グラフィクスとCAD 44－3
（1990．5．18）

アプリケーション・チュートリアル システム

小野 眞

日本アイ・ビー・エム株式会社
東京基礎研究所

アプリケーション・ソフトウェアの使い方の技術を身につけるためのチュートリアル・システムについて述べる。まずシステムに必要とされる機能として、アプリケーションの振る舞いを多面的に見せる必要性を述べ、その一例として検算機能・略図機能を提案する。つぎにこれらのチュートリアル・システムを一般的に作成するために必要な基盤システムについて述べる。方法としてメッセージ・フックを紹介し、その一般化として代行オブジェクトを提案する。

Application Tutorial System

Makoto Ono

Tokyo Research Laboratory, IBM Japna, Ltd.
5-19, Snabancho, Chiyoda-ku, Tokyo 102

A tutorial system on how to acquire the skill needed to use application software is described. The importance of how the function is displayed by the application is described from several asspects, and the check account function and map model are presented as examples. The second part of the paper deals with the implementation. The message hook function is used to develop the tutorial system, and an extension of this function, named object delegation is also described.

1．はじめに

最近のパーソナル・コンピュータの普及にともない、数多くのアプリケーション・ソフトウェアが利用できるようになってきた。数年前まではある機種で利用できるワードプロセッサはたかだかひとつかふたつであったのに対し、現在は選択に困るほどのソフトウェアがしのぎをけずりあっている。またかつては、パーソナル・コンピュータといえども、どちらかといえば単機能的使い方、たとえばワードプロセッサ専用であるとかデータ検索専用といった使い方であったのに対し、最近はひとりのユーザが複数のアプリケーション・ソフトウェアを利用する(しなければならない)状況になりつつある。

したがってユーザはひとつのシステムの使い方を時間をかけて習得しさえすればよいのではなく、短時間で新しいシステムを(次から次へと)マスターしなければならなくなってきた。このときある統一のとれたシステム間の移動であれば前システムで取得した技術を簡単に応用することも可能だが、たとえばメーカの異なるシステム間ではそれも難しい。ましてや今までワードプロセッサしか使ってこなかったユーザが、新しく表計算やデータベースなどを利用しはじめるときには、かなりの努力が必要とされる。

これに対応して各ソフトウェアメーカもノービスユーザを対象としたユーザインタフェースの開発や、ユーザインタフェース管理システムの研究等を行ないはじめている。ところがユーザはシステムを利用する中で、だんだんと熟練してゆくため、ノービスのみを対象としたインタフェースではすぐに使われなくなってしまう（実際単純なWYSIWYG式のワードプロセッサは短時間で文書を仕上げなくてはならないプロフェショナルユーザにはあまり好まれていない）。またどのようなシステム間においても獲得した技術を転換できるような体形(Tetzlaff 1987)は、いまだに利用可能にはいたらない。

そこで必要となるものがユーザ教育と呼ばれているもので、新しいアプリケーション・ソフトウェアを利用する技術を無理なく、そして効率的に獲得するための教育を指す。

本報告では、この教育を手助けするためにコンピュータを利用する方法(これを私達は以後アプリケーション・チュートリアルとよぶ)について検討する。またここで提案するシステムを充分に利用可能とするために必要なアプリケーション・システムのモデルについても紹介する。

2．アプリケーション・チュートリアル

アプリケーション・チュートリアルに要求される機能を統括的に整理することは難しい。また個別のアプリケーションにすべてうまく適用できるかどうかは、アプリケーションの意味論に発展しかねないので、ここでは割愛し、ユーザがアプリケーション・ソフトウェアのモデルも作る課程を考慮しながら検討する。

まず従来のシステムにおいて多く実現されている機能を、ふたつ紹介する。

[1] 説明機能

アプリケーションの機能等の説明を行なう機能。基本的に一方通行のコミュニケーションになる。しかしながら、たとえば現在ワードプロセッサの使い方ビデオがよく売れていることからもわかるように、実際そのシステムで何ができ、どうすればよいのかを手順よく説明しているものは、本(マニュアル)と比べると、無理なく理解するという観点では優れて

