムの再構築を可能にしている．すなわち，LINC を使用して最も効率のよく，保守の行いやすい開発方法の一つがプロトタイピングであると確信する．

6. おわりに

既定概念によって定義できるものが，現代では少なくなってきているのではないだろうか．言い換えれば，不確実な社会，激動する世界において固定された既定概念では，変化に対応，発展していくことができなくなってきているのである．ビジネスの世界においては変化に対応し，自らが変化を作り出していかなければ，常に後処理に追われる毎日を送らなければならない．

環境に対して受け身であれば，われわれソフトウェアに携わる人間は何時まで経っても，頭脳労働者と言われながらも労働集約的な仕事から抜け出せないのである．残念ながら本稿の事例においても，生産性の測定基準を内容ではなく，人/月というメジャーで計っている現状である．常にこちらから提案をし，システムを成長させていかなければならなく，相手の要求が出てきた時では遅いのではないだろうか．

プロトタイピングにより開発されたシステムは，でき上がって終わりのシステムではなく，常にビジネスの環境を先取りして，成長していかなければならない．現場に合わせるための試作モデルを作ることではなく，「次にどのようにあるべきか」を模索するための試作モデルなのではないだろうか．今回の開発を通じて感じたことは，ある程度現状分析ができる場合，単純に業務をシステム化するようなケースでは，プロトタイピングによる効果は小さいと思われる．なぜならば，そのような場合のプロトタイプは，エンドユーザへのレビューをするためだけの方法になってしまうからである．

プロトタイピングにより具体的なモデルを見せ，要求をエンドユーザから引き出すこともよいが，現在ではこの要求自体が明確な場合が少なく，仕様が固まらず，開発のサイクルの中で，開発部門と現場が模索しながら最終的なビジネスの型を求めてい

かなければならない．

最後に今後の課題として，プロトタイピングをもっとビジネスに近づけた積極的な運用方法ができないであろうか．エンドユーザの要求を満たすためだけのシステムばかりを，コミュニケーション等という甘い言葉で包んで満足していては，企業の進歩は危ぶまれる．過去にあったような，現場のニーズを無視したお仕着せ型ではなく，開発部門は請負仕事をやめ，プロトタイプと使い勝手のよい高級言語を駆使して，次々にビジネスのパターンを提示していくことである．開発部門はプロジェクトのイニシアチブをとり，ただ単に要求を満たすために成長をさせるのではなく，エンドユーザに対してニュービジネス，ビジネスの新しい型を創造するような攻撃型の開発を行い，コンピュータルームに籠城することをやめ，ビジネスの第一線で仕事を展開することが，これからのシステム開発でのポイントになるのではないだろうか．

参考文献　山下淳一，「システム開発におけるプロトタイピング技法」，日本ユニシス(株)，1989年．

執筆者紹介　小　川　裕　彦（Hirohiko Ogawa）

昭和31年生．54年立教大学法学部法学科卒業．同年南王運送株式会社入社．納品代行の検品業務に従事した後，57年2月よりコンピュータ業務を担当し，以来，第三事業部のシステム分析・設計・開発を主に，全社のシステム統合化を推進する．現在，第三事業部情報管理課課長．

シャープ(株)におけるエンドユーザ中心型生産管理システムの構築

The Construction of an End-user-oriented Production Control System at Sharp Corporation

青木好治，菊池　豊，斎藤孝夫，伊東　守
梅津長央，小林敬三，小野崎誠，八木沢勝正

要　約　電子機器事業本部ビデオ事業部では生産管理業務に携わっている，われわれエンドユーザが新たに柔軟な生産管理システムを自ら開発した．狙いは生産管理技術の激しい進歩に即応すること，また生産情報を戦略的に活用することであった．

このシステムの構築に当たっては，エンドユーザ中心で開発を進めるという大きな課題以外にも，開発期間の短さや使用端末の制約等，各種の課題が存在した．しかしこれらの課題を，MAPPERを使うことにより克服し，大きな成果を挙げている．

Abstract　We, end-users responsible for production control applications at Sharp's Video Business Operations, have newly developed a more flexible system for production control on our own initiative. The motives for this were for us to keep up with rapid progress in production control technology and to make more strategic use of the information provided by the system.

In developing the system, we were faced with several problems such as a relatively short period of time allowed for development and limited availability of usable terminals in addition to the implementation by no other people than end-users with no prior experience whatever in systems development. However, we have successfully got over those problems by using MAPPER as solution software and achieved what we can take pride in.

1. はじめに

シャープ(株)は事務機器，家電，およびOA機器の総合メーカである．平成元年度に売上高が1兆円を超えた．本社は大阪市，従業員数は約1万8千人，国内の主要な工場は9工場，海外の主要な拠点工場は12にのぼる．今回，栃木工場を中心とする電子機器事業本部ビデオ事業部では，MAPPERを利用したエンドユーザ中心の生産管理システムを導入した．本稿では，その経緯と内容を報告する．

2. システム化の背景

電子機器事業本部は栃木県矢板市にある．従業員2,850人，売上高約3,000億円のこの事業本部は，VTR，ビデオプロジェクタ，ビデオディスク・システム等のビデオ事業部と，カラーテレビ，パソコン等のテレビ事業部，および液晶ビジョン，液晶テレビ等の液晶映像システム事業部からなっている．この事業本部ができたのは昭和43年で，今では全社の売上高の約3分の1を占める戦略事業部門の一つになっている．

ここでコンピュータを利用したシステム化に着手したのは，昭和45年頃であり，生産管理の各指標をコンピュータ化することから始まった．このとき開発し構築したシステムは，その後10数年間も運用された．それは，このシステムを開発した関係者が一時転勤のため現場にいなかったことが理由の一つであった．その人たちが再びこの

事業本部に赴任してきた時，このシステムが依然利用されていたことに大変驚いたという．それだけこのシステムが優れていたと言えるのかもしれない．しかしこの10数年間には管理指標もかなり変わり，また増えていることも事実であった．結果として，その部分は人手で処理されていたり，パソコンや電卓を使って処理されていた．

つまり，このシステムはすでに実態と大きくかけ離れたもので，部署ごとに不統一に処理されており，非常に非効率的なシステムとなっていた．これではこの分野での生産性の向上に限界があると判断せざるをえなかった．一方，ビデオ事業部では生産工程の自動化機器の導入を推進することになり，それを機に，より効率的なシステムの構築，つまりビデオ事業部の実態に合った生産情報統合化システムの開発を目指すことになった．

3. 新システムの目的

新システムの目的は，生産情報の戦略的な活用を図ることにある．コンピュータを単なるデータの処理に利用するだけでなく，性格の異なるデータをいろいろ組み合わせることによって，新しい意味を持たせたデータを作り出し，経営に役立たせることである．具体的には，各種生産情報の集中統合化，製造リードタイムの短縮化，日程計画シミュレーションといったことであり，営業分野で日々入手できる売れ筋商品とか，死に筋商品の情報をいち早く知り，生産計画の変更を実施することによって，生産分野でのむだを極力少なくすることである．

この目的を達成するためには，データの即時入力・即時利用が必要であり，各工程別にデータ管理端末を設置し，リアルタイムに処理を行う必要があった．また本部や協力会社とネットワークによって結合し，同様のリアルタイム処理を行う必要があった．

さらに，若い従業員だけでなく，課長や係長といった現場の管理者が，比較的簡単にデータを加工し，現場の状況を的確に把握できるようになることが必要であった．それには，データの検索・加工を利用者自身で行える体制の確立が重要な要素であった．

今回のシステムでは，事務処理の効率化を推進することも目的の一つであった．いわゆるペーパレス化の推進で，これまで手作業で処理していた事務作業を順次コンピュータ化していくことも狙った．

4. 新システム開発の課題

どのような技術についても言えるが，生産管理技術も時代とともに大きく進歩していく．現実にこの事業本部でもそうであった．そのため，内外の環境の変化に迅速に対応できる柔軟なシステムの開発が必須の状況にあった．

しかし本部のコンピュータ部門は，会社全体のシステムの開発・保守を抱えており，各事業部のシステムまで手が回らないのが実情であった．また従来の方法では，一度システムを構築すると，そのシステムのソフトウェアを変更するのはなかなかむずかしいことが多いのが実態であった．

こういった課題を克服するために事業部が自らシステムを開発し，また変更してい

くことができるシステムの構築を決断した.

すなわち，生産に関する情報を①リアルタイムに，②エンドユーザが自由に検索できるシステムを，③生産管理業務に精通している利用部門で，④独自にシステムを導入し開発・運用・保守できること．つまり生産管理技術の進歩に即応でき，そして生産部門のトータル・システムを大型コンピュータで柔軟にプログラム開発・更新できることを目指したのである.

こうした背景，ユーザニーズから，第四世代言語を利用したシステムの構築を検討することになった.

5. 新システム構築のための要件

新システム開発に際しての第1の要件は，ビデオ事業部が独自にシステム開発を行うということである.

このとき，システムを導入するに当たっての基本的な考え方を，以下のようなこととした.

1) コンピュータの知識のない人でも簡単に，かつ自由にシステムの開発から運用・保守ができること．データの発生する現場で，独自にシステム処理できなければ，変化の激しい生産管理のシステム化はむずかしいと判断した．現場に密着したシステムの構築が大前提であった.

2) すでに導入し利用しているパソコン上の情報資源をそのまま活用すること，とくにデータベースが簡単に取り込めることを考えた．新たにデータベースを構築するとなると，大変な作業になり，むだでもある．そこで新システムでは，こうした膨大な既存のデータベースをそのまま利用することを考えた.

3) すでに多くの従業員がパソコンの操作に慣れているが，そのノウハウを活かすことである．慣れているパソコンの使い勝手と，同じくらいやさしい操作性を持った言語であることが必要であった.

4) 利用部門に生産情報を公開するために，システム自体に機密を保護する機能を持っていることが重要であった.

5) 通信，電子メール，操作支援，学習機能等を持っていることも重要な条件であった.

6) その他，ニーズに即応したレポート形式でデータが加工できること，グラフィック機能を持っていること，ハードウェアのレベルアップに際して，プログラムの書換作業を必要としないこと，等が条件であった.

以上のような条件を満足させることで，柔軟性のあるシステムが構築できると考えた.

さらに，新システムはネットワークによって広がりのあるシステムを目指すことになっており，この面からの条件も必要であった．その際，シャープ自身が生産している自社製品のパソコン，X 68000 を専用の端末機として利用できることも必要であった.

また，当社では全社的なホスト・コンピュータや各事業本部用コンピュータとして，ユニシス以外のコンピュータを利用しているが，この事業本部コンピュータとの接続

も前提条件であった.

そして，ビデオ事業部は工場が分散しており，とくにビデオデッキを生産している第2工場，ムービーを生産している第4工場は1 km以上離れていることから，これら工場間もネットワークで結び，あたかも一つの工場として機能できるようにする必要があった.

さらに，ビデオ事業部は主な企業で11社にのぼる協力企業を抱えている．これらの企業には，この事業部で発生するもののうち，実に70%の加工作業を委託しており，ネットワークで結ばないと新システムの効果が薄れるものであった.

これらの条件が満たされてはじめて新システムの効果が期待できることから，以上の諸条件を新システム構築の大前提とした．こうした考え方を基本にして，システムのメーカ選定に当たった.

システムの導入を検討開始したのは昭和63年4月であった.以上述べてきた条件を整理し，いろいろなメーカを訪問し，実地にシステムの優劣を調査した．その結果，MAPPERがわれわれの設定した条件をすべて満たすものであることが判明した．こうして同年11月にMAPPERを決定し，それを稼働させるハードウェアとして，シリーズ2200モデル200の導入を決定した.

6. システムの概要

今回計画したシステムは次の三つである.

1) 生産管理システム
2) 品質管理システム
3) 生産計画シミュレーション

これらのシステムは図1に示すような，五つの情報入出力・加工処理を効果的に組み合わせることによって実現している.

以下に各システムの概要を述べる.

1) 生産管理システム(実績系)……当事業部で生産しているビデオ製品(ビデオデッキおよびムービーカメラ)は,最終工程のシャーシ完成に至るまでの工程は自動化が進んでおり，ほとんど人手を介することはない．しかし，シャーシ完成後の調整・検査・組立・梱包に関しては，人手による作業が行われており，このラインに関する実績を正確に把握するのが，このシステムの基本になっている．生産情報の把握は，シャーシ完成品の投入時および最終の梱包時に，センサで台数を自動的に算出し，MAPPERデータベースに随時反映する仕組みである.

   また，勤怠情報の正確な把握と，それらの情報に基づく稼働管理情報の作成もこのシステムの一部である．勤怠情報は，朝の出勤時に各自がIDカードを用いてハンディターミナル上に入力，これをMAPPERデータベースに随時反映する仕組みである．また，他職場への応援・外出・早退・残業等，通常の勤務以外のケースが発生した場合も，対象となる作業者が自分でハンディターミナルより入力するものである．このとき作業者の負荷の増大を避けるために，事前にわかっている出張等は勤務予定として，一括して報告することも可能にしている.

2) 品質管理システム……従来の品質管理システムはオフコンで処理していたの

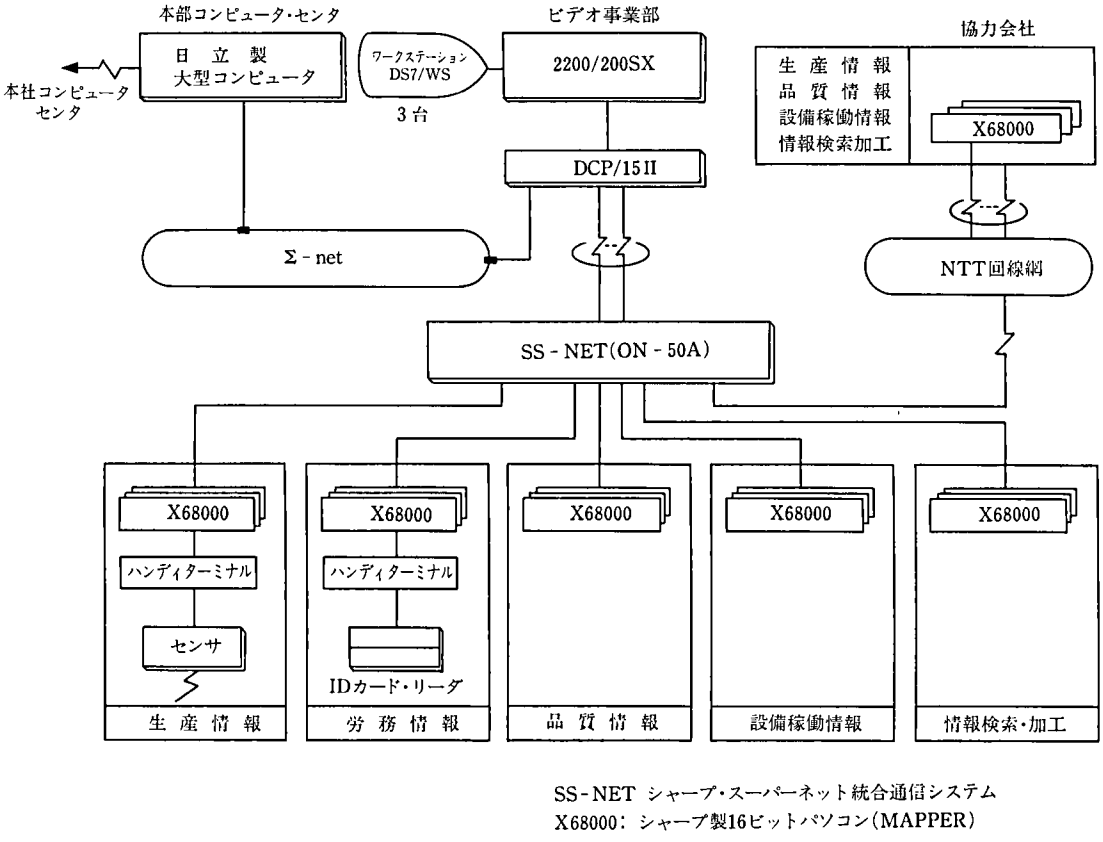


図 1　ビデオ生産情報統合化システム構成図

Fig. 1　Structure of new production control system

で，出力帳表が固定化されていた．そのため，いろいろな検索条件を組み合わせた管理データを出力するのが困難であり，前工程への品質データのフィードバックを的確に行えないケースがあった．

今回の品質管理システムはこの点を解決しており，不良発生データを入力しておけば，どのような検索キーを与えても簡単な会話操作で希望するデータを抽出することが可能となった．またMAPPERをベースとしたQC総合支援パッケージとの連動により，データのグラフ化を行い，品質分析，管理資料作成等に活用している．

3)　日程計画システム(生産計画シミュレーション)……このシステムは，事業本部の大型コンピュータから生産計画情報をネットワークを通して直接受取り，各工程ごとの日程計画を作成し，また実績データを大型コンピュータ側に返送する仕組みである．

まず発行された製品倉入日程に基づきユニット(工程)に展開し，同時に所要量計算を行い，それぞれのユニット(工程)の必要日程表を作成する．

次に作成した必要日程を，ライン割付条件に基づいてラインごとの日程表に展開する．負荷条件の処理を実行する前に，生産の優先順位条件に従って生産順位

を決定し段取負荷を求め，生産上の負荷に加算して負荷工数を算出する．このとき同時に，算出される能力工数と比較して負荷の山崩し等の調整作業を行っている．

生産可能になった日程表に対してリードタイム処理(作業・支給等)を行った上で下位レベルのユニット部品を展開する．一方は支給予定に，もう一方はユニットの必要日程となり，計画システムの入力情報となる．

負荷能力の調整の部分は，自動化するにはあまりにも条件が複雑・多岐にわたるシミュレーション処理であり，人間の判断を多く必要とすることから，MAPPER の会話機能を直接利用して対応している．

7. システムの開発スケジュールと開発体制

システム構築のためのプロジェクトは平成元年１月にスタートした．システム開発に着手したのは２月早々であり，３月末には早くも実績系システムの第一次分が完了し，４月より稼働を開始した(図２)．

システム開発は当社と日本ユニシス(株)との共同開発で進めた．この第一次分は，主に商品の倉入れ実績情報のシステム化であり，売上の基礎データになるものである．

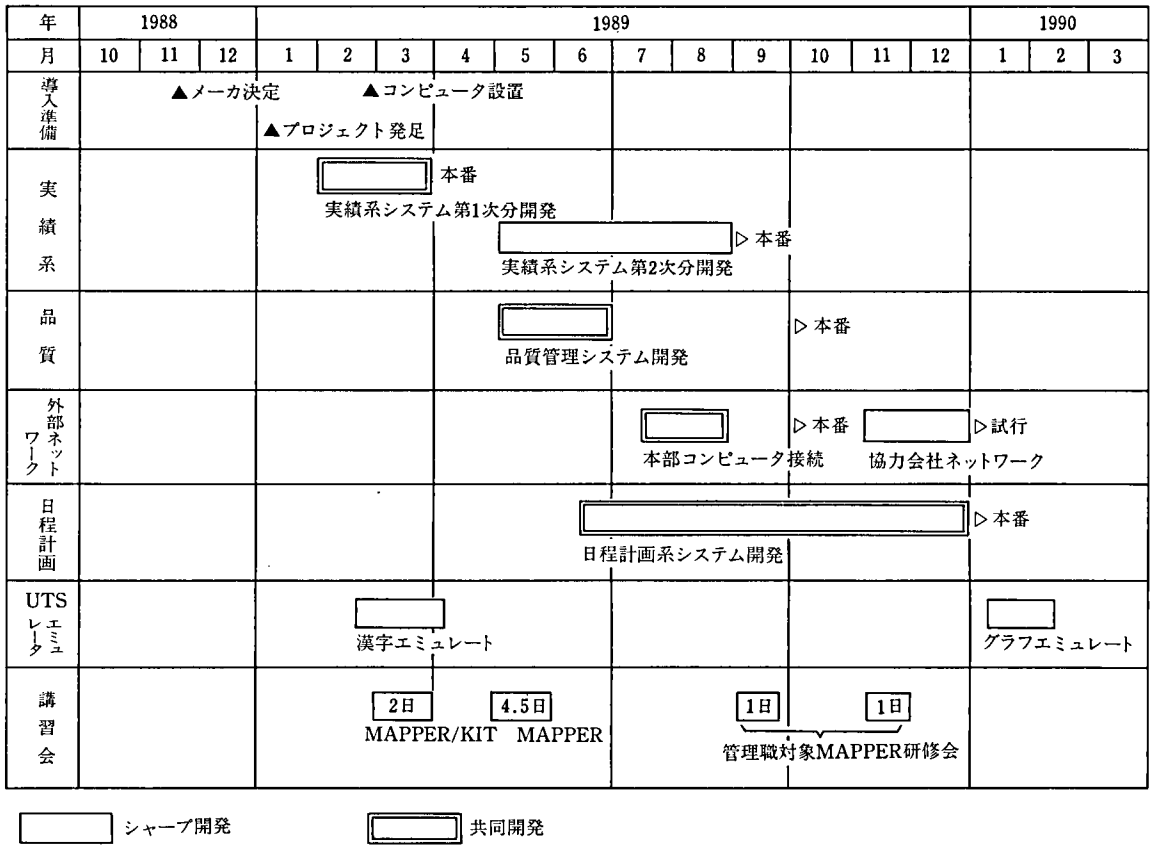


図 2　生産情報統合化システム開発計画

Fig. 2　Development schedule of product information and integrated system

続いて，5月から実績系システムの第二次分の開発に着手した．この部分は労務関係や勤怠システムが中心になっているが，当社が単独で開発を行い，9月から稼働を開始した．

品質管理システムの開発は，このシステムの開発と同時に着手した．このシステムは共同開発で，2か月で開発が終わり10月から本番運用している．

日程計画系システムも共同開発で，品質管理システムの開発終了と同時に開発に着手した．約半年の開発の後，平成2年1月から稼働を開始した．

今回の新システムの重要な要素となるネットワーク・システムは，まず本部コンピュータとの接続を共同開発で進め，平成元年10月から運用を行っている．また協力会社とのネットワーク接続は当社単独で行い，平成元年1月より順次接続を開始した．

前述したように，今回のシステムでは当社のX 68000パソコンをエンドユーザ向けのMAPPER端末として活用しており，平成2年の春からはグラフ機能も使用できるようになっている．

これまでに開発したシステムのボリュームは表1のとおりである．

表1　システム開発ボリューム（ラン本数）
Table 1 A volume of the application systems

平成2年3月末現在

業務システム名	自社開発分(本)	ユニシス開発分(本)	合　計(本)
実績系システム	185	20	205
日程計画システム	80	30	110
品質管理システム (QC支援パッケージ使用)	10	5	15
日立ネットワーク	5*	5*	10*
合　計(本)	280	60	340

