# CafePie:プログラムビューのカスタマイズ機能によるプログラム実行の視覚化

# CafePie: Visualization of Program Execution with Customized Program View

小川 徹[†] 田中 二郎[††]
Tohru OGAWA Jiro TANAKA

†筑波大学 博士課程 工学研究科
Doctoral Program in Engineering, University of Tsukuba
††筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

**概 要**

CafePie は，代数的仕様記述言語 CafeOBJ のためのビジュアルプログラミングシステムである．ユーザはドラッグ&ドロップ操作を用いてグラフィカルにプログラム編集を行う．システムによるプログラムビューがユーザの意図と合わない場合，プログラムモデルの理解が困難になる．本報告では，この解決のためにプログラムビューをカスタマイズ可能な手法を提案する．また，カスタマイズされたビューを用いプログラム編集と実行表示が可能であることを示す．

## 1 はじめに

我々は，ビジュアルプログラミングシステム (VPS) である CafePie を提案し作成している [1] [2]. CafePie の特徴は，ユーザが持つプログラムイメージをシステムに側に効率良く伝える手段として有効な直接操作 [3] を重要視し，マウスによるドラッグ&ドロップという単純な枠組みだけで，全てのプログラムの編集・実行を行えることである．

ビジュアルプログラミング [4] とは，文字列の代わりにアイコンやアニメーションという人間がより理解しやすい図形的表現を用いて人とコンピュータとの相互作用を行う技術のことである．一般に，VPS はプログラムの提示に図形的表現を用い，ユーザのプログラムモデル理解を支援に役立てられる．しかし，システムによるプログラムビューがユーザの意図と合わない場合，プログラムモデルの理解が困難になる．

我々は，プログラムモデルをプログラムビューに反映させる仕組みが必要であると考え，プログラムビューの視覚化カスタマイズ手法を提案する．ユーザは図形の視覚的な意味をカスタマイズすることで自分の意図を図形に反映させることが可能となる．

## 2 関連研究

ToonTalk [5] は子供向けに開発された VPS であり，漫画のキャラクタとその動きでプログラムを表現される．このような独自の表現を用いるシステムは，多彩な提示が行なえる一方で，視覚化されたプログラムの変更が困難である．Visulan [6] は，ビットマップ書換えに基づくビジュアルプログラミング言語である．プログラム表現として扱うビットマップをパターン置換ルールの順序集合で記述する．VISPATCH [7] は図的なルールをもとに図形を書換えるビジュアル言語である．例として図形エディタなどがあり，ユーザやシステムによって発生

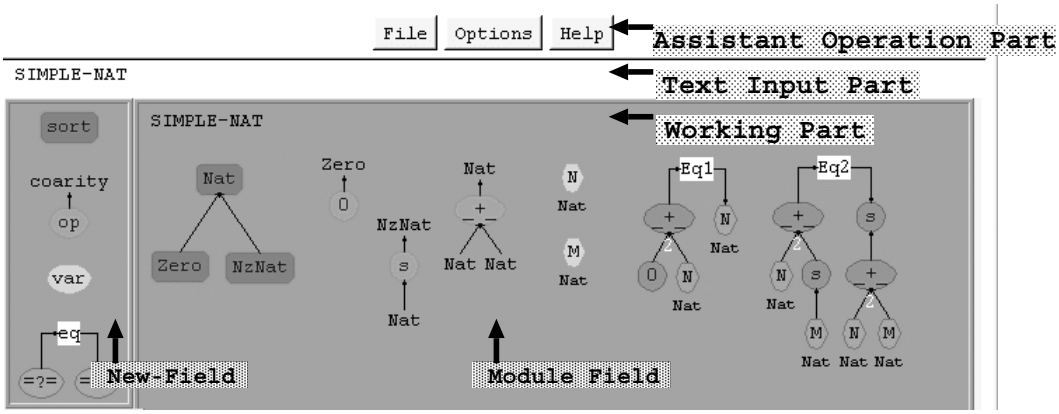


**図1　CafePieの実行画面**

されるイベント(マウスボタンを押すなど)によって書換えが制御できる．これらのシステムは独自のビジュアル的な言語体系を持っているため，視覚化された表現が同じであれば，全て同じ動作をする．また，PP [8] のように既存のテキストベースの処理系に寄生させその資産を有効に活用しようとしたシステムに比べて，一からシステムを構築する必要があり実現が困難である．既存の資産を利用するシステムは，グラフのような単純な表現を用いることで，比較的容易にテキストからビジュアル表現に変換することが可能である．

