岐阜経済大学論集 28 巻 4 号（1995 年）

経営情報処理における GUI の効果と諸問題

安 田 晶 彦

はじめに

コンピュータの事務計算における利用は，コンピュータが商用に開発された時点で既に考えられていたことであった。事務計算にコンピュータが利用されるようになる前に，既に PCS（Punched Card System）と呼ばれる電気機械式の統計機が政府機関や保険会社のような大企業で使用されており，コンピュータは PCS に代わるものと考えられていたからである。

コンピュータは，PCS の計算機構の部分を電子機器に更新する形で事務計算の分野に導入されたので，事務データの初期の入力メディアはもっぱら PCS から受け継がれた紙のカードあるいは紙テープであった。なお，日本での事務計算用の補助記憶媒体は紙カードよりもむしろ紙テープが利用された[1)]。紙カードの作成には，当初は専用の機器が使われたが，後には電動タイプライターの機構を応用した機器が使われるようになった。出力のメディアはもっぱらプリンタで印字された紙の帳票であった。

このようにコンピュータ・ベースの経営情報処理は，今日のようなユーザー・インタフェース（User Interface）の問題とは関わりなく進んできたと言える。しかし，1950年代から60年代にかけて，オンライン・リアルタイム・システム（On-line Real Time System）の開発と導入が盛んになり，それに対応してデータの入出力をVDT（Video Display Terminal）で行う方法についても開発が進行していった。

紙カード（またはテープ）と紙の帳票を入出力のメディアとする形態から，ビデオ表示（Video Display）を入出力のメディアとするようにコンピュータの利用形態が変化した時点で，コンピュータにおけるユーザー・インタフェースの問題が発生してきた。

紙カード（またはテープ）を入力メディアとする場合，電動タイプライター型のカード（またはテープ）穿孔機から印字されたチェック用の帳票と原始伝票とを照合するのが，コンピュータによる事務処理の基本であったが，VDTの導入により，ビデオ表示と原始帳票とを照合する形態に徐々に変化していくようになる。

ビデオ表示による操作はデータ入力から拡張されていき，1960年代に開発されたタイム・シェアリング・システム（Time Sharing System）では様々なコンピュータ操作がビデオ表示を通じて対話形式で行われるようになった。さらに，1980年代以降にパーソナル・コンピュータ（Personal Computer）やワークステーション（Workstation）といった個人利用のコンピュータ・システムが急速に普及していくに連れて，ビデオ表示によるコンピュータ操作の重要性はますます高まっていった。

そして今日では，文字と数値のみからなる従来型のユーザー・インタフェースに代わって，GUI（Graphical User Interface）が注目されるようになっている。

本稿では，経営情報処理におけるGUI普及の効果と，GUIアプリケーションを開発する上での問題点を指摘し，GUIをどのように考えていけば良いのかを考えてみたい。

1. GUI普及の背景

コンピュータは当初，（文字どおり）計算機として開発された経緯を持つため，数値データを扱っていたが，アルファベットをコード化して表現することで文字データの処理も行うようになった。さらに，画像，音声といった多様なデータをデジタル化することで，コンピュータの扱うデータの形式はさらに多様化することとなった。

ユーザー・インタフェースとはユーザーとコンピュータとの境界面を指す用語で，ユーザーによるコンピュータ操作の仕方に関わるものである。当初のユーザー・インタフェースは，もっぱら文字と数字を主体としていた。しかし，コンピュータが画像データを処理するようになった段階で，キーボードからのコマンド（Command）入力に代表される文字によるインタフェースから，マウスを使用したアイコン（Icon）操作に代表される画像によるインタフェースへの発展が見られた。これがGUIで，それに対して従来のインタフェースはCUI（Character User Interface），あるいはCLI（Command Line Interface）などと呼ばれるようになった。

