マイクロコンピュータ 16－2
（１９８１．４．２０）

テキストエディタについての一考察

（テキストエディタの使い易さの向上とテキストエディタに注目した分散処理の概念）

富士通（株）　開発技術部　大沢　光
商品企画部　佐藤健二

１．はじめに

１－１．問題の背景

筆者らはこの稿の主題であるテキストエディタを考察する前に，以下に示す問題に関わってきた。これらの問題は，この考察を通して直接あるいは間接に意識されている。

（１）より現実的なソフトウェアエンジニアリング

所謂ソフトウェアツールを整備していこうとするアプローチでは，ＡＰＧ（アプリケーションプログラムジェネレータ），プログラムコンバータ，テストデータジェネレータなど多種多様な「道具立て」が考えられている。このうち，プログラムの生産性の向上に最も効果的なものは何であろうか。

（２）「使い易さ」（マンマシンインタフェース）の概念

オフィスオートメーションなどを含めて，所謂エンドユーザ部門へのコンピュータパワーの拡散が盛んになりつつある。簡易言語あるいはパッケージプログラムなどがこれに対応するが，これらの一つの「鍵」はその使い易さにあると考えられる。ここで，その使い易さとは何であろうか，使い易さを更に向上するには何をすればよいのであろうか。

（３）ローカルパワーの向上に伴う分散処理

インテリジェントターミナル，パーソナルコンピュータなどローカル側のインテリジェンスの向上に伴なって，ホスト側との処理の分担が問題となっている。一方では全てホスト側が分担する方法があり，他方では共通のデータベースとネットワークコミュニケーションのみホスト側に残す方法が考えられるが，その間にあるであろう「最適な」機能の分担はどう考えたらよいのであろうか。

１－２．テキストエディタへの注目

（１）パーソナルコンピュータとの出会い

数年前からマイクロプロセッサを利用したパーソナルコンピュータが出現している。筆者らもそのうちのいくつかを使用して，いくつかの調査および実験を行った。その結果，パーソナルコンピュータの最大の特徴は「コンピュータであること」であり，（今後，急速に解決されていくであろうが）処理速度と記憶容量さえ問題としなければ充分実用にな

ると確信された。

更に，パーソナルコンピュータのもう一つの特徴を挙げると，それはその「使い易さ」という事である。これは，パーソナルコンピュータが独占して使用出来る事もさることながら，その「設計」が使われる事を原点としてなされている為と考えられる。後にも述べるが，そのポイントの一つは，ディスプレイを利用したスクリーンエディタであり，これによって修正したいコマンドあるいはステートメントを打直す事なくカーソルを使って直接修正したり，出力したリストを再び修正して入力とするなどのオペレーションが可能である。また，他の一つのポイントは，スクリーンエディタの（二次元の）機能がプログラムの中で利用出来る事であり，これによってプログラムはその出力編集の記述順序の拘束から（ある程度だが）解放され，人にとって「自然な」順序でプログラムを記述出来る可能性を示している。

（2）テキストエディタへの注目

筆者らは以上の状況を契機として，（プログラム）テキストエディタに注目した。つまり，テキストエディタに関する整理あるいは議論をする事で，1－1に述べた問題の解決への手掛りが見出せる可能性を感じたからである。つまり，以下の視点からテキストエディタを考察する事が，この稿の狙いである。

（i）テキストエディタの使い易さを向上する事で，プログラミングの生産性はまだ向上する。

即ち，テキストエディタの「使い易さ」とは何か。

（ii）プログラムの中からスクリーンエディタあるいはそれと同等の概念が使用出来ると，「自然な」順序でプログラムを記述出来る。

即ち，そのプログラムアーキテクチャはどんなものか。

（iii）テキストエディタおよびマンマシンインタフェースの多くの部分は，ローカル側で処理可能である。

即ち，その機能分担の方法は何か。

2．テキストエディタに関する「整理」

2－1．テキストエディタの機能

（1）その定義

この稿では，取敢えず「目的とするテキストファイルを作成・編集・修正する事を支援する為のツール」と定義しておく。ここで，テキストとはコード化されたデータを指し，

またテキストファイルとはその論理的区分を持った集りを指している。

（2）その機能

テキストエディタの機能は，基本的に以下の様に分類する事が出来る。これらには，複数のファイルにわたる機能は除かれている。

①――投入・・・・・・input，auto

②―┬挿入・・・・・・insert
　　├削除・・・・・・delete
　　├転写・・・・・・copy
　　├移動・・・・・・move
　　└置換・・・・・・change

