ソフトウェア工学 84－8
（１９９２. 3. 3）

# 絵コンテ式のビジュアルプログラミング

小澤　正樹
東京電力㈱　システム研究所

処理の対象と操作および手順を映画の絵コンテ風に表現することを特徴とするビジュアルプログラミング表現技法について述べる。この技法は数学的対象物をビジュアルに表現する事を主なねらいとしている。最短経路探索のアルゴリズムの定義にこの技法を応用して、具体的な表現形式の検討と評価を行った。その結果ある種の問題領域のプログラミングに適用可能であるという感触を得た。現実のプログラミング課題に適用して、実用面での効果の有無を明らかにする事が課題として残されている。

# Visual presentation like a continuity for visual programing

Masaki Ozawa
Computer & Communication Research Center
Tokyo Electric Power Company

A visual presentation technique for visual programing is discused. Objects, operations and procedure are expressed in the style like a continuity used by film director.The main target of this technique is visualization of mathematical objects. It was applyed to define a argorism that finds the shortest route, and its effectiveness was evaluated. The technique is seemed to be apricable for some kind of programing. It is left as a probrem to discuss how this technique works for actual programing.

## 1　まえがき

近年、パソコンの普及に伴い、計算機の非専門家によって、プログラミングやシステムの開発が行われる事が盛んになって来た。非専門家によるプログラム開発の例としては、シミュレーションやデータ解析などのプログラムをBASICやCを用いて開発するケースがある。一方、表計算などの汎用ソフトの発達に伴い、非専門家による業務システムの開発も盛んになって来た。非専門家によるシステム開発の例としては、表計算ソフトと電子メールソフトを組み合わせた予決算集計システムなどがある。しかし、これらのプログラミングやシステム開発の環境には以下の問題が残されている。

【非専門家向けプログラミング環境の問題】
- 経験を積まなければプログラミングに習熟する事が出来ない。
- 他人の作ったプログラムの理解に時間がかかる。（作成者自身も、作成途中の自分のプログラムが理解しづらい）

【汎用ソフトによるシステム開発の問題】
- データ入力の督促の自動化など、汎用ソフトにない機能を組もうとすると、シェルスクリプトなどの言語によるプロセス定義を要求され、簡易なシステム開発という理想から離れてしまう。

このような問題を解決するために、ビジュアルな表現をプログラミングやシステム開発に応用する技術も現れて来つつある。

（1）プログラム構造をダイアグラムで表現する機能を備えたプログラムエデイター
（2）オブジェクト指向プログラミングのオブジェクト間の関係のビジュアルな表現
（3）アイコンなどで表現されたプロセス部品の組合せによるシステム開発

ところが、プログラミングに的を絞って観察すると、アルゴリズムの組立までを支援するようなビジュアル表現技法は、さほど発展していないようである。そのため、プログラムの理解しやすさを向上させる事は相変わらず困難のようである。そこで、アルゴリズムの組立までも表現できるビジュアルな表現技法を考案し、具体的な例題への適用を試みた。その結果プログラムの表現方法としての可能性を確認したので紹介したいと考えた。

## 2．プログラミングの一般的な手順

一般的に、プログラミングは次の３つのステップで行われる。

(1) 頭の中での問題の定義と解決手段の実験
　頭の中や紙の上のスケッチで問題をイメージ化し、解決手段の試行を行う。
(2) 数学的対象への置き換えと手順の検証
　問題とその解決手段を、集合や関係などの数学的な対象物へ置き換え、アルゴリズムの検証を行う。
(3) インプリメンテーション
　機械で実行可能なプログラムの組立を、プログラミング言語や周辺機器制御言語、ＧＵＩ制御言語などを用いて行う。

数学的な対象物への置き換えのステップまでは、ビジュアルなイメージを用いて作業が行われる。ところが、手順の検証やインプリメンテーションのステップはテキストベースの作業である。そのため、ビジュアルなイメージからテキストベースの言語への翻訳を行いながら作業を進める事を強いられている。

従って、数学的な対象をビジュアルなままで扱える表現技法があれば、非専門家ばかりでなく、専門家にとっても開発しやすいプログラミング環境となると思われる。

また、プログラムの意図すなわち数学的な対象物への置き換えのステップまでの内容がビジュアルに表現できれば、意図の理解が容易になり、再利用すべき部分を見つけ出す事も容易になるので、開発効率を上げる事ができるようになると考えた。

