データベース・システム　72−11
（1989. 7. 20）

オブジェクト指向型ＤＢ応用簡易言語の提案

An Object-Oriented Language for Office Information Systems using Relational Database

長岡満夫　進藤重平　天水昇
Mitsuo Nagaoka　Jyuuhei Shindoh　Noboru Tenmizu
日本電信電話株式会社
NIPPON TELEGRAPH AND TELEPHONE CORPORATION（ＮＴＴ）

あらまし　従来、業務ソフト開発の生産性向上を狙いとして第４世代言語（４ＧＬ）が開発されている。リレーショナル型データベース上での業務ソフト開発が主な適用分野であるが、ＳＱＬ、ウィンドウシステム、ＭＭＬの標準化・進展を踏まえ業務全般に適用できる簡易言語のニーズが高まっている。しかし、管理業務（情報系業務）と現業業務（基幹系業務）までの幅広い範囲に跨る業務分野で生産性向上を図る４ＧＬはまだ存在しない。

これは、４ＧＬそのものが、ソフトウェア工学的な一般的な考え方（ex.部品化、オブジェクト指向、カプセル化等）に基づいた生産物ではなく、ＥＤＰ部門、システム部門に携わるプログラマがある特定の業務分野向きに開発したアイデアプロセッサである場合が多いことが１つの要因であると考えられる。従って、ある分野の業務に有効であった言語も他の分野の業務に対しては、弊害（性能、資源等への影響）をもたらし、プロトタイピングどころか、生産性の低下（チューニング期間）を招く恐れもある。

我々ＮＴＴに於いても社内業務システムの開発に当たって、業務に精通した社員によるソフトウェア開発を目的に、独自に簡易言語を開発し、システムへの適用を進めてきたが、本格的な適用にあたり、考慮しなくてはならない項目の中でも、特にソフトウェアアーキテクチャの観点からオブジェクト指向の考え方を業務システム全般に適用することが、生産性向上のキーポントになると考えた。

本報告では、従来４ＧＬへの要求条件と我々の解釈するＰＣ／ＷＳとＨＯＳＴに分散したＭＭＬアーキテクチャの下でのオブジェクト指向ＤＢ応用簡易言語の考え方、言語仕様への反映、及びその評価を述べる。

Abstract　This paper describes a scheme of office system's applications and an object-oriented language for these applications , which are micro-mainframe link type applications and use vertically distributed relatinal databases. Office information system's applications are divided into two categories,which are managerial applications and operational applications,but higher languages (ex.fourth-generation languages) also are divided into these two categories. However,the impact of the difference between managerial and operational applications upon higher languages may be absorbed by an object-oriented scheme for both applications. The scheme covers windows,databases,descriptive grammar and software development method.

１．はじめに

業務システム開発用簡易言語へのニーズは、業務ソフトの開発生産性向上のみではなく、システムアーキテクチャの観点から分散化によるシステム経済性にも及んでいる。４ＧＬの開発は、このニーズの解決方法の１つとしてボトムアップ方式でインプリメントされつつある。〔1〜5〕

この様なニーズを実現するためには、ウィンドウインタフェース、データベースインタフェース、業務プログラムインタフェース等をある程度標準的仕様として規定する必要がある。また、業務特性の面からも管理業務（情報系業務）と現業業務（基幹系業務）があるが、これらの間の根本的な差異明確にする必要がある。〔8〕 さらに開発者のユーザ層拡大に必要な業務構築パラダイムを明確にする必要がある。〔6,7,9〕

我々は各要素に関し、オブジェクト指向の考え方が解決策の１つのキーポイントになると考え、従来我々が開発してきたＶＧＵＩＤＥの適用性評価結果を踏まえ〔10〕て、基本的な考え方、言語仕様基本概念、及び言語仕様の構成要素とその関連について述べ、具体的な適用例を基に評価結果を述べる。

２．背景

一般にＯＡ業務ソフト開発は、従来管理業務（情報系業務またはエンドユーザコンピューティング業務）開発と現業業務（基幹系業務またはトランザクション業務）開発の大きく２種類に分類される。各々の業務の特徴、相違は表１．に示す通りである。〔7〕

表１．より、従来適用してきたＤＢ応用簡易言語の評価結果、考察する観点としては以下の項目が挙げられる。

①．言語の非手続きと手続き
②．情報資源の構造／形式と操作の隠ぺい
③．情報資源からの業務の独立
④．ユーザ指向の業務起動インタフェース
⑤．業務構築に関する一貫性

上記①．〜 ⑤．に関して、基本的な考え方を明確にするとともに、主にオブジェクト指向の考え方を基に考察する。

表１．ＯＡ業務の分類と特徴

業務分類	業務内容	業務の特徴	代表的簡易言語
現業業務（基幹業務／トランザクション業務）	受注業務 販売業務 設備管理 経理業務	ﾃﾞｰﾀﾍﾞｰｽ構造複雑 ﾚｺｰﾄﾞ単位/ｶﾗﾑ単位の更新／参照 情報表現の対象ﾒﾃﾞｨｱは目的別に特化 情報は業務間でﾄﾗﾝｻﾞｸｼｮﾝを介して加工	LINC-Ⅱ (UNISYS) TELON (Pansophic) MANTIS (Cincom) 等
管理業務（情報系業務／エンドユーザコンピューティング業務）	名簿管理 経営管理（商品売り上げ管理等） 企業情報	ﾃﾞｰﾀﾍﾞｰｽ構造簡単 ﾚｺｰﾄﾞ群単位／ﾃｰﾌﾞﾙ単位の参照／更新 情報表現の対象ﾒﾃﾞｨｱは多様 情報は業務内で閉じて加工編集	FOCUS (Information Builders) MAPPER (UNISYS) 等

