ソフトウェア工学　80－19
（１９９１．７．１９）

CASEツールのU/Iに導入したオブジェクト指向的操作手順

新田 稔
ＳＲＡ 先端技術開発部

ダイヤグラムを用いた分析/設計の方法を支援する上流CASEツールには、ユーザ・インターフェイス(U/I)にGUIを用いたものが多い。我々はGUIの長所を活かすとともに操作に一貫性を持たせるため、「まず操作対象を指定し、次にその対象にメッセージを送ることで操作を完了する」というオブジェクト指向的な操作手順を、開発中のツールSERA-DEのU/Iに採用した。本報告では、オブジェクトの指定方法、オブジェクトの階層、およびメッセージの発信方法について述べる。特にSERA-DEではポインタの動きによってメッセージを発信することが特徴的である。また既存のCASEツールとの操作手間の比較についても報告する。

Object-Oriented-like Operation Sequence in the UserInterface of a CASE Tool

Minoru Nitta
Software Tools & Technology Group
Software Research Associates, Inc.

Many CASE tools which support various diagramming methods use GUI. We have emploied Object-Oriented-like operation sequence -- select an object first, and then send a messege to it to complete an operation sequence -- to emphasize the merits of GUI and to unify the operations in our new CASE tool *SERA-DE*. In this report, we will discuss object selection, object hierchy and messege sending mechanisms. One of the characteristic features of *SERA-DE*'s U/I is the message sending by pointer movement. We also report a comparison of the operation storoke number with two commercial CASE tools.

はじめに

最近、ソフトウェア開発現場でCASEツールが使われるようになってきた。このうち要求分析や基本設計などの上流工程を支援するツールのほとんどには、データフロー図やER図などのダイヤグラムを用いる技法/方法論[1]が採用されており、ダイヤグラムを取り扱うため、そのユーザ・インターフェイスには、ウィンドウやメニュー、ポインタなどを用いたグラフィカルなインターフェイス(以下GUIと呼ぶ)が取り入れられている。

しかし中には、旧来のコマンド・ライン形式のユーザ・インターフェイスを表面的にGUIに転向しただけで、GUIの特徴をうまく活用できていないものや操作に一貫性がとれていないものも見られる。コマンド・ライン形式のユーザ・インターフェイスとは

コマンド名　　パラメータの並び...

という字句の並びを入力することにより、処理を指示するものである(たとえば[2])。

コマンド・ライン形式では、全ての操作が必ずコマンド名の入力から始まる。コマンド名の入力はGUIではメニュー項目の選択に相当するが、コマンド・ライン形式の入力順序を直接あてはめてしまうと、何を行なうにもまずメニュー項目を選んでからということになる。これでは、たとえれば机の上にある雑誌を捨てるとき/片付けるとき、雑誌を手にとる前に「捨てる/片付ける」と宣言するようなもので、操作対象が見えていて触ることができる(Look and Feel)というGUIの長所が活かされない。

また、コマンドの種類は限られているからメニュー項目を選ぶことで対応できるとしても、いろいろな値をとるパラメータの入力には、たとえばウィンドウの一部に「～パラメータを入力して下さい」というプロンプトが現れ、ユーザがそれを読んでからキー入力やポインタでの指示を行なうという手順になってしまう。この入力手順には「現在～パラメータの入力受け付けモード」というモードの種類が増えてしまうことと、手順の主導権がツール側に握られてしまうという弊害がある。一般に、モードの種類が増したり、ユーザが何かの入力を催促や強制される機会が多くなると、ツールは使い辛くなってしまう[3]。また操作の種類によってパラメータを入力するタイミングが異なり、操作の一貫性が欠けてしまうこともある。

さらに、コマンド・ライン形式の「発見的仕掛けを組み込み難い」という欠点もそのまま受け継いでしまう。発見的仕掛けとは、ユーザがツールを使っているうちに、何かのきっかけから、些細ではあるがそのユーザにとっては便利な処理手順(たとえばショート・カット)を見つけていくことができる仕掛けである。発見的仕掛けによって、分厚い操作マニュアルを回避したり、ツールの初心者とベテランがそれぞれの慣れに合致した操作手順をとることができる。

