企業情報システムにおけるデータ中心手法導入の要件†

堀内 一††

データ中心アプローチ (DOA : Data Oriented Approach) とは，データを共有資源とみなし，先にその標準化とモデル化を徹底した後に，システムやソフトウェアの構造を決定する方法の総称である．ソフトウェアやシステムの開発形態とその進め方に大きな変革を求めるものである．その背景には，企業における十数年に及ぶ個別システム開発の繰り返しが，膨大なソフトウェアの累積とその構造的劣化をもたらし，それらの保守を困難なものとし，ビジネス戦略の変更を追随できない硬直性を招来している事実がある．本論では，これまでのコンピュータ化領域の拡大期に用いられてきたシステム開発方法論の問題点を指摘しながら，今後，求められるパラダイム変革の一つとして，要求や機能指向から共有資源であるデータ指向にソフトウェアを構築する意義とその手法の要件を整理する．その要件として，データに対する認識を，単にその型だけに向けるのでなく，データに固有の制約やプロセスをデータにカプセル化する概念を実現する手法のあり方，さらにコードとしてのプログラムを代替するメタ情報の設計および蓄積手法のあり方を述べる．

1. はじめに

今日，企業における情報システムは，ハードウェア技術の革新による変化だけでなく，情報システムに対する認識変化や適用業務の質的変化，あるいは保有するソフトウェア累積量の変化などにより，これまでにない大きな変革を求められている．システム開発方法論についても，それらの変化に対応するため，これまで有効であった方法論をあらためて見直す段階にきている．

データ中心アプローチ (Data Oriented Approach : 以降 DOA と呼ぶ) は，データを共有資源とみなし，先にそのモデリングを徹底し標準データを設定した後，システムやソフトウェアの構造を決定する方法の総称である．システム設計やソフトウェア設計の進め方に，大きな変革を求めるものである．

昨今，システム開発におけるデータ分析手法やデータディクショナリの導入は増えており，データ中心の概念を掲げてシステムの再構築に取り組んでいる企業も多い．しかしながら，一方で，データ分析やデータモデル化の目的や意義について十分な合意が得られず，挫折するものも少なくない．

本稿では，あらためて企業情報システムの今日の状況から DOA の必然性と意義を考察し，DOA の手法が目的とすることを整理しながら，企業にその方法論を導入する上での要件を明らかにしてみたい．

2. データ中心システム開発の目標理念

今日，多くの企業や組織は，既に膨大な量のソフトウェアを保有している．しかし，一方で，それら大量のソフトウェアによる無統制な資源共有はシステム全体の重複や構造劣化をもたらし，その維持に多大な労力支出を強いていることも事実である．**図1**は，大手企業数社における，ここ十数年の保有ソフトウェア量の伸びを示すものである．どの企業も年率十数パーセント以上の成長率でソフトウェアを増やしてきていることがわかる．

これらのソフトウェア成長は，「ソフトウェア生産性の向上」を目的として，プロジェクト単位の開発効率を高めた施策の結果ともいえる．これら企業の中には，ソフトウェア量が既にその管理限界を超え，業務変更によるシステム変更に対応できない状況にあるものも少なくない．DOA は，既に基幹業務のコンピュータ化を一巡した情報システムが直面する課題を取り上げ，その課題解決を目標理念とする方法論である．

DOA では，問題の根元が「ソフトウェアの製造技術」にあるのでなく，「組織が保有するソフトウェア全体の構造」にあるという前提にたつ．つまり，都市としての問題は，住宅やビルの建築工法に起因するものでなく，都市全体の構造とそこにおける統制の課題と考える．ちなみに，情報システムの開発・運用において，DOA が問題とする典型的現象の具体例は次のようなものである．

(1) 基幹業務を支えるデータベースの回りに，数百本のプログラムが作成され，それらのうち，数十本の

† Requirements for Applying the Data-Oriented Method on an Enterprise Information System by HAJIME HORIUCHI (Institute of Advanced Business Systems, Hitachi Ltd.).

†† (株)日立製作所ビジネスシステム開発センタ

プログラムがそのデータベースを更新する.

(2) 「製品コード」のような共通データを一つ変更するために,現有のプログラムの変更箇所を洗い出し特定するだけの作業に数千万円を費やす.

(3) 1本のプログラムを変更したらその影響が数百本のプログラムに波及した.

(4) 一つのプログラムで使用するデータベースを特定することは容易であっても,一つのデータベースを使用するプログラムすべて特定するのに,1週間以上の期間を要する.

(5) プログラムの本数は,日ごとに増加し,やがては数万本のプログラムを持つことが予想される.

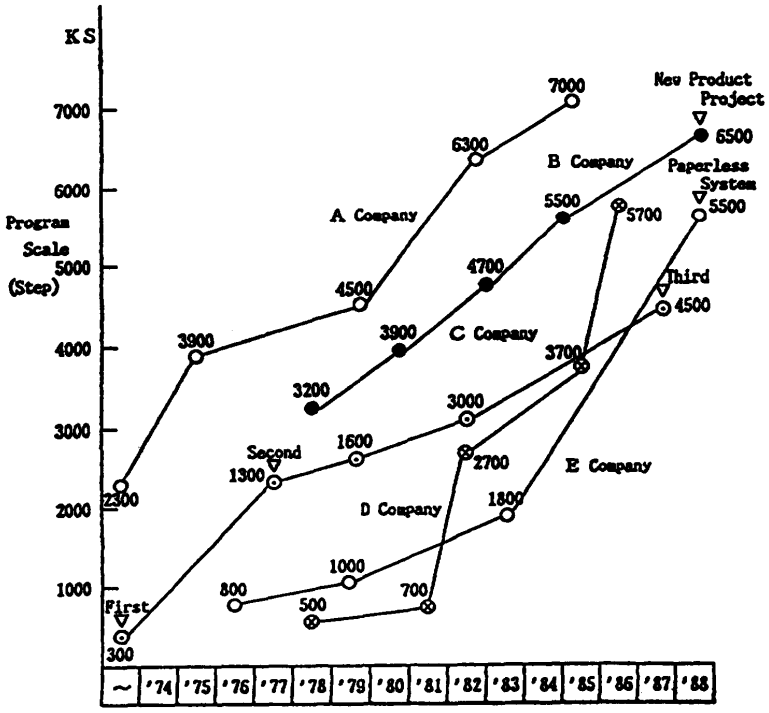

図1 ソフトウェア成長の状況
Fig. 1 Growth in software inventory.

