Vol. 30 No. 4　　情　報　処　理　　Apr. 1989

解　説

2. プログラミング・パラダイムと環境

2.2 視覚的プログラミング環境†

西川博昭††　寺田浩詔††

1. はじめに

コンピュータシステムにおけるプログラミング言語あるいはプログラミング環境の重要性はあらためて強調するまでもないことである．ソフトウェア危機が叫ばれて以来，数々の提案がなされ，また実際に環境として提供されているのは本特集号が示すとおりである．

しかしながら，これらの提案の多くは，その了解性に関するかぎり，プログラミング言語ないしは環境の改革というよりは，進化と捉えるほうが適切である．すなわち，二，三の例外を除いて，因習的な線形的記述法が，歴然と踏襲され，プログラミング言語の構造は1次元でテキスト型のままであり，了解性が大幅に改善されたとは言い難い．これらは，いわゆるプログラマのための進化であり，システムの真のユーザであるエンドユーザにとって直接的に，そのユーティリティの向上が期待できるような改革ではなかった．

本稿の目的は，プログラミング環境の了解性を革新する可能性を秘めた視覚的プログラミング環境[1)-9)]の捉え方，現状ならびに展望を紹介することにある．

本稿に取り上げる視覚的プログラミング環境は，これまでの因襲的な言語・環境では，積極的には活用されていなかった図的な表現形式ないしは言語を，プログラミングという一連の活動に利用しようとするものである．近年，グラフィック関係のハードウェアとソフトウェアの急激な価格の低下が，コンピュータとの通信の手段として図の利用を容易にしたため，視覚的プログラミング環境は盛んに研究されるようになった．しかしながら，さまざまな見地から多様な研究が急速に進められているために，視覚的プログラミング環境が何であるかに関して一致した意見はない現状にある．

† Visual Programming Environment by Hiroaki NISHIKAWA and Hiroaki TERADA (Department of Electronic Engineering, Faculty of Engineering, Osaka University).

†† 大阪大学工学部電子工学科

したがって，本稿ではまず，このような視覚的プログラミング環境の必要性を述べるとともに，現在までに研究・開発されてきた代表的な視覚的プログラミングシステムを示す．続いて，これらのシステムの個々の実現法の詳細に依存しない水準の一つの捉え方として，視覚化の対象を取り上げ，この観点にてらして，視覚的プログラミング環境の現状を簡単に紹介する．最後に，視覚的プログラミング環境の潜在的能力を活用し，本来の目的を達成するための課題と将来的展望について私見をまじえて述べる．

2. 視覚的プログラミング環境の必要性とその現状

2.1 視覚的プログラミング環境の必要性

コンピュータシステムはその誕生以来，ハードウェア主導の発展過程をたどり，ユーザにとっての使いやすさよりも，その性能向上が重視されてきた．このため，ユーザとのインタフェースとなるべきプログラミング環境においても機械指向形の枠組みがはめられてきた．このようなプログラミング環境は，いわゆるプログラマという職業を生み出し，以来，プログラマは，決して十分とはいえないまでも，一般のユーザとコンピュータシステムを仲介する役割を担ってきた．

しかしながら，最近では特に，パーソナルコンピュータなどの出現によって，ユーザとシステムとの関係に新たな潮流が生じている．すなわち，いわゆるまったくの素人であるユーザにも，特別な訓練なしにコンピュータを自在に操れる必要性がでてきた．確かに，現在までに，多くの至便なツールが利用できるようになった．しかしながら，これらの多くは，あらかじめシステム提供者が予想した特定の応用範囲に限定されている．調査によれば，エンドユーザの要求した半分以上のアプリケーションは事前に設計されたアプリケーションには含まれなかったという結果が出ている．当然のことながら，多くのユーザの要求を満たす

よう用意されたパッケージといえども，すべてのユーザの要求を満足させられるわけではない．すなわち，既存のパッケージにない機能を要する場合には，ユーザみずからがプログラミングを行うことが本来望ましい．現在，従来形のプログラマを介した処理に比較して，エンドユーザプログラミングの必要性が急速に増加しつつある．

これまで，プログラミングは，分析的，論理的に考える能力を駆使する活動と捉えるのが当然とされ，多少の困難がともなうことは止むを得ないとされてきた．コンピュータはアルゴリズムを実現する機械であるから，対象とする問題をコンピュータを用いて解決するには，アルゴリズムを発見しなければならないことは明らかであり，その発見は，人間にとって永遠の難問である．すなわち，現在のプログラミングの本質的な問題点は，仮にアルゴリズムが発見されても，それをコンピュータが理解する言語への翻訳過程に困難がともなうことにある．人間の自然な思考は一般に並列的であり，曖昧に始まり次第に厳密化されるにも関わらず，現在までのプログラミング言語の多くは，機械向きの直列形の精密な記述を当初から要求する．このため，プログラマの存在が容認されてきた．しかし，エンドユーザプログラミングの必要性を認める立場をとれば，これまでの機械指向の考え方を一掃し，プログラミングの了解性・至便性を最重要視した，真にユーザフレンドリな観点からのプログラミング環境の改革が前提条件となる．

