データベースシステム 100－7
（1994．10．14）

画像DB用アプリケーション開発言語の設計

大津浩二[†] 松田宜之[†] 金森吉成[†] 増永良文[††] 脇山俊一郎[†††]

[†]群馬大学 [††]図書館情報大学 [†††]仙台電波高専

本論文は、画像データベースのアプリケーションを容易に開発できるオブジェクト指向のスクリプト言語の設計について述べる。この言語を用いて、ユーザは計算機環境から独立した高水準なシナリオを記述することができる。言語は、画像処理オブジェクト定義、シナリオ構造記述、データベース問合せオブジェクト定義、シナリオ起動メッセージなどの機能を持っている。このスクリプト言語の特色は、検索された画像オブジェクトに種々な画像処理を適用することが簡単に表現できることである。さらに、著者等が提案している画像データベースアーキテクチャ上でスクリプトがどのように解釈、実行されるかを例を使って説明した。

A Design of Application Development Language for Image Databases

Kouji Ootsu [†] Takayuki Matsuda [†] Yoshinari Kanamori [†]
Yoshifumi Masunaga [††] Shunichiro Wakiyama [†††]

[†] Gunma University
[††] University of Library and Information Science
[†††] Sendai National College of Technology

This paper describes a design of object-oriented script language to ease application developments for image databases.User can make a high level scenario written in the script,which is independent of computing environments.The script has many functions: definition of image processing objects,description of scenario structure, definition of query objects, start message for scenario, and etc.There is a characteristic of the script: it is ease to write the scenario that users apply some kind of image processing functions to image objects retrieved from databases. Furthermore, we have explained how the script is interpreted and executed under an image database architecture which we have proposed.

1 はじめに

画像データベースのアプリケーションでは、一般に検索された画像データを画像処理関数で加工し、ディスプレイ上に表示する操作が大部分を占める。そのために、アプリケーション開発ではプログラミング言語、画像処理関数、データベース問合せ言語SQL、Xウィンドウシステムなど高度な計算機環境の知識が要求される。従って、ユーザがアプリケーションを開発することは容易ではない。言い換えれば、ユーザの意図した画像データベースの利用シナリオからプログラミング言語程度までへの変換に大きなギャップがあり、そのギャップを埋めるために膨大な知識が要求される。

そこで、著者等は画像データベースのアプリケーション開発を容易にするための画像データベースアーキテクチャ [2] について研究してきた。本研究では、この方針に沿って画像データベースの利用シナリオを高水準で表現・記述できるスクリプト言語の開発を目的としている。

ここでのスクリプト言語は、オブジェクト指向に基づき設計された。計算機環境、画像の可視化、画像処理関数の操作などシナリオの本質に関係しない部分がユーザに全く見えなくなっているので、シナリオを表現・記述することが容易になっている。

2 画像データベースアーキテクチャ

図2に示すように、著者等が提案している画像データベースアーキテクチャ [2] は、

(1) 画像オブジェクトと画像処理ライブラリ（メソッド群）を管理するオブジェクト指向データベース

(2) (1) のオブジェクト指向データベースのクライアントとして

データベースオブジェクトの問合せ処理を行なうQueryプロセッサ

画像処理を行ない、アプリケーションに画像処理オブジェクトを提供する画像処理サーバ

(3) アプリケーションプログラム

の3層型クライアント・サーバモデルを構成している。

OODBサーバ

従来の研究により画像処理には400個を越える非常に多くの処理関数 [3] が存在する。これらの関数群を分類し、クラスライブラリの形でオブジェクト指向データベース (Method DB) で管理する。

一方、画像オブジェクトの汎用性を確保するため、画像オブジェクトからメソッドを排除した、画像データのみからなる汎用画像オブジェクトをオブジェクト指向データベース (Image-Data DB) で管理する。

Queryプロセッサ

Queryプロセッサは、アプリケーションプログラムから送られてきた問合せ条件のOSQL文をODBMSに送る。それにより、ODBMSから検索結果に該当するオブジェクトのOIDが送られてくるので、それをアプリケーションプログラムに返す。

画像処理サーバ (Image Processing Server)