これらの現象は,多かれ少なかれ,今日の企業すべてに共通するものといえる.DOAでは,これらの現象が,本来,共有資源とみなすべきものが適切に管理されず,資源の自給自足形態を許し,その結果に生じる重複と不整合によるものと考える.

つまり,重複と構造劣化の原因の一つは,要求に対する機能設定を優先し,その機能分解によりソフトウェアモジュールを導く構造化概念にあるといってもよい.共有資源に対する配慮が二義的となり,資源固有の処理や制約を,プログラムごとに重複させる傾向を持つからである.「ウォータフォール有害論」として指摘されたことも,このようなリスクに対する警鐘と受けとめることもできる.

3. データ中心型方法論の要件

3.1 これまでのデータ中心型手法の問題点

データ中心またはデータ主導の概念を持つソフトウェア設計技法は,70年代中頃から,ジャクソン法(JSP)[1],ワーニエ法(LCP)[2]として提案されている.プログラムの内部構造を,そのプログラムで取り扱うデータの構造から導く手法を示したものである.G.J.メイヤー[3]らが示した機能分解によるプログラム構造の導出に比して,その基準は客観的で具体的であった.しかし,これらの手法に共通することは,あくまでも単一のシステムやプログラムを開発するための手法であり,事前にデータを要求と独立させて分析する概念は持たなかった.後に,M.A.ジャクソンはJSD (Jackson's System Development)[4]により,データの事前分析を取り入れ,ELH (Entity Life History)分析手法などを示しているが,データ(実体)固有のライフサイクル処理や制約をそのデータ(実体)にカプセル化させる概念は持たなかった.

システムの要求分析やその定義手法の分野では,データフローダイアグラム(DFD)を用いた,E.ヨードンやT.デマルコらの構造化システム分析(SSA)手法[5]がデータ中心の概念を持つものであった.データフローダイアグラムを用いた構造化システム分析手法には,P.T.ワード[6]やE.ヨードン,あるいはD.J.ハートレイら[7]によるリアルタイムSAもある.しかしながら,これらの手法は,データ分析手法を示し,データの流れに着目したシステム仕様記述など,プロセスに着目することの弊害を捉えているものの,あくまでも要求された単一システムの領域に留まっている.つまり,データを共有資源とみなし,個別システムとは独立させて,そのモデル化を図りながらデータ固有のプロセスや制約を事前定義する概念はない.さ

らに，構造化分析手法では，プログラムは DFD（データフローダイアグラム）の上で機能分解基準を用いるために，プロセス重複を回避する上で問題が多い．

データベース設計手法の分野では，データモデル化手法が，数多く提案され，データベースの整合性と一貫性を重視する立場から，要求対応にプログラムとデータを設計する個別手配形態を極度に警戒するようになり，共有資源を設計制約として強く意識する方法論を主張するようになった．データを共有資源とみなして，実世界を反映した概念データモデル化を取り込んだシステム設計手法としては，Nijssen の NIAM[8]，英国政府標準とされている SSADM[9]，あるいは M. A. ジャクソンの JSD などがある．しかし，いずれも分析したデータモデルをデータ辞書に登録し，ソフトウェア開発の基盤として提供するにとどまり，積極的にデータ固有の処理や制約を抽出しカプセル化するに至っていない．

さらに，CASE ツールの登場は，データ資源の管理サイクルとシステムライフサイクルの有機的連動を可能とした．そのような環境を前提とした方法論も提案されるようになった．例えば，J. マーチンの IE（インフォメーションエンジニアリング）[10] などである．しかし，データモデル化を全面に押し出して，そのライフサイクル分析や制約分析の方法を示しながら，多くの方法論は，プログラム生成をその目的に置いており，「ソフトウェア開発生産性」を強調している．抽象データ型やカプセル化の発想を持つものは少ない．

3.2 データ中心開発方法論の要件

システム開発方法論とは，単に，求められるシステムやプログラムを構築する作業手順を示したものではない．企業や組織におけるシステム開発やその運用における問題を解決しながら，繰り返されるシステム開発を，一定の目標概念へ導くものでなければならない．したがって，方法論そのものが，その企業や組織の状況や時代の趨勢と共に変化するものといえる．

これまでの方法論を，R. L. ノーランのステージ論[12]で示された第2段階（基幹業務のコンピュータ化段階）のものとすれば，新たに求められる方法論は，その第3段階（既存システムの整理統合）のものと考えられる．つまり，ノーランの第2段階では，コンピュータ化領域の拡大に重点が置かれ，個別のプロジェクトをもって，要求されるシステムが順次，開発される．望まれるのは，ユーザの要求への準拠性であり，短期間のシステム領域拡大であった．したがって，ソフトウェア開発の生産性向上に過度の重点が置かれたと言ってよい．既に述べたシステムの構造劣化や重複発生はそのような目標概念を持つ方法論の結果といえる．

第3段階の方法論は，コンピュータ化領域の拡大が止まり，それまでに開発されたシステム群の上に，整合性と一貫性を確保するためのものである．個別開発の繰り返しがもたらした不整合や重複を除去しない限り，システムそのものが存続できなくなるからである．そのために，領域拡大が緩やかになったシステム全体を見渡し，共有物と共通項を発見しながら全体のフレームワークを定義する．そして，フレームワークに従ってシステム全体の再構築を考える．共有資源の管理を先に組織化し，その資源統制を基に個別のシステムを構築・保守するアプローチへのシフトが求められる．

データや情報を資源とみなす概念[13),14)]は，単に，資源の有効活用だけを目的とするのでなく，資源に基づくシステム全体の統制確立という命題を果たすものとなる．したがって，データ資源管理は，データを，システム全体の見地からの標準化し，重複を排除しながら，システム全体の一貫性と整合性を維持するためのフレームワークとできることを意味する．