（注）生産性：管理業務へ適用した簡易言語は、現業系業務への適用時２倍程度生産性が低下〔10〕

3．基本的な考え方

業務ソフト開発の生産性向上の基本的な考え方は、システムの分散化と業務記述言語の高度化である。２．で述べた観点をＯＡ業務の相違から見ると下表の通りとなる。

観　　点	現業業務	管理業務	要求／考慮項目
手続きと非手続き	手続き	非手続き	非手続き重視 構造不一致等の 対処が必要
構造／操作隠蔽	全て分離 （性能上）	一部隠蔽 （ｶｰﾄﾞ等）	
情報資源独立性	全て意識 （性能上）	一部独立 （SQL等）	構造不一致，VIEW 等の対処が必要
ﾕｰｻﾞ指向業務起動	ﾎｽﾄ集中	分散（ﾊﾟｿｺﾝ主導）	分散化とｲﾍﾞﾝﾄﾄﾞﾘﾌﾞﾝMMI
業務構築一貫性	重　要 保守ｺｽﾄ 増大	ｱﾄﾞﾎｯｸ業務で保守ｺｽﾄ少	業務構築のｵﾍﾟﾚｰｼｮﾝｽﾄｰﾘが必要

↑
オブジェクト指向の考え方

（１）．非手続き的記述仕様と手続き的記述仕様

情報資源に関する記述形式に関しては、極力非手続き的な記述仕様が可能でなくてはならない。即ち、業務システム内の全ての資源の定義は非手続き形式の定義体を支援する必要がある。

スクリーンペインタ、帳票ペインタ等の記述仕様が世の中の動向であるが、ダム端末エミュレータを対象として画面電文プロトコルとして規定されている場合が多く、ホスト集中型の考え方である。ＭＭＬに代表される垂直分散形態では、ウィンドウインタフェース、ＤＢインタフェースを規定して、管理業務（情報系業務）をＰＣ／ＷＳに完全にオフロードし、小回りのきく情報加工、分析、編集が行える必要がある。また、現業業務（基幹系業務）に関しても、業務の見直し、運用方法の変更に伴って、絶えずエンハンスされるのが通常である。従って、業務仕様に沿って、汎用なウィンドウマネージャから切り離した業務画面仕様を規定し、記述できる非手続き言語インタフェースが必要となる。

本言語仕様では、特にこの言語インタフェースを「論理画面定義体」と呼ぶ。

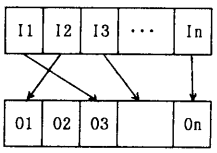
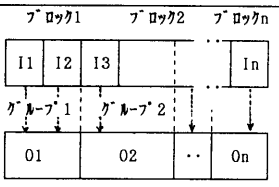
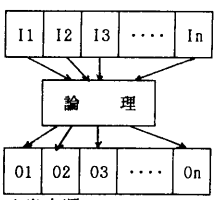
また、情報資源の参照、更新に関しては、システムが自動化可能な部分は極力自動化する必要がある。このために、情報資源をオブジェクトとして抽象化して捉え、外部キー、構造不一致などの問題を考える必要がある。構造不一致に関しては、表２．に示す様に分類されるが、一般に現業業務（基幹系業務）で問題となるのは脈絡不一致である場合が多い。また、構造不一致の問題としては、一般にユーザビューの一部で解決すべき問題と、業務モデル共通に考えられる問題の２点に分類できる。ユーザビューに起因する構造不一致の問題点については、個々の業務によって構造不一致の対処方法が異なる。

この様に、構造不一致の問題から簡易言語と言えども、非手続き言語で全て記述できるとは考えにくく、手続き言語インタフェースは残しておく必要がある。（一般にこの手続き、非手続きの言語仕様の分類は性能問題、適用分野等に大きく影響している。世の中の大半は混在型である）

