データベース・システム 78−17
（1990. 7. 20）

エンジニアリング分野におけるデータベースフレームワークについて

石川 博　泉田 義男　川戸 信明

株式会社富士通研究所

CAD/CAM/CAE やEOA (Engineering Office Automation) といったエンジニアリング分野においては，従来のリレーショナルデータベースとは異なるデータベースのフレームワークが求められている。そこで先ずCAD/CAM/CAE やEOA における技術的要件を洗いだし，(1)設計データの管理，(2)ハイパーメディア，(3)制約管理，(4)分散協調の4つにまとめる。次にフレームワークとしては，最近注目されているオブジェクト指向型データベースを取り上げ，その基本的特徴について考える。その上で前述の技術的要件を，純粋なオブジェクト指向型データベースでどこまで解決できるか，さらにそのままでは解決できない問題に対しては何をオブジェクト指向型データベースに追加すべきかについて論ずる。

ON A DATABASE FRAMEWORK FOR ENGINEERING

Hiroshi Ishikawa　Yoshio Izumida　Nobuaki Kawato

Fujitsu Laboratories, Ltd.

1015 Kamikodanaka, Nakahara-Ku, Kawasaki 211, Japan

Engineering domains such as CAD/CAM/CAE and EOA (Engineering Office Automation) require a new database framework different from conventional relational databases. We analyse requirements for engineering domains and classify them into design data management, hypermedia management, constraint management, and coordinated distribution. Object-oriented databases seem promising for a database framework satisfying these requirements. So we discuss how far pure object-oriented databases cover these requirements and what they need to satisfy uncovered requirements.

1. はじめに

CAD/CAM/CAE やEOA(Engineering Office Automation)といったエンジニアリング分野において，そのフレームワークとしてどんなデータベースが要求されているかについて論ずる。そこでまずCAD/CAM/CAE やEOA において要求されている機能や技術について洗いだす。次にデータベースフレームワークとして，最近注目されているオブジェクト指向型データベース(OODB)〔ISHI90〕〔ATKI89〕をとりあげ，OODBと言われているものの共通項としての特徴と具体的なシステムについて触れる。その上で純粋なOODBで要求事項をどこまで解決できるかを述べ，さらに何をOODBに付け加えるべきかを論ずる。

論文の構成についてのべる。2章では，エンジニアリング分野におけるデータベースフレームワークの要件について述べる。3章では純粋なOODBとそれを用いた設計データの管理について述べる。4章，5章及び6章では要件としてのハイパーメディア，制約管理及び分散協調の技術に対してOODBでどうアプローチするかについて述べる。

2. データベースフレームワーク

この章では，エンジニアリング分野をCAD/CAM/CAE とEOA とに大きく分類し，それぞれの特徴とそこで要求されるデータベース関連技術について論ずる。先ず第一にCAD/CAM/CAE では文字数値で表現される設計に一次的な情報（設計データ）の管理が中心である。さらに設計仕様として，設計データに関する制約の管理が必要になる。そのためには設計対象のモデル化と設計手順の記述が重要になる。また複数の設計者が協調して，最終的に1つの設計対象を設計する点で分散協調的技術が必要となる。

一方EOA は，CAD/CAM/CAE のように設計の本質部分ではないが，設計の業務全体では，設計より遙かに多くの時間を費やしていると考えられる。EOA では，設計の二次的情報である文書の管理が必要になる。設計文書には，テキスト，グラフィクス，イメージ等のメディアを使用することが要求される。そこで関連する情報の間の一貫性といった制約の管理が必要になる。情報の利用の仕方の中心は，検索であり，その加工である。

もちろんこうした分類は必ずしも明確な境界をもたない。そこで以下に両者で要求される技術をまとめて整理してみる。

(1) 設計データ管理

通常の設計対象は，部品階層という複雑な構造を内在的に持っている。そうした構造を直接的に表現したり，複雑さを軽減したりする概念が必要になる。それらの概念とは，部品関係であり，汎化関係である。設計データを管理するにはこうした抽象化のメカニズムを提供する必要がある。