従来型のユーザー・インタフェースであるCUIの場合は，コンピュータ操作に必要な単語，つまりコマンドをユーザーが暗記している必要があった。例えば，パーソナル・コンピュータ用のOS（Operating System）として，今日でも広く利用されているMS-DOS（Microsoft Disk Operating System）では，“dir a: /p”や“copy a:¥*.* b:”といったコマンドをキーボードからタイプ入力することで，磁気ディスク上のファイル管理などを行ってい

る。このようなコマンドを覚えないとコンピュータを操作できないのであるから，オフィスでコンピュータを利用する場合も，ユーザーがコンピュータ操作の教育・訓練を受ける必要があると考えられていたのは言うまでもない。1980年代の前半にオフィスでパーソナル・コンピュータが普及し始めた当初は，「パソコンと言えばBASIC」と考えられており，BASICプログラミングの講習が盛んに行われていた。

従来型のユーザー・インタフェースは，コンピュータ操作に既に習熟しているユーザーにとっても，神経を使わせる面を持っていた。旧式のエディタ(例えばライン・エディタ）でテキストを編集してみれば分かるように，作業状態に関する情報が不足しているために，ちょっとしたキー操作ミスによって，データが削除されてしまったりすることがあるので，ユーザーは慎重にキーボード操作を行わざるを得なかった。アプリケーション・ソフトウェアに誤操作の取り消し（Undo）機能を付加することによって，こうした問題は解決されていったが，ビデオ表示を活用してユーザーの作業状態をさらに確認しやすくすることが求められていた。

このような問題を解決するための方法のひとつがGUIであったわけである。

このような問題点を克服するため，コンピュータをユーザーにとってより身近なものにする研究がなされてきた。特に有名なのはゼロックス（Xerox）社のPARC（Palo Alto Research Center；パロ・アルト研究所）で1970年代に試作されていたワークステーションAltoのユーザー・インタフェースであった。Altoの研究をもとにして1981年に製品化されたゼロックスStarでは，グラフィックス表示のためのビットマップ・ディスプレイやポインティング・デバイス（Pointing Device）としてのマウスなどが装備されていた。

メインフレーム・コンピュータの世界でも（端末サイドで）GUIは必要と考えられていた。コンピュータ操作に関する講習を必要とせずに，ユーザーがオンライン端末を利用できるようなシステムの例としては，銀行のCD（Cash Dispenser；現金自動支払機）があげられる。CDでは，分かりやすい画面表示とタッチスクリーンの活用により，コンピュータ操作の煩わしさを預金者になるべく感じさせないユーザー・インタフェースが実現されている。

しかし，こうした顧客サービスの専用端末を除けば，オフィスでのコンピュータ利用に関わるユーザー・インタフェースの問題は，メインフレームの世界よりもむしろパーソナル・コンピュータやワークステーションといった，小型のコンピュータ・システムを中心に進行していったのである。

GUIはPARCの試作型ワークステーションAltoから，ゼロックスStar，そしてアップル（Apple Computer）社の初期のコンピュータLisaなどを経て商業的に成功したMacintoshへと引き継がれ，発展していった。Macintoshでは，基本ソフトウェアがGUIを統合的に管理するようになっており，アプリケーション・ソフトウェアにもMacintosh風のGUIが求められた。ユーザー・インタフェースが基本ソフトウェアに合わせて統一化されることにより，Macintoshでは多種のアプリケーション・ソフトウェアの操作法に短期間で習熟することがより容易になったのである。

しかし，1980年代にパーソナル・コンピュータの主流を占めていたIBM/PCとその互換機では，GUIを持たないMS-DOSが基本ソフトウェアに採用されており，アプリケーション・ソフトウェアを開発する際に独自のユーザー・インタフェースを設計するのが普通であった。日本でもインテル（Intel）社のMPUとMS-DOSがパーソナル・コンピュータの主流であったので，アプリケーション・ソフトウェアの多くは，独自のGUIを持つことによって製品の特徴をアピールすることに成功していた。

