25周年記念論文

BTRON における統一的操作モデルの提案†

坂　村　　健††

本論文では90年代の技術水準を考慮に入れたパーソナルコンピュータのための新しいアーキテクチャ体系BTRONについて，新たに提案する統一的操作モデルを中心に述べている．現在のコンピュータがエンドユーザにとってわかりにくいと言われる原因の一つとして，コンピュータの操作モデルが一定でないことが挙げられる．そこで，BTRONでは「実身／仮身モデル」と呼ぶエンドユーザのための新しい統一的な操作モデルを作った．プログラムの起動，ファイル管理，テキストエディタなど従来別個の操作モデルを導入する必要があった局面でも「実身／仮身モデル」により設計でき，このモデルが広い範囲のマン＝マシン・インタフェースの操作モデルを統一するのに有効であることがわかった．

1. はじめに

1984年6月から，われわれは VLSI 技術が十分使えることを前提としたノイマン型コンピュータの新しい体系を作るという仕事に着手した．本プロジェクトの最終目的とするところは，基本となる CPU チップをふくめたコンピュータを構成するすべてのコンポーネント，CPU チップから，OS (Operating System) MMI (Man Machine Interface) までを今一度，従来のものに囚われずに作ることである．

BTRON は TRON プロジェクト[1),2)]の中で多国語処理，ファイル管理，MMI，マシン間通信，応用プログラム間でのデータ交換，分散処理などを機能として持つパーソナルコンピュータ向けのアーキテクチャである．またそれらの実現と同時に，複数の開発者が独立に設計でき，ハードウェアの特性の違いを包括することが可能な「開かれた」アーキテクチャを目指している．

そのために，多様性と統一性を両立させるための「パーツの概念」，分散処理を具現化するための「お回し」の思想，コンピュータのさまざまな局面の操作モデルを吸収するための「実身／仮身モデル」などの新しい概念を提案した．この「開かれた」アーキテクチャによって，現在パーソナルコンピュータが社会に普及するにあたっての障害として言われている問題の多くが解決される[3)]．

TRON は現在のパーソナルコンピュータに関する分野の大部分についての更新を目指すきわめて広範囲なテーマを持つプロジェクトとなっている．そのため本論文では，これらの新しい概念のうち「実身／仮身モデル」を中心に述べている．TRON アーキテクチャ全体については参考文献 1)，2)，「パーツの概念」，「お回し」の思想をはじめとする BTRON の他の側面については 3) を参照されたい．

本論文の構成としては，2章が統一的操作モデルの必要性の提示，3章が本研究の歴史的背景，4章がモデルに求められる性質とその観点で見た既存の成果の評価，5章が新しく提案する「実身／仮身モデル」，6章がそれを取り扱うためのプログラムの説明，7章が4章の観点で見た当モデルの評価となっている．

2. 統一的操作モデルの必要性

現在パーソナルコンピュータは国内で数百万台が普及しているといわれる．しかし，未だに「使いにくい」という声が後をたたず，その 90% が使用されることもなく埃を被っているのが現状といわれる．

コンピュータを使用しにくくしている要因には多くの事が考えられるが，その中の一つにコンピュータの操作の対象としての一貫性の無さが挙げられる．人間が広い意味での道具を使う場合，「なにをどうしたら，こうなる」といった操作と結果に関する抽象的な原理，原則の理解を前提としている．たとえばハンマーのような道具に関しては，「物は支えが無ければ下に落ちる」といった，自然に身に付いた環境の物理的操作モデルによって操作方法を類推することができる．この理解すべき概念を「操作モデル」と呼ぶ．従来の多くのコンピュータシステムではこの操作モデルがまったく一貫していなかった．同じハードウェアであっても応用プログラムが変わるとまったく異なる操作モデル

† A Proposal of New Unified Model of Man-machine Interaction in the BTRON Environment by Ken SAKAMURA (Department of Information Science, University of Tokyo).