DOA は，そのような段階における方法論として，データ資源管理を基に，資源の制約を遵守しながら，個別システム開発を行わせるものである．データ中心，データ主導なる概念は，データ資源の制約を重視する資源指向概念にほかならない．したがって，データを資源として管理する仕組み・体制を前提とした方法論となる．システム開発や個別要求の解決は，そのような資源管理を基盤とした束縛の範囲でなされるべきである．**図2**は，そのような概念を持つ DOA 手順を示すものである．資源管理タスクとプロジェクトタスクは分離されている．

4. DOA 手法の目的と課題

データ分析の狙いは，データベース構造決定やデータ重複排除だけにあるのではない．ソフトウェア重複排除やソフトウェア問題の解決にも寄与すべきものである．

今日，システム開発におけるデータモデル手法やデータ正規化手法の使用実績は多い．しかし，その結果はソフトウェア問題解決に積極的に生かされているとはいい難い．データ分析結果をデータディクショナ

リに登録しても，コピーライブラリと同等に扱い，その成果をプログラム作成作業上の効率化に求める傾向が多い．また，正規化やモデル化の手法に振り回され，データ分析の目的を見失うことも多い．例えば，正規化を行いながらも，コードデータに多重の意味付けを与えたり，プログラム間で処理結果を引き継ぐフラグを乱用させたり，プログラム内でデータを再定義したり，あるいはデータのチェック条件をプログラム別に検討している．これらに共通することは，データの曖昧さや複雑さがソフトウェアの複雑性の要因となっていることへの認識不足である．本章では，システム開発においてDOA 手法が何を目的とするかを述べながら，改めてデータ分析の意義とデータ分析手法のあり方を明らかにしたい．

4.1 DOA におけるシステム仕様定義の考え方

（1） 情報要求定義と情報生産

DOA では，要求定義も，システム仕様定義もデータ中心で行う．プログラムを直接成果物とはみなさず情報を成果物とみなす．つまり，ユーザからの要求を「ソフトウェア要求」とみなさず，「情報要求」と考える．ユーザ要求をソフトウェア要求として受け止めることは，いかに優れた CASE ツールを使おうとも，要求の変更がソフトウェアの変更を引き起こす依存関係を断ち切れないことを意味する．多様化するユーザ要求を「情報要求」として定義することが，プログラム変更を最少化することに通じる．つまり，情報要求の多様化は，求められる情報製品をデータ部品の組立（アセンブリング）仕様として定義できるからである．多品種少量生産の製造業でも，ニーズの多様化に対して製造工程の再編成で応じることは不可能である．製造工程を部品対応に用意することで，ニーズの変化の影響を製造工程に与えないように工夫している．データを部品とみなし，その工程としてプログラムを用意すべきであり，それがデータ対応のカプセル化となる．

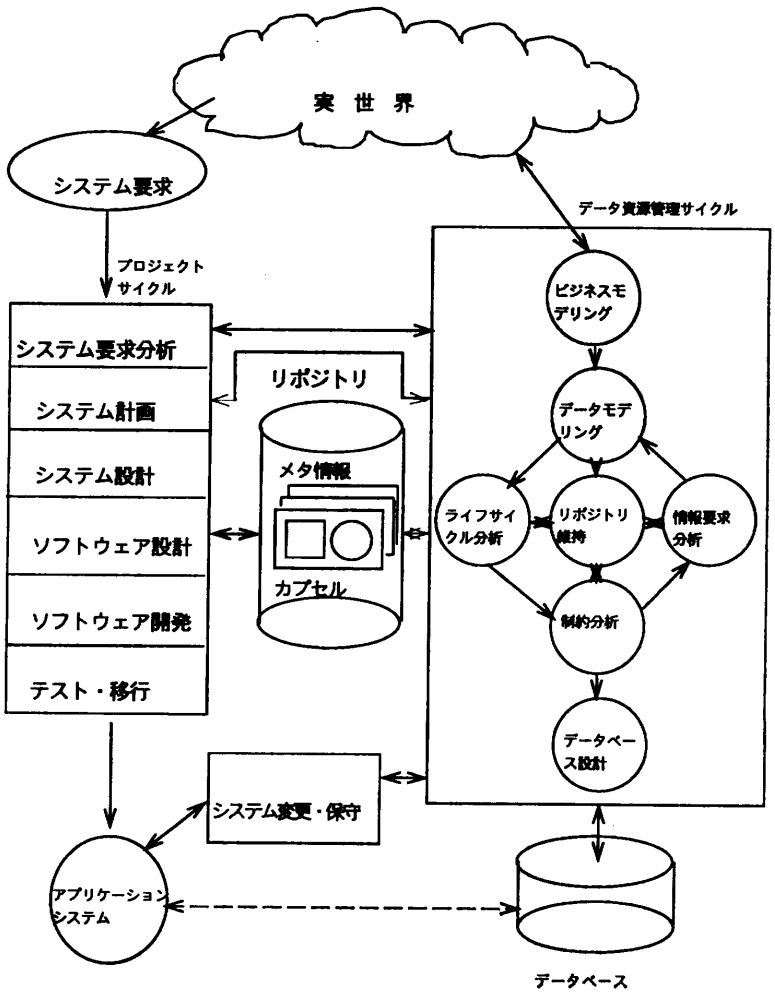

図 2 DOA によるシステム開発手順
Fig. 2 DOA system development process.