これにはまず，機械の処理方式ではなく，ユーザの思考形態を考慮しなければならない．すなわち，人間の思考を司る脳は二つに分かれていて，おのおのは主に反対側の体の運動機能を受けもつと同時に，論理的な言語能力は主に左半球に，視覚的な認知能力は主に右半球の機能であるとされている．換言すれば，左脳は分析的，論理的に，右脳はもっと直観的な感覚で判断する能力を備えている．視覚的プログラミング環境は，在来形のプログラミングに内在する困難を，明快な視覚化によって緩和し，人間が本来有する直感的な判断能力を有効に活用して，よりユーザ指向のインタフェースを確立しようとする試みと捉えることもできる．

2.2 視覚的プログラミング環境の現状

冒頭にも述べたように，視覚的プログラミング環境とは何かという定義は，必ずしも確立しているとは言い難い．しかしながら，「いわゆるソフトウェアのライフサイクルの過程で，意味のある図的表現を活用する環境」と捉えても差支えないであろう．すでに，図的な表現による視覚化の試みが，ソフトウェアライフサイクルのあらゆる側面に適用されているからである．

これらの試みは，視覚化の目標に基づいて，二つの大きな側面から捉えられるとされてきた[5]．すなわち，

i) プログラム開発用のツールに視覚情報を導入し，その了解性の向上を目標とする視覚的支援，および，

ii) 2次元の視覚的表記をソフトウェア設計やプログラム作成のために直接的に用いる視覚的言語である．

視覚化の対象に基づきより詳細にみれば，主として視覚的支援をめざすアプローチには，

(a) データベースの定義・操作の支援をめざすデータの視覚化，

(b) プログラムの構造や実行状況の了解性の向上のためのプログラム/実行の視覚化，さらには，

(c) いわゆるソフトウェアライフサイクル上の要求分析・定義あるいは設計過程における構造化分析手法など[10),11)]を基礎としたソフトウェア設計の視覚化，

(d) 例示によるプログラミングを発展させた視覚的コーチングなどが含まれる．

また，視覚的言語はその視覚化に用いる図的表現形式の違いにより，

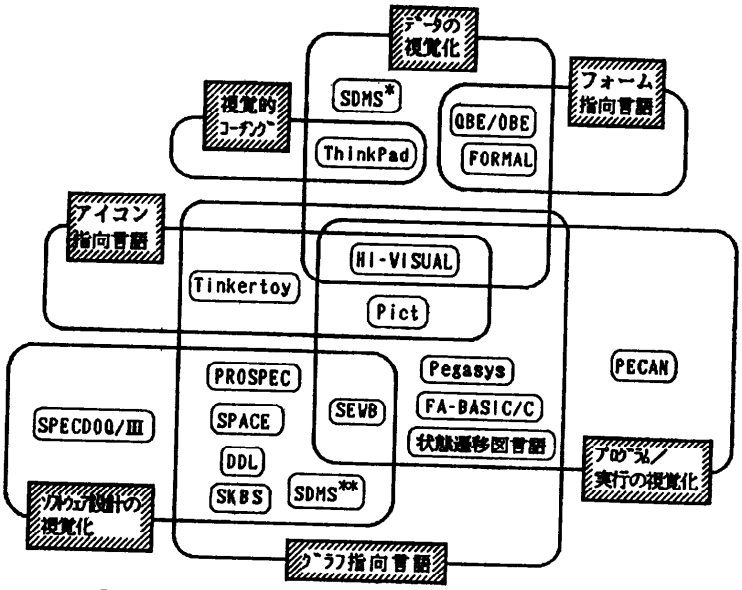
(e) データフロー図，制御フロー図，ブロック図などを用いたグラフ指向言語，

(f) 表およびそれに準じた形式の表現方法を用いたフォーム指向言語，さらに，

(g) アイコニックインタフェースから発展したアイコン指向言語に分類できる．

この発展に典型がみられるように，歴史的には異なる動機から研究開発されてきた視覚化の試みは，次第に融合・統合化される方向に進んでいる．**図-1** は，いわゆるベン図的表現を用いて，これまで研究開発されてきた代表的な視覚的プログラミングシステムとそれぞれが対象としている視覚化の側面の関係を示したものである．この図から分かるように，現在盛んに研究されている視覚的プログラミングシステムの多くは，視覚的支援および視覚的言語の双方にまたがる多様な側面の視覚化を対象としている．

表-1 は，図-1に取り上げた視覚的プログラミングシステムの開発主体，適用分野，視覚化の対象，図的表現形式および特徴を取りまとめたものである．



SDMS *: Spatial Data Management System
SDMS**: Software Development and Maintenance System
図-1 代表的な視覚的プログラミングシステムにおける視覚化の対象