## 3　システムCafePie

CafePieは代数的仕様記述言語CafeOBJのためのVPSである．CafeOBJはモジュールベース，かつ，項書換え系による実行が可能な言語であり，その実行環境でもある．CafePieは当初JDK™1.0.2で開発し始め，現在は1.2に移行している．CafePieの実行画面を図1に示す．図中央の Module Field でプログラムは視覚化・編集され，実行される．その左側に新規アイコン作成用の New-Field がある．この2つをまとめて Working Part と呼ぶ．Working Part の上部に，アイコンのラベル変更に用いられるText Input Part と，(ファイル入出力などの補助的な作業を受け持つ)ボタン群から成るAssistant Operation Part がある．

### 3.1　データ構造とCafePie上での表現

CafePieでは，CafeOBJ言語のモジュールを視覚化し，編集・実行表示が行われる．モジュールに含まれる基本的な要素は，ソート・演算・変数・等式であり，視覚化の際には，特に，データ構造を表す**項**が重要になる．一般に，一階述語論理の言語において項は，変数，定数，関数(演算)とその引数が全て項であるもの，と定義される．定数は引数無し演算と見なせるので，項の構成要素は演算と変数となる． 演算はラベル付の楕円(水色)で表され

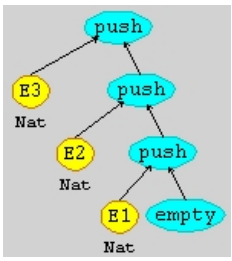


**図2　項の視覚化**

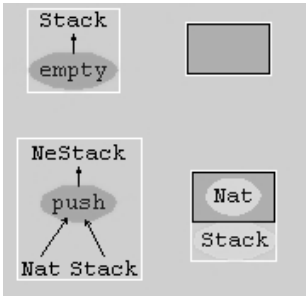


**図3　演算のカスタマイズ**

る．返却値は上方に(図3左上)，引数は下方(第1引数は最左端)に配置される(図3左下)．また，引数から演算の中心，演算の中心から返却値へ，と有向線が引かれる．一方，変数はラベル付の楕円(橙色)で表され，ソートはラベル下方に配置される．この視覚化により，項は木構造で表現される．図2はモジュールSATCK上の項，

```
push(E3, push(E2, push(E1, empty))).
```

を視覚化したものである．

## 4　プログラムビューのカスタマイズ機能

カスタマイズの対象は，プログラムビュー，即ち，データ構造を表現する項である．以下では，項の構成要素，演算をカスタマイズすることを考える．

### 4.1 前後画面ルール

カスタマイズされたビューの対応関係を表示するために，我々はビジュアル言語で良く使用される前後画面ルールを用いる．前後画面ルールとは，シミュレーションのためのプログラミングシステムであるKidSim [9] などで用いられており，書換え前と書換え後とを図示することで図形の書換えルールを示す手法である．左右に図形が配置された場合，左が書換え前，右が書換え後を示す．

演算の構成要素には主にラベル，引数と返却値がある．演算を項の一部としてみた場合，図2 に示すように演算のラベルと部分項が重要となる．返却値は型の照合に重要となるが常に表示する必要はないと考え，デフォルトでは表示を省略する．図3にSTACKにおける演算のカスタマイズ例を示す．ルール1(図3上)は，空を示す演算 empty を矩形にすることを示す．ルール2(図3下)は，要素を積む演算 push を要素をSTACKに積むことを示す．これにより図4 のようにSTACKを積木構造として視覚化できる．

