ソフトウェア工学 77−15
（１９９１．２．８）

アイコニックプログラミングにおける ユーザ操作の解釈

西村　幸浩*、加登　基二*、平川　正人**、田中　稔**、市川　忠男**
*広島大学大学院　**広島大学工学部

計算機についての専門知識をもたないユーザにも使いやすいようなプログラミング環境の構築を目的に、視覚情報の利用という観点に立ってアイコニックプログラミング環境ＨＩ－ＶＩＳＵＡＬを開発した。ＨＩ－ＶＩＳＵＡＬでは現実に存在するものをアイコンで表現し、アイコンの重ね合わせ操作によりプログラミングを行なうことができる。本稿では特に、ＨＩ－ＶＩＳＵＡＬにおける重ね合わせ操作について考察を加えるとともに、ユーザの行なった画面操作から適切なシステム動作が行なえるようにするユーザ操作解釈機能について述べる。

Interpretation of Icon Handling in Iconic Programming

Yukihiro NISHIMURA, Motoji KADO, Masahito HIRAKAWA,
Minoru TANAKA, and Tadao ICHIKAWA

Faculty of Engineering, Hiroshima University
Kagamiyama 1-4-1, Saijo-cho, Higashi-Hiroshima 724, Japan

In order to construct a programming environment that is easy to use for computer non-professionals, we proposed a system which makes use of visual information. The system named HI-VISUAL uses icons as visual information. In HI-VISUAL, icons represent objects actually existing in a real world and programs are created by overlapping icons. In this paper, we describe facilities for overlapping icons and a mechanism for interpreting overlapped icons in HI-VISUAL.

1．はじめに

計算機の能力の飛躍的な向上と低価格化によって利用者層は拡大、多様化しており、計算機の操作に不慣れな利用者が計算機を使用する機会が増加してきている。このような流れの中で、計算機の非専門家にも使いやすく理解しやすいシステムの構築が要求されている。計算機に不慣れな利用者が計算機を利用しようとする場合に問題となるのは、計算機に関する専門知識の必要性である。計算機を使用するにあたっては、扱い方からコマンド名や構文、ツールの使い方などといった非常に多くの知識が必要とされる。そのため、前述のような計算機と人間との間のセマンティックギャップを計算機の側で埋めていこうとする様々な試みが行なわれている。なかでも、視覚情報を用いたアプローチは非常に有効である。ここでいう視覚情報とは、図形や絵や書式などの視覚的表現から得られる情報のことである。

視覚情報の利用という観点から様々な試みが行なわれている[1]。例えばデータの視覚化やプログラム構造の視覚化、プログラミングの視覚化[2]-[4]などの試みがある。

筆者らは、視覚情報としてアイコンとよばれる絵シンボルを用いたプログラミング環境ＨＩ－ＶＩＳＵＡＬの開発[4][5]を行なっている。ＨＩ－ＶＩＳＵＡＬでは現実に存在するオブジェクトをアイコンとして表現しており、アイコンを重ね合わせることによりプログラミングを行なう。そのためユーザは容易にアイコンの示す意味や機能を理解することができる。しかし、ひとつのアイコンが複数の意味を持っているときには、ユーザによるアイコン操作と実際のシステム動作の間の対応が一意でなくなり、実際に起動する関数の決定が困難であるという問題が存在していた。本研究では、現在までに提案されたシステムの特徴をもとに、複数のアイコンの重ね合わせを導入した場合を考察したうえで、実現のために必要なアイコン記述などを決定する。さらにユーザの行った重ね合わせ操作に対する適切なシステム動作を行うための解釈機能の導入を行う。

以下、２章では基本概念及びシステムの概要を述べる。次に３章で複数の重ね合わせに対応した重ね合わせ解釈機能について説明し、４章で本論文のまとめを行う。

2．アイコニックプログラミング環境ＨＩ－ＶＩＳＵＡＬ

本章では、現在筆者らが開発中のアイコニックプログラミング環境ＨＩ－ＶＩＳＵＡＬの概要を述べる。

2．1　ＨＩ－ＶＩＳＵＡＬの基本概念と特長

視覚的な情報をプログラミングに取り入れる試みのひとつにアイコニックプログラミングがある。アイコニックプログラミングとは、ディスプレイ画面上に表示されたアイコン（視覚的に表現されたオブジェクト）を直接操作することによりプログラムを作成する方法である。

