ソフトウェア工学 58－18
（１９８８．２．５）

ＨＩ－ＶＩＳＵＡＬにおける アイコン重ね合わせ機能

岩田誠司*、 田原義信*、 平川正人**、 田中稔**、 市川忠男**

*広島大学大学院工学研究科 **広島大学工学部

筆者らは既に、アイコンと呼ばれる絵シンボルを用いて視覚的にプログラミングを行なうことができるアイコニック・プログラミング環境HI-VISUALを提案している。本論文では、ユーザとの対話性・親和性をより高める目的で開発した、アイコンの重ね合せ機能について述べる。またHI-VISUALにおけるアイコンとアイコンシステムの定義、ならびにアイコンとアイコンプログラムの管理・解釈方法についても述べる。さらにHI-VISUALのアプリケーションのひとつとして事務処理環境を取り上げ、そのインプリメンテーションについて説明する。

Icon Overlapping Facilities in HI-VISUAL

Seiji Iwata, Yoshinobu Tahara, Masahito Hirakawa, Minoru Tanaka, and Tadao Ichikawa

Information Systems Lab., Faculty of Engineering, Hiroshima University,
Shitami, Saijo-cho, Higashi-Hiroshima, 724 Japan

We already have proposed an iconic programming environment named HI-VISUAL. This paper presents an extension of HI-VISUAL to allow icon overlapping. Definitions and icon systems are first given. An interpretation mechanism of icon programs is clearly described. The implementation of HI-VISUAL in an office environment is also presented.

1．はじめに

計算機の普及に伴って計算機の利用者はますます増加および多様化の傾向にあり、計算機の操作やプログラミングを専門としない人々が計算機を利用する機会も増えている。そのため、計算機非専門家にも使いやすい計算機環境の構築が望まれている。このための１つのアプローチとして、ユーザインタフェースに視覚情報を利用することによってユーザとの対話性・親和性を高めようとする研究が近年さかんになってきている[1]~[8]。また視覚媒体としてのアイコンやイメージに関する理論的研究も行なわれている[9]~[13]。

我々は既に、アイコンと呼ばれる絵シンボルを用いて視覚的にプログラミングを行なうことができるアイコニック・プログラミング環境HI-VISUALの研究・開発を行なっている[14]~[20]。HI-VISUALにおいては、データやファンクション等がアイコンとして視覚化されている。プログラミングは、画面上でアイコンを選択・配置し、データの流れに沿ってアイコン間の接続関係を規定していくことにより行なわれる。この過程で、プログラムの実行結果が表示されるようになっており、ユーザはこれを見ながらシステムと対話的にプログラムの作成を行なうことができる。このようなプログラミング方式は従来のテキストベースの方式と比べて、計算機非専門家にもなじみやすい。

本論文では、ユーザとの対話性・親和性をより高めるための試みとして、HI-VISUALに新たに導入したアイコンの重ね合わせ機能について述べる。またこの機能拡張と併せて、従来のHI-VISUALで定義されていた７つのアイコンタイプを取り払っている。すなわち、これまでのようにデータやファンクションなどを別々に取り扱うのではなく、オブジェクトと呼ぶただ１つのタイプで統一的に取り扱うことによって、システムの理解容易性を高めている。

２.では、重ね合わせ機能導入の目的と利点、ならびにその機能を用いたアイコニック・プログラミングについて述べる。３.でアイコンの定義ならびにアイコンシステムの定義を与え、アイコンとアイコンプログラムの管理ならびに解釈メカニズムについて４.で述べる。最後に５.では、システムのアプリケーションのひとつとして事務処理環境を取り上げ、そこでのインプリメンテーションについて説明する。

2．アイコニックプログラミング

2．1　アイコンの重ね合わせ操作

従来のHI-VISUALにおいては、アイコンはシステムで取り扱われるデータ、ファンクション、プログラム等を表わしている。プログラムの記述は、データの流れに沿って、画面上でアイコンを結びつけていくことによって行なわれる。ここで、アイコン間の接続関係を規定するにあたっては、接続しようとするアイコンを画面上に配置し、その後にそれらのアイコンを選択して"接続"操作を指示しなければならなかった。