各システムの詳細については，表-1 に示した参考文献に譲り，以下，個々のシステムの実現法に依存しない水準で，視覚的プログラミング環境が何であるかを探るために，視覚化の対象の観点から，それぞれの試みの目的および特長を簡単に紹介する．

（a） データの視覚化[12)-21)]

データ構造，あるいはデータベースの視覚化を実現する環境に関するものである．

このような環境の本質的な目的は，データベースを図的な観点から構成し，情報検索処理を視覚化して，ユーザが容易に扱えるようにすることにある．

データベースに蓄えられている情報やデータを，表形式などを用いて簡明に表示するアプローチのみならず，空間的な広がりを有する構造として視覚化する環境も研究・開発されている．この環境では，ユーザは，ジョイスティックやポインティングデバイスを用いて空間的な広がりの中を自由に探索することによって，所望の情報を獲得することが可能となっている．

（b） プログラム／実行の視覚化[18)-30),22)-28)]

プログラマや一般のユーザにプログラムとその実行形態を理解させることを主な目的としている．

これらのシステムの視覚化の対象は，ソースコードの了解性の高い表示方法から，多様な側面からのアニメーション形式を用いたプログラムの実行時の動作の視覚化に至るまで，非常に広い範囲にまたがっている．たとえば，表-1 には取りあげなかったが，視覚的なシミュレーションやデバッギングを対象とするシステムも研究されている[29)-32)]．

（c） ソフトウェア設計の視覚化[28),33)-39)]

要求・設計仕様の了解性は，信頼性・生産性の高いソフトウェア環境の構築に，決定的な影響を与える．今後，ソフトウェアがますます大規模，複雑化していくにつれ，それらの理解は困難なものとなっていくと予想される．このため，視覚化技術は，ライフサイクルの全工程を支援するソフトウェア環境の了解性を画期的に向上させるものとして期待されるようになった．プログラムや実行を視覚化するグラフィック技術の成功によって，より広範なプログラミング過程を視覚化する努力が払われるようになってきたためである．

この領域には対照的な二つのアプローチがある．一つは，ユーザの設計に対して，了解性に優れたグラフィックツールを提供することを主な目的とし，直接的にプログラム生成を支援するものではない[39)]．他方は，より厳密なアプローチであり，形式的な図的表現を用いる．このアプローチの目的は，形式的な図的表現とプログラムとの間の一貫性を保証する方法論の探究にある[28),33)-38)]．

（d） 視覚的コーチング[21)]

問題解決プロセスとプログラミングプロセスとの間のギャップを緩和することを目的とした，視覚的な手法による対話的システム環境に関する試みである．

これらのシステム環境では，ユーザはプログラム作成中に，みずからが与えた命令の効果を，頭の中で視覚化する必要はない．その効果は目前のスクリーンに示される．このプログラミング工程は，ほぼ完全にグラフィックを用いた対話によって進められる．このような対話形式は，データの構造図の表示やデータに対する操作の流れを提示するなど，作成者が自身のプログラムを説明するときに一般に用いる方法を模擬している．したがって，ユーザは従来形の言語に付随する