現在までに筆者らはアイコニックプログラミングへの一つの指針となるフレームワークを提案している[4]。この中で用いるアイコンは現実に存在するオブジェクトを表現し、アイコンの重ね合わせによってインタラクティブにプログラミングを行うことができる。このときそれぞれのアイコンの意味は重ね合わされたアイコンの対に応じて決定される。本研究で提案するアイコニックプログラミング環境ＨＩ－ＶＩＳＵＡＬはこのフレームワークに基づいている。その特長として以下のものが挙げられる。

（１）アイコン

アイコンは現実に存在するオブジェクトを表現する。アイコンのイメージは自由で変化可能であり、画面上に自由に配置できる。インタフェースの例を図１に示す。また、複数のアイコンから一つのアイコンを構成できる。（エリアの導入）

（２）関数の起動

アイコンの組合せに対し、その組合せから連想される機能が関数として定義される。ユーザはアイコンを重ね合わせることによって、アイコンの組合せを明示し、希望する関数を起動する。

（３）プログラミング機能

ユーザが行った操作に対して、システムはただちに解釈し、実行を行う。これによりインタラクティブなプログラミングを可能とする。また、ユーザが行なった一連の操作をプログラムとして登録できる。

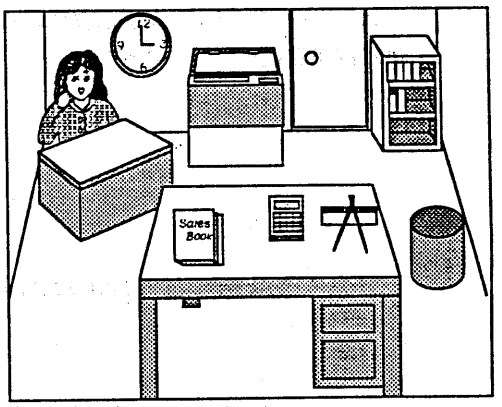


図１　HI-VISUALのインタフェース例

2．2　アイコン

通常の視覚システムでは、データやファンクションなどのオブジェクトを可視化している。しかしながらファンクションを表わすアイコンについてはその機能を正確に表現するイメージを定義することが困難である。そのためＨＩ－ＶＩＳＵＡＬでは、使用するアイコンは現実に存在するオブジェクトを表現する、というアプローチをとっている。ひとつのアイコンは複数の意味をもつことが許されており、アイコンの組合せにより特定の意味が決定される。図２にアイコンの組合せと、それが意味する関数との対応の例を示す。

ここで問題となるのは、ある重ね合わせにおいて起動関数の候補が複数存在することである。つまりアイコンの組合せとファンクションが一意に対応していないため、ユーザの意図にそった意味解釈を行うことが必要となる。このための機能についての詳細は３章で述べる。

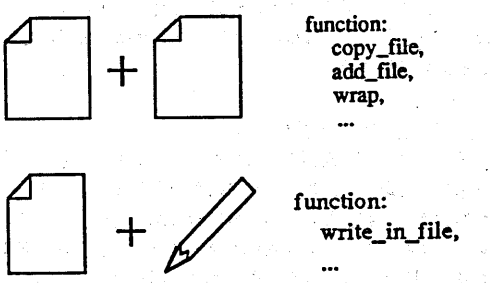


図2　アイコンの組み合わせと関数

2．3　アイコンの重ね合わせ

関数はアイコンの組合せに対し、その組合せから連想されるものが定義される。関数の起動はユーザの重ね合わせ操作によって行なわれる。本研究で扱うアイコンの重ね合わせは二つのアイコンの重ね合わせを拡張した複数アイコンの重ね合わせについて議論する。複数のアイコンの重ね合わせを導入することにより、新たに以下の様なタイプの関数の起動が可能となる。