これに対して、ユーザがより直感的にプログラミングを行なえるようにアイコンの重ね合わせ操作を導入する。この操作は基本的には、重ね合わせられたアイコンの片方がもう一方のアイコンに作用すると理解する。

さらに、このような機能拡張と併せて、従来のHI-VISUALで定義されていた７つのアイコンタイプ（データアイコン、データクラスアイコン、プリミティブアイコン、プログラムアイコン、コントロールアイコン、パネルアイコン、コマンドアイコン）[16]をなくし、すべてオブジェクトとして統一的に取り扱う。すなわち、データやファンクションなどを別々に取り扱うのではなく、現実に存在する物（オブジェクト）に注目し、データやファンクションはオブジェクトの持っている性質として暗示的に表現する。オブジェクトとしてすべてを取り扱うことによって、ファンクションをアイコンとして明示的に表現することの困難さ[9]，[11]，[13]を取り除くことができる。例えば、ファイルというオブジェクトは文字列データを表わし、ごみ箱というオブジェクトは"削除する"というファンクションを表わすために用いられる。

また、データとファンクションの両方の性質を併せ持つアイコンを定義することもできる。例えば、"鉛筆"というオブジェクトは実存する物（データ）としての性質の他に、"書く"というファンクションとしての性質も併せ持つ。

このようなアイコンを用いて、例えばごみ箱を表すアイコンにファイルを表すアイコンを重ね合わせることによってプログラムを記述する。この場合には、重ね合わされたファイルが削除される。ところでデータとファンクションとしての性質を併せ持つアイコンの場合においては、重ね合わされるアイコンの組合せによって起動されるファンクションが定まる。すなわち、例えば鉛筆アイコンと文章アイコンとを重ね合わせた時には鉛筆は文章を書くためのファンクションとして機能し、鉛筆アイコンとごみ箱アイコンとを重ね合わせた時には鉛筆は物（データ）と見なされて削除される。

なお、１つのアイコンに複数の異なったファンクションを定義することを許している。これはアイコンの表現能力を高めるが、その一方、あるアイコンの重ね合わせに対して複数の異なった意味解釈が生じることがある。このためHI-VISUALでは、ユーザはシステムと対話的にやりとりを行なうことによって、その候補集合の中から１つを選択する。

２．２．　プログラムの記述

HI-VISUALの画面は図１に示すように、複数のウィンドウから構成される。それぞれのウィンドウは、プログラミングを行なうためのプログラミングエリアと、プログラミングに利用できるアイコンを表示しているメニュエリアの２つから構成されている。また、システムに対する操作（例えばプログラムの登録や終了）はポップアップメニュを介して指示するようになっている。

事務処理環境におけるアイコンプログラムの例を図２に示す。このプログラムは、(1)売上げデータを売り上げ帳に記入し、(2)担当者と金額、ならびに日付と金額の項目に基づいて表を作成し、(3)それぞれの表からグラフを求め、(4)最後に両方のグラフをフォルダに入れるという処理を表している。ここで、電卓アイコンはユーザの指定した項目について売上額の合計を計算するという機能を表わしている。

このようなプログラムを記述するにあたっては、まずユーザはメニュエリアから売上帳と売上げデータを表すアイコンを選択する。プログラミングエリア上でそれらのアイコンを重ね合わせる操作を行なうとシステムはそれらのアイコンの組み合わせの下で実行可能な処理をユーザに提示する（図３）。画面上のＮｅｘｔを指示すると、システムは実行可能な次の処理を提示する。ユーザはシステムが提示した処理の中から自分が希望するものを指示する。実行結果はただちに表示されるようになっており、ユーザはその実行結果を確認しながらプログラミングを進める。