表-1 代表的な視覚的プログラミングシステム

システム名	開発主体	適用分野	視覚化の対象	図的表現形式	特徴	参考文献
SDMS*	Computer Co. of America	専用(データベース)	データの視覚化	表	視覚的なデータ空間を活用して，情報の組織化や検索を容易にする.	13)
QBE/OBE	IBM	専用(データベース)	データの視覚化 フォーム指向言語	表	基本的な操作数が少なく比較的容易に習得できるデータベース操作言語である.	14),15)
FORMAL	IBM	汎用	データの視覚化 フォーム指向言語	表	階層的な表形式と高機能な操作の導入によって，QBE/OBEより一般的なデータ構造を操作できる言語として定義されている.	16)
HI-VISUAL	広島大学	汎用	データの視覚化 プログラム/実行の視覚化 グラフ・アイコン指向言語	データフロー図	オブジェクトをアイコン化し，トップダウン形にも，ボトムアップ形にもプログラミングが可能である.	18)〜20)
ThinkPad	Brown 大学	汎用	データの視覚化 視覚的コーチング	データブロック図	視覚的な例示によるプログラミング環境であり，Prologプログラムを自動生成できる.	21)
PECAN	Brown 大学	汎用	プログラム/実行の視覚化	制御フロー図	文章形プログラムおよび実行時のスタックの状況を視覚化している.	23)
Pict	Washington 大学	汎用	プログラム/実行の視覚化 グラフ・アイコン指向言語	制御フロー図	文章形プログラムのステートメントをアイコン化した，制御フロー図を用いている．また，実行経過の視覚的シミュレーションが可能である.	24)
状態遷移図言語	Naval 研究所	汎用	プログラム/実行の視覚化 グラフ指向言語	制御フロー図(状態遷移図)	状態遷移の視覚的シミュレーションが可能である.	25)
Pegasys	SRI International	汎用	プログラム/実行の視覚化 グラフ指向言語	機能ブロック図 制御フロー図 データフロー図	視覚化された設計情報とプログラム間の一貫性を検証する機能をもつ.	26)
FA-BASIC/C	日立製作所	専用(制御)	プログラム/実行の視覚化 グラフ指向言語	制御フロー図	シーケンス制御用のプログラミングシステムであり，実行状況の実時間表示ができる.	27)
SEWB	日立製作所	汎用	プログラム/実行の視覚化 ソフトウェア設計の視覚化 グラフ指向言語	データフロー図 制御フロー図(PAD)	設計，プログラミング，テストの各工程を支援するツールによって，ソフトウェアの分散開発環境を提供する	28)
SDMS**	日本電気	専用(通信)	ソフトウェア設計の視覚化 グラフ指向言語	機能ブロック図 制御フロー図(SDL/GR) 表	通信プログラムの設計文書を自動的に作成できる.	33)
DDL	日立製作所	専用(通信)	ソフトウェア設計の視覚化 グラフ指向言語	データフロー図	データフロー図の利点を活用して，部品化を追求している.	34),35)
PROSPEC	Texas 大学	専用(通信)	ソフトウェア設計の視覚化 グラフ指向言語	制御フロー図(状態遷移図)	プロトコルの検証機能をもつ通信処理の設計支援ツールである.	36)
SKBS	富士通	専用(通信)	ソフトウェア設計の視覚化 グラフ指向言語	制御フロー図(SDL/GR)	知識処理手法によって，プログラムを自動生成する.	37)
SPACE	電力中央研究所	専用(事務)	ソフトウェア設計の視覚化 グラフ指向言語	機能ブロック図 表	知識処理手法による COBOL プログラムの自動生成機能および仕様の矛盾の検証機能をもつ.	38)
SPECDOQ/Ⅲ	日本電気	汎用	ソフトウェア設計の視覚化	表 データブロック図 機能ブロック図	多様な図的表現を用いて，仕様書の作成を支援する.	39)
Tinkertoy	Illinois 大学	汎用	グラフ・アイコン指向言語	データフロー図	LISP プログラムの視覚的インタフェースを提供する.	40)

注) SDMS* : Spatial Data Management System
SDMS**: Software Development and Maintenance System

ような人工的な文法の煩雑さをまったく意識する必要はない.

このアプローチの考え方は,特定の具体的な対象を扱うほうが,抽象的な対象を扱うよりも了解性が高いという事実を基礎としている.すなわち,適切な例を選択し使用すれば,具体的な例から抽象的な概念を理解させることが可能であるという立場をとっている.

(e) グラフ指向言語[18)~20),24)~28),33)~38)]

プログラミング言語自身に視覚的な構造を与えようとする試みである.

長年,ソフトウェア開発では,数多くのグラフ指向の図式がプログラムを表現するために利用されてきた.しかしながら,これらは実行可能なプログラムとなるものではなく,単なる補助的手段でしかなかった.したがって,一つの同じプログラムに対して,図式とコードの両方を更新し続けなければ,デバッギング工程では,図式はもはや実行可能なプログラムからかい離し,後の保守に問題を引き起こす原因となっていた.

すなわち,このアプローチの目的は,図式を実行可能にして,単にプログラムを早く作成するだけではなく,ドキュメントの作成と保守を容易にすることにある.

(f) フォーム指向言語[14)~17)]

グラフ形の図式の代わりにフォームと呼ばれる表形式を用いた言語も開発されている.フォームは,もっとも自然なユーザインタフェースの一つである.私生活でも日常の仕事でも,購入申込書やレポートなどにフォームを,多くの人々が利用していることをみれば明らかである.

現在,フォームを用いる方法は,メニュー形式のユーザインタフェースやオフィスオートメーションシステムにおける,データの入力および表示装置,データベースの参照,更新に広く使用されている.

(g) アイコン指向言語[18)~20),24),40)]

先にも述べたように,アイコンやグラフィックシンボルが中心的役割を果たすように設計されたインタフェース[41),42)]から発展した言語である.

このアプローチでは,アイコンの役割はもはやデスクトップオブジェクトの表現やそれらを操作するコマンドのみに制限されない.すなわち,アイコンによって,言語のいわゆるコンストラクトが視覚化されており,アイコンの配置・接続によってプログラムを記述できる.このカテゴリに属する研究も盛んに行われており,表-1 に示した以外に,PT[43)] や Oscar[44)] などがある[6)~9)].

視覚化の効果を,もっとも積極的に追究するアプローチの一つであるが,一見してその意味を把握しやすい反面,あいまいさが残ることが,検討課題として指摘されている.