＊ COBOL言語を使用している

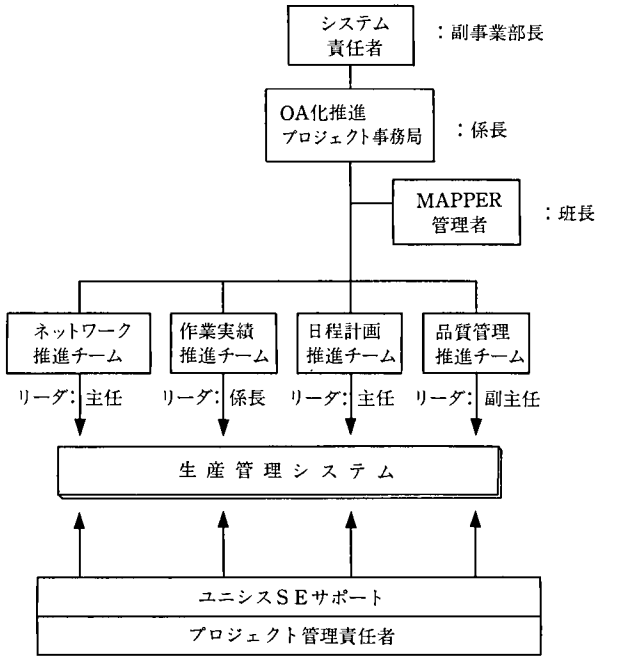
システムの開発に当たっては，プロジェクト・チーム(5名)を編成した．メンバは生産現場の各部署より集めたもので，全員システム開発に携わった経験を持っていなかった．

図3が今回のプロジェクト・チームの体制である．

システム全体の責任者の下にOA化推進プロジェクト事務局を置いた．これは，開発状況の把握と責任者への報告，責任者の意向の各推進チームへの徹底，他部門との調整，推進チームの事務全般，システムのPR，日本ユニシスとの総合窓口，等の役割である．

この事務局の下に，ネットワーク推進チーム，作業実績推進チーム，日程計画推進チーム，および品質管理推進チームの四つを置いた．いずれもMAPPERを中心としたシステムの開発と教育を担当した．またこの他にMAPPER管理者を置き，システム開発・運用の環境整備，システムの管理等を担当させた．

システムの開発を進める一方，MAPPERに関する社内教育にも力を入れた．誰もが利用できるシステムとしてMAPPERを選択したわけで，MAPPERが事業部全体に普及することが，初期の目的達成のためのキーであった．そのためシステムの本格的な稼働を前にした平成元年3月から教育を開始した．まず，MAPPERによるアプリケ



システム責任者
　各推進チームの全体マネジメント
OA化推進プロジェクト事務局
・開発状況の把握，副事業部長への報告
・システム責任者の意向を各推進チームに徹底させる
・他部門との調整を行う
・推進チームの事務全般
・システムの社内PR
・ユニシスとの総合窓口
MAPPER管理者
・システム開発／運用の環境整備，システム管理
推進チーム・リーダ
・チームのまとめ役，開発進捗の管理およびOA化推進プロジェクト事務局への報告(随時)
・OA化推進プロジェクト事務局の意向をチーム内に徹底させる
・利用部門への教育推進
ユニシス SE
・各チームへのシステム支援

図3 開 発 体 制
Fig. 3 Development team

ーション開発支援システムであるMAPPERKITの講習を2日単位で，そしてMAPPERの講習を4.5日単位で実施した．さらに同年9月からは，管理職者を対象にしたMAPPER研修会を実施し，その普及に努めている．

8. お わ り に

システム全体から見て，MAPPERによるデータベースはほとんど揃ったと考えている．各データも端末機から順調に入力されるようになり，それが日々タイムリに更新されて，今後有効に活用できるデータベースになってきている．

日進月歩の管理指標がデータベース化できるようになり，データの加工，その他のデータの活用ができる下地はできあがった．今後はこのデータを利用して，事務処理を機械化し，人手による作業をできるだけ削減していく方向を目指している．そして作業が機械化されて余裕ができた分は，さらに生産性の高い仕事や，付加価値のある仕事に着手できるようにすることが課題である．

今回の新システムが稼働する前は，データの収集に躍起になり，収集が終わるとそれだけで作業が終了しており，そのデータを加工する等といったことは考える余裕もなかった．しかし現在，データの加工・有効活用を考える余裕が生まれたことは，このシステムの効果の一つであろう．

またラインの長はほとんど全員が，端末からデータの検索や必要な報告書の作成ができるようになったことも大きな効果である．

これらの効果を得ることができた要因は，われわれのようなコンピュータ経験のな

いエンドユーザが独自に目的のシステムを開発することを可能にした，MAPPERの使用の容易性と開発生産性の高さであろう．また現在稼働しているシステムを従来のプログラミング言語で開発したとすると，3～4年も必要と見積もられていたが，MAPPERの利用によって開発期間を大幅に短縮できたことも，今回のシステム開発の成功のポイントであると思われる．

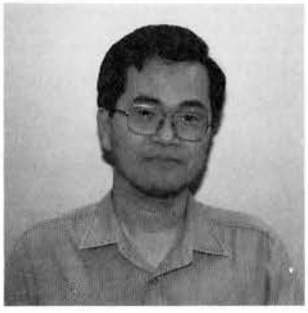
一方，今後もより一層エンドユーザ・コンピューティングを推進し，当事業部に情報の戦略的な活用とそれを基盤とした合理的・積極的な行動が十分に根付くようにすることが長期的な課題である．

執筆者紹介　青　木　好　治（Yoshiharu Aoki）

昭和16年生．39年シャープ(株)入社．テレビ事業部，本社経営企画室，ビデオ事業部生産技術部等を経て，昭和63年11月よりビデオ事業部副事業部長．

菊　池　　豊（Yutaka Kikuchi）

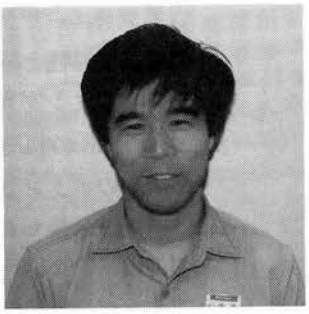
昭和22年生．43年シャープ(株)電子機器事業本部入社．テレビ事業部生産技術部でIE業務担当．58年ビデオ事業部生産技術部でIE業務担当．平成元年1月よりOA化推進プロジェクト業務担当．



斎　藤　孝　夫（Takao Saitou）

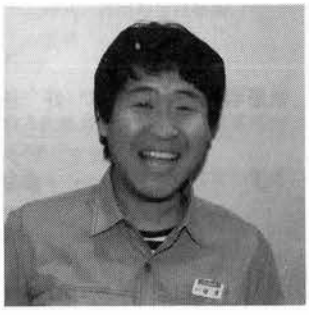
昭和24年生．43年シャープ(株)電子機器事業本部入社．テレビ事業部生産部でCTV生産ライン工程管理，QC業務担当．54年ビデオ事業部第1生産部でQC業務担当．60年生産技術部で製品コスト改善，生産ライン合理化業務担当．平成元年5月よりOA化推進プロジェクト業務担当．

伊　東　　守 (Mamoru Itou)



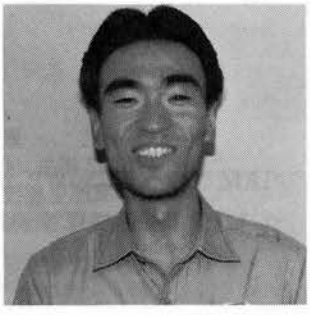
昭和32年生．55年シャープ(株)電子機器事業本部入社．ビデオ事業部第1生産部でVTRメカ受入検査，外製指導業務担当．57年VTRドラム組立ライン工程管理業務担当．62年イギリス工場にてドラム生産指導．平成元年1月よりOA化推進プロジェクト業務担当．

梅　津　長　央 (Nagao Umezu)



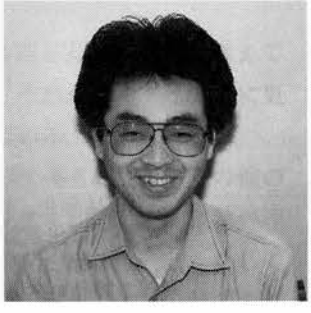
昭和34年生．55年シャープ(株)電子機器事業本部入社．ビデオ事業部第1生産部でVTR組立調整ラインの電気修理業務を経て工程管理業務を担当．平成元年1月よりOA化推進プロジェクト業務担当．

小　林　敬　三 (Keizo Kobayashi)



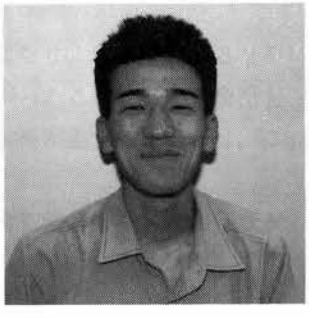
昭和37年生．56年シャープ(株)電子機器事業本部入社．ビデオ事業部第2生産部でVTR組立調整ラインの自動化業務を担当．平成元年1月よりOA化推進プロジェクト業務担当．

小野崎　誠 (Makoto Onozaki)



昭和39年生．58年シャープ(株)電子機器事業本部入社．ビデオ事業部第1生産部で外製品質管理業務担当．60年第1生産部で半田付品質改善プロジェクト業務担当．63年第2生産部でチップ部品装置工程品質管理業務担当．平成元年1月よりOA化推進プロジェクト業務担当．

八木沢　勝　正 (Katsumasa Yagisawa)



昭和41年生．59年シャープ(株)電子機器事業本部入社．ビデオ事業部第2生産部でVTR組立調整ラインの自動化業務を担当．平成元年1月よりOA化推進プロジェクト業務担当．

LSA (LINC Systems Approach) の紹介

An Introduction to the LINC Systems Approach

倉 坪 康 広

要 約 情報システムの適用分野の拡大とともに，ソフトウェア開発のあり方が改めて問われている．ソフトウェア工学をはじめ，いろいろな形で研究・開発が進められ，その成果は徐々にではあるが実践の場で実を結びつつある．

本稿では，日本ユニシス(株)のLINCによるシステム開発方法論であるLINCシステムアプローチと，その概念システムモデル構築技法であるオブジェクト・モデリング技法について紹介する．

Abstract In proportion to the ever-expanding areas of computer systems applications, questions have again arisen as to how software development ought to be. In pursuit of better solutions, various forms of R & D activities, including software engineering, have been under way, and some of the results are beginning to prove themselves useful for practical applications thought the pace is slow.

This paper is intended to bring readers up to date on the Unisys systems design methodology called the LINC Systems Approach and on object modeling, a method for it's conceptual systems modeling.

1. はじめに

LINCは1982年に日本で出荷開始されて以来，600以上の導入実績を持ち，ユーザからも高い評価を得ている．それは，第1にLINCの持つ高生産性ツールとしての機能や設計思想が受け入れられてきたこと，第2に出荷以来，定期的にユーザニーズを反映し時宜を得た機能改善に努めてきたことが高い評価に結び付いているものと確信する．

しかし，ここにきて新たな要請が高まってきた．それはLINCを開発支援ツールとして，より効率的に使用するための統合化されたLINC開発環境の提供への要請である．このような要請に対する解決策として米国ユニシス社のLINC Systems Approach (以下LSA) を包含紹介したい．LSAは，LINCを開発支援ツールとして使用する場合の開発方法論と技法を包含したものである．本稿では，LSAの概略と技法の一つであるLINCオブジェクト・モデリングを中心に述べる．

2. LSA出現の背景

1968年NATOの科学会議にて「ソフトウェア工学」という言葉が提唱され，ソフトウェア危機という言葉も生まれたと言われている．その後，ソフトウェア工学に関する関心は高まる一方で，学術的にも調査・研究が活発化するとともに，その実践の場への技術移転も図られてきた．しかしながら，このような懸命な努力にもかかわらず，その危機感は年々強まる一方である．LINCを取巻く環境においても例外ではない．これまで以上の生産性向上を維持しながら品質を確保していくには，ソフトウェア工学の学術的な場から実践的な場への一層の技術移転が必須である．具体的には，

次に述べるような課題への対応策として，ソフトウェア開発環境の変化を適切にとらえた方法論や技法，またそれらを支援するプロダクトが待望されているという背景がある．

2.1 ソフトウェア産業における課題

1) 属人的なシステム開発の排除……これまでは，設計者を中心としてLINCの持つ設計思想を開発の中に生かして，一応の成果は納めてきた．しかしながら，それらの設計者に蓄積されたノウハウが公開され共有化されることは少なかった．一つには，SEの職人的気質が他人の成果を積極的に受け入れようとすることへの妨げになっていることがあげられる．旧態依然として，属人的なシステム開発の域を脱しきれておらず，開発者の能力に依存する度合いが極めて高い．これは単にソフトウェア開発での生産性向上に対する障壁となっているだけでなく，増加の一途をたどる保守作業においても大きな障壁になっている．他の産業の工業製品に見られるような，ソフトウェアの工業製品化へのアプローチが実践的な見地から求められている．

2) 開発プロジェクトの巨大化・複雑化……これまでのシステム化は省力化・合理化を目指したものが中心であった．システム化の対象業務も，システム化によって比較的効果が発揮しやすい分野から順次手がけられてきた．いかにして大量データを効率的にさばくかに焦点が当てられてきた．ところが，通信技術やOA機器等の最近のめざましい技術の進歩とあいまって，情報システムの適用範囲も部門固有の業務から企業を横断する全社的な業務にまで拡大されつつある．必然的にその開発プロジェクトも巨大化・複雑化してきており開発リスクも大きくなり，その成否が企業にとって重大な関心事となっている．これまで以上に，プロジェクトに関与する人間の間での意志統一，管理するための規範が必要である．

3) 経営環境の変化に柔軟に対応できるシステム構築……最近の情報システムへのニーズは企業戦略そのものを支援する，いわゆる戦略的情報システム，に変化してきている．しかし，既存システムではこのニーズへの対応に限界が見え始めている．開発当初から構築すべき情報システムの全体像を描き，その一部として対象業務の開発が進められてきたわけではなく，個別の問題として業務システムを構築してきたからである．そのため，業務単位にはシステム化による効果は認められるものの，企業的な見地からすると必ずしも有効に機能しているかどうか疑問の余地が残る．また，最近のめまぐるしい経営環境の変化に対応して発生する，新たなニーズを柔軟に吸収するようには作られていない．

4) 高度情報化社会……コンピュータは，社会の中枢を担ういろいろな分野に加速的に浸透しつつある．このことがもたらす弊害としては，ソフトウェアの一部の障害が直接社会全体の機能麻痺につながることがあげられる．従来であれば，障害発生時何とか手作業で対応できえたことも，現在ではもはや不可能に近い．それはとりも直さず，ソフトウェアに対する信頼性へのニーズが高度化してきていることを意味している．ソフトウェアは，一開発者の枠を越えた社会全体の共有財産化しつつあるということである．そのためには，ソフトウェアの品質保証を十分に確保するための考え方が必要である．

2.2 ソフトウェア開発環境の変化

これまで，ソフトウェア工学の成果として構造化技法に代表されるいろいろな方法論や技法が提唱され，徐々にではあるが実践の場に浸透しつつある．それらの方法論は情報システムの企画，要求定義，設計，開発・テストの範囲を支援するものから，ドキュメント管理，プロジェクト管理を支援するものまで広範囲にわたっている．

また，1980年後半から，それらの方法論や技法をコンピュータ上で支援しようとするComputer Aided Software Engineering（以下CASE）ツールが出始めている．その種類も，単に開発工程以下の下流工程を支援するLower CASEツール，要求定義/設計工程といった開発の上流工程を支援するUpper CASEツール，上流工程と下流工程を連携して支援する統合CASEツールまである．さらに，AI技術を応用してシステムの企画工程の支援を対象範囲とするツールも研究・開発に着手している．

このようにソフトウェア工学の成果は，これまでのソフトウェア開発の高品質・高生産性の確保に貢献してきたことは論を待たない．また，これからも大いに期待できる．しかし，より一層のソフトウェアの高品質・高生産性を確保するにはソフトウェアプロダクトの特質を踏まえた方法論や技法が待望される．

3. LSA概説

LSAは，LINCを開発支援ツールとして使用する場合の方法論と技法を包含したものである．すでに，企業分析や情報戦略計画が策定済みの対象業務について，その対象業務を支援する情報システムを構築するために必要な開発ライフサイクルを対象範囲としている．

LSAの開発ライフサイクルは以下の五つのフェーズから構成される．

1) システム調査
2) システム定義
3) システム開発
4) システム導入
5) システム評価

LSAは，これらの各フェーズでの目標，成果物，実施すべき作業をタスク，サブタスクと段階的に階層化しながら網羅的にまとめたガイドラインである．すなわち，一般的に実施すべき開発作業についてのチェックリストである．その内容については各プロジェクトの特性を加味しながらレビューし改善されることを前提としている．同時に，各フェーズで用いる技法についても触れている．これらの方法論や技法は，いずれもLINCによる情報システムの開発とプロジェクト管理に関係した多くのSEの経験の蓄積に基づいたものである．

また，この中で日常使用している言葉や図式表現を用いてビジネスの世界をモデル化するオブジェクト指向の設計技法に基づいた，LINCオブジェクト・モデル（概念オブジェクト・モデル）設計技法を提唱している．そのモデルは，LINC開発環境にそのまま定義できる特徴がある．

3.1 LINC開発環境の統合化

LINCの目標は，長年にわたるデータ処理上の課題を解決するところにあった．それ

は，適切なシステムを，適切な場所に，適切な時期に，適切なコストで構築することである．この達成に向け，LINCの設計思想，開発方法論(LSA)，ソフトウェアプロダクト（LINC）を統合した．

設計思想とは，開発方法論とソフトウェアプロダクトの両方にまたがる基本的な概念を提供するものである．開発方法論とは，組織的なビジネスモデルの定義と，そのモデルを支援する情報システムに変換するための実践的なガイドラインを提供する．ソフトウェアプロダクトは，そのモデルを変換するためのツールである．

3.2 LSAの各フェーズとその概要

ここでは各フェーズの概要について述べる．タスクの詳細やサブタスクについては省略する．

1) システム調査……特定のビジネス領域の内部とその周囲での活動に焦点を当て，現在のビジネスのやり方を理解する．手作業と自動化されている作業の両方について調査する．このフェーズを開始するに際し，すでに企業の情報化計画が立案されており，調査対象のビジネス領域の範囲と目標が定義されていることが前提である．

〔目標〕

- 対象ビジネス領域の内部，またはその周囲で行われている活動の定義
- 情報システムの目標と実現化境界の明確化
- 自動化システムの設計と開発のために必要な要員と資源の識別

〔成果物〕

- 情報システム概要書
- プロジェクト計画書

〔タスク〕

① ビジネス領域の内部や，その周囲で記録されている活動を識別する．
② 情報システムの目標と実現化境界を明確にする．
③ 初期のプロジェクトサイズを見積る．
④ 要員要求を識別する．
⑤ 自源要求を識別する．
⑥ 情報システム概要書とプロジェクト計画書を準備する．
⑦ マネジメントの了解を得る．

2) システム定義……フェーズ1で決定した範囲と目標に従って，フェーズ2では自動化システムを定義する．新システムの概念オブジェクト・モデルを作成することから始め，順次，セキュリティ，バックアップ，リカバリ，ハードウェア，環境ソフトウェア等のシステムの技術的な属性を定義する．外部システムとのインタフェース設計，既存システムから新システムへのコンバージョン方法についてもあわせて設計する．さらに，新システムがユーザに与える影響についても評価する．LINCシステム開発環境支援ツールはこのフェーズを通して，システム定義のため，システム提案に対するドキュメント作成を支援するために利用する．

〔目標〕

- 自動化システムの定義

・インタフェースの設計
・コンバージョン方法の設計
・ユーザ環境への影響の評価

〔成果物〕

・システム提案書
・プロトタイプ
・開発計画書

〔タスク〕

① ドキュメントを収集する.
② 概念システムモデル (LINC オブジェクト・モデル) を作成する.
③ 技術的属性を決定する.
④ 標準を設定する.
⑤ LINC 開発環境に自動化システムを定義する.
⑥ プロトタイプを開発する.
⑦ コンバージョン方法を設計する.
⑧ ユーザ環境への影響を評価する.
⑨ システム開発フェーズの計画を立案する.
⑩ システム提案書と開発計画書を作成する.
⑪ ユーザの同意を得る.

3) システム開発……このフェーズはフェーズ 2 の続きで, 情報システムの開発に焦点を合わせる.

〔目標〕

・システム開発の完了
・ドキュメント化の準備
・テストの完了
・トレーニングの完了

〔成果物〕

・導入可能な LINC システム
・コンバージョン・ソフトウェア
・ドキュメント
・テスト結果
・受け入れ報告書 (オプション)
・導入計画書

〔タスク〕

① LINC で ISPEC とレポートを開発する.
② インタフェース・ソフトウェアを開発する.
③ コンバージョン・ソフトウェアを開発する.
④ 運用上必要なジョブ制御ソフトウェアを開発する.
⑤ ユーザ用ドキュメントを準備する.
⑥ 受け入れテストを指導する.

⑦ トレーニングを指導する.

⑧ システム導入フェーズの計画を立案する.

4) システム導入……このフェーズでは，受け入れテストが完了しているシステムを実稼働環境に導入することである．システムをカットオーバするためのハードウェアとソフトウェアの設置やデータのコンバージョンが含まれる.

〔目標〕

・実稼働ソフトウェアの設置

・新システムの利用開始

〔タスク〕

① ハードウェア，環境ソフトウェア，通信ネットワーク（オプション）を設置する.

② LINC システムを設置する.

③ データコンバージョンを行い新システムをカットオーバする.

5) システム評価……ある一定期間，稼働した後のシステムを評価する．その目的はプロジェクトの成果を測定し，次期システムの開発に向けての準備をすることにある.

〔目標〕

・システム運用上の改善点の識別

・ハードウェア，ソフトウェアの利用状況の確認

・システムの改善点の識別

・開発生産性の決定

〔成果物〕

・システム評価報告書

〔タスク〕

① システム運用についてレビューする.

② 本稼働システムの測定基準を決定する.

③ ユーザ環境への影響をレビューする.

④ 改善要求をドキュメント化する.

⑤ 開発測定基準を決定する.

⑥ システム評価報告書を作成する.

⑦ レビューを指導する.

4. LINC オブジェクト・モデリング技法

この技法はフェーズ 2 の概念システムモデルを作成するための技法である.