本アーキテクチャの中核となる部分で、画像処理オブジェクトの画像処理機能の実行と仮想画像オブジェクトのアプリケーションへの提供を行う。

仮想画像オブジェクトは、データベース上の汎用画像オブジェクトとアプリケーションに依存する画像処理オブジェクト（メソッド）を動的に結合して、アプリケーション層で1つの仮想的な画像オブジェクトに見えるようにしたものである。

画像処理においては、複数の画像処理オブジェクトを利用した複合処理が必要な場合、前の段階までの画像を利用するため、処理履歴を管理、保存する。これにより、途中結果からのやり直しや途中結果を利用しての画像処理実行などが可能となる。

3 画像DB用アプリケーション開発言語の設計

言語の設計方針として、直観的に理解しやすく、学習が容易な高水準なオブジェクト指向のスクリプト言語を採用する。

3.1 基本構文

メッセージは

```
オブジェクト名  –メソッド  パラメータ
                –メソッド  パラメータ
                ⋮
```

の形式で記述される。

パラメータ数やパラメータの解釈はメソッドにより異なる。

各メッセージは空白で区切られ、1つ以上のメソッドーパラメータ対から構成される。

{} (中括弧) は、空白を含んだ文字列を1つのメッセージと解釈する。

[ ] (大括弧) 内のメッセージが実行される際に、メッセージの実行結果との置換が行なわれる。

これらの括弧の使用が図4にある。

最初にオブジェクトを生成する場合には、

```
クラス名  生成オブジェクト名
          –メソッド  パラメータ
          –メソッド  パラメータ
          ⋮
```

の形式で記述する。

3.2 記述順序

シナリオ記述は以下の4構成となる。

1. 画像処理オブジェクト定義
   - 画像処理オブジェクト名
   - 画像オブジェクト名
2. 構造記述
   (a) シナリオ構造定義
   (b) シーン構造定義
   (c) 基本オブジェクト定義
   (d) グループオブジェクト定義
   (e) リンク構造記述
3. DB問合せオブジェクトの定義
4. シナリオ起動メッセージ

記述順序は上記の番号順となる。構造記述内の記述順序は (a),(b),···,(e) の順となる。

3.3 画像処理オブジェクト定義

画像処理オブジェクト名

種々な画像処理関数が複数個組み合わされた画像処理の場合には名前を定義しておき、この名前の指定で画像処理を実行できると便利である。そこで、一連の処理関数から成る複雑な画像処理を一つにまとめて、予め画像処理オブジェクト名 (メソッド名) として定義する。

画像オブジェクトは、画像処理メソッド名の指定により画像処理を実行する。$ 画像処理メソッド名と指定すると、実行時にメソッドの実行結果との置換が行なわれ、具体的な画素値または値などに変換される。

画像処理メソッド名の定義は、以下のような形式の記述とする。

```
define 画像処理メソッド名
        { 画像処理関数名 画像処理関数名 ···}
```

記述例 (1)

リサイズ、ディザの画像処理を連続して実行させたいときの例を以下に示す。

```
define  ip_0  {resize(50,100)  dither}
```

記述例 (2)

図1は、処理が複雑な画像処理の例である。これは顔画像の目の位置を抽出してその位置にマスクをかけ、人物を特定できなくする処理である。この例は、分岐した後、eye_position_detect と dither までを別個の画像処理として実行し、それぞれ画像処理メソッド名 ip_0 (dither まで) と ip_1 (eye_position_detect まで) と定義する。ip_0 までの結果は、処理された画像データであるのに対し、ip_1 の結果は、目の位置座標データに変換されている。2枝が合流する画像処理の masking は、第一パラメータに画像データを、第二パラメータに目の座標データを取る。記述の時、ip_0 と ip_1 の先頭に $ を付ける。これにより、画像と座標値に置換される。図1の記述例を以下に示す。

```
define  ip_0  {resize(50,100)  dither}
define  ip_1  {resize(50,100)  color_to_gray
               edge_detect  eye_position_detect}
define  ip_2  masking($ip_0,$ip_1)
```

