ソフトウェア工学 97－9
（1994．3．1）

オフィス環境向けHI-VISUALシステム

鳥居昭彦† 加登基二† 平川正人‡ 市川忠男‡

†広島大学大学院工学研究科
‡広島大学工学部

オフィス環境への適用を意図したアイコニックプログラミングシステムHI-VISUALの実現について述べる。

HI-VISUALは、データおよびファンクションの両方の性質を持ち得るアイコンをプリミティブとして持つプログラミングシステムであり、実行可能な処理はアイコンの組合わせに応じて適宜決定される。本論文では、まずインタフェース画面上で行った一連のアイコン組合わせ操作を一旦システム側で履歴として蓄えておき、ユーザはその履歴をベースに、必要に応じて操作手順を編集し、汎化することでプログラムを作成する機能について述べる。また、オフィス環境向けに用意したアイコンについて述べる。

HI-VISUAL System for Office Use

A.Torii , M.Kado , M.Hirakawa , T.Ichikawa

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

This paper describes an iconic programming system HI-VISUAL intended for use in office environments.

HI-VISUAL provides primitive icons. Each icon assumes either data or a function. A function is determined depending on a combination of icons. A series of icon combinations specified graphically on a display is stored and managed as a history. Then the user assembles some operations selectively from the history and makes a program as desired. Icons provided for office information processing are presented.

1 はじめに

近年のハードウエア、ソフトウエア技術の進展に伴い、コンピュータの利用が急速に広まってきている。しかし我々がそれを利用するにあたっては、一般に多くの知識や経験を必要としている。例えば、人間がある一つのプログラムを計算機上で作成しようとする時、プログラムをそのまま自然言語で計算機上に記述することは難しく、計算機のための言語の知識が必要となる。人間と計算機の処理形態のギャップについては、これまでは人間がそれを埋めるという形をとっていた。

そこで人間と計算機との対話を柔軟に行うための手段の一つとして、文字情報に加えて図形や絵といった視覚的表現を用いる手法が提案されている。視覚的表現には、

- 絵は言葉よりも認知しやすく、情報量も多い。
- 適切に描かれた絵は言葉の障壁を持たない。

などの特徴がある[1]。この様な特徴を持った視覚的表現をプログラムに利用し、構造や処理の流れを視覚化することにより、コンピュータの操作を熟知していない初心者でもプログラムの理解が容易になると考えられる。我々は、プログラミングにおいて視覚的な対話を用いたアイコニックプログラミング環境 HI-VISUAL の研究を行っている[2]。

本稿では、プログラミングに不慣れなユーザを対象とし、ユーザがインタフェース上で行った操作手順の履歴をユーザ自身が編集し、これによってプログラムを定義する手法について考察する。

これと同様に過去のユーザ操作を履歴として管理し、その編集を許すような試みとして、Mondrian[3]や Chimera[4]などがあるが、これらにおいては応用が描画操作に限定されている。

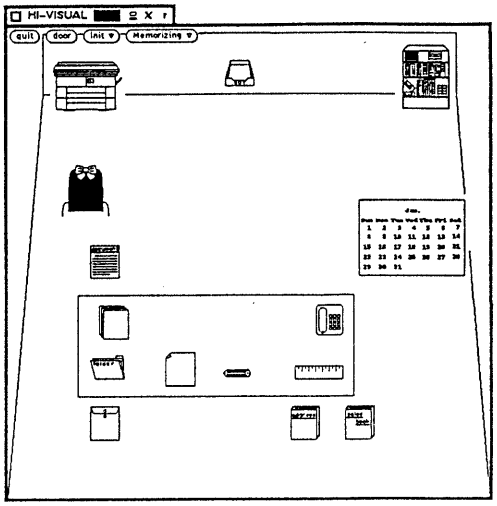


図 1: インタフェース画面

以下、2章で HI-VISUAL の概要とシステム構成について説明し、3章で HI-VISUAL インタフェースと履歴編集ツールを利用した履歴編集の方法、ならびにツールの提供する編集機能について述べ、4章ではオフィス環境におけるプリミティブなアイコンについて説明する。