（1）制約型

複数のアイコンの組合せによって制約の加えられた機能を表現する。あるアイコンが基本的な機能を表わし、他のアイコンがその機能への制約的な意味を表わしている。

（2）多データ型

複数のデータを扱う型。機能を表わすアイコンと複数のデータを表わす複数のアイコンからなる。

複数の重ね合わせの例として、制約型の重ね合わせの例を図3(a)に、多データ型の重ね合わせの例を図3(b)に示す。図3(a)は、定規アイコンと紙アイコンの組合せに対し、ペンアイコンを重ね合わせることにより「図表エディタの起動」という関数を表現し、図3(b)では二つの売上帳アイコンに計算機アイコンを重ね合わせることにより「売上帳の比較」という関数を表現している。ここで関数起動時のアイコンの振舞いについて考えた場合、個々のアイコンはそれぞれ、ファンクション、データ、制約のうちのいずれかの働きをするといえる。

2．4　アイコンクラス記述

本システムにおいて個々のアイコンはオブジェクト指向のクラスにより管理される。アイコン管理のためのすべてのクラスはアイコンクラスのサブクラスとして階層的に定義される。

クラスに記述される属性を以下に示す。

- class：　クラス名
- super class：　スーパークラス名
- attribute：　クラス固有のデータ
- image：　アイコンのイメージ（attributeに依存）
- message：　メッセージ記述
- method：　メソッド記述
- area：　アイコンの持つ空間領域に関する記述

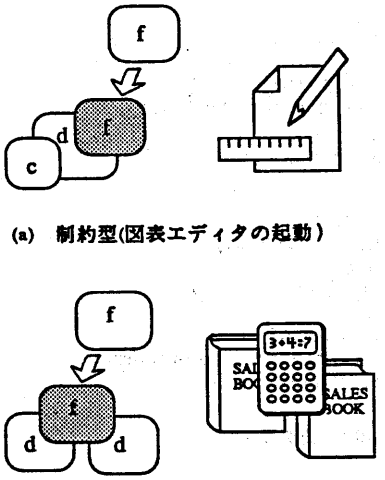


図3　複数の重ね合わせの例

message, method の記述方式は次のとおりである。

proc (d1 d2 ･･･ dn) with (c1 c2 ･･･ cn)

ここで、proc は実行する関数名、dn はデータとなるアイコンのクラスによる引き数列、cn は proc 実行のための制約となるアイコンのクラスによる引き数列を示す。

図4の(a)にペンクラス、(b)に紙クラスの記述例を示す。

起動する関数は重ね合わされたアイコンの属するクラスまたはスーパクラスのメッセージ記述とメソッド記述を利用して決定される。例えばペンアイコンと紙アイコンの重ね合わせの場合、ペンアイコンの属するペンクラスのメッセージ "draw(paper)" を紙アイコンの属する紙クラスに送信すると紙クラスのメソッド "draw(paper)" にマッチし、関数 "draw" が起動関数の候補として選ばれる。

3．重ね合わせ解釈

本章ではユーザの行った操作から適切なシステム動作を行わせるための重ね合わせ解釈機能について述べる。

3．1　概要と方針

ＨＩ－ＶＩＳＵＡＬでは、あるひとつの重ね合わせに対し、起動関数の候補は複数存在する。そのため重ね合わされたアイコンから関数を決定する際に、いくつかのアイコンの組合せやそれぞれの組合せに対応する起動関数の候補がさらに増加する。したがってこれらの候補の中から、よりユーザ意図に沿ったものを選び出す機構が必要となる。

```
class: pen
super_class: stationery
message:
    draw(paper)
    draw-table(paper)with (ruler)
```

(a) ペンクラス

```
class: paper
super_class: stationery
message:
    wrap(paper)
method:
    draw(paper)
    draw-table(paper) with (ruler)
    wrap(paper)
```

(b) 紙クラス

図4 アイコンクラスの記述例

重ね合わせ解釈の方針を以下に挙げる。

（1）アイコンの操作順序

ユーザがあるアイコンを操作し、それによって関数の起動を行おうとした時、直接操作したアイコン及びそれにより直接重ね合わされたアイコン間での関数起動を優先する。

（2）アイコンの重なりに関する情報

アイコンの重なっている領域の広さや上下関係といった情報を利用する（ただし、アイコンの重なりの方向については本研究では考慮しない）。しかし、これらの情報をどの様に適応するのかは応用領域やユーザの好みによって一意に決定できないため、本研究では情報の利用方式（領域優先など）をいくつか提供する。