(2) ハイパーメディア

設計文書では，文字数値だけでなく，グラフィクスやイメージといった異なるメディアを互いに関連させて管理し，利用できるような仕組みが必要になる。いわゆるハイパーメディアの管理が要求される。さらに従来のハイパーメディアでは，データ間のリンクをたどる検索機能だけを提供してきたが，大規模な設計データには，集合的な検索や，手続きを使った検索・加工等の高度な機能が要求される。

(3) 制約管理

設計仕様としての制約管理では，属性の値の範囲や，複数の属性間の関係などの制約を記述できる必要がある。さらにそれらの制約条件を充足するための手段を記述できる必要がある。即ち設計手順の管理である。さらにできあがった情報間の関連を維持し，関連する情報の一方が変わったら，他方も自動的に変わるとか，変更の旨を通知するという技術が要求される。

(4) 分散協調

通常は，1つの設計対象を複数の設計者が分担して設計する。各設計者は，各自の分担する部分を，独自の用語（概念）で設計する。各部分は独立したものでなく，互いに関連するものであるから，その設計したものは，他の設計者から参照される。そこで基本的に情報の共有ができる必要が

ある。さらに独自のビューを保持しつつ，他者にも別のビューを定義することを許すような分散のモデルが必要である。

以上述べた要件は独立したものではなく，互いに関連しあっている。以下ではこれらの要件について詳しく論ずる。

3. OODBと設計データ管理

先ず設計データが内在的に持つ複雑な構造は2つの概念を要求する。ひとつは部品関係であり，もうひとつは概念的な分かり易さのための汎化関係である。さらに大規模な設計データを二次記憶を使って管理する必要がある。設計データの動的性質としてプログラムが定義できる必要がある。さらに設計データを用いる応用全体をプログラミングできる必要がある。これらに対してOODBのどの機能が利用できるかについて述べるために，先ずOODBの一般的な特徴について説明する。

3.1 OODBの特徴

純粋な（裸の）OODBの共通項として，複合オブジェクト(complex object)，オブジェクト識別子(object identity)，クラス階層(class hierarchy)，カプセル化(encapsulation)，ポリモルフィズム(polymorphism)，永続性(persistency)，問い合わせ言語(query language)，プログラミング機能(computation)，コンカレンシー(concurrency)，リカバリ(recovery)，拡張性(extensibility)がある〔ATKI89〕。

・複合オブジェクトは単純なデータを組み合わせて，より高次のデータを定義していく機能である。
・オブジェクト識別子は値以外でオブジェクトの存在を示すものである。
・クラスはオブジェクトの型に相当し，クラス階層は，汎化関係そのものである。汎化関係をとおして，下位のクラスは上位のクラスから属性の定義を遺伝することができる。
・カプセル化はオブジェクトにアクセスするのに，決められたプロトコル（クラスに定義されたメソッド）以外ではできないことである。
・ポリモルフィズムは，上位のオブジェクトのメソッドに対し，その下位のオブジェクトでそれらに独自の実現関数を対応させることである。
・永続性はもちろん二次記憶をもちいてオブジェクトを管理することである。
・問い合わせ言語は，オブジェクトの集合的な検索や，アドホックな問い合わせを処理するためのものである。
・プログラミング機能はいわゆる応用の記述のための機能である。
・コンカレンシやリカバリは基本的なデータベース管理機能の一部である。
・拡張性は，ユーザによるオブジェクトの構造や操作の定義がしやすいことである。

OODBの具体的システムとしては Jasmine〔ISHI90〕，O_2〔LECL88〕，IRIS〔FISH87〕，Orion〔BANE87〕などがある。

3.2 設計データの管理

OODBをもちいて設計データを管理することを考える。OODBにおける複合オブジェクトの考えに従えば，いわゆる部品関係はオブジェクトの属性を使って実現でき，複雑な設計対象を直接的に表現してくれる。汎化関係も，OODBにおける遺伝機能を提供するクラス階層そのものであり，設計対象の複雑さを軽減してくれる。しかしながら，ここで注意すべきことは，汎化関係に伴う遺伝という機能が応用分野によらず，ほぼ一様な意味づけをもっているのに対し，部品関係に対して各応用が与える意味づけ，すなわちどんな機能を期待するかということはまちまちであることである。従って純粋なOODBとして部品関係のセマンティクスを内蔵しているものはない。そのかわり Jasmineではデモンというかたちで，ユーザが属性に対して，応用独自の意味づけを与えることを許している。これは Jasmineのユニークな特徴の1つである。これについては5章の制約管理のところで詳述す