③―┬検索・・・・・・find
　　└参照・・・・・・list

④――その他・・・・・renumber

それぞれのテキストエディタの使い易さは，オペレーションの考え方によって異っているが，それらは以上の「基本機能」をより高度化する為のもので，この稿の狙いの一つは正にこの点にある。

2－2．テキストエディタの流れ

ここでテキストエディタに関連する「技術的」なインパクトを振り返り，それに伴って生れた「概念」の流れを整理しておく。

（1）ＴＳＳの登場

ＴＳＳの登場によって，所謂「対話型」の処理が普及し，従来のバッチ型の編集ユーティリティに代ってテキストエディタが誕生した。初め「投入」，「削除」および「参照」のみに限定されていた機能も，対話型処理の普及に伴いより「対話的」なものへと拡張され，「置換」，「転写」，「移動」，「検索」，「その他」の機能が追加された。（タイプライタ型ターミナルを前提として）マンマシンインタフェースの改善は，コマンド入力の方法を中心として行われた。

（2）ディスプレイ型ターミナルの登場

ディスプレイ型ターミナルの登場は，（紙を必要としない）ソフトコピー化によって「対話」，特に表示のスピードを向上した。更に，ディスプレイスクリーンを明に取り扱う事で，カーソルおよびその移動，画面の（全体あるいは部分的）消去，あるいは字・語・

行単位の処理などを含む機能が追加された（「スクリーン概念」の登場）。これはまた，スクリーンを「分割」し，それぞれ別の目的に使用するなど「マルチスクリーン／ウインドウ概念」へと発展している。ディスプレイ型ターミナルの登場は，スクリーン概念を初めとするいくつかの「概念」を生んだが，その実現はホスト側のプログラムによるものを中心として行われた。

（3）ローカルインテリジェンスの発展

インテリジェントターミナルあるいはパーソナルコンピュータに代表されるローカルインテリジェンスの発展は，最近特に著しい。これは一方ではインテリジェントターミナルにみられる「分散処理概念」を生み，他方では，マンマシンインタフェースをローカル側で吸収しようとする「フロントエンド概念」を生みつつある。

3．テキストエディタに関する議論

以上を背景として，「テキストエディタの使い易さの向上と，テキストエディタに注目した分散処理の概念」を検討する為に，より具体的な議論に移る。

3－1．従来の「使い易さ」に対するアプローチ

いくつかの計算センタで，ＴＳＳのコマンドの使用状況を調べると，通常８０％以上がエディットコマンドである事が分る。これはＴＳＳ全体の使い易さが，テキストエディタの使い易さに左右されている事を示している。

（1）「投入」

テキストの「投入」に関する使い易さは，キーのタッチ数を減らす，あるいはキーボード上の動きを減らすなど所謂「物理量の削減」を中心としたアプローチが行われている。これは，物理的に楽にするだけでなく，投入の「正確性」の増加という目的を持っている。

以下に，それぞれの分類に従って，いくつかのアプローチを整理しておく。

①キータッチ数を減らす

- コマンドキー，ファンクションキー，カタログキー
- ＴＡＢキー，リピートキー，オートリピート
- ショートハンド記法
- その他（ｈｅｎｃｅｆｏｒｃｅ，コマンド名の設計，自動番号付け，・・・）

②キーボード上の動きを減らす

- ニューメリックキー
- その他（キーシフト，シフトロック，・・・）

③キーサーチを楽にする

・ソフトキー

・マルチカラーキーボード

・その他（スペースキー，リターンキー，・・・）

（2）「修正」および「削除」

テキストの「修正」と「削除」は，ある意味でテキストエディタの使い易さの中心をなしていると思われる。これらは，それぞれのオペレーションに対する考え方の違いによる差はあるが，主に「文字列の処理方法の多様化」をアプローチの方法としており，その内容は文字列処理機能の追加，あるいはカーソル機能の強化などからなっている。