†† 東京大学理学部情報科学科

になってしまう．また操作モデルが同じであっても，システムに操作を指示する方法が異なり，わかりにくくなっている場合がある．たとえばコマンドを与える具体的な方法が，メニュ形式であるかファンクションキー形式であるかといった違いが存在している．以下このような指示を与える方法を「操作メソッド」と呼び，操作モデルと区別する．また操作モデルと操作メソッドを合わせた概念を「操作環境」と呼ぶ．

自動車などの複雑な工業製品の場合はハンマのように簡単ではないが，物理的な動作モデルが直接に操作モデルに写像されているため，どのような設計者が設計してもボトムアップ的な制約上その操作モデルは自然と統一が取られてきた．さらに同一の操作モデル上での効率と安全性を追求する試行錯誤の結果，早い時期にステアリング・ホイールとペダルを使う現在の操作メソッドに統一された．そのため統一的な教育によって訓練することができ，そのことが教材の良質化，教師の均質化，教育環境の整備，教育場所，時間の選択の自由につながり，学習を容易にしている．またそこで身に付けた技能はどのメーカの製品に対しても有効である保証があるため，学習によるメリットが大きく学習者は意欲を持つことができる．

これに対し，コンピュータではマイクロプログラムから始まり，ISP (Instruction Set Process), OS, アセンブラ，ライブラリ，高級言語，CLI (Command Line Interpreter), 簡易言語，ユーザの使用する応用プログラムまで，さまざまな抽象度で操作モデルが作られている．さらに，そのモデル間の写像も自由度が大きい．このため，エンドユーザの使用する応用プログラム操作のレベルでは操作モデルにも操作メソッドにも多くのバリエーションが生まれてしまう．この状況では一貫した教育は不可能であり，覚えるべきことの絶対量が多くなり，苦労して身に付けた技能も機種なり応用プログラムが変わると大部分が無駄になってしまう．これはパーソナルコンピュータに限らず，ワードプロセッサなどの専用機ですら問題になっている．これがエンドユーザ，特に高齢者にとってコンピュータを学習すること自体が大変であり，また意欲を持つことができないと思わせる大きな理由になっていると考えられる．

真のエンドユーザ向けのコンピュータを作るには，先にあげた自動車と同様の性格をコンピュータに持たせなければならない．しかし，一般にコンピュータの場合ではボトムアップに操作環境が統一されることはない．そこで操作モデルをトップダウンに設定し，応用プログラムなどをそれに沿って設計するようにしなければならない．このような目的で設計された操作モデルを「統一的操作モデル」と呼ぶ．BTRON ではモデル，メソッドの両面から操作環境を統一することを試みているが，ここでは操作モデルの統一について述べる．

3. 歴史的背景

操作環境を統一するためのモデルを構築する試みは，アメリカを中心に60年代後半から行われてきた．個人が使用するという特殊性を考慮に入れて設計された最初のパーソナルコンピュータは Xerox PARC の開発した Alto パーソナルコンピュータ[4]であり，その上に構築された Smalltalk-72 環境[5]が最初のエンドユーザ向け統一的操作モデルを作る試みと言える．

その後この PARC の研究を継承した形で，Xerox Star, Apple Lisa, Macintosh という一群のマシンが世に出た[6),7)]*．さらに，この Apple のコンセプトを使った GEM, Windows などの Operating Environment と呼ばれるソフトウェアが発表された[8),9)]．Noun/Verb 方式**のモードレスなコマンド体系，マルチウインドウ，ポインティングデバイス，ビットマップといった共通の高度な MMI 機能を持ち，それらを使ってデータ管理，プログラム実行を行うといった特長を持つ MMI 環境をまとめて「統合操作環境」と呼ぶことが多い．

これらを一見すると似ているようだが詳しく見ると必ずしも同一のモデルを持っているわけではないことがわかる．Smalltalk においては，操作モデルはオブジェクトオリエンテッドモデル[10]であり，操作メソッドはポップアップメニュを利用して起動することである．

Star では，操作のどのような局面でも「同じ命令はなんに対しても似た結果を生む」という原則を守った応用プログラムを作ることによって，ユーザの使用経験のない部分についても操作法を類推させようとしている．この操作モデルとしてはオブジェクトオリエンテッドモデルを採用している．操作メソッドとしてはどのような局面でもファンクションキーの組み合わせによって操作を行うという方法を取っている．たとえば文字列の移動にも，文書の移動にも “MOVE” キー

* Star, Lisa, Macintosh の詳細については，それぞれのマニュアルを参照のこと．

** 最初に処理の対象を指定し次に処理を指定する方法．

を使用する．しかし，実際には必要に応じて画面上に新たにファンクションキーを表示しているし，逆に対象によっては適用してはいけないキーもある．またソースとデスティネーションのように二つの指定が必要な動作については部分的にモードを導入しているなど例外が多い．