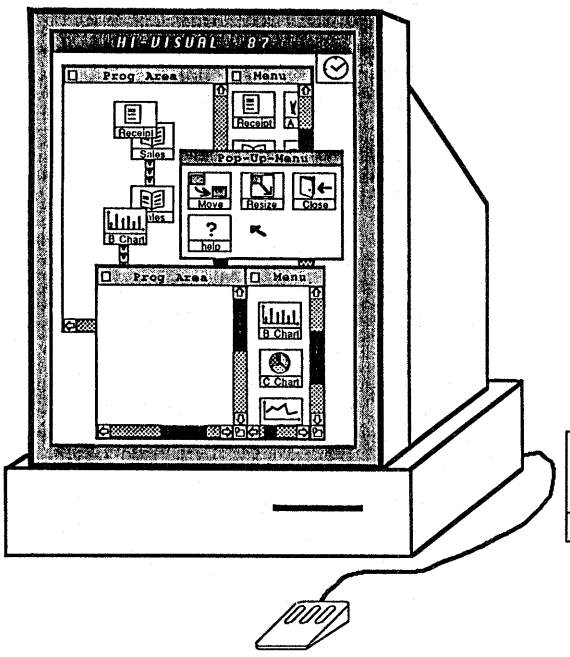

図１．画面構成

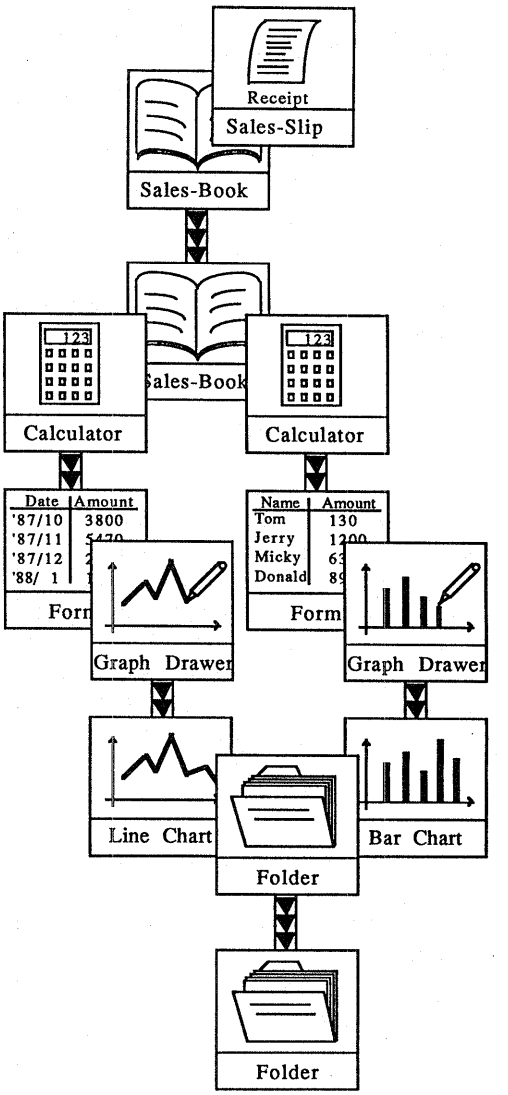

図２．プログラム例

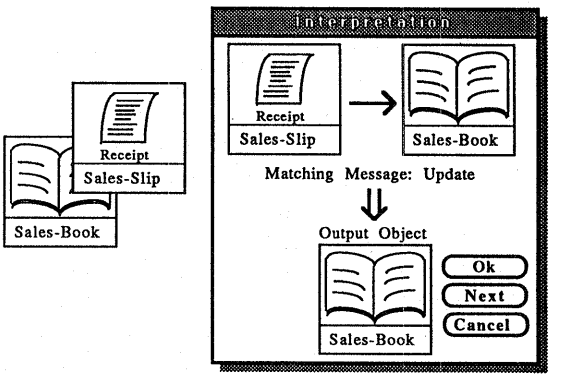

図３．プログランミング操作

作成したプログラムは登録してさらに大きなプログラムを作成するために使用できる。

HI-VISUALにおけるプログラミングの特長は以下のようにまとめられる。

1) 現実世界に存在している物をアイコン化しているので、それが表しているデータや機能をユーザが理解しやすい。
2) 実行結果を見ながら対話的にプログラミングが行なえる。
3) データの流れを視覚化しているために処理の流れが理解しやすい。