## 3．ビジュアルな表現のための基本的な技法

### 3．1　処理手順や操作の表現に関する技法

（1）手順の進展の表現に関する技法
　状況を一つの絵として表現する。手続きの進

行、即ち状況の進展は映画のカメラ台本である絵コンテ風に画面に表示される。繰り返しのためにある状況へ戻るためには、ＶＴＲのリワインドのような表現を用いる。

（２）サブルーチンの呼出し時の引数の表現

サブルーチンの内部から見たオブジェクトと呼び出し側のプロセスの内部から見たオブジェクトの間の同一視を行う事で引数を表現する。同一視は、それぞれのオブジェクト間を線でつなぐ事で表現する。

（３）サブルーチンのアイコン化

サブルーチン自体をアイコンで表現してサブルーチンパレットの中に配列する事により、簡単に呼び出せるようにする。

（４）対象固有の操作の呼び出しに関する技法

選択したオブジェクトに付随するメニューから対象固有の操作を選択する。これについてはSmall Talkのプログラミング環境でが参考になる。また、集合に対する操作については、それらをアイコン化して集めたパレットから選択する事もできる様に考えた。

（５）既存のアルゴリズムの呼出しに関する技法

サブルーチンの呼び出しと同様である。

（６）操作の合成などに関する技法

関数的な（あるいはデータフロー的な）操作を線で連結する事により、合成関数を定義できる。これについては、Hyper COBOL[1]が参考になる。

（７）手続きの構造化に関する技法

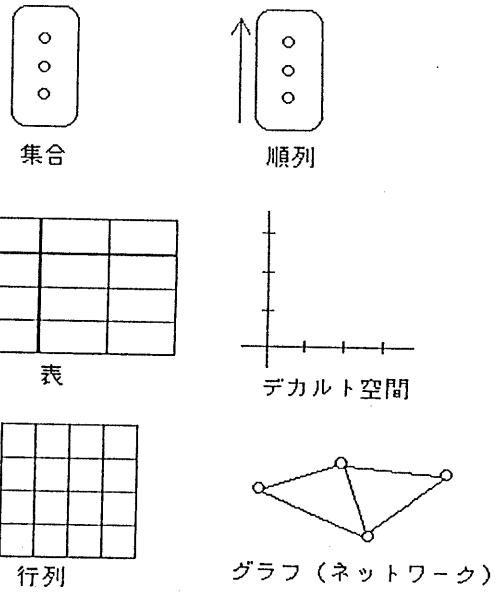
例えばＷＨＩＬＥ文に対応する構造化は次の様に行う。

- ＷＨＩＬＥ文に対応して一つの場面を作る。
- その場面では判断を行う手続きブロックと繰り返される手続きブロックのそれぞれをアイコン化して表示する。
- アイコン化された手続きブロックをクリックするとそれぞれの内容の定義場面が現れる。

## ３．２　処理の対象の表現に関する技法

（１）　数学的な対象物

集合、順列、対、デカルト空間、行列、表、グラフなどがアルゴリズムの定義に必要となる事が多い。これらはそれぞれ、次の様に表現できる。




（２）　クラスの表現技法

クラスは典型として扱い、画面上の表現はインスタンスの表現と同型式とする。これについてはPrograming by Rehearsal[6]の「劇団」が参考になる。

（３）　対象の読み替えに関する技法

アルゴリズムの定義の過程で、処理の対象の属するクラスの読み替えが必要になる。たとえば、(a,b)のような対を{a,b}のような集合に読みかえる場合やソート済みの表の内容を順序集合に読み替える場合である。このような場合オブジェクト指向言語では通常メソッドの継承を利用する。しかし、読み替え操作のアイコンを通して変換を行うことにより、陽に読み替えを行う方法も考えられる。

## ３．３　編集機能

対象のアイコンと操作のアイコンなどを結び付けるために「線を引く」作業などは、Macintoshのマンマシンインタフェースが参考になる。

## ４．適用例

### ４．１　例題の内容

3章で示した技法の範囲で、どこまでアルゴリズムの定義が可能であるかを確かめるために、最短経路探索問題の解法の一つであるNillsonの発見的探索アルゴリズム（Aアルゴリズム）[3]を例題として採りあげた。

