4E-2　ＡＰＬのプログラミング環境に関する一考察

佐貫　俊幸
日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

１．はじめに

近年，Smalltalk-80に代表されるように，言語システムのプログラミング環境，特に高い操作性の統合的な対話型開発支援環境に関する研究が盛んに行なわれている[1]．この背景にはワークステーションの進歩および普及によって，従来からあるホスト・マシン－端末の世界ではなし得なかった高いユーザビリティが実現できるようになった点が挙げられる．

従来ＡＰＬは対話型汎用言語として長く使われてきたが，比較的多くの資源を必要とするため，稼働環境はホスト・システムが中心であった．だがワークステーションの高機能化によってワークステーション上でも稼働するＡＰＬシステムが実現できるようになった．しかしこれらは主としてホスト上のＡＰＬの仕様をそのまま又は低減して移植したもので，対話処理環境あるいはユーザビリティはその稼働環境を充分考慮したものであると言い難い．伝統的なホスト端末の制約のもとで，セッション・マネジャー[2]によって端末画面の情報を管理しているホストのＡＰＬではなしえなかった高い操作性をワークステーションの上で実現させる必要がある．

プログラミング環境に関する議論はエディタ，デバッガ，ツール，ネットワーク，その他様々な要素において行なわれなければならないが，本稿ではＡＰＬ言語とその視覚的プログラミング環境に重点を置いて，ワークステーション上でどのように融合させていけばよいかという観点で，今後ＡＰＬの対話処理環境をいかに向上させたらよいかについて検討を行う．

２．従来のＡＰＬの問題点

プログラムの開発過程において，プログラマーが一番長いあいだ時間を費やすのがエディタである．そこでＡＰＬの関数の編集作業を例に考えてみる．

ＡＰＬにおいてユーザが関数（プログラム）を定義するには２つの方法がある：

（１）"∇"記号により定義モードに変えて一連のプログラム・ステートメントをコンソールから入力し，また"∇"により実行モードに切り換えて定義した関数を実行する方法．

（２）実行モードにおいて，システム函数⎕FXを用いて正準形を関数化して実行をする方法．

（１）の方法を用いた場合には，テレタイプ端末のセッションを基本としたライン・エディタにおいて，行の追加・修正も関数のステートメント番号を明示して行う編集作業が主体となる．一方（２）の方法の場合には，予め編集する関数をワークスペースに用意して，ライブラリーに保管し，必要時に明示的に活動化して用いるという手間が必要である．この場合も（１）の方法に類したセッションで編集が行なわれる．いずれも，関数の編集作業を行っている間は，関数のテキスト情報しかプログラマーの眼には入らない．しかし実際の編集作業においてはワークスペースの中の様々情報，例えば既存の変数名，関数名，その他のシステム情報などを参照する必要がよくある．また関数を実行し，その結果を評価しながら編集を行う必要も多くある．そのたびにモードの切り換えを行っては効率が悪い（図１参照）．

このように，現在のＡＰＬの編集作業一つ取ってみても効率的な対話処理環境が提供されているとは言いがたく，一元的な情報を元に行なわれるプログラミング作業ではその生産性も低下すると思われる．

```
      ∇Z←Comment Q;⎕IO                                    ↑
[1]   A   Show the description in function comment (A)     |
[2]   A   Q : Function Name,   Z : Result                  |
[3]   Z←0ρ⎕IO←1                                            |
[4]   →(3≠⎕NC Q)/0                                  関数の定義
[5]   Z←(1,1↓'A'=Z[;1])/Z←⎕CR Q                           |
[6]   Z←1↓(~∘.∧\∘.=' ')/,Z←⎕TC[2],Z                       |
[7]   ∇                                                    ↓
      Comment 'Comment'                             関数の実行
RANK ERROR
Comment[6]
      Z←1↓(~∘.∧\∘.=' ')/,Z←⎕TC[2],Z                エラーの発生
           ∧
      ∇Comment[⎕]                                          ↑
[0]   Z←Comment Q;⎕IO                                      |
[1]   A   Show the description in function comment (A)     |
[2]   A   Q : Function Name,   Z : Result                  |
[3]   Z←0ρ⎕IO←1                                    関数の編集
[4]   →(3≠⎕NC Q)/0                                         |
[5]   Z←(1,1↓'A'=Z[;1])/Z←⎕CR Q                           |
[6]   Z←1↓(~∘.∧\∘.=' ')/,Z←⎕TC[2],Z                       |
[6]   Z←1↓(~,∘.∧\∘.=' ')/,Z←⎕TC[2],Z                      |
[7]   ∇                                                    ↓
      Comment 'Comment'                             関数の実行
Z←Comment Q;⎕IO
A   Show the description in function comment (A)
A   Q : Function Name,   Z : Result                 結果の確認
英数 半角
```

図　１．関数の編集作業の流れ

A Visual Interface for APL Programming Environment
Toshiyuki SANUKI
Science Institute, IBM Japan, Ltd.