る。

設計対象の動的性質は，OODBのメソッドとして実現できる。メソッドはそうした応用に依存した機能ばかりでなく，検索・更新といった汎用的なデータ操作も提供し，オブジェクトとしてのカプセル化を実現する。応用全体としては，データの操作を行う必要があるが，その中で最も重要なことはデータの柔軟な検索である。OODBはいずれも問い合わせ言語を提供し，多様なデータのなかから条件検索ができるようになっている。しかしながら，メソッド実行も集合的に行うことができる点がJasmineのユニークな特徴である。ポリモルフィズムや拡張性はインクリメンタルな設計に役立つ。さらにOODBを使った設計業務全体をプログラミングすることができる。

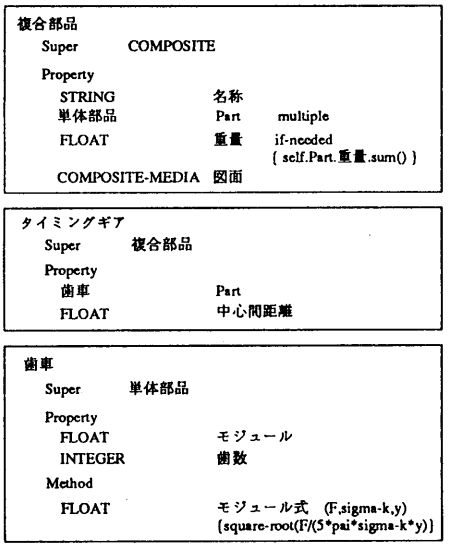

図1　設計対象のオブジェクトモデルの例

このようにして定義された設計データ及びその操作は，OODBによって二次記憶をもちいて効率的に管理される。例えば，図1は機械設計分野における設計対象をJasmineオブジェクトによりモデル化したものの一部である。複合部品のPart属性は，複合部品品が複数の単体部品から構成されていることを示している。また重量属性には，属性アクセスによって起動されるif-needed デモンが付加されていることを示す。タイミングギアのSuper 属性はタイミングギアが複合部品のサブオブジェクトであり，複合部品の属性を遺伝することを示している。歯車にはモジュールを計算する式がメソッドとして定義されている。

以上述べた設計データの管理と操作は純粋なOODBで基本的に対応できる。そこで以下の章では純粋なOODBだけでは，対応できない他の要件に対して何を付け加えるべきかについて述べる。

4.　ハイパーメディア

4.1　メディア管理

EOA はもちろんCAD/CAM/CAE にも，ユーザインタフェースとしてハイパーメディアは重要である。先ず純粋なOODBに対してメディア管理の機能が追加される必要がある。基本的なメディアの型としてはグラフィクス，イメージ，テキストが最低限必要になる。それらは，基本的にはオブジェクトとして定義する。そこでOODBはある程度の長いデータをそのままの形で格納できる必要がある。メディアの操作はオブジェクトのメソッドとして実現する。

オブジェクトの表示とオブジェクトの構造やデータの内容は関連はするが，一応は別物である。つまり表示と構造・内容の間の対応関係は自明なものではない。したがって通常は，応用でデータを表示するとき，何をどのように表示すべきか設計する必要がある。例えば，応用プログラムを表示するときは，その名前やその内容を表すアイコンとしてのイメージを表示したりする。複合オブジェクトとしてのデータの表示には，データ属性名と値の組を表示する。いずれにせよ複合したメディアの管理と表示が必要になる。つまり，一般にオブジェクトの表示には，イメージ，グラフィクス，テキストなどを複合したメディアが必要であり，それらは，部品関係を用いて複合オブジェクトとして管理する必要がある。