2 HI-VISUAL の概要

2.1 アイコンとプログラミング

HI-VISUAL は、アイコンをベースにした視覚言語プログラミング環境である。インタフェース画面には複数のアイコンが表示されている（図 1）。ユーザはマウスを用いて画面上のアイコンを組合わせ、これによってプログラミングを行う。

HI-VISUAL における個々のアイコンは現実に存在するオブジェクトを表しており、特定の役割が決定されておらず、能動的および受動的性質の両方を持ち得る。アイコンが組合せられた時点で、はじめてある特定のファンクションが決定・実行される。なお、アイコン組合せの指定は、インタフェース画面上でのアイコンの重ね合わせという操作で行われる。ファンクションが実行された後、処理内容によって新

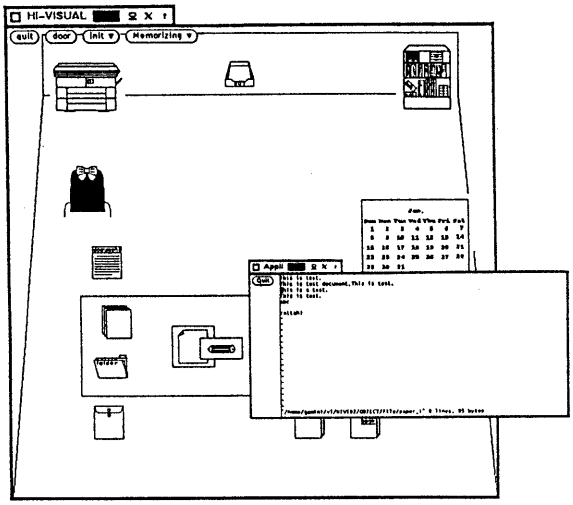


図 2: HI-VISUAL の実行例

たなアイコンが画面上に表示されたり、あるいは画面上のあるアイコンが消去されたりする。これを踏まえてユーザは順次、処理を進めていく。

例えば *paper*アイコンと *pen* アイコンの重ね合わせると、図 2に示すようにエディタが起動される。文書が作成され *document* アイコンが現れるが、このアイコンを *folder*アイコンと重ね合わせすることによって、*folder*ウィンドウが開き、その *document* アイコンをウィンドウの任意の位置に収納することができる。

なお、ある特定のアイコンの組合わせに対して規定される実行可能なファンクションは一つとは限らない。一般には複数のファンクションが想起される。

記述された一連の処理手順はプログラムとして *secretary*アイコンに登録される。その後では、引数として規定したアイコンを *secretary* アイコンに重ね合わせ、これによって登録されたプログラムを実行させる。

2.2 システム構成

HI-VISUAL は以下に示すような４つのマネージャから構成されており、マネージャ間では socket 通信を用いて情報の交換が行われている。

Interface-Manager(IM): インタフェース管理を行う。具体的には、ユーザからのアクションを受け取り、それを IP に伝える。また、OM から送られてきた情報を画面に出力する。

Interpreter(IP): IM から送られてくるユーザアクションに対して、OM からの情報をもとにその意味解釈を行い、PM に指示を出す。

Object-Manager(OM): アイコンやクラス情報の管理を行い、他のマネージャからの要求に応じて情報を提供する。

Process-Manager(PM): IP からの指示に従ってプログラムの実行を制御する。

3 履歴を用いたプログラミング

3.1 履歴を扱う背景

これまでの HI-VISUAL システムでは、一連の重ね合わせの手順をプログラムとして登録するにあたって、登録の開始と終了をユーザが明示的に宣言し、その間にユーザによって行われた処理が全てプログラムとして登録されていた。しかし、以下に述べるような問題点が存在する。

- HI-VISUAL のインタフェースでは、最新の状況しか表示されないため、重ね合わせが進むにつれて記述された過去の一連の重ね合わせ手順の把握が困難である。したがって、過去に戻れることがプログラミング時の心理的負担を軽減する。
- プログラムの引数となるアイコンをプログラム登録過程で指定しなければならない。つまり、ユーザは作成しようとするプログラムを正確に把握しておく必要がある。

履歴情報

class	paper
method	write_paper
input	pen , paper
output	
appear	document
disappear	
overlap	.....