3．APLの視覚環境の整備

前述の編集作業を例に，いかに効率よく行うためにはどのようにしたらよいかを考える．

１つの関数の編集や実行にまつわる情報が，画面上自由にプログラマーに提供できる機能が必要である．対話処理環境を提供する言語システムのユーザインターフェースを向上させる上で，ウィンドウ機能を導入することは有効な方法である[3]．APLのセッションにウィンドウを導入する場合，ワークスペースとの関連が必要である．

図2に具体的な一例を示す．画面左側のウィンドウでは関数の編集が行われ，右上のウィンドウではこのエディタのコマンドが表示されている．エディタのウィンドウの中では，APLのステートメントは一般のテキストエディタと同様に入力・操作できる．編集作業の間に既に使われている変数名を知りたい場合には，エディタのウィンドウとは別に，新たなウィンドウを開く．これが右下のウィンドウで，この中でアクティブなワークスペースが提供され，プログラマーはこのウィンドウ内でシステム・コマンドを発して，現在のワークスペースの状況を得ることができる．またこの中で，参照している関数を実行し，その評価を行い，再度エディタのウィンドウに移り，その結果をもとに編集作業を続けることも可能になる．

また，単に画面上に複数のウィンドウが開け，ワークスペースが各ウィンドウに対応できるようにしただけでは不充分である．複数のウィンドウ間で，関数や変数及データなどが自由にコミュニケーションできるようにしなければならない．

この様に，ワークスペース内の情報を画面上分割して表示することによって，プログラマーが得られる情報の"チャネル"が増え，図1で示したような従来のプログラム編集作業における定義モードと実行モードのシリアルな流れを変えることが可能となる．

4．おわりに

本稿では今後APL言語システムがどのようにあるべきかについて，視覚的なプログラミング環境の側面から一つの提案を行った．

現在この言語システムの設計及びモデルの評価・検討を行っている．今後APLのエクゼキュータの中にこのような機能や使いやすいエディタ等を組み込んで，より効率のよいプログラミング環境を言語システムの中でも提供できるように考えてみたい．

参考文献

[1] A.Goldberg : Smalltalk-80 The Interactive Programming Environment, Addison Wesley, 1984

[2] IBM Corp. : "The APL2 Session Manager" in APL2 System Service Reference Manual, SH20-9218, 1985

[3] W.Teitelman : "A Display-Oriented Programmer's Assistant" in D.R.Barstow et al., eds., Interactive Programming Environment, McGraw-Hill 1984

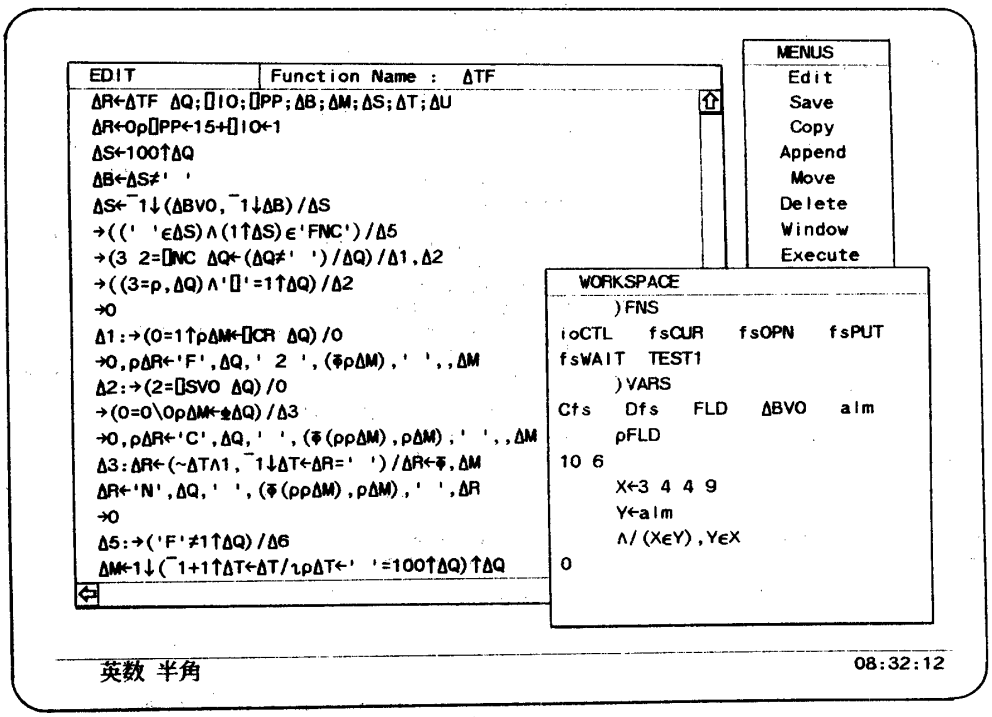

図 2．
ウィンドウを導入した
APLのセッション例