3. アイコニック・システム

3.1 アイコンの定義

アイコンをユーザとシステムとの対話手段として用いるにあたっては、それを単にオブジェクトの視覚表現としてではなく、オブジェクトの意味の記述もいっしょに結びつけて管理する必要がある。このことからHI-VISUALにおいてアイコンは、それが表わしているオブジェクトの意味を記述しているインターナル部(Internal Part)と、オブジェクトの意味の視覚表現を記述しているエクスターナル部(External Part)から成る(表1)。

表1 アイコンの定義

```
ICON := {Internal Part, External Part}

Internal Part := {Substance, Concept, Message}
External Part := {Image, Label}

Message := {transmittable message, acceptable
            message}
```

インターナル部はサブスタンス(Substance)、コンセプト(Concept)、ならびにメッセージ(Message)の3つの属性から構成されている。サブスタンスはアイコンが表わしているオブジェクトの実体を示している。コンセプトはオブジェクトの概念的名称（例えば日付、複写機）であり、ユーザ操作のナビゲーションやアイコンの分類・検索に利用される。メッセージは送信可能メッセージ(transmittable message)と受理可能メッセージ(acceptable message)に分けられる。送信可能メッセージは重ね合わせられた相手側のアイコンに対して送ることができるメッセージ、すなわち定義するアイコンの持っている機能を表している。これとは逆に、受理可能メッセージは相手側のアイコンから受け取ることのできるメッセージを表わしている。重ね合わされたアイコンの意味解釈は、それぞれのアイコンの受理可能メッセージと送信可能メッセージとの間でマッチングをとることによって行なわれる。

エクスターナル部はイメージ(Image)とラベル(Label)という2つの属性から構成されている。イメージはそのアイコンが表わしているオブジェクトを象徴する絵シンボルを示している。ラベルはアイコンの名前を表わしている。エクスターナル部は画面上に表示される。

ところでアプリケーションによってはデータとファンクションを明確に分けたい場合もある。このような場合には、アイコンのメッセージ部の受理可能メッセージをＮｕｌｌにすればそのアイコンはファンクションとして働き、逆に送信可能メッセージをＮｕｌｌにすればデータとして働くことになる。したがって表1に示したアイコンの定義はあらゆるアプリケーションにおいて有効である。

3.2 アイコンシステムの定義

HI-VISUALは特定のアプリケーションに依存しないプログラミング環境である。従って、個々のアプリケーションごとにそこで必要なアイコンを定義することによって、そのアプリケーション用の環境を構築していくことになる。個々のアプリケーションに応じたアイコンシステムは、そこで用いられるアイコンの集合と操作解釈ルールの集合との組として定義される（表2）。

表2 アイコンシステムの定義

```
ICON SYSTEM := {Icon, Rule}

Icon := {Application Dependent Icon,
         Application Independent Icon}
Rule := {User-Interface Definition Rule,
         Overlapping Rule, Priority Rule,
         Exception Handling Rule}
```

アイコンの集合はアプリケーションに固有のアイコン(Application Dependent Icon)とあらゆるアプリケーションに共通に用いられるアイコン(Application Independent Icon)との2つから成っている。アプリケーションに依存しないアイコンの例としては、アイコンの削除、アイコンプログラムの展開を行なうものや、アイコンイメージ作成ツールを含むプログラム支援ツールなどがある。

アイコンシステムに記述されるルールとしては次のようなものがあり、ＩＦ－ＴＨＥＮ形式で記述される。

1．ユーザインタフェースの定義(User-Interface Definition Rule)
　マウスボタンが表わす機能、プログラミングエリアやメニュエリアの配置などをユーザの好みにあうように設定する。

2．重ね合わせ操作の解釈(Overlapping Rule)