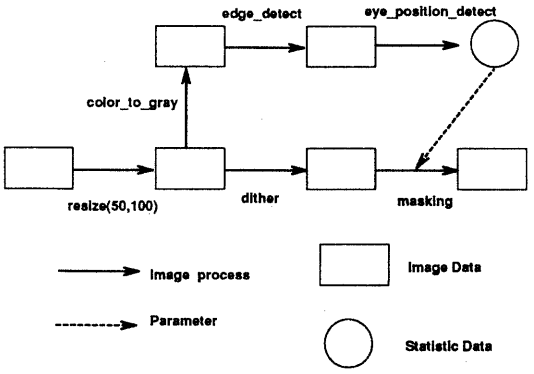

図 1: 画像処理の記述例 (2)

画像オブジェクト名

一般に画像データベースの検索により、画像オブジェクトの集合が返される。集合内の任意の画像オブジェクトについて、個々に別な処理を加えたい場合、個々の画像に名前 (ID) を付け、その ID で画像オブジェクトを識別する必要がある。そこで、画像オブジェクトを識別する ID を予め画像オブジェクト名として定義する。

シナリオ内で画像オブジェクトの識別に image_a,image_b,image_c を用いた場合、以下のように示す。

define {image_a image_b image_c}

3.4 構造記述

ここで記述または定義されるシナリオ、シーン、基本オブジェクト、グループオブジェクト、リンクなどは全てオブジェクトである。

シナリオ構造定義

シナリオはシーンの組み合わせで構成される。シナリオの記述は、以下のようにシーン名の列挙である。

scenario 生成シナリオ名 -component
{ シーン名 シーン名 ···}

シーン構造定義

シーンは基本オブジェクトの組み合わせにより構成され、物理的にはディスプレイ上の 1 ウィンドウに対応する。シーンの記述は、以下のように基本オブジェクトを列挙する。

scene シーン名 -component {
クラス 生成オブジェクト名 –メソッド
パラメータ –メソッド パラメータ ···
クラス 生成オブジェクト名 –メソッド
パラメータ –メソッド パラメータ ···
⋮
}

基本オブジェクト定義

シーンの構造定義に使用する基本オブジェクトを表 1に示す。オブジェクトを生成する際に、同一シーン内ではオブジェクト名の重複を許さないことにする。また、各基本オブジェクトには共通したメソッドや、固有なメソッドが存在する。

表 1: 基本オブジェクト

オブジェクト名	オブジェクトの内容
button	マウスのクリックを検出
canvas	図形表示
label	改行を含まない文字列表示
text	改行を含む文字列表示
listbox	改行を含む文字列を 行単位に表示
entry	キーボードからの文字入力
showimage	一枚の画像表示
showimagebox	画像の集合を表示

グループオブジェクト定義

同種な複数のオブジェクトに対し、同じメッセージを繰り返し発するのは非効率的である。そのため、これらのオブジェクトを 1 つのグループと見なし、グループに対してメッセージを送ることにより、グループ内全てのオブジェクトにメッセージが送られる。グループは、同一シーン上に存在する同種のオブジェクトから成る。グループ内オブジェクトに共通するメッセージのみを受け付ける。グループの記述は、以下のようにグループ内のオブジェクト名を列挙する。

group グループ名 -component
{ オブジェクト名 オブジェクト名 ···}

リンク構造記述

リンク (link) は、あるオブジェクト内であらかじめ定義されている条件が成立したら、指定したオブジェクトに対してメッセージを送るオブジェクトのことである。リンクは、条件を満たした時にメッセージを複数送ることができる。

リンクの条件には、

buttonpress　マウスボタン 1 が押された
buttonrelease　マウスボタン 1 が放された
press-return　リターンキーが押された

などがある。

リンクの記述は、以下のような形式とする。

link　リンク名
　　-from　条件が発生するオブジェクト名
　　-condition　成立する条件
　　-action　{ メッセージ　メッセージ　…}

アクションのパラメータの中でオブジェクト名を記述する場合、記述しているシーン以外のオブジェクトも扱えるように、オブジェクト名の指定は、シーン名. オブジェクト名とする。

3.5　データベース問合せオブジェクトの定義

画像データベースへの問合せを行なうオブジェクト (dbquery) を設ける。

データベース問合せオブジェクトに、OSQL 文（現在標準化されていない）が与えられるものとする。OSQL 文で $ だけのフィールドが存在する場合がある。このときは検索実行を指示するメッセージから、$ の実値が渡される。データベース問合せオブジェクトは、検索結果として、検索条件に該当するオブジェクトの OID 集合を得る。そして、oid メッセージが送られたら、所有している OID 集合を返す。