LINC オブジェクト・モデリング技法とは自動化システムを構成する入力画面，問い合わせ画面，レポート，インタフェース等のオブジェクトについて機能的に図式表現した概念システムモデルである．それぞれのオブジェクトは関連データ，ビジネス規則，プロファイルの定義を記述した支援ドキュメントを含む．概念システムモデルはそのまま LINC 開発環境に定義できる（図 1）.

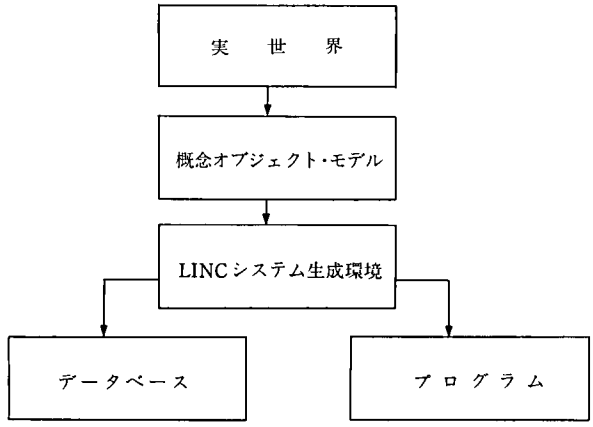


図 1　LINC オブジェクト・モデリング
Fig. 1　LINC object modeling

4.1　LINC の考え方

LINC の設計思想上の大きな特徴として二つあげられる．第一は，ビジネス活動をイベント，コンポーネント，プロファイルの三つの要素で表現しようとする考え方である．それぞれの特徴は以下の通りである．

1)　イベントは日常のビジネスの活動によって発生する情報である(たとえば，受注・購買・入金・支払・在庫調整の情報)．

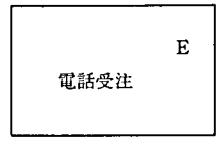
2)　コンポーネントとは企業活動を支援するために不可欠な情報である(たとえば，顧客・商品・倉庫・営業所といった情報)．

3)　プロファイルとは利用者のさまざまなニーズに合わせた視点（ビュー）である(たとえば，売上順の顧客情報・売れ筋商品の把握を提供する視点)．

第二は，従来の方法では，入出力画面，データベース構造，処理ロジックはそれぞれ分離したものとして捉えていたが，LINC ではイベント，コンポーネントをそれぞれ一つのオブジェクトとして捉え，各オブジェクトごとに入出力画面，処理ロジック，データベース構造をカプセル化するオブジェクト指向の基本となる概念を採用している点である．オブジェクトとしては各活動を構成するイベント，企業活動を支援するコンポーネント，問い合わせ画面，レポート，要約情報を格納するメモコンポーネント等がある．

4.2　表記法について

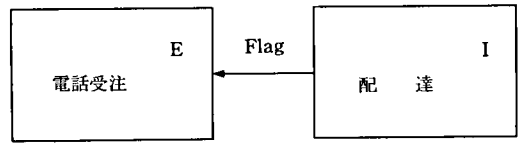
1)　オブジェクトの表記……各オブジェクトは四角形の中に，オブジェクトの型とオブジェクト名を記述する．たとえば，電話受注イベントを表現すると下図のようになる．



同様にして，他のオブジェクトは以下のように表現する．

・コンポーネントオブジェクト　　“C”
・問い合わせオブジェクト　　“I”
・要約情報オブジェクト　　“M”
・レポートについては底辺部分を曲線にし，“R”を記入する．

2) 他のオブジェクトとの関連づけ……情報の追加，または更新を必要とする場合は矢印の付いた直線でオブジェクト同士を関連付け，Auto または Flag を直線の横に記述する．矢印は情報の追加，または更新をする対象のオブジェクトを指す．



例：電話注文に対して，配達完了のマークをつける

このほか Recall（他のオブジェクトの呼出）を明示的に表現したければ Auto, Flag と同じように表現する．

3) 支援ドキュメント……各オブジェクトにはデータ項目，ビジネス規則，プロファイルの定義についてドキュメント化する．

4.3 LINC オブジェクト・モデルの構築

LINC オブジェクト・モデルは，以下のようなプロセスを経て詳細で完全なモデルとなる．

1) 初期モデルの作成
2) イベントの合理化
3) レポートと問い合わせの考察
4) コンポーネントのレビュー
5) パフォーマンスの考察
6) 機密保護の考察
7) インタフェースの考察
8) 整理と保存の考察
9) 単一データベースの仮定のレビュー
10) 単一機械装置の仮定のレビュー
11) システム化の目標のレビュー

4.4 各プロセスの概略

1) 初期モデルの作成……自動化システムは単一データベースで単一機械装置で実現されるものと仮定する．フェーズ1の調査フェーズで定義された活動（表1）は入出力使用のイベントとして，一つのオブジェクトを表現する．それぞれのオブジェクトは，モデルの中に名前の付いた四角形で表現し，“E”（イベント）の文字を右上の角に記入する（図2）．

表1　ビジネス活動分析
Table 1 Business activity analysis

機能領域	記録されている活動
販売	電話注文
	郵便注文
在庫	書籍の入庫
	書籍の配達
	在庫調整
	顧客からの返品
	出版社への返品
経理	出版社への発注
	現金受領
	クレジット会社からの現金受領
	クレジットノート
	仕訳入力
	出版社への送り状
	支払の承認
	信用限度の確認
	出版社への支払
顧客関連	注文の変更

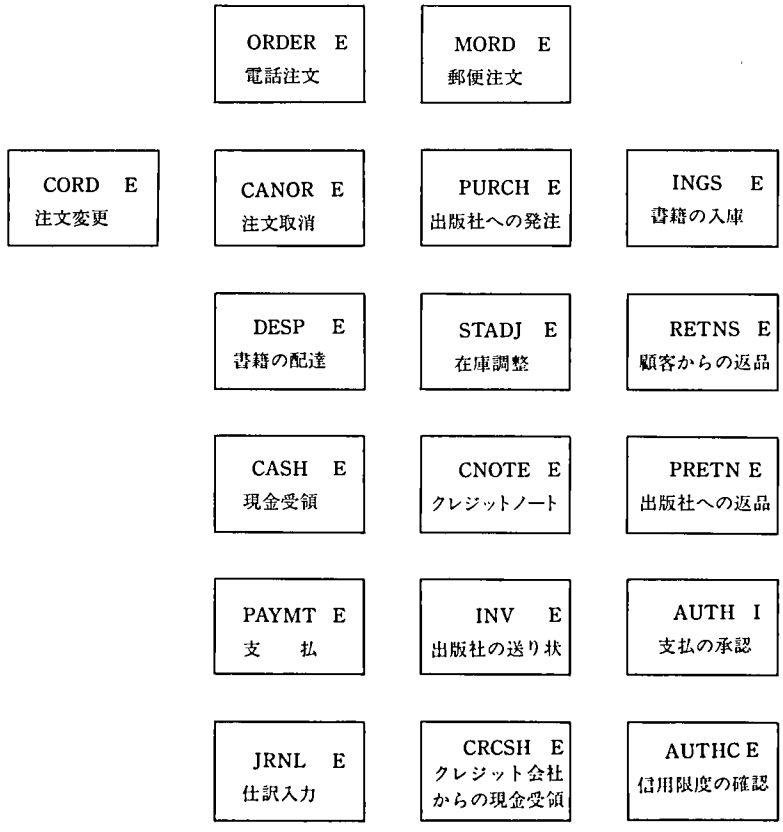


図 2　LINC オブジェクト・モデル：初期版

Fig. 2　LINC object model: first cut

それぞれのオブジェクトを詳細に分析し，以下の内容を決定する．

- 機能（注釈）
- 使用方法（入力・出力・入出力）
- トランザクションの件数と有用期間
- データ項目とその編集属性
- 画面の移り変わり（前の画面・次の画面）
- 妥当性検証のための編集方法とエラーまたは警告メッセージ
- ビジネス規則

これらの情報は，後で LINC で定義するためにドキュメント化する．この分析過程で妥当性検証のために必要なコンポーネントが識別される．右上の角に“C”の文字を入れた名前の付いた四角形を書いてこれらを表現し，オブジェクト・モデルを更新する．編集や処理で使われるプロファイルについては，後に LINC で定義するためにドキュメント化する．

2)　イベントの合理化……イベントの物理データベースの実現方法は，すべてイベントのユニークなデータ項目が一度だけ現われるように同一の物理構造に格納する方法を採っている．これにより，プロファイルは複数のイベントにまたがったビューを提供できる．しかし，このイベントの実現方法はイベント内で使用する

データ項目の命名方法が，ディスクの使用効率やプロファイルの効率的な使用の両面に対して大きな影響を及ぼす可能性がある．つまり，不必要にユニークな名前を付けると，イベントの物理的な容量の増大につながる．さらに，本質的に同じ意味を持ったデータ項目に異なった名前を付けるとプロファイルの共有が困難となる．したがって，イベントのデータ項目名はイベント間で最大限同一のデータ名が使えるように合理化すべきである．また，データ名はイベント名で修飾されることに着目すべきである．

3) レポートと問い合わせの検討……ユーザが要求する出力情報（問い合わせやレポート）について一つのレポートオブジェクト，または問い合わせオブジェクトを設定する．四角形の中央にオブジェクト名を記入し，右上の角に，レポートであれば“R”，問い合わせであれば“I”を記入する．

それぞれのオブジェクトについて以下の内容を決定する．

・機能（注釈）
・データ項目
・データの取り出し方法，計算で使用するデータの源を含む
・画面の移り変わり（前の画面・次の画面）
・セキュリティの制限

これらの情報は，後でLINCで定義するためにドキュメント化する．この分析中に，データを取り出すために使うプロファイルが識別される．

4) コンポーネントのレビュー……妥当性検証のためのオブジェクト（コンポーネント）は，活動オブジェクトと問い合わせオブジェクトが定義された時に認識される．右上の角にCを記入した四角形で表現する．

以下の内容を決定するために，それぞれのオブジェクトを詳細に分析する．

・機能（注釈）
・トランザクションの件数と有用期間
・データ項目と編集属性
・妥当性検証のための編集方法とエラーや警告メッセージ
・画面の移り変わり（前の画面・次の画面）
・ビジネス規則

これらの情報は，後でLINCで定義するためにドキュメント化する．編集や処理で必要となるプロファイルが認識される．

5) パフォーマンスの考察……LINCオブジェクト・モデルへの影響について，以下の観点からレビューする．

・イベントやコンポーネントに対するプロファイルの数
・LINCサブシステムの利用
・残高の算出方法とその格納
・I/Oの発生頻度が高い指令（Auto. Entry, Look, Up, Determine, Flag）

必要に応じてLINCオブジェクト・モデルを更新する．

6) セキュリティの考察……LINCの持っている自動機密保護機構をレビューし，使用の可否を決定する．もし，他の機密保護機構が必要な場合には，それを支援

するためのオブジェクトを定義する.

7) インタフェースの考察……あるオブジェクトが他のオブジェクトに対して更新または情報の追加を必要とする場合, Flag または Auto と記入した矢印で示す. 外部システムとのインタフェースが必要な場合も, 適当な ISPEC タイプでそれらの関連を示す.

8) 統合と保存……イベント・オブジェクトに含まれる詳細情報の統合について, 以下の観点から検討する.

- 詳細データはどのくらいの期間, オンラインで必要になるか.
- 詳細データのどのような要約情報がオンラインで必要とされるか.
- 要約情報を取り扱うため, どのような追加 ISPEC が必要か.
- 統合はどのくらいの頻度で発生するか.
- 一度にといれだけのデータを統合するか.
- ユーザがそのシステムをアクセスしている間に統合できるようにすべきか.

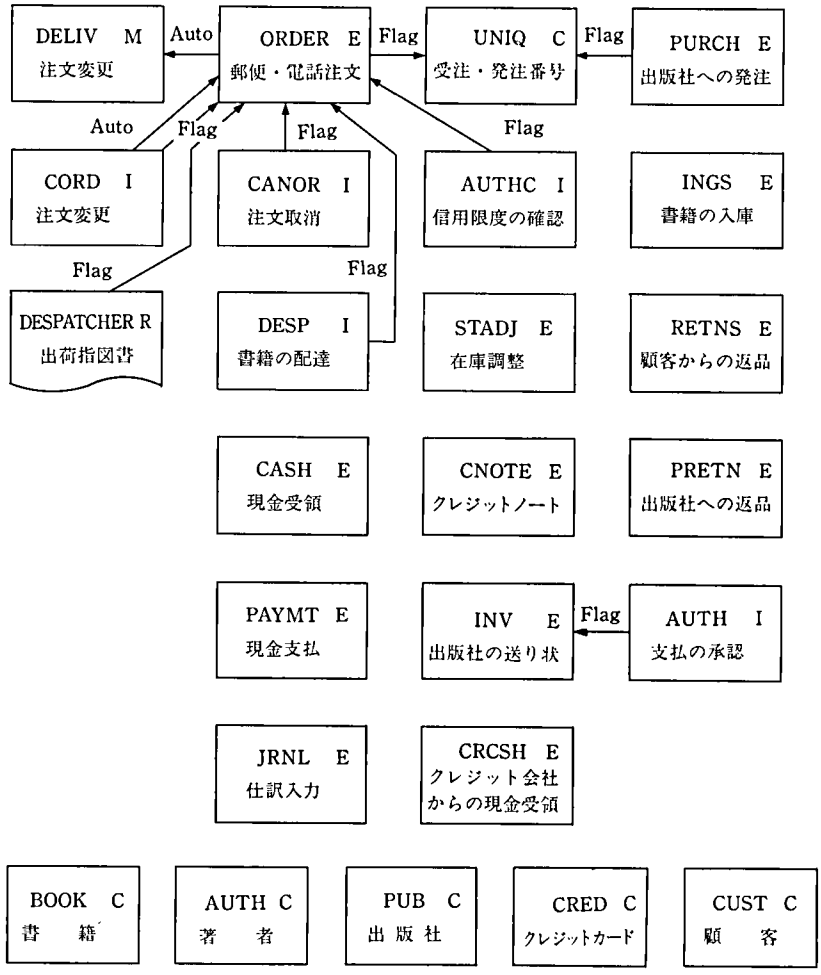


図 3　LINC オブジェクト・モデル：II 版

Fig. 3　LINC object model: eleventh cut

表2　イベント・データ合理化後のマトリックス表
Table 2 Event data item after rationalization

Event Data Item	LE	ED	DE	ORDER	PURCH	INGDS	STADJ	RETNS	PRETN	CNOTE	INV	CASH	CRCSH	PAYMT	JRNL
TRANDATE	5	N		X	X	X	X	X	X	X	X	X	X	X	X
ORDERNO	6	N		X	X	X		X	X	X	X				
CUSTOMER	20	A		X				X		X		X	X		X
ISBN	10	N		X	X	X	X	X	X		X				
PUBLISHER	30	A		X	X	X		X	X		X			X	
PAPER—HARD	1	A		X	X	X	X	X	X		X				
CREDITCARD	1	A		X											
CREDITNO	15	N		X									X		
QUANTITY	5	+		X	X	X	X	X	X		X				
AMOUNT	7	+	2	X	X	X	X	X	X	X	X	X	X	X	X
BACKORD	1	A		X											
ORDERTYPE	1	A		X											
DESPATCHED	5	N		X											
CANCEL	1	A		X											
AUTHORITY	3	A		X						X	X				

要約情報のためのメモコンポーネントを識別する．四角形の右上の角にMを記入したオブジェクトを追加する．

9) インテリグリティの考察……LINCインテグリティ（保全/非保全）オプションを使用するかどうかを決定する．

10) 単一データベースの仮定のレビュー……LINCオブジェクト・モデルをレビューし，単一または複数のデータベースのどちらが適切かを決定する．複数のデータベースが必要な場合には，それぞれのデータベースに含まれるオブジェクトを決定する．これらの新しい導入境界を区別し，新システムとのインタフェースを識別する．

11) 単一機械装置の仮定のレビュー……LINCオブジェクト・モデルをレビューし複数の機械が必要かどうかを決定する．

12) システムの目標のレビュー……目的達成を保証するため，当初のシステム化の目標に照らし合わせ，LINCオブジェクト・モデルをレビューする．

図3は以上のプロセスを繰り返して作成された，LINCオブジェクト・モデル，そして表2はイベント・データ合理化後のマトリックス表の例である．

5. おわりに

以上，LSAとオブジェクト・モデリング技法の概要について紹介してきた．今後の参考になれば幸いである．ソフトウェア工学の成果として各種の方法論や技法，またそれらを支援するプロダクトが出現し，実務面に応用された効果が発揮されつつあるのは事実であるが，その有効性については未知数である．しばらくは紙や鉛筆をツー

ルとしてソフトウェア開発せざるを得ない状況が続きそうである．しかし，常に工学的アプローチに注意を傾けながら積極的に実践に活かそうとする心構え，また方法論や技法の吸収に向けての基盤整備が急務であると考える．

参考文献
［1］ R. C. Goyette, LINC Systems Approach, 米国ユニシス社, 1989.
［2］ T. DeMarco, ソフトウェア開発プロジェクト技法, 近代科学社, 1987.
［3］ J. マーチン, C. マックルーア, ソフトウェア構造化技法, 近代科学社, 1986.
［4］ Edward Yourdon, 構造化技法によるソフトウェア開発, 日経マグロウヒル社, 1987.
［5］ 有沢誠, ソフトウェア工学, 岩波書店, 1989.
［6］ 宮本勲, ソフトウェアエンジニアリング；現状と展望, TBS 出版社, 1982.
［7］ 山本欣子, ソフトウェアの知識, 日経文庫, 1984.
［8］ 日本情報処理開発協会編, 情報化白書, コンピュータエージ社, 1990.
［9］ システム監査基準書, 日本情報処理開発協会, 1985.
［10］ 一貫支援の基盤を固め部品化技術を核に発展するシステム開発支援ツール, 日経コンピュータ, 1990.3.12.

執筆者紹介　倉　坪　康　広（Yasuhiro Kuratsubo）
昭和48年日本ユニシス(株)入社．主に流通業のユーザ・アプリケーション・システムの設計・開発，60年よりLINCを中心としての適用サービスに従事．現在，システム技術本部4GL技術部所属．

拡大を続けるMAPPERの世界

The Ever-expanding MAPPER World

柳　沢　　勝

要　約　MAPPERが世に現れてから約16年の月日が過ぎようとしている．この間，MAPPERはいろいろと豊富な機能を追加してきた．MAPPER出現の当時と比べて，社会環境は著しく変化している．企業間の競争はますます激しくなり，その中で情報は資源としての地位を確立してきた．そして，企業を存続させるためには，情報を正しく管理することが必要になってきた．

また，エレクトロニクスと通信の技術の進歩は，コンピュータを小型で高性能にしかつ安価にした．このことによりコンピュータは，今までコンピュータを使用したことのない人々の中にまで浸透してきた．このように変化する環境の中でMAPPER自身もさらに進化する必要がある．

本稿では，MAPPERが今後どのような進化を遂げようとしているかを述べている．

Abstract　It has been almost sixteen years since MAPPER was first placed on the market place. In the meantime, MAPPER has been making progress with a wide range of powerful functions added one after another in response to user requirements. Compared with the days when MAPPER became available, we now see remarkable changes in social environments. The tougher the business competition is getting, the more important and higher the position of information as a business resource has grown. It has also become necessary to place information under good management for business survival.

In addition, technological progress in electronics and communications has made computers smaller in size, higher in performance and lower in price, thus enabling computers to make inroads into even the hands of people who have never used them before. To respond to the environments that are checking like this the MAPPER shell also requires its own further enhancements.

This paper describes what future enhancements are planned to for MAPPER.