アイコンの重ね合わせ方に応じて種々の解釈を与える。すなわちアイコンを重ねる順番、あるいは重ねる位置（例えば上、下、左、右の方向）に対応して、それぞれ異なった処理を割り当てることができる。

3．データ／ファンクションの優先順位(Priority' Rule)

アイコンの重ね合わせが行なわれた時に、そのアイコンがデータあるいはファンクションのどちらとして働く可能性が高いかを記述する。このルールを用いることによって、重ね合わせの解釈を効率的に行なうことができる。

4．例外処理(Exception Handling Rule)

重ね合わせられたアイコンの間で一致するメッセージがなかった場合の処理を記述する。

4．システム管理

4．1　ディスクリプタ

HI-VISUALにおいてアイコンならびにアイコンプログラムはディスクリプタを用いて管理されている。アイコンはオブジェクト指向の概念に基づいて階層的に管理されるようになっており、その管理のためにオブジェクトクラスディスクリプタ（ＯＤ）とアイコンディスクリプタ（ＩＤ）の2つを用意している。またユーザが作成したアイコンプログラムは、ジョイントディスクリプタ（ＪＤ）を用いて管理・実行される。

オブジェクトクラスはオブジェクト指向モデルにおけるクラスの概念に相当するものである。図4に示すように、ＯＤにはクラスの階層関係に関する情報と、そのクラスに属するアイコンが受理／送信できるメッセージに関する情報が記述される。階層構造にしたがった情報の継承も行なわれる。アイコンの重ね合わせ操作により実行される機能は、このＯＤに記述されている情報に基づいて決定される。

ＩＤは個々のアイコンに関する情報を管理するものである。図5に示すように、ＩＤにはアイコンの識別子、アイコンが属するクラスの名前（識別子）、アイコンの名前、大きさ、座標、アイコンイメージ、ならびにそのアイコンが表しているオブジェクトの実体に関する情報などが記述される。

アイコンプログラムの実行はジョイントディスクリプタを参照しながら行なわれる。ＪＤには重ね合わされたアイコンの組、それらのアイコンの組合せにおいて選ばれたメッセージ、ならびにそのメッセージの実行結果のアイコンが属するオブジェクトクラスに関する情報が記述されるようになっており、プログラム中のすべてのアイコンの接続関係はひとつのＪＤにまとめて管理される（図6）。

Object_Class Identifier
Class Name
Superclass id
Subclass id
Constraint
Class Values
Concept
Def. Instance Values
Acceptable Messages
Message Name
Concept
Pointer to Executable Module
Output Object_Class id
Transmittable Messages
Message Name
Concept
Constraint

図4．オブジェクトクラスディスクリプタ（ＯＤ）

Icon Identifier
Object_Class Identifier
Icon Name
Size
Location
Image_File_Pointer
Help_Message Pointer
Status
Instance Values

図5．アイコンディスクリプタ（ＩＤ）

Joint Identifier
Line_id
Input Icons
Executed Message
Output Icon
Post Line_id
⋮
Line_id
Input Icons
Executed Message
Output Icon
Post Line_id

図6．ジョイントディスクリプタ（ＪＤ）

4．2　解釈・実行メカニズム

重ね合わせの解釈は、ＯＤに記述されている受理可能メッセージと送信可能メッセージに関する情報を基に行なわれる。以下にその概略を説明する。

Step 1. 重ね合わせられたアイコンについて、ある一方をデータとして働くオブジェクト、他方をファンクションとして働くオブジェクトとみなす。このとき、システムに優先順位ルールあるいは重ね合わせ操作の解釈ルールが記述されていればそれに従うものとする。そうでない場合にはシステムが暫定的に重ね合わせたアイコンをファンクションとみなす。

Step 2. データと仮定したアイコンの受理可能メッセージと、ファンクションと仮定したアイコンの送信可能メッセージとの間でマッチングを行なう。

Step 3. もし一致するメッセージがあればそれを実行し、結果をポップアップメニュとして表示する(図3)。ユーザの希望する機能と異なる場合(ユーザがＮｅｘｔを指示)には、Step 2.に戻って一致する次のメッセージを探す。