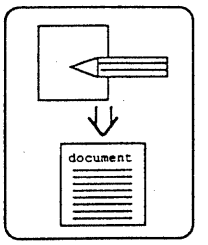


図 3: 履歴をもとにした一操作の表示

このような問題点を解決するために履歴をベースにしたプログラミング機能を新たに導入する。

ここでいう履歴プログラミングにおいては、インタフェース画面上で行った一連のアイコン組合わせ操作がシステム側で履歴として蓄えられていて、ユーザがある操作手順を残したい時に履歴の編集機能が呼出される。そして、ユーザは履歴をベースに必要に応じて操作手順を編集し、汎化することによってプログラムを作成する。

3.2 履歴編集

HI-VISUAL インタフェースと履歴管理/表示ツールを使った履歴編集の方法について説明する。

履歴表示にあたっては、個々のアイコン重ね合わせ操作を表示の単位とする。例えば、図 3 に示すように *pen* と *paper*が重なってファンクション write_paper が実行され、その結果出力アイコンとして *document* が得られたという対応が履歴の基本要素として表示される。なお、その時の内部管理構造を図 3中に併せて示す。

履歴を表示する際に、操作の流れを画面上に表示するわけであるが、ユーザが行った通りの順にアイコンの重ねあわせを一列に表示するだけでは、個々の操作におけるデータの流れを把握する事が困難であり、一連の操作列を取りだすことが困難になる。このため、履歴情報をもとにデータフローグラフを作成し、これを表示の際に用いる。

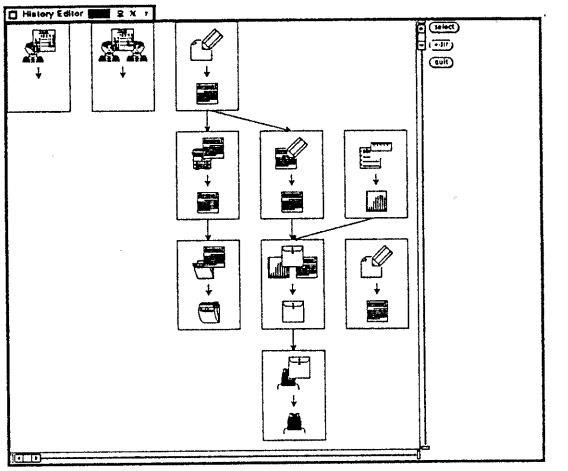


図 4: History_Editor（初期画面）

データフローグラフの作成方法としては、個々の操作の入力アイコンが、その操作より以前に行われた操作の出力アイコンに同一名のアイコンが存在するかどうかを探索する。存在すればその操作間にリンクを張る。なお、履歴情報はシステム起動時からすべて保存するのではなく、あらかじめ規定しておいたステップ数だけを保存する。

HI-VISUAL 上での履歴編集をサポートするためのツールとして History_Editor を提供する（図 4）。History_Editor の初期画面には、履歴情報をもとに作成したアイコンの重ね合わせ操作のデータフローグラフが表示されており、画面上におけるリンクは操作間の一つのデータアイコンの流れを表している。

History_Editor の役割として、

- 重ね合わせ操作の履歴、あるいは編集途中のプログラムの全体を把握することができる。
- 編集の対象を、画面上に表示されている操作やリンクから直接指定することができる。
- 過去のある時点の状態にプログラミング画面を戻すことができる。

が挙げられる。

HI-VISUAL 上での履歴をもとにしたプログラムの編集と登録の手順は次の通りである。

1. ユーザが行ったアイコンの重ねあわせの操作列の一部をプログラムとして残したい時に、History_Editor を起動する。

2. History_Editor 上に表示された履歴からユーザが必要な操作列を取り出す。その後、取り出された操作列のみ画面上に表示される。

3. History_Editor で取り出された操作列に対して、修正や新たな操作の追加、不要な操作の削除、そしてプログラムの引数となるアイコンの指定等の編集を行う。

   History_Editor で編集対象を指定すると、HI-VISUAL プログラミング画面 (図 1) は指定された過去の状況に戻る。その画面上で操作の修正や追加を行うこともできる。

