ソフトウェア工学 97－3
（1994．3．1）

GUIの絵コンテ式ビジュアルプログラミング

小澤 正樹
東京電力（株）システム研究所

過去に作成したプログラムを再利用しようとする場合には、プログラミングに精通していても内容の理解に苦労する事が多い。そのような事から理解しやすくプログラムを記述できることを目標として、

・処理対象をビジュアルに表現する方法
・映画のカメラ台本の様にプログラムの処理手順を記述する絵コンテ方式
・ビジュアルな操作でプログラムを組立てるプログラマーインタフェース
・GUI部品への絵コンテの組み込み

について検討した。処理対象のビジュアル表示によれば、データの参照、更新が理解しやすくなる。絵コンテ方式によれば、データフロー形式で表現しきれない複雑な制御構造も表現できる。 プログラム部品のビジュアル表現方法を定義するためのツールの検討が今後の課題である。

Continuity Style Visual Programing for GUI

Ozawa Masaki
Computer & Comunications R & D Center
Tokyo Electric Power Company

When you are going to reuse programs that are developed in the past, it may be troublesom to understand them. Aiming more undesrtandable definition of programs

·Visual expression of objects
·Continuity style description of computation process
·Programer interface for programing through visual operation
·Assembling the Continuity style description into GUI objects

are researched. Visual expression of objects makes it easy to understand reference or modification of them. Continuity style description makes it easy to define a complex algorithm which is difficult to define in data-flow style. Preparation of tools for definition of visual expression of each objects remainds for future research.

まえがき

本システムは業務処理システムを開発する人、あるいは研究のためにプログラミングを行う人々を対象としている。プログラミングに精通していても、他人や自分が過去に作成したプログラムを再利用しようとする場合には内容の理解に苦労する事が多い。そのような事から本システムでは、理解しやすくプログラムを記述できることを目標として

・処理対象をビジュアルに表現する方法
・映画のカメラ台本の様にプログラムの処理手順を記述する絵コンテ方式
・ビジュアルな操作でプログラムを組み立てるためのユーザ（プログラマー）インタフェース
・ＧＵＩ部品への絵コンテの組み込み

について検討した。

処理対象のビジュアル表現とは、例えばプログラムの中で表構造のデータの要素を指定する場合に、表の形の図を用いて表現できるなど、ビジュアルな表現方法で処理対象を記述する方法である。

絵コンテとは本来、映画のカメラ台本の事であり、映画の画面イメージのスケッチを並べたものである。ここでいう絵コンテ方式とは、プログラムの処理の各ステップを表すイメージを絵コンテの様に並べて処理手順を表現する方法である。

ビジュアルプログラミングを行うプログラマの便宜のために、ＧＵＩ技術を駆使したユーザインタフェースを検討している。

また、目的プログラムとユーザとの会話処理手順もビジュアルにプログラミングできる様にするため、ＧＵＩ部品への絵コンテの組み込みについても検討している。

以下に、これらの概要について述べる。

１．プログラム部品の種類

プログラムの処理対象を表現するために利用するプログラム部品について以下の様に整理を行った。これらをすべて、ビジュアルに表現する事が本研究の目標である。

（１）抽象データ型部品

数値変数やテキスト変数、あるいは配列や木構造などを表す部品である。

（２）サーバーインタフェース部品

プリンターサーバーやデータベースサーバーなどを利用するための部品である。

（３）データー部品

抽象データ型部品と、その情報をファイルに入出力する機能を組み合わせた部品である。

（４）永久オブジェクト部品

オブジェクトＤＢに格納されたオブジェクトを表す部品である。

（５）ＧＵＩ部品

データを表示するイメージ形成機能、ウインドウ部品およびユーザアクションに対する処理手順を一体化した部品であり、データの表示を簡易にする場合に用いる。

（６）イメージ形成部品

データを画面表示する機能部品は、通常ウインドウ部品とイメージ形成機能を組み合わせて部品化するＭＶＣモデルが用いられるが、ここでは部品の再利用性向上をねらいに、ウインドウ部品からイメージ形成機能を分離した部品を定義している。