このことはアプリケーションごとに異なる操作法をユーザーに強制する原

因となり，本来の目的であるデータ処理とはあまり関係のない訓練を必要とするようになったのである。パーソナル・コンピュータの場合，MS-DOS 用のアプリケーションとして日本で 1980 年代後半以降に広く普及した日本語ワープロ一太郎と表計算ソフト Lotus 1-2-3 の例をあげると，ディスクからのデータファイルの呼び出し方が，一太郎では “ESC・T・L”（エスケープキー，T キー，L キーを順番に押す）と操作するのに対し，1-2-3 では “/・F・R” と操作することになっていた。基本的なキー操作がアプリケーションごとに異なるため，ソフトウェアを導入する上ではユーザーの選択の幅が広がるが，オフィスにおけるコンピュータ利用の促進という面からは，訓練の必要性の面で問題を残していた。また，当時のソフトウェア開発企業は，こうした独自の操作法によって製品の特色を出そうとしていたので，多種多様なユーザー・インタフェースが開発されることとなったのである。

インテル社の MPU 用に開発された Microsoft Windows は，モトローラ（Motorola）社の MPU を使用していたアップルの Macintosh に対抗するもので，Windows の開発により，IBM/PC とその互換機も統一的な GUI を持つに至った。Windows は 1990 年に発売された Version 3.0 から商業的に成功し，GUI 基本ソフトウェアとしての地位を占めるようになった[2)]。

日本でもジャストシステム製の MS-DOS 上のジャストウィンドウなど，独自の GUI ソフトウェアが開発されたが，例えば，ジャストウィンドウ用のアプリケーションは，主にジャストシステム製の日本語ワープロ一太郎を始めとする同社のファミリー製品であり，ジャストウィンドウは MS-DOS 上の統合アプリケーションとしての位置づけにとどまっている。このように，国産の GUI 基本ソフトウェアは残念ながら現在のところ普及するに至っていない。

GUI が普及するに至った背景には，パーソナル・コンピュータやワークス

テーションのような小型コンピュータの普及がまずあげられる。パーソナル・コンピュータは個人利用を前提としているため，対話型のアプリケーション・ソフトウェアが主流を占めており，アプリケーション開発に際してユーザー・インタフェースに重点を置かざるを得ない事情があったのである。

そして，ユーザー・インタフェースの重要性が認識されたのは，何と言ってもオフィスにおけるワードプロセッサの普及に他ならなかった。ワードプロセッサは初めから対話型処理を前提としていたうえ，ユーザーにコンピュータの難しさをできるだけ感じさせない工夫が求められたので，ユーザー・インタフェースの改良は不可欠であった。

GUIの普及にさらに貢献したのは，図形・画像処理ソフトのような，文字と数値以外のデータを扱うソフトウェアであった。Macintoshの場合，GUIは図形処理やDTP（Desk Top Publishing）といった分野でパーソナル・コンピュータの利用を促進するため，グラフィックスを活用したWYSIWYG（What You See Is What You Get）と呼ばれる機能を早くから可能にしていた。

さらに，特に注目しなければならないのはコンピュータ・ゲームの発達である。ゲーム・ソフトウェアはコンピュータ・グラフィックス，コンピュータ・サウンドの発達を促しただけでなく，子どもにも容易に操作が可能なように，分かりやすいユーザー・インタフェースを持っていた。

ワードプロセッサや図形処理ソフト，あるいはグラフィックスを用いたゲームなどのユーザー・インタフェースは，直接操作（Direct Manipulation）と呼ばれている[3)]。

これらが総合的に作用して，GUIの発達が見られたのであった。

2. 経営情報処理におけるGUIの効果

経営情報の処理に関しては，ユーザー・インタフェースの問題は，コンピュータの操作訓練を要せずに，いかにしてオフィスの情報処理の効率化を計るかにあった。

コンピュータに関する知識については，一般に ① 情報処理の基礎知識，② 他の分野で情報処理技術を利用するための応用知識，③ 情報処理の専門知識の3段階に分かれているが，経営情報の処理では①②までの知識，あるいは通常のコンピュータ・リテラシーで十分に対応できるようなユーザー・インタフェースが求められていた。

