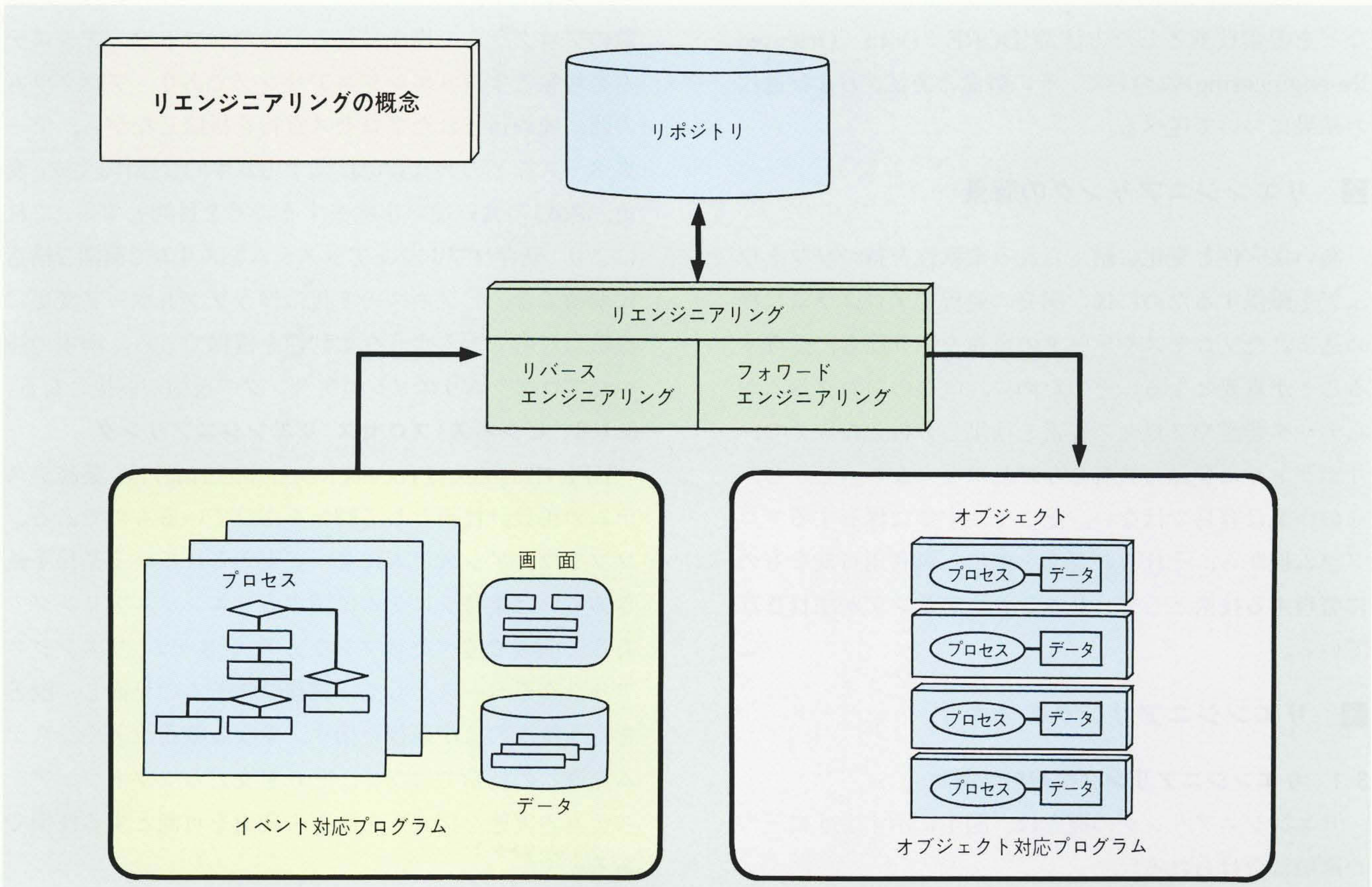
データ中心アプローチによるリエンジニアリング支援技術の適用実験

Experimental Use of Data Oriented Approach in Re-engineering Methodology

山川敦夫* *Atsuo Yamakawa*
上林高治* *Takaharu Kanbayashi*
石本真希* *Maki Ishimoto*
宮本信夫* *Nobuo Miyamoto*
飯田啓三* *Keizô Iida*
堀内　一** *Hajime Horiuchi*
秋庭真一** *Shin'ichi Akiba*
山村信幸*** *Nobuyuki Yamamura*