いると考えられる。

[2] テスト機能

説明したものをどの程度理解しているかを試すために、あらかじめ答えの用意してある問題を解かせる機能。単純な操作方法等を教えるならともかく、一般的にはオーサリング・システムを伴わないと容易には作れない。作り方によっては無機質な感じのするシステムになりかねず、強制でもされない限り使われないおそれがある。

上記の機能のみでは、アプリケーション・ソフトウェアがあくまでもブラックボックス的な存在として位置づけられており、ユーザの中にアプリケーションのモデルが構築しにくいままである。

私達は、ユーザにモデルを構築しやすくさせるために、次の三つの機能を提案する。

[3] エージェント機能

自分、または他のユーザの代行をする機能であり、上記[1], [2]の拡張と考えられる。典型的な使い方は、「間違えてみせる」ことである。たとえばワードプロセッサの"挿入モード"は、メタファ等では非常に説明しにくい。これは一度ビデオ等で説明した後、実際の例文作成時にエージェントがわざとモードを間違え、文を上書きして消す様子を見せる、すなわちエージェントが間違うことにより、なぜそうなったかをユーザに気づかせる。

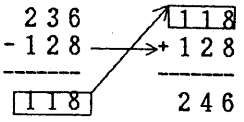

図1 検算

[4] 検算機能(フィードバック)

自分の操作の結果がどうなっているのかを、別の側面から提示する機能。これは算数を教えるときに、図1のように常に横で検算を見せることで、自分が何を勘違いしているかに気づかせるという方法(佐伯 1986)を一般化したものである。たとえばデータベース定義や表計算定義においても同様に、他の側面からフィードバックを提供する機能は、細かく説明するよりも、理解しやすいと考えられる。

[5] 略図機能

ユーザは通常与えられた世界、この場合はアプリケーション・ソフトウェアを理解するにあたり、まず簡単なモデルを頭の中につくると仮定する。この場合このモデルは純粋なモデルというよりは、その世界を何か別の世界でみたてたものというレベルであろう。そしてそのモデルを基盤に、「どのような操作を加えるか・どのような視点からながめてみるか」を考える。佐伯(1986)はこのモデルを略図と呼んでいる。多少観念的になるが、算数の勉強のときに現実世界とモデル(たとえば数式)の間におはじきやタイルのレベルの略図をおいて思考を開始するのに相当する。アプリケーション・チュートリアルにおいても同様に、オンライン・データベースのモデルと実際の発注業務や帳簿の間にたとえば架空の伝票でモデル化するなどの手法は、システ

ムの理解を助けるばかりでなく、システム設計者のモデルと利用者のモデルのくいちがいを減少させる効果もあると考えられる。

[4], [5]を自動的に作成することは現状では不可能である、という以上にこういったチュートリアルを作ることは、実際のアプリケーション・プログラムを作ることと同様以上の労力を必要とするだろう。そこでつぎにチュートリアル・システムをより一般的に作成できるようにするための仕組みとして、アプリケーション作成時にどのような機能が必要かを検討する。

3. アプリケーション・チュートリアルの構成

アプリケーション・ソフトウェアが作成されている環境に応じて、チュートリアルシステムの作り方も異なってくる。

[1] シュミレーション方式

アプリケーション・ソフトウェアの機能の中で、必要なものをすべてシュミレートする方式。既存のアプリケーション・ソフトウェアには特別な機能や制限を設ける必要のない反面、

(1) チュートリアル・システムの構成がかなり複雑になる（あつかうことのできる機能に実質上限界ができる）。
(2) ユーザが、定義されていない機能を試すことができない。
(3) アプリケーション・ソフトウェアとのバージョンの整合性が容易でない（たとえばアプリケーションの方で新しいデバイスを使えるようになったとしてもチュートリアルでも使えるようになっていないなど）。