Lisa, Macintosh では操作方法の統一に主眼が置かれている．操作モデルはオブジェクトオリエンテッドを完全に離れ，統一されていない．たとえば，デスクトップの階層で Star と同様にアイコン*を使用しているが操作の対象指定のために使用しているだけで，操作モデルは昔ながらのファイルの考え方を採用している．しかし操作メソッドとしてはメニュー選択を採用しており局面によってメニュ内容が変化するので使用禁止などの例外導入の必要が無い．

Macintosh では応用プログラム開発を第三者に公開し，常駐の高度なユーティリティ群があるのでそれを使用した方が効率がいいということで結果的に応用プログラムの外部仕様が統一されることを期待するといった方針を取っている．このため応用プログラムをメーカ純正品のみに限定している Star, Lisa と比較した場合，多様な応用プログラムが短期間で揃ってはいるが，操作方法についてはバリエーションが多く統一性は犠牲になっている．GEM, Windows なども一種の OS として提供されるもので Macintosh と方針は同様である．

このように既存の統合操作環境はなんらかの問題点を持っている．次章ではその問題点を整理し操作モデルに求められる性質を明らかにする．

4. 操作モデルに求められる性質

統一的操作モデルを決定するということは，結果的にそのマシンを性格付けてしまうことであり，その決定には慎重さが要求される．このような統一的操作モデルに要求される性質を述べると以下のようになる．

1)　出来るだけ多くの操作局面に無理なく対応していること．

現在のコンピュータシステムでは，CLI レベルでのファイル管理（ファイルの生成，消滅，複写，移動など）とテキストファイル内のデータの編集（文字の生成，消滅，複写，移動など）などでは全く異なるモデルを使用している．これらを同じ操作モデルで扱えれば，エンドユーザは異なる複数の体系を覚える必要がなく有利である．しかし同時にこの統一に無理があってはならない．局面ごとに例外規定が多くあるようではかえって混乱を招く．また統一するために操作の手順が煩雑になることも許されない．

この点で Lisa, Macintosh はファイル管理とデータ編集が別扱いでありこの要求を満足していない．Star では対応させようとしているが例外が多いことが問題である．

2)　コストパフォーマンスが良いこと．

反応スピード，必要ハードウェア価格，習熟後の効率といったものもエンドユーザにとって重要である．統一的操作モデルの必要性もその時点での技術水準 (State of The Art) の中で考えねばならない．この点で操作モデルを統一するための試みの一つであるオブジェクトオリエンテッドモデルには問題があり，適当な価格で十分な効率を出すことができない．現時点においてはノイマン型コンピュータ上に，オブジェクトオリエンテッドモデルを乗せて文字のような単位から文書まですべてを扱うのは効率が悪い．

Smalltalk はパフォーマンスを上げるには多量のメインメモリを必要とする．Star もインプリメンテーションを Mesa 上に構築したオブジェクトオリエンテッド環境で行ったため[11]効率を実用的なレベルに引き上げるために，強力なハードウェアを必要とした．

3)　新しい応用プログラムのインストールなどのシステムの変更に柔軟に対応できること．

必要十分な応用プログラムセットを設定しそれらを統一することだけを目的として操作モデルを決定した場合，新しい応用プログラムの導入により統一が崩れる可能性がある．エンドユーザの要求の多様性を考えると，パーソナルコンピュータでは必要十分な応用プログラムセットというものは無いと考えるべきであり，応用プログラムの開発が第三者に公開され，同じ目的に対しても複数の製品が競争するというのが品質の向上のために望ましい．このようなユーザからの要求を考えた場合，統一操作モデルには十分な柔軟性が必要とされる．これに対しオブジェクトオリエンテッドモデル的な操作モデルを応用プログラムの起動に直接適用しているデスクトップ*と呼ばれる統合操作環境では，文書が決定された時点でそれに対する応用プログラムが決定される．そのため，同じ文書を異なる応

* 処理対象としての情報のまとまりを視覚的に表現したグラフィックシンボル．文書などの文房具を模したものが多い．

* アイコンを使用し机の上に文書が置かれている状態を視覚的にシミュレートした操作環境．

用プログラムで処理するといった，応用プログラムを道具として対象である文書と切り離して扱う自由度がない．

Lisa, Macintosh ではデスクトップ環境を採用しているため，ファイルはタイプを持ち応用プログラムと密接に結び付いている．ただし Macintosh では一度起動した応用プログラムからはファイル呼び出しが自由なので，これを利用して異なるタイプのファイルを取り込むことができる．

