1F−3

構造化データフローモデルによる視覚的プログラミング*

坂庭剛史†　佐渡一広‡

群馬大学工学部§

1　はじめに

ユーザによるプログラムやデータの記述とその運用において、一貫した操作と視覚的表現を用いるような視覚的プログラミング環境を構築した。構造化データフローモデルを基にして、その構成要素であるモジュールの入出力に処理結果の入出力と内部情報そのものの入出力という二種類の入出力機能を与えることによって、インターフェイス画面を含むプログラムや使われるデータの作成、実行、修正などのユーザがシステムに求める操作全てをモジュールとして視覚的に表現し、その対話的な操作でシステムの全ての機能を表した。

2　モジュールの機能と構造

モジュールはデータや処理、それらの集合、処理の手順を表すための構成要素である。ユーザはプログラムやデータの作成、その管理などの操作全てをモジュールの複製や、入出力関係を指定(リンク)することで行なう。

モジュールは視覚的表現と、行なう処理(またはデータそのもの)や自分の表示状態を示す内部情報とを持ち、その情報そのものの入出力機能を持つ。視覚的表現は自分を識別するためのアイコンと内部における処理やデータを表示するウィンドウで構成される。処理は視覚化されないコードか、他のモジュールのグラフや集合からなる構造であり、前者はプリミティブな処理モジュールとしてあらかじめ与えられる。

プリミティブモジュールには他にも何も情報を持たない空モジュールやデータそのものを表すデータモジュールがある。空モジュールは新しいプログラムを定義するための作業領域や、モジュール群を管理し、グループとして保存するためなどに用いられる。

図1は簡単な計算処理の例である。モジュール“?”の内部にあるモジュール“$\sqrt{D}$”の内部情報は構造化された他の処理モジュールのグラフによってできている。“+”、“−”、“÷”で表される処理モジュールは入力として得られたテキストデータが数字であれば計算をして結果をテキストデータとして出力するものであり、その他の数字やアルファベットはテキストデータモジュールである。図において、モジュール“?”のみでは入力のテキストデータを全て数字に書き換えない限りデータは流れず、実行は行なわれない。

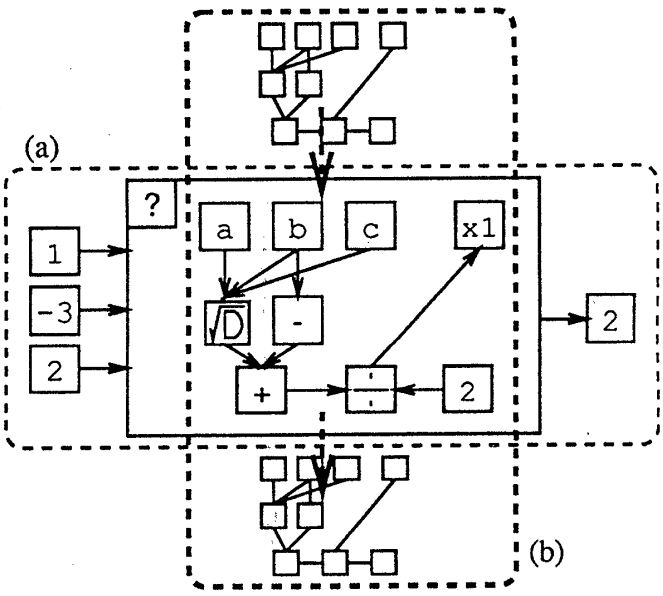

図1: 簡単な計算処理の例

処理モジュールは内部処理に与える入力データの入力や処理結果の出力(a)と内部の処理構造の入出力(b)の二種類の入出力機能を持っている。(a)はモジュール“?”の外部に入力モジュール群と出力モジュールが接続され