なお、ＳＱＬに関しては、非手続き性、ＭＭＬでのコマンドインタフェースとの統一性からカーソル処理インタフェースは採用しない。

表２．　構造不一致とその対処方法

（注）構造不一致の種類についてはM.Jackson等による　（注）◎:必須、○:ｱﾄﾞﾎｯｸに使用、△:殆ど使用されない

種　類	意　味	事　例　等	現業業務(基幹系業務)	管理業務(情報系業務)
順序不一致 （入り交じり 不一致）		・テーブルをあるカラムでソートして出力 ・入力トランザクションファイルを分類コードによって出力	・SQLでのORDER BY ・プログラム分割 （分類ｺｰﾄﾞを扱う専用ﾙｰﾁﾝ化） （命令自体は非手続きに記述） ◎	同左 ◎
境界不一致		・テーブルをある値でｸﾞﾙｰﾋﾟﾝｸﾞして出力 一般のCOBOL等では、ｺﾝﾄﾛｰﾙﾌﾞﾚｰｸによる制御*を行っている。 (*ﾃﾞｼﾞｮﾝﾃｰﾌﾞﾙ等)	・バッチ系の業務 特にﾚﾎﾟｰﾄ出力 ・画面表示編集 （上記は画面定義体等に非手続きで定義） ○	同左 ◎
脈絡不一致	➡入力順  ➡出力順	・商品ﾚｺｰﾄﾞと受注ﾚｺｰﾄﾞによる受注毎の売上額一覧作成等の従来ﾃﾞｰﾀﾍﾞｰｽの加工編集、業務処理に見られる	・テーブル間での外部キーでの結合 （テーブル間ﾄﾗﾊﾞｰｽ） ➡JOINは使わない ➡ｱｸｾｽﾙｰﾄが固定／浮動の場合あり 一般のCOBOL等では、配列(添え字付き)変数による処理となる。 ◎	・テーブル単位のＪＯＩＮ演算等 ➡性能が問題 ➡一意識別詞の必要性 ➡ユーザが陽にJOIN演算意識 △

（２）．情報資源の構造／形式と操作の隠ぺい

非手続き性を重視すると、情報資源の表現形式に関しても、"メディア"として抽象化し、基本操作をメディアを介して（プログラマが記述しなくても）利用できることが必要である。

情報の表現形式に関しては、内部表現と外部表現に分離してハンドリングすることが必要であるが、従来は情報の外部表現に対応してプログラム毎にハンドリングする必要があっ

た。さらに、内部表現から外部表現への対応は、（1）．でも述べた様に、各論理画面対応に個々のＡＰ（アプリケーションプログラム）に記述（またはＣ言語の構造体として記述）する必要があった。（図１．参照）

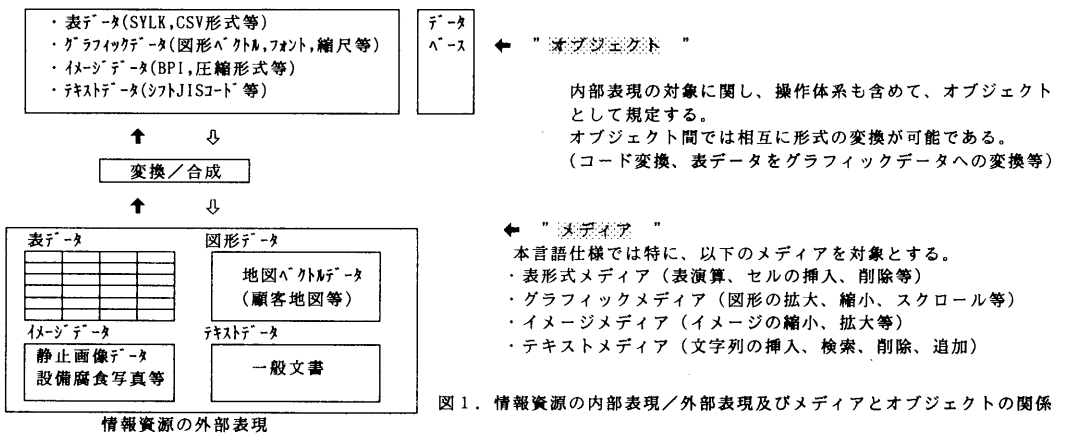


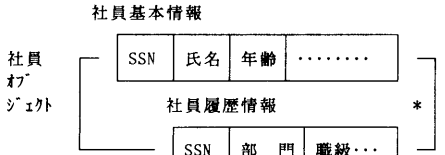
図１．情報資源の内部表現／外部表現及びメディアとオブジェクトの関係

本言語では、情報の外部表現形式及びその操作をカプセル化したものを「メディア」と表現する。
また、内部表現と対応する操作体系をカプセル化し、「オブジェクト」と表現する。

（３）．情報資源からの業務の独立

データ所在場所（ディレクトリ）、アプリケーションが扱うデータ構造（ディクショナリ）に関しては業務プログラムとは極力分離して考えることが必要である。
従来のデータベースの観点らのデータ独立性の他に、プログラム言語からのデータ独立性を考慮する必要がある。我々は、データの存在場所に関する一元的な定義を「ＶＩＥＷ」定義として仕様化し、アプリケーションが扱う構造については「オブジェクト」定義として仕様化する。
つまり、「ＶＩＥＷ」定義では、メモリテーブル、ローカルＤＢ、リモートＤＢ等のアプリケーションから見た存在場所に関する定義を一元的に可能として、性能の向上対処、システムからみた分散、運用条件に関する規定条件も吸収可能とする。
また、「オブジェクト」定義では、以下の様な場合には顕著な効果がある。

①．「履歴情報」を扱う場合のオブジェクト
社員の基本情報と社員の履歴情報は処理効率、運用性等から一般には分離した情報構造をとることが多い。
一方、アプリケーションロジックから参照する場合には社員の基本情報と履歴情報を１つのオブジェクトとして扱う場合が多い。



（注）＊：1:n等のﾏｯﾋﾟﾝｸﾞ関係（ｵﾌﾞｼﾞｪｸﾄ定義体の中で定義する），SSN:SOCIAL SECURITY No.