という欠点がある。

[2] メッセージ・フック方式

多くのウインドウ・システムにおいては、アプリケーション・ソフトウェアは物理的にユーザの操作を読み取るのではなく、論理的なイベント(メッセージ)を受け取ることで動作する。この場合、直接ユーザの操作をアプリケーション・ソフトウェアにわたすのではなく、いったんチュートリアル・システムでフックし、解釈した後ふたたび送りなおす。または必要なイベントを生成し、これをアプリケーションに送ることにより、チュートリアルに必要な機能を利用することができる(図2)。

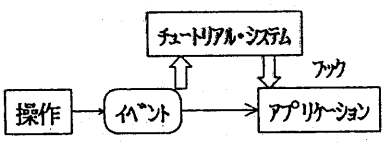

図2 メッセージ・フック

以上のふたつを整理してみると図3のように考えられる。すなわちアプリケーション・ソフトウェアをふたつの部分、ユーザインタフェース管理(UI)とデータ管理(DM)に分ける。[1]では(UI)も(DM)もすべてチュートリアル・システムでシュミレートするのに対し、[2]では(DM)は現存のアプリケーションをそのまま用い、(UI)のみを対象にユーザ操作の解釈・シュミレートを行なう。実際には(DM)を充分理解した上でないと、エラーへの対応等ユーザ操作の模倣だけではうまくいかないことも多い。とはいうものの[2]の方が

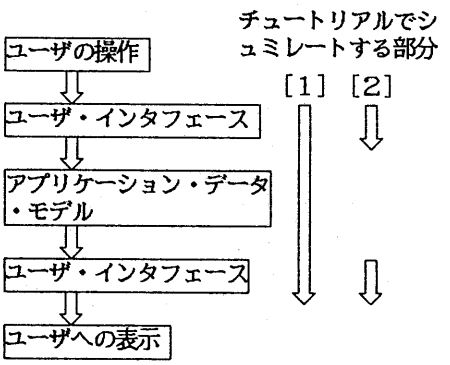

図3　アプリケーションの構造

圧倒的にチュートリアル・システムを作りやすい理由は、(UI)に限られているとはいえ、ある程度論理的かつ統一されたプロトコルでシステムが稼働しているからである。

そこでこのプロトコルをもう少し拡張する方法についてさらに検討する。

[3] ＭＶＣモデル

Smalltalk-80には、モデル・ビュー・コントロール(ＭＶＣ)構造と呼ばれる基本的枠組みがある。ビューとコントロールは協調してモデルの表示・操作を行なう(図４)。これは先の(UI)と(DM)の対応関係にもなっている。このときビューとモデルの間には従属(dependency)関係が存在し(ビューはモデルに従属している)、この二者の間には changed / update というプロトコルが存在する。すなわちモデルはその内容が変化する(させられる)と自動的にchangedというメッセージを受け取る。このメッセージを受け取ったモデルは自分に従属している全てのビューにupdateというメッセージを送り、updateを受け取ったビューはモデルを参照し、モデルの情報を表示する。

その他コントロールは全てメッセージによってモデルを操作していると考えると、これらのメッセージをフックし、シュミレートすることができれば，単純ではあるが(ユーザインタフェースのシュミレートではなく)モデルをチュートリアルから制御できると考えられる。