データ中心アプローチによるリエンジニアリング　データ中心アプローチによるリエンジニアリングは，既存のプログラムの解析を通じて，再利用可能なオブジェクトを抽出するリバース過程と，そのオブジェクトによって新しいソフトウェアを作るフォワード過程で構成する。

高い保守性と柔軟性を持つソフトウェアを確保するためには，膨大な既存のプログラムからデータ要素やプロセス要素を抽出し，分析したうえでリポジトリに登録し共有しなければならない。そのため，すでに保有するプログラム群から，これらの要素を抽出し，再利用可能なものに整理する技術としてリエンジニアリングが注目され始めている。

リエンジニアリングは，新しい情報システムパラダイムに対応するためのシステム再構築方法論として，今後その重要性が増すものと思われる。

* 株式会社オージス総研　** 日立製作所 ビジネスシステム開発センタ　*** 日立西部ソフト株式会社

1 はじめに

リエンジニアリングとは，システムの保守あるいは再構築のための技術の総称である。深刻なソフトウェア保守問題の解決あるいはダウンサイジングなど，新しい情報システムパラダイムへ対応するためのシステム再構築方法論として，その重要性が指摘されているものである。ここでは，リエンジニアリングのためにデータ中心アプローチ(DOA：Data Oriented Approach)，リポジトリなどを要素技術とした方法論(DORE：Data Oriented Re-engineering)について，その概念と方法，および適用の結果について述べる。

2 リエンジニアリングの背景

高い保守性と変化に耐えられる柔軟性を持つソフトウェアを確保するためには，開発の過程でプログラムに埋め込まれたプロセスやデータの重複を取り除き，整理することが重要となる。そのために，既存のプログラムからデータ要素やプロセス要素を抽出し分析したうえで，リポジトリに登録し共有しなければならない。しかし，その作業は容易ではない。そこで，すでに保有するプログラム群から，それらの要素を抽出し再利用可能なものに整理する技術として，リエンジニアリングが注目されている。

3 リエンジニアリングの体系

3.1 リエンジニアリング技術の階層

リエンジニアリングの概念は，**図1**に示すように三つの階層に分けられる。

3.1.1 プログラムリエンジニアリング

プログラムリエンジニアリングとは，スパゲッティ状態になった個々のプログラムを解析し，その中に存在するデータ要素やプロセス要素を整理するとともに，簡潔で重複コードを持たないプログラムに作り替える過程および技術を指す。個々のプログラムの変更や保守作業の効率を大幅に改善するものである。

3.1.2 ソフトウェア システム リエンジニアリング

ソフトウェア システム リエンジニアリングとは，複数のプログラムで構成される一つのソフトウェアシステムを対象とするリエンジニアリングであり，プログラムの間に埋め込まれたプロセス重複を排除しながら，データベースなどの共有資源に関する基本的な操作(生成，変更，消滅)の食い違いを除去することを目的とする。これにより，既存ソフトウェアシステムをスリムで簡潔な構造に変換でき，ビジネスの変化に伴うソフトウェア変更に機敏に対応できるような柔軟性を確保できる。前項で述べたプログラムリエンジニアリングの技術が前提となる。

3.1.3 ビジネス プロセス リエンジニアリング

BPR(Business Process Re-engineering)は，業務システムの再設計技術として脚光を浴びているものである。ソフトウェアシステムによって支えられている業務手続きあるいは業務システムに関するリエンジニアリングである。前項で述べたソフトウェア システム リエンジニアリングをベースとして，業務システムの見直し，改善を行う技術および過程を指す。単なる業務改善やシステム見直しの技術ではなく，それを支えるソフトウェアシステムの実態を踏まえた改善や変更を可能とする技術である。