（3）重ね合わせかたによるデフォルトの検索方法

重ね合わせ方に対応した起動関数の検索方法を設定する。設定された検索方法にしたがって検索を行う。

（4）過去の関数起動情報

ユーザが以前行った関数の起動方法に関する情報を利用する。つまり、アイコンの重ね合わせ方と起動された関数の対応関係の履歴を利用する。履歴は次の二種類のレベルのものを利用する。

1）クラスレベルの履歴

ユーザが実際に行った関数起動の例を蓄えたもの。重ね合わされたアイコンの状態をクラスレベルで保持するとともに起動した関数を保持する。

2）データ、ファンクションレベルの履歴

ユーザが実際に行った関数起動の例から、その場合のアイコンの振舞いを取り出し、蓄えたもの。データやファンクション、制約の配置方法に関する情報を保持している。

3．2 解釈オブジェクト

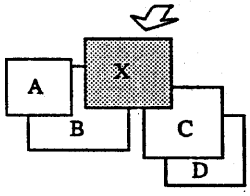
本システムでは重ね合わせ解釈を行なうため特別なオブジェクトを用意する（インタプリタオブジェクトと呼ぶ）。インタプリタオブジェクトは解釈に必要な情報の管理を行なうとともに、その情報に基づいてユーザによる重ね合わせ操作の解釈を行なう。

3．2．1 アイコン管理

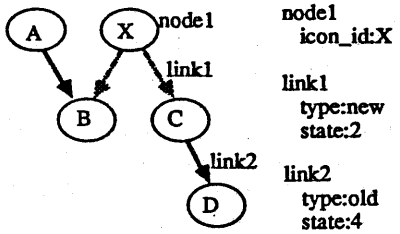
インタフェース画面上に表示されているアイコンに関する情報を管理するためにグラフ構造を用いる。グラフには関数検索時に必要ないくつかの情報を保持させる。グラフのノードにはアイコンを対応させ、リンクにアイコン間の関係を対応させる。図5に例を示す。図5(a)にアイコンの重ね合わせの状態を、図5(b)に対応するグラフを示す。また、エリアに関する情報の管理を行う。ノード、リンクに持たせる情報を以下に示す。

・ノード：icon_id（アイコン識別子）

・リンク：type（種類）,state（領域）,direction（方向）



(a)アイコンの重なり状態



(b)グラフ構造

図5 アイコンの重なり状態とグラフ構造

3．2．2 履歴、関数パターン管理

ユーザの行なった操作のパターン（履歴パターン）や、デフォルトによる検索時の、関数と重ね方のパターン（関数パターン）の管理を行なう。

ユーザがアイコンの重ね合わせによりある関数を起動した場合、関連するアイコンの組合せ,及び重ね合わせのパターン（クラスレベルの履歴）を生成し蓄積する。さらにデータ、ファンクションレベルの重ね合わせのパターンを生成し管理する。

パターンで管理する属性を以下に示す。

1)クラスレベル

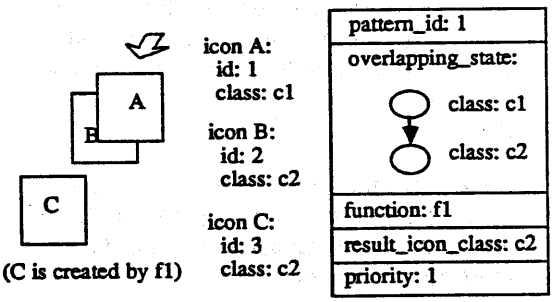
・pattern_id：	パターン識別子
・overlapping_state：	重ね合わせの状態
・function：	起動関数
・result_icon_class：	出力アイコンクラス
・priority：	優先度