1. はじめに

人が発明した道具の中で，有用な物の一つにコンピュータがある．飛行機や自動車等のように有用な物は他にもあるが，コンピュータほど短期間のうちに進化し，社会に大きな影響を与えた物は今までになかった．しかも，コンピュータは今現在も急速に進化し続けている．

今日コンピュータは，われわれの生活の中に深く根を下していて，もはや必要不可欠な存在になっている．広く社会では，金融・製造・流通・サービス・行政等のあらゆる分野でコンピュータが利用されている．ひとたびコンピュータが停止すると，われわれの社会生活が著しく混乱することは，金融機関でのコンピュータ障害の例を挙げるまでもなく明らかである．われわれはコンピュータの恩恵を受けているとともに，ある種の束縛を受けているといっても過言ではない．

コンピュータは，便利であるが利用する上で問題もある．コンピュータを利用する

ためには，それなりの専門知識を必要とすることである．この専門知識を修得するためには，ある程度の訓練期間と経験が必要である．コンピュータが進化しその性能が上がると，さまざまな業務をコンピュータで処理する要求が利用者から上がってくる．専門家を促成するわけにはいかないので，急増する要求をすべて満足するためには，必然的に専門家の数が不足してしまう．そして，満たされない要求が積み残される．いわゆるバックログの増加である．この状態では，コンピュータの専門家と利用者の双方が欲求不満に陥る．慢性的なバックログを解決するために，ソフトウェアを部品化して再利用するとか，流通ソフトウェアを利用するとかが，提唱されているがどれも決定的な効果をあげているとは言えない．比較的効果のある方法の一つとして，ニーズ（業務）を最も理解している利用者自身に業務をシステム化してもらう方法（エンドユーザ・コンピューティング）がある．そのためには，利用者に彼ら自身が簡単にコンピュータを使い，アプリケーション・システム（業務システム）を構築できる環境を提供することが必要である．

MAPPERは，そのような環境を利用者に提供している数少ないソフトウェアである．しかしながら，MAPPERは初めからそのような環境を利用者に提供するために開発されたのではなかった．初期のMAPPER (Maintaining Preparing and Producing Executive Report) は，その名前のとおり計算機能を持ったレポート作成ツールであった．MAPPERが単なるツールから進化し，利用者にアプリケーション・システム構築の環境を提供できるようになったのは，コンピュータの進化と共に発生するさまざまな要求をタイムリに捕え，それに的確に応えた結果である．

前述したようにコンピュータは，これからも今まで以上のスピードで進化するであろう．本稿では，MAPPERがコンピュータの進化にともなって，これから新しく発生する要求にどのように対応しようとしているかについて述べる．

2. コンピュータを取り巻く環境の変化

コンピュータが発明されてから約半世紀が過ぎようとしている．この間にコンピュータは，非常な勢いで進化してきた．進化の勢いは収まるどころかさらに加速している．コンピュータに対してこれからどのような要求が発生するかを考える前に，現在コンピュータを取り巻く環境にどんな変化が起こっているかを述べ，その変化がどのような要求を導き出すかを考えてみたい．

2.1 エレクトロニクスの技術の進歩

真空管からトランジスタ，トランジスタからIC (Integrated Circuit), LSI (Large Scale Integration), VLSI (Very Large Scale Integration) へとエレクトロニクスの技術は，日々目ざましい進歩を遂げている．これらの技術の進歩により，コンピュータは小型になり処理速度が増加し性能も良くなっている．その一方で，価格はますます安くなっている．コンピュータの小型化・高性能化と低価格化は，コンピュータをわれわれの身近なものにしている．大企業では近い将来に，部門で一台のコンピュータはもちろんのこと，一人一台のパーソナル・コンピュータが導入されると予想される．

2.2 通信技術の進歩

光ファイバの実用化や衛星通信等の通信技術の進歩により，大量のデータを高速で安価に運ぶことができるようになった．このことにより，コンピュータは遠隔地と高速で接続することが可能になり，高度なネットワークの構築ができるようになってきた．

2.3 メディアの変化

オンライン処理の出現初期には，コンピュータで取り扱うデータの種類は，文字データだけであった．やがて，ファクシミリ，光ディスク，画像処理機器，音声処理機器等が出現し，さらに通信技術の進歩により大量データを高速に送れるようになると，新しい機器をコンピュータで利用するために文字以外のデータを取り扱うことが必要になってきた．

2.4 企業環境の変化

社会は非常なスピードで変化し，その先行きも不透明になっている．また，規制の緩和や諸外国からの市場の解放要求により，企業間の競争は激化している．このことは一方ではビジネスチャンスの増大にもなっている．このような環境の中で，企業が生き残るためには，膨大な情報の中から必要な情報をタイムリに選別し経営に生かすことが不可欠である．

さらに為替レート，労働コスト，貿易摩擦等の経済的要因により拠点を海外に求める企業も増加している．このような場合，時差・距離に無関係に世界中から 24 時間体制でグローバルな情報を収集する必要がある．

企業は急速に変化する環境に対応するために，柔軟性のあるシステムを構築し貴重な情報を管理することが必要である．

2.5 利用形態の変化

コンピュータ出現初期の主な利用形態は，大学・研究機関における科学技術計算であり，企業における事務計算であった．これらの利用形態は，単に人間の行う計算処理を高速化したものであり，その目的は省力化にあった．その後，データ処理を連結して一連の流れとして特定の業務をシステム化するようになった．さらにデータが蓄積されてくると，それらのデータを分析し経営のための有効な情報を出力する高度な利用形態に発展した．これらのシステムから出力された情報は，経営のための意思決定に使われるようになり，最近ではさらに発展した戦略的情報システム構築の必要性が叫ばれている．利用形態は，単純な事務処理から経営に携わる戦略的情報処理に変化している．

2.6 利用者の増加

バックログの増加にともなって，業務のシステム化が実現するまでの時間が長くなってきた．このため，利用者の中に自分達の手で業務をシステム化したい要求が出てきた．また，企業経営者自身も社会の変化に対応するためさまざまなデータをいろいろな角度から分析する必要に迫られている．これをいちいち情報システム部門に命じていたのでは，ビジネスチャンスを失ってしまいかねない．これを防ぐためには，自分自身でコンピュータを操作することが必要になる．すなわち，コンピュータの利用者は限られた専門家から利用者自身へと広がってきた．安価なコンピュータが市場に

出回るようになってきたこともこの傾向に拍車をかけている．

2.7 処理形態の変化

初期のコンピュータは高価なものであり，その利用技術もそれほど高度でなく，システム化する業務も限られたものであった．したがって，少数のコンピュータで何もかも集中的に処理するのが一般的であった．また，メーカも汎用機に主力を注ぎトップエンド・マシンの規模がメーカの優位性を示していた．業務量が拡大するに従い，少数のコンピュータに処理を集中するやり方に次のような不具合が現れてきた．

1) システムの管理は一元的にできるが危険も集中してしまい，システムの信頼性を低下させる．

2) コンピュータの処理スピードが業務量に追いつかなくなり，頻繁なグレードアップが必要になる．

3) 変化する要求に対応してアプリケーションを改修するには，柔軟性に欠ける．

近年，安価で高性能な汎用コンピュータ，部門コンピュータやパーソナル・コンピュータの出現と通信技術の進歩等により，ネットワークを介して柔軟なシステムを構築する傾向が見られる．

3. MAPPERの対応

今まで述べてきたコンピュータを取り巻く環境の変化を分析すると，各変化に対応して次のキーワードが考えられる．

エレクトロニクス技術の進歩	小型で高性能なコンピュータ
	安価なコンピュータ
	一人一台のコンピュータ
通信技術の進歩	高度なネットワーク構築
メディアの変化	多様な情報表現
企業環境の変化	戦略情報システム
	グローバル化
利用形態の変化	事務処理からの脱却
	戦略情報システム
利用者の増加	エンドユーザ・コンピューティング
	優れた利用者インタフェース
	(MMI-Man Machine Interface)
	一人一台のコンピュータ
処理形態の変化	分散
	ネットワークによる結合
	階層別利用

これらのキーワードから導き出される世界は，

1) 小型で高性能なパーソナル・コンピュータが市場に登場し，一人一台のパーソ

ナル・コンピュータが当たり前になる.

2) 今までコンピュータを使用しなかった人が使い始め,利用者が爆発的に増える.

3) 誰にでも使用できる使いやすさが求められる.

4) 分散されたコンピュータ同士の接続が今まで以上に重要になる.

5) 業務によりパーソナル・コンピュータ,部門コンピュータ,基幹コンピュータを使い分けるようになる.

6) 分散されたコンピュータ間でのデータ交換が時間と地域に関係なく行われる.

7) 新しいコンピュータ機器が次々に市場に登場して,それを利用するようになる.

等のようなものであろう.近未来に出現するであろうこのような状況に対応するためMAPPERは,いくつかの計画を用意している.これらの計画の方向は,利用者のためのMAPPERであり,ネットワークのMAPPERである.次にMAPPERの計画が具体的にどのようなものであるのかを述べる.

3.1 利用者インタフェースの充実

コンピュータを初めて使用する人達が増える中で,それらの人達にMAPPERを使ってもらうためには,使いやすいことが必要である.そこで,多くの利用者に対しては,今までのインタフェースの一貫性を保ちながら,新しくやさしいインタフェースを提供するための一つのステップとして,"Look and Feel"の概念をMAPPERへ取り入れようとしている.

この機能によって利用者は,MAPPERのファンクション名を覚えることなく,MAPPERの画面を見ながら必要な処理を指示することが可能になる.利用者は,MAPPERのファンクションを実行する場合にファンクション・バー(機能見出し)を使用できるようになる.ファンクション・バーに任意のファンクションを割り当て,そのファンクション・バーに対応するファンクション・キーを押すことにより,該当ファンクションを呼び出すことが可能になる.また,親しみやすいMAPPERのロゴを任意に定義することができるようになる.アイドル・ロゴ表示時に,F1キー(サインオンに対応)を押すとあらかじめ登録しておいた初心者用の使用者識別名でサインオンが行われ,サインオン・ロゴが表示される.

図1は,ロゴの表示例である.(a)がアイドル・ロゴ,(b)がサインオン・ロゴである.

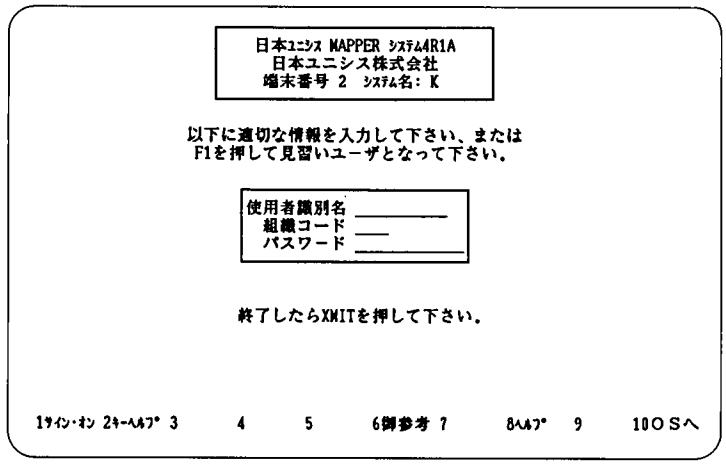
つづけてF1キー(レポートに対応)を押してレポートを表示してみよう.F1キーを押すとドロワ(タイプ)の一覧表が図2のように表示される.

カーソル移動キー(↓)により,ドロワ(タイプ)を選択し送信キーを押すとレポートの一覧表が図3のように表示される.

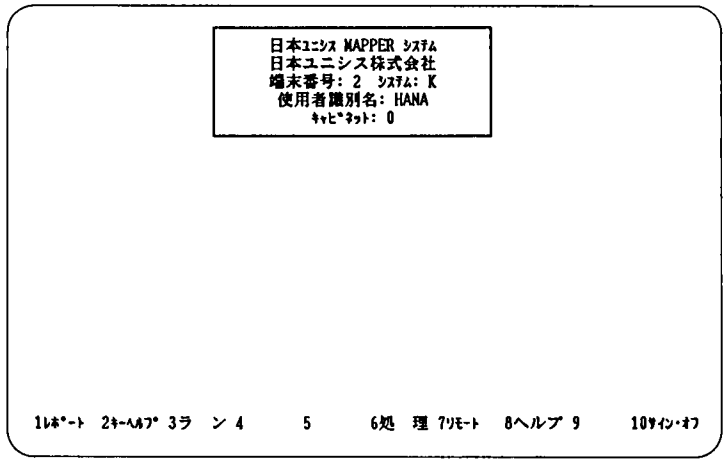
さらにカーソル移動キー(↓)により,レポートを選択し送信キーを押せば,図4のようにレポートが表示される.

F6キーを押すことにより処理の一覧が図5のように表示され,MAPPER利用者は任意の処理を選択することができる.

このようにファンクション・キーとカーソル移動キーにより,必要な処理を呼び出すことができる.しかし,このやり方は初心者には親しみやすいが,MAPPERの操作に慣れた利用者にはかえって煩雑な操作になってしまう.そこで切り換えにより,従



（a） アイドル・ロゴ



（b） サインオン・ロゴ

図 1　ロゴの表示例

Fig. 1　Sample display of logo

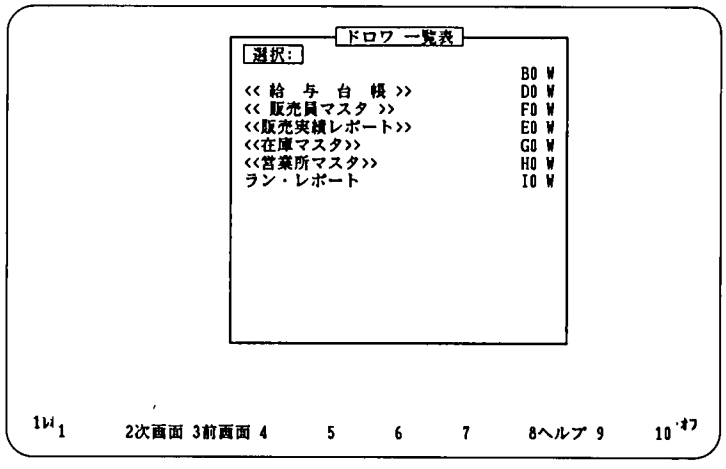


図 2　ドロワ（タイプ）の一覧表の表示例

Fig. 2　Sample display of drawer summary table

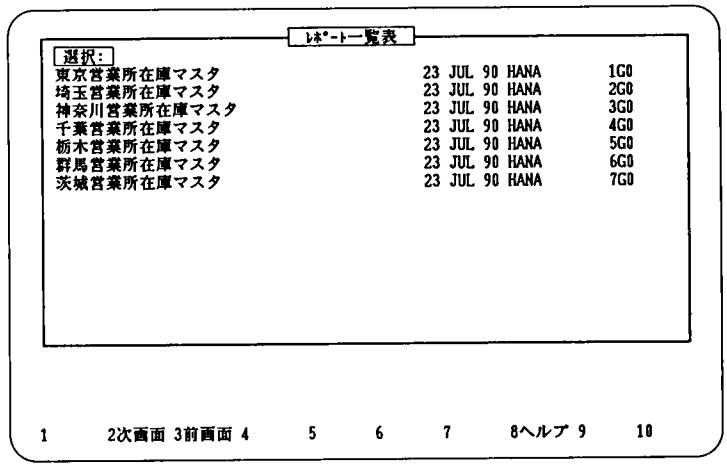


図 3 レポートの一覧表の表示例

Fig. 3 Sample display of report summary table

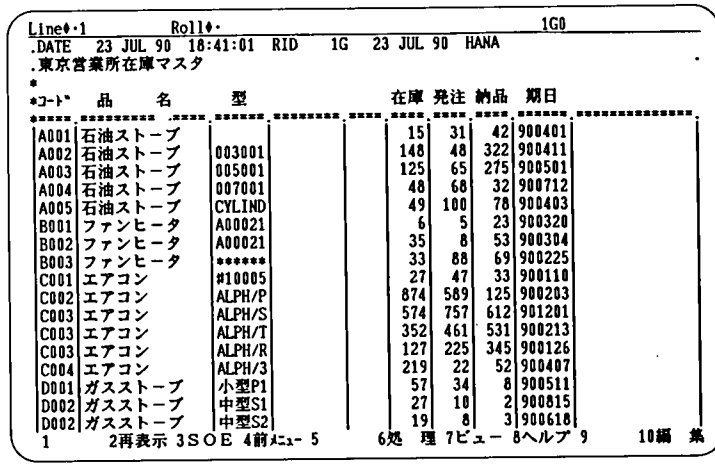


図 4 レポートの表示例

Fig. 4 Sample display of report

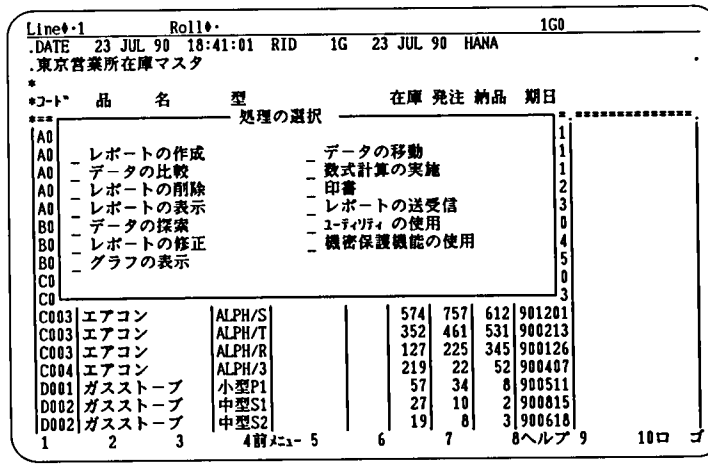


図 5 処理の一覧の表示例

Fig. 5 Sample display of function summary table

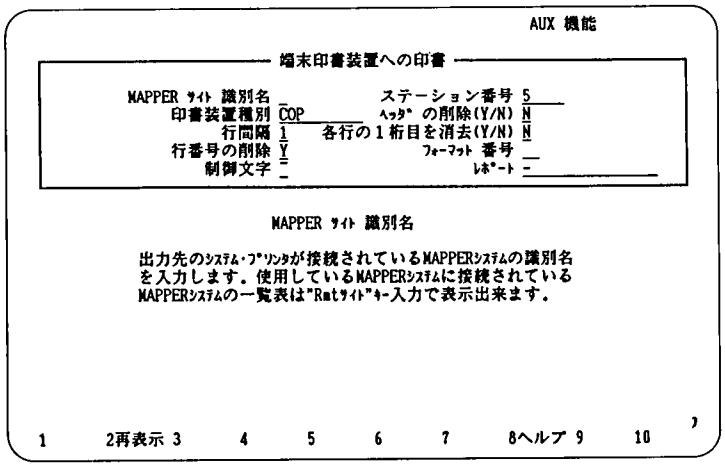


図 6　HELP の表示例

Fig. 6　Sample display of HELP

来の操作と新しい操作を選択できるようにしている．

また，HELP にも改良が加えられて，今までの目次選択型 HELP（HELP 画面をロールしながら必要な説明を探す形式）に変わって，現在のカーソル位置により，利用者が何を説明して欲しいかを認知して，適切な説明文を表示するように改良が加えられた．

図 6 は新しい HELP の表示例である．

この例の場合，メニュー画面のカーソル位置の部分（AUX 機能の MAPPER サイト識別名）に何を入力すればよいかが説明されている．

これらの利用者インタフェースの充実は，MAPPER だけで行っているので，インテリジェンシィのない端末でも使用できる．さらに良い利用者インタフェースを目指して，パーソナル・コンピュータの高度なインテリジェンシィを利用するための作業もすでに始まっている．この作業によりアイコンとマウスを使用して MAPPER のファンクションを呼び出すことや，MAPPER のランからウインドゥを開き，パーソナル・コンピュータのソフトウェアと連動して，そこにグラフ，イメージ，レポートを表示することが可能になる．これにより MML（Micro Mainframe Link）が達成でき，MAPPER はさらに進化することになる．

3.2　MAPPER 適用機種の拡大

今から約 16 年前，MAPPER はシリーズ 1100 上に初めて姿を現わした．その後しばらくの間は，シリーズ 1100 上に留まっていたが，1984 年に初めてシリーズ 1100 以外の機種に移植された．MAPPER 6 である．MAPPER 6 は小型の MAPPER 専用機であり，その使用範囲も単体で使用するという限られたものであった．しかしながら，MAPPER 6 は MAPPER の他機種への移植の先駆けであり，記念すべき機種である．

1988 年になると，新しく UNIX*マシンである U 5000/85 上に MAPPER を移植し

* UNIX：AT&T ベル研究所が開発し，AT&T がライセンスしている．

た．MAPPER Cである．このMAPPERの特徴は二つある．一つは，OSがUNIXであること，もう一つはMAPPERがC言語で書かれていることである．これらの特徴は，MAPPERが機種に束縛されなくなったことを意味している．とくにMAPPERが一般的な高水準言語であるC言語で書かれていることは，移植性が著しく増したことを示している．事実，U 5000/85に続いてU 6000シリーズでも，ハードウェアの登場と同時にMAPPERが移植された．U 5000/85とU 6000シリーズに移植されたMAPPERは，さらに1990年にAシリーズにも移植された．

またMAPPERは，ごく近い将来にPW²ファミリにも移植される予定である．この時のPW²ファミリのOSとしては，MS　OS/2*を予定している(OS/2 MAPPER)．

MAPPER自体を記述している言語がC言語に統一され，ソースコードが一つになることは，ファンクションの互換性がすべての機種で保証されることにもなる．ファンクションの互換性が保証されることは，MAPPERで作られたアプリケーション・システムが機種に関係なくどこでも動くと言うことである．したがって開発マシンと実行マシンを容易に分けることができる．また，いろいろな機種にMAPPERを移植するとき，その機種の能力を最大限に引き出すために機種に依存したファンクションは必要である．機種の規模による容量等の違いも，その機種の能力を最適にする上で必要であり，用途によって利用する機種を幅広く選択できるようにしている．

ここで少しOS/2 MAPPERについて述べてみたい．OS/2 MAPPERは，OS/2とPM** (Presentation Manager) の環境で稼働するシングルユーザ用のMAPPERである．しかし，パーソナル・コンピュータ同士をLAN (Local Area Network) で接続することによってクライアントとサーバの関係を作ることができる．この場合，クライアントとなるパーソナル・コンピュータ上にOS/2 MAPPERが乗っている必要はなく，OSはMS-DOS***でもかまわない．そのかわり，MAPPERステーションと呼ばれる特別なソフトウェアを乗せるだけでよい．OS/2 MAPPERが乗っているパーソナル・コンピュータを含めて最大4台のパーソナル・コンピュータでMAPPERを使用することができる．OS/2 MAPPERの接続例は図7のとおりである．

OS/2 MAPPERはPMの機能を十分に活用し，従来のMAPPERに比較して操作性は格段に向上している．複数のウインドゥを開いて，そこで複数のレポートを参照したり，グラフを描くことができるようになっている．

OS/2 MAPPERの登場により基幹業務(メインフレーム：MAPPER 1100，Aシリーズ MAPPER)，部門処理（オフィス・コンピュータ：MAPPER C)，個人処理（パーソナル・コンピュータ：OS/2 MAPPER）の3階層のMAPPERが完成する．今までは，個人の処理を行うにも基幹業務または部門処理のMAPPERを利用していたが，これからはMAPPERの利用者は必要に応じて，どの階層のMAPPERを使用するか選択することが可能になる（後述するが，利用者自身意識せずに各階層のMAPPERを使用することが可能になる)．しかも，同一の操作で各階層のMAPPERを使用することが可能になり，そこで作ったアプリケーション・システムはどの階層でも利用できるようになる．

* MS　OS/2：米国Microsoft社の登録商標．
** PM　　：米国Microsoft社の登録商標．
*** MS-DOS　：米国Microsoft社の登録商標．

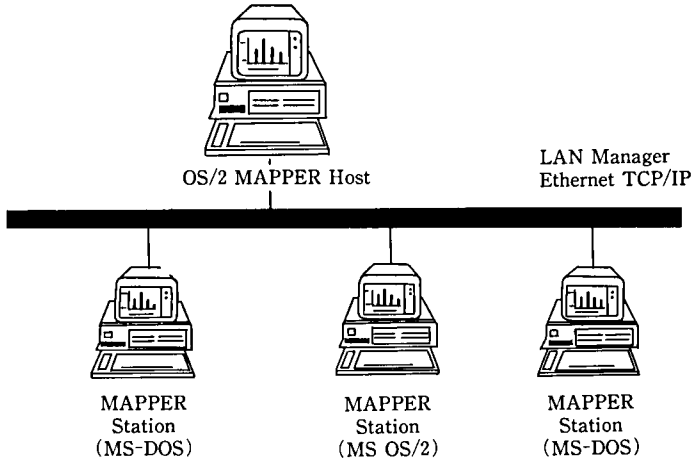


図 7　OS/2MAPPER 接続例
Fig. 7　Example of OS/2 MAPPER connectivity

3.3　ネットワークの拡大

いろいろな機種にMAPPERが移植された今日，ネットワークに関する機能の充実は，今後MAPPERが最も力をいれなければならない分野の一つである．とくに前述の3階層のMAPPERをより有効に利用するためには，各階層のMAPPERを有機的に接続しなければならない．