（７）状態遷移部品

永久オブジェクトが実世界の変化を反映するための状態遷移を記述するための部品である。直感的に理解しやすいＪＳＤ[1]の状態遷移図法を用いる。

２．プログラマーインタフェースの概要

ビジュアルプログラミングのユーザ（プログラマー）インタフェース画面は、次により構成する。

（１）絵コンテ編集画面
（２）ＵＩ（ユーザインタフェース）編集画面
（３）部品検索画面

絵コンテ編集画面は、文字通り絵コンテを編集する画面である。絵コンテが複数の場面から構成される場合はスクロールする事ができる。

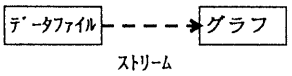
ＵＩ編集画面は目的プログラム実行時のユーザインタフェースを編集するための画面であり、この上でウインドウやメニューなどのＧＵＩ部品の配置の調整や、部品の結合を行う。

部品検索画面は、分類体系にそって部品を検索する画面であり、部品の使用例（サンプルプログラム）をたやすく見られる。

３．絵コンテ編集画面

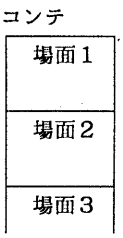
３．１　絵コンテ方式のねらい

C言語のfor文やwhile文に相当する処理はストリームのデータフローで記述可能である。例えば株価などの時系列データをグラフ化する処理は次の様に書ける。



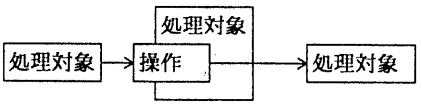
しかしデータフロー形式ではでは記述が困難なアルゴリズムもあるため、絵コンテ方式を考案した。

絵コンテは「場面」の連鎖である。「場面」には「データフロー場面」と「if場面」、「for場面」や「while場面」があり、それらを組み合わせて処理手順を組んでいく。



３．２　データフロー場面

データフロー場面では、処理対象を表すシンボルと、処理対象に対する操作を組み合わせて、データフロー形式で処理手順を記述する。例えば、「ある処理対象が持つ操作に別の処理対象を入力した結果」は、下の様なデータフロー図で表現する。



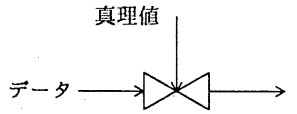
操作を表すシンボルの検索は、処理対象を表す部品が表示するメニューから行う。

３．３　データフロー形式でのアルゴリズム表現

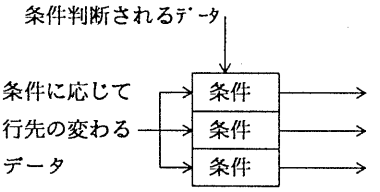
データフロー図的な表現方法でも、ある程度のアルゴリズム記述は可能である。（SynchroWorks[3]などに実現例を見る事ができる）

本システムではデータフローを配管に見立てるアイデアにより、「バルブ」アイコンによってif文の機能を表現し、「バルブ」の集合である「スイッチ」アイコンでcase文の機能を表現する事とした。

・バルブによるデータフローのコントロール



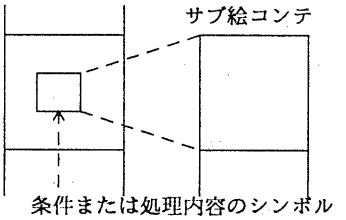
・スイッチによるデータフローのディスパッチ



３．４　if場面、while場面、for場面

「if場面」、「while場面」や「for場面」により通常のプログラミング言語と同様な記述が可能となる。

while場面やfor場面で参照される分岐条件や処理の内容はサブ絵コンテで表現する。



４．絵コンテの処理対象のビジュアル表現

４．１　処理対象のビジュアル表現の目的

プログラミングは一般に次の５ステップで行われる。

（１）問題のスケッチ

問題を頭の中や紙の上でスケッチし、解決手段の試行を行う。

（２）数学的対象への置き換え

問題とその解決手段を集合などの数学的対象物へ置き換える。

（３）アルゴリズムの検証

何らかの形式的手段でアルゴリズムの検証を行う。デバッグ段階に泥縄式に行う事も多い。

（４）効率化

対象物に関する表現方法を工夫して、記憶量や計算量を削減する。

（5）インプリメンテーション

プログラミング言語や周辺機器制御言語などを用いて機械で実行可能なプログラムの組立を行う。

数学的な対象物への置き換えのステップまでは図や表を用いて行われる事が多い。

アルゴリズムの検証は、時には図を用いて行われ、時には記号や式を用いて行われる。

しかし、効率化やインプリメンテーションは完全にテキストベースの作業であるため、図や表の世界からテキストの世界への翻訳を強いられる事が多く、プログラムの意味の理解が困難な理由となっている。従って、インプリメテーション作業を、それより前の段階の様にビジュアルに表現できればプログラムの理解が容易となり、再利用も楽になる事が期待される。また、新規に作成する場合にも手順の検証が容易になると思われる。