まずAアルゴリズムを簡単に説明する。

【ステップ1】
開始ノードをexpandする。すなわち、OPENというリストに後続ノードを入れ、開始ノードに遡るポインタを付けておく。

【ステップ2】
最少の経路コストを与えるノードをOPENから除きXとする。それがゴールであれば成功として終了する。OPENが空であれば、失敗として終了する。

【ステップ3】
その他の場合は、Xをexpandし、ステップ2へ。

## 4．2　ビジュアルに表現すべき操作

Aアルゴリズムを定義するために必要な操作を以下に整理する。

- ネットワーク構造のクラスを呼び出す操作
- 開始ノードやゴールノードを特定する操作

<expandするサブルーチンの定義>

  - expandサブルーチンの定義の開始操作
  - 後続ノードの集合を生成する操作
  - 開始ノードに遡るための経路情報を属性として張り付ける操作
  - 経路コスト計算関数呼び出し操作
  - 後続ノードをOPENに加える操作
  - サブルーチンのアイコン化

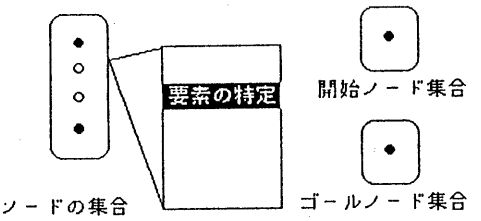
- 開始ノードをexpandする操作
- OPENの中から最少コストノードを選別する操作
- 選別したノードをOPENから除く操作
- 除いたノードとゴールの同一性を判定する操作
- 同一でなければexpandの対象として、繰り返しを行う操作
- 同一であれば、開始ノードまで遡って、最少コストの経路の表示を出力する操作

## 4．3　ビジュアルな表現の実際

上で整理した操作を実際にビジュアルに表現するための視覚的デザインを考案した。まだ机上のスケッチにすぎないが、一部について紹介する。

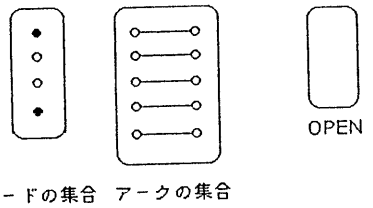
（1）開始ノードやゴールノードを特定する操作

ノードの集合に付随した操作のメニューから要素の特定操作を選択し、マウスで適当なノードをクリックする事で、開始ノードやゴールノードとして特定する。



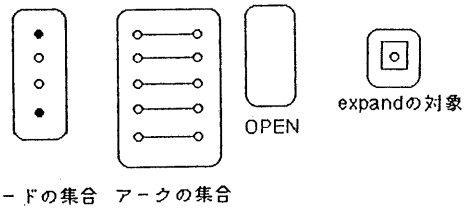

（2）OPENという集合の定義

メニューを手繰ってクラスライブラリの中から集合を選択して画面の中に配置し、OPENという名を付与する。（この時点では空集合）



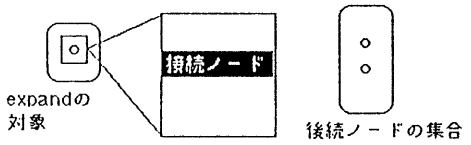

（3）expandサブルーチンの定義の開始

これはサブルーチンの内部から見た情景である。expandの対象集合を宣言し、サブルーチンの内部では無視されるオブジェクトを消して、サブルーチンの開始場面として宣言する。