オフィスで利用するアプリケーション・ソフトウェアの開発においては，コマンドの使用は最小限にとどめることが要請された。そこで考案されたのがメニュー方式である。

メニュー方式はテキスト・ベースのビデオ表示画面でコマンドを暗記して入力する必要がないことから，経営情報処理のアプリケーション・ソフトウェアで多用されてきた。これは，アプリケーションの諸機能を番号順，またはアルファベット（またはカタカナ）で識別し，ユーザーは数字キー，またはアルファベット（またはカタカナ）キーを押して，機能を選択するというものである。カーソル移動キーが利用できる機種の場合にはメニューの項目間の移動にカーソル移動キーが使えるようにするのが一般的である。

そして，メニュー形式は，ポップアップ・メニュー（Pop Up Menu）方式やプルダウン・メニュー（Pull Down Menu）方式へとさらに発展していく。

ポップアップ・メニューは，非表示になっているメニュー画面を特定のキーを押すことにより表示させる方式である。これはいわばテキスト・ベースのウィンドウと言うべきものであり，現在ではGUIアプリケーションの

補助的なメニューとして使われている。

プルダウン・メニューはマウス操作に適しているとされており，GUIアプリケーションで最も普及しているメニュー方式である。

現状ではユーザーが文字・数値データを入力するのにタイプライター型キーボードを操作する必要があるが，まずはメニュー方式の採用により，本来のオフィス業務とはあまり関わりのないコマンドの暗記からユーザーは解放されることになった。

さらに，マウスとグラフィック画面上のアイコンを利用して，ユーザーの目に映った画像をそのまま利用できるようになれば，さらに操作性が高まることが期待された。

ただ，タイプライター型キーボードの操作に習熟したユーザー（例えばワープロ検定取得者）にとってはマウス操作は最小限にとどめたいところであり，経営情報処理にマウスとアイコンを用いることが最良の手段であるとは必ずしも考えられていないという点には注意しなければならない。GUIアプリケーションにおいても，こうしたキーボード入力に合わせたショートカットキーを提供しているのが普通であり，マウスは基本的には初心者向けの補助的なデバイスであるという見解も存在するのである。また，従来型のCUIに慣れてきたユーザーの中には，GUIをあまり評価しない傾向があり，コマンド入力の方が処理を素早く行えるという保守的な見解も今のところ根強く存在している。

手書き文字入力[4)]や，音声認識などがまだ十分に普及していない状況で，タイプライター型キーボードによる操作が依然として重視されている以上，GUIの経営情報処理における効果は，伝統的な情報処理の能率という観点からすると，それほど大きいものではないと言うことができるのである。

しかし，GUIの効果は単にこうした伝統的な能率の観点からのみ捉えら

れるものではない。まず，ユーザー・インタフェースの統一的な Look & Feel が提供されることによる操作性の向上という観点から，GUI の効果を考えてみたい。

ディスプレイ表示のカラー化，そして高解像度化に対応して，アプリケーション・ソフトウェアの外観を見栄えのあるものにする工夫が不可欠になってきている。アプリケーション開発においては，カラー表示を活用して操作する上で見やすい画面を設計することが要求される。

対話型アプリケーションの開発において，本来のデータ処理よりもユーザー・インタフェース開発の方が手間がかかるようになったのは，やはりディスプレイ表示のカラー化・高解像度化があったからに他ならない。

さらに，アプリケーションごとに異なる外観や操作法があって，ユーザーが操作法の習得に時間をかけるのではなく，各アプリケーションに統一的な外観と操作法を持たせることによって，ユーザーにとっての操作の容易さを提供することも重要である。これは OS のレベルで GUI を統一化することによって可能となりつつある[5)]。