図　２．履歴情報を扱うオブジェクトの例

②．図形、マルチメディア等を扱う場合

地図や設備図面等の図形情報では、一般にレイヤ構造で表現される場合が多いが、このレイヤ構造に対応してレイヤ対応の図形を１つのオブジェクトとして扱う場合が多い。
図　３にその代表的な例を示す。

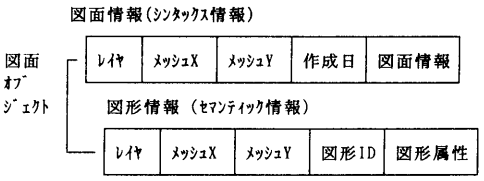


図　３．図形情報を扱うオブジェクトの例

また、オブジェクトには上記の様に複合的なオブジェクトの他に、アトミックオブジェクトが存在する。アトミックオブジェクトはそれ自体で意味有るもので意味的にそれ以上に分割できない最小単位である。一般にこの様なアトミックオブジェクトに関しては、その取り得るデータタイプと処理仕様があり、これは外部表現（メディア）にマッピングする仕様まで規定するものとして定義することが必要である（なぜならば、システムイベントとして取り扱うオブジェクトはこのアトミックオブジェクトであることが多いからである）。
即ち、アトミックオブジェクトはシステム内で業務に依存しない共通的なオブジェクトである。
さらにアトミックオブジェクトに関しては、以下の様な継承機構を設ける必要もある。
即ち、システム、業務ユニット、プロセス、メディアの業務管理単位の階層関係の間で継承される必要がある。なお、アトミックオブジェクトとしては各種コード情報、日付け等があり、外部表現でのメディアと対応付ける。
（図４．にその概要を示す）

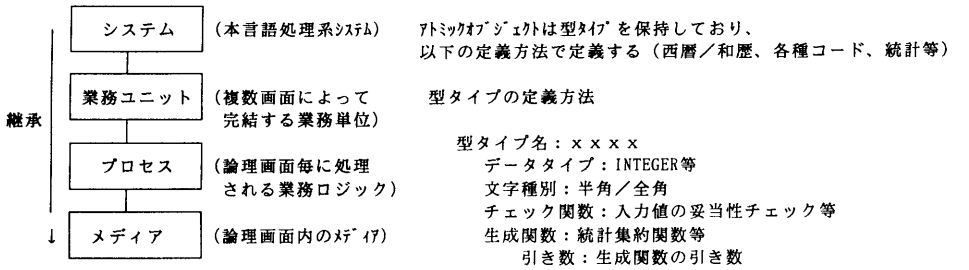


図　４．業務の管理単位とオブジェクトの継承

（４）．ユーザ指向の業務起動インタフェース

アプリケーションロジックは、様々なインタフェースを想定して記述するが、業務システムとしてのイベントを明確にすることによって、「イベント駆動型」で記述することが必要である。ＥＤＰ部門ではない、エンドユーザコンピューティング分野では、幅広いエンドユーザ層を支援する業務起動インタフェースを提供できなくてはならない。また、プロトタイプのためにも「イベント駆動型」が最適である。

パソコン系に完全にオフロードした業務のウィンドウインタフェースは特にこの考え方が重要であり、ウィンドウマネージャ等の物理入出力系とは別に業務仕様に沿ったイベントの規定、インタフェースが必要となる。

例えば、ＰＭ／Ｘ－ＷＩＮＤＯＷ等はスクリーン／マウス／キーボードに関する操作イベントプロトコルを規定している。しかし、ＰＭ／Ｘ－ＷＩＮＤＯＷ等は、（１）．で規定する汎用物理画面インタフェースであり、業務仕様に規定される「論理画面」インタフェースを提供していない。

一般にイベントの発生などの業務仕様に関しては、「論理画面仕様」に基つき、その仕様に沿った、業務プログラムの各ロジックを起動させるインタフェースは、部品（ブラックボックス化、ホワイトボックス化いずれにせよ）化による業務ソフト開発の生産性向上効果が大きい（ｻﾌﾞｳｨﾝﾄﾞｳ等）。

また、本言語仕様で扱うイベントの種類に関しては、以下のシステムイベントとロジックベントの２種類に分類される。

①．システムイベント
　システムが処理を施すイベントであり、システムイベントとして規定するアイコン、プルダウンメニュー、ダイアログボックス等で取り扱われるユーザ動作である。

②．ロジックイベント
　システムがプロセスを起動するイベントであり、本言語仕様で記述した業務内ロジックを選択後実行される引金となるユーザ動作である。

（５）．業務構築に関する一貫性

業務構築に関する一貫性とは、システムに組み込む方法としてのオペレーション仕様を提供し、あるパラダイムを固く守ることである。このパラダイムは、ユーザが考える思考方法をそのものに違和感ないインタフェースとする必要がある。このパラダイムの代表的なものとしてHyperCard等に見られるインタフェースがあるが、ここでは、HyperCard等のエンドユーザコンピューティング（管理業務に適用される場合が多い）の他、現業業務（基幹系業務）への適用も必要条件として、以下の図　５に示すパラダイムを考える。