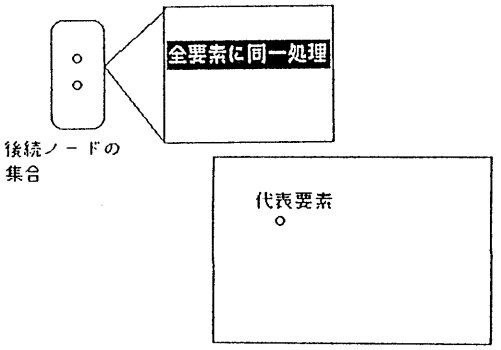
（４）後続ノードの集合を生成する操作

ネットワーク構造を構成しているノードに、もともと後続ノード集合を生成する操作が付随しているものと考えた。

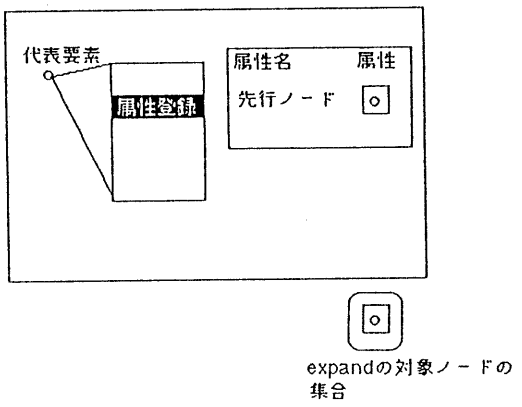



（５）開始ノードに遡る経路情報の張り付け

まず、後続ノードの集合の各要素に同一操作を実行する事を宣言すると、代表要素のイメージが画面に現れる。

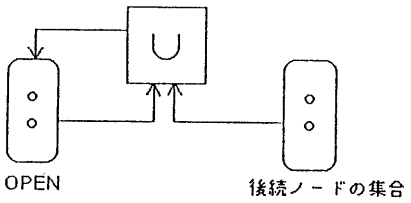



次に代表要素に付随する操作のメニューから、”属性登録”を選択すると、属性表が現れるので、expandの対象ノードを属性として張り付ける。



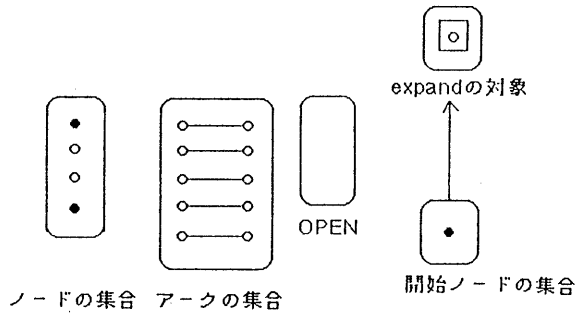

（６）後続ノードをOPENに加える操作

集合操作のパレットの中から和集合を作る操作を取り出して適用する。



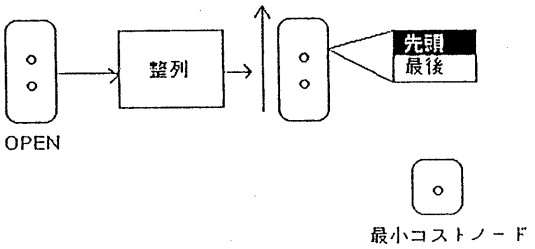

（７）開始ノードをexpandする操作

expandサブルーチンのアイコンをクリックする事によりサブルーチンを呼び出すと、expandの対象集合を尋ねて来るので、開始ノード集合を指定する。



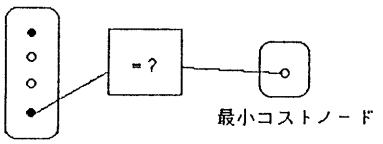

（８）OPENの中の最少コストノードの選別

集合操作のパレットの中から順序集合を作る整列操作を取り出して、適用する。整列操作のアイコンをクリックすると、順序の判断方法のサブ画面が現れる。そのサブ画面の中で整列のキーとなる属性を指定する。



（９）除いたノードとゴールの同一性判定

集合操作のパレットの中から同一性判定操作のアイコンを取り出して適用する。そうするとシステムから「真偽値を出力するか、判断分岐するか」と問い合わせてくる。




（１０）繰り返し操作

判断分岐を選択すると、真理値に応じたジャンプ先を順番にシステムが聞いてくるので、目的の場面をまず作る。同一でない場合には、最小コストノードを新しくexpandの対象集合の要素とし、ビデオのリワインド風にexpandサブルーチンを呼び出す場面に戻る。（HyperCardのリンクボタンの設定操作が参考になる）。