伝統的な観点からは，経営情報が文字情報や数値情報であるという視点と結びついているため，キーボード操作が重視されていたのであるが，画像データを扱う場合には，マウスのようなポインティング・デバイスを多用する必要が出てくる。キーボード入力を主体とした文字・数値データの入力では，むしろショートカットキーを活用する方が望ましい。しかし，文字や数値以外のデータをディスプレイ表示で確認しながら行う場合は，キーボード操作では限界がある。

現時点では大別して ① コマンド入力，② メニュー方式のキー入力，③ ショートカットキー入力（機能キー方式を含む），④ マウス操作，⑤ マウスとキーボードの併用の 5 種類のオペレーションが併存しており，コンピュータ操作の初心者には④のマウスによる直接操作が推奨されるものの，複雑なオ

ペレーションの場合はユーザーが各自のスタイルを作り，必要に応じてアプリケーションのカスタマイズを行う必要性もある。④⑤のマウス関連の操作が必要となるのは主として画像データなどの，キーボード操作が馴染みにくいオペレーションが中心である。

メディアの多様化により，オフィスでのコンピュータ処理においても文字と数値のみからなる文書や集計表ばかりでなく，チャートやイラスト，写真などを活用したコミュニケーションが求められるようになっている。GUIは，こうしたオフィスにおけるマルチメディア化との関連で捉えられなければならない。

経営情報処理において伝統的な文字と数値データから，画像・音声データへのメディアの多様化が進んでいった場合に重要なのが，これらのデータのアプリケーション間での共有化や連携である。

アプリケーション間でのデータの共有化は，CUIの時点でも考えられないことではなかった。しかし，これは基本的にはテキスト・ファイルのベースで行われていた。扱われてきたのは文書データのほか，例えば，表計算やデータベース間でデータを交換するためのCSV（Comma Separated Value）形式の数値・文字データなどがある。

しかし，このレベルでの共有化は，アプリケーション間にデータの互換性を持たせるにとどまっていた。そして，GUIとともに重視されるようになった（後述の）オブジェクト指向的な方法により，データとそれを作成・編集するアプリケーションとの統一的な扱いが確立することで，データの共有化や連携はいっそう進化することとなったのである。

GUIは伝統的な文字・数値データだけでなく，画像あるいは音声データも含む統合的な情報処理には欠かせない要素となってきている。

しかし，こうしたデータの統合的な処理は，GUIアプリケーションの機

能の豊富化をもたらし，そのことが再び操作法の習得を難しくしていくというジレンマを生み出している。開発者側は機能を豊富化することで製品の特色を出そうとするが，ユーザー側からは結局，「マニアでない限り機能は使いこなせない」という声が返ってくる。結局は訓練の必要性が出てくるため，機能の豊富化が操作の容易さと両立しない局面が出てくるのである。もちろん，統一的な Look & Feel が提供されることで，異なるアプリケーションを利用する場合でも統一的な操作が可能になるという局面は存在する。しかし，基本的には，高度な経営情報処理の方法をユーザーが習得しないとGUI の効果は現われないと言えよう。

従って，経営情報処理の分野で画像や音声データの果たす役割を明確化しつつ，GUI ソフトウェアの活用を図っていく必要があるのである。

3. GUI アプリケーション開発上の諸問題

GUI アプリケーション・ソフトウェアが普及するに従って，新たな開発環境や開発の方法論が急速に注目されるようになっている。

GUI との関連で最も注目を集めているのはオブジェクト指向プログラミングであろう。オブジェクト指向プログラミング言語として有名な Smalltalk 自体が GUI 開発を意図したものであり，オブジェクト指向を取り入れたプログラム開発環境の多くが GUI アプリケーションの開発を目的としているのである。

オブジェクト指向プログラミングの最も基本的な特徴は情報のカプセル化（Encapsulation）にあると言われているが，これは従来のようなデータ構造とそれを処理する手続きとがひとまとまりになることにより，より強固なモジュール化が可能になるという考え方である。さらに，こうしたモジュールを用いた場合に，システムの再利用もより容易になると言われている。とい