ユーザ要求はできるだけ早期に，情報要求として定義すべきである．つまり，「なになにして欲しい」を「こんな情報が欲しい」として定義すべきである．このような概念は，構造化システム分析手法などでも，データフローダイアグラム（DFD）によるシステム記述として主張されてきたことであるが，プログラムは DFD におけるプロセス（バブル）の機能展開によって求められ，データ固有処理を導く概念はなかった．

（2） 集約化によるシステム仕様定義

DOA では，システム仕様定義もデータ中心で行う．つまり，機能表現やプロセス表現に頼らず，カプセル化された部品の組み合わせ仕様として，求められるシステムを記述することを目的とする．システムは機能要素の構成として捉えることもできるが，それぞれの要素が操作するオブジェクトの構成として捉えること

もできる．操作の対象となるオブジェクトはデータの形態をとるので，機能分解手法よりも分析が容易である．システムの記述は，あらかじめ用意された基本カプセル群の上に，要求ごと，イベントごとに求められるオブジェクト（集約カプセル）とそれを導く集約操作の組み合わせとして行う(図 3)．

異なるオブジェクト群から，それらを組み合わせて別なオブジェクトを作り出す操作は，一般に，集約化 (aggregation)[15)~17)] と呼ばれる．すべての物質は，わずか 100 種余りの原子によって構成され，原子の組み合わせが分子を構成し，分子がまた物質を作り出している．このとき，例えば水素原子が，水の分子に組み込まれるときも，他のいかなる分子に組み込まれるときも，水素原子の特性は，一切変わらずに継承される．そのような組み合わせ操作を集約化操作に求める．つまり，組み合わされる要素の特性を変化させたり，部分的な抽出を伴う集約であってはならない．つまり，基本的な要素から，それらを組み合わせて有意な成果物を求める部品化は，基本要素が持つ型，固有操作，および固有制約を，そのまま変更することなく継承させるものでなければならない．継承を一時的，局所的な要請に従って動的に変化させることは，単にプログラム作成作業を支援するものであって，共有化を危うくするものとなる．

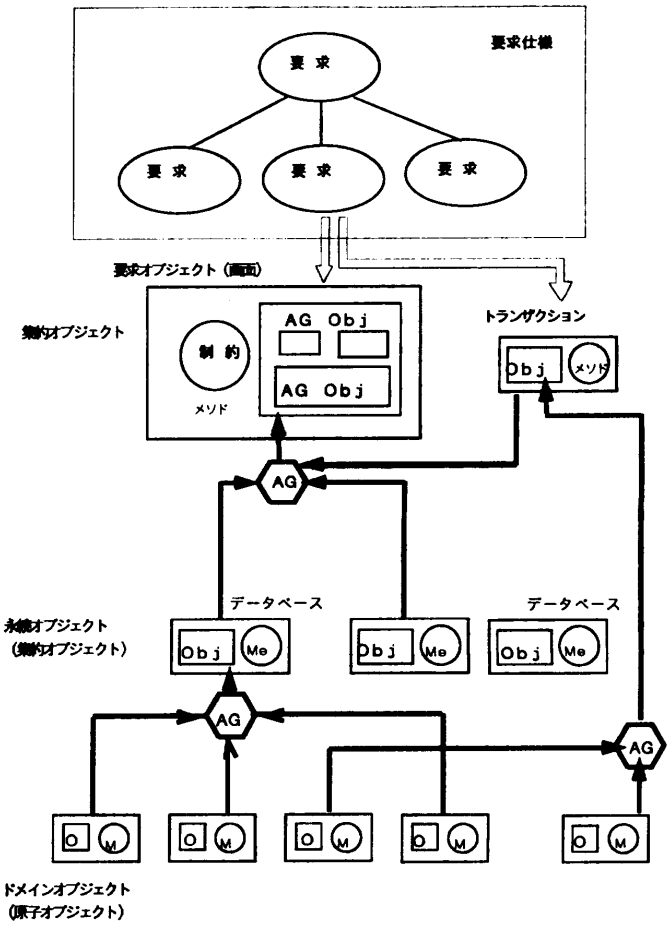

図 3　集約化によるシステム記述
Fig. 3　System specification by aggregation technique.

集約操作は，データモデルでいえば，メタデータに関するもの，つまり型 (type) の上での操作であって，直接，実現値を操作するものではない．データベース分野でビュー (view) 操作と呼ぶものに似ている．ただし，集約物もまた独立した存在として（独立したオブジェクトとして）扱われる．つまり，集約物が別の集約物に集約されることもある．

4.2　カプセル化とその原則

（1）　カプセル化 (encapsulation)

カプセル (capsule) とは，ある働きや機能を実現する手段を一つの容器に封じ込めたものをいう．その目的は，対象の独立性確保，認識の容易性，操作の容易性，あるいは内部構造の保護などに置かれる．

ソフトウェアにおけるカプセル概念には，古くからデータと処理を同時に抽象化することをねらった抽象データ型 (ADT : Abstract Data Type) がある[11)]．ただし，サブタイプ概念は持たなかった．DOA ではカプセルを構成するのに，次のような原則を用いる．

- カプセルは，一つのデータ要素について，一つ用意され，その型に合致するデータの実現値と，その実現値を操作する固有の操作とそのカプセルを成り立たせる固有の制約から構成される．
- カプセルの型は，属性の組み合わせによって識別される．
- カプセルは一つ以上の属性を持つ．
- 一つのカプセルの操作，または制約は，他のカプセル内のデータを直接，操作または参照してはならない．つまり，一つの操作，または制約は対象とするオブジェクトを唯一つだけ持つ．

（2）　集約カプセル

一つ以上のカプセルから構成されるカプセルを，集約カプセルと呼ぶ．その集約は，何らかの目的を持ち，何らかのイベントによって引き起こされる．集約

カプセルは次のように説明される.

●異なる複数のカプセル型を集めて，一つの有意なカプセル型としたもの.

●一つの集約カプセルを構成する要素として定義されたカプセルは，同時にすべて存在しなければならない.

●集約カプセルは固有の属性を持つ.

●集約カプセルも固有の操作と固有の制約を持つ.

集約カプセルを用いることで，複数カプセル相互の関連を単純化でき，カプセル構成方法と操作方法を統一でき，システムの制御も単純化できる.

例えば，カプセル「商品」と「注文」の二つの上にまたがる制約検証を考えてみたい.

カプセル型：商品

属　性：{商品番号，商品名，在庫数量，標準価格，割引販売合計}

カプセル型：注文

属　性：{注文番号，顧客番号，注文日付}

制約命題：「受注時，もし，商品在庫数量が 1,000 個を超えていたら，その受注は標準価格を 10% 割引し，割引して販売した数量を加算する」

この制約を，もし「商品」カプセルのものとすれば「注文」カプセルインスタンス生成の操作が「商品」カプセルの中に組み込まれることになる．また，もし「注文」カプセルのものとすれば「注文」カプセル内に，「商品」カプセルの割引販売数量を加算する操作が組み込まれる．両者に依存関係が生じる．もう一つの方法は，両方のカプセルに同一の制約命題を組み込むことであるが，重複となることは言うまでもない.