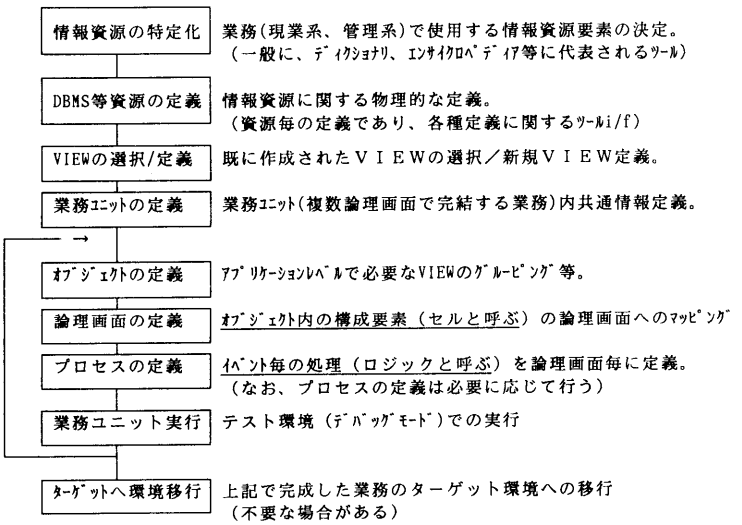


図　５．業務構築のパラダイム

4．言語仕様構成要素（基本概念）とその関連

（1）．基本的な考え方からの言語仕様（基本概念）での対処

3．に於て、基本的な考え方を述べたが、ここで基本的な考え方を基に、言語仕様への反映、及び仕様の範囲について述べる。

①．非手続き的記述仕様と手続き的記述仕様

非手続き的記述仕様と手続き的記述仕様の適用範囲は、業務画面と対象とするＤＢ資源との対応関係を業務仕様と考えると、表２．の様に整理できる。

表　２．非手続き的記述仕様と手続き的記述仕様の範囲　◎：多い，○：少ない，△：殆ど無い

業務画面とアクセス情報資源との対応				本言語での記述仕様	管理業務	現業業務
業務用画面	業務での構造不一致なし			非手続き仕様 論理画面とオブジェクト定義のマッピング	◎	○
	業務での構造不一致あり	順序不一致	SQL等の範囲内で対応 (ex.order-by等)	非手続き仕様 論理画面とオブジェクト定義のマッピング	◎	○
		境界不一致	ｺﾝﾄﾛｰﾙﾌﾞﾚｰｸ処理にて対応	非手続き仕様 論理画面とオブジェクト定義のマッピング	◎	◎
		脈絡不一致	外部ｷｰでのﾄﾗﾊﾞｰｽ (ﾄﾗﾊﾞｰｽ経路固定)	オブジェクト定義 で非手続き仕様 オブジェクト定義時にｱｸｾｽ経路 を固定化	○	◎
			外部ｷｰでのﾄﾗﾊﾞｰｽ (ﾄﾗﾊﾞｰｽ経路浮動)	検索／更新ロジック で手続き仕様 ｵﾌﾞｼﾞｪｸﾄのｱｸｾｽと手続き言語仕様	△	◎

②．情報資源の構造と操作の隠ぺい、情報資源からの業務の独立

情報資源の構造と操作の反映については、表２．に相当するオブジェクト（アトミックオブジェクト：基本となる型タイプも含む），メディアとオブジェクト間の合成、変換等に反映されている。

③．ユーザ指向の業務起動インタフェース

論理画面：付随するプロセス＝１：１に記述できる点で、論理画面をオブジェクトと見ることもできる。これによって画面間でプロセスを共用することが可能となり、各種コード情報の参照等に対しては論理画面を部品として利用可能である。

④．業務構築の一貫性

業務構築に際して、一貫性を保証するためには、概念相互の関連を業務システムのメタ情報として管理しつつ、一貫したオペレーション定義をこのメタ情報を基に簡易に構築できる必要がある。本言語仕様では、特に、業務構築のためのＭＭＩ情報として３層に分類して各種メタ情報を管理し、３．で述べた基本概念に沿って一貫性を保証する。