2)データ、ファンクションレベル

・pattern_id：　　パターン識別子
・overlapping_state：　重ね合わせの状態
・priority：　　優先度

クラスレベルの履歴パターン管理の例を図6に示す。図6(a)にユーザによる関数起動の状態を、図6(b)に対応する履歴パターンを示す。

またデフォルトの情報として複数の重ね合わせ方法の分類に対応したパターン（重ね合わせ方と起動する関数との対応）の管理を行う。



(a) 関数起動状態　　(b) 履歴パターン

図6　履歴パターン(クラスレベル)の管理

3．2．3　解釈手順

重ね合わせ解釈はメソッド"interpret"によって行われる。解釈にはグラフ、履歴パターン、関数パターンを利用する。

解釈手順を以下に示す。

（1）操作されたアイコンの情報からグラフを更新する。
（2）グラフに保持されている情報から、可能なアイコンの組合せ（サブグラフ）に対し、優先順位を決定する。
（3）優先度の高いものから順次、履歴パターンによる検索を行う。
　(3.1) クラスレベルのパターン検索
　(3.2) データ、ファンクションレベルのパターン検索
（4）関数パターンによる検索を行う。

3．3　解釈機構

解釈はインタプリタオブジェクトと一般オブジェクト（アイコン）間のメッセージのやり取りによって実現される。その解釈結果にしたがってアイコン間のメッセージ送信が行なわれ、関数が起動される。ユーザアクションから解釈起動用のメッセージ等を生成するためのオブジェクトとしてアクションマネージャを用意している。

解釈を行なう場合、各オブジェクトは解釈用の特別なメッセージ、メソッド（interpret, mess_trans, create, delete, move, next, executed）を用いる。

以下では解釈用メソッドについて説明する。

（1）アイコン生成メソッド：create (x overlap)
　ここで x は特定のアイコンを示し、overlap は重なり情報を示す。以下、同様。
（2）アイコン削除メソッド：delete (x overlap)
（3）アイコン移動メソッド：move (x overlap)
　（1）から（3）はアイコン操作用のメソッドで、グラフの更新を行なうもの。
（4）解釈メソッド：interpret (x overlap)
　アイコン x が重なり情報 overlap をともなって操作されたとき解釈を実行し、解釈結果としてメッセージ mess_trans を生成する。
（5）次候補要求メソッド：next
　次候補の mess_trans を生成する。
（6）メッセージ送信メソッド：
　　mess_trans (x proc (A.a B.b ･･･) (C.c ･･･))
　ここで小文字 x, a, b, c は特定のアイコンを示し、大文字 A, B, C は特定のクラスを示す。ここで A.a, B.b はデータ引数、C.c は制約引数を示す。また proc は起動関数を示す。これら引数のうち proc は unknown という値も許される。
　アイコン x に対し、関数 proc を起動させるようなメッセージを引数アイコンをともなった形式で送信する。また proc が unknown の場合は引数クラスのマッチするような適当な起動関数が代入される。
（7）終了メソッド：executed(proc (A.a ･･･) (C.c ･･･))
　引数として起動関数及び引き数列をもつ。
　解釈を終了し、履歴パターンを生成する。

各オブジェクトが保持するメッセージ、メソッドは以下の通りである。

（1）アクションマネージャ
　メッセージとして create, delete, move, interpret をもつ。
（2）インタプリタオブジェクト
　メッセージとして mess_trans をもつ。
　メソッドとして create, delete, move, interpret, next, executed をもつ。
（3）アイコン
　メッセージとして next, executed をもつ。
　メソッドとして mess_trans をもつ。