①文字列処理機能を追加する

・置換

・転写

・移動

・その他（挿入，削除，・・・）

②文字列あるいは繰り返しの指定方法を楽にする

・あいまい指定（伏字，任意指定，・・・，）

・範囲の指定

・回数の指定

・その他（間合せ，・・・）

③カーソル機能を便利にする

・カーソルキー（上下左右，↑，↓，→，←，ホーム，・・・）

・ジョイスティック，マウス

・その他（ワード単位の移動，ポインタジャンプ，・・・）

（3）「表示」および「検索」

「表示」と「検索」に対するアプローチは，「指示の方法の改善」の他に，特に「結果の表示方法に関しての改善」を中心としている。

①文字列あるいは繰り返しの指定方法を楽にする

・あいまい指定（伏字，任意指定，・・・）

・範囲の指定

・回数の指定

・その他（正順サーチ／逆順サーチ，・・・）

②表示（参照）方法を改善する

・プリティプリント

・ページ表示

・スプリットスクリーン，マルチスクリーン

・その他（スクロール速度，スムーズスクロール，ポーズキー，・・・）

（4）その他

以上，テキストエディタの機能に対応してその使い易さへのアプローチを述べたが，この他にも以下に示すような問題があり，それぞれに改善が図られている。

①コマンドの設計（コマンド名，コマンドシンタックス，・・・）

②覚え易さ（コマンド数，・・・）

③慣れの問題

④その他（ＯＯＰＳキー，コミュニケーションライン，カラー表示，・・・）

３－２．今後のアプローチ方向としての分析

以上に従来のアプローチを述べたが，それらの「流れ」としての今後の改善の可能性は，次の三つのポイントを中心になされていると考えられる。これは前に述べた「技術的インパクト」に対応している。

（1）「編集機能」の追加と「言語」としての改善など

修正および削除を中心として行われている機能の追加は，（プログラム）テキストという一種の「文章の編集」という立場からの改善と考える事が出来，これは引続き可能性を持っている。

また，指示方法の改善は，ある意味で「言語」としての問題で，いかに少ない言葉で多くの情報を正確に述べる事が出来るかといった追求がなされている。

（2）ディスプレイのより高度な使用

ディスプレイ型ターミナルのインパクトは，その早さ，静かさもさることながら，そのスクリーンの拡がりとソフトコピーという性格にあると考えられる。

現在カーソルを利用したフルスクリーンオペレーションを含めてその「二次元の運動性」が追求されているが，これは引続き改善の努力が払われるであろう。

また，「ソフトコピー」の性質は，その都度画面をユーザの必要とする対象として対応付ける事を可能にする大きな手段であり，マルチスクリーン，ウインドウなどを始めとし

て更に改善する可能性を持っていると思われる。

（3）ローカルインテリジェンスの利用

ローカルインテリジェンスの利用は，「処理の分担」と「マンマシンインタフェースの吸収」といった二つの面からの可能性が追求されている。

前者は一つにはインテリジェントターミナルに見られるテキストの部分的編集あるいはシンタックスのチェックなどの分担であるが，ローカルパワーの向上に伴ないより大きな分担が考えられている。

また，後者ではキーボード入力，一部のディスプレイ出力の実現が行われているが一寸とした改善を含めたマンマシンインタフェースの改善はローカル側の分担に移りつつある。

3－3．パーソナルコンピュータのテキストエディタ

（1）スクリーンエディタ

多くのパーソナルコンピュータのテキストエディタは，フルスクリーンオペレーションを前提としたものである。これは，カーソルによる移動およびそれによる挿入・修正・削除が行える他に画面のスクロールアップ／ダウンなどの内容も含んでいる。そして，この機能は，ユーザにもプログラムもにも公開されており，これがパーソナルコンピュータの使い易さを作っていると考えられる。

（2）ダムターミナルのプログラム

ディスプレイ型のダムターミナルのスクロール，画面消去等の画面制御の多くは，ソフトウェアによって行われている。因みにそのプログラムの内容を調べてみると，上に述べたパーソナルコンピュータのスクリーンエディタとほぼ同様の「部品」によって成っている事が分る。つまり，パーソナルコンピュータとダムターミナルの差異は，（もちろんエディタに限ってだが）製品の目的もさることながらその「同じ」部品がユーザおよびプログラムに公開されているかどうかという設計上の考え方の点にあると思われる。

3－4．プログラミングに関する考察

アプリケーションプログラムの構成を調べると，目的毎に違いはあるが，その殆どが30～50％以上を出力の編集に費やしているようである。これは，プログラミングのしやすさのある部分が出力の編集を改善する事で実現する事を暗示している。

（1）一つの例示

例として，Ｄｉａｂｌｏ（Ｈｙｔｅｒｍ）型プリンタによる出力をプログラミングの立場から考察してみる。Ｄｉａｂｌｏ型プリンタは，その基本的機能としてラインを逆順

に送る事が可能である。これは，出力編集をプログラミングするとき，出力する順序にとらわれた編集を行う必要がない事を意味している。従って，人が帳票を作成する順序とほぼ同じ順序でプログラムを記述する事が出来る。