4) 自動操作プロファイル*の作成が可能なように考える．

現在実現されている Star などの統合操作環境においては自動操作プロファイル作成のための十分な機能がない．たとえば，デスクトップにおいては処理対象となる対象の指定は，ファイル名による指定はなくポインティングデバイスによる位置指定だけしか用意されていない．CLI のようにコマンドを並べて単純なプロファイルを作ることが不可能になっている．そのため，定型業務であろうがなかろうがすべての関連操作をユーザ自身が毎回指示してやることが要求される．逆にプロファイル作成を可能にしようとしたものについては名前呼びが必要になり，操作モデルと異質なディレクトリ環境を露出してユーザにパス名を書くことを要求している．

以上の問題に関しては Star, Lisa, Macintosh で状況は同じであるが，Star ではこの目的のためのCASPというプログラム環境の別仕様があり，Macintosh ではシステム全体でのファイル名のユニークネスを強制的に維持しているのでプロファイルを作りやすくなっている．Lisa ではプロファイルという考え自体を排除している．

5.「実身／仮身モデル」

以上のような要求に対し，それらを解決するべきモデルとして「実身／仮身モデル」を設計した．通常，個人が情報を生み出し伝達する過程をサポートするもっともプリミティブな媒体は紙であり，その本質は文章と図形である．これが本モデルの構築に当たっての基本的考えである．たとえば，実際の情報には数式や楽譜のように特殊な構造を持ったデータが存在するが，この構造自体を真にサポートするには専用の応用プログラムが必要となる．しかし，それを記述し他に伝えるに当たっては紙の上と同様に図形として扱えばよい．「実身／仮身モデル」の目的はこのような紙の自由度で使用できる基本的な操作環境の構築である．

「実身／仮身モデル」ではオブジェクトオリエンテッドモデルと異なりデータとそれを扱うプログラムは明白に分離されている．本章では「実身／仮身モデル」のデータの静的な関係モデルの部分について述べる．それらを操作するための基本的な手段を提供するプログラムはシステムで用意されているというのが前提である．これをシステム応用プログラムと呼んでいる．システム応用プログラムについては次章で述べる．

5.1 実　身

応用プログラムで取り扱う単位であるまとまった情報そのものを「実身」(「じっしん」と読む) と呼ぶ．実身の構造には「文章」(1次元構造のデータ)，「図形」(2次元構造のデータ)，「付箋」(一般的でない構造のデータ) の3種類の形態がある．この3種類を「基本データ構成」と呼ぶ．

文章と図形は BTRON における共通の情報交換の基盤であり，文章，図形の解釈 (表示) のためのプリミティブ (標準基本ルーチン) がシステムに常に存在し，すべてのプログラムから使用できる．このため文章，図形の解釈法は一般化していると考えられる．

これに対しアプリーケーションにより構造が決定される，解釈法が一般化されていない情報，解釈する主体 (応用プログラム) を特定しない限り無意味になってしまう情報を「付箋」と呼ぶ．今までのコンピュータの応用プログラムのデータ構造はすべて付箋にあたる．これに対し BTRON の応用プログラムは，固有のデータ構造を使用するのではなく主たるデータは一般化された文書，図形の実身で格納し，機能に係るためにどうしても基本データ構成では表現できない付加的データを集めて付箋として添付する．たとえばデータベースではレコードは文章で，インデックス情報は付箋で，という具合に蓄積される．

本モデルにおいてはこのような実身の複数組み合わせの構造をデータ構成と呼ぶ．異なるデータ構成間の変換時に，専用の変換プログラムがシステムに登録されていない場合でも (付箋データは変換出来ず特有の情報はある程度失われるが) 基本データ構成へ分解することによって主たるデータの交換が保証されるのが「実身／仮身モデル」の特長である．