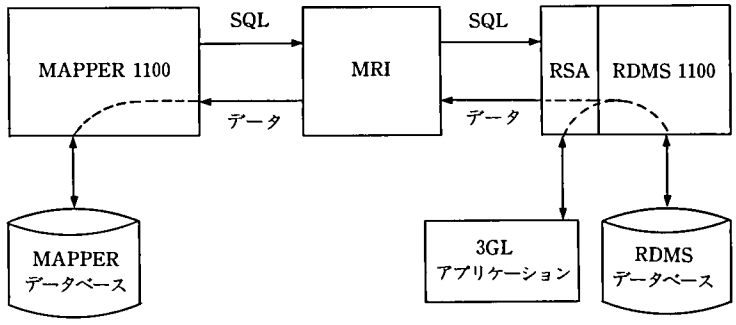
MAPPER 1100同士の接続では，ファイル転送(@CPY命令)とジョブ転送(@RRN命令)が可能である．また，MAPPER 1100とMAPPER Cの接続では，@HST命令によるパススルー機能が使用可能である．パススルーでは，MAPPER 1100は，MAPPER Cを端末として認識して，データストリームを送るようになっている．さらに効果的なネットワークを構築するには，MAPPER Cで持っているネットワーク命令群が有効である．ネットワーク命令群には，@NET命令，@NOF命令，@NRD命令，@NWR命令，@NRM命令等の命令がある．この命令群により，相手のMAPPERにサインオンして，相手のMAPPERのレポートを参照したり，書き込んだりすることができる．また，相手のMAPPERで特定の命令を実行してその結果を得ることも可能である．この命令群では，MAPPER内でプロトコルを決め，それにより他のMAPPERと双方向のコミュニケーションを行っている．このネットワーク命令群はすべてのMAPPERで利用可能になるので，機種に関係なくMAPPER間でコミュニケーションを行えるようになる．また，ネットワーク命令群は，下位のコミュニケーションのプロトコルから独立するように設計されている．このため，利用者はネットワーク命令群を使用するだけでホストマシンがLANで接続されていようが，WAN (Wide Area Network) で接続されていようが，その接続形態と無関係に他のホストのMAPPERとデータの交換が可能になる．

3.4　MAPPER以外のデータベースの参照

前述のネットワークが実現できたとき，並行して考えなければならないのはデータベースをどのような方法で交換するかである．MAPPERで個人の処理をしている最

中に部門のデータや基幹業務のデータが必要になったとき，それらのデータを簡単に抽出できれば便利である．必要なデータがMAPPERのデータであるのならば，ネットワーク命令群を使用してデータの交換が簡単に行える．ところが必要なデータが，一般的に使用されているMAPPER以外のデータベースの時には，そう簡単に抽出することはできない．この要求に応えるのがMRI (MAPPER RDMS Interface) である．MRIは，標準的なデータベース参照言語であるSQL (Syntax Query Language) により，外部のリレーショナル・データベースをMAPPERで取り扱うために作られたものである．MRIは，初めはMAPPER 1100で実現された．当初は一つのラン命令 (@SQL命令) のみが提供されていた．この命令は，SQL構文をRDMS 1100 (Relational Database Management System) に渡すものであった．さらに便利にするため，良く行う操作をまとめて，@FCH命令，@RAM命令等の命令を新たに作った．

図8は，MAPPER 1100，MRI，およびRDMS 1100の関連を図示したものである．



RSA：Relational Sytax Analyzer

図 8　MAPPER 1100，MRI，RDMS 1100の関連図

Fig. 8　Relationship of MAPPER1100, MRI and RDMS1100

MRIの考えはMAPPER Cにも引き継がれた．MAPPER Cの場合，対象となるリレーショナル・データベースは，ORACLE*である．やがてMRIは，Aシリーズ MAPPERやOS/2 MAPPERにも引き継がれる予定である．

一つのホスト上で一般のデータベースを扱えるようになると，次には別のホスト上にあるリレーショナル・データベースを取り扱いたくなる．このような要求に応えるためMRIは，他のホスト上にあるMRIとコミュニケートしてデータの交換を行えるように進化する．複数のホストがある場合には，MRIだけの導入でそのホストのデータを別のホストのMRIに渡すことができる．

前述のようにMRIは，SQLによりデータベースを参照するように設計されているので，米国ユニシス社以外のデータベースでも参照できる可能性を持っている．たとえば，MRIが，SNA** (System Network Architecture) を通してIBM社のマシンと接続し，DB 2***のデータを抽出することである．IBM社側のマシンに特別なソフ

* ORACLE：ORACLE Corporationの登録商標．
** SNA：IBM社の情報通信アーキテクチャ
*** DB 2：IBM社の開発したリレーショナル・データベース管理システム

トウェアを置くことによって，データベースが参照できるように計画中である．

実際に，米国ユニシス社では，図9のような環境でMRIを使用して，OS/2 MAPPERの端末から，各マシンにあるデータの一覧表を作るデモンストレーションを実施している．

MAPPERとMRIの最終目標は，利用者にデータベースのあり場所を意識させないことであり，どのホストを使用しているかを意識させないことである．利用者は，望んだデータがあたかも自分自身のホスト上にあるかのように参照することが可能になる．

パーソナル・コンピュータが一人に一台に割り当てられ，そこで仕事をする環境を考えてみよう．パーソナル・コンピュータの記憶容量は，一昔前に比べて大きくなったと言ってもやはり限られたものである．仕事に必要なデータをすべてパーソナル・コンピュータに持たせることはできない．したがって，ある情報は基幹業務を処理するコンピュータから，またある情報は他のコンピュータから持ってくる必要がある．この場合利用者は必要な情報が，どのような形（データベースの種類）でどのコンピュータにあるかを覚えておく必要がある．また，実際にデータを移動する時にもある種のユーティリティを利用することが必要である．これは，利用者にとっては，かなり面倒なことと言わなければならない．前述のデモンストレーションで行ったように，データのあり場所を一覧表等によって正しく管理できれば，その情報を利用して利用者に何の意識もさせず，必要なデータを必要な時にいろいろなコンピュータから収集することは，MAPPERとMRIにとっては容易である．

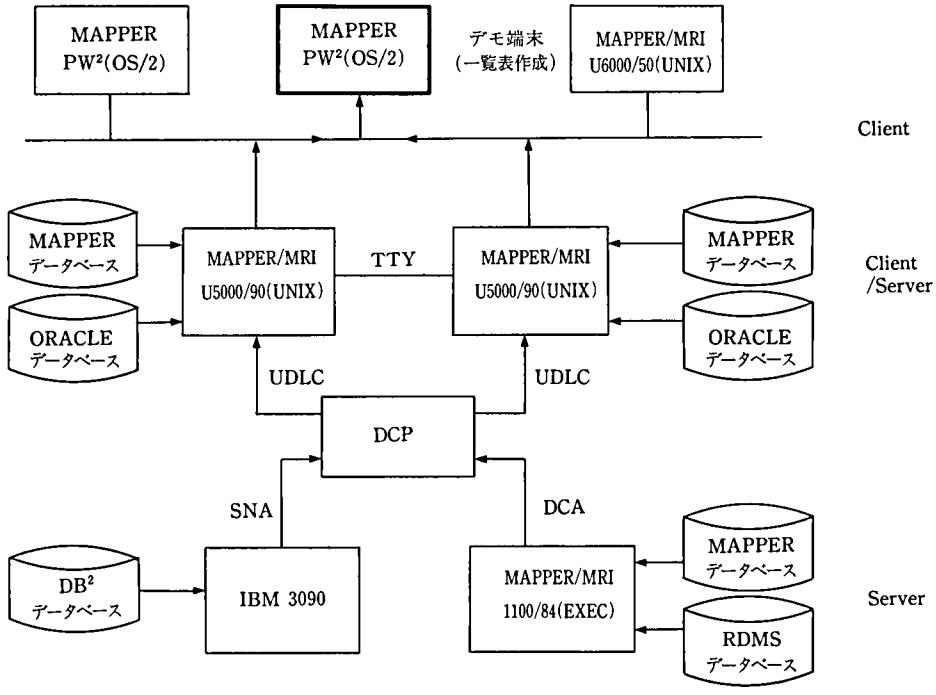


図 9　MRIデモの構成

Fig. 9　MRI demonstration environment

このような仕組みを作れることは，他の4GLの追従を許さないであろう．

3.4 その他の機能の充実

MAPPERは，利用者が必要とする機能をタイムリに提供することによって進化し，その地歩を確固たるものにしてきた．ここで過去にどのような必要性から機能が追加されたかを，具体的な例を挙げて振り返ってみよう．

MAPPERでは外部の3GLの世界と連動するために，バッチスタート機能(バッチランの起動）とバッチポート機能（バッチランからのMAPPERラン・ファンクションの起動)が用意されている．この二つの機能を使用することによって，MAPPERは3GLの世界とデータを交換することができる．これらの機能は非同期であった．3GLの世界と同期を取りたいという要求を満たすために，MTQ (MAPPER Transaction Queuing)の機能が追加された．この機能を使用することによってMAPPERは，3GLの世界（COBOLプログラム）と会話をしながらデータの授受ができるようになった．また，ランを容易に作成したいという要求に応えて，会話機能からMAPPERラン・ファンクションを自動生成するためにRUNランの機能が追加された．さらに定期的に行う処理を自動化したいという要求に対応して，バックグラウンド・ランの機能が追加された．バックグラウンド・ランを利用することにより，たとえば一定量のトランザクションが処理されて，マスタレポートが更新されたとき，そのマスタレポートを各営業所や支店に自動的に配布することが可能になる．

次に，将来，追加しようとしている機能のいくつかを紹介してみよう．多くの利用者から要求されているレポートの桁数の拡大もAシリーズMAPPERやMAPPER Cで実現し，今はMAPPER 1100で実現するために開発を実施している．またMAPPERのソート処理でシステムのソートルーチンを利用することも計画している．システムのソートルーチンは，当然のことながらソート専用に作られているため，これを利用することによりMAPPERも高速のソートを実行することが可能になる．さらに，MAPPERシステムの安全性を高めるためMHFS (Multi Host File Sharing)におけるホット・スタンバイの機能も計画している．すでに述べたように，パーソナル・コンピュータの豊富な機能を利用することも開発中である．

MAPPERは，今まで利用者の要求を吸い上げて新しい機能を実現してきた．そして今後もこの方針は変えない．

今まで述べてきたようにMAPPERは，

1) 適用機種の拡大
2) ネットワークの拡大
3) MAPPER以外のデータベースの参照

を実現することにより単なる言語 (Language) の範ちゅうを越えて，アプリケーション・システム開発と実行の環境を提供するプロダクトに進化すると言える．

4. おわりに

コンピュータ技術者の不足は深刻な状態になってきている．ソフトウェア開発の生産性を上げるために，MAPPERはさらに多くの人たちに使われるであろう．MAPPERは，利用者の要求に応えることによって進化してきたし，これからさらに進化し

拡大する．その一つは，いかにCASE（Computer Aided Software Engineering）ツールとの統合を行うかである．米国ユニシス社では，MAPPERに対してCASEツールとしての特徴を付加することを計画している．また，今までMAPPERは米国ユニシス社独自のアーキテクチャの上で使用可能としていた．MAPPERが，UNIXやMS OS/2等のオープンな世界に登場したことを含めさらにグローバルな4GLとして進化させたい．

執筆者紹介　柳　沢　　勝（Masaru Yanagisawa）
昭和45年早稲田大学教育学部卒業．同年日本ユニシス(株)入社．ベーシック・ソフトウェアの開発と保守に従事．現在，OAソフトウェア部MAPPERソフトウェア課所属．

データベースを再編成すべきか否か...それが問題だ

To Reorg or not to Reorg: That is the Question

T. Miller

レスポンスタイムは遅い．リカバリテープは増える．Maper 0は毎日，大きくなってくる．調べてみると20個のランがシステムに大きな負荷をかけていることがわかった．これは6か月前と同じ状況だ．確かランデザイナが一生懸命がんばってランの書き換えや効率的なコマンドの使い方でいくらかI/Oを節約したはずだ．ここで乾いた雑巾を絞るようにまだ効率アップが計れるだろうか．あるいは，もう少し考えれば何か解決の糸口が見えてくるだろうか．最初の自問の答はノー，次の答はイエスである．

実行パフォーマンスを上げたい．ランで上げなければMAPPERだ．MAPPERは巨大なレポーティング・データベースである．MAPPERデータベースの構成や形式はデザイナの経験によって決まることが多い．COBOLプログラマがデザインしたMAPPERのデータベースとCプログラマのものとではまったく違うことがよくある．共通して言えるのは，どちらもMAPPERのフレキシビリティ（柔軟性）を十分に利用していないことだ．

筆者はこの小論でデータを再編成するいくつかの方法について考えを述べてみたい．その方法とは次の3点である．

1) ADD RID法……最も単純で効率的な方法
2) パラグラフ化……玄人好みの方法で効率的．一般ユーザにとっては難解な手段
3) タイプ分散法……リレーショナルな使い方．階層構造データにこの方法を採用するにはデザイン上の工夫が必要

ここで問題の解決法は一つだけとは限らないことにご注意いただきたい．時と場合によりさまざまである．つまり，データベースを何回くらい更新するか，検索するか，あるいはデータを追加するかでデザインを考えなくてはならない．

1. ADD RID法

筆者から質問を一つ．読者がMAPPERの環境下で最悪と思われる状況や使い方を一つ挙げていただきたい（ただしハードウェアのトラブルは除く）．

すぐに次のような答が返るはずだ．

① コーディネータがABORTと入力すること．
② 100,000行のレポートをソートすること．
③ Pオプションを指定せずにマッチ機能を使うこと．
④ モード218，タイプEの全レポートを削除してしまうこと．
⑤ 本番用のランに@LOG制御文を挿入したまま忘れてしまうこと．

(Thurman Miller, Information Center, NIKE Inc.)

なるほどそれぞれもっともな理由がある．そこで筆者はと問われれば，

「RIDの行数を変えること」

と答えたい．

つまり，レポート，リザルトを問わず，データ行を1行あるいは1,000行追加したり削除すること，これが最悪の使い方だとあえて申し上げたいのである．レポートの行数が変化するとどうなるか．まずそのRID全体がディスクとリカバリ・テープに書き込まれる．ただし，リザルトや外部ファイル指定のレポートの場合は，ディスクだけに書き込みされる．10,000行のレポートに1行追加すると，ディスクだけでもおよそ200〜300回のI/Oを実行することになる．決して無視できる負荷ではない．

さて，「ADD RID法」とはどんな方法であろうか．データベースへデータの追加があった場合，それをすべて記録するエリアがADD RID（追加データ専用のレポート）である．複数のタイプにデータを分散しているリレーショナル・データベース型の時は，各タイプごとにADD RIDを持つ．この手法を採用するには，あるデータ項目を見つけようとするときADD RIDを探すようにランを修正する必要がある．それと最終的にADD RIDと通常の本番用データベースを定期的（毎月・毎週・毎日の夜間処理等）にブレンドする別ランが必要となる．

データレコードの削除に代わる方法にはいくつかある．

① そのデータレコードに削除レコードであることを意味する空白を書き込む．

② 行タイプを変える．

③ ①または②のどれかだが，同時に実施した人と日時も書き込んでおく（共用データベースの場合，ミスオペの責任をハッキリさせるため）．

ADD RIDの一例を挙げよう．

```
.DATE 89 JAN 04  17:04:34  RID    10B   89 FEB 04   JDOE
.(006)
*
* Last Name       First Name      Address           City         St
*================.==========.====================.=============.==.
```

ここで2行目は，次に書き込み可能な行番号を記録する場所である．次に続く空白行へデータを書き込む時は，この方法がおそらく最も効率的である．FNDで最初の空白行を探す方法もあるが，本稿では割愛する．

さて，このデータベースへレコードを追加するラン制御文は次のようになる．

```
@LOK,m,t,10 .                          専有使用のためロックする
@LZR,m,t,10 <LastLine>15 .             レポートの最終行番号を得る
@RDL,m,t,10,2 3-3 <NextLine>13 .   次の空白行の番号を得る
@IF <NextLine> LE <LastLine> GTO LIN +2 ; .   書き込む余裕はあるか？
@LN+,m,t,10,<LastLine>,100 .           余裕なし、空白行１００行追加する
@WRL,m,t,10,<NextLine> 'Last','First','Address','City',¥
'St' [,<LastName>,<FirstName>,<Address>,<City>,<State> .
@INC <NextLine> .
@WRL,m,t,10,2 2-5 ,,'('<NextLine>')'  ULK .
```

このレコードは100回に1回しかレポート行を追加しないようになっている．

ADD RID法ではHAPPERの最も重要な特徴の一つ，リアルタイム処理ができない．しかし，次のような工夫をすれば解決する．

・目的の行を探索する時は：

[従来の方法]

```
@BFN,m,t,r,,lab Q 'key field' [,<key> .   マスターＲＩＤの検索
```

(注)labは，〈key〉に合致する行が見つからない時の行き先ラベル

[工夫した方法]

```
@BFN,m,t,r,,LIN+1 Q 'key field' [,<key>  GTO LIN +2 .  マスターＲＩＤの検索
@FND,m,t,r,lab '' 'key field' [,<key> .                 ＡＤＤ  ＲＩＤの検索
```

対象データベースにあるかどうかをBFN命令で探索し，もし見つからなければADD RIDを探索する．

・レポートをブレンドする時は：

[従来の方法]

```
@SRH,m,t R<開始RID>-<終了RID> 'key field' [,<key>  RNM -1 .
```

[工夫した方法]

```
@SRH,m,t DR<開始RID>-<終了RID> 'key field' [,<key>  RNM -1 .
@SRH,m,t,<ADD RID>,6,,LIN+3 D 'key field' [,<key> .
@SOR,-0 '' 'key field' [,1 .
@MCH,-1,-0 DB 'key field' [,1 'key field' [,1  RNM -1 .
```

(最後のリザルト−1には抽出した全部のレコードが入っており，ソート済みである)

対象データベースを検索し（ソート済みであること），ADD RIDをサーチしたリザルトとブレンドするのである．

このランは定期的にスケジュールする必要がある．このランを実行すると数分から数時間かかるだろう．しかし，ADD RID以外のレポートの行数が変化するのは，このランを実行している時だけなのだ．ここがミソである．新レコードは一括して同時にブレンドされる．通常のように新レコードが発生した都度ではない．

本番用データベースへレコードをブレンドすると同時に，すでに削除済みのレコードを除去する働きもする．削除済みデータレコードを'将来利用するため'保存しておく場合がある．しかし失ったレコードを保存テープから探し出すため，コーディネータはどれほどの時間を費やしていることか．削除済みレコードを短時間に保存できれば，コーディネータの負担も多少は軽くなるはずである．

この方法で大事な点は，なぜADD RIDを使うのか，そのRIDはどこにあるのか，そしてブレンド作業がいつ行われるのか，ユーザへ周知徹底することだ．

以上，「ADD RID法」について述べた．このデータベース再編成法が読者の会社に適しているかは，次のポイントで判定できる．

① 行数が毎日10回以上も増減するレポートがあるか？10,000行を越える大規模レポートの時は5回以上のものがあるか？

② I/O使用率90〜100％，CPU利用率20〜40％に着目せよ．これでI/Oがボトルネ

ック(隘路)になっているのか，あるいはCPUの使いすぎ状態なのかが判断できる．このような状況下ではワークステーション側にいるユーザの応答時間は良くない．I/O使用率はSIP/PARによる分析で簡単に判明する．

データベース再編成に「ADD RID法」が実際に効果的かどうかはわからない．何かパイロットプロジェクトから試行してみるのがよいだろう．

ここでI/O負荷が大きいコマンドは次のとおり．

@ADD　このコマンドはマニュアルの記述から削除すべきと筆者は個人的に考えている．たとえばMCH命令をBオプション指定で使用することをランデザイナに徹底するのがよい．

@LN　LN+，LN−，LNX等，類似命令も同類である．別の方法で十分同じ機能を果たすことができる．たとえば，LN−命令でレポート内の全行を削除する等は，間違えた使い方の最たるものであろう．これなど，RDC命令で見出し行を読み，そのリザルトをREP命令で置換すればよい．確かにランのステップは多少増えるが，I/O負荷の減り具合から言えば文句はないはずである．

2. パラグラフ化法

読者は何ゆえに行タイプというものをユニシスが考え出したとお考えか．考えれば考えるほど，大したことを発明したなあと感嘆せざるをえない．行タイプという考え方になれるまでには少々経験が必要ではあるが，いったん理解できるとその利用価値は実に大きい．

本稿では「パラグラフ化」とは何か，またなぜ筆者が推奨しているかについて述べたい．

MAPPERレポートでいうパラグラフ化とは，複数のレコードレイアウトを使うという意味である．つまり，一つのレポート内の複数行で1件の情報を記録するのである．従来からある手法に，1件の情報を別々のタイプに分散させる方法がある．この従来手法を使いたい方は，次の一文を読んでいただきたい．

タイプに分散する方法を例示してみよう．いまリレーショナルデータのレイアウトを想定し，1件のレコードが五つのタイプにまたがって存在しているとしよう．まず単純なケースとして1件のレコードを更新することを考える．簡単だ．1レコードを呼出し一つまたは複数の画面に表示する．続いて次のレコードへ移る．読み込み1回，書き込み1回である．...ほれ間違いだ！　レコードは五つのタイプに分散しているのである．実際には次のようになる．

```
BFN 命令でタイプBを探す（2〜10回のI/O）
1/5の情報を変数に読み込む

BFN 命令でタイプCを探す（2〜10回のI/O）
1/5の情報を変数に読み込む
```

これを繰り返す．1件のレコードを全部読み込むまで10〜50回のI/Oが必要となる．

レコードを表示する（1〜100回のI/O）

再びBFN命令でタイプCを探す（2〜10回のI/O）
1/5の情報をレポートへ書き込む

..あとは説明を要しまい．

このように1件のレコードを読み書きするため20〜100回のI/Oが必要である．さらにリカバリテープへも5回（五つのタイプへWRL命令を実行するごとに1回）の書き込みが発生する．

たとえ各タイプの行位置が同じでも，データを書き込むフィールドが各タイプ共通であっても，システムに与える負荷は大きいのである．

こんな単純な1回の操作で，これだけのI/Oを大量消費するのだから，毎日1000行のレポートを更新することを考えてみていただきたい．パラグラフ化の手法を採用すると，この例の1件レコードはすべて1レポート内に，しかも連続した行に格納できる．

前述の例は「パラグラフ化法」を用いると次のようになる．

BFN命令でレコードを探す（2〜14回のI/O）
RLN命令で1行ずつ5回，データを読み込む

一つまたは複数の画面でレコードを表示する（1〜100回のI/O）

再度，BFN命令でレコードを探す（2〜14回のI/O）
複数行にわたる1件レコードの全情報を1回のWRL命令でレポートへ書き戻す

これで終りである．I/O回数も少なく，リカバリテープへの書き込みも1回で済む．

この例でI/O回数を2〜14回と言い，前例では2〜10回と言った．誤植ではない．いま1000レコードが五つのタイプに分散しており，これを一つのパラグラフにまとめたとすると，全部で5000行になる．1000行のレポートよりも5000行のレポートにI/O回数が余分にかかるのは当然である．