データベース問合せオブジェクトの定義は、以下のような形式の記述とする。

dbquery　オブジェクト名
　　-osql　検索条件の OSQL 文
　　-oid

3.6　シナリオ例

シナリオの例として、患者 (patient) とその顔写真 (faceimage) のオブジェクトが格納されている画像データベースを利用する場合を考える。患者オブジェクトは、属性として名前 (name)、症例番号 (case_no)、年齢 (age) を所有しており、顔写真オブジェクトは、属性として症例番号 (case_no)、画像データ (i_data) を所有しているとする。

図 3にシナリオの例を与える。シーン s_0 は検索条件（年齢）の入力を行ない、シーン s_1 は検索された全症例番号を表示して、画像は症例番号で選択できる。シーン s_2 は s_1 で選択された症例番号とそれの全ての顔写真を表示する。

我々の提案する 3 階層の画像データベースアーキテクチャでは、上位のアプリケーション層でスクリプトにより直接画像オブジェクトを操作できる。この画像オブジェクトは、中間層で作られる仮想画像オブジェクトである。問合せオブジェクトに画像処理メッセージが与えられたとき、問合せオブジェクトは、画像オブジェクトの OID と画像処理メソッド名とを Image Processing Server に送る。これにより、仮想画像オブジェクトが生成される。このメカニズムを以下に例を用いて詳しく説明する。

仮想画像オブジェクト生成

シナリオでは問合せオブジェクト set_faceimage と画像処理メソッド名 ip_0 を定義している。set_faceimage は、検索結果として該当する患者の顔写真オブジェクトの OID を所有している。

仮想画像オブジェクトは、図 2に示した 3 層間の関係により以下のように生成される。

(1) set_faceimage オブジェクトは、OSQL 文を DB Query Processer に送る。

(2) DB Query Processer は、(1) で送られてきた OSQL 文を ODBMS に送る。OSQL 文を受け取った ODBMS は、検索を実行し、該当する画像オブジェクトの OID を DB Query Processer に送る。

(3) DB Query Processer は、(2) で送られてきた OID を set_faceimage に送る。

(4) set_faceimage に、画像処理メソッド名 ip_0 を送る。

(5) set_faceimage は、(3) で送られた OID の集合と (4) で送られた画像処理メソッド名とを Image Processing Server に送る。こ

の時、set_faceimage は、(4) で送られた画像処理メソッドに置換を行ない、Image Processing Server に具体的な画像処理名を送る。

(6) Image Processing Server は、(5) で送られた OID の画像を ImageData DB から取り出す。

(7) Image Processing Server は、(5) で送られた画像処理のライブラリを Method DB から取り出し、これを (6) で Image DB から取り出した画像オブジェクトと結合する。この結び付けにより、指定した処理が施された仮想画像オブジェクトが Image Processing Server 内に生成される。

(8) Image Processing Server は、(7) で生成した仮想画像オブジェクトの OID の集合を set_faceimage に返す。

上に示した 3 階層間の関係により、set_faceimage は仮想画像オブジェクトの OID を所有する。画像を表示する場合、set_faceimage は、シーン s_2 のグループオブジェクト g_0 に所有している仮想画像オブジェクトの OID を渡す。仮想画像オブジェクトの OID が渡された g_0 は、この OID を用いてグループ内の各画像オブジェクトに画像を表示するメッセージを発する。