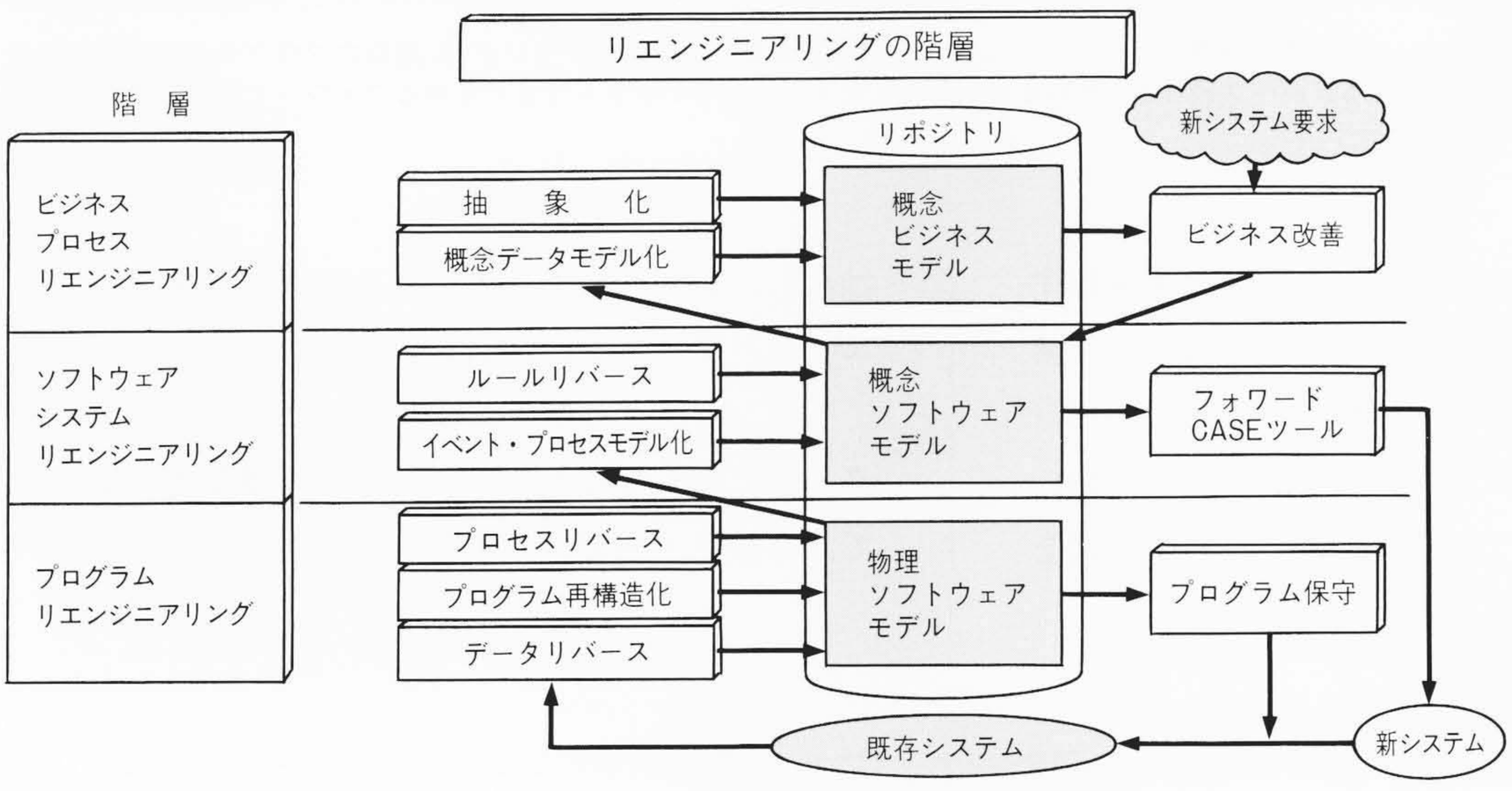


図1 リエンジニアリングの階層 リエンジニアリング過程は，それぞれ独自の目標を持つ三つの階層に大きく分けられる。

4　DOREの概要

リバースエンジニアリングは，既存のプログラムから論理を抽出し，プログラムの仕様書へ戻すための技術と考えられてきた。しかし，DORE(DOAによるリエンジニアリング)では，リバースエンジニアリングを単にドキュメントなどソフトウェアのビュー情報を作り出すだけでなく，再利用可能なソフトウェア要素を抽出する技術とみなしている。そのため，プログラムを構成する論理に目を向ける前に，そのプログラムが処理の対象とするデータ要素に目を向けて，その構造と特性を分析することにより，論理の意味的追跡の負担を大幅に軽減させることを目標としている。

4.1　DOAによるモデル化

DOAとは，ソフトウェアの設計および開発にあたってデータを共有資源とみなし，先にそれらの分析を通じて重複を排除し，それぞれのデータに固有のプロセスを洗い出し，それぞれのデータにくくり付ける方法を指す。

一般に，データに固有なプロセスを当該データにくくり付けてデータとともに扱う概念を抽象データ型と呼ぶ。またデータ固有のプロセスをくくり付けることを，外部に対してその内部の詳細を隠ぺいすることからカプセル化とも呼ぶ。抽象データ型もカプセル化も，オブジェクト指向で用いられている概念である。