4．2　ビジュアル表現の範囲

（1）物理的構造の表現

行列処理のMPL[4]や、リスト処理のLIVE[5]など処理対象の物理的な構造をビジュアルに表現するビジュアルプログラミングの研究が行われているが、本システムでは、画像など2次元のデータやグラフ構造のビジュアル表現方法を検討した。

しかし、処理対象の物理的構造のビジュアル表現の考えを一般の処理対象に広げると、

- 対象物は名前、年齢、体重など絵にならない属性を多く持っている。
- 一つの絵は非常に多くを同時に物語れるため、逆に特定の意味を示す力が弱い。
- 「排他性」や「整列」のような操作的概念を、絵で表現する事は困難である。

などの理由から表現が難しいため、範囲を限定して検討した。

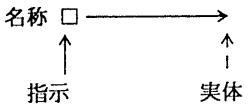
（2）論理的構造の表現

木構造を表すチャートや2次元の表が、例えば組織図や性能比較表など、日常的に広く利用されていることから、論理的あるいは抽象的な関係構造の場合には、一般の処理対象に対してもビジュアル表現が有効であると考える。

4．3　処理対象の論理的構造のビジュアル表現方法

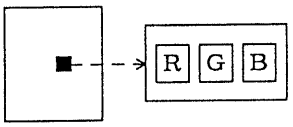
（1）指示と実体

「指示」はプログラム変数に該当し、「実体」はその変数に束縛されたオブジェクトに該当する。指示・被指示関係をビジュアル表現するには、指示を表すシンボルと実体を表すシンボルを線で結べば良い。指示を表すシンボルには通常「名称」が記述される。



（2）オブジェクトとその属性

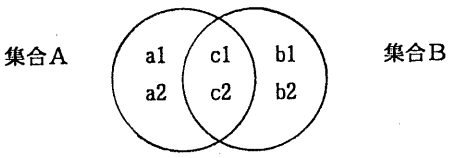
オブジェクトのシンボルと、その属性を表すシンボルのグループを同時に表示してそれらを線で結ぶ。例えば、画像データの要素（画素）とその属性のRGB値の関係の表現は下の様になる。



（3）集合

集合の表現に広く使われているオイラー図を用いる。

2つの集合を表す図を重ねた上で、必要な部分をマウスでクリックして選択する事により、集合の和や積をビジュアルに表す事ができる。（選択した部分を強調表示するなど）



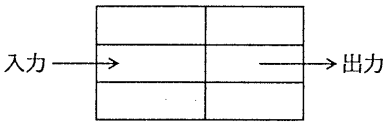
（4）2次元の表

関係DBの「表」の様に、同じ種類の物の属性値を並べた表の場合は、表の各列の上に属性名を表す文字列を付ける。

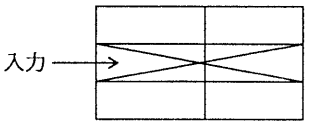
自治体名	人口

主キー列の値により検索や削除を行える。

・検索の場合　自治体名　人口



・削除の場合　自治体名　人口



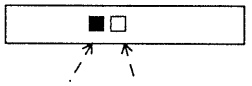
鉄道運賃表のようにオブジェクト間の関係の関数を示す表は、下の様に配列の積として表現できる。

	新橋	有楽町	東京
新橋			
有楽町			
東京			

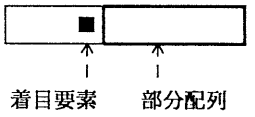
（5）配列

配列に関して、次の様な表現方法を考えた。

「ある要素の隣の要素」

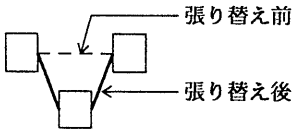


「ある要素の後ろの部分配列」



（6）リスト構造

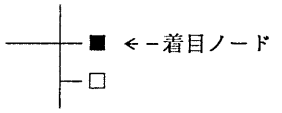
リストの張り替え操作も、ビジュアルに表現できる。



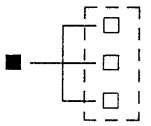
（7）木構造

木構造に関して次の様な表現方法を考えた。

「あるノードの隣のノード」



「あるノードの直属配下のノード集合」

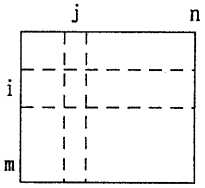


4．4　物理的構造のビジュアル表現

2次元空間構造やグラフ構造をもった対象物に関しては、物理的構造のビジュアル表現も有効である。

（1）2次元空間のデータ

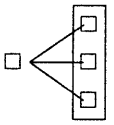
画像データの様な2次元空間のデータの場合、下の様に座標を表す値あるいは変数を書き添えた形の表現を行う。（この方式はMPL[4]による）