### 4.2 カスタマイズされた表現における情報

カスタマイズされた表現には，演算への参照，構成する図形要素，各図形間の幾何的情報，という情報が含まれる．まず，演算を参照することで，実際のラベル名，引数や返却値の型(ソート)を保持する．また，使用できる図形の種類には，変数，矩形，丸型矩形，楕円，ユーザ定義がある．この中で変数は引数に対応しており，CafePieで用いる図形をそのまま用いる．カスタマイズされた表現は，それ以外の要素を用いて編集される．また，ユーザ定義とは多角形や星型などの任意の形をサポートしており，また画像などを呼ぶことができる．各図形には，形・サイズ・色・ラベルの情報が含まれている．また，図形間の幾何的情報は，現在のところ図形間の距離と方向を示すベクトル情報をそのまま保存している．このためレイアウトは固定になるが，ユーザが決めた定義をそのまま用いるため，配置が一意に決まり，システムはユーザの希望する場所へそのまま配置することが可能となる．

### 4.3 カスタマイズ機能の適用例

このカスタマイズ機能によって，一つのプログラムモデルに対し複数のビューを定義することが可能となる．例えば，積木構造(図4) の代わりにSTACKを人の並びと見なし，演算 empty を行き止まり，演算 push を列の最後に人が並ぶこと考える．STACKの要素を人の表情として他に定義することで，図5のような視覚化も可能である．


図4　視覚化

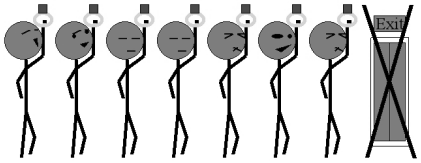


図5　視覚化(人の並び)

## 5 カスタマイズ表現を用いた編集と実行

### 5.1 編集における項のモデル化

我々は，演算への代入の概念に注目し，項の編集操作をモデル化した．項の編集は，引数への代入とその逆操作であると捉えることができる．カスタマイズ以前の木構造に対する項の編集は、

- 変数を作成する(項の初期化)．
- 変数の上に演算を重ね合わせる(項の追加)．このとき変数は演算によって置き換わる．もし，演算に引数があるならば，その引数はそれぞれ変数に置き換わる．
- 項にある演算のラベルを Module Field の枠外に移動させる(項の削除)．この演算より下の項(部分項)は全て削除され，変数に置き換わる．

という手順で行なわれる．この簡略化されたメカニズムは，変数を代入の対象とし，変数を設けることで木構造で視覚化されていなくても適用可能であるという利点がある．項がカスタマイズされた場合でも代入対象である変数を持つため，CafePieの持つドラッグ&ドロップ操作で容易に編集することが可能である． カスタマイズ表現を用いた項の編集は図

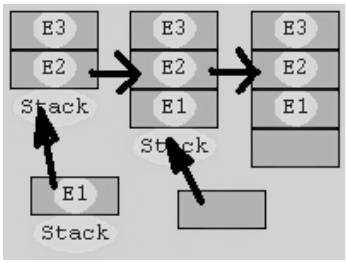


図6　カスタマイズ表現を用いた項の編集

6 のようになる．変数へのドラッグ&ドロップ操作を用いることで左から右にSTACKが編集される．

### 5.2 等式の編集

STACKにおいて演算 pop は，STACKから先頭の要素を取り除いたSTACKを返す，というSTACKの操作を表す演算である．CafeOBJ 言語において，操作，即ちプログラム動作を与えるのが等式である．等式は両辺に項を持ち，右辺項から左辺項への書換えルールとみなせる (図7)．