そのために BTRON では文章，図形を処理する基本的な手段として基本文章エディタと基本図形エディタの2種類のシステム応用プログラムが提供される．

* コマンドプロファイル，コマンドプロシージャなどとも呼ばれる．

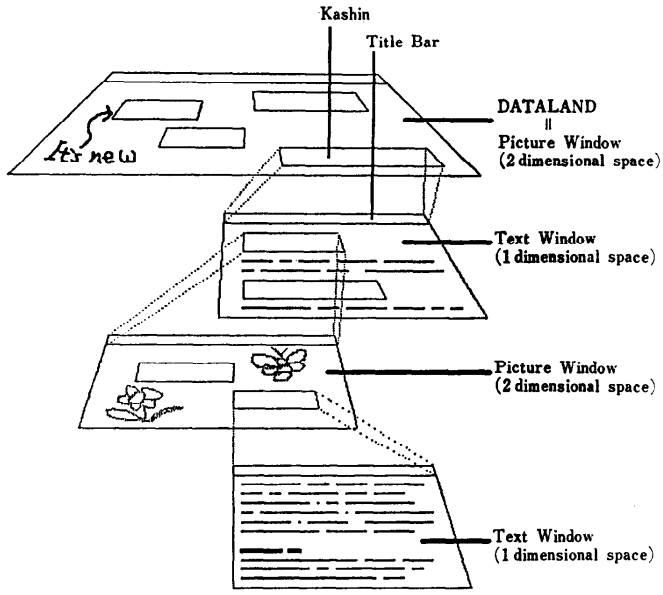

図-1　仮身の階層構造
Fig. 1　Structure of KASHIN.

5.2 仮　身

実身を扱うための手掛かりを仮身（「かしん」と読む）という．仮身は実身を指すディスクリプタである．BTRON では仮身が文字，図形と同列の情報の単位として文章，図形の中に混在することを許した．これにより，基本エディタを使って仮身自体を文字や図形と同様に取り扱うことが可能となった．基本エディタの通常の機能として，ファイル管理機能を吸収しているわけである．またある実身に含まれる仮身が次の実身を指しているという関係のネスティングによりディレクトリ的なツリー構造の情報の管理や，章，節，項といった構造を写像した情報の格納も可能になる．さらに仮身が実身を指すという関係は多対1対応が可能なので，同じ実身を異なった観点から分類する，重畳したツリー構造を実現することができる．また，文章，図形の中に混在しうるという利点を生かし，仮身を整理するリストに直接注意書きなどを図形である手書きで書き込むといったことも可能である（図-1）．

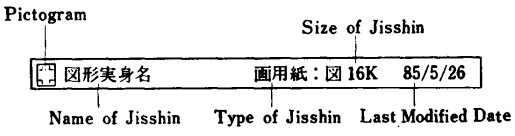

図-2　仮身の構成例
Fig. 2　An Example of KASHIN Format.

仮身は基本エディタによって，短冊状の「もの」として画面上に表示される．この短冊は対応する応用プログラムを表すピクトグラム（絵文字），その仮身の指す実身の名前，データ構成，サイズと最終修正日付からなる（図-2）．

5.3 ウィンドウ

実身の内容を表示し，応用プログラムで作業するための環境を提供するものをウィンドウという．ウィンドウはタイトルバー，内容表示面より構成される．タイトルバーは仮身と同型であり，ウィンドウを必要とする応用プログラムが起動された場合，仮身をタイトルバーとする形でその位置にウィンドウが開かれる．

ウィンドウも仮身の特殊な形態として実身中に埋め込むことができるので，図形実身を指す仮身を文章実身の中に埋め込みウィンドウを開いて挿絵とすることも可能である．さらに仮身と実身の関係は多対1対応が可能なので，ある挿絵の更新はその挿絵を使用しているすべての仮身を通して見られる内容の変化を引き起こす．

ユーザと会話可能な状態にあるウィンドウでは，応用プログラムによりコントロールパネル，スクロールバー，変形ハンドル，メニュバーなどの MMI パーツがウィンドウの周囲に表示され，これを利用して応用プログラムを操作することができる（図-3）．

実身をウィンドウ内で実際にどう表現するかは応用プログラムに任される．仮身を表示しウィンドウを用意してその中で他の応用プログラムを起動するという機能は，基本エディタによって実現されている．

5.4 実身への応用プログラムの適用法

応用プログラムへのリンクと必要なパラメータをまとめたパラメータ付箋と呼ぶ付箋が存在する．一般的にこの種の付箋はそれが指す応用プログラムと同じ名前を持っている．仮身のメニュ内には，その実身に含まれるすべてのパラメータ付箋名がサブ項目としてエントリされる．これを選択することにより，その実身を対象として応用プログラムを起動させることができる（図-4）．