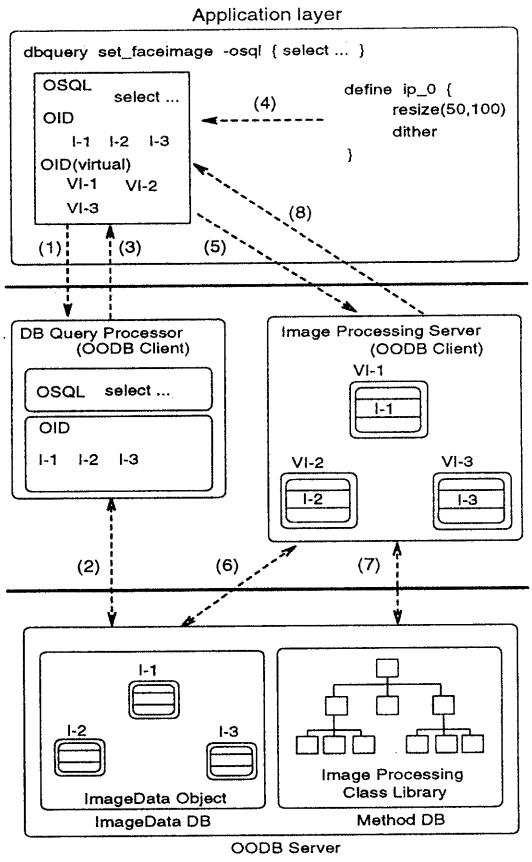

図 2: 仮想画像オブジェクト生成時の 3 層間の関係 [2]

3.7 対話処理による複合画像処理

シナリオ作成時に画像処理関数を指定できないときは、対話的に適当な関数を決めることが必要となる。

このような対話処理システムは、シナリオ作成支援のために必須である。現在、図 2の画像データベースアーキテクチャ上で稼働する対話処理システムを開発している。

対話処理システムは、処理結果を履歴として管理することができるので、対話処理を終了した後、使用した一連の処理関数、処理手順をシナリオ内に書き込むことができる。

3.8 オブジェクトの識別

シナリオ記述時に、画像オブジェクト名の定義により、各画像に任意の ID を付けて識別することができる。シナリオの起動時に、仮想画像オブジェクトは、問合せオブジェクトの検索結果から生成される集合として与えられる。

前者と後者は自動的に対応づけられる。しかし、シナリオを見ながら特定の仮想画像オブジェクトに更なる処理をする場合や別のシーンで再度利用したい時には、それを対話的に処理し、保存する必要がある。画像処理サーバがこのようなオブジェクトの管理機能を持っている。対話処理では、ユーザは、ディスプレイ上（表 1の showimagebox を使用）でブラウジングし、記述時に用いた ID と実際に使用する仮想オブジェクトの実体とを対話的に結合する。これにより、それ以後のシナリオで必要とする仮想画像オブ

ジェクトをIDで指定して利用できる。

3.9 関連研究

現在、マルチメディア・ハイパーメディアの論理構造を記述する国際標準言語HyTime[1][4][5]がある。また、HyTime仕様に沿ったシステム開発も行なわれた[6]。このHyTimeでは各メディアのオブジェクト間に存在する上下関係、順序関係、参照関係などを記述できる。

今後は、本研究でのスクリプトとHyTimeの規格との整合性について検討しなければならない。

本研究のスクリプト言語は、HyTimeよりも上位レベルにあるマルチメディアのアプリケーション開発言語を意図したものである。現在このようなスクリプト言語に相当するものに国際標準言語SMSL[7]が検討されている。

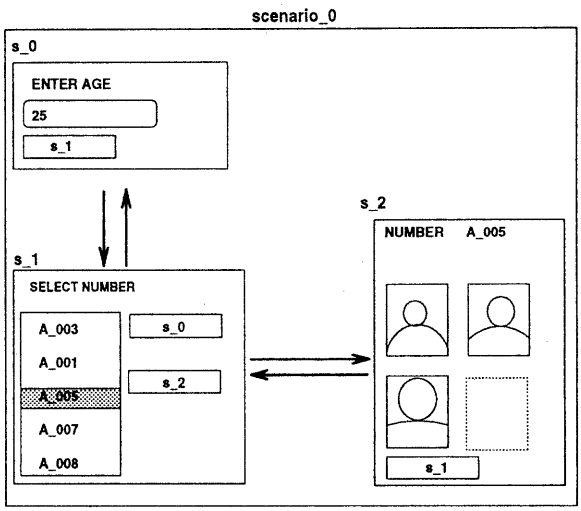

図 3: シナリオの例

4 おわりに

本稿では画像DB用アプリケーション開発のスクリプト言語の設計について述べた。スクリプト言語と提案している画像データベースアーキテクチャとの関係を説明した。

ここで提案したスクリプト言語は、主に画像処理の視点から画像データベースを利用したアプリケーション開発を容易にすることを目的としている。種々な画像処理関数の適用が必須となる分野、例えば医療画像データベースなどではこのような観点からのアプリケーション開発言語は重要であると考える。