上記の制約命題は，「受注」というイベントに対応するものと考えるべきである．つまり，「受注」イベントに対応する集約カプセル「受注」を用意して，その「受注」カプセルに上記の制約をメソドとして組み込むことでカプセル化の原則を守ることができ，カプセルの独立性を確保できる.

このとき，集約カプセル「受注」は次のような性格を持つ.

●「受注」イベントが関心を持つカプセル群を集約している.

●「受注」イベントに対応する受注トランザクションからメッセージを受け取る.

●下位の「商品」,「注文」カプセルを制御するカプセルとなる.

●受注イベント対応の「受注」トランザクションは，「商品」,「注文」カプセルそれぞれに対応したトランザクションに分解される（トランザクション正規化と呼ぶ）.

このように，集約カプセルは，外部のイベントや要求とは独立に存在するカプセルを，イベントや要求の観点から結合させる方法を提供し，カプセルに対する統一的扱いを可能とするものとなる．集約カプセルは，イベント対応の制御や，外部へのサービス提供のためのカプセルとなる.

4.3 データ分析の課題

データ分析の目的は，上記のようなカプセル化とそれに基づくシステム仕様定義を可能とすることにある.

データ分析は外見的に異なるデータを分解し，共通要素を発見し，要素の共有関係として，データを再定義することを目的とする．その過程で，そのシステムにおいて最も基本的なデータ要素をドメインとして列挙する.

（1） ドメインの抽出と制約の定義

物質を分解して原子を発見したように，一つの企業における情報成果物を分解し，それらを構成する最も基本的なデータ要素を発見する．そのようなデータ要素をドメイン（値域）と呼ぶ．ドメインは次のような特性を持つものである.

●複数の集約物に共有され，それ以上の分解を必要としない値の集合．ただし，値は事前列挙される必要はない.

●その集合を規定する明確な制約 (constraint) を持つ.

●サブタイプを持つが，集約化構造は持たない.

情報システムを構成する要素の多くは，基本的なオブジェクトであるドメインの集約物として定義される．データベースとしての表（テーブル）は列（カラム）の集約物である．列はドメインのサブタイプとなる．画面も同様に，ドメインの集約物として定義可能であり，ドメインの特性（制約）を確実に継承する．データ分析作業では，まず画面や帳票類からデータ項目を抽出し列挙した後，それらの横にらみを通じてドメインの抽出を図る．**表1**は列挙されたドメインの例である．ドメインはそれぞれ固有の制約を持つ．固有の制約とは，制約として定義された命題のすべてが，いつでも成立していなければならないものを指す．つまり，操作の状況によって，部分的な成立を許すべきものではない.

（2） 実体の抽出とモデル化

カプセル概念によるデータ部品化は，その部品化がプログラムの部品化よりも有効で効率的であることが前提となる．これまでも，プログラムの自動合成を目的に多くの努力がプログラム部品化に向けられてきた．しかし，実務的に製造すべき対象物を列挙できるか，またはその対象物の存在範囲を特定できなければ，共通項を導くことはできない．一つの企業における，プログラムの事前列挙は不可能に近いが，データについては，実世界の制約に基づいたモデル化を通じて，実体列挙やその存在範囲が特定可能である．

DOA では，現存するシステムにおける情報要求（画面，帳票）を先に洗い出し，次のような手順で実体抽出を行う．

①データ項目の抽出
②ドメインの抽出
③データ正規化
④実体の抽出と実体関連の把握
⑤サブタイプの検出
⑥データ標準名称の確定

実体関連の把握は，E/R 図でもバックマン線図でもよい．課題は正規形データから帰納的に得られる実体を，その属性構成や，具備すべき制約を分析して，そのサブタイプやスーパタイプを発見することに重点を置く（**図 4** 参照）．

（3） 基本集約物の定義と制約の定義

データモデル図による実世界の表現の狙いは，情報システムが表現すべき実体とその汎化階層を明らかにすることにあった．それらの実体の多くは，情報システム内で，データベースとして記録されるものとなる．しかし，ソフトウェアは，画面，トランザクション，テーブル類など，データベース以外のデータも操作する．それらのデータもすべて，ドメインかその集約物に基づいて定義できる．そのような定義を通じて，外見的に異なる形態をとるデータに，共通制約の束縛などの基盤を与え，システム一貫性の維持を容易とし，一つの事実の変更に伴う波及を最小化できる．**図 5** はそれら異なる形態のデータの関連をメタ構造として示したものである．

表 1 ドメイン一覧の例
Table 1 Sample of domain list.

ドメイン区分	役割	ドメイン	略号	形　式
時間	日付	和暦日付	DT-J	TT/YY/MM/DD
		西暦日付	DT-W	YYYY/MM/DD
		ユリシーズ暦	DT-Y	DDD
		年月	YM	YY/MM
		月	MM	MM
	時間	12時間時刻	T12	HH/MM/SS
		24時間時刻	T24	HH/MM/SS
		時分	HM	HH/MM
		日数	DS	DDD
量	個体量	個数	QT	NNNNNN
		ダーズ	DZ	NNNNNN
	流体量	キロリットル	KL	LLLLL
		ガロン	GL	LLLLL
		CC	CL	CCCCC
	重量	トン	TG	TTTT
		キログラム	KG	KKKK
名称	人名	姓名	NM	漢字姓名
		組織内略称	AN	最大4文字のカナ
	組織名	企業名漢字	EN-J	漢字の法人名
		企業名英字	EN-E	英文表示の法人名
		部門名	DPN	漢字表示の部門名
場所	地名	国内地名	LC-J	住居表示法による表示
		海外地名	LC-W	英文字による表示
識別	コード	商品コード	GC	詳細を別に定める
		受注コード	OC	同　上
		顧客コード	CC	同　上
		部門コード	DC	同　上
		社員コード	EC	同　上
	区分	商品販売分類区分	GGC	販売統計用分類区分
		商品検査分類区分	GIC	検査結果区分
		商品保管状態区分	GSC	倉庫保管状況区分
		商品扱い区分	GAC	扱い担当区分