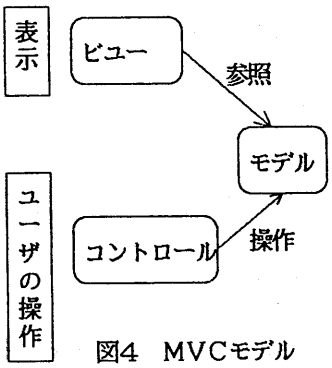

図4　ＭＶＣモデル

[5] 代行オブジェクト

前記のモデルは全ての制御をメッセージによって行なうことを基本としている。この場合、単なるメッセージフックではフックするオブジェクトが全てを制御するため複雑になりすぎる。そこで以下に述べるオブジェクトの代行(delegation)を提案する。基本的にはメッセージフックを一般化したもので図５に示す通り対象となるオブジェクトへのメッセージは、いったんこの代行オブジェクトに転送され処理される。このメッセージを引き続きもとのオブジェクトに送るかどう

かはこの代行オブジェクトの判断による。ここで注意すべき点は、代行オブジェクトの代行を定義してもかまわない点である。すなわちメッセージフックのチェーンが自由に定義できる(図6)。
このような機能をもったベースシステム上のアプリケーション・ソフトでは各部分(オブジェクト)の間のプロトコルを公開することにより、比較的容易にチュートリアルシステムを構築できるだけでなく、実行時にメッセージの流れを制御できる柔軟なシステムを設計できる。

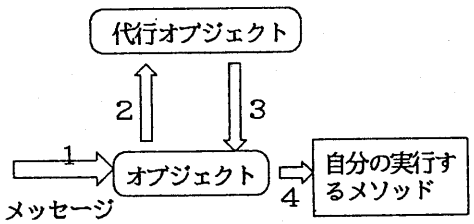

図5　代行オブジェクト

4．おわりに

アプリケーション・チュートリアルに要求される機能として単なるアプリケーションのの説明だけではなく、多面的に操作できる環境が必要であることを述べ、その例として検算の機能・略図の機能を述べた。

つぎにそれらの機能を構築する方法として、全てをシュミレートするのではなくできるだけアプリケーション・ソフトウェアを利用する方法として、メッセージフックの手法を述べた。既存のウインドウシステム等ではまだユーザインタフェースのフックによるシュミレートしか行なえないが、将来アプリケーションが基本的にメッセージ交換によって構築された場合、代行オブジェクトという考え方を導入することによって、より一般的にチュートリアル・システムを構築できることを述べた。

現在私たちは、IBM PS/55上に独自のメッセージ交換環境を試作してチュートリアル・システムの構築を行なっている。今後は、既存のウインドウ環境(たとえば X-Windows, OS/2 Presentation Manager等)において、ユーザメッセージを用いたうえでの、考察も行ないたい。

現在フライトシュミレータなしの航空機は存在しないだろう。めまぐるしく発展するソフトウェアにおいても、使い方の技術・そして教育は今後大きな課題となっていくだろう。

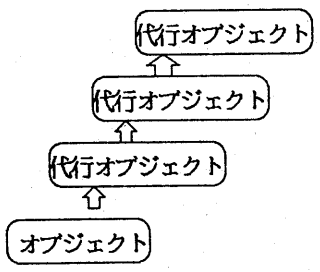

図6　代行オブジェクトのチェイン

参考文献

[1] 佐伯胖:コンピュータと教育. 岩波新書 (1986)

[2] Tetzlaff, Linda: "Transfer of Learning: Beyond Common Elements", ACM CHI+GI 1987 Proceedings (1987), pp.205-210

[3] Mack, Robert L., Lewis, Clayton, & Carroll, John M.: "Learning To Use Word Processors: Problems and Prospects", ACM Transaction on Office Information Systems, Vol. 1, (1983), pp.254-271

[4] Casner, Stephen & Lewis, Clayton: "Learning About Hidden Events in System Interactions", ACM CHI+GI 1987 Proceedings (1987), pp.197-203

[5] Akscyn, Robert, Yoder, Elise, & McCraken, Donald: "The Data Model is the Heart of Interface Design", ACM Proceedings of CHI '89 (1989), pp.115-120

[6] Goldberg, Adele & Robson, David "SMALLTALK-80 The Language and its implementation" (1983), Addison-Wesley