3. 視覚的プログラミング環境の課題と展望

先にもふれたように,視覚的プログラミング環境に関して共通の理解が得られているとは言い難い現状にある.この分野が非常に新しいものであることも一つの原因と考えられるが,共通の理解は,この研究の進展のためだけではなく,潜在的なユーザに広く利用されるためにも重要となる.プログラミング環境の成熟にはいわゆる使い込みが大きな役割を果たすからである.

本稿では,視覚的プログラミング環境を,「いわゆるソフトウェアのライフサイクルの過程で,意味のある図的表現を活用する環境」と定義した.この延長線上に共通の理解を得るとしても,『意味のある図的表現』を明確にしなければならない.このためには,図的表現に関する基礎研究とともに,単にシステム設計,プログラミングの分野に留まらず,これまであまり関心が払われてこなかった他の分野での図的表現の活用法に目を向けるとともに,統合的に捉える積極的な姿勢が必要である.

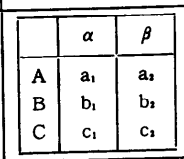
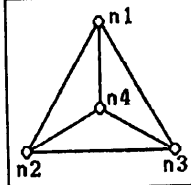
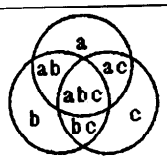
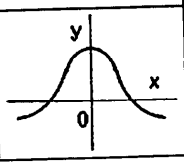
3.1 意味のある図的表現とは

情報の(広い意味での)視覚的な表現形式は一般に,文,式,図,および,絵に大別される.文,式と図,絵の違いは明らかであるが,ともに2次元表現である図と絵についても,以下の知見が得られている[45)].

一般に言語と呼ばれるものは,狭義には日本語,英語などの自然言語のみを指すが,広義にはコンピュータ言語,数式,論理式などの人工言語も含められる.すなわち,有限個の要素とその配列規則(文法)によって無限個の文や式を生成可能で,かつこれらの数限りない文や式の意味が,要素の意味とその配列規則から導出できる場合,その表現体系は言語と呼ばれる.表-2 に示す表現形式をもつ図は,いわゆる絵と違って,これらの要件を満足する図的言語と捉えられている.

一方,絵の場合は,それを構成する要素の種類は不定であって,個々の絵に用いられる要素は,無数といってもよいほど多様性があり,その解釈はきわめて主観的となるため,形象系と呼ばれる.しかし,先に述べた図的言語に属さない図とともに,この形象系につ

表-2 図的言語の分類とその特徴

	表現形式の一例	意味単位	配列規則	解釈	例
行列系		平面を横に分割する行線と縦に分割する列線.	行と列の交叉関係が意味をもつ.	線がもつ平面分割機能を利用して, 交叉する行項目と列項目との関係を表す.	関係表, 決定表, フォーム, 時刻表, 工程図, マトリックス図
グラフ系		アークと呼ぶ線分とノードと呼ぶ点(または図形や文字).	ノード間を連結するアークの有無が意味をもつ.	線がもつ連結機能を利用して, ノード間の関係を表す.	データフロー図, 制御フロー図, ブロック図, 組織図, 工程図, 地図, 航路図, KJ 図, 連関図, 系統図, 過程決定計画図, アローダイヤグラム
領域系		任意の形状の閉曲線	閉曲線相互の包含, 交叉, 分離, 隣接関係が意味をもつ.	閉曲線がもつ平面分割機能を利用して, 部分平面間の関係を表す.	ベン図, 地図, 三面図, KJ 図, 親和図.
座標系		座標軸を示す軸線, 位置を示す点, 数量や方向や運動を表す線分.	座標軸が示す場(フィールド)における点や線の大小関係, 位置関係が意味をもつ.	点がもつ位置決め機能と, 線がもつ量表示, 方向指示, 運動軌跡表示などの機能を利用して, 位置や量を数量的に表す.	棒グラフ, 円グラフ, 地図, 天気図などの分布図, 三面図, 航路図, 列車ダイヤ図

いても, たとえその解釈が主観的で一意的に定まらないとしても, 形象としての意義を追究する姿勢もまた必要である. 形式性をあまりにも追究すれば, 視覚的プログラミング環境といえども, 再びユーザ指向でなくシステム指向の環境となってしまう危険がある. エンドユーザプログラミングの次には, より自由な個性化が求められると予想されるからである.

3.2 図的表現の積極的な活用のために

図-2 は, ベン図的表現を用いて, 表-1 に示した視覚的プログラミングシステムとそれぞれが基礎としている図的表現形式, すなわち, 図的言語との関係を示している. また, グラフ系の図的言語に対しては, 状態遷移図あるいはいわゆるフローチャート的な表記法を含む制御フロー図, 並列性を自然に表現できることなどから最近特に注目されているデータフロー図, さらに, これらの二つの表現形式に比較して, より緩やかな解釈が許される一般的な機能ブロック図, データブロック図などが含まれることも併せて示している.

図-2 から分かるように, 視覚化のための図的言語では, いわゆるグラフ系, 行列系といった, これまでコンピュータシステムの分野で慣れ親しまれてきた表記法への偏りがみられる.