例えば，その手順は次のようである。

①帳票の枠を作る

②罫線を引く

③適当な所から数字を入れる

④タイトルを入れる

⑤日付を入れる

⑥完成

もし，プリンタが正順にしか動かなければ，プログラムが最終の出力イメージの編集を意識して，先頭から１行分ずつ編集して出力する必要がある。しかし，（単純な事ではあるが）逆順に送れるという自由度を前提とする事により，人が仕事をするのと同じ感覚でプログラムを記述出来る訳である。

この事情は，プロッタの場合も同様で，これもペンが上下左右つまり二次元に移動出来る事による「自然さ」が利用出来る。また若干意味あいは異なるが，ＦＯＲＴＲＡＮの〔１Ｈ＋〕も同様の役割を持っていると思われる。

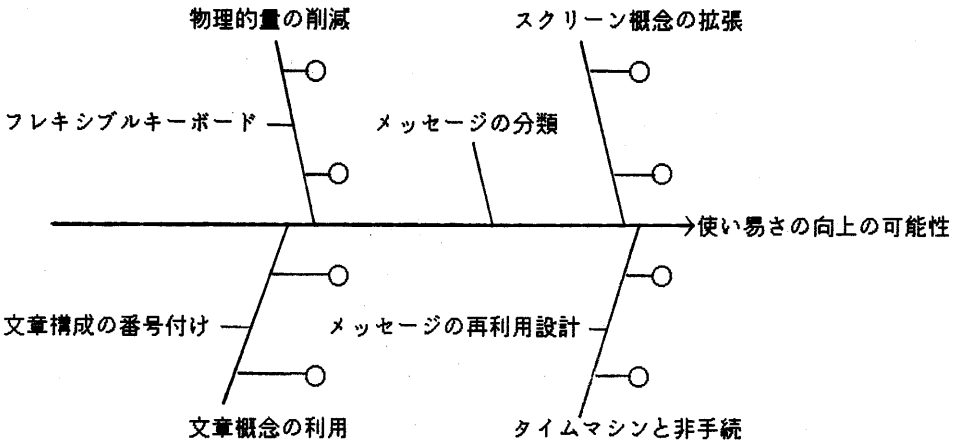
プログラムを記述する上で，人が仕事をするのと同じ感覚を反映できるという事は，他方そのプログラムの読み易さも増すという結果をもたらす。

４．テキストエディタの「展開」

以上の議論をへて，この稿もいよいよその主題であるテキストエディタの使い易さを向上する為の可能性を論ずる段階になった。

４－１．テキストエディタの使い易さの向上

テキストエディタの使い易さを向上するためのいくつかの可能性とアプローチの為の概念を検討する。使い易さの向上の可能性は以下の構造になっていると考える。



（1）物理的量の削減

物理的量の削減は，使い易さを向上する一つの大きな柱であり，その可能性は引続き追求されるべきものである。それは，前述した通り，キータッチ数を減らす，キーボード上の動きを減らす，キーサーチを楽にするといった事それぞれに可能性を含んでいる。例えば，コマンドキーの発展を考えれば，一つの言語あるいはシステムにとらわれずにより柔軟さを付ける事により，各言語毎あるいは，アプリケーション毎のキーワードを持つ「フレキシブルキーボード」ともいえるキーボードが考えられ，これによって，ＦＯＲＴＲＡＮでのテキスト入力における”ＤＥＭＥＮＳＩＯＮ”を”ＤＩＭＥＮＳＩＯＮ”に自動的に直すなどの配慮は全く不必要になる。また言語側でのチェックは不用になる。

（2）「文章」概念の利用

対象としているテキストは，プログラムという一種の「文章」と考えると，次のような対比が行える。

通常の文章	プログラム文
文字	文字
語	予約語
文	命令文
節，項，章	プロシジャ（サブルーチン）
本	プログラム

ここでの使い易さの向上は，文章を作成する際に行っている事をプログラムの作成に応用する可能性が考えられる。

例えば，すでにいくつかのワードプロセッサで行われているが，語，文といった単位での編集がある。これによって，語，文，章といった文章単位でのカーソリングとその単位での消去，移動，追加といった事が可能となる。また文章作成の時の番号付け，すなわち，１．１．とか　１．１．２　とかいったものを許す事により，サブルーチンＡのａをｂに変えるといったオペレーションの範囲指定として，１．２のａをｂにチェンジといった指定も可能となったり，サブルーチンＡの消去を１．２の消去といった指定も考えられる。