*Visual Programming Based on Structured Data Flow Model

†Takeshi Sakaniwa

‡Kazuhiro Sado

§Department of Computer Science, Gunma University

る場合を表している。この場合は入力のテキストデータ1、-3、2が、それぞれ内部の入力用データの代わりに用いられ、結果が計算される。結果は外部出力モジュールの中に代入される。

この場合内部のテキストデータは変化しない。外部入力が全て与えられない時は必要な部分は内部のデータが用いられるため、内部のデータはデフォルトの値として使う。

また、モジュール“?”の内部構造の表示ウィンドウを変形して、上部の入出力データモジュール群のみを表示することによって余分な計算部分を隠し、更にコメントを加えることによって、飾られたインターフェイス画面を作れる。

モジュールの間の接続は図では同じ矢印で表されているが、実際には細かいパラメータがあり、それが設定されることによって様々な種類の接続を行なう。例えば、内部のテキストデータモジュール“b”から処理モジュール“−”への接続は「テキストデータモジュールの内部情報(=b)を処理モジュール“−”の第2入力データとして減算処理に送る」という意味の選択をユーザと対話的に行なう。

図1(b)は内部構造そのものの入出力を示している。例えば一部を変更した別のプログラムを作るために複製するとか、プログラムの構造をプリンタに出力するなどの、「作成したプログラム自身に対しての処理を行なう」ために必要である。ユーザはこれを直接実行するか、間接的に記述を行なうかのどちらかをリンクの際に指定できる。

3 ユーザによるモジュールの操作

全てのデータ、処理がモジュールという統一した表現で表されており、ほとんどの記述や実行はモジュールの複製か関係の記述によって行なわれる。関係の記述はモジュールの出力を他のモジュールの入力に接続するものだが、接続が記述として残るか、一度実行されて切断されるかを選択できる。操作手順としてはほとんど同じだが、前者はプログラムの記述を行ない、後者は入力がプログラムかデータかで処理コマンドやプログラムの実行をそれぞれ行なう。

4 部品としてのモジュールの利用

一度記述されたモジュールは他のモジュールの部品として使える。図2はモジュール“?”に与えられる入力データの値の一つをマウスを使って対話的に様々な値を入力し、その入力データに対する計算結果をグラフの形に変換して座標上の点で示し、その軌跡を残すプログラムの例である。画面上部にあるモジュールは内部のボックスをマウスで移動することによって位置に対応する数字を出力する。

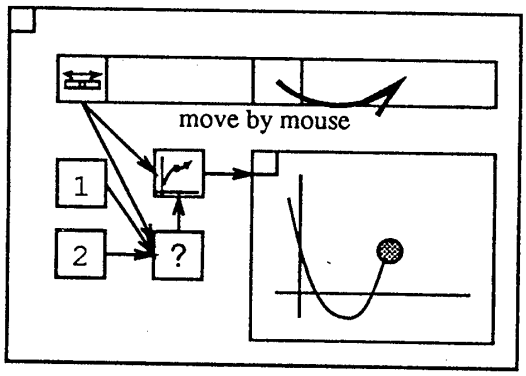

図2: 応用例

ボックスは実際には空モジュールであり、そのX座標をテキストデータとして出力することでマウスによる数値入力を実現している。変化した値はモジュール“?”に伝えられ、計算結果がグラフ化されて表示される。また、他のデータ(1や2)をユーザの指示によって変更することもできる。

5 おわりに

モジュールの入出力に位置、大きさなどを含む内部構造の入出力を採り入れることにより、処理手順、ユーザインターフェイス画面を含むデータの全ての記述と運用において、一貫した操作性と視覚的表現を実現した。

今後の課題として、高度な操作環境を提供するために、ある程度アプリケーション分野を制限した上で、用いられる高レベルデータ構造と操作モジュールを構築し、その表現や操作性を評価することが挙げられる。

また、ユーザの操作を簡単にするために直接的な操作の種類を少なくしたが、その反面、リンクの手順が複雑になった。直接操作と間接操作との融合をどのようにはかるかが問題である。