視覚化の本来の目的の達成には, 図的表現のさらに積極的な活用が必要であることは明らかである. すなわち, システムの性質を包括的に捉えられる領域系やシステムの定量的あるいは動的な挙動を陽に示す座標系などに属する表現形式も含めた統合的な環境の構築が必要である. 要求定義から設計, プログラミング, 保守・運用を統合的に一貫して支援できるシステムがまだ構築されていないのはその証左である.

システムの分野にとらわれず, 広い視野にたてば, 図的な表現法を積極的に活用した図的な思考法に関する試みが盛んに行われている. 表-2 に示した KJ 図による発想法などはその先駆的研究である. また, 統合的品質管理 (TQC) の分野においても, 親和図, 連関図, 系統図などの多様な表現形式が思考の手段に広く用いられている[46]. これらへの積極的な取り組みが, 視覚的プログラミング環境の将来には必要である[47].

3.3 統合的な環境の実現のために

統合的な視覚的プログラミング環境の実現には, 先に述べた図的表現に関する基礎的な研究および積極的な活用に加えて, よりユーザ指向にシステムの設計思想を転換しなければならない. すなわち, これまでの

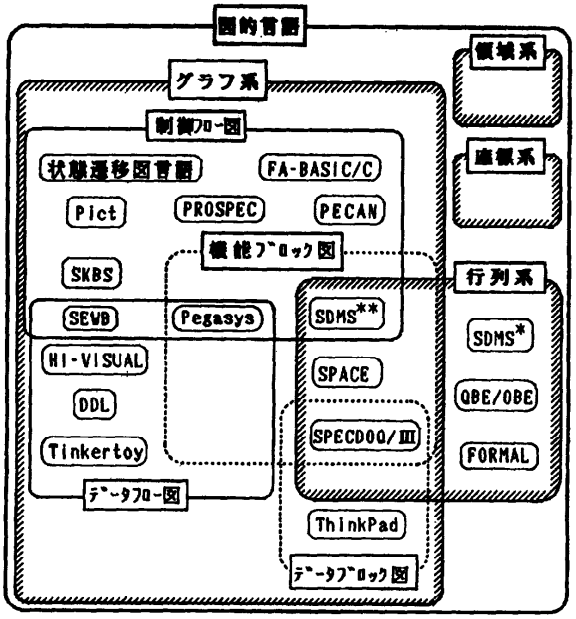


図-2 現状の視覚的プログラミングシステムに活用されている図的言語

プログラミング環境は，視覚化のいかんを問わず，対象とする処理機械の実行形式に依存した手続き性などの本質をそのまま反映した仕様をもつ言語ないしは表現形式をユーザに強いてきたきらいがある．ユーザ指向のシステムにとって，このような傾向は明らかに好ましいことではない．エンドユーザ，プログラマを問わず，ユーザは一般に，問題解決の当初から，当面する問題を処理機械に依存したある特定の表現形式を用いて，手続き的に記述できる段階に達しているわけではないからである．したがって，真にユーザフレンドリな環境の実現には，先に述べた一般的なブロック図的な表記法，さらにはユーザ自身の定義をも含めて，複数以上の表現形式を用いた非手続き的記述を支援する，いわゆるマルチリンガルな環境[48]が実現されるべきである．

この結論に従えば，視覚的プログラミング環境の定義に関してよく議論の対象になるマルチウィンドウ環境についても新たな捉え方が必要である．これまでウィンドウの利用は，視覚的プログラミング環境に直接的には関係ないとされてきた[5]．その理由として，i）ウィンドウは必ずしもプログラミングに関する図的表現を含んでいるわけではないこと，および，ii）プログラミングは必ずしもマルチウィンドウを用いることに依存するわけではないこと，などが指摘されてきた．しかしながら，マルチリンガルな記述方式を導入して，対象とする問題を多様な側面から自然に表現するためには，単に平面的なウィンドウ環境に留まらず，より自然な3次元的な空間の広がりをもつウィンドウシステムも将来の視覚的プログラミング環境をイメージするために重要な鍵となると考えるのが自然である[49]．

筆者らも，図的データ駆動言語（Diagrammatical Data-Driven Language : D^{3}L）[50]を核にして，i）従来から種々の専門領域で広く使用され定義・解釈が明確に確立している信号流れ図，状態遷移図などから副作用が生じないD^{3}Lで記述された高度並列処理プログラムを自動的に生成する方式[51]が可能であることを実証するとともに，ii）この方法の拡張の可能性を検討した結果，定義・解釈が必ずしも厳密ではないが，ユーザの直感的理解を助け問題を定式化する上で重要な役割を果たす一般的なブロック図や一般的なシーケンス図からもデータ依存図式が誘導可能であるという見通しを得て，真にユーザフレンドリな超高位図的言語処理システムを構築する努力を続けている[52),53]．