コマンド・ライン形式の操作手順はファンクション指向的であると言える。コマンド・ライン形式は、ユーザの前に文字端末しかなかった時代には手軽で効率的なものであっが、ハードウェア、ソフトウェアの進化によりGUIが普及してきた現在、CASEツールにはGUIの特長を十分に活用した操作に一貫性のあるユーザ・インターフェイスが必要である。我々は、これには「まず操作対象(オブジェクト)を指定し、次に処理内容を指示する(メッセージを送る)。メッセージの発信により一単位の操作が完了する」というオブジェクト指向的な操作手順が適当であると考えており、現在開発中のSERA-DE(ダイヤグラム法を用いたソフトウェアの要求分析や設計を支援するツール)に、この操作手順を持つユーザ・インターフェイスを採用している。以下、SERA-DEのオブジェクト指向的操作手順について説明する。なおSERA-DEはルールを与えることにより種々のダイヤグラム法を支援するツールとなるメタ・ツールである

が[4]、ここでは説明をわかりやすくするためにデータフロー図[5]の作成を想定している。

1.SERA-DEのオブジェクト指向的操作手順

1.1.オブジェクトの指定

オブジェクト指向的操作手順は、まず操作対象(オブジェクト)を指定し、次に処理内容を指示する(メッセージを送る)という手順である。SERA-DEではポインティング・デバイスとして3ボタンのマウスを仮定しているが(2ボタン・マウスでは二つのボタンを同時に押すことで三番目のボタンの代用を、1ボタン・マウスではキィ・ボードのシフト・キィやコントロール・キィと同時に押すことで他のボタンの代用が可能である)、オブジェクトを指定するには、ポインタを対象の上に置いて左ボタンをクリック(ボタンを押してすぐに離す動作)するか、あるいは目的のオブジェクトを囲む矩形領域のひとつの頂点にポインタを置いて左ボタンを押し、ボタンを押したまま対角の頂点にポインタを移してボタンを離すことで行なう(図1)。選択されたオブジェクトは、その旨をユーザに示すために強調表示される。

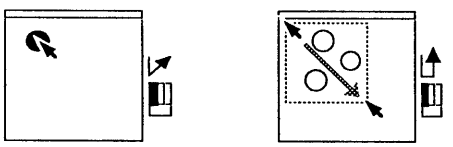
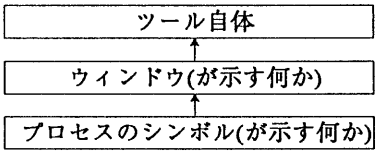


図1 オブジェクトの指定

マウスの右ボタンも左ボタンと同様の操作でオブジェクトを指定するのに用いられる。ただし、左ボタンを用いて新たにオブジェクトを指定するとそれ以前のオブジェクト指定は解除されるが、右ボタンを用いると新たにオブジェクトを指定しても以前のオブジェクトの指定は解除されないものとし、対象物が複数のとき2番目以降のオブジェクトを指定する(オブジェクト指定の拡張)ために使っている。

さて、指定されたオブジェクトは階層構造をとる。たとえばウィンドウ内でプロセスを表すシンボルが指定されたとき、オブジェクトの階層は次に示すようになる。



発信されたメッセージは、まず最下層のオブジェクトに送られる。そのオブジェクトがメッセージを解釈できればそこで処理が始まるが、解釈できなければメッセージはひとつ上層のオブジェクトに送られる。したがって、たとえば削除というメッセージはプロセスのシンボルで解釈されてそのシンボルが削除され、閉じるというメッセージは(プロセスのシンボルでは解釈されないので)ウィンドウで解釈されてウィンドウが閉じられ、また終了というメッセージは(プロセスのシンボルでもウィンドウでも解釈されないので)ツール自体で解釈されそのSERA-DEの処理が終了するということになる。

1.2.メッセージの発信

SERA-DEでは、上で述べたようにして指定されたオブジェクトにメッセージを発信する手段として、メニュー項目を選択する方式とポインタの動きによってメッセージを示す方式とを採用している(図2)。