4.2 メディアの直接操作

メディアが表示されると，それをポイントして関連するメディアに直接アクセスすることが必要になる。この機能を直接操作と言う。直接操作には，複合オブジェクトとしてのメディアの他にその構成要素としてのメディアの位置情報を管理する必要がある。そうした情報は基本的にはオブジェクトとして実現すれば良い。それらをまとめるのがウィンドウオブジェクトである。

ウィンドウ上の表示されたオブジェクトをマウスでクリックすると，その点を含む表示領域を持つオブジェクトを前述の位置情報から選び出す。選択されたオブジェクトに対する操作は，そのオブジェクトに対して，メッセージを送って実現できる。システムは，こうした基本的なメディア及びウィンドウ管理の機能を提供し，ユーザがそれらを組み合わせて，独自のユーザインタフェースを実現できるようにする。

テキストの内容からオブジェクトを検索することも考えられる。つまり，表示テキスト中にあるオブジェクトをクリックするとそのオブジェクトが表示される。これに対しては前述のようにテキスト中のオブジェクトの位置情報を管理することが考えられる。一方，テキスト中のオブジェクトを表す文字列と，それの指示するオブジェクトとの対応表をテキストとは別に管理し，テキスト中の文字列を切り出して，それをもとに対応表を検索し，対応するオブジェクトを求めるという方法もある。この方法では，実テキストを変更せずにオブジェクトを追加できるという柔軟性がある。またテキストの変更や再表示に対して，位置情報を再計算する必要がないという利点がある。

4.3 メディアの柔軟な検索

ハイパーメディアにおける最も普通のアクセス方法は，データ間のリンクをたどる方法である。これは基本的にオブジェクトの属性をたどればよい。しかし，エンジニアリング分野のデータの特徴は，それが複雑かつ大規模であるという点である。したがってリンクをたどるアクセスだけでは十分ではなく，複雑な条件を使った検索や手続きを使った検索などが必要になる。それにはOODBの問い合わせ言語が利用できる。特にメソッドを問い合わせのターゲット部や条件部に指定して，集合的にアクセスできる点が Jasmineのユニークな特徴である。メソッドは，他の属性を組み合わせて新しく属性を導出することにも使える。OODBはアドホックな問い合わせをインタラクティブに解釈もできるし，あらかじめプログラミングしておき，コンパイルして効率よく実行することもできる。

例えば，図2はハイパーメディアインタフェースにおけるメディアの表示とアクセスの例である。この例ではタイミングギアの集合から，その中心間距離が７０㎜未満のものを検索したあと，該当するオブジェクトを表示し，さらにその図面を直接指示によって表示したところである。

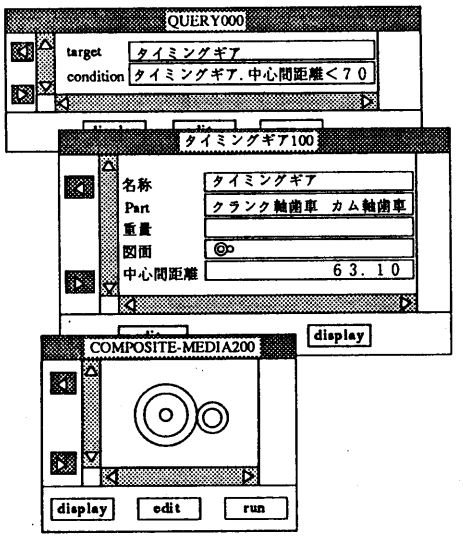

図2　ハイパーメディアインタフェースの例

5. 制約管理

制約管理は以下の2つの仕事に分類される。第一の仕事は，設計の際に，設計対象の属性の値や，複数の属性間に制約条件を与え，その制約条件を

満たすように設計を進めていくことである。この意味の制約を設計制約という。第二の仕事は，いったん設計した対象において，関連する情報の一方の変更を他方へ伝播することである。この意味の制約を一貫性制約という。前者の設計制約の管理については，純粋なOODBの上にかなりの機能を追加する必要がある。一方，後者の一貫性制約の管理については，純粋なOODBの機能でかなりカバーできる。先ずこれから述べることにする。