図　６にこの３層を示す。

ここで業務システムのメタ情報は、図６に示す各種定義の定義体であり、ユーザ指向のウィンドウインタフェースを介して定義される。

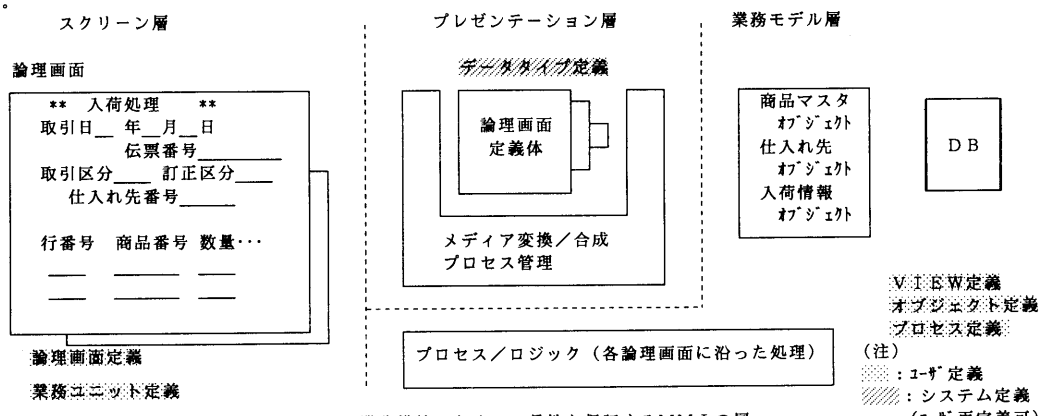
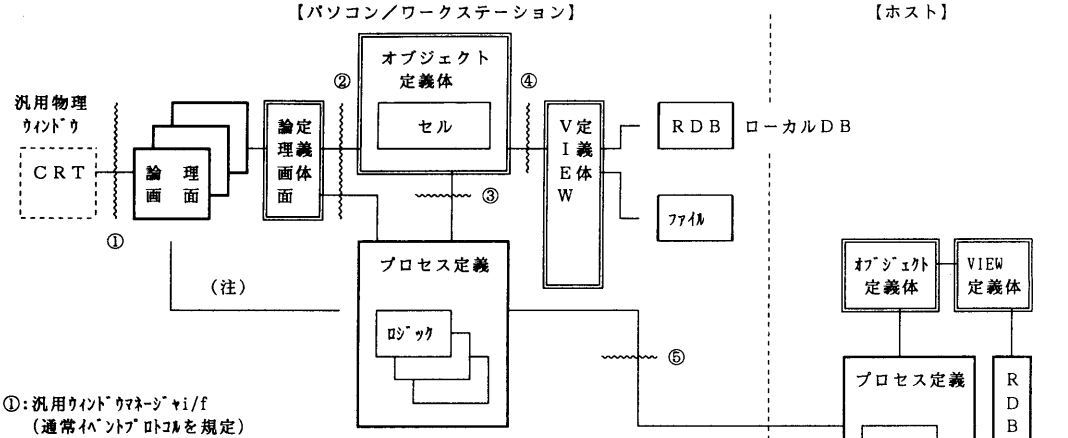


図　６．業務構築のための一貫性を保証するＭＭＩの層

（2）．言語仕様構成要素間の関連　　に示す通りである。

2．で述べた、オブジェクト指向ＤＢ応用簡易言語に対する考え方を基に、言語仕様構成要素（基本概念）は、図 7．



①:汎用ｳｨﾝﾄﾞｳﾏﾈｰｼﾞｬi/f
（通常ｲﾍﾞﾝﾄﾄﾞﾘﾌﾞﾛﾄｺﾙを規定）
②:論理画面に従ったｲﾍﾞﾝﾄﾄﾞﾘﾌﾞﾛﾄｺﾙi/f
③:ｵﾌﾞｼﾞｪｸﾄの参照/更新/追加/削除i/f
④:ｵﾌﾞｼﾞｪｸﾄ実体のｱｸｾｽ経路選択i/f
（現業系では予めｽﾀﾃｨｯｸに決定可能
管理系ではﾀﾞｲﾅﾐｯｸに決定）
⑤:ﾘﾓｰﾄｱｸｾｽ(ﾎｽﾄでの業務処理時)
なお、ｲﾍﾞﾝﾄがﾌﾟﾛﾄｺﾙとなるがｲﾍﾞﾝﾄ=SQLｺﾏﾝﾄﾞ時にはｼｽﾃﾑとしてSQL処理部を起動し、対応する探索結果ﾘｽﾄを返却する。

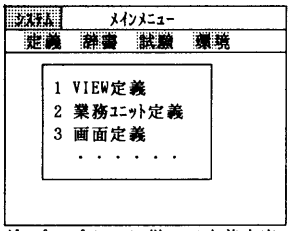
（注）
業務として完結する論理画面とﾌﾟﾛｾｽを業務ﾕﾆｯﾄと定義
（業務ﾕﾆｯﾄは排他/ﾘｶﾊﾞﾘに影響）

図　7．言語仕様構成要素とその関連

5．適用例と適用効果

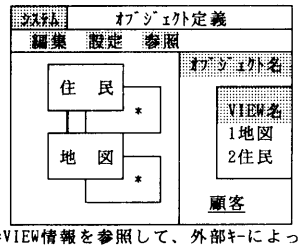
地図を利用した顧客管理システム構築を例にとる。①～⑥に顧客管理の各種画面に沿った定義例を示す。（図５パラダイム
（注）：ｼｽﾃﾑｲﾍﾞﾝﾄ(ｼｽﾃﾑの処理系が持つ論理画面からのｲﾍﾞﾝﾄ)　：ﾛｼﾞｯｸｲﾍﾞﾝﾄ(業務での論理画面からのｲﾍﾞﾝﾄ)

①各種定義ﾒﾆｭ



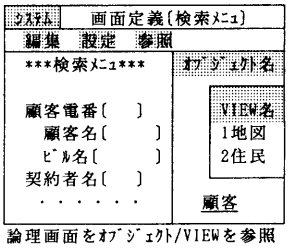
ﾎﾟｯﾌﾟｱｯﾌﾟﾒﾆｭｰに従って定義内容を選択していく

③ｵﾌﾞｼﾞｪｸﾄ定義



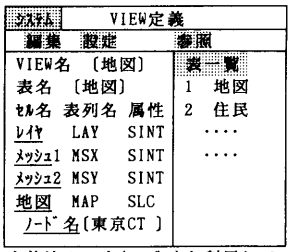
*VIEW情報を参照して、外部ｷｰによって住民と地図を顧客ｵﾌﾞｼﾞｪｸﾄとして定義