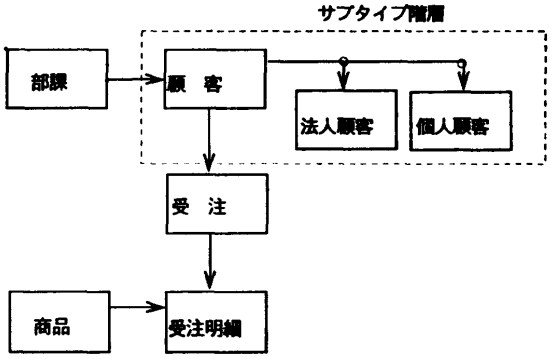

図 4 実体関連図
Fig. 4 Entity-Relationship.

（4） データライフサイクルの分析と定義

データ分析の課題は，基本的要素の発見を行うことにある．しかし，その基本的要素としての特性は，データの型だけでは特定できない．その動的な振る舞いや制約条件を明らかにすることで，はじめて可能と

なる．DOA では，正規化や汎化操作などをデータの型に基づいて分析した後，その基本要素の固有操作として，その要素の発生から消滅にいたるイベントを明らかにし，個々のイベントに対応した操作を定義する．

基本要素のライフサイクルに対応した操作を DLCP (Data Life Cycle Process) と呼ぶ (図 6)．実務的には，DLCP はコンピュータ処理能力を考慮して，データベース(表) 単位に用意する．

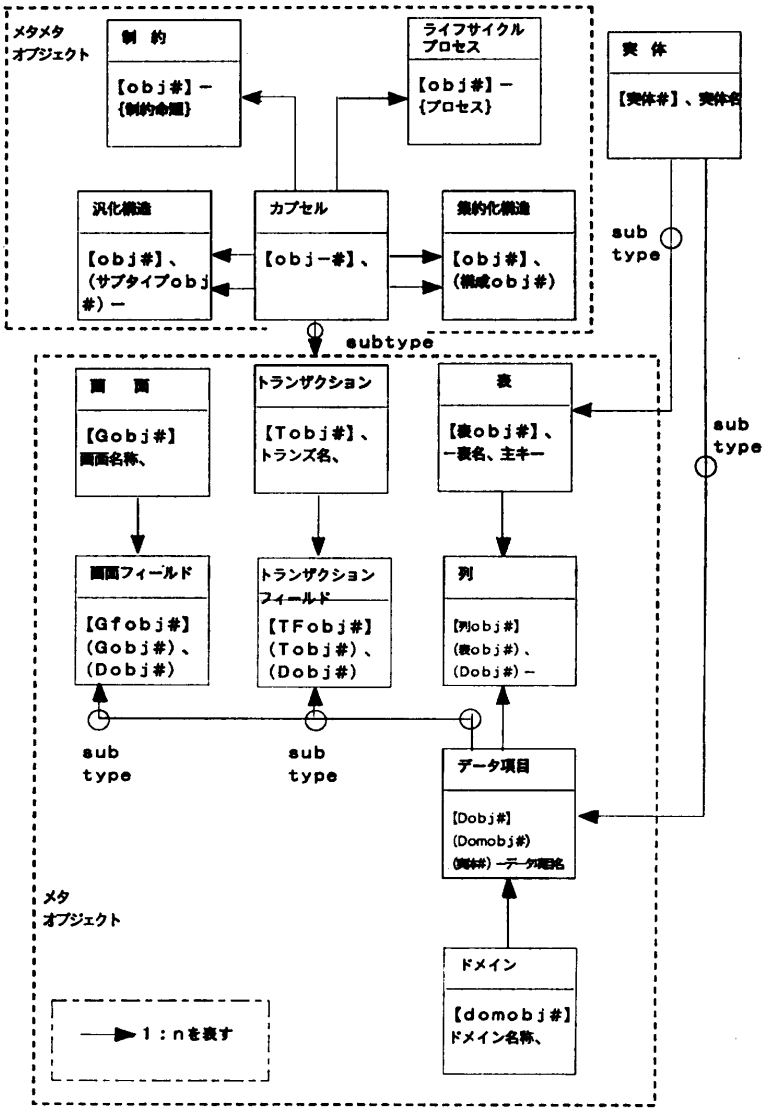

図 5 システム構成オブジェクトのメタ情報
Fig. 5 Meta-Information of software system.

(5) 制約の分析と定義

a．カプセル制約の分類

データ分析を通じてデータ固有の制約を明らかにしてカプセル化するためには，次のような分類をもって制約を分析することが必要となる．

①対象による分類

カプセル制約は，まず大きく2種類に分けられる．イベント対応の制約とカプセル対応の制約である．

- イベント制約：イベントと1対1対応に存在する制約である．多くの場合，イベントに対応する集約カプセルを用意してカプセル化する．
- オブジェクト制約：カプセルごとに存在する制約である．カプセルに対して1対1でカプセル化する．

②検証内容による分類

カプセル（集約カプセルも含む）制約の検証内容による分類には次のようなものがある．

- 形式制約：カプセルの型を検証する制約．
- 存在制約：カプセル実現値の生成，削除に関する制約．
- 値制約：カプセルの属性値を規定する制約．

③検証実行時による分類

イベント制約に関するもので，その制約検証を実行させる時点による分類で，次のように分けられる．

- プレ制約：特定なイベントにより実行されるアクションの実行前に検証すべき制約．
- ポスト制約：同様に，アクションの実行後に検証すべき制約．

b．制約の継承と検証実行

カプセル化すべき制約の組み込み先と，その検証実行時点は異なる．制約のカプセル化と制約命題実行指示のカプセル化は異なる．継承により集約カプセルに制約を引き継いで，その検証が必要とされた時点で実行することを意味する．前述のカプセル構成概念を例に，その位置づけを探ってみたい．