このように「実身／仮身モデル」ではデータと応用

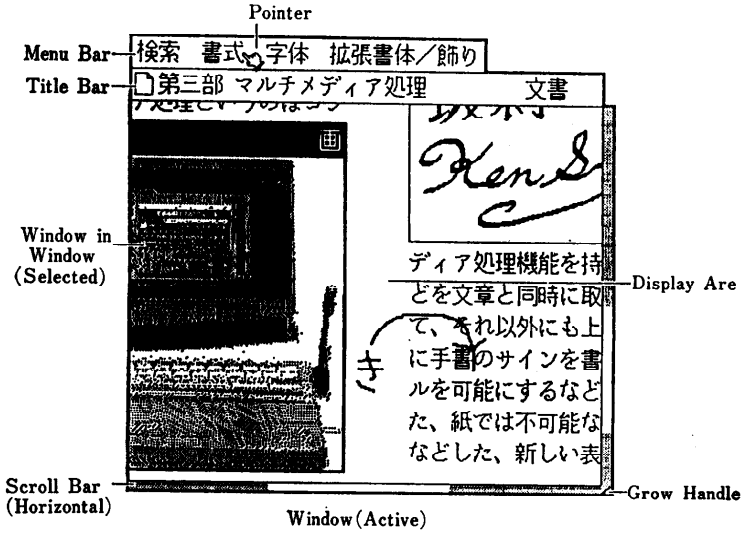

図-3 ウィンドウの構成例
Fig. 3 An Example of Window Format.

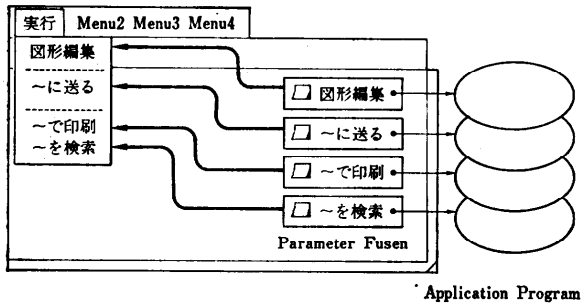

図-4 パラメータ付箋とメニュ内容の関係
Fig. 4 Relations between Parameter FUSEN and Menu Contents.

プログラムはパラメータ付箋を仲介としてつながっているだけであり，この関係は基本エディタを利用してユーザが変更することができる．パラメータ付箋を実身に添付すると，メニュにその付箋の名前が項目として追加される．操作的にはデスクトップ的であり処理する対象から応用プログラムを限定しているが，その関係が1対多対応でありユーザが変更できるという点がオブジェクトオリエンテッドモデルとは異なる．

ユーザがパラメータ付箋をコピーしてパラメータにディフォールト（暗黙値）などを記入して個別に命名してもよい．たとえば，宛先などのパラメータをすべて記述しておいた統合送信ルーチン（6.5 参照）のパラメータ付箋を作り「Mr. A へ送る」と命名すれば，この付箋を添付した実身はメニュ項目に「Mr. A へ送る」を持つことになり，この項目を選択すると即 Mr. A への送信が行われる．

6. システム応用プログラム

実身／仮身モデルに対する基本操作について述べる．システムの操作はすべて以下のシステム応用プログラムを通して行われ，BTRON 核（通常の意味の OS のレベル）を直接操作する CLI 的なものは存在しない．この部分はユーザにとって統一的操作モデルの一貫なので，効率重視の設計となっている．

特に基本エディタはファイル操作と応用プログラム起動の機能を受け持っているので必ず必要である．また，基本エディタはファイル管理機能としての必要上，同一の実身中の編集のみでなく実身を超えて仮身などのデータを移動する機能を持ち，この過程で必要ならば移転先の要求するデータ構成へ自動的に変換する．

また，検索，印刷，送信，3種の統合操作ルーチンを基本エディタから別の応用プログラムとして分割した．ワードプロセッサのような応用プログラムにおいて検索，印刷，メール送信などを行っている状態は一種のモードと考えることができるが，この機能を別プログラムとしたために，プログラムが動作しているという基本的モードとして扱えるようになった．ネスト構造を持つ複数の仮身／実身を対象として全体にこれらの操作を適用できるのも，基本プログラムと分割したためである．

以下のシステム応用プログラムのみで，BTRON はデスクトップ機能と原稿用紙的なワードプロセッサ，描画プログラム，データベースとしての使用が可能であり，個人の利用であればこれ以上の応用プログラムは必要としない場合が多いであろう．当然専用の応用プログラムをインストールした方が高機能で効率もよいであろうが，システム応用プログラムは同一の操作モデルを中心に展開された機能であり，渾然一体としてさまざまな機能を組み合わせて使用できるという自由度と，常に使っているので使用に慣れているという

利点を持つ.