DOAもオブジェクト指向の発想を持つ。ただし，現行システムで保有するデータをオブジェクトとして認識するボトムアップの概念を持つことを特長とする。

DOAでは，データにはそれぞれ固有の処理があって，その処理はデータ対応に検討，作成すべきであると考える。したがって，データとはレコード型やデータ型で表現されるものだけでなく，そのデータに固有の操作(生成，変更，消滅)とそのデータの形式や存在の条件を示す固有の制約(Constraint)も含むものと考える。例えば，「和暦日付」という基本的なデータについてみれば，その日付の形式や値を規定する制約(検証条件)は，「和暦日付」データに固有のものを容易に定義することができる。

一方，ユーザーが必要とするデータ，例えば「受注画面」は，「和暦日付」や「顧客番号」など基本データ(基本オブジェクト)をいくつか組み合わせた複合物とみなすことができる。そのような複合物を集約オブジェクト(Aggregate Object)と呼ぶ。ここでオブジェクトと呼ぶのは，いずれもそれぞれ固有の操作，制約を持つものである。

4.2　DOREによるリバース過程

DOREによるリバース過程は，大きく三つの過程から構成される。それぞれの作業内容は以下のとおりである。

(1)　データリバース

オブジェクトの核になるデータを抽出する過程である。大きくデータモデリング過程とプログラム内データ分析過程との二つに分けられる。

データモデリング過程は，リバース対象のソフトウェアシステムがかかわりを持つ実世界の領域を明らかにし，そこでのビジネス実体(顧客や注文など)を抽出するとともに，その実体に対応するプログラム内データ要素を明らかにする過程である。

プログラム内データ分析過程は，プログラム内に存在するデータをオブジェクトとして把握するもので，プログラマによって任意に設定されて，任意の名称を与えられているデータを洗い出し，それらの間の等値関係を明らかにしながら，その重複状況や依存関係を明らかにする過程である。プログラムを構成する命令を吟味し，例えば，COBOLのMOVE命令で，“MOVE A TO B”とあれば，データAとデータBは同じ値(等値)を持つ関係にある。膨大なソースコードを，そのような観点から解析するツールが必要となる。

(2)　プロセスリバース

プログラムを解析して，それぞれのデータの生成，変更，消滅に関する処理を抽出する。それらの処理をデータのLCP(ライフサイクル処理)と呼ぶ。DOAでは，LCPをデータに固有のプロセスとしてカプセル化することをその特長とする。

(3)　ルールリバース

データに固有の制御を抽出する過程である。プログラムにコードとして埋め込まれている制御処理を分解し，データごとに固有な制約としてカプセル化する。

プログラムを構成するコードの多くはIF文による制御処理に費やされており，その比率は70%以上にも及ぶ。また，プログラムに存在する多くの制御処理は，トランザクションを処理するための構造を持つため，必要とする処理を必要とする時点で組み込む。そのため，同一データに関する同一のチェック処理などが，多くのプログラムの間に重複する結果となる。プログラムの重複をもたらす最大の原因は，そのようなトランザクション対応の制御処理の実現方法にあり，それが今日のビジネス系のシステムのプログラムステップ数を，数倍大きなものとしている。もしそれらの制御処理を，データ対応の固

有処理としてカプセル化できれば，プログラムサイズは数分の一に圧縮できる。

しかし，現実のプログラムでは単一のデータに固有な処理だけでなく，複数のデータの組(関連)に固有な処理が多い。例えば，「顧客番号」と「契約種別」の関連チェック，あるいは「社員番号」と「プロジェクト番号」の関連チェックなどである。その比率は，プログラムで実行されているチェック処理の90%以上となる。従来，そのようなチェック処理はアプリケーションごとに異なる「アプリケーション論理」とみなされてきた。DOREではそのようなチェック処理も，「顧客番号」と「契約種別」の二つの基本オブジェクトから構成される集約オブジェクトのものとみなし，カプセル化する。プログラムコードの多くは，そのような集約オブジェクトの制約として置き替えられる。

5 DORE適用実験の概要

株式会社オージス総研と日立製作所は，平成4年4月から研究プロジェクトを発足させ，DOREに関する適用実験を行っている。すでに，リバース作業の実験を完了し，いくつかの課題と見通しを得ている。以下にその結果について述べる。今回の適用実験でのリバース手順は**図2**に示すようなものである。

5.1 データリバースとその結果

データリバースは，既存ソフトウェアが処理の対象としているデータを明らかにする過程である。ここでは，従来もデータベース設計で用いられているデータ正規化を中心とした分析技法が有効であった。