5.1 一貫性制約の管理

部品関係を例にして考える。部品関係によって関連づけられた2つのオブジェクトを考え，親部品と子部品とする。子部品の仕様が変更になると，それをもちいている親部品を表示したり，その親部品の関連する属性を再計算したりする必要がある。言い換えると，これは複合オブジェクトの内部的な制約をどのように維持するかという問題である。この問題に対しては，OODBのなかでは，Jasmineだけが持つデモンという機能を用いて解決できる。例えば，親部品の重量は子部品の重量の和であるという制約があるとする。この場合は，親部品の重量属性にif-needed デモンを設け，それに全ての子部品の重量の和を計算させるプログラムを定義する。親部品の重量属性にアクセスする度にデモンが起動され，最新の値が計算される（図1参照）。

別の例として，親部品の前に子部品はなく，親部品の後に子部品はないという制約を考える。これを実現するには，親部品のインスタンス化(instantiate) のためのメソッドのafter デモンに子部品をインスタンス化する手続きを記述し，親部品の消去(destroy) のためのメソッドのbeforeデモンに子部品を消去する手続きを記述する。このように，デモンをメソッドに記述できる点もJasmineの特徴の1つである。この例で注意すべきことは，この制約はすべての応用について必要なわけではないことである。つまり既存の部品を組み合わせて設計を行うボトムアップの設計ではすでに子部品が存在していることが前提になっているので前述の一環性制約は満足する必要がない。つまり部品関係に関する意味づけは，各応用によって異なるので，それをユーザが記述できるデモンのような仕組みを提供する必要がある。

5.2 設計制約の管理

設計仕様としての，属性値や複数属性間の制約条件の記述に対しては，OODBにかなり追加をする必要がある。先ず設計仕様として，制約条件を記述できる必要がある。そしてその制約条件を満たす解を生成する手段を記述する必要がある。この制約条件は，単にチェックするためだけにあるわけではない。制約条件が最初から満たされることはまれである。制約条件が満たされないときに，ユーザはどうするかということも記述できる必要がある。つまり，制約条件の充足のための手段，充足すべき制約条件，及び制約条件の充足失敗時におけるアドバイスの3つを設計ゴールとして，記述できる必要がある。さらにそのアドバイスには設計ゴールを含む。ユーザがこの設計ゴールを定義したら，システムはそれを用いて，制約推論を行う。システムはその推論機構を提供する必要がある。推論で制約の充足が失敗すると，いわゆるバックトラックが必要になる。これは，ユーザが設計ゴールのアドバイスの部分で記述したゴールを実行するので，ユーザがバックトラックを制御できるという意味で知的バックトラックと呼ばれる。システムは，この知的バックトラックの機能を提供する必要がある。

制約推論に関しては，制約プログラミング〔LELE88〕という手法があるが，これは制約条件の記述だけをユーザが行い，あとの制約充足はすべてシステムがやるというものである。つまり，制約充足の仕組みがブラックボックスになっている。しかしながら，設計においては失敗後の処置も含めて，その手順が設計のノウハウの1つなので，こうしたブラックボックス化は不都合であり，そのかわり，設計ゴールのようにノウハウをユーザが直接に記述し，それを解釈するメカニズムをシステムが提供する必要がある。

図3は設計ゴールの例である。モジュールを決定するのに，設計メソッドとしては標準モジュールの表を検索することを指定している。制約の充足が失敗したら，モジュールの値を増やすようアドバイスが与えられている。このゴールは3つの変数に依存していることを示す。