本研究のスクリプト言語を一般的なマルチメディアデータのアプリケーション開発言語へと拡張することが考えられる。この場合には、基本オブジェクトの追加、オブジェクト定義構文の追加で対応できるかについて、さらに詳細な検討が必要である。

謝辞 本稿をまとめるにあたり、文献やSMSL言語について御教示戴いたキャノン(株)の内藤広志氏に感謝いたします。

参考文献

[1] ISO/IEC 10744. Information technology — Hypermedia/Time-based Structuring Language(HyTime),1992

[2] 脇山、川島、金森、増永: 分散処理環境における汎用画像オブジェクトの構成, 情報処理学会データベースシステム研究会,99(1994)

[3] 画像処理サブルーチン・パッケージSPIDER USER'S MANUAL, 電子技術総合研究所 (1980)

[4] 今郷詔: HyTime: 文書の拡張としてのハイパーメディア記述言語, 情報処理学会 データベースシステム,91-6(1992)

[5] 小町祐史: HyTime (ハイパメディア及び時間依存情報のための構造化言語) の概要, 情報処理学会 アドバンスト・データベース・シンポジウム'93 講習会資料 (1993)

[6] 内藤広志、佐藤衛、松山洋一、山下真司、柵木孝一: アクティブ機能を持つハイパーメディアマニュアル Hydra-II の試作, 情報処理学会 アドバンスト・データベース・シンポジウム'93(1993)

[7] 小町祐史: マルチメディア／ハイパーメディア情報交換の標準化動向, 情報処理学会誌 pp632-641,July(1994)

```
# Image-Processing definition
define ip_0 {resize(50,100)   dither}
# scenario  scenario_0  definition
scenario scenario_0  -component  { s_0  s_1  s_2 }
# scene  s_0  definition
scene s_0 -component {
    label  la_0    -x 10   -y 10   -string     {ENTER AGE}
    entry  en_0    -x 10   -y 40
    button bu_0    -x 10   -y 70   -string     s_1
    link  link_0   -from   en_0    -condition  press-return
                   -action   {set_patient  -query  [s_0.en_0  -string]}
    link  link_1   -from   bu_0    -condition  buttonpress
                   -action   {{s_1  -map}  {s_0  -unmap}
                              {s_1.li_1  -string  [set_patient  -oid]}}
}
# scene  s_1  definition
scene s_1 -component {
    label   la_1    -x 10    -y 10   -string   {SELECT NUMBER}
    button  bu_10   -x 100   -y 10   -string   s_0
    button  bu_11   -x 100   -y 100  -string   s_2
    listbox li_1    -x 10    -y 100
    link    link_2  -from   bu_10  -condition  buttonpress
                    -action   {{s_0  -map}   {s_1  -unmap}}
    link    link_3  -from   li_1   -condition  buttonpress
                    -action   {{set_faceimage  -query  [s_1.li_1  -select]}
                              {s_2.la_21  -string  [s_1.li_1 -select]}}
    link    link_4  -from    bu_11  -condition  buttonpress
                    -action   {{s_2  -map}  {s_1  -unmap}
                              {set_faceimage  -exec  $ip}
                              {g_0  -image_id   [set_faceimage -oid]}}
}
# scene  s_2  definition
scene  s_2  -component  {
     label   la_20  -x  10  -y   10  -string NUMBER
     label   la_21  -x 100  -y   10
     showimage   im_20  -x  10  -y  100
     showimage   im_21  -x  10  -y  200
     showimage   im_22  -x 100  -y  100
     showimage   im_23  -x 100  -y  200
     group   g_0    -component    {im_20  im_21  im_22  im_23}
     button  bu_2   -x  10  -y  300   -string  s_1
     link    link_5   -from  bu_2   -condition    buttonpress
                      -action    {{s_1  -map}  {s_2  -unmap}}
}
# query-object  definition
dbquery   set_patient    -osql   {select patient   from patient
                                       where age < $}
dbquery   set_faceimage  -osql   {select faceimage from faceimage
                                       where case_no =  $}
# main
scenario_0  -start  s_0
```

図 4: 図 3 の記述例