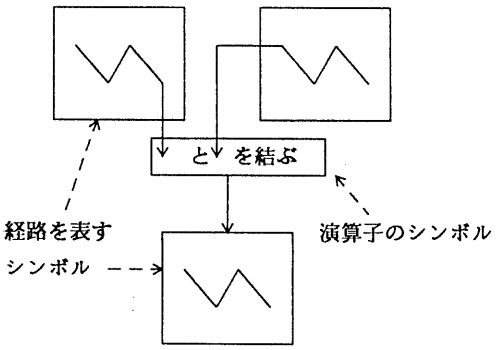
（2）グラフ構造

グラフ構造に関して次の様な表現方法を考えた。

「あるノードに隣接するノードの集合」



「2つの経路を接続する操作」



５．絵コンテ方式の評価

５．１　絵コンテ方式によるプログラム記述例

絵コンテ方式を例題へ適用して、C++言語で記述した場合との比較を行った。例題として採用したのは、

①ノードの各対を連絡するエッジは高々１つ

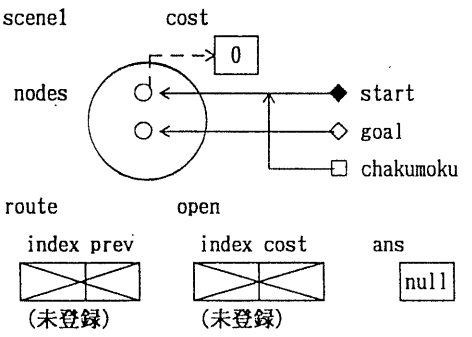
②経路を構成するエッジのコストの和が経路全体のコストに等しい

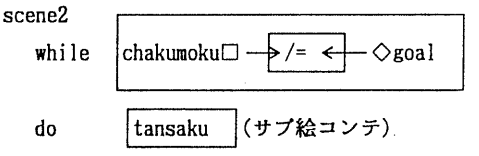
という条件でコスト最小の経路を見つけるアルゴリズムである。

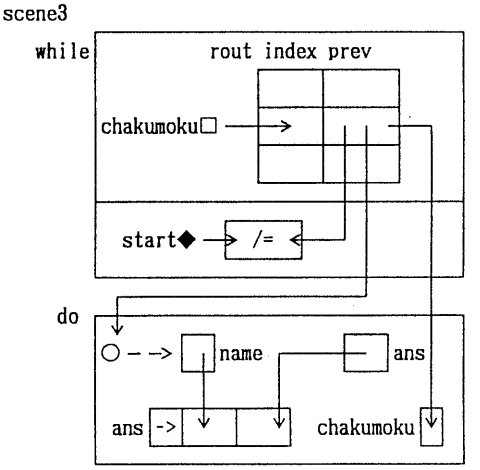
（１）C++言語で記述したプログラムの一部

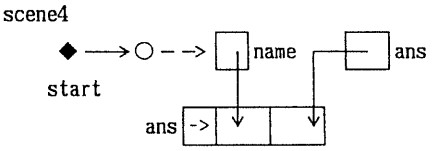
```
saitan(CString start,CString goal,CString *ans)
{static int  route[TabSize];
 static COST open[TabSize];
 int  i, startNo, goalNo, chakumoku, next;
 COST chakumokuCost;
 CNode *node;
   startNo = nodeNo(&start);
   goalNo  = nodeNo(&goal);
   chakumoku = nodeNo(&start);
   chakumokuCost = 0;
   for( i=0 ; i < size ; i++) open[i]=FALSE;
   while(! (goalNo == chakumoku) ){
      int nghbrSize = neighbors(chakumoku);
       //ｸﾞﾛｰﾊﾞﾙ変数nghbrsに隣接ﾉｰﾄﾞ番号が入る
      for(i=0; i < nghbrSize; i++){
         node = nghbrs[i];
         int x = node->num;
         COST cost = costTab[chakumoku][x] +
                     chakumokuCost;
         if((!open[x])||(cost<open[x])){
            open[x]=cost;
            route[x]=chakumoku;
      } }
      for(i=0; i< size ; i++)
         if(open[i]&&(open[i]>0)){next=i;break;}
      for(i=next; i< size ; i++)
       if(open[i]&&(open[i]<open[next]))next=i;
      chakumoku = next;
      chakumokuCost = open[next];
      open[next] = FALSE;
   }
   while(! (chakumoku == startNo) ){
      node=nodes[chakumoku];
      *ans = "->" + (node->name) + *ans;
      chakumoku = route[chakumoku]; }
   node = nodes[startNo];
   *ans = (node->name) + *ans;
}
```