問題はパラグラフ化に馴染めるかどうかである．まずパラグラフ化するには，各パラグラフの第1行目はそのレポートの見出しと一致したフィールド構成になっていることが必要である．パラグラフの2行目以降はどうなっていてもかまわない．次の例を見ていただこう．

```
#                                  Birth   Marital    #
# Last Name       First Name M     Date     Status   Dep
#================.===========.=.========.========.====.
 Doe              John         W 01/01/60 Single      0
#1212 La Junta Street      Colorado Springs CO 80810
#1500 N. Broadway Rm 2401 Aurora             CO 80817
#06/01/75 04/15/89 Robinson     Jeff     713-868-9704
#WZ04674 CO 06/30/90 Chev 4dr 89 08525
```

これで1レコードである．このようなレコードがたとえば1000レコード，次から次へと重なっていると想像していただきたい．とても会話機能では手が出せないのだ．唯一，ランを使うしかない．

さて，このレコードのうち，1行だけを用いてレポートを作成することを考えてみよう．1000行をサーチする代わりに5000行をサーチすることになる．それではこのレコード内の2行を用いてレポートを作成するとどうなるか．5000行を2回サーチしなくてはならないから，合計10,000行のサーチとなる．

なんだパラグラフ化してもかえってI/O負荷が大きくなるのじゃないか...と思われるだろう．会話機能は論外だし，1000行対5000行では，1タイプ内にある1件レコードを探すにも余計な時間がかかりそうだ．

確かに．しかしアプリケーションの多くはオーダエントリやデータ更新だけであることが多い．統計レポート処理等と言っても実際には受注データを入力し，印書してデータ更新を行い，最後に請求書を作成するのが主な内容であったりする．受注データを1件処理するにしても5〜10回の更新処理が必要であろう．このようなアプリケーションにはパラグラフ化が向いている．

パラグラフ化の特徴を挙げると，

① 応答時間をドラマチックに削減できる．

② データの更新や照会等に要するI/Oが大幅に節減できる．

③ レポートの取扱いはランデザイナ（ユーザ）にまかせる．内容の一貫性と正確性を保ちながら，MAPPERの経験が少ない人でも手早く内容を書き換えたり照会することができる．

④ モードやタイプの所要量を減らせる．筆者は可変行数レコードでパラグラフ化したアプリケーションを見たことがある（4行〜120行/レコード）．システムへの負荷はわずかであった．その例ではパラグラフ化に加えて，データを同一タイプ内の複数のレポートに分散させていた．

パラグラフ化しようとする場合，最も簡単なのは基本とするタイプへそれぞれのタイプをアペンド（追加）して，複数行からなるレコードを作成する方法である．こうしてレコードを構成する各行の左端フィールドにキーフィールドを持つ．少々めんどうだがそれだけのメリットがある．

行タイプが分かれていても各行の第1フィールドで識別できると，会話機能が使えるようになる．つまり，ある行タイプでデータをサーチし，次に両方の行ともキーフィールドを持っているため別の行タイプのデータとマッチさせることができる．別々のデータ行を結合するにはこの方法が最も簡単である．

8行以上のマルチレコードの時は，実際のデータベースに対してインデックスを管理したくなる．こうすれば1レコードを探そうとするとき8000行をすべてをサーチせず，1000行で済むのである．インデックスはたとえば，次のようなフォーマットにする．

```
*
*Key field  RID Line#
*==========.===.=====.
```

このインデックスをBFN命令で探すことにより，項目を探すためのI/O回数は少なくなるであろう．インデックスからRID番号とレコードの行番号が得られるので，対象レコードは次の1回のI/Oで直接アクセスできることになる．アプリケーションによってはこのインデックスに別の情報も含めておくと，データ内容を高速検索することも可能となる．

インデックス法はある程度の負荷（データの重複の問題やインデックスを夜間に保守する問題）増となる．しかし，パラグラフ化したデータベースをインデックスでアクセスすると，負荷増のためのコストがかかるもののアクセスタイムを縮める効果を期待できる．

3. タイプ分散法

先ほど複数タイプにレコードを分散させる方法について批判しておきながら，なぜ「タイプ分散法」を取り上げるのか．適不適は状況によるからである．アプリケーションによってはリレーショナル・データベース型であることが，どうしても必要なことがある．

これは筆者の解釈であるが，MAPPERはタイプを分散させることを念頭に開発されていると思う．キャビネット，引き出し—キャビネットに納まる8個の引出し—整然とした形でひとまとまりのデータを含んでいるキャビネットと考えると，このデザインの面白さにUnisysは気づかなかったのではとすら思えてくる．

本題に戻る．タイプ分散の可否を判断したい時には，次のように自問自答してみるとよい．

“〈ある事柄〉について，全社的に管理されているさまざまな情報と個々のグループが知りたい情報とが異なっているか否か”．

いわゆるリレーショナルなデータベースを作り上げる時のアドバイスを若干行いたい．リレーショナル型を考える時に，同時に構造型のデータベースを考えてはいけない．データを単にタイプに分散させるだけでなく，できるだけ多くのタイプにまたがってRIDを作るのがよい．もちろんその時には，論理的に分散されることが重要だ．アクセスタイムについて言えば，できるだけ少ない回数のサーチでレコードを見つけるには，データをどのようにグループ化できるかということが課題である．

データが類似しているからという理由だけでグループ化してはいけない．つまり住所データは全部あるタイプにまとめるとか，このタイプは会計データだけにする，としないことだ．グループ化にするにあたって，I/O回数を減らせるちょっとしたヒントがある．

まず頻繁に使われるレポートの上位五つ，あるいは1日20回とか50回とか多く使われるレポートをピックアップする．そしてこれらのレポートのデータがなるべく接近するようなタイプを作る．タイプ分散法ではレポートを編集するとき，複数タイプのサーチやマッチが必要になることが多いので，これは現実的な方法といえる．複雑なレポートを1タイプだけで作れるとしたら，I/O回数は大きく減らすことができる．もちろんこれは使用頻度の非常に高いレポートにだけ有効だ．このやり方では，上司の各前と彼の電話番号が別のレポートになってしまうといった危険性はある．しかしそれでもこの方法は検討に値する．

1. リース物件管理システム

リース物件管理システムは，企業がリース会社とリース契約を結び物件を賃借する場合のリース契約管理，保守契約管理，リース物件管理および会計処理を対象として，

- リース取引き業務における事務作業の省力化
- 将来にわたってのリース料の試算等から企業における予算編成・設備投資計画等に資する資料提供

を狙いとした付加価値の高い社内OAシステムである（図1）．

1) 特　　徴

① 今後予定される物件賃借の予定データを含めた予算部署ごとの月別または，年別のリース料の試算を行うことで，予算編成・設備投資計画等のための資料を提供する．

② 本システムで蓄積するMAPPERデータベースを汎用検索機能で条件抽出し，計算・作表等の加工を加えることで非定型業務への対応が容易にできる．

③ リース取引き業務で発生する費用および支払い情報に関しての仕訳データを作成し，総合会計システム（UNIMACS）に自動的に引渡す．また，他の会計システムについても容易に連動が可能である．

2) 構　　成

本システムは，リース契約締結からリース期間終了までのリース取引き業務全般を対象として，次の三つの機能より構成される．

① 契約管理……リース契約書に記載されているリース契約条件等の契約情報を管理する．

- リース契約
- 保守契約
- 再リース契約
- リース契約異動履歴

② 物件管理……リース物件に関する基本情報および異動履歴情報を管理する．

- 物件移動
- 物件解約
- 移動履歴
- 修繕履歴

③ 会計処理……リース取引き業務で発生する費用および支払い情報の管理と仕訳データを作成する．

- リース料等の費用計上
- リース料等の支払い
- リース料の試算

2. 卸売業総合情報システムWIN'S

WIN'S（Wholesale Information Network System）は卸売業の中でも，とくに情報指向型の加工食品卸売業，菓子卸売業，雑貨卸売業等を対象に業界の特性を考慮して開発されたシステムで，販売管理，分析・予測，一般経理，給与計算

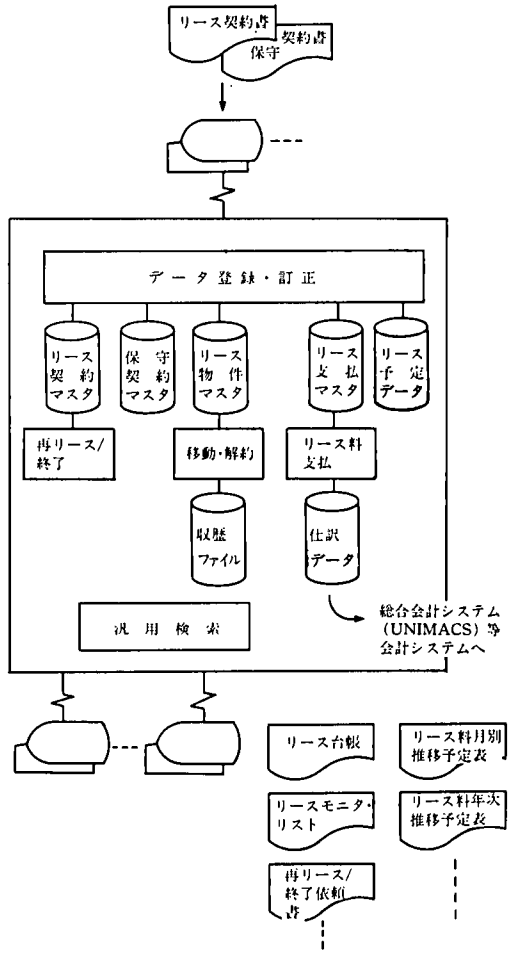


図1　システム概念図

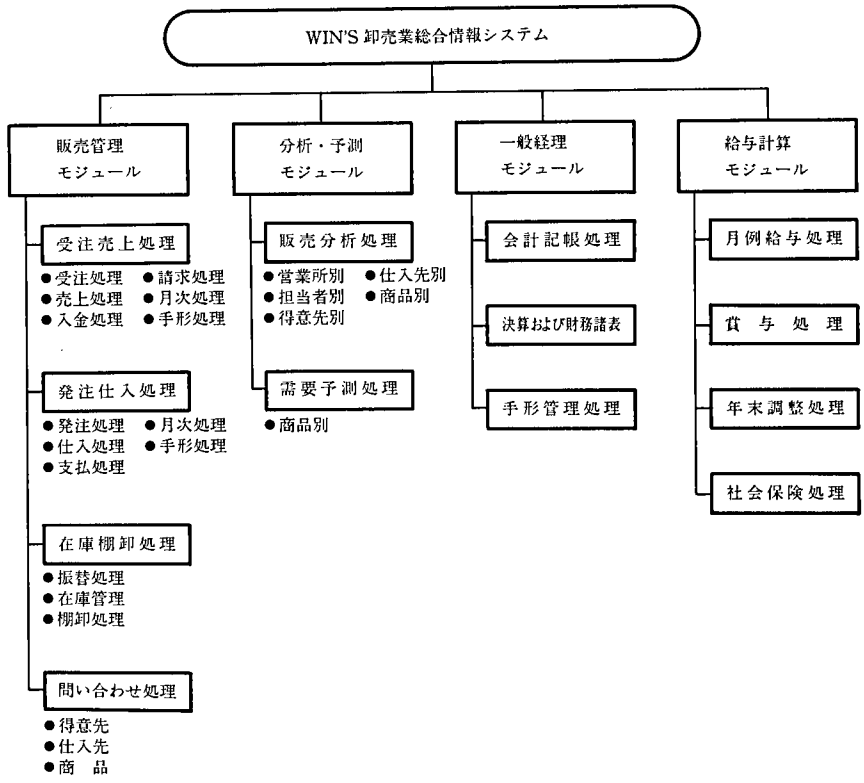


図2 WIN'S 卸売業総合情報システムモジュール構成図

の四つのモジュールで構成されている（図2）.

これらモジュールは，すべてLINCで開発されているため，ユーザへの適用も容易である．とくに販売管理モジュールは，ユーザのカスタマイズを前提に，基幹業務をプログラム化してあるので，多様化するユーザニーズや常に変化する経営環境に柔軟に対応できる構造になっている.

このシステムを導入することにより，

- 受発注処理からの一連の処理による企業情報処理のシステム化
- 多品種少量物流に対応した物流システムの確立
- 単品管理による利益管理システムの確立
- 販売分析による企業経営に直結した情報のシステム化
- 需要予測による適正在庫管理
- 定型業務である経理，給与業務の早期システム化

等が可能となり，企業として独自の総合情報システムを容易に構築できる.

1) 特　　徴

① ユーザが自社業務に合うように，システムを変更して使用するTailoredシステムとして開発されている（販売管理モジュール）.

② カスタマイズを前提としてプログラムサイズのコンパクト化が行われている.

③ 機能の追加・変更を容易にするため，LINCで開発され，各プログラムはスケルトンや共通ロジックを使用した構造になっている.

④ プログラムの実行方式は，誰にでも簡単に操作できる業務メニュ選択方式である.

⑤ オペレータごとのセキュリティIDとパスワードにより機密保護を行っている.

⑥ オペレーションを容易にするため，入力項目を説明した操作補助画面（HELP面画）が用意されている.

⑦ 帳表は，リモート出力端末かLBP(Laser Beam Printer) のいずれかに出力できる.

2) 機　　能

WIN'Sを構成する各モジュールの主要な

機能として，以下の豊富な機能が用意されている．

① 販売管理モジュール
- JCA で受信した受注データの処理
- 入力端末ごとに伝票番号の自動採番管理が可能
- 得意先優先順位による在庫引当
- 得意先ごとに在庫引当時の欠品処理方式が選択可能
- 得意先と請求先，仕入先と支払先を分離することによる柔軟な請求/支払管理
- 売上伝票の即時/バッチ発行が可能
- 単品利益管理

② 分析・予測モジュール
- 販売管理データを使用した販売分析・需要予測
- 営業所・担当者・得意先・仕入先・商品別の販売分析が可能
- 6種類の予測手法で分析し，その中で最適な予測手法を選択し，向こう6か月の商品の需要予測を行う

③ 一般経理モジュール
- 取引会計記帳処理
- 決算および財務諸表作成処理
- 手形管理帳表作成処理
- 予算，資金管理等の経営管理帳表作成処理
- 比較損益計算，比較貸借対照表等の財務分析帳表作成処理

④ 給与計算モジュール

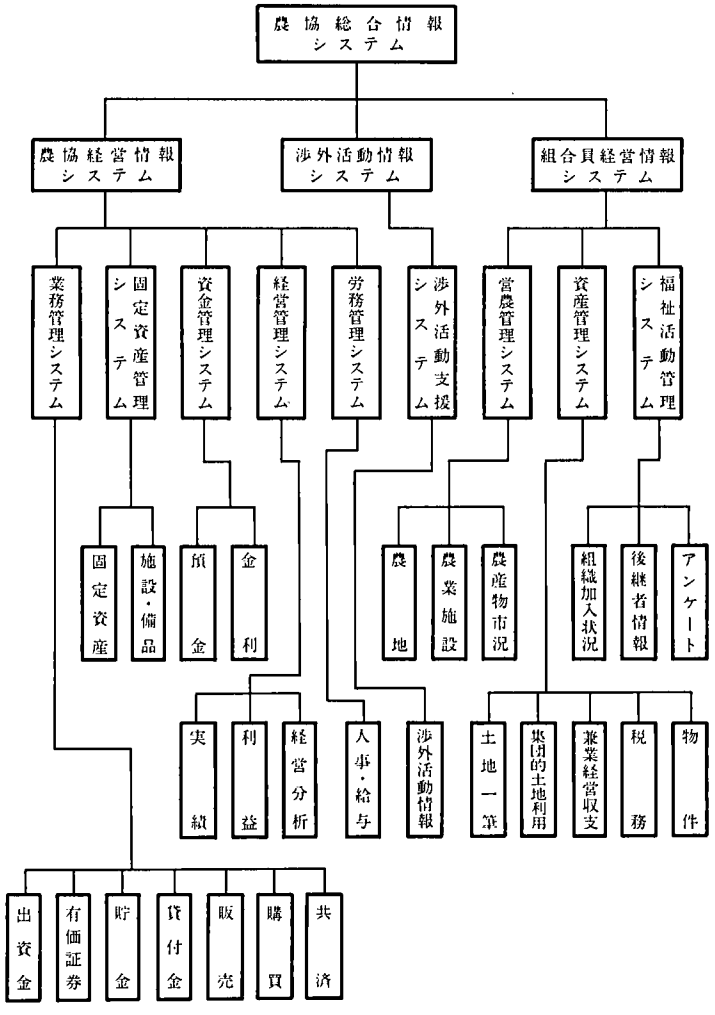


図3　農協総合情報システム

・月例給与処理
・賞与処理
・年末調整処理
・社会保険処理（オプション）

3. 農協総合情報システム ATOMS

ATOMS（Agriculture Total Management System）は，組合員一人一人の情報をベースに，信用・経済・共済・その他事業の縦割情報を有機的にリンクさせた農協総合情報システムのパッケージである．全中（全国農業協同組合中央会）の構想をにらみながら，各々の業態間の恒根をなくして，総合事業体としてのメリットを生かした総合情報システムを早期に構築し，単位農協の経営のあり方を見直すとともに，組合員へのサービスの充実をはかることを目的にしている（図3）．

1) 特　　徴

① 農協経営情報システム・組合員経営情報システム・渉外活動情報システムにて構成されており，各々のサブシステム単位で段階的導入が可能である．

② 本所・各支所の全部署を結び，信用・経済・共済・その他の各事業システムと連動しているが，オンライン部分はすべて"LINC"にて開発されているためカスタマイズ作業が容易である．

③ 各連合会システムとは共存の位置づけにあり，単位農協として必要な，より詳細な情報を補完するシステムである．

④ 各連合会システムとのインタフェースは密であり，回線・M/T・フロッピ等でデータの受渡しが可能である．

⑤ ペーパーレスシステムを指向しており，伝票帳票の出力は必要最小限にとどめ，検索画面を豊富に用意している．

2) 機　　能

① 業務管理システム

・信用業務（県信連の勘定系・情報系システム）の補完システムとして，貯金の支所・科目別計画対比および自由金利型定期貯金の特別管理を行う．また貸付金の支所別資金計画対比・貯貸率・平均利回り・運用状況管理を行う．

・経済業務（販売・購買サブシステム）については，播種・出荷予約・荷受・分荷および送り状・市況受信・入金精算・販売実績照会と，予約・発注・受入・未払金・供給・未収金・転送・精査・購買実績照会・在庫照会を行う．

・共済業務は，予約・新契約・異動・自動車共済・立替金・共済資金・借入貸付金・精査・共済実績照会を行う．

② 固定資産管理システム……固定資産・備品を一件ごとに管理し，有効活用を図るための情報提供を行うとともに，減価償却計算を行い決算に対応する．

③ 資金管理システム……調達については貯金サブシステムにて，運用については貸付金・有価証券・預金サブシステムにて，効率的な資金の運用活動を行えるよう支援する．

④ 経営管理システム……販売・購買・共済・その他各事業の支所別実績・利益・予算管理を行いながら経営の推移を把握する．

⑤ 労務管理システム……人事に関する情報をあらゆる角度から，ただし機密保護制約の中で検索する．

⑥ 営農管理システム……土地の一筆ごとの現況・意向・利用形態および生産品目別価格推移を把握するとともに，農業用施設を管理し青色申告時の原価償却計算サービスを行う．

⑦ 資産管理システム……土地の現状・利用状況を把握し，効率的土地利用および税制対策を行う．また兼業経営収支管理としてアパート・マンション経営の立案・施行・入居者斡施・契約・賃貸料管理を行う．

⑧ 福祉活動管理システム……農協における各々の組織に加入している組合員管理およびその活動状況を把握し，組織の活性化を支援する．また後継者情報管理や各種アンケート結果集計を行う．

⑨ 渉外活動支援システム……渉外活動を支援するために信用・経済・共済・その他事業に関する商品情報・個人情報・取引情報を提供する．

オフィスプロセッサ，シリーズ8 10000E
Office Processor, Series 8　10000E

日本ユニシスでは，その販売開始以来10年以上にわたり，さまざまな業種の数多くのお客様に最適な能力のオフィスプロセッサを提供してきたが，情報処理システムを攻めの武器として活かす戦略情報システム（SIS）への客先志向に応え，従来製品に比べ，そのシステム能力の大幅な向上を図ったシリーズ8最上位モデル「10000 E」を発表した（写真1）．

「シリーズ8　10000 E」は，汎用小型コンピュータを凌ぐリレーショナル・データベース（RDB）処理能力と高いシステム拡張性を備えると共に，SIS構築に欠かせない「高速RDB処理」，「大規模ネットワーク対応」，「より使いやすいエンドユーザ・コンピューティング機能」を実現している．

以下に「シリーズ8　10000 E」の特徴の一端を紹介する．シリーズ8全体のラインナップおよび10000 Eの概略仕様については，図1および表1に示すとおりである．

1．高性能化のための先進ハードウェア

1）　バスの強化……命令を実行する基本処理装置（BPU）の内部データバス，システム・バスを従来の32ビット幅から64ビット幅と拡大．主記憶，キャッシュメモリ，演算回路，高速チャネル等，相互間の情報転送時間を短縮．

2）　パイプライン制御方式の強化……命令実行処理を「命令プリフェッチ」，「命令デコード」，「オペランド・アドレス計算」，「アドレス変換とキャッシュ・ヒットの判定」，「キャッシュ・データの読み出し」，「演算実行」の六つのステージに分けた6段のパイプライン制御方式を採用し，従来の4段パイプライン制御方式に比べ命令の並行処理数を増加させ，命令実行性能を向上．

表1　シリーズ8　10000 E概略仕様

諸　元		仕　様
主記憶容量		16～64メガバイト
基本処理装置	パイプライン制御	6段
	キャッシュ・メモリ	128キロビット
	パ　ス　幅	64ビット
	10進演算・乗除算	専用高速演算機構内蔵
RDBエンジン搭載台数		最大2台
IOプロセッサ搭載台数		最大2台
ディスク容量		1.64～32.8ギガバイト
ワークステーション接続台数		最大256台
通信回線数		最大256回線
基本筐体サイズ（mm）		795W×745D×1600H
OS		DPS 10

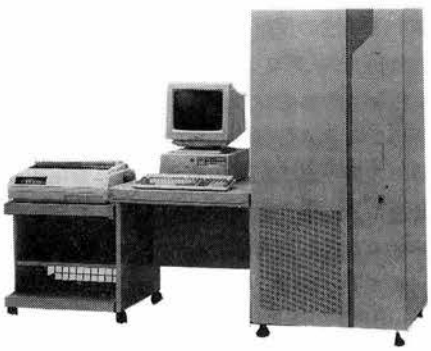


写真1　シリーズ8　10000E

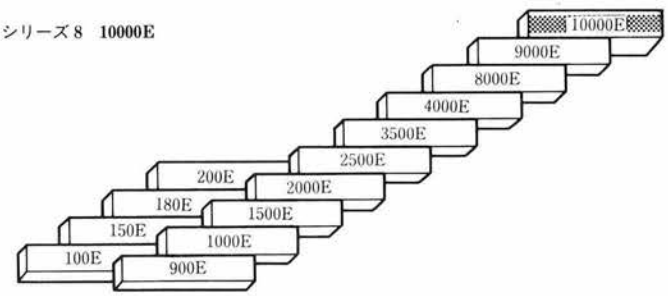


図1　シリーズ8 ラインナップ

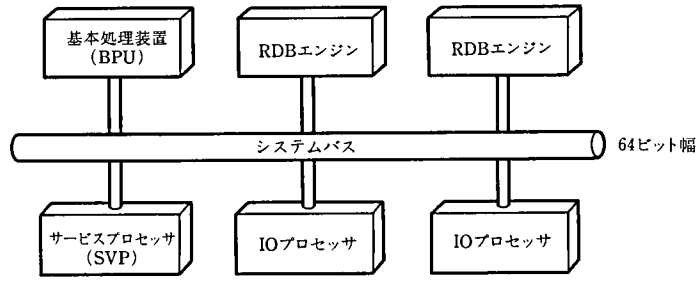


図2　シリーズ8　10000Eのプロセッサ構成

3)　キャッシュ・メモリの強化……命令用とデータ用のキャッシュを分け全部で128キロビットに拡大しキャッシュ・ヒット率の向上を実現.