Step 4. 現時点のデータとファンクションに関する仮定の下では一致するメッセージがない場合には、その関係を逆にしてStep 2.に戻る。

Step 5. 最終的に一致するメッセージがなければ、システムに記述されているルールに基づいて例外処理を行なう。

次に具体例として、Report FolderアイコンとSales Reportアイコンが重ね合わされた場合について考える。このときのシステム内部の様子を図7に示す。ここで、Report FolderアイコンはFolderというオブジェクトクラスに、Sales ReportアイコンはTyping Paperというオブジェクトクラスにそれぞれ属していると仮定する。なお、Typing Paperクラスの上位クラスとしてLetter Formクラス、またその上位クラスとしてFormクラスが定義されているものとする。

重ね合わせの解釈は以下のように行なわれる。

1. Formクラスに属するアイコンとFolderクラスに属するアイコンとが重ね合わされた場合、Formクラスのアイコンはファンクションとして、Folderクラスのアイコンはデータとして働くことが多いことを表すルールが書かれている。そのため、Sales Reportアイコンをファンクション、Report Folderアイコンをデータと仮定する。
2. Report Folderアイコンの属するFolderクラスの受理可能メッセージとSales Reportアイコンの属するTyping Paperクラスの送信可能メッセージとの間でマッチングを行なう。
3. 一致するメッセージがないのでクラスの階層構造に従って上位クラスのメッセージを探索する。Folderクラスの受理可能メッセージとTyping Paperの上位クラスであるFormクラスの送信可能メッセージとの間でadd_contentというメッセージが一致するので、そのメッセージに対応する機能（レポートをフォルダに入れる）を実行する。

つまり、２つのアイコン間の局所的な解釈はアイコンのメッセージ部に記述されている情報に基づいて行い、大域的な解釈はシステムに記述されているルールに基づいて行なう。

５．応用例

現在、HI-VISUALのアプリケーションのひとつとして事務処理環境を取り上げ、実際にシステムを開発中である。但し、３.で述べたように、応用に依存したアイコンのセットを変更するだけでシステムは他の応用にも適用することができる。

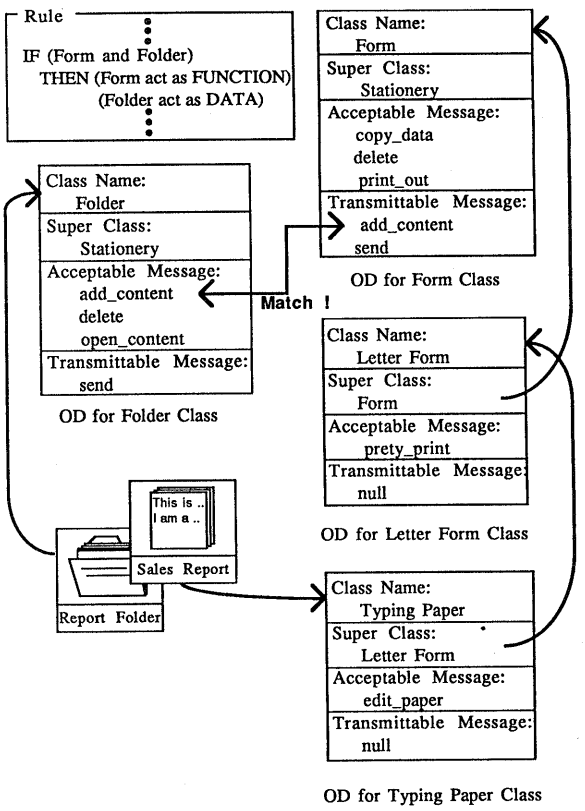

図７．解釈メカニズム

図８に事務処理環境におけるオブジェクトクラスの階層構造を示す。この階層構造において、Officeクラスより下位のクラスが事務処理環境用のものであり、それ以外のクラスはすべてのアプリケーションに共通する部分である。