⑤論理画面定義



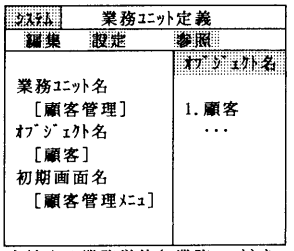
論理画面をｵﾌﾞｼﾞｪｸﾄ/VIEWを参照しながら定義していく

②ＶＩＥＷ定義／選択



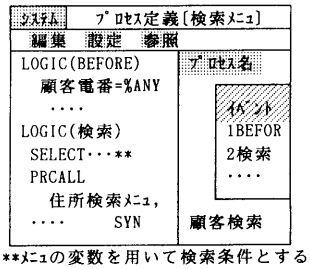
定義済みの表(ﾃｰﾌﾞﾙ)を利用してＶＩＥＷを定義する

④業務ユニット定義



完結する業務単位（業務ﾕﾆｯﾄ）を関連するｵﾌﾞｼﾞｪｸﾄ,初期論理画面で定義

⑥プロセス定義（不要な場合がある）



**ﾒﾆｭの変数を用いて検索条件とするイベントを参照しながら定義

上記例等での定性的な適用効果として、以下がある。

①. 適用業務の認識手段（業務構築のパラダイム）提供、新技術対応の依存部分からの独立（メディア等）によって、ＳＥ要員等の確保が容易になる。

②. 従来のユーザ教育がサブシステム間で閉じていたものが本言語仕様によって技術教育全般が包含されていることによる技術者不足の解消が可能となる。
エンドユーザコンピューティングの促進が可能となる。

③. システム統合化のための分散開発が可能になること。
例えば、４．で述べた業務構築の一貫性に於ける各層は各々分散しつつ分析、設計、開発、保守が可能である。

また、本簡易言語での目的である、管理系業務と現業系業務の扱いに関しては、以下の観点で同等に対応可能である。

①. 手続き言語は現業系業務における性能、構造不一致対処のために従来の構造化言語として残しておくべきである。但し、構造不一致の大半は、管理系業務／現業業務でのオブジェクトの考え方で対処可能である。

②. ユーザ指向の業務起動インタフェースでは、イベント駆動は管理系業務／現業系業務で共通であり、本言語仕様は両者に適用可能である。管理系業務はＰＣ／ＷＳでの分散処理、現業系業務はＰＣ／ＷＳとＨＯＳＴでのイベントを介したＭＭＬ分散処理により、対処可能である。

③. 業務構築に関する一貫性に関する管理系業務と現業系業務の相違に関し、プロセスの記述量に関する相違（これも構造の不一致からくる派生項目）であり、図５に見る様にパラダイムとしては大差が無い。

なお、定量的に検証した適用効果については現在検証中であり別途報告したい。

６．おわりに

本報告では、ＯＡ業務（管理業務、現業業務）の相違を簡易言語による構築の観点から明確にし、システム統合化に向けたオブジェクト指向のＤＢ応用簡易言語について述べた。

御討論頂いた、（株）ＡＴＲ　橋本博士他多くの業務システム開発担当者の方に感謝致します。

（参考文献）

1. 情報処理振興事業協会,"ﾌﾟﾛﾄﾀｲﾋﾟﾝｸﾞﾂｰﾙの有効性と効果的な利用方法の調査報告書", 平成元年２月
2. James Martin,"Fourth-Generation Languages",Prentice-Hall,1985
3. 情報処理振興事業協会,"第４世代言語(4GL)の動向調査及Σｼｽﾃﾑでの利用に関する調査研究"
4. 情報処理学会－関西支部ソフトウェア研究会資料,昭和63年４月
5. Don Leavitt,"More Than End-user Tools,4GLs Add to 的 Productivity",SOFTWARE NEWS,April,1986
6. PAMERA ZAVE,"The Operational Versus the conventional approach to software development" CACM, Feb.1984,Vol.27,No.2,PP.104～118
7. 手島歩三,"システム統合化の手法について",情報処理学会「利用者指向の情報システム」ｼﾝﾎﾟｼﾞｳﾑ, 昭和63年6月
8. John R.CAMERON,"An Overview of JSD",IEEE Trans. SE, Vol.SE-12,No2.,Feb.1986
9. Robert Balzer,Thomas E.Cheatham Jr.,Cordell Green," Software Technology in the 1990's:Using a new Paradigm",IEEE COMPUTER,Nov.1083,pp.39～45
10. 進藤,長岡,天水,川手,黒川,"垂直分散ｼｽﾃﾑ向きDB応用簡易言語の評価",ﾃﾞｰﾀ工学研究会,1989.7

＜参考＞言語仕様

ここで示す言語仕様は、本言語が規定する言語仕様である。

【 画面定義体 】論理画面定義時に展開される言語

```
SCREEN＝論理画面名
OBJECT＝論理画面が対象とするオブジェクト名
DISPLAY＝（STR=論理画面ﾍｯﾀﾞ,表示矩形座標,表示色,表示方法）
MEDIA＝（NAME=ﾒﾃﾞｨｱ名,表示矩形座標,I/O種別,ｵﾌﾞｼﾞｪｸﾄ対応関係,ﾒﾃﾞｨｱのﾃﾞｰﾀﾀｲﾌﾟ,入力操作ｵﾌﾟｼｮﾝ,ｴﾘｱ表現方法）
EVENT＝（NAME=ｲﾍﾞﾝﾄ名, ﾒﾃﾞｨｱ名, ﾄﾘｶﾞ選択方法）
```