4)　リレーショナル・データベース・プロセッサ「RDBエンジン」の2台搭載……RDB処理能力をさらに強化するため，従来モデルでその性能向上が実証されているリレーショナル・データベース処理専用付加プロセッサ「RDBエンジン」の2台搭載が可能．すでに作成済みのデータベース検索プログラムやソート処理プログラムをまったく変更せずに「RDBエンジン」を利用する環境で稼働させることができ，処理能力の飛躍的向上が可能.

5)　入出力処理装置「IOプロセッサ」の2台搭載……入出力処理能力をさらに強化するため「IOプロセッサ」を最大2台搭載可能とし，システムスループットの大幅な向上を実現.

2.　大幅に強化したシステム拡張性

1)　ディスク容量……SIS構築の要であるリレーショナル・データベース（RDB）の増大に対応するため，従来モデルのディスク最大容量19.6ギガバイトを大幅に拡大し，ポジショニング・タイム13ミリ秒の高性能9インチ820メガバイト・ディスクを最大40台，32.8ギガバイトまで拡張.

2)　ワークステーション／プリンタ接続台数……ワークステーション／プリンタ接続可能台数を従来の2倍である各々256台まで拡大し，エンドユーザ・コンピューティング機能をさらに強化.

3)　通信回線数……大規模ネットワーク・システム構築に対処するため，接続通信回線数を最大256回線まで拡大．また，ホストコンピュータ10000E上に，ソフトウェアとして「オンライン・トランザクション処理システム」を導入した場合，リモートに端末制御コンピュータのシリーズ8を設置することにより，最大1024台の端末機から予約・受注・照会等の処理を行うことが可能.

4)　ネットワーク機能のオープン性拡充……ネットワークにおけるデファクト・スタンダードであるTCP/IP，OSIを支援することによりU6000等のUNIXシステム，あるいは他ホストシステムとの容易なネットワーク形成が可能．またISDNへの接続も可能とし，ネットワーク機能がより充実.

3.　高い信頼性の保証

1)　サービス・プロセッサ（SVP）……SVPによるシステム立上げ時，およびシステム運用時のシステム各部の状態診断機能と異常発生時のステータス記憶・表示機能を強化することにより，保守員の的確な判断を助け，ダウンタイムの短縮を実現.

2)　基本処理装置（BPU）の演算回路二重化……BPUの演算回路を二重化し，演算を二つのプロセッサで行い，その結果のつき合わせチェックを行うと共に，内部バスのパリティチェックを充実し，BPU内の自己診断能力を強化.

3)　温度異常の2段階検知……温度異常とファン異常を二重に検知．とくにファンの間歇故障に対してはファン異常状態を個別に検知し，ハードウェア専用ディスクに記録することにより保守時間の短縮を実現.

4)　二重化システム……2台のCPUからディ

スクファイルを共有するデュアルディスク機能，ディスクファイルを二重化するミラーディスク機能，LAN経由のマルチワークステーションより随時相手CPUを切替えられるマルチホスト機能等，より安全性の高いシステム構築のための数々の機能を提供．

4．使いやすさをさらに追求，第4世代言語「ECHO-G」，システム開発支援システム「PRODUCE」を強化

1) 第4世代言語「ECHO-G」……日々，変化する情報の要求に即座に対応でき，エンドユーザ・コンピューティングをより対用しやすくするため，オブジェクト指向の第4世代言語「ECHO-G」は，RDBエンジン活用による高速化を実現しているだけでなく，操作ガイダンスとして，プログラム一覧画面や定義体一覧画面，ファイル一覧画面を完備．さらに従来の対話型処理に加え，一括処理も可能にし，より一層エンドユーザ・コンピューティングの幅が拡大．

2) システム開発支援システム「PRODUCE」……システム開発環境については，従来からある「PRODUCE」を大幅に改良し，画面・帳表等のアウトプットのプロトタイピングが容易にでき，エンドユーザの要望に対し即座にまた正確に対応が可能．

以上述べたとおり，「シリーズ8　10000 E」は，もはや汎用コンピュータに肩を並べるレベルまでそのシステム能力を拡充させているが，オフィスプロセッサが目指した'より現場に密着し，より使いやすく'という従来からの柱も確実に継承させつつ，さらに一歩前進させたコンピュータ・システムである．

ユニシスAシリーズ
“モデルS”シリーズ
汎用コンピュータ・システム

1．Aシリーズの設計思想

Aシリーズ汎用コンピュータ・システムは，デスクトップのマイクロAシステムから今回発表の中小型システム“モデルS”シリーズ，さらには超大型機A17シリーズまで統一された設計思想のコンピュータ・システムを規模に応じて自由に組み合わせ利用することができる．さらに統一されたネットワーク思想BNAで各コンピュータ・システムを接続し，企業全体，部門，個人のそれぞれの仕事が最も効率良く行えるような情報処理ネットワーク・システムの構築が容易に実現できる．

Aシリーズは，ユニシスがこれまでこの業界でリーダシップを発揮してきた高水準言語，仮想記憶，システムのモジュール化，作業の容易さ，第四代言語といった領域で引続き先進性を発揮してる．このためシステム構築，保守，運用，要員教育といったあらゆる分野で，投資テストを最小限に押さえることが可能となっている．

今そのユニシスAシリーズの設計思想および資産を継承し発展させ，新たに最先端技術を実装することによって，よりコンパクトに，より高性能に，そしてより高信頼性を実現したA4，A6システム“モデルS”シリーズが誕生した（図1）．

Aシリーズの統一した設計思想は，以下の通りである．

1) 高水準言語をマシン・アーキテクチャに反映，すなわちOSも含めDB/DC等のすべての基本ソフトウェアを高水準言語で作成している．これにより，時代に即した新機能，新ソフトウェアの開発・保守を容易にしている．

2) ダイナミックな仮想記憶，マルチプログラミング，マルチプロセッミングワークフロー制御等ユーザに負担をかけず，ハードウェアとソフトウェアを最大限に活用できる機能を実現している．このユーザ・インタフェースの簡単化により，容易に高度なシステムを効率よく稼働させる使い勝手の良さを提供している．

2．A4，A6“モデルS”シリーズの位置付け

コンピュータを導入する企業にとって最大の関心時は，“変化への対応”である．将来，情報量が飛躍的に増大した際，またビジネス形態が変化した場合，“変化に対応”してシステムをスムーズに再構築できるフレキシブルな性能である．ユニスAシリーズはデスクトップのマイクロAシステムから超大型のA17システムまで，120倍の性能範囲を単一アーキテクチャでカバーし，実行コードレベルで完全な互換性を保証している．ビジネスの進展に伴い情報量が飛躍的に増大しても，

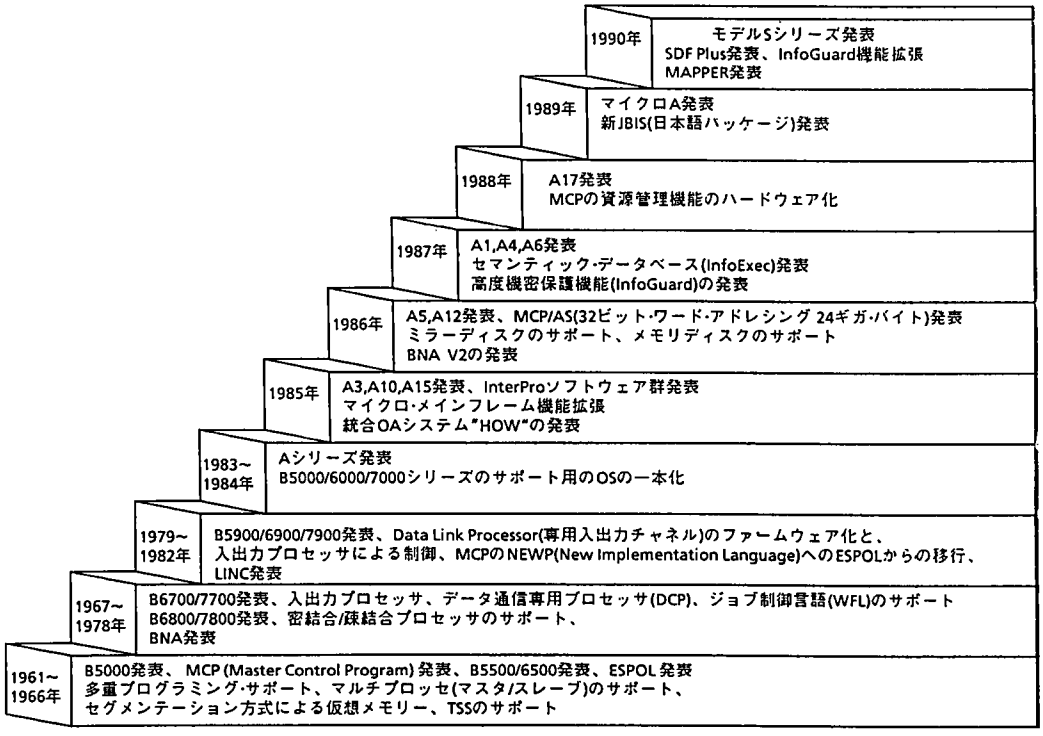


図1 Aシリーズ発展の歴史

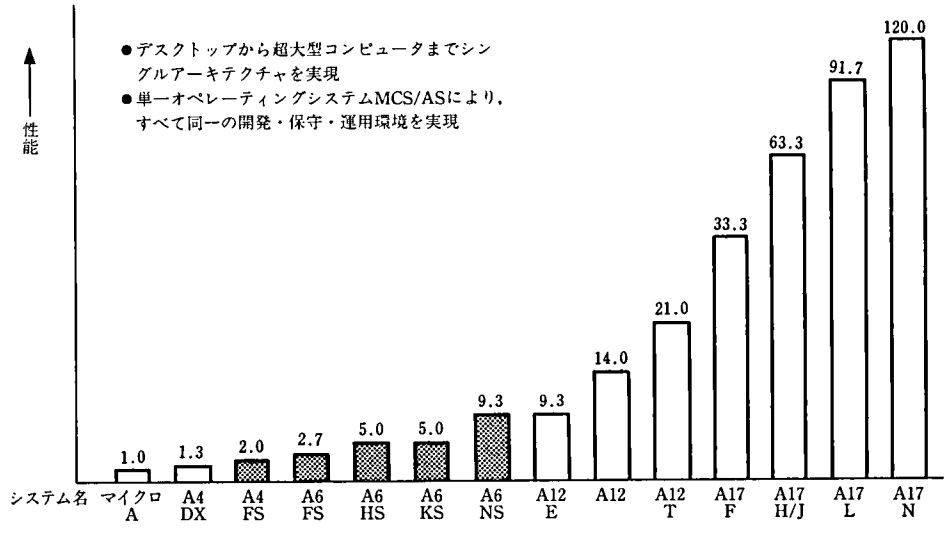


図2 Aシリーズ内でのモデルSシリーズの位置づけ

プログラムやオペレーションの変更を一切必要とせず，容易により高性能なシステムへの移行や分散処理による複数システムへの移行が可能である．“モデルS”シリーズでは，シリーズ内でのフィールド・アップグレードはもちろんのこと，既設のA1, A4, A6の各モデルからのフィールド・アップグレードも可能であり，過去および現在の投資に対する保証をしている．また評価の高い拡張OSであるMCP/ASを中心としたシステムの柔軟性を保証する高生産性ソフトウェア群，呼称プロダクティビティ・センタにより，“変化への対応”に柔軟に適合し，より高度なシステムをより経済的に実現することができる．

とくにシステム開発・保守における高生産性を実現するLINC II，エンドユーザの情報活用力を飛躍的に高めるMAPPERの提供は，柔軟性の高い戦略的情報システムの構築を可能としている．

“モデルS”シリーズは，これからコンピュータを導入し戦略的情報システムを構築しようとしている企業の主コンピュータ・システムとして，またすでに大規模システムを構築されている企業の戦略的部門コンピュータ・システムとして最適である（図2）．

3. “モデルS”シリーズの概要

“モデルS”シリーズは，「A4-FS」，「A6-FS」，「A6-HS」，「A6-KS」および「A6-NS」の5モデルから構成される（写真1）．

これらの“モデルS”シリーズでは，1チップ・プロセサの採用によるコンパクト化，高信頼性を実現している．“モデルS”シリーズのプロセサには，デスク・トップAシリーズであるマイクロAシステムで開発されたSCAMP（Single Chip A series Mainframe Processor）チップの機能拡張版を採用しており（写真2），従来の“モデルX”シリーズではカード2枚で1プロセサを構成していたものが，1チップ・プロセサとしてカード1枚の上に搭載されている．さらにこの1チップ・プロセサをカード1枚に2基実装することも可能となった．

この1ボード当たり2CPUの実装技術により，“モデルX”シリーズのA6-KX（2CPU搭載）を半分の設置面積で実現したA6-HS，A6-KXの筐体に4CPUを搭載したA6-NSモデルが実現できた．これらSCAMPチップを始めとする最先端テクノロジの採用により，より高い信頼性の実現，設備条件の大幅な向上が図られている（表1）．

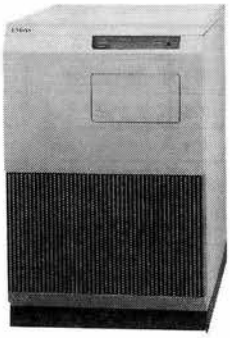


写真1(a)　A4-FS，A6-FS，A6-HSモデル

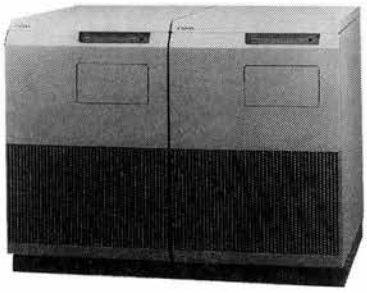


写真1(b)　A6-KS，A6-NSモデル

1)　モデルSシリーズの特徴

①　コンパクトな筐体による省エネルギ設計……高さ，奥行各74 cm，幅46 cm(96 cm)，設置面積0.34（0.69）平方メートルの小さな筐体を実現している．（カッコ内はA6-KS，A6-NSモデル．）

②　拡がる業務に対応可能な最大192メガバイトのメモリを実装……A4-FS，A6-FS，A6-HSで最大96メガバイト，A6-KS，A6-NSでは最大192メガバイトのメモリが実装可能である．

③　ワンボタン・スタート&ストップによる操作……システムの立上げ，停止は一つのボタンを押すだけであり，面倒な操作は不要である．

④　入出力ベース共有機能による耐障害性の高いシステムの提供……入出力ベース共有機能により複数ホストから，入出力チャネル（DLP）が共有でき，各ホストと磁気ディスク装置等の入出力機器を二重の経路で

表1 "モデルS" シリーズのモデル別主要仕様

モデル名	A4-FS	A6-FS	A6-HS	A6-KS	A6-NS
相対性能	1	1.3	2.5	2.5	4.7
プロセッサ数	1	1	2	2	4
キャシュメモリ (KB)	0	96	96×2	96×2	96×4
主記憶容量 (MB)					
最小	12	12	24	24	24
最大	96	96	96	192	192
基本キャビネット数	1	1	1	2	2
I/Oプロセッサ数 (HDP数)	1	1	1	2	2

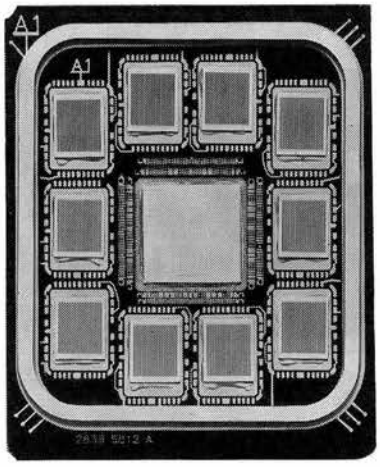


写真2 SCAMP チップの写真 (実物大)

接続でき，信頼性の高いシステムの構築が可能となる．

2) "モデルS" シリーズの最先端技術

① 分割モードの実現による耐障害性の向上，運用の柔軟性の向上……A 6-KS (2 CPU) モデルと A 6-NS (4 CPU) モデルでは，図3のように単一システムとして稼働できるだけでなく，必要に応じて1台のコンピュータ・システムを2台のコンピュータ・システムとして分割して利用することも可能にした．分割モードでは1台のコンピュータ・システムは，それぞれ独立したオペレーティング・システム：MCP/AS のもとで，独立した二つのシステムとして稼働させることができる．分割モードでは，一つのシステムがなんらかの障害により停止しても，他のシステムはそのまま処理を継続することができる．

図4 (a) の例は，分割モードと単体モードを組み合わせてオンライン処理の信頼性向上と事後処理の効率化を実現している例である．

また，図4 (b) は，分割モードと単体モードを用いて，仕事のピークに合わせてシステムの処理能力を有効に活用している例である．

② 日本語対話型メニューによる操作性の向上……操作の容易性という視点では，日常業務の運用からデータベースやオンライン・ネットワークの構築，変更に至るまで，日本語による対話型メニュー方式で実行できることがあげられる．実行するプログラムの名前やシステムの指令を知らなくともメニューから必要な項目を選択するだけで操作ができる．さらにメニュー上の項目で意味がわからないものがあっても，画面上のそのわからない用語の所へカーサを持ってゆき，スペシファイ・キーを押すだけで，その用語の説明を見ることができる．

操作員が操作中に各種解説書を参照することはほとんど不要となった (図5)．

③ 2大4GL, LINC II と MAPPER の連係強化によるシステム開発保守性および運用性の向上……システムの開発・保守面での機能向上では，オンライン・データベース・システムのような基幹システムを，対話形式で開発・保守できる LINC II の機能強化があげられる．エンドユーザ・コンピューティングを支援する MAPPER のデータとして LINC II データベースより抽出，活用することを可能にした．このため，基幹データベースをより容易にエンドユーザが活用できるようになった．戦略的情報システムの基幹システムの構築に LINC II を

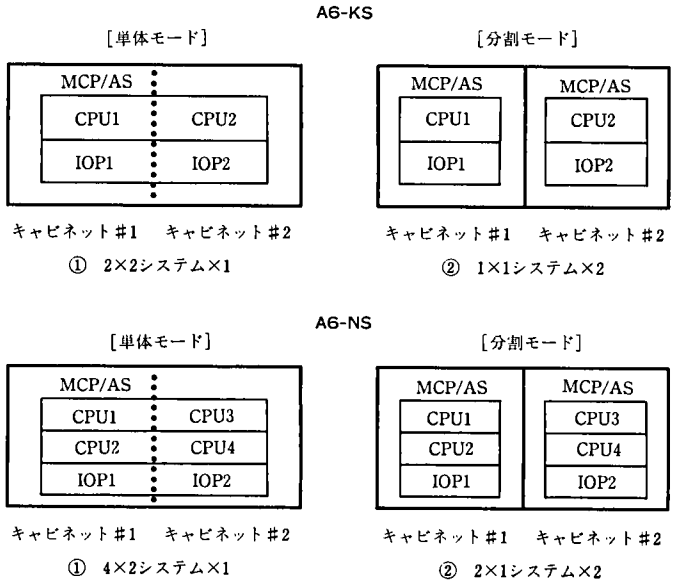
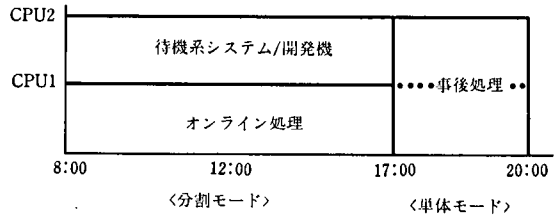


図3　単体モードと分割モード

オンライン処理と事後処理の効率化
- オンライン処理の耐障害性の向上
- 事後処理は処理能力を最大限使用して残業時間を短縮

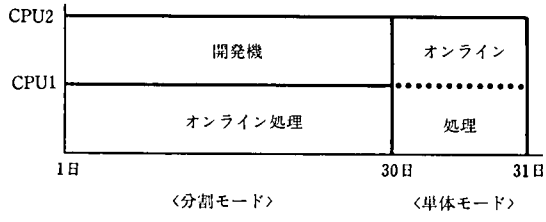
1日の仕事(A6-KSの例)
処理能力



(a)

仕事のピークに合わせて処理能力を有効活用
- 月初，月中の余力利用で開発専用機とオンライン業務用機として利用
- 月末のピーク時には，処理能力を最大限使用して大量のトランザクションをレスポンスの低下なしで処理

1か月の仕事(A6-KSの例)
処理能力



(b)