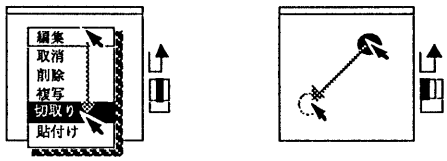


図2 メッセージの発信

メニュー項目の選択

メニュー項目を選択するには、まずマウスの中ボタンを押してメニューを出し、次にボタンを押

したままポインタを目的の項目へ移動してボタンを離す。したがってたとえば、プロセスのシンボルを作成するには、作成位置を選択するために目的の位置にポインタを置いて左ボタンをクリックしてから中ボタンを押し、ノードメニューの項目プロセスの上にポインタを移してボタンを離すという手順をとる。

オブジェクトを指定してからメニュー項目を選択するという手順をとるので、該当のオブジェクトに対して意味を持たない項目を選べなくしたり、オブジェクトに対して必須であってまだ施されていない操作をガイドするような項目の表示、またはその項目が選ばれたときのツールの処理を予見させるような項目の表示を行なうことができる。

メニュー項目の選択可不可の制御については、選択できる項目だけをメニューに載せることも考えられるが、SERA-DEでは、メッセージを解釈するオブジェクトに階層があるのでメニュー項目を頻繁に切り替えることによるユーザの混乱を避けるため、また選択できなくともどのような項目があるのかをいつもユーザに見せておくために、選択不可能な項目も薄い表示でメニューに載せている。

さてオブジェクト指向的操作手順では、同じメッセージでもそれを送るオブジェクトによって処理内容を変えるという、プログラミング言語のオーバーロードのようなことが行なえる。メニュー項目として指定されるメッセージでは、たとえばプロセスのシンボルを削除する処理とデータフローのアークを削除する処理とは同一の項目削除で示すことができ、プロセスの削除、データフローの削除という二つの項目を用意する必要はない。

ポインタの動き

ポインタの動きによってメッセージを発信する方式では、オブジェクトを選択するために左ボタンを押した後、ボタンを押したままポインタを移動してボタンが離されるまでの動きがメッセージとなる。送られるメッセージをユーザに確認するため、ボタンが押されている間、表示上のフィードバックが行なわれる。たとえばプロセスのシンボルの位置を移動するには、そのシンボルの上で左ボタンを押し、ポインタを目的位置に移動してからボタンを離すという手順をとる。ユーザへのフィードバックとして、ボタンが押されている間、シンボルの外淵を示す図形がポインタにつれて動く。

ポインタの動きによるメッセージの発信には、他のオブジェクトなどをメッセージのパラメータとするときに、それらをひとつの操作の流れの中で直接指定できるという長所がある。またメニュー項目の選択に比べて、操作の直接感が強い。

しかし一方、メニュー項目の選択ようにメッセージの明確な識別を行なうことができないため、同一のオブジェクトに対してポインタの動きで指定されるメッセージを複数設定することは、やや困難である。これを行なうには二つの方法が考えられる。ひとつはポインタの動く方向を追跡してメッセージを識別する方法である。たとえばプロセスのシンボルの上にポインタを置いてボタンを押した後、同一方向へポインタが動くとシンボルの移動となり、ジグザグに動くとシンボルの削除になるという方法である。

もうひとつはSERA-DEで採用している方法で、ポインタが動き始めるまでの時間差によって発信するメッセージを区別するものである。たとえば、プロセスのシンボルの上にポインタを置いてボタンを押し、すぐにポインタを動かすと移動のメッセージとなり、その後ボタンを離した位置にシンボルが移動する。またプロセスのシンボルの上にポインタを置いてボタンを押し、少し待ってからポインタを動かすとデータフロー作成のメッセージとなり、その後他のプロセスのシンボルの上でボタンを離すと、最初のシンボルからそのシンボルへデータフローのアークが作成される(図3)。

SERA-DEでこちらの方法を採用したのは、メ

ッセージが送られる(ポインタが動き始める)前にユーザに確認のフィードバックを出し始めることができるので、ユーザは自分の意図とは違うと気づいた時点でその操作をキャンセルすることができるからである。上の例では、データフローの作成を行なう場合、プロセスのシンボル上でボタンを押して少し待つとポインタの形が「十字」に変わり、以降プロセスのシンボルとポインタの間に現れた線がポインタの動きに連れて動いて、ユーザにデータフロー作成のメッセージを発信する旨を示す。ポインタの形が「十字」に変わった時点でボタンを離すと、データフローの作成は行なわれない。