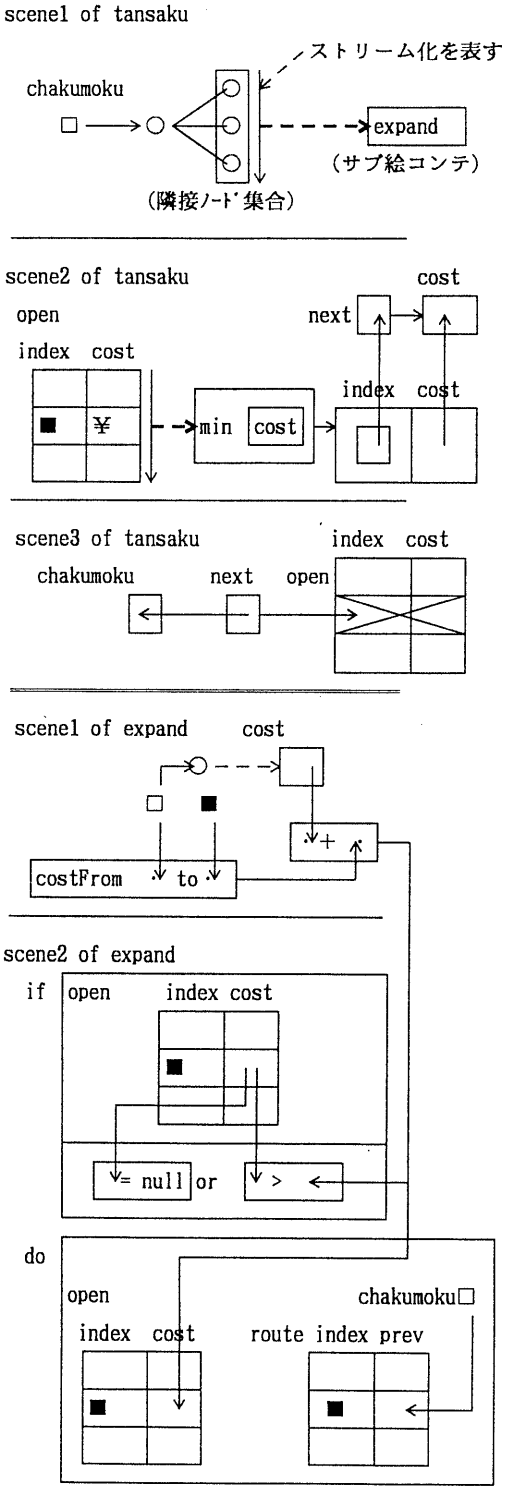
（２）絵コンテで記述した場合











5．2　絵コンテ方式の効果

絵コンテ方式を適用する事により次の効果が期待できる。

（1）絵コンテ方式によれば、データフロー形式で表現しきれない複雑な制御構造も表現できる。

（2）絵コンテの各場面をデータフロー形式にする事により関連する変数やオブジェクト、および操作を1つのシーンにまとめて一望する事ができる。また、関数の意味も把握しやすくなる。