うのは，カプセル化されて一体となったデータ構造と手続きは，モジュールを利用したアプリケーションから遮蔽されているため，モジュールを変更しても，インタフェースに変更がなければアプリケーション自体に及ぼす影響は存在しないと考えられるからである。

これは構造化プログラミングにおけるモジュールの独立性のいっそう徹底した考え方であるため，大規模なソフトウェア開発も容易になると考えられている。

実際のソフトウェア開発においては，オブジェクト指向プログラミングを可能にする言語，およびその処理システムが存在しなければならず，Smalltalkやc++がそうしたプログラミング環境として有力視されている（ただし，日本ではもっぱらc++が注目されている）[6]。

また，そうしたプログラム開発環境を想定したオブジェクト指向の分析・設計論も盛んに提唱されている[7]。分析・設計のプロセスは特定のプログラミング言語を想定したものではないが，実際問題としてはやはりオブジェクト指向プログラミングの要件を満たす言語の存在が前提とされている。

オブジェクト指向はGUIの直接的なプログラミング・パラダイムとしてはやや抽象的なものである。GUIをさらに意識したプログラミング手法として，イベント駆動型（event driven）プログラミング，あるいはメッセージ駆動型プログラミングがあげられる。

従来のプログラミングではデータの入力，処理，そして情報の出力という一連の処理を想定しており，ワードプロセッサのようにユーザーのキーボード（またはマウス）操作に従って処理を行うプログラムを開発することは想定されていなかった。前者の場合はプログラマーの意図した処理手順が中心であり，ユーザーはプログラムの指示に従ってデータ入力などを行う受動的な位置づけを与えられていたにすぎなかった。後者の場合は，ユーザーの能

動的な操作が中心であり，プログラムはユーザーの操作を待って待機する受動的な存在となっている。イベント駆動型プログラミングは，後者のワードプロセッサに見られるような対話型処理のソフトウェア開発のためのプログラミングである。

このように，GUIアプリケーションを開発する上では，オブジェクト指向やイベント駆動といったプログラミングの方法論が重要であるが，標準的なプログラミング言語処理システムだけでは，GUIアプリケーションを開発することができない。SmalltalkのようにGUIの機能を内蔵した開発環境もあるものの，通常はGUI基本ソフトウェアのAPI（Application Programming Interface）を利用するためのツールキットが必要となる。ただし，Cなどのプログラミング言語からAPIを利用すると，同じCのプログラムでありながらGUI基本ソフトウェアごとに全く異なったものになってしまうという問題点がある。

その他，GUIに必要な様々なリソース（例えばアイコンやダイアログ・ボックス，それにプルダウン・メニュー）を作成するための開発ツールを使用することが多い。

従来であれば，ビジネス・アプリケーションを開発するには ① コンピュータ適用業務の知識，② コンピュータのハードウェアとソフトウェアの基礎知識，そして ③ 例えばCOBOLのような事務処理用プログラミング言語の知識の3つが最低限必要とされていた。ただ，プログラミング言語については，高水準言語に関する限り比較的短期間での習得が可能であり，主な困難は大規模なソフトウェアの構築と保守に伴う諸問題にあった。

現段階でGUIアプリケーションを開発するには，②の段階でネットワークの知識を必要とするようになったほか，③の段階でかなりの多様な知識を持たないと業務用ソフトウェアの構築を行い得ないところにまできているの

である。もしも，本格的なビジネス・アプリケーションを一から開発するとしたならば，C言語を習得した後に，GUI基本ソフトウェアのAPIを学び，C++言語を学んだ上でAPIを利用するためのクラス・ライブラリ[8]を利用しないと，本格的なGUIアプリケーションを構築し得ない状況となっている。