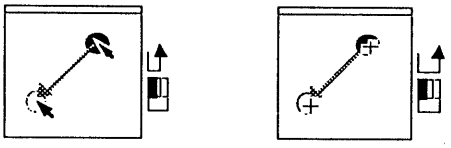


図3 ポインタの動きの多重化

SERA-DEでは、ポインタの動きによるメッセージを多重化したものとして、上の例以外に、

●ポインタを指定するオブジェクトがない位置に置いてボタンを押し、すぐにポインタを動かすと最初にボタンを押した位置と現在のポインタ位置を対角の頂点とする矩形領域内にあるオブジェクトを指定する操作となり、少し待ってから(ポインタの形が「手」に変わる)動かすと表示のスクロール操作となる。

●エンクロージャ(他のオブジェクトを囲むオブジェクトの種類)の枠の四隅のいずれかでボタンを押し、すぐにポインタを動かすとエンクロージャのサイズ変更の操作になり、少し待ってから(ポインタの形が「十字」に変わる)動かすとエンクロージャに他のオブジェクトを所属させる操作になる。

がある。いずれの場合も時間差の後の方に、頻度が少ないか、あるいは作用の大きい操作を設定している。ポインタの形が変わった後すぐボタンを離すとこれらの操作はキャンセルされる。

メニュー項目の選択にあったメッセージのオーバーロードは、ポインタの動きによっても可能である。すなわち、どのオブジェクトの上にポインタがあるかによって、同じポインタの動きが異なった処理を指示する。たとえば、データフローの中ほどにポインタを置いてボタンを押し、ボタンを押したままポインタを移動するとデータフローの折り曲げになるが、ポインタをデータフローの端点において同じ操作を行なうと、データフローの出発/到着プロセスの変更になる(図4)。

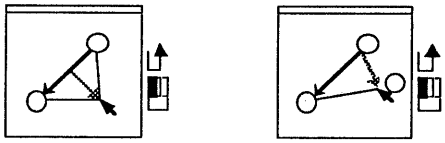


図4 ポインタの動きのオーバーロード

2.既存のCASEツールとの比較

2.1.比較の方法と対象ツール

SERA-DEでデータフロー図を作成するときの操作手順を、操作ステップ数に注目して既存のCASEツール(ツールA[6]、ツールB[7])と比較した。比較した操作は、

(1)プロセスの作成
(2)プロセスの移動
(3)プロセスの削除
(4)プロセスをデータストアに変更
(5)データフローのアークの作成
(6)データフローのアークの折り曲げ
(7)データフローの出発プロセスの変更

である。

操作の手間は、マウス･ボタンを離している状態からポインタで目的物を指してボタンを押してそのままの状態に保つ(hold:以下H)か、ボタンをすぐに離すこと(click:以下C)、およびマウス･ボタンを押している状態からポインタで目的物を指