4. 編集が終われば プログラム名を付け、それを登録しておく *secretary* アイコンを HI-VISUAL プログラミング画面で指定する。

3.3 History_Editor の編集機能

本節では、ツールが提供する各種編集機能について例を示しながら説明する。

図 5は図 4からプログラムとして登録するのに必要な操作列だけを取り出したものである。なお説明の都合上、各操作にアルファベットを付加する。操作 A では、*calendar* でミーティングメンバである二人のスケジュールを確認してスケジュールの調整を行い、Bでミーティングを開くための文書 *doc_a* を書き、Cで資料を書いて、さらに Dでそれら二つの書類をまとめて封筒に入れている。Eでは、*secretary* が書類をそのメンバに送り届けるというプログラムをすでに保持しているので、それを利用している。

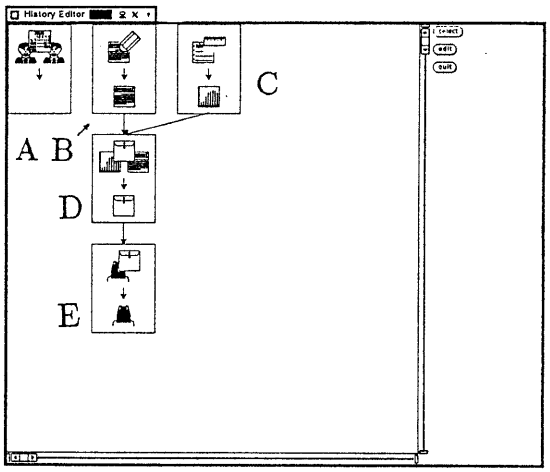


図 5: 編集対象の操作列

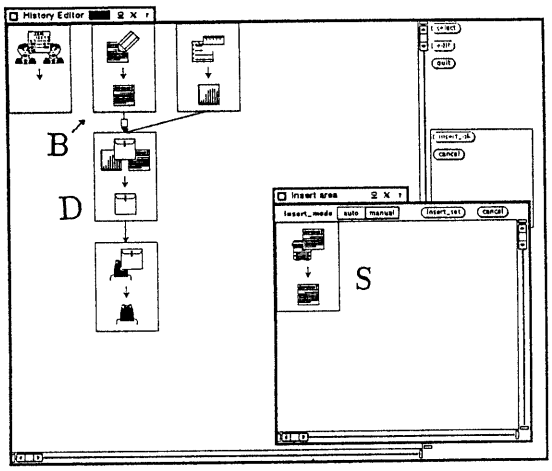
3.3.1 挿入

History_Editor 上で挿入の対象を指定し、HI-VISUAL 上で重ね合わせを示すことで、編集中の操作列の途中、始端、終端に新たな操作を追加することができる。

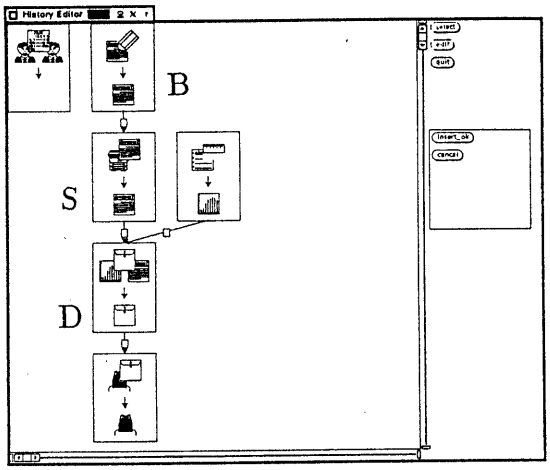
例えば、図 5において操作 BD間に新たな操作を追加する時、HI-VISUAL プログラミング画面では、履歴情報をもとに、指定されたリンク先の操作 Dの実行前の状態に戻り、アイコンが再表示され、これによって操作 D以前で生成されたアイコンを使ったプログラム定義が行われる。