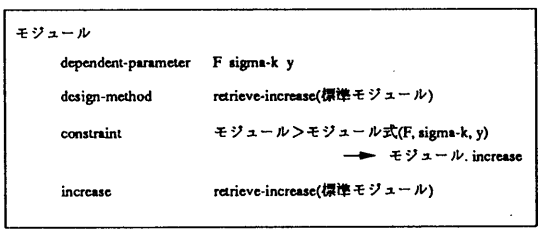

図3　設計ゴールの例

6. 分散協調

複数の設計者が，協調して仕事のできる環境を提供する必要がある。それには，先ずひとまとまりの設計対象を管理するコンテクストを定義でき，それを複数の設計者が共有できる必要がある。これは Jasmineでは，知識ベースという単位に相当し，ユーザは一度に複数の知識ベースを利用できる。すなわちユーザは共有の知識ベースと，各自の知識ベースをかさねあわせて利用できる。この機能は基本的に純粋なOODBの範囲でできる。

一方，設計者は，他の設計者の設計結果を参照するときに，自分の概念で参照したりすることがある。逆に自分の概念で設計したものを，他の設計者に参照させるために共通の仕様に変換したりする必要がでてくる。いずれにせよ，定義した側と，参照する側で概念の間の相互変換のルールを記述する必要がでてくる。これは分散したノードにあるオブジェクトの属性間の制約関係に相当する。こうした機能は，自律分散型データベースモデル〔HEIM85〕の拡張の一種と考えることもできる。

例えば，図4は概念の変換のルールの一部である。クランクケースの設計者が，シリンダ上段距離と考えるものが，実は他の設計者が担当する設計結果を足算して求められていることを示している。

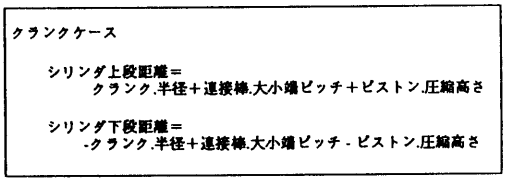

図4　変換ルールの例

また，このように，ある設計者の設計結果が，他の設計者の設計仕様になったりするので，一方の設計者の担当するオブジェクトの構造や値の変更を，それを参照する他の設計者へ各種メディアを用いて通知する必要がある。さらに自動的に変更を反映する場合もあり得る。変更というイベントの起動には，デモン(if-updated)が利用できるし，変更の通知にはLAN 上でのメール機能（マルチメディアを含む）が必要になる。

7. おわりに

以上でCAD/CAM/CAE やEOA といったエンジニアリング分野で必要となる技術要件を分析し，次に最近注目されているOODBの一般的特徴を述べ，そのうえでOODBを用いてどこまで要件が満足できるか，満足できない要件に対しては何をOODBに追加すべきかについて論じてきた。こうした検討に添ったかたちで，現在エンジニアリング向けのデータベースの設計を行っている。もちろん，この論文ですべてを尽くしたわけではない。重要であるがここでは述べなかった技術要件として，バージョン管理や設計トランザクションというものがある。これらの要件は，応用ごとに違う側面を持ち，一般的に論ずることは難しいが，今後の研究課題の1つとしたい。

参考文献

〔ATKI89〕 M. Atkinson et al., "The object-oriented database manifesto," in Proc. DOOD '89 Conf. Kyoto, Japan 1989.
〔BANE87〕 J. Banerjee et al., "Data model issues for object-oriented applications," ACM Trans. Office Information Syst. vol. 5, no. 1, Jan. 1987.
〔FISH87〕 D. H. Fishman et al., "Iris: an object-oriented database management system," ACM Trans. Office Information Syst. vol. 5, no. 1, Jan. 1987.
〔HEIM85〕 D. Heimbigner et al., "A federated architecture for information management," ACM Trans. Office Information Syst. vol. 3, no. 3, July 1987.
〔ISHI90〕 H. Ishikawa, "An object-oriented knowledge base approach to a next-generation of hypermedia system," in Proc. IEEE COMPCON Spring '90 Conf. San Francisco, CA. 1990.
〔LECL88〕 C. Lecluse et al., "O_2 an object-oriented data model," in Proc. ACM SIGMOD Conf. Chicago, IL. 1988.
〔LELE88〕 W. Leler, Constraint programming languages, Addison-Wesley, Reading, MA. 1988.