ツール / 操作	SERA-DE	ツールA	ツールB
プロセスの作成	3ステップ プロセス作成位置を左ボタンで指定(C)した後中ボタンを押してノードメニューを出し(H)、項目プロセスを選ぶ(D)。	1ステップ～8ステップ 現在コマンドが挿入でなければ、右ボタンを押して描画メニューを出し(H)項目挿入を選ぶ(D)。シンボル･インジケータがプロセスを示していなければインジケータを左ボタンで何回かクリックし(クリックすると次のシンボルが現れる)プロセスを出す(最悪5回C)。プロセス作成位置を左ボタンで指定(C)する。	2ステップ プロセス作成位置を中ボタンで指定(C)して追加メニューを出し、左ボタンで項目プロセスを選ぶ(C)。
プロセスの移動	2ステップ(単独) 4ステップ(グループ) 単独移動は、移動するプロセスの上で左ボタンを押し(H)、目的地へ移動してボタンを離す(D)。グループ移動は、まず移動する複数のプロセスを指定するために、プロセス群の左上で左ボタンを押し(H)、右下に移動してボタンを離す(D)。この2点を対角線とする矩形領域で囲まれたプロセスが指定される。次にそのうちのひとつのプロセスの上で左ボタンを押し(H)、目的位置へ移動してボタンを離す(D)。	2ステップ～4ステップ(単独) 3ステップ～5ステップ(グループ) 単独移動は、まず現在コマンドが移動でなければ、右ボタンを押して描画メニューを出し、項目移動を選ぶ(H-D)。次に移動するプロセスの上で左ボタンを押し(H)、目的位置へ移動してボタンを離す(D)。グループ移動も、まず同様に現在コマンドを移動にする(H-D)。次に移動する複数のプロセスを指定するため、プロセス群の左上を左ボタンで指定し(C)、右下に移動して左ボタンを押す(H)。その後目的位置へ移動してボタンを離す(D)。	4ステップ(グループ) (単独移動は操作の設定なし) 移動するプロセス群の左上を中ボタンで指定(C)して追加メニューを出し、左ボタンで領域の移動を選ぶ(C)。プロセス群の右下を左ボタンで指定(C)した後、目的位置を左ボタンで指定(C)する。
プロセスの削除	3ステップ 削除するプロセスを左ボタンで指定(C)した後、右ボタンを押して編集メニューを出し(H)、項目削除を選ぶ(D)。	1ステップ～3ステップ 現在コマンドが削除でなければ、右ボタンを押して描画メニューを出し(H)、項目削除を選ぶ(D)。削除するプロセスを左ボタンで指定(C)する。	2ステップ 削除するプロセスを右ボタンで指定(C)してプロセスメニューを出し、左ボタンで項目削除を選ぶ(C)。
プロセスをデータストアに変更	3ステップ 変更するプロセスを左ボタンで指定(C)した後、右ボタンを押してノードメニューを出し(H)、項目データストアを選ぶ(D)。	1ステップ～8ステップ 現在コマンドが置換でなければ、右ボタンを押して描画メニューを出し(H)、項目置換を選ぶ(D)。シンボル･インジケータがデータストアを示していなければ、インジケータを左ボタンで何回かクリックしてデータストアを出す(最悪5回C)。変更するプロセスを左ボタンで指定(C)する。	2ステップ 変更するプロセスを中ボタンで指定(C)してプロセスメニューを出し、左ボタンで項目データストアに変更を選ぶ(C)。

表1 操作ステップ数の比較

操作＼ツール	SERA-DE	ツールA	ツールB
データフローの作成	2ステップ データフローの出発点となるプロセスの上で左ボタンを押し(H)、到着プロセスへ移動してボタンを離す(D)。	2ステップ～4ステップ 現在コマンドが挿入でなければ、右ボタンを押して描画メニューを出し(H)、項目挿入を選ぶ(D)。データフローの出発点、到着点となるプロセスを順に左ボタンで指定(2回C)する。	4ステップ データフローの出発点となるプロセスを中ボタンで指定(C)してプロセスメニューを出し、左ボタンで項目出発点-フローを選ぶ(C)。次に到着プロセスを中ボタンで指定(C)してプロセスメニューを出し、左ボタンで項目到着点-データフローを選ぶ(C)。
データフローの折り曲げ	2ステップ 折り曲げるデータフローの上でボタンを押し(H)、目的位置へ移動してボタンを離す(D)。	5ステップ～12ステップ まず屈曲点を作る。現在コマンドが挿入でなければ、右ボタンを押して描画メニューを出し(H)、項目挿入を選ぶ(D)。シンボル･インジケータを左ボタンで何回かクリックして屈曲点を出す(最悪5回C)。折り曲げるデータフローを左ボタンで指定(C)する。次に屈曲点を移動する。右ボタンを押して描画メニューを出し(H)、項目移動を選ぶ(D)。屈曲点の上で左ボタンを押し(H)、目的位置へ移動してボタンを離す(D)。	3ステップ 折り曲げるデータフローを中ボタンで指定(C)してデータフローメニューを出し、項目移動を選ぶ(C)。目的位置を左ボタンで指定(C)する。
データフローの出発プロセスの変更	2ステップ 現在の出発プロセスからデータフローが出ている上で左ボタンを押し(H)、新たに出発点とするプロセスへ移動してボタンを離す(D)。	2ステップ～4ステップ 現在コマンドが移動でなければ、右ボタンを押して描画メニューを出し(H)、項目移動を選ぶ(D)。現在の出発プロセスからデータフローが出ている点を左ボタンで指定(C)し、新たに出発点とする左ボタンでプロセスを指定(C)する。	3ステップ データフローを右ボタンで指定(C)してデータフローメニューを出し、項目出発点の変更をぶ(C)。新たに出発点とするプロセスを左ボタンで指定(C)する。