操作 Bによって生成された *doc_a* に対し、プログラミング画面上で新たな操作 (列) を指示すると、その情報は History_Editor に送られ、一旦別のウィンドウにその操作 (列) が表示される。図 6(a) では、もとの *doc_a* に対してコピーを行い、別の *doc_b* を生成するという新たな操作 Sが追加された時の様子を表している。

追加操作の指示が終われば、それらの操作 (列) が編集元のプログラム中に埋め込まれる。この時、新たな操作の入出力アイコンが挿入元の操作の入出力アイコンと共に同一のクラスで



(a)



(b)

図 6: 操作列の挿入

あった時に限って、リンクの張り直しが可能となる。

この例では、追加された操作 Sの出力アイコンが doc_b で、操作 Dの入力アイコンはもとの doc_a であるが、同一クラスであるのでリンクの張り直しが可能になる。操作 Sに対して、新たに BS, SD間にそれぞれリンクが張られる。またこの時、操作 Dの入力アイコンが doc_b に変更される。この様子を図 6(b) に示す。

3.3.2 削除

消去する操作がデータフローの始端あるいは終端の操作であれば直ちに消去可能であるが、データフローの途中にある操作の消去に対しては、データフローのリンクに意味的矛盾が生じないようにする必要がある。

例えば、図 6(b) において操作 Sが消去の対象となった場合、BD間の入出力アイコンのクラスが一致するため、操作 Sは消去可能となる。しかし、操作 Dの削除のようにクラスの不一致が生じる場合には、ユーザに消去不可能であることを示す。

3.3.3 変更

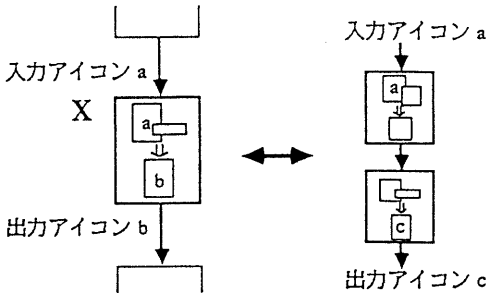
- 操作の変更

  図 7(a) に示すように、ある操作 Xの差し替えを行う時に用いる。挿入の場合と同様に、HI-VISUAL プログラミング画面は操作 Yの重ね合わせが行われる直前の状態にまで戻ってアイコンを再表示する。

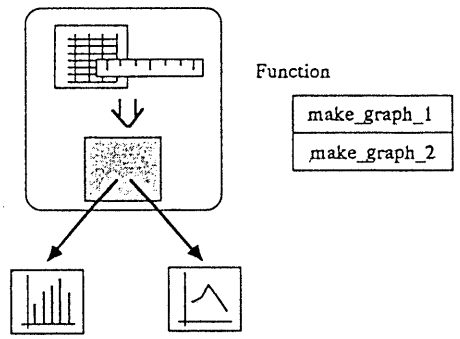
  ユーザによって規定された置き換えるべき操作 (列) を操作 Xと置き換えるにあたって、入出力アイコンのクラスの整合性が取れているかどうかチェックを行う。図 7(a) の場合、出力アイコンに関しては、もとの操作 Xの出力アイコン b と新たな操作の出力アイコン c が同一のクラスであるかどうかを調べる。同一クラスであれば指示通りにリンクを張って変更操作を終了する。そうでなければリンクを切断することになるのでユーザにその旨を伝える。

- ファンクションの変更

  HI-VISUAL では一つの重ね合わせに対して解釈パターンが複数存在する場合がある。そこでアイコンの重ね合わせに対して、別の実行可能なファンクションを割り当てることができる。まず、実行可



(a) 操作の変更



(b) ファンクションの変更

図 7: 変更

能なファンクション候補に対して、変更後にリンクを張り直すことが可能かどうかを調べ、変更可能なファンクションだけがユーザに示される。図 7(b) にその例を示す。例えば、*list* と *ruler* の重ね合わせから２つの変更可能なファンクションがある場合には、それらがユーザに提示され、編集時に選択することが可能となる。