プログラム内データの抽出と等値関係の分析では，プログラムの中で一つのデータと等値関係にあるデータを特定するのに，データの移送関係，データ形式上の相対位置関係などを主体にツールを作成して分析した。しかし，データの等値関係はプログラム内に限定されず，複数のプログラムにまたがっても存在する。現実のプログラムでは，それぞれのプログラマーによってプロセスが任意に作り込まれていることから，ツールによって容易に検出できない等値関係も多い。特に，複数の処理で共通領域を用いてデータ移送を行っている場合は，ソースコード レベルでの静的解析では限界があり，等値関係にないものまでが抽出されることもある。分析ツールの精度向上が今後の課題である。

5.2 プロセスの解析と2項オブジェクト抽出

既存プログラムのプロセスとルールを解析するために，いったんソースコードを分解し2項オブジェクトとしてリポジトリに格納した。2項オブジェクトとは，プログラムを構成する命令群を，操作と操作されるデータ項目とから構成されるオブジェクトに変換したものである。例えば，

MOVE A TO B

というCOBOLのデータ移送命令は，

[(A, B)MOVE]

と表現されるオブジェクトとする。[ ]がオブジェクト

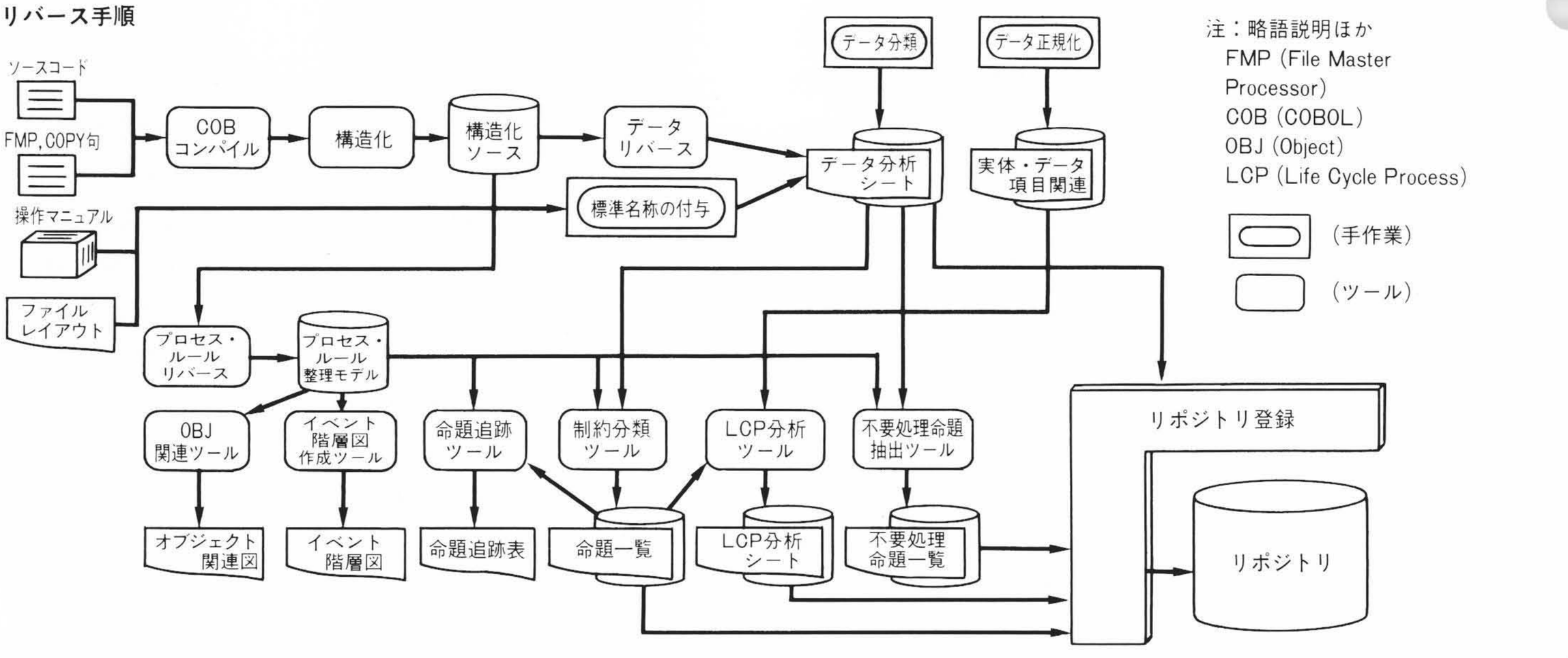


図2 DOREリバースエンジニアリング手順　既存ソースプログラムの解析の過程と，開発したツールの相互関連を示す。

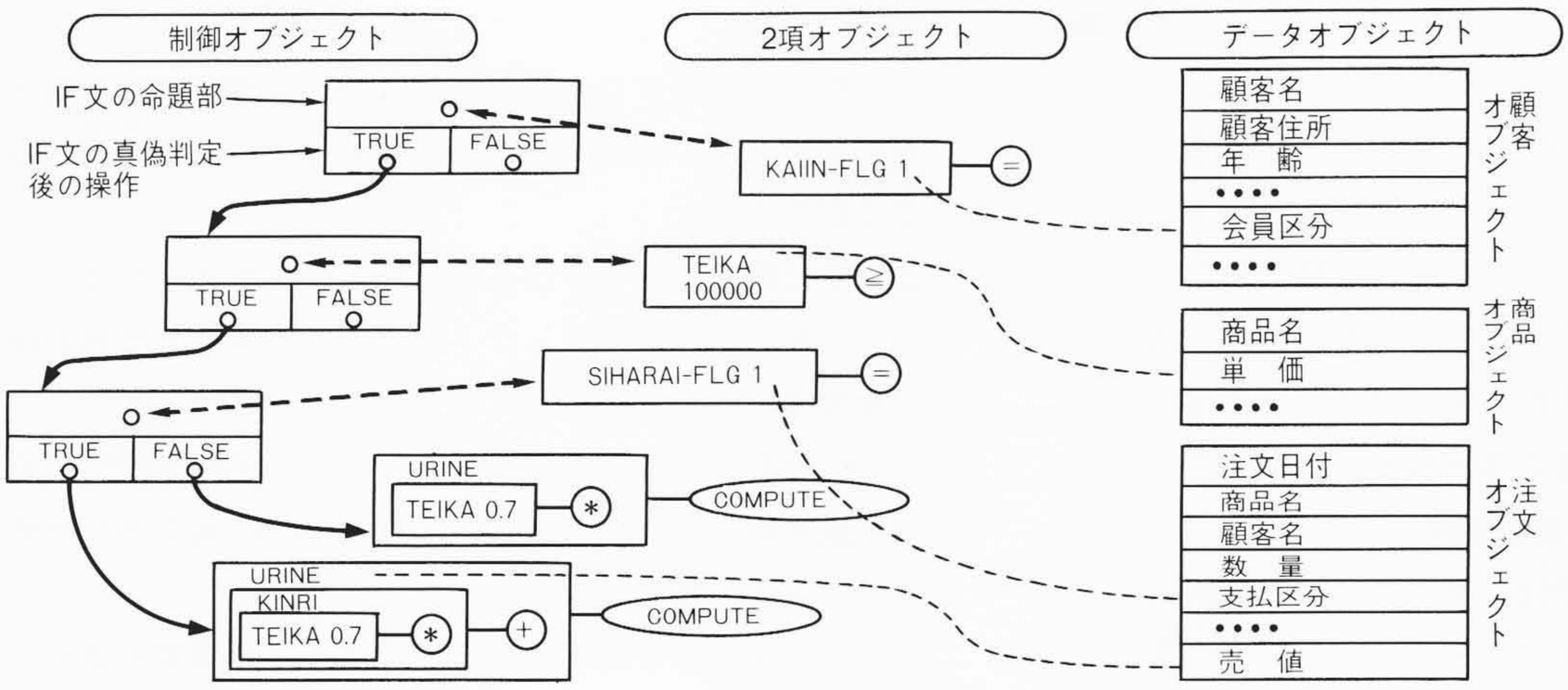
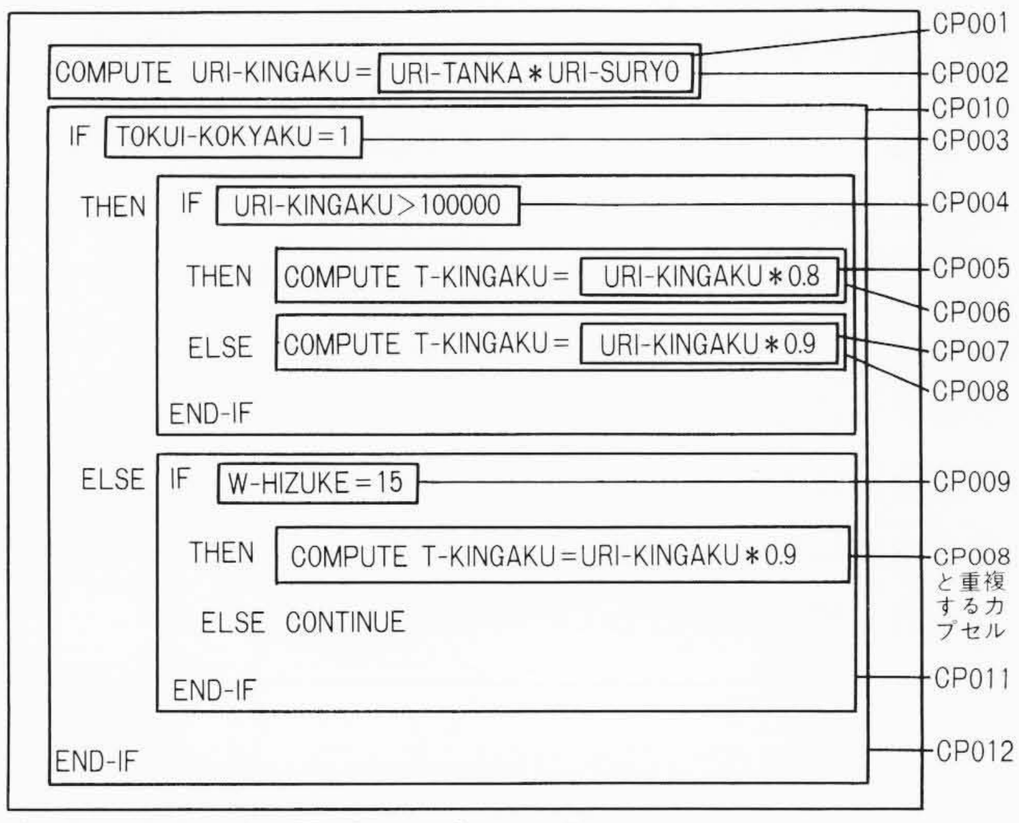