Officeクラスのサブクラスとして、"文具(Stationery)"、"備品(Equipment)"、"秘書(Secretary)"の３つのクラスを用意している。文具クラスのサブクラスには用紙（手紙、図表、売上げ帳）、フォルダ、ならびにペンといったオブジェクトを表わすクラスがある。備品クラスのサブクラスとしては、キャビネット、コピーマシン、ゴミ箱などといったオブジェクトを表わすクラスがある。

秘書クラスはユーザ定義の処理を登録するためのものであり、ユーザが作成したアイコンプログラムはこの秘書クラスの送信可能メッセージとして登録される。秘書アイコンは複数の処理を管理するので、各処理ごとに個別のアイコンを割当てる場合に比べて、アイコンイメージを作成する手間を省くことができる。また、このクラスのサブクラスとして"文書処理専用の秘書"や"データ処理専用の秘書"を表すクラスを定義することによって、ユーザは自分の好みにあった環境を構築することも可能である。

６．おわりに

本論文では、HI-VISUALにおけるアイコンの重ね合わせ機能について述べた。また、アイコンならびにアイコンシステムの定義を与え、アイコンとアイコンプログラムの管理・解釈方法について述べた。さらにHI-VISUALのアプリケーションのひとつとして事務処理環境を取り上げ、そのインプリメンテーションについても説明した。

また現実世界に存在する物（オブジェクト）に注目して、それをアイコンとして定義した。アイコンをオブジェクトというひとつの型で取り扱い、それらのアイコンを画面上で重ね合わせることによってプログラミングを行うことができるようになっており、ユーザとの対話性・親和性に優れている。

システムはＮＥＷＳワークステーション上にＣ言語を用いて構築している。ＯＳとしてＵＮＩＸ、ウィンドウシステムとしてＸ－Ｗｉｎｄｏｗを用いている。現在のところシステム部分が約5000行、データ部分が約2000行である。