５．考察

５．１　プログラムの分かりやすさ

本手法を適用した場合のプログラムの分かりやすさなどについて評価する。

（１）操作の可視化

実際のプログラミングでは各種の基本的な操作や対象固有の操作も可視化する事が望まれるであろう。そうするとアイコンの種類が増えるため、図柄が紛らわしくなりアイコンの意味が判然としない場合も多いと予想される。そのため各アイコンに自分を解説する機能を付加する事が必須と思われる。

（２）操作の対象の可視化

上記の適用例では、グラフが対象でありながら、直接の操作の対象は集合であり、グラフという数学的対象物を可視化するに至らなかった。対象間の関係まで踏み込んで可視化すれば、一層分かりやすくなったと思われる。しかし、そうすると対象のクラスごとに様々な表現方法が必要となり、「誰が可視化するのか」という問題が発生する。各アプリケーションやライブラリの開発者がそれぞれの問題領域の可視化を行うというアプローチが解決策の一つであろう。

（３）処理手順の分かりやすさ

本手法ではプログラムの実行にともなう状況がビジュアルに表現されるため、テキストベースのプログラミングに比べて読みやすくなると予想される。

（４）関数呼出の分かりやすさ

関数あるいはサブルーチンの引数を数学的対象のイメージのままで表示できるため、引数の意味が分かりやすくなると予想される。

（５）編集の容易さ

本手法では個々のオブジェクト固有の操作の選択はオブジェクト（の属するクラス）に付属するメニューから行える様にして編集の便宜を図っている。しかし、オブジェクトや操作のアイコンの間を線で繋ぐ操作を利用者に任せずに、システムが自動的に行う様に工夫すべきであったと反省している。テキストベースのプログラミングでも各種のエディタが利用でき、今後はハイパーテキスト機能を持ったエディターなども出てくると思われる（たとえば、関数の呼出文をクリックするとその関数の編集画面が開くなど）。そのため、一層の編集機能を備えなければビジュアルプログラミングが利用者の支持を得る事は難かしいと思われる。

（６）新クラスの構成

既与のクラスおよび操作のみを組み合わせてプログラムを定義する事を考えたが、今後は可視化したオブジェクトを組み合わせて新しいクラスを定義したり、新しいクラスの定義と同時にそれを可視化する事ができる様なプログラミング環境を検討して行きたい。

５．２　他の手法との比較

ビジュアルプログラミングには種々雑多なアプローチがある。その中のいくつかについて本稿で提案した手法との違いに着目しながら紹介したい。

（１）HI-VISUAL[3]

HI-VISUALはデータフロー図風にプログラムを

定義するビジュアルプログラミング手法であり、アイコンの重ね合わせでプログラムを定義していくやり方を採っている。特に操作を表すアイコンがなく、関係する２つのオブジェクトの一方に能動的な役割を与えて暗黙に操作を示す点に特徴がある。また生成されるオブジェクトとの間を線で結ぶ作業は自動的に行われる。

（２）Programing by Rehearsal[6]

Programing by Rehearsalはいくつかの「劇団」から選抜した「役者」たちに「舞台」上での演技を教える事によって「作品」を作るシステムである。「キュー」を送られた「役者」が何をしたらよいかを作者がして見せる事によってプログラムコードが書かれて行く。「舞台」そのものを新しい「役者」として登録して、部品化する事もできる。舞台裏の「役者」も含めたさまざまな配役の「役者」の間で「キュー」を送り合う事により、複雑なアルゴリズムも構成可能である。

（３）Think Pad[7]

Think Padは、関数の処理対象のデータの変化を利用者がコーチする事により手続きが定義されていくシステムである。木構造などの抽象的な対象を表現するデータ構造のレベルで可視化する手法であるが、状況の進展の表現方法に本敲で提案した手法と似ている面もある。

## ６．現実のプログラミングへの活用

### ６．１　活用可能な領域

ここで提案した手法が活用できそうな問題領域について考察する。

（１）数値処理

時系列データの解析などでは、入力データを順序集合と見なしたり、集合の要素をｎ次元空間の点と見なす事により、本手法が適用できる。

（２）画像処理

画素の配列を順序集合と見なす事により、本手法の適用が可能と思われる。また画像処理の結果として抽出された線分や領域などの高次の対象物を可視化する事により、プログラムの理解しやすさが増すと思われる。