[想定するデータカプセル]：

顧客：顧客番号，顧客名，契約番号，住所

注文：注文番号，顧客番号

契約：契約番号，契約日付，契約区分

商品：商品番号，商品名，在庫数量，輸出区分

仕入先：仕入先番号，仕入先名，仕入先住所

[想定する画面]

受注画面：受注番号，顧客番号，受注日付
{商品番号，注文数量，注文単価}

[制約例]：

①「注文番号は7桁の英数字であること」：この制約は，「注文」カプセルの形式制約である．検証実行タイミングは，注文インスタンス生成時．

②「受注受付時，新しく付与する「注文番号」はすでに存在しないものであること」：この制約は，「受注受付」というイベント対応のものである．したがって，それを扱う画面カプセル中の「注文番号」カプセルのものである．ただし，「注文」データカプセルの「注文番号」の形式制約を引き継ぎ，「受注受付」イベント発生時に実行する．

③「新しい顧客を顧客カプセルに登録するとき，その顧客がすでに存在しないこと」：この制約は，「顧客」カプセルの存在制約である．ただし，この制約は，顧客登録画面カプセルに引き継がれ，画面中の「顧客番号」カプセルの入力時実行されるものとなる．

④「受注が禁輸出品に対するものであるときは，その顧客の契約区分が，特別なものとなっているかを確認する」：この制約は，「注文」，「顧客」，「契約」の三つのカプセルにまたがるものである．したがって，三者の関係を検証する制約は，集約カプセルとして「取引」を定義して，そのカプセルのものとする．「取引」カプセルは注文トランザクションを受け付けるカプセルとなる．

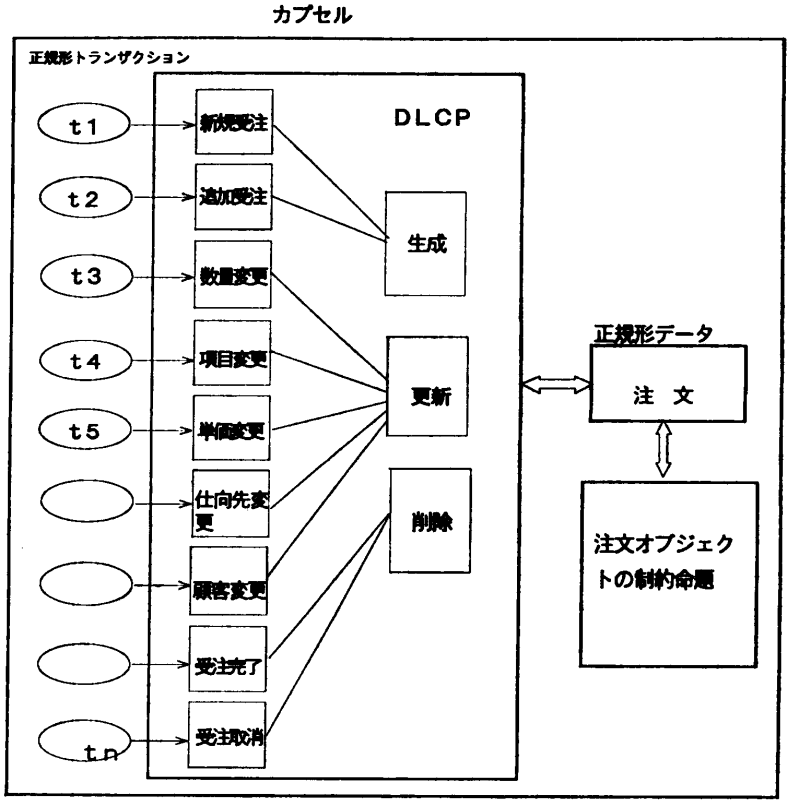

図 6 DLCP (Data Life Cycle Program)
Fig. 6 DLCP (Data Life Cycle Program).

既に，一部の DBMS では，制約の定義と実行を DBMS 側で負担するものがある．トリガやアサーションによって定義できるものもある．また，SQL の標準化でもトリガ，アサーションは論議されている．しかし，DBMS に対する制約定義は，ともすると，イベント対応に必要とする制約すべてを一つの手続きとして定義しがちである．そのような定義は手続きの間に制約の重複を生むものとなる．DOA では，まずデータカプセルに固有の制約を先に定義し，そのイベント別の検証は，イベントごとに求められる集約カプセルを明らかにしてから，その集約カプセルに継承させてから実行することを考える．

4.4 トランザクション分析とカプセル制御階層

（1） トランザクション正規化

データベースなど基本的なカプセルは外部のイベントや要求とは独立な存在である．それらのカプセルの内容を更新するトランザクションの発生も独立である．DOA では，イベント対応に発生するトランザクションを，更新対象となる基本的カプセル単位に，分解してから引き渡す(図 7)．これをトランザクション正規化と呼ぶ．その狙いは，外部に依存するトランザクションの制御をカプセル内に持ち込むことを避けることにある．つまり，データに基づくカプセル凝固性 (cohesion) を損なわないように，トランザクション制御

は，トランザクションによって引き起こされる，そのトランザクションに固有な集約カプセルにまかせる．

（2） カプセル制御階層

DOA では，各カプセルはカプセル外部に，そのカプセルを制御する操作を持つ．オブジェクト指向で考えられるオブジェクト概念の多くが，オブジェクトに当該オブジェクトの制御をメソドとして内包した自律オブジェクトを想定しているのに対し[16),17)]，カプセル間でのメソド重複を回避できる．つまり，二つ以上のカプセルが，共通する制御メソドを外部に持つことが制御の重複と不整合を回避させるものとなる．

DOA では，制御の共通化を通じて得られるカプセル階層をカプセルの制御クラスと呼ぶ．制御クラスはイベントや要求対応に多様なものが存在する．

トランザクション正規化を前提に，オンラインシステムにおけるトランザクション処理で必要とされるカプセル制御クラスを示したものが図 8 である．

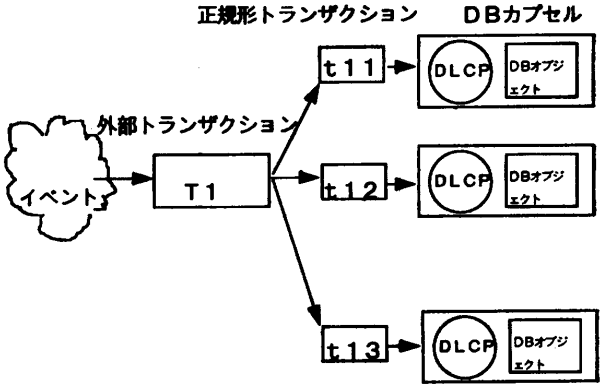

図 7 トランザクション正規化
Fig. 7 Transaction normalization.