プログラマーがこれだけの知識を短期間に習得するのはかなり困難であり，従来のソフトウェア開発に比べて生産性も低下せざるを得ないため，より容易に開発を行うためのプログラム開発環境も必要とされている。

Windowsを例にとれば，例えばMicrosoft Visual Basicがあり，簡単なGUIアプリケーションの構築には先のプログラミング知識を用いるよりも，はるかに生産性の高いプログラム開発ができるようになっている。しかし，Visual Basicはオリジナルなアプリケーションを開発する上でのツールとしては限界があり，本格的なビジネス・アプリケーションはかなりの大規模プロジェクトとして開発されざるを得ない状況となっている。

また，特にパーソナル・コンピュータの場合，GUI基本ソフトウェアは，特定のソフトウェア企業がGUI基本ソフトウェアを独占的に供給しているのが現状である。基本ソフトウェアが標準化されれば，ソフトウェアの移植性も高まり，同一の機能を持つGUI基本ソフトを複数のソフトウェア開発企業が開発することが可能になるが，そうした標準化は難しい状況にあると言わざるを得ない。

また，ワードプロセッサや表計算ソフトウェアのようなビジネス・ソフトウェアもまた，大手のソフトウェア企業によって開発されたものを利用する外はない状況になってきている。これは先に述べたように，本格的なGUIアプリケーションをオリジナルに開発するには資金，技術力，人的資源が必要で，大手のソフトウェア開発企業でないとそうしたソフトウェアは開発で

きないからである。ソフトウェアの世界では，販売実績のあるソフトウェアほどバグの修正などが進んで使いやすくなり，そうした信頼性の高いソフトウェアほど普及するという循環がある。また，ワードプロセッサなどではデータの互換性の点から多くのユーザーが共通の製品を使用していた方が便利なため，大手企業のアプリケーションによる寡占状態が生じやすい側面を持っている。

結　び

本稿ではGUIのビジネス・アプリケーションを利用・開発していく場合の効果と問題点を明らかにすることに主眼を置いた。しかし，本稿の範囲ではGUIビジネス・アプリケーションの開発方法について，具体的な内容を展開することができなかった。この点は今後の課題としたい。

〔注〕

1) 南澤宣郎『日本コンピューター発達史』日本経済新聞社，1978年，12頁。

2) パーソナル・コンピュータ用のGUI基本ソフトウェアはMacintoshとWindowsの他にも様々な製品（例えばIBM OS/2のPresentation Manager）がある。また，主にUNIXを基本ソフトウェアとするワークステーションにおいても，X-Windowを始めとするGUI基本ソフトウェアが開発され，普及するに至っている。

3) Ben Schneiderman, *Designing the User Interface,* Adison-Wesley Publishing, 1987.（東基衛・井関治監訳『ユーザー・インタフェースの設計』日経BP社，1987年。）

4) キーボード操作に習熟すると，手書きよりも文字入力が速くなるため，ワードプロセッサなどでは当分の間キーボードが主たる入力装置であり続けると考えられる。

5) MacintoshやWindowsにはアプリケーションの設計ガイドラインがあり，GUIをこのガイドラインに沿って開発することが要請されている。

6) ただし，実際にはC++をベターCとして利用しているにすぎないという指摘がなされていることも事実である。

7) 分析・設計に関する代表的な方法論としては以下の文献がある。
Peter Coad & Edward Yourdon, *Object-Oriented Analysis, Second Edition,*

Prentice-Hall, 1991.（羽生田栄一監訳『オブジェクト指向分析（OOA）』第2版，トッパン，1993年。）

James Rumbaugh, *et al., Object-Oriented Modeling and Design,* Prentice-Hall, 1991.（羽生田栄一監訳『オブジェクト指向方法論 OMT』トッパン，1992年。）

8) クラス・ライブラリを自作することも可能であるが，製品を利用することもできる。例えば，Windows 用の C++ クラス・ライブラリとして Microsoft Foundation Class や，Borland Object Windows がある。