（３）アプリケーション間連携

データ種別に対応した複数のアプリケーションを組み合わせてシステムを開発する技術が提案されている[10]。このような技法の適用例としてデータベースから検索したデータに関連する画像を画面の指定した位置に表示させ、利用者が選択した画像に関連したビデオの再生を行うシステムをあげる。この場合、ＧＵＩアプリケーション、データベースアプリケーション、静止画アプリケーション、およびビデオアプリケーションを組み合わせて目的のシステムを構築する方法が考えられる。データベースから検索したデータの集合、そのデータに関連する画像の集合、画像をメニューとして配列するサブルーチンなどを、本稿で提案した手法で可視化する事により、アプリケーション間連携でのシステム開発を効率化できると思われる。

### ６．２　活用できない領域

一方、本手法が効果的でない領域も多い。

（１）大規模なアプリケーションシステムなどの開発では、個人的なプログラム開発と異なり、以下のように高度な作業内容と緻密な工程管理が必要とされている。

- 実世界の分析
- 要求の選定
- 利用できる資源の範囲の決定
- 外部とのプロトコールの決定
- プロセスやストレージなどの配置の決定
- プロセスやデータのインプリメンテーション
- システムの移行
- 環境変化への適応

このため、アルゴリズム開発手法が重要な位置を占める事はない。

（２）通信制御ソフトの開発では状態遷移ダイアグラムまたは状態遷移表のエデイターとシミュレーターあるいは自動検証システムが有効である。

（３）小規模な事務システムの開発では、伝票のイメージでデータ構成を定義したり、フォルダーのアイコンを使ってデータフローを定義する技法が効果的であると考えられる。

（４）シミュレーションを手軽に行うためには、オブジェクト間のイベント送受信関係をビジュアルに定義する技法などが効果的であると考えられる。

## 7．まとめ

本稿では、プログラミングの内容の理解を助ける手法として、アルゴリズムの対象物をビジュアルに表現し、手順の流れを絵コンテ風に表現する手法を考案し、最短経路選択アルゴリズムの定義を例題として、本手法の適用可能性を確認した。

このようなプログラム定義手法を活用できる問題領域のタイプを見極める事が、今後の課題として残されている。そのため、現実のプログラミング課題に適用して、実用面での効果の有無を確認したいと考えている。その中で、対象の関係の表現にまで踏み込んで一層わかりやすい表現を追求したい。

## 参考文献


[1]原田実:ソフトウエア生産性ツール（下）－オートメーション方式編,日経コンピュータ,No.65,pp.175-199(1984,3,19).

[2]Nilsson,N.J.:Probrem-Solving Methods in Artificial Intelligence, New York: McGraw-Hill,1971.

[3]Hirakawa,M.,Tanaka,M. and Ichikawa,T.: An Iconic Programing System,HI-VISUAL, IEEE Trans. Software Eng.,Vol.16,No.10, pp.1178-1184(1990).

[4]Glinert,E. P. and Tanimoto,S. L.:Pict: An Interactive Graphical Programing Environ-ment," in IEEE Comp.,Vol.17,No.11, pp.7-25(1984)

[5]Smith,D. C.:Pygmalion:A Creative Programing Environment, PhD dissertion,Dept. of Computer Science,Stanford University (tech.report STAN-CS-75-499),1975.

[6]Finzer,W. and Gould,L. :Programing by Rehearsal,Byte,Vol.9,no.6,June 1984,pp. 187-210.

[7]Rubin,R. V.,Golin,E. J. and Reiss,S. P.: ThinkPad:A Graphical System for Programing by Demonstration,IEEE SOFTWARE,pp. 73-78(1985).

[8]Edel,M.:The Tinkertoy GraphicalPrograming Environment,IEEE Trans. on Softw. Eng.,Vol.14,No.8, pp.1110-1115(1988).

[9]:電総研,"ビジュアル・プログラミング"可能なオブジェクト指向言語を開発,日経ＡＩ,1991.2.11.

[10]:Ｗｉｎｄｏｗｓアプリケーション間連携機構ＯＬＥ,日経バイト,No.96,1992.2,pp.239-266