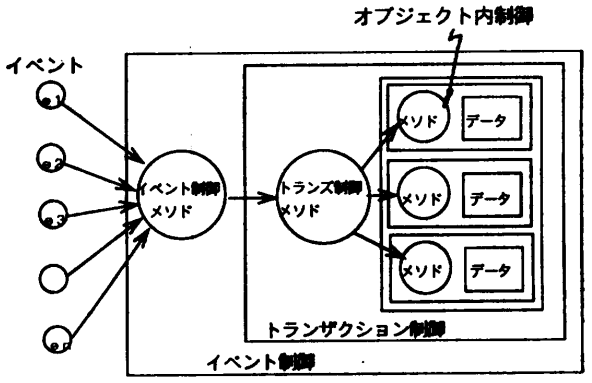

図 8 カプセルの制御階層
Fig. 8 Control class of capsule.

5. 結　　言

以上，これまでのソフトウェアエンジニアリング手法が持つ問題点を指摘しながら DOA におけるデータ分析の意義と手法の狙いを述べた．特にシステム再構築のために，カプセルを共有資源とみなした重複排除と一貫性維持は，極めて有効な方法論となる．DOA の適用事例は数多いなかで，データのモデル化にとどまり，そのデータ分析の成果がソフトウェア構築方法に反映されない事例が多いのも，データモデル化の狙い，あるいは志が理解されず合意を得られていないことにある．本論では，既にオブジェクト指向の分野では常識とされている手法も含めて，あらためて DOA 手法の特長を述べた．

参 考 文 献

1) ジャクソン，M. A.：プログラム設計の原理，日本コンピュータ協会 (1976).
2) ワーニエ，J. D.：ワーニエプログラミング法則集(鈴木君子訳)，日本能率協会 (1974).
3) Myers, G. J.: *Composite/Structrured Design*, Van Nostrand Reinhold (1978).
4) Jackson, J. M.: *System Development*, Prentice-Hall (1982).
5) デマルコ，T.：構造化システム分析とシステム仕様（高梨智弘ほか訳)，日経 BP 社 (1986).
6) Ward, P. T. and Mellor, S. J.: *Structures Development for Real-Time Systems*, Yourdon Press (1985).
7) ハートレイ，D. J.，リバイ，I. A.：リアルタイム・システム構造分析（立田種宏訳)，日経 BP 社 (1989).
8) Nijssen, G. M. and Halpin, T. A.: *Conceptual Schema and Relational Database Design—Fact Oriented Approach*, Prentice-Hall (1989).
9) Longrth, G. et al.: Structured Systems Analysis & Design Methodology Manual, NCC Publication (1987).
10) Martine, J.: *Strategic Data Planning Methodologies*, Prentice-Hall (1982).
11) Liskov, B. et al.: Specification Techniques for Data Abstraction, *IEEE Trans. Softw. Eng.*, Vol. SE-1, No. 1 (1975).
12) Nolan, R. L.: Managing Crises of Data Processing, *Harvard Business Review*, Vol. 3, No. 4 (1979).
13) デュレル，W.（IRM 研究会訳）：データ資源管理，日経 BP 社 (1987).
14) ホートン，W. F.：インフォトレンド，オーム社 (1988).
15) Smith, J. M. and Smith, D. C.P.: Database Abstraction ; Aggregation and Generalization, *ACM TODS*, Vol. 2, No. 2 (1977).

16) Shlaer, S. and Mellor, S. J.: *Object-Oriented System Analysis*, Yourdon Press (1988).
17) Coad, P. and Yourdon, E.: *Object-Oriented Analysis*, Yourdon Press (1990).
18) 酒井博敬，堀内 一：オブジェクト指向入門，オーム社 (1989).
19) 堀内 一：データ中心システム設計，オーム社 (1988).
20) 堀内 一：システム開発パラダイムと高水準データモデル，情報処理，Vol. 32, No. 9 (1991).
21) Sakai, H. and Horiuchi, H.: A Method for Behavior Modelling in Data Oriented Approach to System Design, *IEEE COMPDEC* (1984).
22) 堀内 一：CASE におけるデータ中心アプローチの意義，情報処理学会データベース研究会 (1989. 5).
23) 堀内 一：データ中心システム設計—その必然性と可能性—，情報処理学会情報システム研究会 (1988. 11).
24) 味村重臣，山田 進，堀内 一：データベースシステムの設計と開発，オーム社 (1983).
25) 山田信雄，堀内 一：カプセル化手法を用いた情報系データベースの一貫性維持について，情報処理学会利用者指向の情報システムシンポジウム (1989).

(平成2年12月13日受付)
(平成4年 1 月17日採録)

堀内 一 (正会員)

昭和43年早稲田大学商学部卒業．同年(株)日立製作所入社．(株)日立製作所コンピュータ事業部およびビジネスシステム開発センタを兼務．教育本部で社内外のデータベース教育およびコンサルテーションなどを経て，現在はビジネスシステム開発センタで CASE コンセプト開発およびデータ中心設計技法のコンサルテーションに従事．同社のシステム開発技法 HIPACE, HIPLAN の開発にも従事した．研究テーマ：データモデリングおよびデータ中心システム設計技法など．著書：「データ中心システム設計」，「オブジェクト指向入門」，「データベースシステムの設計と開発」など．情報処理学会情報規格調査会「概念データモデル化機能専門委員会」幹事，日本規格協会「情報資源スキーマ調査研究委員会」幹事．システム監査学会，経営情報学会各会員．