（3）データフローや絵コンテの階層構造をビジュアルに記述できるためプログラムの編集が容易になる。

（4）配列や表の内部構造を図示できる事からデータの参照、更新が理解しやすくなる。

6．UI編集画面でのビジュアル表現

UI編集画面は目的プログラムのUI画面の定義作業を行うための画面である。この編集画面の上で、GUI部品の配置や形の調整、および他の部品との結合関係の定義を行う。

（1）GUI部品への絵コンテ組み込み

本システムではGUI部品に対して絵コンテ方式のビジュアルプログラミングを直接的に行う事ができる。

GUI部品に絵コンテを組み込むための編集画面は次の手順で呼び出す。

①UI編集画面上のGUI部品を選択する。

②メニューから絵コンテ編集を選択する。

③GUI部品に対してマウスクリックなどのユーザアクションを模擬的に行うと絵コンテ編集画面が表示される。

絵コンテの中には、ユーザアクションを表すシンボルが表示される。そのシンボルをクリックするとマウス位置などの属性を表すシンボルが表示される。

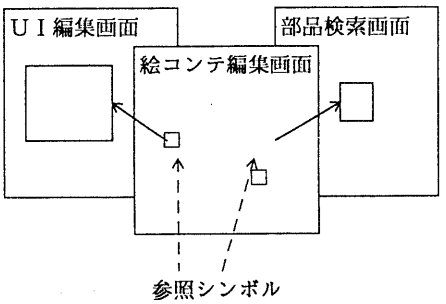
（2）絵コンテ外のオブジェクトの参照

UI編集画面上の他のGUI部品や、部品検索画面上のデータ部品などの絵コンテ外オブジェクトと、絵コンテ内のオブジェクトの間の入出力を、次の様な手順により定義する。

①まず絵コンテ編集画面に「汎用参照シンボル」を置く。

②次にそこから参照したい部品に向かってマウスを移動すれば、参照シンボルが部品独自のシンボル

に変身する。



（３）ＧＵＩ部品間の結合関係の可視化

ＧＵＩ部品間の結合関係を表す結線を、ＵＩ編集画面上でＧＵＩ部品に重ね合わせて表示し、表示された結線の中から一つを選択すると、関係する絵コンテ編集画面が開く事ができる。

（４）画面遷移

ビジュアルに、かつ直接的に画面遷移を記述するために、「画面切り替え部品」を定義した。絵コンテの中で、画面切り替え部品に信号を送る事で画面遷移を起動する。画面切り替え部品と遷移先の画面の結合をビジュアルに定義できる事は言うまでもない。

７．今後の課題

本システムの様に、部品独自の表現方法を用いる方式の場合は、部品のビジュアル表現方法を定義するためのツールの充実が特に重要である。

実用化のためには、デバッグ支援機能や、部品検索機能、部品の利用方法や内部構造の解説機能などの充実も必要である。

また、これと並行して、ＯＡなど実際の応用に役立つ部品ライブラリの検討を進める事も必要である。

参考文献

[1]Jackson,M.A.: System Development, Prentice Hall International,1983

[2]Tanaka,Y.and Imataki,T.: Intelligentpad: A HYPERMEDIA SYSTEM ALLOWING FUNCTIONAL COMPOSITIONS OF ACTIVE MEDIA OBJECTS THROUGH DIRECT MANIPURATIONS, Proc of the IFIP 11th World Computer Congress,1989,pp.541-546

[3]橋本仁: ｵﾌﾞｼﾞｪｸﾄ指向をﾍﾞｰｽにしたﾋﾞｼﾞｭｱﾙ･ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ環境「ｼﾝｸﾛﾜｰｸｽ」,日経ｲﾝﾃﾘｼﾞｪﾝﾄｼｽﾃﾑ別冊, 1992春,pp.42-55

[4]Yeung,R.: MPL - A Graphical Programming Environment for Matrix Processing Based on Logic and Constraints,IEEE Proceedings, Workshop on Visual Languages,1988,pp.137-143

[5]二木誠司,松田芳樹,藤原ひろみ:構造化ｵﾌﾞｼﾞｪｸﾄの視覚的ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ, 情報処理学会研究報告 Vol.93,No.11,93-PRG-10(1993),pp.9-16

[6]萩谷昌己:視覚的ﾌﾟﾛｸﾞﾗﾐﾝｸﾞと自動ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ, ｺﾝﾋﾟｭｰﾀｿﾌﾄｳｴｱ,Vol.8,No.2(1991),pp.27-39

[7]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).

[8]杉本明,北村操代,中田秀男,川岸元彦,小島泰三: 対話型ｼｽﾃﾑ視覚的構築用ｸﾗｽﾗｲﾌﾞﾗﾘ:GhostHouse(I) --設計方針と概要--,情報処理学会第46回全国大会

[9]小澤正樹:絵コンテ式のﾋﾞｼﾞｭｱﾙﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,情報処理学会研究報告,Vol.92,No.16,92-SE-84(1992)