4. おわりに

本稿では，コンピュータの大衆化にともなって，これを専門にしないエンドユーザプログラミングを支援する立場から，視覚的プログラミング環境が潜在的可能性をもち，今後ますます重要になることを述べた．

空間的な広がりをもつ視覚情報の利用は，より自然なマンマシン・インタフェース，さらにはマシンを意識する必要のない理想的なインタフェースの実現には，欠くことのできない要素技術となると予想される．

しかしながら，真にユーザフレンドリなシステムの

構築には，視覚的プログラミング環境だけで十分なわけでもない．本特集号に解説されている新しいプログラミング環境のためのさまざまな技法と併せて，この小文が，ユーザに優れて友好的なプログラミング環境を考えるうえでいくらかでも参考になれば幸いである．

参考文献

1) Grafton, R. B. and Ichikawa, T. (eds.): Special Issue on Visual Programming, IEEE Comp., Vol. 18, No. 8 (1985).
2) 市川忠男：視覚的プログラミングについて，電子通信学会コンピュテーション研究会報告，COMP 86-46, pp. 43-46 (1986).
3) 平川，田中：視覚言語の動向，電子通信学会誌，Vol. 69, No. 12, pp. 1256-1259 (1986).
4) Chang, S.: Visual Languages: A Tutorial and Survey, IEEE Software, pp. 29-39 (1987).
5) Shu, N. C.: Visual Programming, p. 315, Van Nostrand Reinhold Company Inc., New York (1988).
6) Proceedings of IEEE Workshop on Visual Languages (1984).
7) Proceedings of IEEE Workshop on Visual Languages (1986).
8) Proceedings of IEEE Workshop on Visual Languages (1987).
9) Proceedings of IEEE Workshop on Visual Languages (1988).
10) Suydam, W.: CASE Makes Strides toward Automated Software Development, Computer Design, Vol. 26, No. 1, pp. 49-70 (1987).
11) Ward, P. T.: The Transformation Schema: An Extension of the Data Flow Diagram to Represent Control and Timing, IEEE Trans. Softw. Eng., Vol. SE-12, No. 2, pp. 198-210 (1986).
12) Donelson, W. C.: Spatial Management of Information, Proc. of ACM SIGGRAPH '78, pp. 203-209 (1978).
13) Herot, C. F.: Spatial Management of Data, ACM Trans. Database Syst., Vol. 5, No. 4, pp. 493-514 (1980).
14) Zloof, M. M.: Query-by-Example: A Data Base Language, IBM System Journal, Vol. 16, No. 4, pp. 324-343 (1977).
15) Zloof, M. M.: QBE/OBE: A Language for Office and Business Automation, IEEE Comp., Vol. 14, No. 5, pp. 13-22 (1981).
16) Shu, N. C.: FORMAL: A Forms-Oriented, Visual-Directed Application Development System, IEEE Comp., Vol. 18, No. 8, pp. 38-49 (1985).
17) Yao, S. B., Hevner, A. R., Shi, Z. and Luo, D.: FORMANAGER: An Office Forms Management Systems, ACM Trans. on Office Information Systems, Vol. 2, No. 3, pp. 235-262 (1984).
18) Hirakawa, M., Iwata, S., Yoshimoto, I., Tanaka, M. and Ichikawa, T.: HI-VISUAL Iconic Programming, Proc. of IEEE Workshop on Visual Languages, pp. 305-314 (1987).
19) Ichikawa, T. and Hirakawa, M.: Visual Programming-Toward Realization of User-Friendly Programming Environments, Proc. of Fall Joint Computer Conference '87, pp. 129-137 (1987).
20) Hirakawa, M., Iwata, S., Tahara, Y., Tanaka, M. and Ichikawa, T.: A Framework for Construction of Icon Systems, Proc. of IEEE Workshop on Visual Languages, pp. 70-77 (1988).
21) Rubin, R. V., Golin, E. J. and Reiss, S. P.: ThinkPad: A Graphical System for Programming by Demonstration, IEEE SOFTWARE, pp. 73-78 (1985).
22) Teitelman, W. and Masinter, L.: The Interlisp Rrogramming Environment, IEEE Comp., Vol.14, No 4, pp. 25-34 (1981).
23) Reiss, S. P.: PECAN: Program Development Systems That Support Multiple Views, IEEE Trans. Softw. Eng., Vol.11, No.3, pp. 276-285 (1985).
24) Glinert, E. P. and Tanimoto, S. L.: Pict: An Interactive Graphical Programming Environment, IEEE Comp., Vol. 17, No. 11, pp. 7-25 (1984).
25) Jacob, R. J. K.: A State Transition Diagram Language for Visual Programming, IEEE Comp., Vol. 18, No. 8, pp. 51-59 (1985).
26) Moriconi, M. and Hare, D. F.: Visualizing Program Designs through PegaSys, IEEE Comp., Vol. 18, No. 8, pp. 72-85 (1985).
27) 松崎，秦，浜野，倉島，鳥井：ペトリネット的表現を持つシーケンス制御用言語，情報処理学会論文誌，Vol. 28, No. 6, pp. 585-593 (1987).
28) 津田，葉木，前沢，石本，高橋：ワークステーションによるソフトの分散開発環境 SEWB，日経コンピュータ，No. 154, pp. 149-160 (1987).
29) Melamed, B. and Morris, R. J. T.: Visual Simulation: The Performance Analysis Workstation, IEEE Comp., Vol. 18, No. 8, pp. 87-94 (1985).
30) Smith, R. B.: The Alternate Reality Kit: An Animated Environment for Creating Interactive Simulations, Proc. of IEEE Workshop on Visual Languages, pp. 99-106 (1986).