表1 (つづき) 操作ステップ数の比較

してボタンを離すこと(drag:以下D)を各々1ステップと数えた。各操作に要したステップ数と手順を表1に示す。

ツールAの全般的な操作手順は「コマンドを選んでから操作対象やその他パラメータを指定する」というもので、コマンド·ライン形式の操作手順をそのままGUIへ転向したものと言える。選択されたコマンドは「現在コマンド」として登録されるので、同じコマンドを繰り返し用いるときには2回目からコマンドの選択を省略できる。

ツールBの全般的な操作手順は「操作対象を指定してからコマンドを選び、その後パラメータ等を指定する」というものである。本報告のオブジェクト指向的操作手順との違いは、コマンドの選択が必ずしも一単位の操作の終了ではないことと、ポインタの動きによるメッセージの発信がないことである。

2.2.比較結果

ツールAには「現在コマンド」や「現在シンボル」があるので、同じ操作を繰り返すときには有利であるが、そうでない場合には他の二つに比べて操作ステップ数が多くなる。

SERA-DEとツールBはほぼ同程度の操作ステップ数であったが、ツールBでは、操作対象を指定すると同時にその対象に有効な項目を全て列挙したひとつのメニューが現わすことで、1ステップを節約している。しかしその反面、複数のメニューに同じ項目が含まれたり(たとえばプロセスメニューとデータフローメニューの両方に削除項目がある)、項目名が説明的で長いもの(たとえばデータフローメニューの出発点の変更項目)になるなど、メニューが多少繁雑になっている。

SERA-DEには、メニューを使わずポインタの動きでメッセージを送る操作があり、そのときにツールBよりステップ数が少なくなっている。

おわりに

本報告では、我々が現在開発しているCASEツールSERA-DEに採用したオブジェクト指向的操作手順を紹介し、既存の二つのCASEツールと操作の手間を比較した。GUIの標準化は各方面で進められているが(たとえば[8][9])、まだウィンドウやメニュー、ダイヤログの表現方式が統一されているだけで、作図などのツールの本質的な処理に関わる操作手順はアプリケーションに任されている。最新のCASEツールでは、本報告で紹介したオブジェクト指向的操作手順と類似のユーザ·インターフェイスを採用したものもあるので、操作全般に対して何らかの標準や統一基準を示すことができれば、ツールを作る側、使う側にとって大きな利益となるだろう。

なおSERA-DEは、協調作業支援機構SERA[10]に含まれるツールの一つである。

参照資料

[1] J.Martin, C.McClue "Diagramming Techniques for Analysts and Programmers" Prentice-Hall 1985

[2] S.R.Bourne "An introduction to the UNIX Shell" AT&T Bell Laboratories 1979

[3] Apple "The Macintosh User Interface Guaidelines" Addison-Wesley 1985

[4] 新田, 鳥居「汎用的なダイヤグラム法表現モデルとそのエディタ」情報処理学会研会報告 91-SE-77 1991

[5] T.DeMarco "Structure Analysis and System Specification" Prentice-Hall 1979

[6] "Software through Picture" Interactive Development Environments, Inc.

[7] "CASEBench" Mentor Graphics Corporation

[8] Sun Microsystems,Inc. "OpenLook Graphical User Interface Functional Specofication" Addison-Wesley 1989

[9] Open Software Foundation "OSF/Motif User's Guide" Prentice-Hall 1990

[10] 歌代, 石塚, 新田「SERA ダイヤグラム法を用いた多人数による分析/設計支援ツール」第16回jus UNIX シンポジウム 1990