（３）タイムマシンと非手続

例えば，あるアプリケーション（富士通のＩＮＴＥＲＡＣＴ）では，一つのコマンドバッファを持っており，一つ前の入力を参照する事が出来，またそれを修正し再利用出来るという配慮を払っている。また，他では，ある程度大きいバッファの中に数時点前までのコマンドやコンピュータ出力をロギングしており，それを自由にリストし，修正して使う事が出来る。これらを考えてみると，入力したもの，それに対して出力されたものに関し，その履歴を通して過去にもどれる（一種のタイムマシンか？）という事であり，これは「非手続化」という概念に繋がっていると考えられる。

これを実現する一つの手段が「スクリーン概念」で，これは，動きが一次元から二次元に変る事により，まわりが展望出来る事に加えて，その中を自由に移動出来，また過去の履歴を参照するという事と通じている。

（４）スクリーン概念の拡張

テキストというものは，ちらっと見られればいいもの，参照・利用したいものなど，いくつかの目的に応じた特性を持っており，それらを区別する事によってより高度なオペレーションを実現出来ると考えられる。

例えば，マルチスクリーンはテキストの部分分割などに限定して利用しているが，これをより進めて，それぞれのスクリーンに特別の目的を持たせる事が考えられる。例えば，あるスクリーンは，コマンド入力のスクリーンとして，過去のコマンドをロギングしておく，またあるスクリーンはコンピュータの出力メッセージを出すなどである。因みに，許可されていないキーを押すとブザーで知らせるのも，１つの音スクリーンと考えられ，これは分ればよいだけのメッセージに対応している。

４－２．テキストエディタに注目した分散処理

（１）テキストエディタをフロントエンドとしたアーキテクチャ

前にＤｉａｂｌｏ型プリンタを例として，プログラム記述の自然さがペリフェラルの二

次元性を利用する事により実現出来る事を述べた。これと同様に，システムとユーザの間に二次元の機能を持ったスクリーンエディタを介在させるアーキテクチャを採用する事によって，一方ではユーザ側はこれを通してシステムを見る事が出来，すべてのメッセージをエディタの対象とする事が出来るし，他方プログラム側はその機能を利用する事によるプログラム記述の自然さ，簡潔さを得る事が出来る。

（2）マンマシンインタフェースの処理はローカル側で考える

テキストエディタの使い易さの向上として，ホスト側でフルスクリーン機能の追加等が行われている。しかし，そのプログラムは大きくまたホスト負荷も高い。これをローカル側（例えばパーソナルコンピュータ）で行ってみると，簡単なプログラムにより全く同様の効果を得る事が出来る。これらの事から，テキストエディタを始めとするマンマシンインタフェースの向上は，おおむねローカル側で行う方が効果的であると考えられ，（1）に述べたフロントエンドとしてのテキストエディタを含めて，多くの部分のマンマシンインタフェースをローカル側で分担するシステムアーキテクチュアがコンピュータシステム全体をすっきりしたものとすると考えられる。

5．おわりに

以上この稿では，（プログラムの）テキストエディタの使い易さの向上を中心としたいくつかの考察を行ってきた。その内容はまだ体系的といえるものではないが「使い易さ」の向上に対する構造とその可能性を分析検討するための一つのきっかけとなれば，この稿の目的は満たされると考えている。また以下にこの考察を行った過程で気の付いたいくつかの問題点を挙げておく。これも，上記の問題に関連して，検討の対象として取り上げる事によって，よりよいコンピュータシステムを実現する事になるであろう。

（1）メッセージの設計の大切さ

再利用可能なコンピュータメッセージは，入力コマンドの形で出力させるという事による使い易さを論じたが，これは，他のコンピュータメッセージについても同様で，例えば，ＨＥＬＰメッセージの中の例示を一部修正して入力として使うなどと同じ事である。また，メッセージの種類に応じた分類を付けて，それを仕事の整理に利用するなどの工夫をこらし，コンピュータシステムの使い易さを向上する事が重要である。

（3）プログラム記述の自然さに対するソフトウェアアーキテクチャ

プログラムの書き易さ，読み易さの一部は，記述の順序の「自然さ」による事を示したが，これは，ソフトウェアエンジニアリングの一つのアプローチが，出力の編集を含めて

，プログラムのアーキテクチャ，ペリフェラルといった面からも研究する必要があると考えている。

（以上）