6.1 基本文章エディタ

文章実身のための基本エディタ，文章エディタは，文章などの1次元に連鎖しているデータに対して使われるエディタの総称である．文章エディタはリプレースエディタで，選択した対象と入力を置き替える形で働く．1次元構造のデータ（文字列）では，そのデータの前後関係の情報がユーザにとっての本質なのでリプレースエディタを採用した．基本文章エディタは「文章編集」という名前のパラメータ付箋によって実身と結び付けられている.

6.2 基本図形エディタ

図形実身のための基本エディタ．図形エディタは，図形，表などの2次元に広がったデータに対して使われるエディタの総称である．図形エディタはオーバーライトエディタで，新たに入力したセグメントが位置的に同一の場所にあったセグメントに重ね描きされる．2次元構造のデータ（図形）では，そのデータの占める位置の情報がユーザにとっての本質なのでオーバライトエディタとした．基本図形エディタは「図形編集」という名前のパラメータ付箋によって実身と結び付けられている.

6.3 統合検索ルーチン

文字列，仮身，図形セグメントを混在して検索することのできるルーチン．サーチ，チェンジ，ソート機能を使用できる．検索対象として文字列を指定して，読みのあるオブジェクト（文字列，仮身，文字列図形セグメント）を同等に扱うこともできる．これ以外にも検索対象のパラメータに直接仮身を指定することもできる．これによって，エディタ中での検索，置換機能と条件検索で文書を捜し出すといった機能を融合できた．統合検索ルーチンは検索対象，置換対象，制限条件をパラメータとする「検索する」という名前のパラメータ付箋によって実身と結び付けられている.

6.4 統合印刷ルーチン

文字列，仮身，図形セグメントを混在して印刷することのできるルーチン．統合印刷ルーチンは用紙指定，印字品質指定，部数などをパラメータとする「印刷する」という名前のパラメータ付箋によって実身と結び付けられている.

6.5 統合送信ルーチン

文字列，仮身，図形セグメントを混在して電子メールとして送信することのできるルーチン．統合送信ルーチンは宛先指定，部数，回覧指定などをパラメータとする「送る」という名前のパラメータ付箋によって実身と結び付けられている.

7. 評　価

最後に，先に挙げた統一的操作モデルに求められる性質に関して「実身／仮身モデル」を評価する．以下の項目番号は4章のそれに対応している.

1) 「実身／仮身モデル」ではファイル管理機能も編集動作として考え，基本エディタに融合している．また，ユーザの扱う基本的な情報の形態を，データの構造上融合に無理があるということで文章と図形に大きく分類し，その上でそれぞれを中心に融合している．この2種については互いにネスト可能な独立した二つの系統として実現し，その違いをユーザに明白に提示しているため操作に無理がない．さらに，その時点で可能な動作を適切なコマンド名でメニュ表示しユーザに選択させる方式のため「指示が可能でありながら，行ってはならない動作」といった例外も生まれない.

2) オブジェクトオリエンテッド的な概念を随所に使用しているが，実際の実現には従来通りのデータとプログラムを分離した手法を使用する．また，関連が深い機能が独立した応用プログラムとして小さくまとまるため，強力な一つのプログラムですべてを統合するのに対しコストパフォーマンスが良い.

3) データと応用プログラムはパラメータ付箋を仲介としてつながっているだけであり，この関係は基本エディタを利用してユーザが変更することができるため，新しい応用プログラムのインストールなどのシス

表-1 統合操作環境の操作モデルとしての評価

Table. 1 Evaluation of Integrated Environment from the Viewpoint of Interaction Model.

	Smalltalk*	Star	Lisa	Macintosh	BTRON
1) 操作の局面	○	△	×	×	○
2) コストパフォーマンス	×	△	○	○	○
3) システム変更	○	×	△	○	○
4) 自動操作	○	△	×	△	△

(○: good, △: fair, ×: poor)

* Smalltalk 自体は当初のエンドユーザ向けという目的から外れ，専門家向けのプログラミング言語／環境として Smalltalk-80 に至っている．そのため比較の基準が他の3者とは異なる場合が多い．3),4)がよしとせされているのは言語としての特質上当然であり，言語としての側面を見ることがエンドユーザ向きとは言い難い以上，ここでの比較は参考までのものである．たとえば，インストールしていないオブジェクト表記の保存については既存のファイルシステムをそのまま使用しており，1) について問題がある．しかし，言語として OS の機能をトランスペアレントに使用するための機能なので目的が違うともいえるので無視している.

テムの変更に柔軟に対応できる.