3.3.4 引数アイコンの指定

プログラムの登録にあたっては、入力パラメータとして機能すべきアイコンを提示する。入力パラメータとして提示されなかったアイコンについては、プログラム実行時に常にそのアイコン（インスタンス）が参照されることになる。

例えば、図 5において操作 Bの入力アイコン *doc_a* を引数アイコンとして宣言すれば、実行時に特定の *document* アイコンを引数として渡すことで、そのプログラムを起動させることができる。これとは逆に引数アイコンとして宣言しなければ、実行時には常にこの *doc_a* を使用することになる。

3.3.5 アイコンの名前付けの指定

あるファンクションを実行した結果、新しくアイコンが生成される場合がある。例えば *document* を *copy-machine* に重ねた時には、出力アイコンとして新しい *document* が生成される。この様な場合に、システムでは新しく生成されたアイコンにある特定のネーミングを与えたり、プログラム実行時に動的にユーザが指定したりすることができる。

4 オフィス環境におけるアイコン

システムでは、オフィス環境への適用に際して、いくつかのオフィス環境特有のアイコンを用意している。それらについて以下に説明する。なお、以下でいう書類アイコンとは、document や list, graph 等のアイコンをさす。

folder: 書類アイコンとの重ね合わせによって、folder ウィンドウが開き、画面上の任意の位置に収納できる。逆に単独で起動することでウィンドウが開き、任意のアイコンをそのフォルダから取り出すことができる。書類アイコンの特定のフィールド値の順に自動的にリンクが張られ、それに沿ってブラウジングを行うことができる。

bookshelf: アドレス帳やスクラップブックのようなデータベース機能として存在する。

calendar: *document* との重ね合わせで任意の日付に覚書として記憶させておいたり、

member との重ね合わせでその人のスケジュールを登録・検索することができる。さらに別の *member* を次々と重ね合わせていくことによって、重ね合わされたメンバ全員のスケジュールを把握することができる。また、*secretary* と重ね合わせことによって、プログラムをある決まった日に実行するように指定することもできる。

member: ある一個人を表している。*document* との重ね合わせでメイルを送ったり、*calendar* との組合わせでスケジュール管理等に用いられる。

group: 複数の *member* をまとめて管理するためのもので、構成メンバを自由に変えることができる。これを用いて、例えば *document* との重ね合わせで構成されるメンバ全員にメイルが送られる。また、*calendar* との重ね合わせによって、構成されたメンバ全員のスケジュールを把握することができる。

mailbox: *document* や *envelope* との重ね合わせでメイルを出したり、また外部からのメイルを受け取る。*calendar* と組合わせて日時指定を行うこともできる。

5 まとめ

オフィス環境を対象としたアイコニックプログラミングシステム HI-VISUAL の実現について述べた。

まず、ユーザの行った履歴からプログラムの作成をサポートする History_Editor について述べた。このツールでは、履歴をもとにデータフローグラフを表示する。ユーザはそのグラフから必要な部分だけを取り出す。プログラム編集中には、HI-VISUAL プログラミングインタフェース上で追加されたり、あるいは修正された操作内容が History_Editor に送られてデータフローグラフが更新されるため、現在のプログラムの処理の流れを容易に把握することができる。

また、スケジュール管理を中心に、オフィス業務の支援を目的としたいくつかのオフィス環境向けアイコンについて述べた。

本システムは、SPARC-station 上で C, X-View, Scheme を用いて構築中である。

参考文献

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

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

[3] H.Lieberman. "Mondrian: A Teachable Glaphical Editor," in [5]

[4] D.Kurlander and S.Feiner, "A History-Based Macro By Example System," *Proc. SIGGRAPH and SIGCHI, User Interface Software and Technology(UIST '92)*, ACM, pp.99-106, 1992.

[5] A.Cypher(ed.), "Watch What I Do: Programming by Demonstration," The MIT Press, 1993.

[6] L.Ford and D.Tallis, "Interacting Visual Abstractions of Programs," *Proceedings of 1993 IEEE Symposium on Visuall Languages*, IEEE Computer Society Press, pp.93-97, 1993.