図4　分割モードの使用例

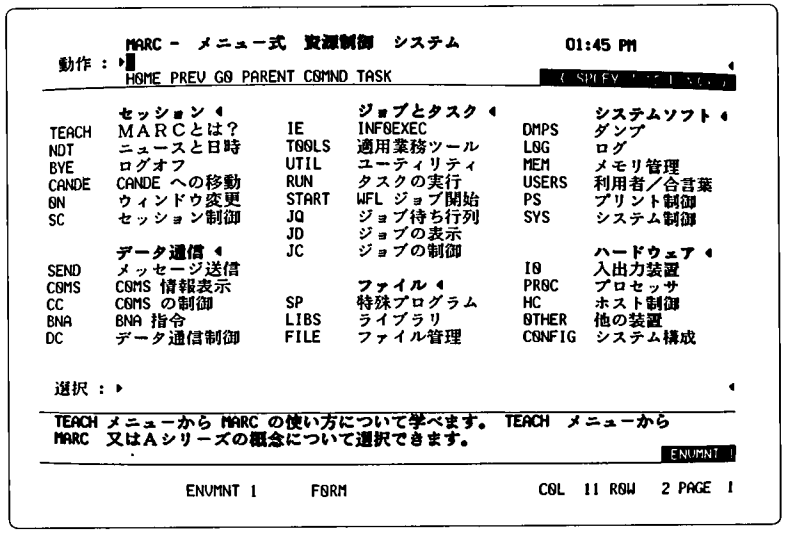


図5 日本語対話型メニューの代表的MARC画面

<プロダクティビティセンター>

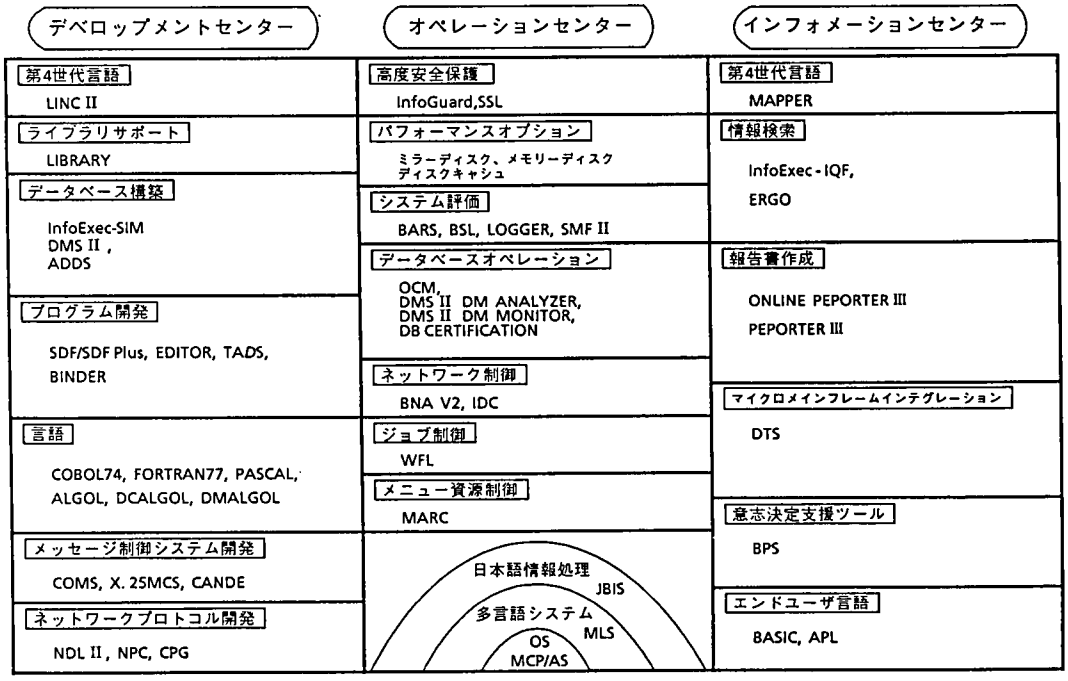


図6 プロダクティビティセンタの体系

活用し，エンドユーザ部門にMAPPERを解放することにより，開発・運用・保守面でより生産性の高いシステムが構築できる．

④ データ通信機能強化による拡がるネットワークの実現……従来の内蔵型データ通信プロセサDC-DLPが，機能拡張されEDC-DLPとして新たに提供される．機能拡張の最大のポイントは，ネットワークを高水準言語NDL II (Network Definition Lan-

guage II）で記述することが可能になったことである．このNDL IIによりユニシス標準プロトコルはもちろんのこと，各種プロトコルに簡単に対応でき，異機種間接続が容易に実現できる．

また，Aシリーズネットワーク・アーキテクチャBNA V2も機能強化された．IEEE 802.3に準拠したCSMA/CD方式のイーサネットLANにより，業界標準のTCP/IPが支援され，ファイル転送，プログラム間通信等が容易に実現できる．

さらに，IBM社ネットワークへの乗り入れを可能とするSNAゲートウェイを各種用意しており，BNA V2とIBM社のSNAとが密結合したファイル転送，および端末相互運用等を容易に実現させることができる．

⑤ プロダクティビティ・センタの強化……情報活用の利便性，およびシステム開発・運用・保守の生産性向上のためのプロダクティビィティ・センタを構成する他の多くのソフトウェア群も，より一層の機能強化が図られている．

とくに，拡張オペレーティング・システムMCP/ASのファイル指定機能の強化，対話型画面設計ソフトウェアSDFを機能強化したSDFPLUSの提供，セキュリティのためのInfo Guardの機能強化等，新しいソフトウェアの提供や既存のソフトウェアの機能強化があげられる（図6）．

Aシリーズの中小型機として新登場した“モデルS”シリーズについて，その特徴，位置付けを紹介してきたが，今後もLINC/MAPPERの機能強化，およびその連携強化，各種ワークステーションとの連携強化，FAX，音声，高度情報処理網ISDN等のマルチメディア対応等の機能強化が予定されている．

■　■

高木晴夫，小坂　武著

SIS　経営革新を支える情報技術

日本経済新聞社，B 6 判，290 pp.，
1990，1,700 円

今，戦略情報システム〝SIS〟ブームと言われ，どの書店でも各種 SIS に関する本が順調に売れていると聞く．何がこの SIS ブームを支えているのか，その背景をここで少し考えてみたい．

今日，経営者の課題の一つに人手不足があげられる．これは各社とも人材育成に手落ちがあったわけではなく，今のたて割り組織では，急激に変化している市場に対応するシステム構築ができる人材を探してもいないということであろう．さらに今の企業は〝始めに物ありき社会〟にあってマネジメントの基礎が育成されており，〝始めに顧客ありき〟時代に入った今では十分に対応できがたいことが底にあるのではなかろうか．ややもすると戦略情報システムとか，マーケティングとかいう言葉に対し，自分のものとして考えるより，学問の一部とみて離見するフシがあったようにも思える．

しかし成熟社会に入り，消費者の価値観・多様化現象が市場が廻り始め，商いの常識すら大きく崩れ出し，企業に，組織に，人間に，インパクトを与え，改めて経営見直しの一歩として研究が始められた結果が SIS ブームの背景ではないだろうか．

本書のはしがきで，著者はこのように述べている．「21 世紀へ向けて経営は，革新による企業間競争を戦わねばならない．従来の延長である低コスト化戦略だけでは勝ち目がなく，絶えず経営革新を進め事業の質的転換，新規事業の創出そして競争範囲を拡大する戦略が要求されている．」

このような革新による経営戦略を実行していくことこそが，21 世紀への成功を確実にする．さらに序章でも企業の都合で「良いものを安く売る」という経営だけではやっていけない，とも．

この歯切れの良さ，切り口のシャープさは全章に共通して流れ，現在の市場変化に対する洞察の深さとその自信の程が脈々と伝わってくる．発想の転換・創造的発想・質的差別化，これらを目耳にして久しい．しかし，具体的に何をどのようにすべきかに苦労している経営者に，何故かを理解させ，大きな自信を与えてくれた本と位置付けても過言ではあるまい．

本書の目的「SIS の強さの本質は何か」，「その強さを発揮できるように SIS をいかに構築するか」である．決してコンピュータだけのものではなく，完全に変わってしまった市場対応策として他社に先駆け作り上げる企業そのものを，SIS と位置付けなければならないことを事例で説明している．さらに成功事例の中で企業の本当に強い所（SIS の強さ）はどこに特徴があるかを述べて，SIS は企業改革であることから投資も高く効果も長期的にしか現われないため取り組む時の姿勢に十分なる注意を促している．

I　SIS その強さの本質

第 1 章　SIS への発展のプロセス

1. 初期 DP 時代　「点」としての発達
2. 拡大 DP 時代　「線」としての発達
3. MIS 時代　「面」への広がり
4. EUC 時代　「立方体」への広がり
5. SIS 時代　「四次元」への広がり

ここでは，情報システムの発達過程を歴史的背景により浮き彫りにし，適用分野の広がりという観点から捉えている．とくに EUC 時代までを三次元の広がりとし，SIS は時間の概念を加えて四次元と位置づけ，速さとつながり方のある企業活動で業務が拡大すると，新商品の開発やマーケティングの変化が激しい今日的環境に，いかに優位であるかが述べられている．

第 2 章　価値創造のプロセス

1. 環境の変化に対処する企業経営
2. EUC 時代までの企業組織
3. SIS 時代の企業組織
4. 価値活動の連鎖と SIS
5. SIS の展開
6. SIS の機能と人間軸への影響

70 年代の高度成長期において，企業は規模の経済で進んだ，そして成熟期に入り時代が大きく変わった．これは，従来までの市場の見直しを根本的に変えざるを得ないことになり，従来の常識では環境がより見にくいものとなり，各企業は対応

に苦慮する．その結果，それまでの組織に対する考え方を改めることとなる．ここからSIS時代の企業組織論・価値活動・価値連鎖の創造，その価値活動の連鎖のすべてに情報技術を浸透させる．業務のつながり方をきめ細かく説明しており，そこから「人間軸」という概念を導き出している．著者が述べたい大きなポイントがここで幕が開くわけである．

第3章　SIS構築のポイント

1. 情報システム計画の位置付け
2. 情報システム計画の発展プロセス
3. 従来の代表的な計画方法
4. 情報システム計画方法の分類
5. 新しい情報システム計画の要件
6. 人間軸からの情報システム計画

ここで強さの本質が生きるSIS構築方法を学ぶ前提として，システム計画の発展プロセスを述べ，計画方法の代表的なものとして，BSP法，CSF法を紹介している．

このCSF法がコミュニケーションを志向したものの一つで，ユーザに立脚した方法と認めながらも経営戦略や業務体系を構築していないことを述べている．また浸透と拡散による企業の分類では，分類することそのものの大切さを説き明かしてくれている．

そして「従来の情報システム計画の実施では，心理的抵抗の問題を正面から扱う必要はなく，組織上の権限を用いるだけで十分であった」と人間軸の不在を立証している．そして「SISの成功は人間軸と技術軸の積」と簡にして要を得る言である．

II　SIS構築の具体的方法

第4章　人と組織への接近

1. 人間軸の世界
2. 組織文化の政治
3. 組織文化の政治の基本要素
4. 「意味の体系」の再編成・人間軸の活動プロセス
5. メタ認識による創造破壊

前述のように，本書が他と異なるのは「人間軸」，「技術軸」という概念を打ち出しているところであろう．

その意味から，この章は著者が一番読者に訴えたい部分である．価値観が多様化する現在，対応策に努力されている経営者にとっては，「何か変化が起こると発生する抵抗」に接し，必ず明解に理解できるであろう．組織文化については「意味の体系」として人々の心に共有されているとし，さらに組織文化の複合的特徴に言及し組織の要因を分類し，タスク・技術・人間を取り上げている．SISの構築とは〝技術〟を変化させることであるが，組織メンバの心の中にある意味の体系にとっては拒否反応が生じる．これが社会的慣性である．このクダリは大変参考になると同時にマネージの大きなヒントになるのではなかろうか．

さらに著者は意味の体系と文化政府について，規範，価値観，パワーの要素を上げ概観している．なかでも規範における「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，その実施に対する意欲も高くなる」，価値観について「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，その有益性を見落としている」と個の社会で忘れかけている人間社会の大切さを申し述べている．

きっとこの章が心に残る章とする読者が多いのではなかろうか．そして意味の体系の再編成メタ認識と進むが，読者各位には是非熟読してもらいたいところだ．

第5章　人間軸の展開

1. 解凍・移動・再凍結の方法論
2. 計画の関係組織と活動展開
3. 「解凍」のステップ
4. 「移動」のステップ
5. 「再凍結」のステップ

前章で学んだ意味の体系を作り変えるに当たり，解凍・移動・再凍結の3ステップをどのような活動で進めるかを述べている．私自身この章からは組織内における己れに対し常に反省し，見つめることの大切さを学んだが読者はいかがなものだろう．

第6章　経営戦略としてのSIS

1. 「技術軸」の意味
2. 従来の情報システムからの引き継ぎ
3. 経営戦略を具現化するSIS
4. 経営戦略の意味
5. 戦略立案とSIS計画の接点
6. 戦略ドメインの考え方

7. 技術軸の活動ステップ

第7章 技術軸の展開

1. 戦略立案
2. 業務機能展開
3. 情報定義
4. 情報システム展開
5. 計画策定

第6章，第7章で「技術軸」の概念を前述の人間軸と比して説明している．なかでも今までのピラミッド型組織と「ぶんちん型」組織のメリットおよび，その必要性が述べられている．

ここではデータベースが企業で活用するということの本当の姿と，その時の組織のあり方を述べ，企業の競争優位の源泉にふれ，価値活動の連鎖の再構成のためには新しい業務機能体系を設計しなければ価値活動の連鎖が実現できない理由がよく理解されるであろう．

次いで経営戦略の意味，戦略立案とSIS計画の接点・ドメインの考え方・技術軸の活動ステップと進み，前章人間軸の解説と比して筆運びがハイテックに感ずる章でもある．

第8章 人間軸と技術軸の統合

1. 人間軸と技術軸による平面
2. 成功を危うくする二つの抵抗
3. 組織階層を降りるタイムラグ
4. 人間軸の責任者

前述の通り，本書は10章によって構成され，第8章で人間軸と技術軸の統合がなされる．

ここでは今まで学んできた二つの概念が統合され活動し，成功に至るための注意点である．とくに失敗のケースを面白い例で説明され大変わかりやすく，失敗しないためのキメ細かい説明は感動的でもある．

III ケース研究・成功への道

第9章，第10章はケース研究として成功したケース，失敗したケースがそれぞれ一例ずつ書かれている．これらは実例であるためか迫力に富み，大変人間的にドロドロした内容が読む者を引きつける．

この章は，それぞれ「人間軸と技術軸からの事例の分析」を先に読み，その後全体を再読することをすすめたい．オプトニクス社のケースでは決して完全に成功したというのではなく，反省を必要とするところが多く残っていることからSIS構築のむずかしさを十分教えられると同時に，何を注意すべきかを考えさせられるということから参考になるケースであろう．

本書は2名の著者により，リポートされており，高木晴夫氏は慶応義塾大学ビジネススクールで経営学を専攻し，経営者の認識力や組織革新について研究されている．小坂武氏は日本ユニシスの情報システム専門家としてMIS. DSSの開発を多く手がけている．装丁も美しくでき上っており，金色のSISのロゴも戦略的ですらある．多くの経営者の座右の書として，マーケティング担当者のバイブルとして，是非お読み戴きたい一冊である．

（流通マーケティング部　横須賀亮一）

S. shlaer
S. J. Mellor 著

オブジェクト指向システム分析
Object-Oriented-Systems Analysis
啓学出版，A5判，xii＋179 pp.，
1990，2500円

ソフトウェアの新しいパラダイムとしてオブジェクト指向という言葉をよく聞くようになった．Smalltalkに代表されるような「オブジェクト指向プログラミング言語」から，最近では「オブジェクト指向設計」，「オブジェクト指向データベース」，「オブジェクト指向分析」といったようにオブジェクト指向の範囲が広がってきた．しかしながら，プログラミング言語以外の分野でのオブジェクト指向に関する参考書はあまり出版されていなかったようである．

本書は数少ないオブジェクト指向分析（OOA）に関する書籍であり，原題は“Object-Oriented systems Analysis”で，1988年の出版である．その日本語訳が1990年の2月に出版された．

現時点でOOAは確立された技術ではなく，さまざまな提案が行われているが，OOAはオブジェクト指向という観点で，少なくとも二つの技術的な流れを受け継いでいるように思える．

第1は，オブジェクト指向プログラミング言語からの技術的な流れである．以下のようなキーワードでその特徴を表せる．

・データと手続きのカプセル化

・メッセージ交換による処理メカニズム
・クラスによる汎化階層と継承

第2としては関係データモデルや実体関連モデル（ERモデル），意味データモデル等の情報モデリングからの流れである．この分野でのオブジェクト指向的な特徴は以下のようになる．

・実世界における実体の直接表現
・複合オブジェクトの直接表現
・オブジェクト間の汎化関係の直接表現

したがって，OOAで構築するモデルとしては以上の2種類の要素のミックスされたものがイメージとして浮かび上がってくる．すなわち，データ（属性）と手続き（振る舞い）が一体となったオブジェクト間の関係や，メッセージ交換を主体として実世界を抽象化したものである．

以上のようなことを考えながら本書を読み始めた．本書は以下のような章から構成されている．

半ばまで本書を読み進んでいって，「ちょっと違うぞ」という気持ちになった．本書のほとんどが情報モデルの構築に関することで占められている．オブジェクトのデータ構造のモデル化であり，データだけでなく手続き（振る舞い）も持ち，独立したアクターとしてのオブジェクトのイメージがいっこうに出てこないのである．

また，本書で述べている情報モデルも言葉としてはオブジェクトという言葉を使い，スーパークラス/サブクラスの表現等を含んでいるが，ほとんど関係データモデルと同じである．各オブジェクトは表形式の構造で表現され（すなわち，属性は正規化する），各インスタンスや関係を識別するために属性に識別子（キー）を人為的に導入している．さらに，複数関係を表現するために，関連付けオブジェクト（もちろんこれも表形式である）を導入する．

情報をモデル化することはもちろん重要なことであるし，それを関係データモデルで表すことも実現を考えると実際的なことかもしれない．しかしながら，オブジェクト指向の香りがしないのである．

そこでもう一度，分析とは何であるか考えてみると，さまざまな要素があるが，実世界を抽象化し，モデル化することであると言ってもよいと思う．それでは分析におけるオブジェクト指向とは何であろうか．次の二つの意味としてとらえることができる．

1)　分析作業を行っていく方法がオブジェクト指向である．
2)　分析作業で構築するモデルがオブジェクト指向である．

このように考えながら本書の第9章を読んで，なるほどと思った．少なくとも上記の1)において，本書が進める分析のやり方はオブジェクト指向なのである．

情報モデルの上で明らかになったオブジェクトごとに，そのオブジェクトの状態の変化を洗い出し，状態モデルを作るのが2番目のステップである．状態変化は何らかのイベントによって引き起こされる．イベントにはそのオブジェクトの外部（他のオブジェクトやシステムの外側）で発生するものと，内部で（時間経過等によって）発生するものがある．また，イベントによって引き起こされた状態変化の結果別のイベントが発生する．このイベントと状態変化の関係をオブジェクトごとに状態遷移図でモデル化する．当然複数のオブジェクト間でのイベントのやりとりも出てくる．

3番目のステップとしてはプロセスモデルを作ることである．イベントによって状態変化が引き起こされると，何らかのアクションが取られる．オブジェクトごとに各イベントに対応してとられるアクションを洗い出してデータフローダイアグラムで表現したものがプロセスモデルである．

このように情報モデル・状態モデル・プロセスモデルの3種類のモデルで実世界を抽象化しようとしている．ここで使われる道具としては，データフローダイアグラム・状態遷移図・エンティティ関連図等，目新しいものではない．すべて，構

造化分析で利用されてきたものばかりである．そういえば，著者の一人であるS. J. Mellor は P. T. Ward との共著で1985年にプロセス制御等のリアルタイムシステムへの構造化分析技法の適用を提唱した著書“Structured Development for Real-time System”を表している．

構造化分析と同じ道具を使っていても本書の述べる分析のアプローチは明らかに異なったものであり，オブジェクトを中心に分析を進めていくものである．また，本書には簡単であるが，分析フェーズ以降のプロジェクトの各フェーズへのモデルのつながりに関しても述べている．

本書でいう分析以降のフェーズとは以下のものである．

- 外部仕様フェーズ
  - システム境界の定義
  - 外部インタフェースの定義
- システム設計フェーズ
  - ソフトウェアアーキテクチャの設計
  - データ構造設計
  - プログラム分割
- 実現フェーズ
  - データ収集
  - プログラム設計
  - コーディングとテスト
  - 統合と受け入れ

本書ではオブジェクト指向分析によって先にあげた三つのモデルを構築するよう提唱しているが，その中で本書は情報モデルを中心に解説したものである．他の二つのモデルに関しても，執筆中とのことなので全体の評価はその出版を待ちたい．

また，オブジェクト指向ということを離れて本書を読んだ場合，データモデリングのわかりやすい入門書として読むこともできることをつけ加えておく．

（システムプロダクト本部
ソフトウェア三部　山下淳一）

■ ■

小川裕彦の**南王運送(株)におけるプロトタイピングによるシステム開発事例**は，「人の動き」をタイムリに捉え，有効に人材を活用していくことを目的に，一つの独立したシステムとして出退勤処理システムを位置付け，LINCを開発支援ツールとしてプロトタイピングによる開発に挑戦した事例である．

シャープ(株)電子機器事業本部ビデオ事業部では，生産管理業務に携わっているエンドユーザが新たに柔軟な生産管理システムを自ら開発した．青木好治らは，**シャープ(株)におけるエンドユーザ中心型生産管理システムの構築**の中で，開発期間の短さや使用端末の制約等各種課題をMAPPERを使うことにより克服し，大きな成果を挙げたことを報告している．

情報システムの適用分野の拡大とともに，ソフトウェア開発のあり方が改めて問われている．ソフトウェア工学をはじめ，いろいろな形で研究・開発が進められ，その成果は徐々にではあるが実践の場で実を結びつつある．倉坪康広は**LSA (LINC Systems Approach)の紹介**の中で，LINCによるシステム開発方法論であるLINCシステムアプローチと，概念システムモデル構築技法であるオブジェクトモデル技法について紹介している．

MAPPER出現の当時と比べて，社会環境は著しく変化し，その中で情報は資源としての地位を確立してきた．さらに，企業を存続させるためには，情報を正しく管理することが必要になってきた．このように変化する環境の中ではMAPPER自身もさらに進化する必要がある．柳沢勝は，**拡大を続けるMAPPERの世界**の中で，MAPPERが今後どのように進化を遂げようとしているかを述べている．

☆

掲載論文梗概

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 10　No. 2　(No. 26)

発行日　平成2年8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)585-4111 (大代表)
印刷所　三美印刷株式会社