図3　2項オブジェクトによるプログラム論理の置き換え　ソースプログラムの命令は，2項オブジェクトとしていったん抽出され，それらを用いてIF文などの制御をオブジェクトとして抽出する。

を示す。(A, B)がデータA, Bから構成される集約データを示す。MOVEはその(A, B)に固有のプロセスとなる。2項オブジェクトを構成するデータは，あらかじめ求められている標準データおよびそれと等値関係にあるデータと関連づけ，重複関係を把握した。2項オブジェクトによるプログラム論理の置き換えの様子を**図3**に示す。COBOLソースコードの一部を**図4**に，そのソースコードから得られた2項オブジェクトを**図5**にそれぞれ示す。**図5**で同一カプセル番号を持つものが2項オブジェクトである。

5.3　プロセスの解析とルールの抽出

IF文による各種の制御命令は，これら2項オブジェクトを基に制御を表現する集約オブジェクト(制御オブジェクトと呼ぶ。)として抽出できた。これにより，これまで人間の追跡作業に頼っていた制御ロジックの取り出しを，機械的にオブジェクトの組み合わせとして取り出すことを可能とし，ツールによる自動化の可能性を高くした。今回の実験でもツールにより，一つの制御ロジックに関連するデータやプロセスを表現する命令を集約オブジェクトとして自動的に抽出できた。



注：CPXXXは，次の図5中のカプセル番号を示す。

図4　COBOLソースコード　COBOLソースコードが，どのように分解されカプセルとして抽出されるかを示す。

6　DOREの適用実験と問題点

DORE適用実験の結果，いくつかの問題点が抽出されている。

6.1　再利用不要オブジェクトの消去方法

プログラムから抽出される制御オブジェクトは膨大な数に及ぶ。しかし，その多くはDBMS(Database Management System)のリターンコードチェックなど処理

カプセル番号	カプセル構成オブジェクト	固有プロセス
CP001	URI-TANKA	*
CP001	URI-SURYO	*
CP002	URI-KINGAKU	COMPUTE
CP002	CP001	COMPUTE
CP003	TOKUI-KOKYAKU	=
CP003	1	=
CP004	URI-KINGAKU	>
CP004	100000	>
CP005	URI-KINGAKU	*
CP005	0.8	*
CP006	T-KINGAKU	COMPUTE
CP006	CP005	COMPUTE
CP007	URI-KINGAKU	*
CP007	0.9	*
CP008	T-KINGAKU	COMPUTE
CP008	CP007	COMPUTE
CP009	W-HIZUKE	=
CP009	15	=
CP010	CP004	
CP010	CP006	
CP010	CP008	
CP011	CP009	
CP011	CP008	
CP012	CP003	
CP012	CP010	
CP012	CP011	