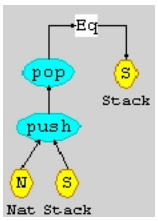


**図7　等式表現**

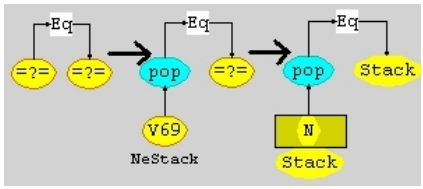


**図8　等式の編集**

図8に等式の編集過程を示す．左から右に行くにしたがって等式が作成されている．真中の図から右端の図において，カスタマイズされた演算を用いて編集されているのがわかる．このように，項の編集において以前の表現と互換性が保たれているため，カスタマイズした表現を使っても等式を編集することができる．

### 5.3 プログラム実行表示

CafePieにおける実行処理はCafeOBJ インタプリタを介して行われる．図9はSTACKの項，

```
pop(push(E1,push(E3,push(E2,pop(pop(
  push(E3,push(E4,push(E1,empty)))))))))
```

をゴールとして与えた実行結果を表示している．

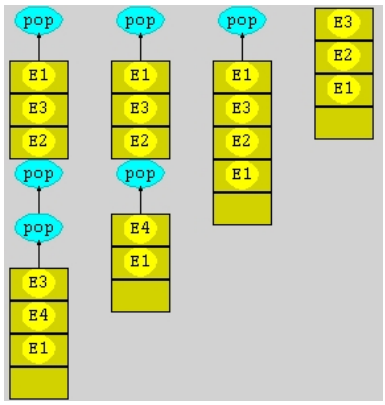


**図9　カスタマイズ表現を用いた実行表示**

ゴール(図9左端)はカスタマイズされた形でユーザが与える．項の構成要素である演算はそれぞれ対応する演算のモデルを所持している．このため，CafePieはカスタマイズされた表現からでもインタプリタが解釈可能なテキスト表現に変換することができる．インタプリタから結果を受け取るとシステムは表示を開始する．結果を受け取ると，CafePieはカスタマイズされた対応関係から判断して表示を行う．図9のように簡約が進み，結果として

```
push(E3,push(E2,push(E1,empty)))
```

を得る (図9右端)．

## 6 まとめ

プログラムモデルをプログラムビューに反映させる仕組みが必要であると考え，プログラムビューの視覚化カスタマイズ手法を提案した．また，本手法をCafePieに実装することで，カスタマイズされたビューを用いプログラム編集と実行表示が可能であることを示した．

## 参考文献


[1] T. Ogawa and J. Tanaka. Double-Click and Drag-and-Drop in Visual Programming Environment for CafeOBJ. In *Proceedings of International Symposium on Future Software Technology (ISFST'98)*, pages 155–160, Hangzhou, October 28-30 1998.

[2] T. Ogawa and J. Tanaka. Realistic Program Visualization in CafePie. In *Proceedings of the fifth World Conference on Integrated Design and Process Technology (IDPT'99)*, Dallas, Texas, June 4-8 2000. (CD-ROM), Copyright (c) 2000 by the Society for Design and Process Science, (SDPS), 8 pages.

[3] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.

[4] B. A. Myers. Taxonomies of Visual Programming and Programming Visualization. *Journal of Visual Languages and Computing*, 1(1):97–123, 1990.

[5] K. Kahn. ToonTalk(TM) – An Animated Programming Environment for Children. *Journal of Visual Languages and Computing*, June 1996.

[6] 山本 格也. ビットマップに基づくプログラミング言語 visulan. **インタラクティブシステムとソフトウェア** *III*, **日本ソフトウェア科学会** *WISS'95*, pages 151–160. 近代科学社, 1995.

[7] Y. Harada and K. Miyamoto. VISPATCH: Graphical rule-based language controlled by user event. In *Proceedings of the 1997 IEEE Symposium on Visual Languages*, pages 162–163, Capri, 1997.

[8] J. Tanaka. PP : Visual Programming System For Parallel Logic Programming Language GHC. *Parallel and Distributied Computing and Networks '97*, pages 188–193, August 11-13 1997. Singapore.

[9] Alan Cypher and D.C. Smith. KidSim: End User Programming of Simulations. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 27–34, 1995. Denver, CO.