(注)(1).ｲﾍﾞﾝﾄ選択方法として以下がある
lineSELECT（１行入力が終ってﾄﾘｶﾞとなる）
cursorSELECT（カーソル位置で入力が終ってﾄﾘｶﾞとなる）

(2).ｴﾘｱ表現方法として、文字列｜数字列、￥／＄数字列、[文字列｜数字列]、(文字列｜数字列)等があるがユーザDEFINE可能である。

(3).ﾒﾃﾞｨｱの種別による定義方法の差

①.表形式メディア
ﾒﾃﾞｨｱ名,対象ｵﾌﾞｼﾞｪｸﾄﾃｰﾌﾞﾙ名,表示矩形座標,
外枠の罫線ﾀｲﾌﾟ,HLINE,TLINE,SCROL_ON,
COLUMN（ｾﾙ名,長さ,ｵﾌﾞｼﾞｪｸﾄとの対応,I/O種別,ｽｸﾛｰﾙ,ﾒﾃﾞｨｱﾃﾞｰﾀﾀｲﾌﾟ,ﾁｪｯｸ要否,編集要否）

②.テキストメディア
ﾒﾃﾞｨｱ名,表示矩形座標,I/O種別,ｵﾌﾞｼﾞｪｸﾄ対応,
ﾒﾃﾞｨｱﾃﾞｰﾀﾀｲﾌﾟ,入力操作ｵﾌﾟｼｮﾝ,ｴﾘｱ表現方法

③.図形メディア
ﾒﾃﾞｨｱ名,対象ｵﾌﾞｼﾞｪｸﾄﾃｰﾌﾞﾙ名,表示矩形座標,
外枠の罫線種別,ﾚｲﾔ指定,ZOOM_ON,BUFF_ON,SCROL_ON

上記でのﾒﾃﾞｨｱﾃﾞｰﾀﾀｲﾌﾟはｱﾄﾐｯｸｵﾌﾞｼﾞｪｸﾄに対するﾃﾞｰﾀﾀｲﾌﾟである。従って、省略した場合、ｼｽﾃﾑ定義のﾃﾞｰﾀﾀｲﾌﾟ利用。

【 プロセス定義】 業務ユニット内でプロセス定義時展開

```
PROCESS  ﾌﾟﾛｾｽ名, 走行環境種別, ｵﾌﾞｼﾞｪｸﾄ名, 走行ｻｲﾄ,
         論理画面名, ENTRY名
   LOGIC：ｲﾍﾞﾝﾄ名                    ┐
          手続き言語ｽﾍﾟｯｸ             ├ ｲﾍﾞﾝﾄで必要な業務処理
   ENDLOGIC                          ┘
ENDPROCESS
```

(注)ｲﾍﾞﾝﾄ名はﾊﾟｿｺﾝ/ﾜｰｸｽﾃｰｼｮﾝ内ﾛｰｶﾙ処理(ﾕｰｻﾞ起動)のｲﾍﾞﾝﾄとﾌﾟﾛﾄｺﾙとﾎｽﾄへのｲﾍﾞﾝﾄ依頼処理に分かれる。手続き言語ｽﾍﾟｯｸは、SQL(ｶｰｿﾙ処理除く), IF,WHILE,CASE等の構造化言語,及びﾕｰｻﾞﾜｰｸｴﾘｱ宣言(DEF)等から成る

【 ＶＩＥＷ定義】　VIEW定義時に展開

```
VIEW  VIEW名, ＤＢ種別, 対応ﾃｰﾌﾞﾙ名、
          存在場所=（SITE, ﾃﾞｰﾀﾍﾞｰｽ名）
          主キー=ﾕﾆｰｸｷｰとなるｶﾗﾑ
  EQUIVALENCE  ｾﾙ名=（対応ｶﾗﾑ名ﾘｽﾄ）
  CELL  NAME=ｾﾙ名, COLUMN=対応ｶﾗﾑ名／*ｾﾙ名は日本語化*／
          ・・・・・・・・・・・・・
  CELL  NAME=ｾﾙ名／*EQUIVALENCEで指定したもの*／
ENDVIEW
```

【 オブジェクト定義】ｵﾌﾞｼﾞｪｸﾄ定義時に展開

```
OBJECT  NAME=ｵﾌﾞｼﾞｪｸﾄ名
     CELL  NAME＝RDBﾃｰﾌﾞﾙ名 〔SIZE＝返却IOｻｲｽﾞ〕
        VIEW     対応するVIEW1名
     CELL  NAME＝RDBﾃｰﾌﾞﾙ名 〔SIZE＝返却IOｻｲｽﾞ〕
        VIEW     対応するVIEW2名
           WHERE  VIEW名2.XXX＝VIEW名1.XXX  AND
                  VIEW名2.YYY＝VIEW名1.YYY  AND
                  VIEW名2.ZZZ＝VIEW名1.ZZZ
                 /*VIEW1とVIEW2の対応付け */
ENDOBJECT
```