図7に各オブジェクトのメッセージ送受信関係を示す。

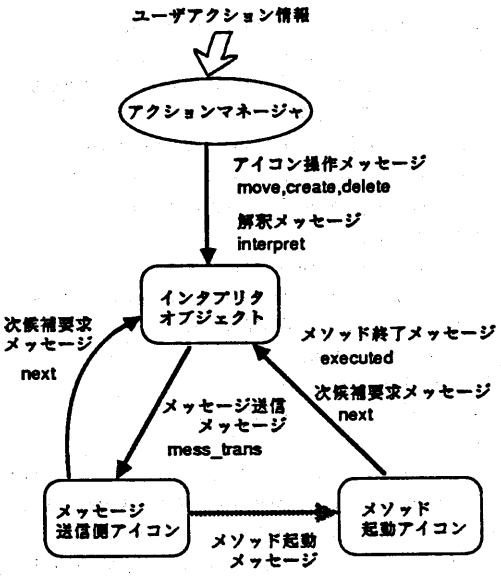


図7　メッセージの流れ

3．4　関数起動手順

ユーザアクション発生から関数起動までの手順を以下に示す。

（1）ユーザアクションが発生すると、アクションマネージャはインタプリタオブジェクトにメッセージ interpret を送信する。

（2）インタプリタオブジェクトは解釈を行ない、その結果に従ってメッセージ送信側のオブジェクトにメッセージ mess-trans を送信する。

（3）メッセージ送信側のオブジェクトはメッセージ受信側のオブジェクトに関数起動メッセージを送信する。

（4）メッセージ受信側のオブジェクトはメソッドの起動をユーザに確認し、ユーザの意図したものなら実行するとともに、インタプリタオブジェクトに終了メッセージ executed を送信する。そうでなければ次候補要求メッセージ next をインタプリタオブジェクトに送信する。

4．おわりに

本稿では、アイコンの重ね合わせを利用したアイコニックプログラミング環境ＨＩ－ＶＩＳＵＡＬにおける重ね合わせ機能について考察し、ユーザの重ね合わせ操作の解釈機能について述べた。解釈機能の実現のため、解釈用のオブジェクトを定義し、メッセージのやりとりによる解釈機構を導入した。これにより、複数のアイコンの重ね合わせによる関数起動を可能にするとともに、ユーザの行った画面操作に対する適切なシステム動作の実行を可能とした。

解釈方法についての問題点のひとつとして、ある応用領域での重ね合わせ解釈が別の応用領域では必ずしも適切でないということが挙げられる。この点についての検討は今後さらに行なう必要がある。また、本研究ではユーザによる一連の処理の流れを利用した解釈は行っていないが、ユーザの行う操作はそれ以前に行った操作と関連する場合が多い。この点を考慮した解釈方法の拡張が考えられる。

またＨＩ－ＶＩＳＵＡＬにおいてのアイコンによるプログラミング手法はユーザにとって馴染みやすいが、重ね合わせと関数の適切な対応が重要となる。特に複数の重ね合わせを導入するにあたっては、ユーザに混乱をきたす場合が発生しやすいため、対応関係を十分考慮して設計する必要があると言える。

今後の課題として、先に述べた点を考慮した解釈方式の拡充、重ね合わせと関数の対応関係に関する検討、またプログラミング機能の拡充（制御構造等の導入）などが挙げられる。実験システムは、視覚的プログラミング環境構築ツール[6]を利用して、現在ワークステーション上に構築中である。

謝辞

本研究を進めるにあたり、システム開発に御尽力頂いた広島大学工学部第二類情報システム研究室の瀬下順一氏、与倉秀次氏、ならびにご討論及び有益なご意見を頂いた同研究室の諸氏に感謝する。

参考文献

[1] N.C.Shu, "Visual Programming", Van Nostrand Reinhold Company, New York, 1988

[2] W.Finzer and L.Gould, "Programming by Rehearsal", Byte, Vol.9, No.6, pp.187-210, 1984

[3] Robert V.Rubin, Eric J.Golin and Steven P.Reiss, "Think Pad: A Graphical System for Programming by Demonstration", IEEE SOFTWARE, Vol.2, No.2, pp.73-79, 1986

[4] M.Hirakawa, M.Tanaka, and T.Ichikawa, "An Iconic Programming System, HI-VISUAL", IEEE Trans. on Software Engineering, Vol.16, No.10, pp.1178-1184, 1990

[5] M.Hirakawa, M.Yoshimi, M.Tanaka, and T.Ichikawa, "A Generic Model for Constructing Visual Programming Systems", Proc., IEEE Workshop on Visual Languages, pp.124-129, 1989

[6] M.Hirakawa, M.Yoshimi, and T.Ichikawa, "A Universal Language System for Visual Programming", Proc., IEEE Workshop on Visual Languages, pp.156-161, 1990