31) Numao, M. and Fujisaki, T.: Visual Debugger for Prolog, Proc. of IEEE Conf. on Artificial Intelligence Applications, pp. 422-427 (1985).
32) Isoda, S., Shinomura, T. and Ono, Y.: VIPS: A Visual Debugger, IEEE Software, Vol. 4, No. 3, pp. 8-19 (1987).
33) Hagiwara, N. and Iwamoto, K.: A Graphic Tool for Hierarchical Software Design, Proc. of IEEE Workshop on Visual Languages, pp. 42-46 (1984).
34) Shirasu, H.: Innovative Approach to Switching Software Design Using Dataflow Concept, Proc. of International Switching Symposium, S-B 4. 3 (1987).
35) 前島，田辺，鈴木，猪熊，稲吉：データ駆動論理 (DDL) プログラミングにおける支援環境，電子情報通信学会交換システム研究会資料，SE 87-144, pp. 79-84 (1987).
36) Chow, C. and Lam, S. S.: PROSPEC: An Interactive Protocols, IEEE Trans. Softw. Eng., Vol. 14, No. 3, pp. 327-338 (1988).
37) 藤本，鈴木，加藤：知識ベース利用の SDL 支援システム，情報処理学会論文誌，Vol. 29, No. 5, pp. 497-505 (1988).
38) 原田 実：COBOL プログラム自動生成システム SPACE における仕様の視覚化と抽象化，電子情報通信学会論文誌 (D), Vol. J 71-D, No. 12, pp. 2555-2562 (1988).
39) 北川，石井，森，東：システム要求仕様作成管理システム SPECDOQ/Ⅲ，NEC 技報，Vol. 40, No. 1, pp. 28-34 (1987).
40) Edel, M.: The Tinkertoy Graphical Programming Environment, IEEE Trans. on Softw. Eng., Vol. 14, No. 8, pp. 1110-1115 (1988).
41) Smith, D. C., Irby, C. and Kimball, R.: The Star User Interfaces: An Overview, AFIPS NCC, Vol. 51, pp. 515-528 (1982).
42) Lodding, K. N.: Iconic Interfacing, IEEE CG&A, Vol. 3, No. 2, pp. 11-20 (1983).
43) Hsia, Y. T. and Ambler, A. L.: Construction and Manipulation of Dynamic Icons, Proc. of IEEE Workshop on Visual Languages, pp. 78-83 (1988).
44) Coote, S., Gallagher, J., Mariani, J., Rodden, T., Scott, A. and Shepherd, D.: Graphical and Iconic Programming Languages for Distributed Process Control: An Object Oriented Approach, Proc. of IEEE Workshop on Visual Languages, pp. 183-189 (1988).
45) 出原，吉田，渥美：図の体系一図的思考とその表現，p. 252, 日科技連出版社，東京 (1986).
46) 二見良治：TQC に役立つ図形思考法，p. 220, 日科技連出版社，東京 (1986).
47) 佐藤，田中，門田，山下：概念図作成支援のための図の意味記述，情報処理学会知識工学と人工知能研究会報告，88 AI 50 15, pp. 181-188 (1988).
48) Reiss, S. P.: Working in the Garden Environment for Conceptual Programming, IEEE Software, Vol. 4, No. 6, pp. 16-27 (1987).
49) Bolt, R. A.: The Human Interface, Van Nostrand Reinhold Company Inc., New York (1984).
50) 西川，寺田，浅田：履歴依存性を許すデータ駆動図式，電子通信学会論文誌 (D), Vol. J 66-D, No. 10, pp. 1169-1176 (1983).
51) 寺田，西川，浅田：D³L による図的言語処理体系とその一実現法について，電子通信学会第 1 回データフローワークショップ予稿集，pp. 119-126 (1986).
52) 西川，浅田，寺田：超高位図的言語処理システムの設計思想，第 33 回情報処理学会全国大会論文集，4F-9 (1986).
53) 寺田，西川，岩田，岡本，宮田，小守，嶋：VLSI 向きデータ駆動プロセッサ：Q-π，電子情報通信学会論文誌 (D), Vol. J 71-D, No. 8, pp. 1383-1390 (1988).

(平成元年 2 月 16 日受付)