今後の課題としては、

1) 知識ベースを用いた高度ナビゲーション機能の提供
2) アイコンイメージの生成支援ツールの作成

などがあげられる。

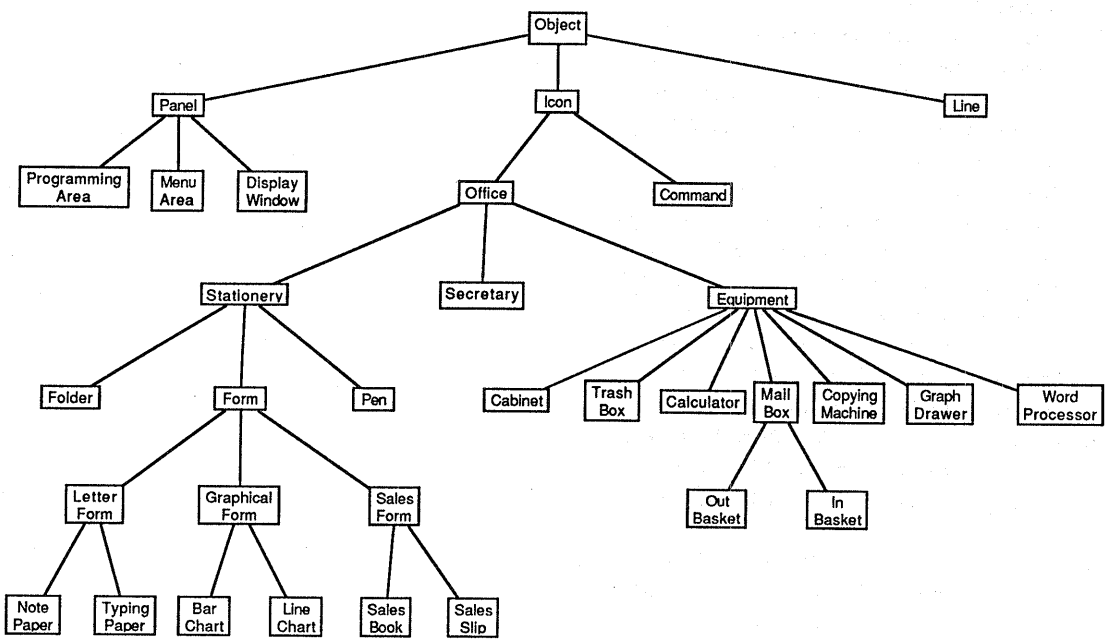

図８．オブジェクトクラスの階層構造

謝辞

本研究を進めるにあたり、システムのインプリメントにご協力頂いた情報システム研究室の山本武氏、吉見信氏に深く感謝する。

参考文献

[1] S. K. Chang, T. Ichikawa, and P. A. Ligomenides (eds.), Visual Languages, Plenum Press, New York, 1986.

[2] S. K. Chang, "Visual Languages: A Tutorial and Survey," IEEE Software, Vol.4, No.1, pp.29-39, 1987.

[3] B. A. Myers, "Visual Programming, Programming by Example, and Program Visualization: A Taxonomy," Proc., Conference on Human Factors in Computing Systems, pp.59-66, April 1986.

[4] R. B. Grafton and T. Ichikawa (eds.) "Visual Programming," Special Issue of IEEE Computer, Vol. 18, No. 8, 1985.

[5] 平川、田中："視覚言語の動向"、電子通信学会誌、Vol. 69, No. 12, pp.1256-1259, 昭61.

[6] Proceedings of IEEE Workshop on Visual Languages, December 1984.

[7] Proceedings of IEEE Workshop on Visual Languages, June 1986.

[8] Proceedings of IEEE Workshop on Visual Languages, August 1987.

[9] K. N. Lodding, "Iconic Interfacing," IEEE Computer Graphics and Applications, Vol. 3, No. 2, pp.11-20, 1983.

[10] S. K. Chang, "Icon Semantics - A Formal Approach to Icon System Design," International Journal of Pattern Recognition and Artificial Inteligence, Vol. 1, No. 1, pp.103-120, 1987.

[11] K. Matsumura and S. Tayama, "Visual Man-Machine Interface for Program Design and Production," Proc., IEEE Workshop on Visual Languages, pp.71-80, Dallas, June 1986.

[12] S. K. Chang, "Icon Purity - Toward A Formal Theory of Icons," Proc., IEEE Workshop on Visual Languages, pp.3-16, Linkoping, August 1987.

[13] S. L. Tanimoto, "Visual Representation in the Game of Adumbration," ibid, pp.17-28, 1987.

[14] N. Monden, Y. Yoshino, M. Hirakawa, M. Tanaka, and T. Ichikawa, "HI-VISUAL: A Language Supportig Visual Interaction in Programming," Proc., IEEE Workshop on Visual Languages, pp.199-205, Hiroshima, December 1984.

[15] I. Yoshimoto, N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa, "Interactive Iconic Programming Facility in HI-VISUAL," Proc., IEEE Workshop on Visual Languages, pp.34-41, Dallas, June 1986.

[16] M. Hirakawa, S.Iwata, I. Yoshimoto, M. Tanaka, and T. Ichikawa, "HI-VISUAL Iconic Programming," Proc., IEEE Workshop on Visual Languages, pp.305-314, Linkoping, August 1987.

[17] 岩田、吉本、平川、田中、市川、"アイコンプログラムの作成・実行環境の開発"、情報処理学会第33回全国大会、pp.683-684、昭和61-10.

[18] 吉本、岩田、平川、田中、市川、"アイコニック・プログラミング環境HI-VISUAL"、情報処理学会、ソフトウェア工学研究会資料52-1、昭和62-2.

[19] 田原、岩田、平川、田中、市川、"HI-VISUALにおけるシステム統合化機能"、電気四学会中国支部連合大会、p.225、昭和62-10.

[20] 岩田、田原、平川、田中、市川、"アイコンの重ね合せを許したアイコニックプログラミング"、同上、 p.226. 昭和62-10.