図5　2項オブジェクトとその上に表示された集約関係
同一のカプセル番号を持つものが，2項オブジェクトであることを示す。

環境に依存する制御や，フラグなどプロセスデータの制御に関するものである。実際のプログラムでは，そのような制御オブジェクトが80%以上を占める。本来，それらは再利用する必要のないものである。もし，事前にそのような環境依存性，実現方式依存性を持つオブジェクトで再利用の必要のないものを，何らかの方法で消去できれば，リバース作業は大幅にそのコストと作業時間を削減できる。不要制御論理の事前検出が課題となる。

6.2 制御オブジェクトの意味理解

DOREによるリバースでは，ソースコードに対応する2項オブジェクトを最小オブジェクトとし，その集約関係として制御を表現した。それにより，ツールによる機械処理は可能となった。しかし，その集約オブジェクトを構成する2項オブジェクトが持つ意味を人間が理解できるとは限らない。例えば“IF KAIIN-FLG=1 THEN～”という制御文を「顧客が正会員ならば～」という意味に抽象化することは容易なことではない。しかし，プロセスリバースによる分析の精度を向上できれば，“KAIIN-FLG”と「顧客番号」の間の集約関係も把握できる。また“KAIIN-FLG”オブジェクトが取り得る値を列挙することにより，「正会員」，「準会員」，「非会員」などのサブタイプが存在することも把握可能となる。標準用語辞書を用意し，オブジェクト名称の自動置き換えで人間が理解できる表現も可能となる。

7 DORE実験の意義と今後の課題

DOREは，リエンジニアリング全体をカバーする方法論である。現在，そのうちリバース過程だけをブレークスルーした状況である。しかし，今回の共同研究によって，次のような事項が確認できた。

(1) 既存プログラムを構成するソースコードを2項オブジェクトとして抽出，整理できること。

(2) 複雑なプログラム内制御を，2項オブジェクトを基底とする集約オブジェクトに置き換えられること。

(3) 集約オブジェクトを構成する各データ要素を，等値関係によってプログラム外部の標準データに結び付け，標準データを基に重複を把握できること。

以上の結果，次のような事項の可能性が期待できるようになった。

(1) 複雑な制御フローを追跡しながら論理の抽出を行う方法に比べて，機械処理による自動化の可能性を高めることができる。

(2) 単に既存のプログラムを仕様書に戻すことだけのリバース概念を，オブジェクトによる整理を通じて再利用可能要素を抽出する概念に代えることができる。

(3) CASEツールによるフォワード過程(新プログラムの生成過程)で必要となるオブジェクトを，あらかじめ準備する方法が確保できる。

しかし次のような事項については，より研究を進めていかなければならない。

(1) 今回の研究，実験では特定な処理環境を前提にして進めた。対象とするプログラム言語もCOBOLだけであった。また実験の対象とプログラム本数も10本に満たない小規模のものであった。この方法論をどのような環境や規模にも適用させるためには，まったく別な配慮が求められる。例えば，ユーザー環境ごとに組み込まれた独自のツールや言語，プラットフォームソフトウェアのバージョンの相違などへのきめ細かな対応が求められる。

今後，リエンジニアリングサービスをビジネスとするならば，そのような個別の差異とそれへの対処方法をノウハウとして蓄積することが必須(す)となるであろう。

(2) 先の**図1**に示すように，リエンジニアリングの範囲は広い。今日求められるものは，ビジネス戦略を達成するためのビジネス方式，組織構成，情報システムのすべてを見直し改善するBPR方法論である。今後，DOREをベースにそのような上流技術との統合を考えたい。

8 おわりに

ここでは，リエンジニアリング全体をカバーするDOREの概要，適用実験を中心に述べた。

株式会社オージス総研と日立製作所のDOREに関する共同研究の結果，機械処理による自動化の可能性，オブジェクトによる整理を通じてリバース概念を再利用可能要素を抽出する概念に代えること，などの実現が期待できるようになった。今後は，DOREをベースに新しい技術の統合，開発に努力していく考えである。

参考文献

1) 堀内：データ中心システム設計，オーム社(1988)
2) 酒井，外：オブジェクト指向入門(1990)
3) 堀内，外：データ中心アプローチによるリエンジニアリングの方法，情報処理学会，研究報告，93-IS-42(1993)
4) 山川，外：データ中心アプローチによるプログラム論理の抽出，情報処理学会，研究報告，93-IS-42(1993)