4) 仮身というディスクリプタが存在しパラメータとして使用できるので,自動操作プロファイルの作成時に,選択によらずに実身の特定が行える.また,パラメータ付箋というかたちでパラメータを一括して扱うことができる.さらに,統合検索ルーチンを利用することによって名前や内部データによる条件検索の結果として実身を指定できる.これらの特長により,データを受け取り仮身に指定したパラメータに従い処理してデータを返すといった応用プログラムの自動操作が可能となる.しかし,インタラクティブな応用プログラム操作の自動化については操作シーケンスをレコーディングするといった単純な方法を除いては,内容についての十分な知識を持ち副作用の予想をしないと対応できない.可能な方法は,動作をもっともよく知る応用プログラム作成者自身が,ユーザとインタラクティブに交信しながらプロファイルを生成するルーチンを個別に提供することである.BTRON はそのサポートを行う.

以上の評価をまとめたのが前ページの表-1 である.

8. おわりに

BTRON というのは直接パーソナルコンピュータの外部仕様を定義しているのではない.中心となる思想,概念の一本化と表面的なデザインの多様化を両立させようというアーキテクチャである.あるマシンのソフトウェアも含めた外部仕様という形で定義してしまえば操作の統一は簡単であるが,メーカにとっては独自性が出せないということで問題があるであろうし,すべてのエンドユーザが同じソフトウェアを使うということは不自然である.また,仕様を一本化しても結局は必要に応じて新しいアプリケーションが導入されるであろうし,それを禁止すれば進歩の余地のないアーキテクチャとなってしまう.

説明のときに使用するアナロジーであるが「ソフトウェアも含めたコンピュータを自動車のようにしたい」というのが BTRON の目標である.自動車はその中心となる思想,概念が一本化され,教習所で訓練を受ければどのメーカの自動車にも乗ることができる.それでいて意匠,性能,目的などの多くの点で多様性が存在し競争と共存が行われる健全な市場が形成されている.

BTRON 核と外核 (ユーティリティ群) の提供と規格によって,さまざまな応用プログラムに対しても可能なかぎり操作モデルと操作メソッドを統一する,その上で,統一的操作モデルについては努力しての習得を前提とし,応用プログラムの違いによって本質的に操作モデルを変えざるを得ない部分については逆に学習の容易さを最優先にする設計をすることとする,というのが BTRON での方針である.この方針により,初心者としての学習の容易さと熟練者となってからの効率の良さのトレードオフに対し明確な回答を出すことができた.学習の期間が全体の使用期間に比べて短ければ,熟練者となってからの効率の良さを目指すべきであり,自動車と同程度の工業製品であるコンピュータでは,努力なしに習得できるはずはないということを前提としてもよいはずである.しかしこのためには有効な統一的操作モデルの存在が必要である.本論文で提示した「実身/仮身モデル」が,そのような BTRON の目標を実現する基礎となるものと信じている.

参考文献

1) 坂村 健: TRON トータルアーキテクチャ,アーキテクチャワークショップインジャパン'84.
2) 坂村 健: リアルタイムオペレーティングシステム-1-TRON, 第24回オペレーティングシステム研究会 (1984).
3) 坂村 健: BTRON スーパー・パーソナル・コンピュータ,計算機アーキテクチャ研究会資料 57-4 (1985).
4) Thacker, C. P., McCreight, E. M., Lampson, B. W., Sproll R. F. and Boggs D. R.: Alto: Personal Computer, Computer Stuctures, Principlesand Examples (1982).
5) Krasner G.: Smalltalk-80: Bits of History, Words of Advice, Addison-Wesley Publishing Company (1983).
6) 坂村 健: 高機能ワークステーションのアーキテクチャ,情報処理, Vol. 25, No. 2, pp. 93-102 (Feb. 1984).
7) 坂村 健: コンピュータ・アーキテクチャ,共立出版 (1984).
8) "A Graphics-Based Environment for the PC", The Seybold Report on Professional Computing, Vol. 3, No. 10, pp. 3-9 (Jun. 1985).
9) "Microsoft Windows: A Peek at Future Operating System", The Seybold Report on Professional Computing, Vol. 4, No. 1, pp. 1-6 (Sep. 1985).
10) Goldberg A. and Robson D.: "Smalltalk-80: The Language and its Implementation", Addison-Wesley Publishing Company (1983).
11) Curry G. and Ayers R.: Experience with Traits in Xerox Star Workstation, IEEE Transaction on Software engineering, Vol. SE-10, No. 5, pp. 519-527 (Sep. 1984).

(昭和60年7月1日受付)